Numerical Mathematics I, 2015/2016, Lab session 6

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 initial value problems of the form

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

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

1.1 The θ-method

1. Study Chapter 7 and in particular (Textbook, Section 7.4 - 7.7 & 7.9).
2. 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.

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

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

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

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

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

4. Show that the θ-method satisfies the *root condition* (Textbook, Equation 7.25). Moreover show that the θ-method is consistent. (Here it is sufficient to verify that (Textbook, Equation 7.27) holds.) Conclude that the θ-method is *convergent*.

5. Consider integration of (1) over one step-size with $N = 1$, hence

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} f(t, u(t))dt, \quad h = t_{n+1} - t_n.$$

Show that applying the trapezoidal rule to approximating the integral yields a locally $\mathcal{O}(h^3)$ accurate method. To which value of θ does this trapezoidal method correspond to?

6. Show that applying the θ-method to the test problem $\frac{du}{dt} = \lambda u(t)$, gives

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

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

8. Consider the initial value 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. Confirm that the solution is given by

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

1.2 The n-body problem

1. Study (Textbook, Section 7.10.2).

2. Consider the two body problem, where the center body is static. The corresponding adimensionalised system of ODEs is given by

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

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

Moreover you are given (try to show this) 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

$$\hat{\mathbf{J}} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{I}_3 & \mathbf{0} \end{pmatrix}, \text{ where } \mathbf{B} = -\frac{4\pi^2}{\|\mathbf{x}\|_2^3}\mathbf{I}_3 + \frac{12\pi^2}{\|\mathbf{x}\|_2^5}\mathbf{x}\mathbf{x}^T. \tag{6}$$

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

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

3. Show that the following orbit is a solution to the scaled 2-body problem

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

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

4. 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

Introduction

Compare the three methods defined by $\theta = 0, 1/2, 1$ on the problem given by (5). For $\theta = 0$ this method is explicit, hence this should be implemented first. It is advised to first verify that the explicit method works properly (test the consistency) before moving on to the more complicated case $\theta \neq 0$.

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 initial value problem 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_{end}. 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.

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 your implementation of the Newton method[*] to solve (3). The input `df` should be a function handle to the Jacobian $\hat{\mathbf{J}}$ of $\mathbf{f}$ (so inside `odeSolveTheta.m`

[*]In case you do not have a proper implementation of the Newton method, you can instead use the given function `newton_pcode.p`. The input and output of this function are equivalent to that of your own implementation.

you should use df to construct the Jacobian $\mathbf{J}$ of $\mathbf{F}$ as done in (4)). Make sure your function checks the value of θ to determine whether or not the method is implicit, if $\theta = 0$ the solution should just be explicitly updated.

Comparison

Consider the IVP given by (5). Test the stability by considering $\lambda = -100$ and the following values of h

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

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

Now consider $\lambda = -1$. Determine the order of accuracy by computing the error at $t = t_{\text{end}} = 10$ for $h = 2^{-i}$, where $i = 0, \ldots, 7$. 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 $t = t_{\text{end}}$ against the logarithm of the step-size h for all three methods.

2.2 The n-body problem

Two body problem

First we consider the simple two body problem with a static center body. Make a function called `twoBodyF.m` that evaluates the right-hand side $\mathbf{f}$. Also make a function `twoBodyJac.m` that evaluates the Jacobian $\mathbf{J}$ as given by (6). 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)
```

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`. You are also given an implementation of a family of (explicit) Adams-Bashforth methods, read the corresponding header found in `odeSolveAB.m`.

Test the fifth-order Runge-Kutta method[†] and fifth-order Adams-Bashforth method together with your implementation of the θ-method (for $\theta = 0, 1/2, 1$). 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 $t \in [0, t_{\text{end}}] = [0, 5P]$, where P is the orbital period. Use $h = 2^{-i}P$ for $i = 1, \ldots, 8$. Based on your results (study the error and efficiency) pick one of the five methods for simulating the solar system in the last part of this exercise.

[†]Instead of a Butcher array, you may use as input `B = 5`, which will give a predefined fifth-order method.

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 January 1st, 2016. 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)

The input to the ODE solvers used here should be an $N \times 1$ array, 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 one of the methods (the one you chose while testing the 2-body problem) to compute the solution after 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

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

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('00:00 1-Jan-2016'), 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 (you may give different anwers for $\lambda = -100$ and $\lambda = -1$)? Motivate your answer.

[‡]They are ordered as follows: Sun, Mercury, Venus, Earth, Earth's Moon, Mars, Jupiter, Saturn, Uranus, Neptune and Pluto

3.2 The n-body problem

Two body problem

1. Consider the fifth-order Adams-Bashforth formula (AB5) and the fifth-order Runge-Kutta method (RK5). Discuss the differences and advantages/disadvantages of both methods (order of accuracy, stability, efficiency, adaptivity).
2. How small do you expect you need to choose h such that the trapezoidal method has the same error as the fifth-order Runge-Kutta with $h^* = 2^{-8}P$?
3. Which method did you choose? Motivate your answer.

The solar system

1. What is the geometrical meaning of $\tau(t)$?
2. When will the next two solar eclipses occur (since January 1st, 2016)? What about the next two lunar eclipses? Can you also go back in time and confirm the dates of the eclipses of last year?