P. Grohs
W. Wu

Spring Term 2016

Numerical Analysis II

ETH Zürich
D-MATH

Problem Sheet 12

Problem 12.1 Mathematical Pendulum

The mathematical Pendulum equation is given by

$$\begin{cases} \dot{q} &= p, \\ \dot{p} &= -\sin q. \end{cases} \tag{12.1.1}$$

(12.1a) Show that, (12.1.1) is a Hamilton system for the Hamilton function

$$H(p,q) = \frac{1}{2}p^2 - \cos q.$$

Solution:

$$-\frac{\partial H}{\partial q} = -\sin q = \dot{p}, \qquad \frac{\partial H}{\partial p} = p = \dot{q}.$$

(12.1b) Show that, the energy $H(p,q)$ is a conserved quantity of (12.1.1).

Solution:

$$\frac{\mathrm{d}}{\mathrm{d}t}H(p,q) = \frac{2}{2}p\dot{p} + (\sin q)\dot{q} = p(-\sin q) + (\sin q)p = 0.$$

(12.1c) Complete the MATLAB-template `pendelEuler.m` by solving (12.1.1) using the explicit Euler method. Plot the evolution of the energy as a function of time. What do you observe?

Solution: The energy is not conserved (energy drift). See Figure 12.1 and Listing 12.1.

Listing 12.1: Evolution of the energy for the explicit Euler method.

```
function pendelEuler

%Initialising
N=100;
T=10;
h=T/N;

%right hand side of the pendulum equations
f=@(y) [y(2); -sin(y(1))];

%energy function (Hamilton function)
```

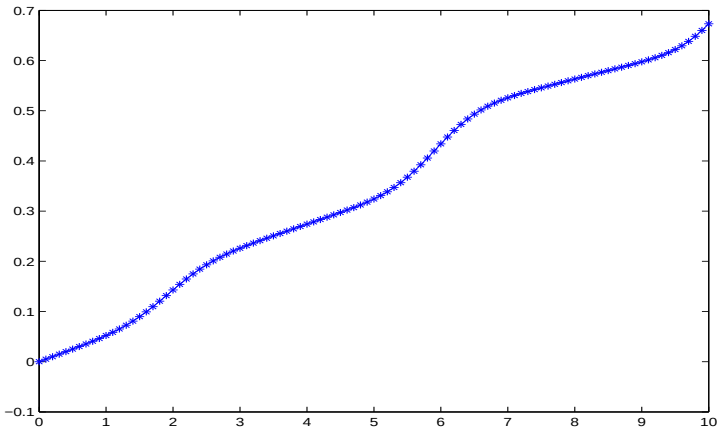

Figure 12.1: Evolution of the energy for the explicit Euler method.

```
H=@(y) 0.5*y(2)^2-cos(y(1));

%initial conditions
y=[pi/2;0];

%array in which you save the evolution of the energy
energy=zeros(N+1,1);
energy(1)=H(y);

for ii=1:N
    %perform one step of the explicit Euler method
    y=y+h*f(y);

    %save the energy
    energy(ii+1)=H(y);
end

%plot the evolution of the energy
figure;
plot(0:h:T,energy,'*-')
```

(12.1d) Complete the MATLAB-template `pendelMpr.m` by solving (12.1.1) using the implicit midpoint method and `nNewton` iterations of the Newton method. Execute the function for different values of `nNewton` and plot the evolution of the energy. What do you observe?

Solution: The energy is conserved if enough Newton iterations are used (no energy drift, only engery oscillations). See Figure 12.2 Figure 12.3 and Listing 12.2.

Listing 12.2: Evolution of the energy for the implicit midpoint method.

```
function pendelMpr

%Initialising
```

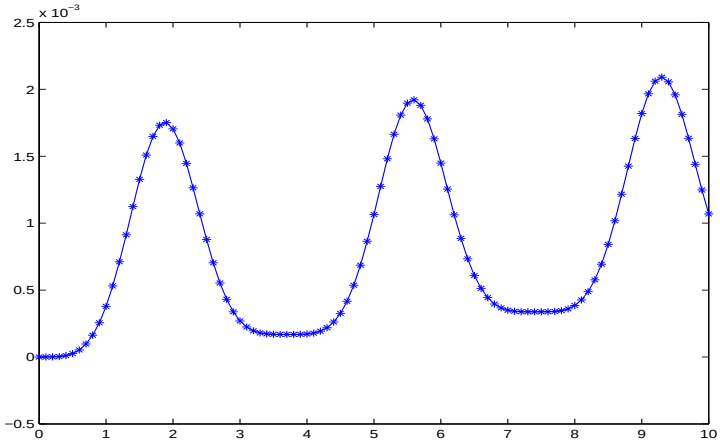

Figure 12.2: Evolution of the energy for the implicit midpoint method with one Newton iteration.

