KTH Engineering Sciences

# Computer Exercise 3
# Parabolic equations

The purpose of this computer exercise is to study time-dependent heat transfer problems by solving parabolic PDEs with numerical methods. In the first part you will implement Euler's explicit method in one dimension, and make comparisons of explicit vs implicit methods using Matlab built-in functions. In part two you will implement the Crank-Nicolson method for a two-dimensional problem.

### Part 1: 1D problem with Explicit Euler and built-in MATLAB commands

A metallic rod is at constant temperature when the left end of it is heated up for a short time. A little later the rod will therefore get warmer also at the right end. Eventually the whole rod will cool off again. The right end of the rod is insulated during the process, so no heat leaks out.

After rescaling to dimensionless form, the following partial differential equation can be set up for the heat diffusion process through the rod:

$$\frac{\partial u}{\partial \tau} = d\frac{\partial^2 u}{\partial x^2}, \qquad \tau > 0, \qquad 0 < x < 1, \qquad d = 0.35, \tag{1}$$

with boundary conditions

$$u(0,\tau) = \begin{cases} \sin\left(\frac{\pi\tau}{a}\right), & 0 \le \tau \le a, \\ 0, & \tau > a, \end{cases} \qquad \frac{\partial u}{\partial x}(1,\tau) = 0, \qquad a = 1.2,$$

and initial condition

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

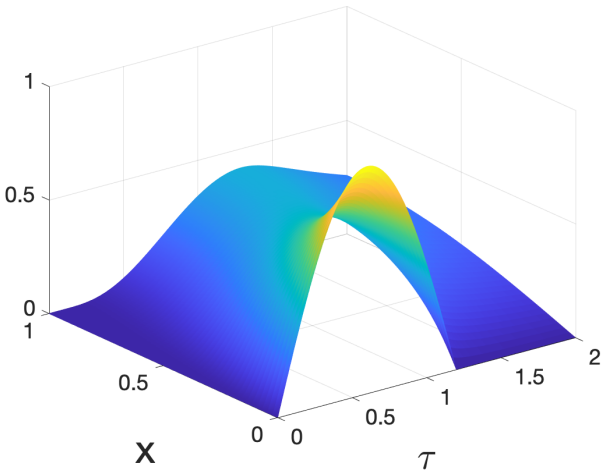
The dependent variable $u(x,\tau)$ represents the temperature at time $\tau$ and point $x$, in suitable units.

Discretize (1) with the method of lines (MoL). Use a constant step size $\Delta x$ in space and central differences to obtain an ODE-system of the form

$$\frac{d\mathbf{u}}{d\tau} = A\mathbf{u} + \mathbf{b}(\tau), \qquad \mathbf{u}(0) = \mathbf{0}. \tag{2}$$

(a) Solve the system (2) with the Explicit Euler method upto $\tau = 2$. Study the solution and the stability of the method. In particular:

- Choose $\Delta x$ and $\Delta\tau$ such that method is stable. Compute a solution and make a 3D-plot of the temperature $u$ as a function of $x$ and $\tau$. (You can for instance use `surf` or `mesh`.) Verify that the solution looks like in the picture to the right.




- State the theoretical stability condition for this problem in terms of $\Delta\tau$ and $\Delta x$. (It is the same as in the case with only Dirichlet boundary conditions.) Write the condition on the form

$$f(\Delta\tau, \Delta x) \leq c,$$

  where you specify the function $f$ and the numerical value $c$. Verify that the condition is sharp by computing two solutions where $\Delta x$ and $\Delta\tau$ are first taken just below and then just above the stability limit. Check that you indeed get a stable and then an unstable solution.
- Plot the temperature distribution in the rod at the fixed time $\tau = \tau_1$, where $\tau_1 = 1.1$, i.e. plot $u(x, \tau_1)$. Are the results as expected?
- Plot the temperature at the left and right ends of the rod as a function of time, for $0 \leq \tau \leq 2$, i.e. plot $u(0, \tau)$ and $u(1, \tau)$. Put both plots in the same figure. Are the results as expected?

*Hints:*

To make your calculations efficient construct $A$ as a sparse matrix in MATLAB and use vector variables for $\mathbf{u}$ and $\mathbf{b}$, so that $A\mathbf{u} + \mathbf{b}$ in the right hand side can be computed directly in MATLAB with matrix and vector operations, without e.g. using loops over the individual elements of $\mathbf{u}$ in every time step.

To make the requested plots it may help to store the whole approximate solution in a large matrix $U$ in which each column (or row) holds the solution at one time step; the size of $U$ will be (number of time steps)×(number of spatial grid points). Make sure to include initial and boundary conditions in $U$ so that the full solution is shown.

