KTH Engineering Sciences

Computer Exercise 1
Ordinary Differential Equations and Elliptic Equations

Ordinary Differential Equations

This part of the computer exercise concerns the numerical solution of the initial value problem for ordinary differential equations (ODEs). The accuracy, stability and computational cost of a few numerical schemes are studied, in particular related to the topics of adaptive (variable) stepsize and stiff problems.

Part 1: Accuracy and stability of a Runge-Kutta method

In this part you will make some numerical experiments with the following Runge-Kutta method:

$$\begin{aligned}
k_1 &= f(t_n, u_n),\\
k_2 &= f(t_n + h, u_n + hk_1),\\
k_3 &= f(t_n + h/2, u_n + hk_1/4 + hk_2/4),\\
u_{n+1} &= u_n + \frac{h}{6}(k_1 + k_2 + 4k_3), \quad t_n = nh, \quad n = 0, 1, 2, \ldots
\end{aligned}$$

The goal is to apply the method to a linearized Landau–Lifshitz equation for the magnetization vector $\boldsymbol{m}(t) \in \mathbb{R}^3$, given as the ODE system

$$\frac{d\boldsymbol{m}}{dt} = \boldsymbol{a} \times \boldsymbol{m} + \alpha\, \boldsymbol{a} \times (\boldsymbol{a} \times \boldsymbol{m}), \qquad \boldsymbol{m}(0) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Here $\alpha = 0.07$ is a damping parameter and $\boldsymbol{a} = \frac{1}{4}[1, \sqrt{11}, 2]^T$ is a fixed vector.

(a) Implement the Runge–Kutta method and solve the ODE for $t \in [0, T]$ with $T = 50$. Show plots of the components of $\boldsymbol{m}$ as a function t. Also plot the trajectory of $\boldsymbol{m}$ in a three-dimensional plot using Matlab's `plot3` command and `axis equal`. Add the fixed vector $\boldsymbol{a}$ to the plot (using `hold on`) to verify visually that the points $\boldsymbol{m}(t)$ stays in a plane perpendicular to $\boldsymbol{a}$ and that the vector $\boldsymbol{m}(t)$ becomes parallell to $\boldsymbol{a}$ as $t \to \infty$.

(b) Check the order of accuracy of the method as follows. Run the problem with constant stepsizes using $N = 50$, 100, 200, 400, 800 and 1600 steps with t in the interval $[0, T]$, i.e. $h = T/N$. Compute the approximation $\tilde{\boldsymbol{m}}_N(T)$ of $\boldsymbol{m}(T)$ for each N and plot the differences $|\tilde{\boldsymbol{m}}_N(T) - \tilde{\boldsymbol{m}}_{2N}(T)|$ between[1] these approximations as a function of h in a `loglog`-plot. Estimate the order of accuracy from the graph. (See the notes on order of accuracy in

[1]Here $|\cdot|$ is the Euclidean (2-) norm, $|\boldsymbol{m}| = \sqrt{m_1^2 + m_2^2 + m_3^2}$

Canvas for how (and why) this works. Note that by using differences you do not need the exact solution to find the order of accuracy.)

Hint: Be careful to take the correct number of steps to reach $t = T$. If you get the order of accuracy to be one, there is some mistake in your MATLAB-code!

(c) Compute the theoretical step size stability limit h_0 for the problem. To do this, first rewrite the system on the standard form

$$\frac{d\boldsymbol{m}}{dt} = A\boldsymbol{m}, \qquad A \in \mathbb{R}^{3\times 3}.$$

Then compute the eigenvalues of A (with MATLAB) and use the stability region for the Runge–Kutta method to find h_0. The stability region is defined as those z in the left half of the complex plane for which $|1 + z + z^2/2 + z^3/6| \leq 1$. (You do not need to derive this.)

Explain how you compute h_0 based on the eigenvalues and the stability region. It will typically involve solving a nonlinear equation numerically. State which one and how you solve it. (You may use MATLAB's built-in functions or you can use your own.)

(d) Verify the step size stability limit empirically by testing different step sizes h and inspecting the results. Plot one stable and one unstable solution.