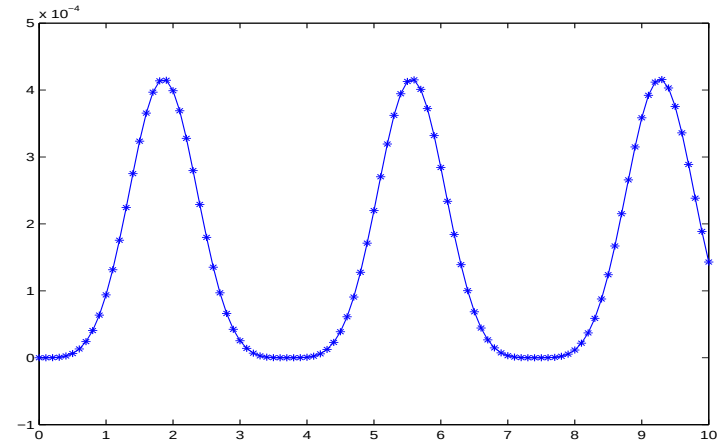

Figure 12.3: Evolution of the energy for the implicit midpoint method with two Newton iteration.

```
N=100;
T=10;
h=T/N;

%number of newton iterations
nNewton=1;

%right hand side of the pendulum equation, as well as its
   Jacobian
f=@(y) [y(2); -sin(y(1))];
Df=@(y) [0 1; -cos(y(1)) 0];

%function for the finding roots problem transformation and
   its Jacobian
F=@(y,y0) y-y0-h*f((y0+y)/2);
DF=@(y,y0) eye(length(y))-h/2*Df((y0+y)/2);
```

```
18
19 %energy function (Hamilton function)
20 H=@(y) 0.5*y(2)^2-cos(y(1));
21
22 %initial conditions
23 y=[pi/2;0];
24
25 %array in which the evolution of the energy is saved
26 energy=zeros(N+1,1);
27 energy(1)=H(y);
28
29 for ii=1:N
30     %perform one step of the implicit midpoint method
31     y0=y;
32     %with nNewton steps
33     for jj=1:nNewton
34         y=y-DF(y,y0)\F(y,y0);
35
36     end
37     %save the energy
38     energy(ii+1)=H(y);
39 end
40
41 %plot the evolution of the energy
42 figure;
43 plot(0:h:T,energy,'*-')
```

(12.1e) Complete the MATLAB-template `pendelMprKonv.m`, in which the convergence of the energy oscillations of the implicit midpoint method is investigated. The template solves (12.1.1) using the implicit midpoint for different time steps and saves the evolution of the energy in an array `energy`. You can read the maximal energy oscillation directly from this array, because the energy at the beginning is given. What do you observe?

Solution: The energy oscillations converge with the same order as the method. This is in fact a theorem, see [NUMODE, Thm. 4.4.91]. See Figure 12.4 and Listing 12.3.

Listing 12.3: Convergence of the energy oscillation for the implicit midpoint method with three Newton iterations.

```
1 function pendelMprKonv
2
3 %initialisation
4 N=100:100:1000;
5 T=10;
6
7 %fixed number of newton iterations
8 nNewton=3;
```

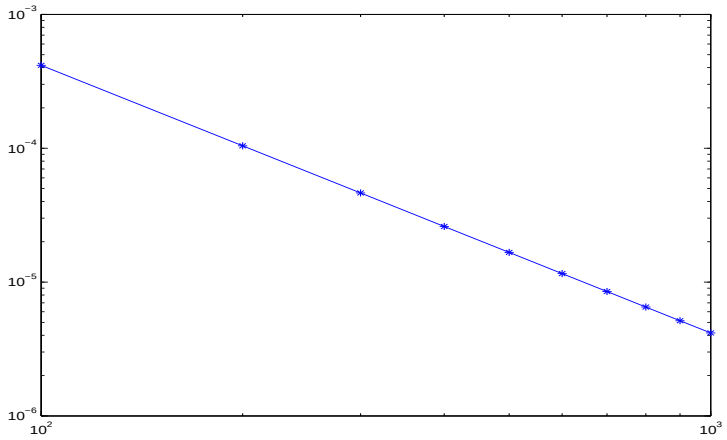

Figure 12.4: Convergence of the energy oscillation for the implicit midpoint method with three Newton iterations.

```
9
10  %energy function (Hamilton function)
11  H=@(y) 0.5*y(2)^2-cos(y(1));
12
13  %initial condition (is used several times)
14  yStart=[pi/2;0];
15
16  %array, which saves the oscillations of the energy
17  energyoscillation=zeros(size(N));
18
19  %for all numbers of steps
20  for kk=1:length(N)
21
22      %set the initial condition
23      y=yStart;
24      %determine the step size
25      h=T/N(kk);
26
27      %formulate the root finding procedure
28      %(the function depends on h!!)
29      f=@(y) [y(2); -sin(y(1))];
30      Df=@(y) [0 1; -cos(y(1)) 0];
31      F=@(y,y0) y-y0-h*f((y0+y)/2);
32      DF=@(y,y0) eye(length(y))-h/2*Df((y0+y)/2);
33
34      %array, in which you save the evolution of the energy
35      energy=zeros(N(kk)+1,1);
36      energy(1)=H(y);
37
38      for ii=1:N(kk)
```

```
        %perform one step of the implicit midpoint method
        y0=y;
        %with nNewton steps
        for jj=1:nNewton
            y=y-DF(y,y0)\F(y,y0);

        end
        %save the energy
        energy(ii+1)=H(y);
    end
    %read the maximum absolute energy oscillation
    energyoscillation(kk)=max(abs(energy));
end

%loglog plot
figure;
loglog(N,energyoscillation,'*-');

%determine the algebraic convergence
fit=polyfit(log(N),log(energyoscillation),1);
fprintf('order of convergence %2.4f\n', fit(1));
```

