Assignments V

Exercise 1 *Write a* MATLAB*-function which solves the initial value problem for a system of differential equations via the θ-method.*

The θ-method with step size h for the differential equation $u' = f(t, u)$ is given by

$$u_{n+1} = u_n + h\big[(1-\theta)f(t_n, u_n) + \theta f(t_{n+1}, u_{n+1})\big].$$

Here $\theta \in [0, 1]$ is a parameter; for $\theta = 0$ it is the forward Euler method, for $\theta = 1$ the backward Euler method, and for $\theta = \frac{1}{2}$ the Crank-Nicolson method, but θ can take other values as well. The function $f : \mathbb{R} \times \mathbb{R}^k \to \mathbb{R}^k$, with $k \in \mathbb{N}$ defines the system (if $k > 1$) of differential equations.

The input parameters are, in this order, a function-handle which describes the right-hand side $f(t, u)$ of the differential equation, a vector S with requested points in time, a column vector with initial conditions u_0, the parameter θ, and the step size h.

The output is, in this order, a column vector of times T, and a matrix U with in every row the solution at the time corresponding to that row in T.

1. The purpose of this exercise is to get insight (by doing it) into the modus operandi of an ode-solver (without the complications of variable step sizes).
2. Obviously, in this exercise, you can *not* use a built-in ode-solver of MATLAB.
3. You may assume that the function f accepts as input a number t and a column vector u of length k, and returns a column vector of length k.
4. The initial values u_0 correspond with the starting time given by the first element of S.
5. When the input vector S consists of *more than two* elements, then the output is given for those points in time only. On the other hand, when the input-vector S consists of exactly two elements, then the function gives output for *all* calculated points in time (including the times in S).
6. Even though your function computes with fixed step size h, the output should be a good approximation at the requested times given in S by the user.
7. Apply your function at the very least to the simple cases $u' = -u$ and the system $u_1' = u_2$, $u_2' = -u_1$, and compare the outcomes for different values of θ, but also for a (simple) system of equations.

Exercise 2 *Write a* MATLAB*-function which shows the movement of planets.*

The movement of planets (and the sun) is governed by Newton's laws of gravitation. First, the acceleration a is equal to the force F divided by the mass m (i.e. $F = ma$). Second, the gravitational force which two masses m_1 and m_2 exert on each other is given by

$$F_z = \frac{m_1 m_2 G}{r^2},$$

where r is the distance between the masses and $G = 6.672 \cdot 10^{-11}\mathrm{Nm^2kg^{-2}}$ is the gravitational constant. The direction of the force is of course along the line connecting the masses.

The input parameters are a vector m with the masses of the planets (say of length p), a matrix of initial positions ($n \times p$), a matrix of initial velocities ($n \times p$), and the length of time T over

which the movements should be computed. The function makes a picture of the orbits and returns as output a vector of points in time and a matrix of corresponding positions and velocities of the planets.

1. You can now use the built-in ode-solvers in MATLAB.
2. The function should work for any number of planets p and for movements in both 2 and 3 dimensions ($n = 2, 3$).
3. Take care to choose all units of physical quantities consistently.
4. Make several pictures, including ones of
   - the earth circling around the sun in 2 dimensions; this is also an excellent *test-case* for your program.
   - at least three planets around a sun in 3 dimensions;
   - a sun-earth-moon system;
   - a bounded, regular movement (i.e. planets do not "escape" to infinity for large T) of two equal masses in 2 dimensions. Attempt a similar thing for three equal masses (none of which is stationary).

Exercise 3 *Write a* MATLAB*-function which solves the Korteweg-de Vries equation.*

The Korteweg-de Vries equation describes moderately high waves in a shallow channel, and after rescaling is given by

$$\frac{\partial u}{\partial t} = -\frac{\partial^3 u}{\partial x^3} - 6u\frac{\partial u}{\partial x} = -\frac{\partial}{\partial x}\left(\frac{\partial^2 u}{\partial x^2} + 3u^2\right).$$

Here u is the height of the wave above the mean water level. We consider this equation on an interval $x \in [-L, L]$ with periodic boundary conditions (i.e. $u(t, x + L) = u(t, x - L)$). The equation is nonlinear; the term $6u\frac{\partial u}{\partial x}$ brings about the (almost) breaking of the waves.

The input parameters are the (almost) periodic initial conditions u_0 (a function handle), the length of the interval L, the length of time T, the number of time steps S, and optionally the number of spatial discretization points N.

The output are a vector of times $\{i\frac{T}{S}\}_{i=0}^{S}$, a vector of discretization points, and a matrix containing the profile of the solution at those points in time and space (and such that it is easy to plot). Moreover, the function produces an animation of the solution.

1. Discretize the spatial derivatives (using centered differences), and decide with to do with the boundary conditions.
2. Make pictures for several initial conditions, including what happens when
   - $$u_0(x) = f(x; \lambda) \stackrel{\text{def}}{=} \frac{\lambda}{2}\frac{1}{\cosh^2(\sqrt{\lambda}x/2)},$$

     for several (large) values of λ. If all is well, these are waves which propagate at constant velocity.
   - $u_0(x) = f(x - \frac{L}{2}; \lambda_1) + f(x + \frac{L}{2}; \lambda_2)$ for different values of λ_1 and λ_2 (with $\lambda_i \gg 1/L^2$). This shows the interaction of waves with different velocities.
   - $u_0(x) = \mu \sin(\pi x/L)$ for fairly large L and several values of the amplitude μ. This shows, among others, the breaking of the waves and, for large time, special recurrent behaviour.
3. The animation can be produced by repeated plotting (`drawnow`). It is useful to add an optional input parameter which controls the speed of the animation (e.g. using `pause`).