Numerical Mathematics I, 2017/2018, Lab session 6

Deadline for discussion: Monday 04/06/18 at 5 pm

Keywords: initial value problem, time integration

Remarks

- Make a new folder called `NM1_LAB_6` for this lab session, save all your functions in this folder.
- Whenever a new `MATLAB` function is introduced, try figuring out yourself what this function does by typing `help <function>` in the command window.
- Make sure that you have done the preparation before starting the lab session. The answers should be worked out either by pen and paper (readable) or with any text processing software (LaTeX, Word, etc.).

1 Preparation

In this lab session we will consider solving the Cauchy problem (or initial value problem)

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}(t)), \quad \mathbf{u}(t_0) = \mathbf{u}_0, \quad t \in [t_0, T], \tag{1}$$

where $\mathbf{f} : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$.

1.1 The θ-method

1. Study Chapter 8 and in particular (Textbook, Section 8.4 - 8.7 & 8.9).
2. Consider the the first step in solving the Cauchy problem (for $N = 1$)

   $$u_1 = y_0 + \frac{h}{2}(f(t_0, y_0) + f(t_1, u_1)),$$

   where the trapezoidal formula is used. Show that this approximation leads to

   $$|u_1 - y_1| \leq \frac{h\tau(h)}{1 - hL/2},$$

   for $hL < 2$, where L is the Lipschitz constant of f (w.r.t. y) and $\tau(h)$ is the global truncation error given by

   $$\max_{t \in [t_0, T]} \left| \frac{y'''(t)h^2}{12} \right|.$$

3. Consider the so called θ-method given by

   $$\mathbf{u}_{n+1} = \mathbf{u}_n + h(\theta \mathbf{f}_{n+1} + (1 - \theta)\mathbf{f}_n). \tag{2}$$

   Show that for $\theta = 0$ this method is explicit and for $\theta \neq 0$ it is implicit. Hence for $\theta \neq 0$ each step requires the solution of a (possibly nonlinear) system of equations.

4. Show that (2) can be written as the rootfinding problem $\mathbf{F}(\mathbf{u}) = \mathbf{0}$ where $\mathbf{F}$ is given by

$$\mathbf{F}(\mathbf{u}) = \mathbf{u} - \mathbf{u}_n - h(\theta \mathbf{f}(t_{n+1}, \mathbf{u}) + (1-\theta)\mathbf{f}(t_n, \mathbf{u}_n)). \tag{3}$$

   Show that the Jacobian of $\mathbf{F}$ is given by

$$\mathbf{J}(\mathbf{u}) = \mathbf{I} - \theta h \hat{\mathbf{J}}(t_{n+1}, \mathbf{u}), \tag{4}$$

   where $\hat{\mathbf{J}}(\mathbf{u})$ is the Jacobian of $\mathbf{f}$ and $\mathbf{I}$ is the $N \times N$ identity matrix.

5. Show that the θ-method satisfies the *root condition* (Textbook, Equation 8.25), and is therefore zero-stable. Moreover show that the θ-method is consistent. (Here it is sufficient to verify that (Textbook, Equation 8.27) holds.) Conclude that the θ-method is *convergent.*

6. Consider the test problem $y' = \lambda y$

   (a) Show that application of the θ-method gives

$$u_{n+1} = \frac{1 + (1-\theta)h\lambda}{1 - \theta h\lambda} u_n.$$

   (b) Show that the θ-method is *unconditionally absolutely stable* (A-stable) for $\theta \geq \frac{1}{2}$.

7. Consider the Cauchy problem given by

$$y'(t) = \lambda(y(t) - \sin(t)) + \cos(t), \quad y(0) = 1, \quad t \in [0, 10], \tag{5}$$

   for some constant λ.

   (a) Confirm that the solution is given by

$$y(t) = e^{\lambda t} + \sin(t).$$

   (b) Show that $f(t, y) = \lambda(y(t) - \sin(t)) + \cos(t)$ is Lipschitz continuous w.r.t. the second argument, hence showing that the Cauchy problem is well-posed.

1.2 The n-body problem

1. Study (Textbook, Section 8.10.2).

2. Consider the two body problem, where the center body is static. The corresponding adimensionalised (the resulting time scale is given by one year) system of ODEs is given by

$$\begin{aligned} \frac{d}{dt}\mathbf{u} = \mathbf{f}_1 &= -\frac{4\pi^2}{\|\mathbf{x}\|_2^3}\mathbf{x} \\ \frac{d}{dt}\mathbf{x} = \mathbf{f}_2 &= \mathbf{u}, \end{aligned} \tag{6}$$

   where $\mathbf{u} = (u_1, u_2, u_3)^T$.