Problem 12.2 Störmer-Verlet Method and Phase Volume

The volume in the phase space is an important invariant of the Hamilton's equations. Let $\boldsymbol{\Phi}^t$ be the continuous evolution of a Hamilton's equations and let $\mathbf{V}$ be some measurable subsets of the phase space. We identify these subsets as a set of initial values for the Hamilton's equations and define its time-based evolution as

$$\boldsymbol{\Phi}^t\mathbf{V} := \left\{ \boldsymbol{\Phi}^t\mathbf{y} \,\middle|\, \mathbf{y} \in \mathbf{V} \right\}.$$

The volume conservation in the phase space [NUMODE, Def. 4.2.1] is then given by

$$\int_{\boldsymbol{\Phi}^t\mathbf{V}} 1 \,\mathrm{d}\mathbf{x} = \int_{\mathbf{V}} 1 \,\mathrm{d}\mathbf{x} \quad \text{for all } t > 0. \tag{12.2.1}$$

(12.2a) Show that (12.2.1) holds if $|\det \mathbf{W}(t; t_0, \mathbf{y})| = 1$ for all $t > 0$ and for all $\mathbf{y} \in \mathbf{V}$, where $\mathbf{W}(t; t_0, \mathbf{y})$ is the Wronskian [NUMODE, Eq. (1.3.33)].

Solution:

$$\int_{\boldsymbol{\Phi}^t\mathbf{V}} 1 \,\mathrm{d}\mathbf{x} = \int_{\mathbf{V}} |\det D\,\boldsymbol{\Phi}^t\mathbf{x}| \,d\mathbf{x} = \int_{\mathbf{V}} |\det \mathbf{W}(t; t_0, \mathbf{x})| \,\mathrm{d}\mathbf{x}.$$

(12.2b) Now we want to numerically investigate, to what extent this important invariance is conserved by the Störmer-Verlet method. The Störmer-Verlet method is a symplectic method for solving differential equations of the form

$$\begin{cases} \dot{\mathbf{y}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= \mathbf{f}(t, \mathbf{y}), \end{cases} \tag{12.2.2}$$

whose continuous evolution is naturally volume conserving (see [NUMODE, Thm. 4.2.3]).

The Störmer-Verlet method is given by

$$\begin{aligned}\mathbf{v}_{k+1/2} &= \mathbf{v}_k + \frac{h}{2}\mathbf{f}(t_k, \mathbf{y}_k),\\ \mathbf{y}_{k+1} &= \mathbf{y}_k + h\mathbf{v}_{k+1/2},\\ \mathbf{v}_{k+1} &= \mathbf{v}_{k+1/2} + \frac{h}{2}\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}).\end{aligned}$$

State the variational equation [NUMODE, Eq. (1.3.34)] for (12.2.2) and transform it so that it is of the same form as (12.2.2).

Solution: The variational equation is

$$\dot{\mathbf{W}}\left(t; t_0, \begin{pmatrix}\mathbf{y}_0\\ \mathbf{v}_0\end{pmatrix}\right) = \begin{pmatrix}0 & \mathbf{I}\\ D\mathbf{f}(t,\mathbf{y}) & 0\end{pmatrix}\mathbf{W}\left(t; t_0, \begin{pmatrix}\mathbf{y}_0\\ \mathbf{v}_0\end{pmatrix}\right).$$

Let us define matrices $\mathbf{V}$ and $\mathbf{Y}$ by

$$\mathbf{W} = \begin{pmatrix}\mathbf{Y}^\top\\ \mathbf{V}^\top\end{pmatrix}.$$

It follows

$$\begin{cases}\dot{\mathbf{Y}} &= \mathbf{V},\\ \dot{\mathbf{V}} &= \left(D\mathbf{f}(t,\mathbf{y})\mathbf{Y}^\top\right)^\top.\end{cases}$$

(12.2c) Complete the MATLAB template `stoermerverlet.m`, in which the initial value problem (12.2.2) and its variational equation for $\mathbf{f}(t,\mathbf{y}) := -\sin(\mathbf{y})$ are solved simultaneously using the Störmer-Verlet method. Plot the evolution of the determinant of the Wronskian as a function of time. What do you observe?

Solution:

