Problem Sheet 13

Problem 13.1 A Volume Preserving Splitting Scheme

Consider the following ODE

$$\dot{\mathbf{y}} = f(\mathbf{y}) = \begin{pmatrix} -y_2 - \frac{y_1}{a^2+y_3^2} \\ y_1 - \frac{y_2}{a^2+y_3^2} \\ \frac{2\arctan\left(\frac{y_3}{a}\right)}{a} \end{pmatrix}, \quad \mathbf{y} \in \mathbb{R}^3, \quad a > 0. \tag{13.1.1}$$

(13.1a) Show that the flow of the ODE (13.1.1) is volume preserving.

Solution: [NODE, Lemma. 4.2.3] says it suffices to show that f is a divergence-free vector field, that is, $\operatorname{div} f(y) = 0$, for all $y \in \mathbb{R}^3$. We have

$$\operatorname{div} f(\mathbf{y}) = \sum_{j=1}^{3} \frac{\partial f_j}{\partial y_i}(\mathbf{y}) = -\frac{1}{a^2+y_3^2} - \frac{1}{a^2+y_3^2} + \frac{2}{a^2+y_3^2} = 0,$$

as desired.

On one hand, [NODE, Lemma. 4.2.5] dictates that there does not exist a Runge-Kutta scheme that is volume preserving for all problems in $\mathbb{R}^3$. On the other hand however, any SSM that preserves quadratic invariants is volume preserving in $\mathbb{R}^2$.

(13.1b) Split the vector field f, given as the right hand side of (13.1.1), as a sum of two-dimensional divergence-free vector fields, that is, as $f = f_1 + f_2 = \begin{pmatrix} f_1^1 \\ 0 \\ f_1^3 \end{pmatrix} + \begin{pmatrix} 0 \\ f_2^2 \\ f_2^3 \end{pmatrix}$.

HINT: Consider the construction in [NODE, Lemma. 4.2.6]

Solution: For a given function $g : \mathbb{R}^3 \to \mathbb{R}^3$, denote by g^i, its i^{th} component. Using the splitting construction from [NODE, Lemma. 4.2.6] , we are led to define functions

$$f_1(\mathbf{y}) = (f_1^1(\mathbf{y}), 0, f_1^3(\mathbf{y}))^\top,$$
$$f_2(\mathbf{y}) = (0, f_2^2(\mathbf{y}), f_2^3(\mathbf{y}))^\top,$$

where

$$f_i^3(\mathbf{y}) = -\int \frac{\partial f^i}{\partial y_i}(\mathbf{y}) dy_3.$$

It follows

$$f_1^3(\mathbf{y}) = -\int \frac{\partial f^1}{\partial y_1}(\mathbf{y})dy_3 = \int \frac{1}{a^2 + y_3^2}dy_3 = \frac{\arctan\left(\frac{y_3}{a}\right)}{a},$$

and

$$f_2^3(\mathbf{y}) = -\int \frac{\partial f^2}{\partial y_2}(\mathbf{y})dy_3 = \int \frac{1}{a^2 + y_3^2}dy_3 = \frac{\arctan\left(\frac{y_3}{a}\right)}{a}.$$

The desired splitting is then given through

$$f_1(\mathbf{y}) = \left(f^1(\mathbf{y}), 0, \frac{\arctan\left(\frac{y_3}{a}\right)}{a}\right)^\top,$$
$$f_2(\mathbf{y}) = \left(0, f^2(\mathbf{y}), \frac{\arctan\left(\frac{y_3}{a}\right)}{a}\right)^\top,$$

where f^1 and f^2 are the second and third component functions of f.

Now that we have the splitting $f(\mathbf{y}) = f_1(\mathbf{y}) + f_2(\mathbf{y})$ we can construct a volume preserving SSM. Take a Runge-Kutta SSM that preserves quadratic invariants, and denote by Ψ_i^h the flow of this SSM when applied to the ODE $\dot{\mathbf{y}} = f_i(\mathbf{y})$, $i = 1, 2$. The functions f_1 and f_2 are divergence free vector fields, and are (essentially) two-dimensional, while the flows Ψ_i^h are volume preserving. Hence, by applying a suitable splitting scheme (here we use Strang splitting) we can construct a volume preserving scheme

$$\Psi^h = \Psi_1^{h/2} \circ \Psi_2^h \circ \Psi_1^{h/2}$$

since the composition of volume preserving schemes is again volume preserving.

(13.1c) Finish the MATLAB code

```
function y = GaussStep(y0, f, Df, h)
```

which computes one step of the Gauss collocation scheme of order 4. The inputs are the initial value `y0`, the right hand side of the given ODE `f`, the derivative of the right-hand side `Df`, and the stepsize `h`. In order to compute the coefficients of the given Runge-Kutta method use the codes `collCoeffs.m` and `GaussNodes.m`, and in order to solve the underlying implicit system for the stages, use Newton's algorithm by completing the template `newton.m`.