Part 2: A nonlinear stiff system

The absolute stability of a numerical method for initial value problems is particularly important when we want to solve *stiff* problems. Here we study an ODE-system known as Robertson's problem modeling the reactions of three chemicals A, B and C,

$$A \to B \qquad\qquad B + C \to A + C \qquad\qquad 2B \to B + C$$

For the three reactions above, let r_1, r_2 and r_3 denote their *rate constants*, i.e. how fast the reactions are. Furthermore, let x_A, x_B and x_C be the (scaled) concentrations of A, B and C respectively. The following system of ODEs then describe the evolution of the concentrations as a function of time t:

$$\begin{aligned}
\frac{dx_A}{dt} &= -r_1 x_A + r_2 x_B x_C, \\
\frac{dx_B}{dt} &= r_1 x_A - r_2 x_B x_C - r_3 x_B^2, \\
\frac{dx_C}{dt} &= r_3 x_B^2.
\end{aligned}$$

The rate constants have the following values: $r_1 = 5.0 \cdot 10^{-2}$, $r_2 = 1.2 \cdot 10^4$ and $r_3 = 4.0 \cdot 10^7$. We are interested in how the concentrations change over the long time interval $t \in [0, 1000]$ when we start from a state with only chemical A present,

$$x_A(0) = 1, \qquad x_B(0) = 0, \qquad x_C(0) = 0.$$

The large disparity between the rate constants introduces time scales in the system of very different size. The fastest changes happen over the order of $O(10^{-3})$ time units and the slowest over the order of $O(10^3)$.

(a) Start by solving the problem with the explicit Runge–Kutta (RK) method given in Part 1. Since the problem is stiff the stepsize h has to be very small to avoid numerical instability. Therefore, start by solving it for the shorter time $t \in [0, 10]$ and find (empirically) how small h you need to take for the solution to be stable. Report this value. Plot the solutions of x_A, x_B and x_C as a function of $t \in [0, 10]$, both with `plot` (linear scale) and `semilogy` (log scale).

(b) The stability properties of the numerical method are given by the eigenvalues of the Jacobi matrix for the right hand side. Compute those eigenvalues for your numerical solution in every timestep and plot them as a function of $t \in [0, 10]$. There are three eigenvalues. One is always zero. It is enough to plot the other two.

The largest eigenvalue (in magnitude) should determine the stepsize limit for the RK method. Verify that this is (roughly) true[2] by comparing with your empirical finding above. Note that the stability limit depends on the solution itself since the problem is nonlinear. It will therefore change in time. Does it get more or less severe as t increases?

(c) Compute the solution for $t \in [0, 1000]$ with the RK method. Keep in mind that the stability limit will be slightly different for this case than for the case $t \in [0, 10]$. Report which h you used, the final values of x_A, x_B, x_C at $t = 1000$, and how long time it took on your computer. (You can use Matlab's `tic` and `toc` commands to time it.)

Hint: To make the computations faster, do not save the solution (for plotting) or compute eigenvalues. To figure out the $T = 1000$ case it may be good to first do some simpler cases like $T = 20$, $T = 50$, $T = 100$, ...

(d) Use the Implicit Euler (IE) method to solve the problem. It is implemented in the MATLAB function file `impeuler.m` available on Canvas. Learn how to use it with `help impeuler`. Also look through the code and make sure you understand it.

Verify that IE does not have a stability limit by checking empirically that "any" timestep h gives a stable solution. Choose h so that you get, visually, the same solution for $t \in [0, 10]$ as with the RK method. (You can use a rather large h.) Then run for $t \in [0, 1000]$ with the same h and plot the solution.

(e) Compare the efficiency of the two methods by measuring how long time (using `tic/toc`) it takes to compute a solution to $t = 1000$ with different accuracies. Make a table of the form:

method	h	error	computational time
RK			
IE			
IE			
$\vdots$			

Include at least four rows for IE with h of different magnitude.

[2] The transition between unstable and stable solutions is not as clear-cut here as in Part 1.

Compute the errors (in Euclidan norm as before) at $t = 1000$ by using the following accurate values:

$$\begin{aligned} x_A(1000) &\approx 0.293414227164, \\ x_B(1000) &\approx 0.000001716342048, \\ x_C(1000) &\approx 0.706584056494. \end{aligned}$$

Also, modify the code in `impeuler.m` here so that it does not save the solution in every step, to get a fair timing comparison.

Comment on and discuss the results. In particular, answer the questions below:

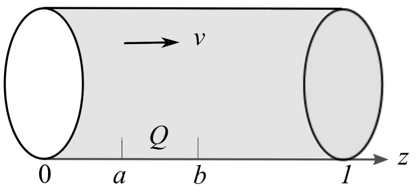
- What are the factors that influence the timings and the errors of the methods?
- Why is RK faster than IE when very accurate solutions are sought, but not otherwise?
- For which desired error levels (roughly) is IE faster than RK?
- How would the comparison change if the same experiments were made for the problem in Part 1 above?

Elliptic Equations

In this part of the exercise you will solve elliptic PDEs in one and two dimensions. You will use the finite difference method on a simple geometry (an interval and a rectangle).