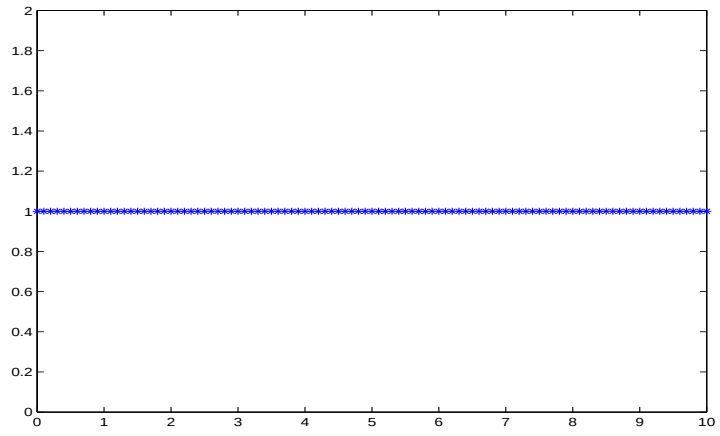

Figure 12.5: Evolution of the determinant of the Wronskian

Listing 12.4: Evolution of the determinant of the Wronskian

```
function stoermerverlet

% initialisation
N=100;
T=10;
h=T/N;

% function f
f=@(y) -sin(y);

% initial condition
v0=0;
y0=pi/2;

% prepare arrays
v=zeros(N+1,1);
y=zeros(N+1,1);

% set initial conditions
v(1)=v0;
y(1)=y0;

% functions and vectors for solving the differential equation
F=@(y,Y) Y*(-cos(y));
V=zeros(2,N+1);
Y=zeros(2,N+1);
V(:,1)=[0;1];
Y(:,1)=[1;0];
detW=zeros(N+1,1);
detW(1)=det([Y(:,1)';V(:,1)']);

for ii=1:N
% solve the IVP
    v05=v(ii)+h/2*f(y(ii));
    y(ii+1)=y(ii)+h*v05;
    v(ii+1)=v05+h/2*f(y(ii+1));

% Solve the variational equation
    V05=V(:,ii)+h/2*F(y(ii),Y(:,ii));
    Y(:,ii+1)=Y(:,ii)+h*V05;
    V(:,ii+1)=V05+h/2*F(y(ii+1),Y(:,ii+1));

% compute the determinant of the Wronskian
    detW(ii+1)=det([Y(:,ii+1)';V(:,ii+1)']);

end
```

```
47
48 % plot the evolution of the determinant
49 figure;
50 plot(0:h:T,detW,'*')
51 ylim([0 2])
52 max(detW)-1
```

(12.2d) Prove that the observation we can draw from subproblem (12.2c) is not unexpected. This of course implies that the Störmer-Verlet method conserves the phase volume even though the Wronskian is approximated by the Störmer-Verlet method.

HINT: The Störmer-Verlet method satisfies the claim of [NUMODE, Lem. 4.2.7] in the same sense as in subproblem (12.2c).

Solution: [NUMODE, Lem. 4.2.7] implies that the derivative (componentwise) of the discrete evolution is the same as the discrete evolution of the variational equation. We have

$$D_{\binom{\mathbf{y}_0}{\mathbf{v}_0}} \boldsymbol{\Psi}^h \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{v}_0 \end{pmatrix} = \boldsymbol{\Psi}^h \mathbf{W}(0),$$

where

$$\boldsymbol{\Psi}^h \mathbf{W}(0) := \begin{pmatrix} \mathbf{Y}^\top \\ \mathbf{V}^\top \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{Y} \\ \mathbf{V} \end{pmatrix} = \boldsymbol{\Psi}^h \begin{pmatrix} \mathbf{Y}_0 \\ \mathbf{V}_0 \end{pmatrix}.$$

The Störmer-Verlet method is symplectic and additionally satisfies [NUMODE, Def. 4.4.12]

$$D\,\boldsymbol{\Psi}^h \begin{pmatrix} \mathbf{y} \\ \mathbf{v} \end{pmatrix}^\top \mathbf{J}\, D\,\boldsymbol{\Psi}^h \begin{pmatrix} \mathbf{y} \\ \mathbf{v} \end{pmatrix} = \mathbf{J}.$$

Taking determinants of both sides gives $|\det\left(D\,\boldsymbol{\Psi}\binom{\mathbf{y}}{\mathbf{v}}\right)| = 1$, which is what we wanted to show.

Problem 12.3 Hamiltonian Differential Equation

We consider the differential equation

$$\begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} = \frac{1}{p^2+q^2} \begin{pmatrix} p \\ q \end{pmatrix}. \tag{12.3.1}$$

on $U := \mathbb{R}^2 \setminus \{0\}$.

(12.3a) Show that, the flow of the differential equation (12.3.1) is symplectic on every simply connected sub domain V of U.

HINT: For an appropriate definition of the logarithm (12.3.1) is a Hamiltonian differential equation on V with the Hamilton function $H(p,q) = -\operatorname{Im}\log(p+\mathrm{i}q)$.

Solution: On every simply connected domain one can find a meaningful definition of the logarithm (see the book Complex Analysis by Serge Lang p.120-121) which can be differentiated and hence show that (12.3.1) is a Hamilton differential equation.

(12.3b) Why is (12.3.1) no Hamilton differential equation on the entire domain U?

Solution: One can not find meaningful definition of the logarithm on all of U.

(12.3c) Implement the symplectic Euler method [NUMODE, Eq. (4.4.27)] for (12.3.1) in a MATLAB function

`[t,p,q]=SympEuler(p0,q0,tspan,h)`.

The initial value is given by `p0,q0`, range of integration `tspan` is of the form $[T_0, T_{\text{end}}]$, `h` is the step size, `t` is the time vector and `p,q` contain the approximated solutions at every time.

HINT: Use the MATLAB function `fsolve()` to solve non-linear systems of equations. The input and output parameters of this function are explained in `doc fsolve` or `help fsolve`

Solution: See Listing 12.5.