   (a) Show that the Jacobian matrix $\hat{\mathbf{J}}$ of $\mathbf{f}(\mathbf{u}, \mathbf{x}) = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}$ is given by

$$\hat{\mathbf{J}} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{I}_3 & \mathbf{0} \end{pmatrix}, \text{ where } \mathbf{B} = -\frac{4\pi^2}{\|\mathbf{x}\|_2^3}\mathbf{I}_3 + \frac{12\pi^2}{\|\mathbf{x}\|_2^5}\mathbf{x}\mathbf{x}^T. \tag{7}$$

The matrix $\mathbf{x}\mathbf{x}^T$ denotes the outer product given by

$$\mathbf{x}\mathbf{x}^T = \begin{pmatrix} x_1x_1 & x_1x_2 & x_1x_3 \\ x_2x_1 & x_2x_2 & x_2x_3 \\ x_3x_1 & x_3x_2 & x_3x_3 \end{pmatrix}.$$

(b) Show that the following orbit is a solution to the adimensionalised 2-body problem

$$\mathbf{x}_{\text{exact}}(t) = R\begin{pmatrix} \cos(\omega t) \\ \sin(\omega t) \\ 0 \end{pmatrix},$$

where $\omega = \sqrt{\frac{4\pi^2}{R^3}}$.

3. How many equations (and unknowns) do we get when solving the n-body problem (hence with n non-static bodies)?

2 Lab experiments

2.1 The θ-method

Explicit method

First the θ-method for $\theta = 0$ will be implemented (keep in mind that you will extend your function in the next part such that it works for any θ, so keep θ as an input variable). Your function should integrate an ODE of the form (1). Hence the input of your function should include `f` (the right-hand side function $\mathbf{f}$), `u0` and the time interval of integration `tRange` which should be a 2×1 array containing t_0 and T. Write a `MATLAB` implementation called `odeSolveTheta.m`, the header of your function should be of the following form

```matlab
% Performs integration of the system of ODEs given by
%            d/dt u = f(t, u(t)), u(tRange(1)) = u0
% using the theta-method
% INPUT
% f          the right-hand side function, the output should be a
%            N x 1 array where N is the number of unknowns
% tRange     the time interval of integration
% u0         initial solution at t = tRange(1) (N x 1 array)
% df         a function that evaluates the jacobian of f
% theta      defines the method
% h          the step-size
% OUTPUT
% tArray     array containing the time points
% solArray   array containing the solution at each time level
%            (the ith row equals the solution at time tArray(i))
function [tArray, solArray] = odeSolveTheta(f, tRange, u0, df,...
    theta, h)
```

The input `df` is not yet needed for the explicit method.

Test your implementation by comparing the numerical solution of (5) to the exact solution given in the preparation.

Implicit method

Extend the implementation you wrote before such that it works for arbitrary values of θ. Recall from the preparation that for $\theta \neq 0$ the θ-method requires the solution of a (possibly nonlinear) system of equations. For this you should use the given implementation of Newton's method to solve (3), read the header of `newton.m`. Always use `tol = 1E-9, maxIt = 50` as input to Newton's method[*].

The input `df` (of the θ-method) should be a function handle to the Jacobian $\hat{\mathbf{J}}$ of $\mathbf{f}$. So inside `odeSolveTheta.m` you should use `df` to construct the Jacobian $\mathbf{J}$ of $\mathbf{F}$ (see (4)) which is an input to Newton's method. Make sure your function checks the value of θ to determine whether or not the method is implicit, if $\theta = 0$ the solution should be explicitly updated.

Comparison

Consider the IVP given by (5). Test the stability of the θ-method for $\theta = 0, 1/2, 1$ by considering $\lambda = -100$ and the following values of h

$$h_i = 2 \cdot 10^{-2} + (i-4) \cdot 10^{-4}, \quad i = 1, \dots, 7.$$

Determine whether your experiments are in agreement with the theoretical regions of absolute stability.

Now consider $\lambda = -1$. Determine the order of accuracy of the θ-method for $\theta = 0, 1/2, 1$, by computing the error at $t = T = 10$ for $h = 2^{-i}$, where $i = 0, \dots, 7$. Measure the time needed to compute the solution. Summarise your results of the accuracy test in one figure, plot the logarithm of the error at $t = T$ against the logarithm of the step-size h for all three methods.

2.2 The n-body problem

Two body problem

Consider the two body problem with a static center body. Make a function called `twoBodyF.m` that evaluates the right-hand side $\mathbf{f}$ as defined by (6). Also make a function `twoBodyJac.m` that evaluates the Jacobian $\hat{\mathbf{J}}$ as given by (7). The headers of your functions should be of the following form

```
% INPUT
% t          current time (not used)
% solVec     current solution (should be 6x1 array)
% OUTPUT
% dSolVec    right-hand side
function dSolVec = twoBodyF(t, solVec)
```

```
% INPUT
% t          current time (not used)
% solVec     current solution (should be 6x1 array)
% OUTPUT
% J          Jacobian matrix
function J = twoBodyJac(t, solVec)
```

[*]The test problem (5) is linear in y, hence Newton's method should converge in a single step.