Solution:

Listing 13.1: One step of the Gauss collocation.

```
function y = GaussStep(y0, f, Df, h)

%calculate the methods coefficients
c = GaussNodes(1);
[A,b] = collCoeffs(c);
nNewton = 50;

```

```
% Initialise the values
y = y0;
k0 = [y;y];

% Rephrase the Gauss method as a root finding problem
F=@(k1, k2) [k1;k2]-[f(y+h*(A(1,1)*k1+A(1,2)*k2));
   f(y+h*(A(2,1)*k1+A(2,2)*k2))];

DF=@(k1, k2)
   eye(length(k0))-h*[A(1,1)*Df(y+h*(A(1,1)*k1+A(1,2)*k2)),
   A(1,2)*Df(y+h*(A(1,1)*k1+A(1,2)*k2));...
        A(2,1)*Df(y+h*(A(2,1)*k1+A(2,2)*k2)),
           A(2,2)*Df(y+h*(A(2,1)*k1+A(2,2)*k2))];

% Solve the implicit system with "nNewton" iterations of
   Newton's method
k=newton(k0,F,DF,nNewton);

% Find the solution y at the new point
y = y+h*(b(1)*k(1:3)+b(2)*k(4:6));
end
```

Listing 13.2: Newton method.

```
function x = newton(x0, F, DF, nNewton)

% Starting value for iteration
x = x0;
tol = 1e-16;

% loop
for ii = 1:nNewton

    %check whether DF(x) is invertible
    assert(det(DF(x(1:3), x(4:6)))~=0);

    %Newton step
    x = x-DF(x(1:3), x(4:6))\F(x(1:3), x(4:6));
    if abs(F(x(1:3), x(4:6))) < tol
        break
    end
end

end
```

(13.1d) Consider the inital value problem

$$\dot{\mathbf{y}} = \begin{pmatrix} -y_2 - \frac{y_1}{a^2+y_3^2} \\ y_1 - \frac{y_2}{a^2+y_3^2} \\ \frac{2\arctan\left(\frac{y_3}{a}\right)}{a} \end{pmatrix}, \quad \mathbf{y}(0) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad a = 1.$$

Find its solution by using the previously described volume-preserving splitting scheme. Here Ψ_1^h and Ψ_2^h are both to be computed using steps of the Gauss collocation method of order 4. Complete the template `VolumePreservingSplitting.m` in which you should use `GaussStep.m` from (13.1c) in each step, and Strang splitting to compute the approximate solution.

Solution:

Listing 13.3: Implementation.

```
function y = VolumePreservingSplitting

a = 1;
f1 = @(y) [-y(2)-y(1)./(a^2+y(3).^2); 0; atan(y(3)/a)/a];
f2 = @(y) [0;y(1)-y(2)./(a^2+y(3).^2); atan(y(3)/a)/a];
df1 = @(y) [-1./(a^2+y(3).^2), -1,
   y(1).*y(3)./(a^2+y(3).^2).^2;0 0 0; 0 0
   1./(a^2+y(3).^2).^2];
df2 = @(y) [0 0 0; 1,  -1./(a^2+y(3).^2),
   y(2).*y(3)./(a^2+y(3).^2).^2; 0 0 1./(a^2+y(3).^2).^2];

t = 0:0.05:5;
h = 0.05;
N = length(t);
y0 = [1; 1; 1];
y(:, 1) = y0;
for j = 2:N
    y(:, j) = GaussStep(y(:, j-1), f1, df1, h/2);
    y(:, j) = GaussStep(y(:, j), f2, df2, h);
    y(:, j) = GaussStep(y(:, j), f1, df1, h/2);
end
```

Problem 13.2 The Symplectic Euler Method

We consider the Hamiltonian differential equation:

$$\dot{\mathbf{p}} = -\frac{\partial}{\partial \mathbf{q}} H(\mathbf{p}, \mathbf{q}), \qquad \dot{\mathbf{q}} = \frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}, \mathbf{q}).$$

Show by direct calculations that, the so called symplectic Euler method from the lecture:

$$\mathbf{p}_1 = \mathbf{p}_0 - h\frac{\partial}{\partial \mathbf{q}} H(\mathbf{p}_1, \mathbf{q}_0), \qquad \mathbf{q}_1 = \mathbf{q}_0 + h\frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}_1, \mathbf{q}_0),$$

is in fact symplectic.