Listing 12.5: Implementation of subproblem (12.3c)

```
function [t, p, q] = SympEuler(p0, q0, tspan, h)
% symplectic Euler for
%
%  [p',q'] = [p,q]/(p^2 + q^2)
%
% Input paramters:
% p0: initial value for p
% q0: initial value for q
% tspan: [t0, T] starting and stopping time
% h: timestep size
%
% Outputparameters
% t: dicrete time points
% p: solution for p at time points
% q: solution for q at time points

% Time intervall
% number of time steps
N = ceil(diff(tspan)/h);
h = diff(tspan)/N;
t = linspace(tspan(1),tspan(2),N+1);

% memory allocation
p = zeros(1,N+1); p(1) = p0;
q = zeros(1,N+1); q(1) = q0;

% functions for each equation
dp = @(P,Q) (P/(P^2 + Q^2));
dq = @(P,Q) (Q/(P^2 + Q^2));

for k = 2:length(t)
  % solve p implicitly
  F = @(P) (p(k-1) + h*dp(P,q(k-1)) - P);
  options = optimset('Display','off');  % Turn off Display
  p(k) = fsolve(F,p(k-1),options);

  % calculate q explicitly
```

```
  q(k) = q(k-1) + h*dq(p(k),q(k-1));
end
```

(12.3d) Implement the classical Runge-Kutta method of order 4 [NUMODE, Eq. (2.3.11)] for (12.3.1) in a MATLAB function

`[t,p,q]=RK4(p0,q0,tspan,h).`

The initial value is given by `p0,q0`, range of integration `tspan` is of the form $[T_0, T_{\text{end}}]$, `h` is the step size, `t` is the time vector and `p,q` contain the approximated solutions at every time.

Solution: See Listing 12.6

Listing 12.6: Implementation of subproblem (12.3d)

```
function [t, p, q] = RK4(p0, q0, tspan, h)
% RK4 method for
%
%  [p',q'] = [p,q]/(p^2 + q^2)
%
% Input paramters:
% p0: initial value for p
% q0: initial value for q
% tspan: [t0, T] starting and stopping time
% h: timestep size
%
% Outputparameters
% t: dicrete time points
% p: solution for p at time points
% q: solution for q at time points

% Time intervall
% number of time steps
N = ceil(diff(tspan)/h);
h = diff(tspan)/N;
t = linspace(tspan(1),tspan(2),N+1);

% memory allocation
p = zeros(1,N+1); p(1) = p0;
q = zeros(1,N+1); q(1) = q0;

% system function for x = [p;q]
f = @(x) (x./(x(1)^2 + x(2)^2));

x = [p(1);q(1)];
for k = 2:length(t)
  k1 = f(x);
  k2 = f(x + h/2*k1);
```

```
  k3 = f(x + h/2*k2);
  k4 = f(x + h*k3);

  x = x + h/6*(k1 + 2*k2 + 2*k3 + k4);
  p(k) = x(1);
  q(k) = x(2);
end
```

(12.3e) Implement the MATLAB function `convergence.m` that uses (12.3.1) to determine the order of convergence of the symplectic Euler and the Runge-Kutta method in a numerical experiment. Use the start values $p(0) = 3$, $q(0) = 4$ and the stop time $T = 1$. The step size h should run through the values $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}, 2^{-7}$. What are the rates of convergence of both methods?

HINT: The analytic solution is $p(T) = 3\sqrt{\frac{2}{25}T + 1}, q(T) = 4\sqrt{\frac{2}{25}T + 1}$.

Solution: We get the following orders of convergence:

- `Order of convergence of SympEuler = 1.0012`,
- `Order of convergence of RK4 = 4.0295`.

For the order of `RK4` leave out the last two values of the error because the machine accuracy is reached. See Listing 12.7 and Figure 12.6.

Listing 12.7: Implementation of subproblem (12.3e)

```
% script for convergence studies

% allocate memory
kmax = 7;
h = (1/2).^(1:kmax);
N = length(h);
err_rk = zeros(1,N);
err_se = zeros(1,N);

% stop time
T = 1;

% initial data
p0 = 3;
q0 = 4;

% exact solution
p_ex = sqrt(2*T/(p0^2+q0^2)+1)*p0;
q_ex = sqrt(2*T/(p0^2+q0^2)+1)*q0;

for k = 1:N
  [t, prk, qrk] = RK4(p0, q0, [0 T], h(k));
```

```
  err_rk(k) = sqrt((prk(end) - p_ex)^2 + (qrk(end) - q_ex)^2);

  [t, pse, qse] = SympEuler(p0, q0, [0 T], h(k));
  err_se(k) = sqrt((pse(end) - p_ex)^2 + (qse(end) - q_ex)^2);
end

figure(1)
loglog(h,err_rk,'r*-', h, h.^4/10^10, 'k-.',h, err_se, 'b+-',
   h, h, 'k:')
grid on
legend('Runge-Kutta', 'Order 4', 'Symplectic Euler', 'Order
   1',...
       'Location','SouthEast')
print -depsc 'errorRate.eps'

% find convergence rate
p=polyfit(log(h),log(err_se),1);
fprintf('Convergence rate of SympEuler = %2.4f\n',p(1));
p=polyfit(log(h(1:end-2)),log(err_rk(1:end-2)),1);
fprintf('Convergence rate of RK4 = %2.4f\n',p(1));
```

