Homework 1: Heat equation
Max. 4.0 p

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. Reports should contain answers to all questions stated and proper motivation for each, e.g. derivations. Submit code via e-mail.

Deadline: See homepage

1 Heat equation

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

$$q_t = \nabla\cdot(\nabla q) + S, \quad (x,y) \in [0,1]\times[0,1], \quad t \geq 0 \qquad (1)$$
$$q(x,y,0) = 0 \qquad (2)$$
$$\mathbf{n}\cdot\nabla q = 0 \quad (x,y) \text{ on the boundary} \qquad (3)$$

where **n** is the outward unit normal to the boundary. We consider first a smooth heat source

$$S(x,y) = \exp\left(-\frac{(x-x_S)^2+(y-y_S)^2}{w^2}\right),$$

centered at the point $\mathbf{x}_S = (1/2,1/2)$ with width $w=0.2$. Second, we consider a time-variable point heat source which may be described by a "delta-function",

$$S(x,y,t) = \delta(x-x_S, y-y_S)g(t)$$
$$g(t) = \begin{cases} 2, t < 1/4 \\ 0, t \geq 1/4 \end{cases}$$

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. In the first case it is continuously heated with a diffuse source in the middle. In the second case it is heated at a point for 0.25 seconds and then the source is turned off.

1.1 Analytical preamble (0.5 p)

1. What is the flux vector for the heat equation (1)?
2. Determine $Q(t) = \int_0^1\int_0^1 q(x,y,t)\,dxdy$ as a function of t.

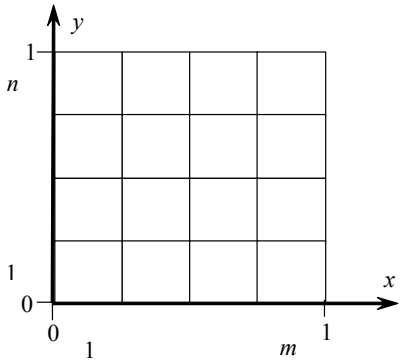

Figure 1: Finite volume grid

2 Discretization and implementation (1.0 p)

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 $m\Delta x = n\Delta y = 1$. The square is covered by mn rectangular cells.

1. Derive a 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

$$\frac{d}{dt}Q_{ij} = \Delta_5 Q_{ij} + S_{ij}, i = 1,...,m, j = 1,...,n$$

where Δ_5 is the classical 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 **Q** be an m x n array that contains the Q_{ij} values. The Laplacian can be expressed as

$$\Delta_5 \mathbf{Q} = \mathbf{Q}\mathbf{T}_x + \mathbf{T}_y\mathbf{Q}$$

where **T** represents the second derivative 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, its coefficient 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"*). One option is to use Kronecker products (`en.wikipedia.org/wiki/Kronecker product`) to construct this difference matrix. The corresponding function is called **`kron`** in Matlab. It is highly recommended to use the function **`reshape`** for rearranging *m* x *n* arrays into *mn* x 1 column vectors and vice versa.

You are strongly encouraged to write an efficient program which can handle fine resolutions in reasonable time. Take care to make the code clean and readable. On an average workstation the solver should be able to handle $m = n = 800$ or more without much trouble aside from plotting such a large data set. To get this efficiency, move as much work as possible out of the time loop and obviously use **`sparse`** format. Use the Matlab **`profiler`**! Hint: Is the matrix in the linear system constant in time? The time step? Consider LU-factorization using the function **`lu`** … with proper parameters in and out, use **`help lu`**.

3 Numerical results (1.0)

In your report, please include the following computational results:

1. **Solution plots** for some time levels before and after $t = 1/4$, for both source functions S.

2. **Convergence:**
   Choose a point (x0,y0) and compile a table (or a plot) which shows that the error behaves like

   $$O(\Delta t^p) + O(h^r)$$

   where $h = \Delta x = \Delta y$. Determine p and r. Try the smooth $S(x,y)$ first. What values would one anticipate from theory? Next, try the time-variable point source $S(x,y,t)$. What p and r do you get? Why is it a bad idea to look at the error in ∞ - or L_2-norm for this case?

3. **Numerical conservation:**
   Demonstrate that the method is numerically conservative by looking at

   $$\int q dx dy = \Delta x \Delta y \sum Q_{ij}$$

   for $0 < t < 2$. Compare the computed result to the expression computed in Section 1.1.
   Note: Conservation in “eye norm" is not enough! Think about
   a) time-discretization and how your code handles the discontinuity in $g(t)$
   b) space-discretization. Where in the cell does (1/2,1/2) appear? Different for odd or even m, n.

4 Refinements (1.5 p)

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

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

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 scales each row of a matrix. How do you scale the columns?

2. Implement a solver for 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

$$q_x(0,y,t) = -1, \quad q_x(1,y,t) = -1$$
$$q(x,0,t) = \frac{1}{\pi}\sin \pi x, \quad q(x,1,t) = \frac{1}{3\pi}\sin 3\pi x + 1 \qquad (4)$$

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

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 the fully discrete problem only one of the **T** operators needs to change for the suggested new BC (4). There are different options to enforce Dirichlet BCs in a finite volume method. For the conditions (4), you may choose a grid as shown in given in Figure 1, *only shifted half a cell in the y-direction* so the cell *midpoints* (and not cell *boundaries*) are found at $y = 0$ and $y = 1$. Present convergence results as in section 3 and discuss conservation in the context of non-homogenous boundary conditions.