Part 3: Finite difference approximation in 1D

Consider a short cylinder with a small cross section. In the cylinder there is a slowly moving fluid with high thermal diffusivity. The fluid is heated in a small section and the heat convects with the fluid along the cylinder. In suitable units for z and the fluid velocity v, the steady state temperature distribution $T(z)$ is then determined by the convection–diffusion equation



$$-\frac{d^2T}{dz^2} + v\frac{dT}{dz} = Q(z), \qquad 0 < z < 1.$$

The driving function $Q(z)$, modeling the heat source, is defined as

$$Q(z) = \begin{cases} 0, & 0 \le z < a, \\ Q_0 \sin\left(\frac{(z-a)\pi}{b-a}\right), & a \le z \le b, \\ 0, & b < z \le 1. \end{cases}$$

At $z = 0$ the fluid has the inlet temperature T_0,

$$T(0) = T_0.$$

Beyond $z = 1$ the pipe is poorly insulated and the liquid is cooled down since heat is leaking out to the exterior, which has the temperature T_{out}. This assumption can be modeled by the following boundary condition at $z = 1$:

$$-\frac{dT(1)}{dz} = \alpha(v)(T(1) - T_{\text{out}}), \qquad \alpha(v) = \sqrt{\frac{v^2}{4} + \alpha_0^2} - \frac{v}{2},$$

where α_0 is the heat transfer coefficient for the non-convective ($v = 0$) case. Use the following values of the parameters: $a = 0.1$, $b = 0.4$, $Q_0 = 7000$, $\alpha_0 = 50$, $T_{\text{out}} = 25$ and $T_0 = 100$.

(a) Consider first the case $v = 1$. Solve the boundary value problem with the finite difference method using MATLAB. Discretize the z-interval $[0, 1]$ with grid points $z_j = jh$ and $h = 1/N$. Hence, $z_0 = 0$ and $z_N = 1$, so that $N - 1$ is the number of inner grid points in the interval, which may or may not be the same as the number of unknowns in the finite difference method, depending on how you implement the boundary conditions. Use at least second order accurate approximations for the differential equation and the BCs.

- Plot the solutions $T(z)$ you get with $N = 10$, 20, 40 and 80 in the same graph.
- Report the temperature values computed at $z = 0.5$ for $N = 80$, 160, 320 (three T-values).

(b) Now solve the problem for $v = 1,\ 5,\ 15,\ 100$, and plot the solutions in the same graph. Interpret the curves based on the physical problem they model. Why do they look like they do? Are the results as expected? Note: For higher v you will need to take a smaller h (larger N) to get the correct behavior of the solution close to $z = 1$. Report which h you take for the different v.

OBS! For full credit your implementation should exhibit second order accuracy when the grid is refined as in (a). You may need to take even larger N to verify that it is second order.

Hints: If you have used a second order stencil and second order approximation of the boundary conditions but still only observe a first order convergence rate it is quite likely that you have a made a small mistake somewhere. (This is very easy to make!) Here are two common minor mistakes that will turn a second order method into a first order method:

- Your implementation of the number of unknowns is not compatible with your N and step length h. You may have used N unknowns when your implementation of the boundary conditions is based on using $N-1$ unknowns, for instance.

- You may evaluate the source function Q in slightly the wrong points. You might use $Q(z_j - h)$ instead of $Q(z_j)$ or something similar.

Additionally, even if you have a second order method, you may observe first order convergence if you do not read off the T value in exactly the same point for the different grids. Instead of $z = 0.5$ you might be looking at $z = 0.5 - h/2$ for instance.

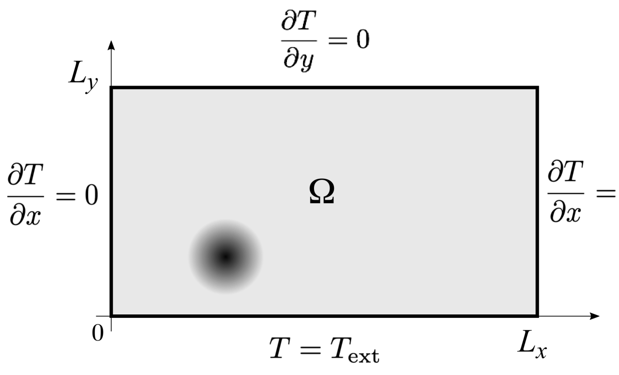
Part 4: Finite difference approximation in 2D