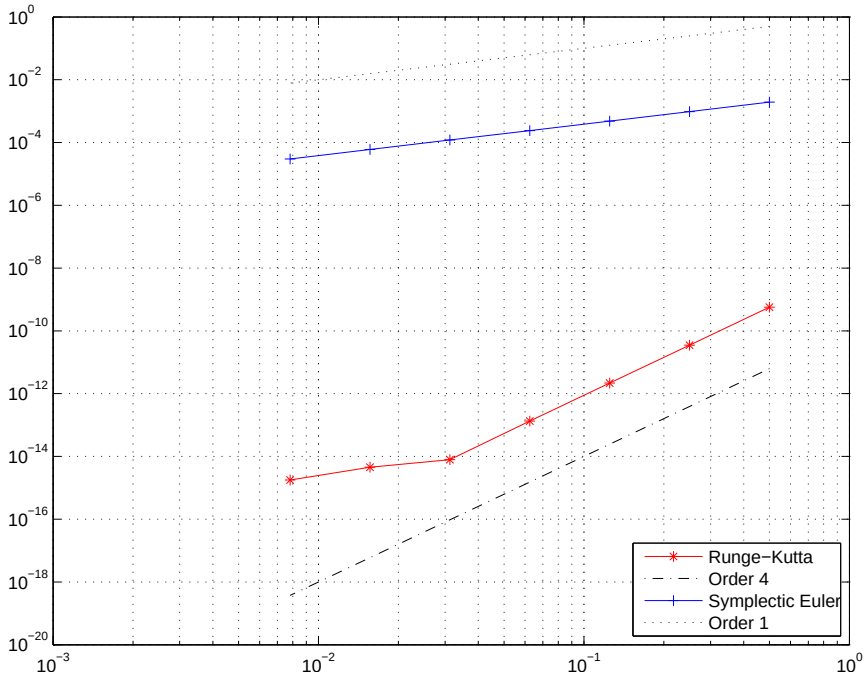

Figure 12.6: Convergence rate subproblem (12.3e).

We consider the following Hamilton function $H(p,q) = -\operatorname{Im}\log(p+iq)$ to the initial value problem (12.3.1) with the initial values $p(0) = 3$, $q(0) = 4$. log is the logarithm function in MATLAB.

(12.3f) In a MATLAB file `energy.m` plot the trajectory of the Hamilton function $H(p,q) = -\operatorname{Im}\log(p+iq)$ for the solution of the symplectic Euler method and the RK4-method. Use the stop time $T = 100$ and the step size $h = \frac{1}{2}$. Also plot the trajectory of the numeric solution in the (p,q) plane.

Solution: See Listing 12.8 and Figure 12.7, Figure 12.8 and Figure 12.9.

Listing 12.8: Implementation of subproblem (12.3f)

```
% script for plotting numerical solutions

% Invariant
I=@(p,q) -imag(log(p+i*q));

%  time intervall
Tspan=[0 100];

% initial data
p0=3;
q0=4;

h=1/2;

[t1,p1,q1]=RK4(p0,q0,Tspan,h);
[t2,p2,q2]=SympEuler(p0,q0,Tspan,h);

I1=I(p1,q1);
I2=I(p2,q2);

figure;
plot(t1,I1,'r*-',t2,I2,'b+-');
xlabel('t'); ylabel('Energy');
legend('RK4','Sympl Euler');
print -depsc energy.eps

figure; plot(p1,q1,'r*-');
xlabel('p'); ylabel('q');
legend('RK4');
print -depsc RK4.eps

figure; plot(p2,q2,'b*-');
xlabel('p'); ylabel('q');
legend('SympEuler');
print -depsc SympEuler.eps
```

(12.3g) In a MATLAB file `energyVar.m` investigate the oscillation of the Hamilton function $H(p,q) = -\operatorname{Im}\log(p+\mathrm{i}q)$ for the solutions of the Runge-Kutta method in dependence

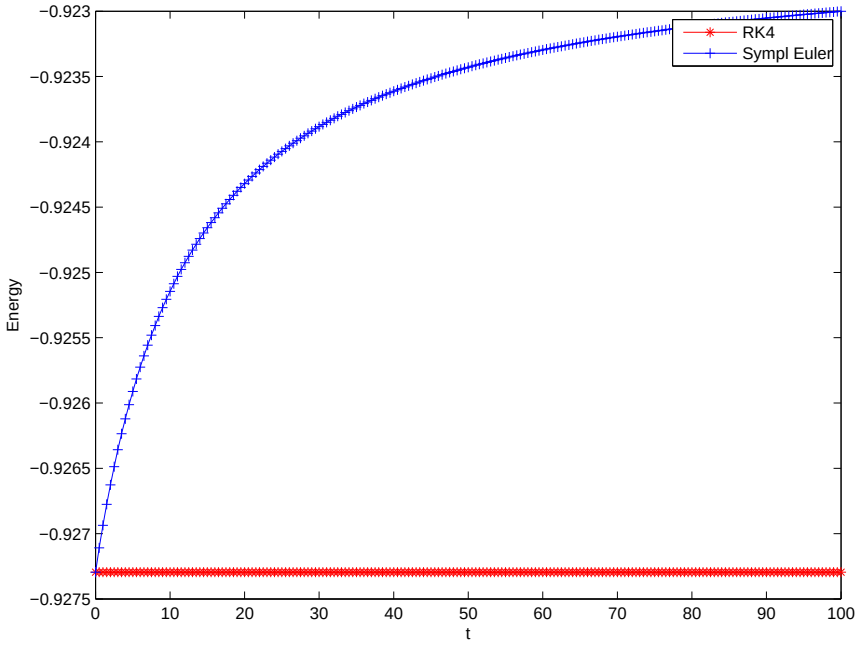

Figure 12.7: subproblem (12.3f): Trajectory of the Hamilton function

on the step size h. Use the stop time $T = 1$. The step size h should run through the values $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}, 2^{-7}$. Plot the results.