(b) In this part of the exercise you shall solve (2) with MATLAB's built-in ODE solvers and compare the results. The methods are `ode23` (explicit method), `ode23s` (implicit method) and `ode23s` with the option `Jacobian` set to $A$, the matrix in (2). We denote this method `ode23sJ`. (Use `odeset` to set the option, and `help odeset` to read about its significance.) The three methods shall be used under similar conditions (same problem, same tolerance) and for three spatial step sizes $\Delta x = 1/N$ corresponding to $N = 100$, 200 and 400. The comparison shall comprise the number of time steps needed to reach $\tau = 2$, and the actual time needed for each computation (measured with MATLAB's `tic/toc` commands). Collect your statistics in a table of the type:

| $N$ | # time steps | | | computational time | | |
|---|---|---|---|---|---|---|
| | `ode23` | `ode23s` | `ode23sJ` | `ode23` | `ode23s` | `ode23sJ` |
| 100 | | | | | | |
| 200 | | | | | | |
| 400 | | | | | | |

Think about:

- Why do the three methods use so different number of time steps?
- How does the number of time steps for `ode23` grow (approximately) when $N$ is doubled. Why is that?
- Why are the computational times so different for the three methods?
- Why is the `Jacobian` option in `ode23sJ` used, and why is $A$ the right value to use for it?
- Which method works best for this parabolic problem?

**Part 2: 2D problem with Crank–Nicolson**

We now consider a time-dependent version of the problem in Part 2 of Computer Exercise 2. As before, a metal block occupies the region $\Omega = [0 < x < L_x,\ 0 < y < L_y]$ in the $xy$-plane. Let $u(x, y, \tau)$ denote the temperature in the point $(x, y)$ at time $\tau$. The block is kept at constant temperature $u = T_{\text{ext}}$ at $y = 0$ and insulated at the other three sides. It has been unheated for a long time when, at $\tau = 0$, the external source modeled by the function $f$ is switched on and starts to heat it. This means that $u$ satsifies the parabolic equation,

$$\frac{\partial u}{\partial \tau} = \Delta u + f, \qquad f(x, y) = 100 \exp\left(-\frac{1}{2}(x-4)^2 - 4(y-1)^2\right), \tag{3}$$

for $(x, y) \in \Omega$ and $\tau > 0$. The boundary conditions are

$$u(x, 0, \tau) = T_{\text{ext}}, \quad \frac{\partial u}{\partial y}(x, L_y, \tau) = 0, \quad (\text{for } 0 < x < L_x),$$
$$\frac{\partial u}{\partial x}(0, y, \tau) = \frac{\partial u}{\partial x}(L_x, y, \tau) = 0, \quad (\text{for } 0 < y < L_y),$$

and initial data is given by $u(x, y, 0) = T_{\text{ext}}$. As in Computer Exercise 2, use $L_x = 12$, $L_y = 5$ and $T_{\text{ext}} = 25$.

Discretize (3) with the Crank–Nicolson method. You should be able to reuse much of the code from Computer Exercise 2, but be careful with the sign of the matrix approximating $\Delta$. It may be different from the sign you used before, depending on your implementation.

Also note that if you implemented boundary conditions as separate equations in Computer Exercise 2, you must change it. Here you need to eliminate the unknowns corresponding to boundary/ghost points after you have discretized the boundary conditions.

(a) Solve the PDE (3) to $\tau = 40$ using the Crank-Nicolson method. Choose your spatial and time steps in a suitable way. Start with a rather low resolution, i.e. large $\Delta\tau$ and $\Delta x$. High resolution solutions can take quite long time, even when sparse format is used for all matrices. (Sparse format is necessary here, or the code will be very slow.)

(b) Plot the solutions at the fixed times $\tau = 0$ (initial data), $\tau = 1$, $\tau = 4$, $\tau = 12$, $\tau = 22$, and $\tau = 40$. I.e. make six plots of $u(x, y, 0)$, $u(x, y, 1)$, etc.

(c) Plot the solution in the point $(x, y) = (6, 2)$ as a function of time for $0 \leq \tau \leq 40$, i.e. plot $u(6, 2, \tau)$. In the plot, also mark the temperature value you obtained in $(6, 2)$ when you did (2d) in Computer Exercise 2. Verify that $u(6, 2, 40) \approx 47.0$, which should be slightly lower than the value from Computer Exercise 2.

*Note:* Here you do not need to save the whole solution in a big matrix as in Part 1, just the value at $(6, 2)$.

Think about:

- Why is Crank–Nicolson a good method to use for this problem?
- What is the relationship between the solution computed here and the solution computed in Computer Exercise 2?