You are given a MATLAB implementation of the explicit Runge-Kutta method, which is defined by the Butcher array B. Read the header of odeSolveRK.m.

Test the second-order Runge-Kutta method[†] and your implementation of the trapezoidal method (θ-method for $\theta = 1/2$). Use $R = 1$ and let the initial condition be given by

$$\mathbf{u}_0 = \mathbf{u}_{\text{exact}}(0), \quad \mathbf{x}_0 = \mathbf{x}_{\text{exact}}(0).$$

Compute the solution for $t \in [0, T] = [0, 5P]$, where P is the orbital period. Use $h = 2^{-i}P$ for $i = 3, \ldots, 12$.

Jacobian-free method

For more complicated ODEs the computation of the Jacobian of $\mathbf{f}$, denoted by $\hat{\mathbf{J}}$, can be a tedious task (e.g. if we consider the n-body problem for any n). Therefore you are given an implementation of an approximate Newton method, Broyden's method[‡], which does not require a Jacobian to be given by the user, see the header of broyden.m.

Modify your implementation of the θ-method such that Broyden's method is used to solve (3) if no Jacobian is supplied. In this case the input variable df is an empty array; this can be tested using isempty(df). Consider the trapezoidal method using Newton's method and using Broyden's method. Compare the resulting error and computation time when letting $h = 2^{-10}P$.

The solar system

You are given the MAT-file solarSimData.mat containing the adimensionalised velocities and positions of 11 "bodies"[§] of our solar system at May 30th, 2018. This file contains

- velAndPos: 6×11 array, for 11 bodies this contains (per column) the velocity and position vectors
- bodyMass: 11×1 array, the mass of each body
- bodyData: some additional information about the bodies (may be ignored at first)

For the time integration you can use the given function nBodyF.m which evaluates the right-hand side of the system of ODEs. The second input of this function should be the $N \times 1$ array containing the velocities and positions, where N is the number of unknowns. Hence the initial condition (given by the array velAndPos) should be reshaped into a 66×1 array. The reshaping can be done using MATLAB's built-in reshape function.

Use the Trapezoidal method to compute the solution after one year, using a time step of one hour (the time scale is given by one year). For animating the trajectories you can use the given simulateSolarSystem.m function.

The trajectories of the sun, earth and moon can be used to predict certain astronomical events. Predict the next two solar and lunar eclipses. During a solar eclipse the following quantity τ will be approximately equal to minus one

$$\tau(t) := \frac{\mathbf{d}_{ME}(t)^T \mathbf{d}_{MS}(t)}{\|\mathbf{d}_{ME}(t)\|_2 \|\mathbf{d}_{MS}(t)\|_2},$$

[†]Instead of a Butcher array, you may use as input B = 2, which will give Heun's method.

[‡]In Broyden's method an approximation of the Jacobian is iteratively constructed, for those interested in learning more about Broyden's method see (Textbook, Equation 2.19).

[§]They are ordered as follows: Sun, Mercury, Venus, Earth, Earth's Moon, Mars, Jupiter, Saturn, Uranus, Neptune and Pluto where $\mathbf{d}_{ME}(t) = \mathbf{x}_M(t) - \mathbf{x}_E(t)$ and $\mathbf{d}_{MS}(t) = \mathbf{x}_M(t) - \mathbf{x}_S(t)$. The vectors $\mathbf{x}_S(t), \mathbf{x}_M(t)$ and $\mathbf{x}_E(t)$ are the positions of the sun, moon and earth respectively. Use a tolerance of $|\tau(t) + 1| < 3 \cdot 10^{-4}$.

Remark: Given a value t^* (in seconds) at which a solar eclipse is supposed to occur, you can calculate the corresponding date by

```
datestr(addtodate(datenum('09:00 30-May-2018'), tStar, 'second'),...
    'HH:MM dddd dd mmmm yyyy').
```

3 Discussion

3.1 The θ-method

1. Discuss the differences of the θ-method for $\theta = 0, 1/2, 1$ (order of accuracy, stability, efficiency, implicit/explicit). What are the names of these famous methods?
2. Do the found stability limits agree to the theory? Do the theoretical orders of accuracy of the methods agree to those found in the accuracy test?
3. What would be your method of choice for the model problem you considered in the experiment? How does you choice depend on the value of λ? Motivate your answer.

3.2 The n-body problem

Two body problem

1. Consider the trapezoidal method and the second-order Runge-Kutta method (Heun's method). Discuss the differences and advantages/disadvantages of both methods (order of accuracy, stability, efficiency).
2. Discuss the differences between using Newton's method and using Broyden's method (as a nonlinear solver for the trapezoidal method).

The solar system

1. What is the geometrical meaning of $\tau(t)$?
2. When will the next three solar eclipses occur (since May 30th, 2018)? What about the next two lunar eclipses? Can you also go back in time and confirm the dates of the eclipses of last year?