Solution: See Listing 12.9 and Figure 12.10. We observe that RK4 preserves the Hamilton function accurately (up to machine accuracy).

Listing 12.9: Implementation of subproblem (12.3g)

```
% Invariant
H=@(p,q) -imag(log(p+i*q));

% allocate memory
kmax = 7;
h = (1/2).^(1:kmax);
N = length(h);
err_rk = zeros(1,N);
err_se = zeros(1,N);

% stop time
T = 1;

% initial data
p0 = 3;
q0 = 4;

for k = 1:N
  [t, prk, qrk] = RK4(p0, q0, [0 T], h(k));
  H_rk = H(prk, qrk);
```

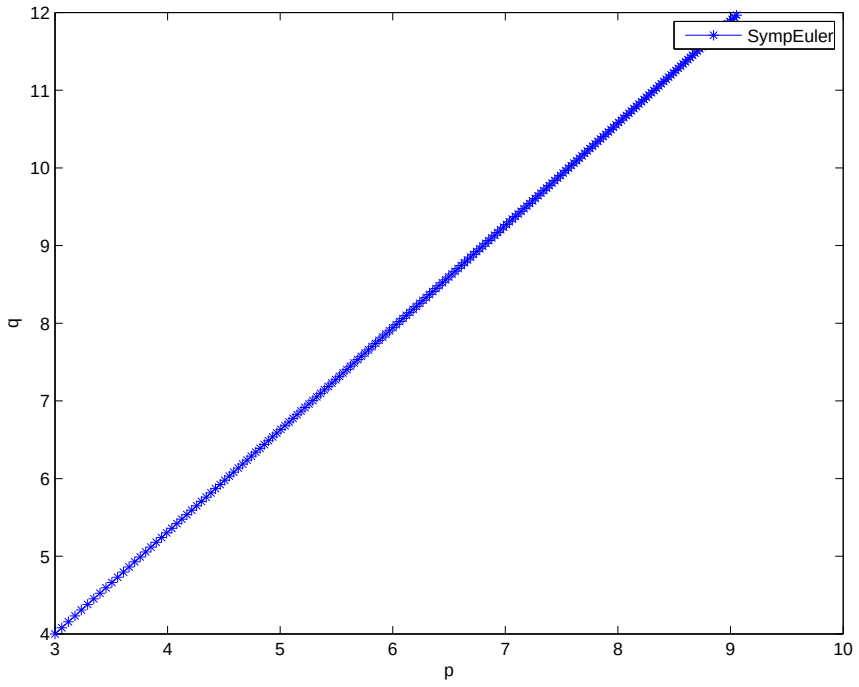

Figure 12.8: subproblem (12.3f): Trajectory of the symplectic Euler method

```
  diff_rk(k) = max(H_rk) - min(H_rk);

  [t, pse, qse] = SympEuler(p0, q0, [0 T], h(k));
  H_se = H(pse, qse);
  diff_se(k) = max(H_se) - min(H_se);
end

figure(1)
loglog(h,diff_rk,'r*-', h, h.^4/10^10, 'k-.',h, diff_se,
   'b+-', h, h, 'k:')
legend('Runge-Kutta', 'Order 4', 'Symplectic Euler', 'Order
   1',...
       'Location','SouthEast')
grid on;
print -depsc 'errorVar.eps'
```

(12.3h) Prove that, classical Runge-Kutta method of order 4 [NUMODE, Eq. (2.3.11)] conserves the Hamilton function corresponding to (12.3.1) accurately (in absence of rounding error).

HINT: $\operatorname{Im}\log(p+\mathrm{i}q) = \arg(p+\mathrm{i}q)$.