In this part we consider heat conducted through a 2D rectangular metal block occupying the region $\Omega = [0 < x < L_x,\ 0 < y < L_y]$ in the xy-plane. At $y = 0$ the block is kept at the same temperature as the surrounding air $T = T_{\text{ext}}$. It is insulated at the other three sides. An external source modeled by the function $f(x, y)$ heats the block. The following elliptic problem for the temperature distribution $T(x, y)$ can then be formulated:

$$-\Delta T = f, \quad (x, y) \in \Omega,$$

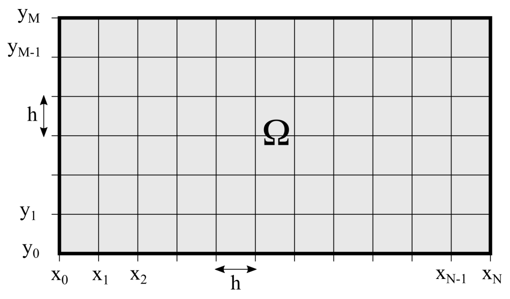
with boundary conditions

$$\begin{aligned}
T(x,0) &= T_{\text{ext}}, & 0 < x < L_x,\\
\frac{\partial T}{\partial x}(0,y) &= 0, & 0 < y < L_y,\\
\frac{\partial T}{\partial x}(L_x,y) &= 0, & 0 < y < L_y,\\
\frac{\partial T}{\partial y}(x,L_y) &= 0, & 0 < x < L_x.
\end{aligned}$$



$\frac{\partial T}{\partial y} = 0$
L_y
$\frac{\partial T}{\partial x} = 0$
Ω
$\frac{\partial T}{\partial x} = 0$
0
$T = T_{\text{ext}}$
L_x

Use the finite difference method to solve this problem with the parameter values $L_x = 12$, $L_y = 5$ and $T_{\text{ext}} = 25$. The approximation should be of at least order two. Discretize the rectangular domain into a quadratic mesh with the same, uniform, stepsize $h = L_x/N = L_y/M$ in the x- and y-directions[1]. Similar to Part 1, $M-1$ and $N-1$ are the number of inner grid points in each direction, which may or may not be the same as the number of unknowns in the finite difference method, depending on how you implement the boundary conditions. Solve the resulting linear system with backslash in MATLAB.



(a) Compute the solution $T(x, y)$ with $f \equiv 2$ and $h = 0.2$ ($N = 60$). (Be careful with the sign here! Do not miss the minus sign in front of Δ.) Visualize $T(x, y)$ using the MATLAB function `mesh`. What is the computed T-value in the point $(x, y) = (6, 2)$ inside the block?

(b) When f is constant, the exact solution to the PDE is a second order polynomial in y, i.e. of the form $T(x, y) = c_0 + c_1 y + c_2 y^2$. Use the PDE and the boundary conditions to find the coefficients c_j for the solution in (a). (Note that the Neumann BC on the left and right sides of the block are satisfied for all choices of coefficients.)

(c) For the case in (a) the numerical method actually gives the *exact* solution of the PDE. First compare the solutions from (a) and (b) in the point $(x, y) = (6, 2)$ to verify this. Then find an expression for the error term $R(x)$ of the central difference approximation

$$\frac{g(x+h) - 2g(x) + g(x-h)}{h^2} = g''(x) + R(x),$$

and use it to explain why the numerical solution is exact!

(d) Solve the problem (numerically) with the following localized heat source

$$f(x, y) = 100 \exp\left(-\frac{1}{2}(x-4)^2 - 4(y-1)^2\right).$$

Report the the three T-values obtained at $(x, y) = (6, 2)$ when you use $h = 0.2$ ($N = 60$), $h = 0.1$ ($N = 120$) and $h = 0.05$ ($N = 240$). Here it is important to use MATLAB's `sparse` format for the matrices. Otherwise the problems with small h take very long time. Visualize the solution as above for $h = 0.1$. This time also plot it with the `imagesc` and `contour` commands. (You do not need to find the analytic solution for this case!)

OBS! For full credit your implementation should exhibit second order accuracy when the grid is refined. (You may want to check values at $(6, 2)$ for additional h to verify your convergence rate.)

Hint: If you have problem getting second order accuracy, please look at the hints in Part 1 for the 1D case above. In 2D it is even easier to make small mistakes of the types mentioned there. Be particularly careful to read off the value in the right point, $(x, y) = (6, 2)$ and not something like $(x, y) = (6 + h/2,\ 2 - h/2)$. Also make sure $M = 5N/12$ is an integer, so that the step size is indeed the same in the x- and y-directions.

[1]This means that $M = \frac{L_y}{L_x} N = 5N/12$ must be an integer.