Numerical Mathematics Assignment 6 Peter-Jan Derks 11065540

Exercise 1

THETAMETHODS solves the initial value problem for a system of differential equations via the theta -method. [T,U] = THETAMETHODS(FUNC,S,U0,TH,H) The input parameters are, a function-handle FUNC which describes the right-hand side F(T,U) of the differential equation, a column or row vector S with requested points in time, a column vector with initial conditions U0 the parameter ?, 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.

In example of an input for the function is:

>> [T,U] = thetamethods(@(t,x) [x(2);-x(1)], [1;3.3;10], [1;0] , 0.6, 0.05)

This gives output:

T =

```
   1.0000
   3.3000
  10.0000
```

U =

```
   1.0000         0
  -0.6583   -0.7375
  -0.8702   -0.3958
```

Discription

The function makes use of multinewton to solve the function of the thetamethod as the root of the following equation needs to be found:

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

A function handle f is made with variables u_{n+1} and t. To solve multinewton the derivative of this function f needs to be known, so the function calculates it using numjac (at t=0). Then a new function ft is made from f with u_{n+1} as the only variable. Ft and df are put into multinewton, with starting point u_n.

The previously described steps are all iterated in a for loop to calculate u for a timespan requested by the user. The counter i goes from t1 to t2 in steps of h.

When the input vector S consists of more than two elements thetamethods makes use of the spline function to calculate the values of u at the requested S values by the user. The output is only given for the points in time in the S vector. If the input-vector S consists of exactly two elements, then the function gives output the for all calculated points in time (including the times in S).

Error estimate

If h is to big it is possible that newton does not converge. But decreasing h makes the function slower but also makes the spline function more precise. The built-in ode solvers change h while running which makes them more precise and quicker.

Questions (point 7 in the assignment) and Picture

The system $u'1 = u2, u'2 = -u1$

```
>> x = linspace(1,10,30)
>> [T,U] = thetamethods(@(t,x) [x(2);-x(1)], [x], [1;0] , 0.1, 0.05)
>> [T,U] = thetamethods(@(t,x) [x(2);-x(1)], [x], [1;0] , 0.3, 0.05)
>> [T,U] = thetamethods(@(t,x) [x(2);-x(1)], [x], [1;0] , 0.6, 0.05)
>> [T,U] = thetamethods(@(t,x) [x(2);-x(1)], [x], [1;0] , 0.9, 0.0
```

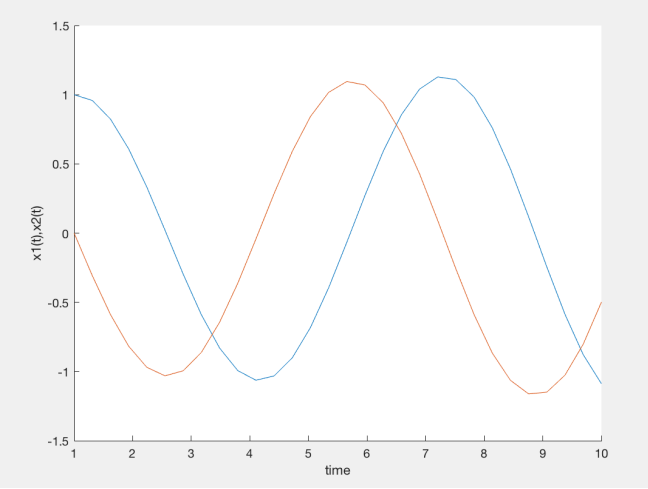


Figure 1 th = 0.1

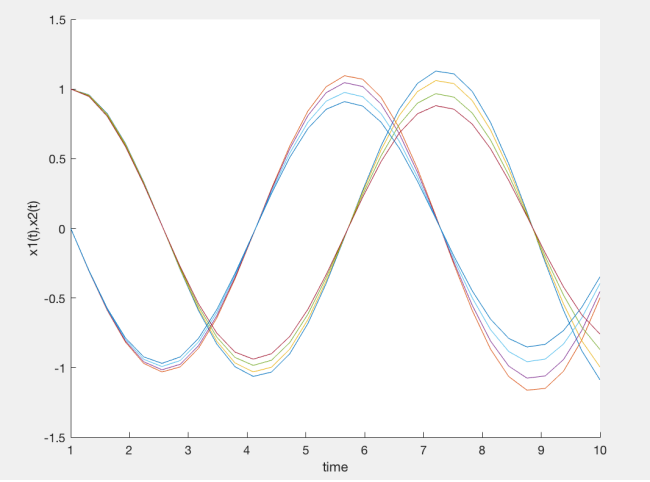
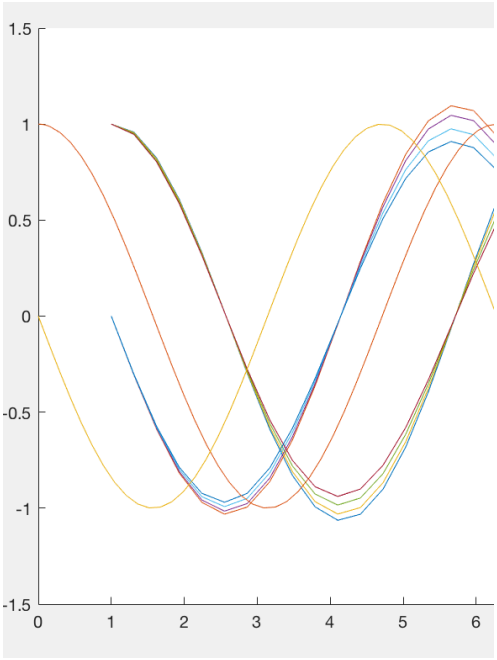