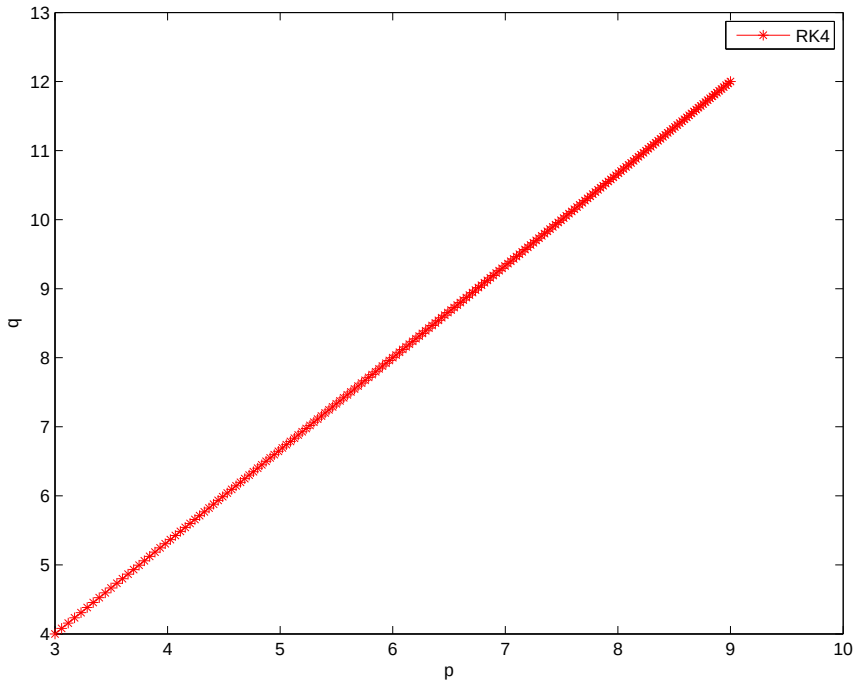

Figure 12.9: subproblem (12.3f): Trajectory of RK4

Solution: For appropriate $r_1(p_0, q_0), \dots r_4(p_0, q_0)$ we have:

$$
\begin{aligned}
k_1 &= r_1(p_0, q_0)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix} \\
k_2 &= \left(r_2(p_0, q_0) + \frac{h}{2} r_1(p_0, q_0) \right)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix} \\
k_3 &= \left(r_3(p_0, q_0) + \frac{h}{2} r_2(p_0, q_0) + \frac{h^2}{4} r_1(p_0, q_0) \right)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix} \\
k_4 &= \left(r_4(p_0, q_0) + h r_3(p_0, q_0) + \frac{h^2}{2} r_2(p_0, q_0) + \frac{h^3}{4} r_1(p_0, q_0) \right)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix}
\end{aligned}
$$

respectively for appropriate $s_1(p_0, q_0), \dots s_4(p_0, q_0)$:

$$
\begin{aligned}
k_1 &= s_1(p_0, q_0)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix} \\
k_2 &= s_2(p_0, q_0)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix} \\
k_3 &= s_3(p_0, q_0)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix} \\
k_4 &= s_4(p_0, q_0)\begin{pmatrix} p_0 \\ q_0 \end{pmatrix}.
\end{aligned}
$$

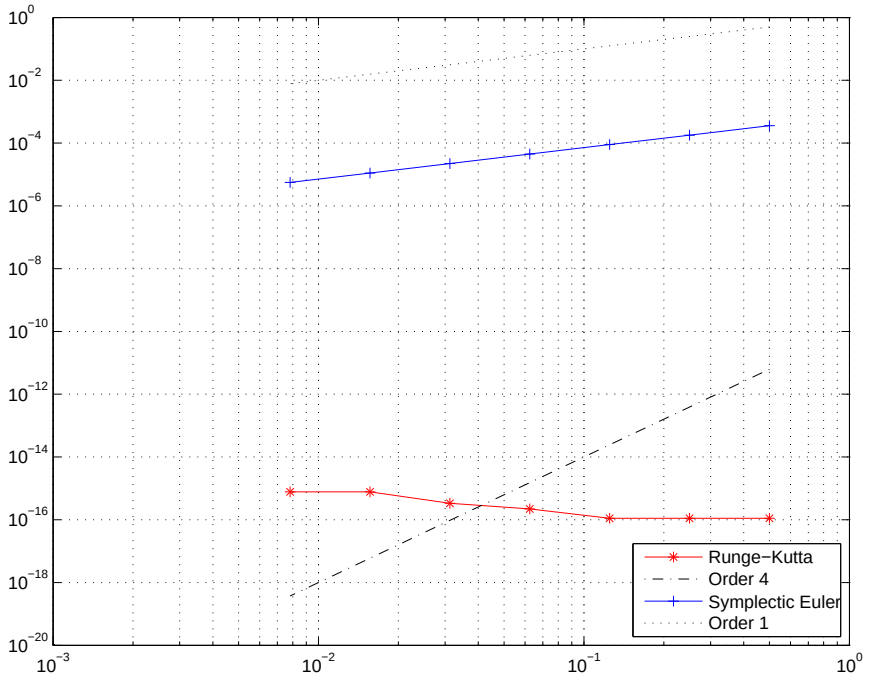

Figure 12.10: subproblem (12.3g): Variation of the Hamilton function

Therefore

$$\begin{aligned}\begin{pmatrix} p_1 \\ q_1 \end{pmatrix} &= \begin{pmatrix} p_0 \\ q_0 \end{pmatrix} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= \left(1 + \frac{h}{6}(s_1(p_0, q_0) + 2s_2(p_0, q_0) + 2k_3 s_3(p_0, q_0) + k_4 s_4(p_0, q_0))\right) \begin{pmatrix} p_0 \\ q_0 \end{pmatrix}.\end{aligned}$$

Hence one easily sees

$$\frac{p_1}{q_1} = \frac{p_0}{q_0}, \tag{12.3.2}$$

which implies $\arg(p + \mathrm{i}q) = const.$

In fact, the same conclusion holds for all explicit Runge-Kutta method with similar proof.

Published on 16 May 2016.

To be submitted by 24 May 2016.

References

[NODE] Lecture Notes for the course “Numerical Methods for Ordinary Differential Equations”.

[NUMODE] Lecture Slides for the course “Numerical Methods for Ordinary Differential Equations”, SVN revision # 52913.

Last modified on May 17, 2016