KTH Engineering Sciences

Computer Exercise 3
Partial differential equation of parabolic type

A metallic rod of length L [m] is initially of temperature $T = 0$ [C]. At time $t = 0$ a heat pulse of temperature $T = T_0$ and duration t_P [s] hits the left end (at $x = 0$) of the rod. At the right end (at $x = L$) the rod is isolated. After some time the rod will therefore be warmer in the right end and then cool off again. The following partial differential equation can be set up for the heat diffusion process through the rod:

$$\rho C_p \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}, \quad t > 0, \quad 0 < x < L.$$

The boundary conditions are

$$T(0,t) = \begin{cases} T_0, & 0 \le t \le t_P, \\ 0, & t > t_P, \end{cases} \qquad \frac{\partial T}{\partial x}(L,t) = 0,$$

and the initial condition is

$$T(x,0) = \begin{cases} T_0, & x = 0, \\ 0, & 0 < x \le L. \end{cases}$$

In the PDE, ρ is the density [kg/m^3], C_P the heat capacity [J/kg·C] and k is the thermal conductivity [J/m·s·C] of the rod.

The purpose of this computer exercise is to:

- Scale the problem to dimensionless form,
- Discretize the scaled problem with the Method of Lines (MoL),
- Investigate stability properties of Euler's explicit method,
- Compare an explicit and an implicit adaptive method,
- Show how an implicit method can be made more efficient,
- Visualize the result in two- and three-dimensional plots.

Part 1: Rescaling to dimensionless form

Show that with the new variables u, ξ and τ defined by

$$T = T_0 u, \qquad x = L\xi, \qquad t = t_P\tau,$$

the problem can be transformed (scaled) into the following dimensionless form

$$\frac{\partial u}{\partial \tau} = a\frac{\partial^2 u}{\partial \xi^2}, \qquad \tau > 0, \qquad 0 < \xi < 1,$$

with boundary conditions

$$u(0,\tau) = \begin{cases} 1, & 0 \le \tau \le 1, \\ 0, & \tau > 1, \end{cases} \qquad \frac{\partial u}{\partial \xi}(1,\tau) = 0,$$

and initial condition

$$u(\xi,0) = \begin{cases} 1, & \xi = 0, \\ 0, & 0 < \xi \le 1. \end{cases}$$

Give an expression for the only remaining parameter a (in terms of the original parameters) and show that it is dimensionless. From now on assume that a has the numerical value $a = 1$.

Part 2: Discretization

Discretize in space (the ξ-variable) using the constant stepsize h and central differences to obtain an ODE-system

$$\frac{d\mathbf{u}}{d\tau} = A\mathbf{u} + \mathbf{b}(\tau), \qquad \mathbf{u}(0) = \mathbf{u}_0,$$

where $\mathbf{u}_0$ is the zero vector. Show how the discretization is done, including difference approximations of the differential equation and the boundary conditions. Specify the elements of A, $\mathbf{b}(\tau)$ and $\mathbf{u}_0$ as well as their dimensions (sizes). Also show the discretized grid of the ξ-axis and how the gridpoints are numbered.

Part 3: Solution and visualization

Numerical part: Discretize the ODE-system in Part 2 with Euler's explicit method. Use constant time step Δt. To make your calculations efficient you should write your code so that the vector $A\mathbf{u} + \mathbf{b}$ is formed directly, not from component form. Store the whole approximate solution (including initial and boundary conditions) in a large matrix U, as

$$U = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 0 \\ 0 & \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \times & \cdots & \times \\ \cdot & \cdot & \cdot & \cdot & \cdots & \times \\ \cdot & \cdot & \cdot & \cdot & \cdots & \times \\ \cdot & \cdot & \cdot & \cdot & \cdots & \times \\ 0 & \times & \times & \times & \cdots & \times \end{pmatrix}$$

Graphical part: Use e.g. `surf` to draw a 3D-plot of the solution u as a function of τ and ξ. Experiment with different values of the discretization step h and Δt and study stability. Submit one graph showing a stable solution and one graph with an unstable solution. Present the values of h, Δt and $\Delta t/h^2$ in the two cases. Also make a graph that shows 2-dimensional plots of $u(\tau,\xi)$ for $0 \le \xi \le 1$ at four fixed timepoints $\tau \approx 0.5$, 1, 1.5, 2.

Part 4: Built-in Matlab solvers

In this part of the exercise you shall compare two built-in Matlab solvers: the explicit method `ode23` and the implicit method `ode23s`, suitable for stiff problems. The two functions shall be run under similar conditions (same problem, same tolerance) and for three sizes of the stepsize h corresponding to $N = 10, 20, 40$ grid-points on the ξ-axis. The comparison shall comprise

- Number of time steps needed to reach $\tau = 2$,
- CPU-time needed for each computation (e.g. with Matlab's `tic/toc` or `cputime`),
- Maximal stepsize that each method could take.

Collect your statistics in a table of the type:

N	timesteps		CPU-time		$\Delta t_{\max}$	
	`ode23`	`ode23s`	`ode23`	`ode23s`	`ode23`	`ode23s`
10						
20						
40						

Comment on the statistics. Can you explain the observed results for the two methods?

Part 5: Optimized Matlab solvers

It was seen in Part 4 that the number of time-steps is considerably smaller when using a stiff method. However the default implementation is not efficient for stiff problems coming from parabolic PDEs. The main reason is that all the linear systems of equations $Ax = b$ that are to be solved in each time step need a lot of number crunching since they are based on using the backslash-operator; Gaussian elimination is performed on a *full* matrix each time $Ax = b$ is solved. However, the system matrix of these equations is in fact tridiagonal, but this is not taken advantage off in the default implementation of `ode23s`. With the Matlab function `odeset`, options can be set by the user in order to make the computation more efficient. Do `help odeset` and study the options for `'Jacobian'` and `'Jpattern'`. With the help of these options, the computational cost can be reduced considerably. Experiment with both of them and collect statistics regarding the number of floating point operations as was done in Part 4.

Draw conclusions. What type of ODE-method should be used for parabolic problems: stiff (implicit) or non-stiff (explicit)? Why? What is the structure of the jacobian of the ODE-system? (Sparse? Banded?) Why is this important? How can it be exploited?