Figure 2 th = 0.1, th = 0.3, th = 0.6, th = 0.9

As can be seen in figure 1 and figure 2 different values of theta produce a different graph. The most precise of the chose values above is theta 0.1 For some reason, which I am yet to figure out the graph of the same system of equations solved using ode45 gives a different graph:



The input to draw this graph is:

```
g=@(t,x) [x(2);-x(1)]
[t,x]=ode45(g,[0 2*pi],[1;0]);
plot(t,x)
```

Exercise 2

PENDULUM shows the movement of a forced friction free pendulum
Peter-Jan Derks

[] = PENDULUM(AF,BT,N) describes the movement of a pendulum of which the pivot is periodically moved up and down by solving the differential equation: $x'' + (1 + AF(\sin(BT))\sin(x) = 0$ where AF $\geq$ 0 is the amplitude of the forcing and BT > 0 the frequency. The input parameters are AF, BT and the number of periods N of the forcing. There is no output, but the function produces a picture of the time-τ map by choosing a number of initial points and plotting the iterations of those points under the map Φ_τ.

Description

The second order differential equation:

$$x'' + (1 + \alpha(\sin(\beta t))\sin(x) = 0$$

Can be rewritten as two first order differential equations:

$$x0 = x(1)\ and\ v0 = f(x(2))$$

f(x) = f(x) $= \sin(x)\ (1 + \alpha(\sin(\beta))$

which looks likes this in the matlab function:

```
g=@(t,x) [x(2);-sin(x(1))*(1+af*sin(bt*t))];
```

The function makes use of the built-in ode solver ode45 to solve this system of equations. It does this for 50 different startingpoints linearly divided between $\pi \leq x0 \leq \pi$ and $-3 \leq v0 \leq 3$. To linearly divide the startingpoints linspace is used thus a good spread over the x0 and v0 values is calculated. The built-in function wrapToPi is used to keep x0 between π and $-\pi$.

Error estimate

In my function I have set some options which increase the accuracy compared to the default settings of ode45. The function is reliable and works $\alpha \geq 0$ and $\beta > 0$. To produce figure 3 takes the function:

```
>> tic, pendulum(0,5,50) ,toc
Elapsed time is 1.558811 seconds.
```

Picture and Questions

pendulum(0,5,50)

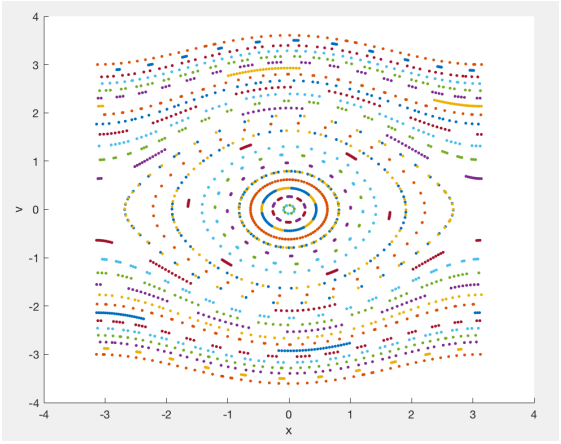


Figure 3 pendulum(0,5,50)

pendulum(3,5,50)

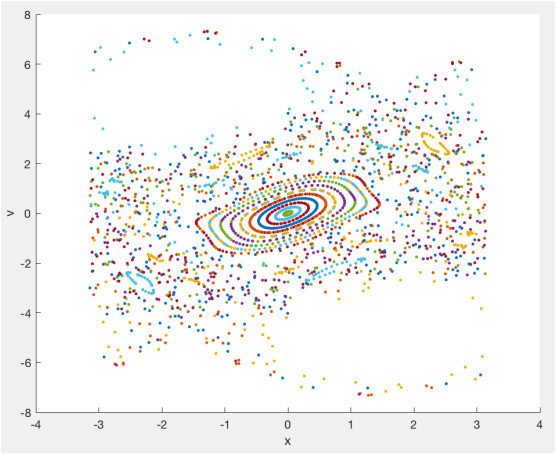


Figure 4 pendulum(3,5,50)

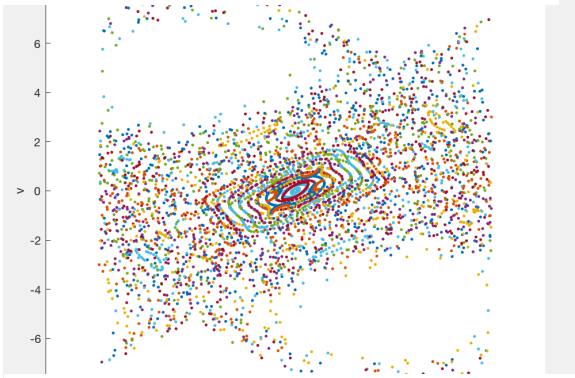


Figure 5 pendulum(5,5,50)

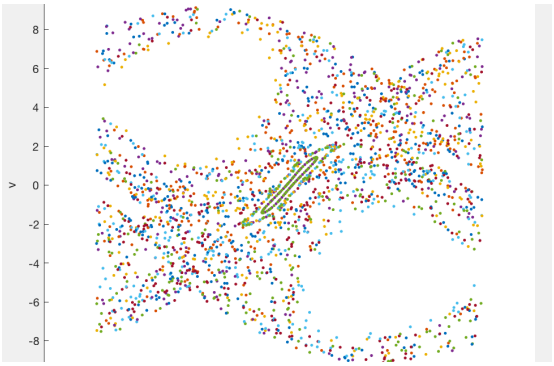


Figure 6 pendulum(10,5,50)

The bigger alfa gets the less symmetry occurs and the more chaotic the time-tau map becomes.