Solution: We need to show that, for the Jacobi-matrix $\frac{\partial(\mathbf{p}_1,\mathbf{q}_1)}{\partial(\mathbf{p}_0,\mathbf{q}_0)}$ the iteration rule holds:

$$\frac{\partial(\mathbf{p}_1,\mathbf{q}_1)}{\partial(\mathbf{p}_0,\mathbf{q}_0)}^\top \mathbf{J} \frac{\partial(\mathbf{p}_1,\mathbf{q}_1)}{\partial(\mathbf{p}_0,\mathbf{q}_0)} = \mathbf{J}.$$

We do this by differentiating the iteration rule of the symplectic Euler

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{p}_0 - h\frac{\partial}{\partial \mathbf{q}} H(\mathbf{p}_1, \mathbf{q}_0) \\ \mathbf{q}_1 &= \mathbf{q}_0 + h\frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}_1, \mathbf{q}_0) \end{aligned}$$

with respect to $(\mathbf{p}_0, \mathbf{q}_0)$ and get

$$\begin{aligned} \frac{\partial \mathbf{p}_1}{\partial \mathbf{p}_0} &= I - hH_{\mathbf{pq}}\frac{\partial \mathbf{p}_1}{\partial \mathbf{p}_0} \\ \frac{\partial \mathbf{p}_1}{\partial \mathbf{q}_0} &= -hH_{\mathbf{pq}}\frac{\partial \mathbf{p}_1}{\partial \mathbf{q}_0} - hH_{\mathbf{qq}} \\ \frac{\partial \mathbf{q}_1}{\partial \mathbf{p}_0} &= hH_{\mathbf{pp}}\frac{\partial \mathbf{p}_1}{\partial \mathbf{p}_0} \\ \frac{\partial \mathbf{q}_1}{\partial \mathbf{q}_0} &= I + hH_{\mathbf{pp}}\frac{\partial \mathbf{p}_1}{\partial \mathbf{q}_0} + hH_{\mathbf{qp}}, \end{aligned}$$

where $H_{\mathbf{qq}}, H_{\mathbf{pp}}, \ldots$ are the appropriate entries in the Hesse-matrix. In matrix notation:

$$\underbrace{\begin{pmatrix} I + hH_{\mathbf{pq}} & 0 \\ -hH_{\mathbf{pp}} & 1 \end{pmatrix}}_{=:\mathbf{A}} \left(\frac{\partial(\mathbf{p}_1,\mathbf{q}_1)}{\partial(\mathbf{p}_0,\mathbf{q}_0)}\right) = \underbrace{\begin{pmatrix} I & -hH_{\mathbf{qq}} \\ 0 & I + hH_{\mathbf{qp}} \end{pmatrix}}_{=:\mathbf{B}}.$$

We now use

$$\mathbf{A}^{-1} = \begin{pmatrix} (I + hH_{\mathbf{qq}})^{-1} & 0 \\ hH_{\mathbf{pp}}(I + hH_{\mathbf{pq}})^{-1} & 1 \end{pmatrix}$$

to show that

$$\mathbf{B}^\top \mathbf{A}^{-\top} \mathbf{J} \mathbf{A}^{-1} \mathbf{B} = \mathbf{J}.$$

Alternatively we could also show that,

$$\mathbf{A}\mathbf{J}^{-1}\mathbf{A}^\top = \mathbf{B}\mathbf{J}^{-1}\mathbf{B}^\top.$$

if we use $H_{\mathbf{pq}} = H_{\mathbf{qp}}^\top$.

Problem 13.3 Spring Pendulum

We consider the frictionless spring pendulum system with a massless spring, see [NUMODE, Ex. 4.4.35]. The Hamilton function:

$$H_K(\mathbf{p}, \mathbf{q}) = \frac{1}{2}(p_1^2 + p_2^2) + (\sqrt{q_1^2 + q_2^2} - 1)^2 + q_2$$

describes a spring pendulum with frictionless spring in cartesian coordinates i.e. $q_1 = x_1, q_2 = x_2$(Here x_2 axis points downwards). Notice here $\frac{1}{2}(p_1^2 + p_2^2)$ denotes the kinetic energy K of spring. In polar coordinates (r, φ), the kinetic energy K can be rewritten as

$$\begin{aligned} K &= \frac{1}{2}(p_1^2 + p_2^2) = \frac{1}{2}(\dot{x_1}^2 + \dot{x_2}^2) \\ &= \frac{1}{2}(\dot{r}^2 + r^2\dot{\varphi}^2). \end{aligned}$$

The generalized momentum conjugate to polar coordinates (r, φ) are

$$\begin{aligned} p_r &= \frac{\partial K}{\partial \dot{r}} = \dot{r}, \\ p_\varphi &= \frac{\partial K}{\partial \dot{\varphi}} = r^2\dot{\varphi}. \end{aligned}$$

Hence $K = \frac{1}{2}(p_r^2 + r^{-2}p_\varphi^2)$ in polar coordinates, and the Hamilton function is

$$H_P\left(\begin{pmatrix} p_r \\ p_\varphi \end{pmatrix}, \begin{pmatrix} r \\ \varphi \end{pmatrix}\right) = \frac{1}{2}(p_r^2 + r^{-2}p_\varphi^2) - r\sin(\varphi) + \frac{1}{2}(r-1)^2 .$$

(13.3a) Formulate an *explicit* symplectic Euler-method for

$$\dot{\mathbf{p}} = -\frac{\partial}{\partial \mathbf{q}} H_P(\mathbf{p}, \mathbf{q}), \quad \dot{\mathbf{q}} = \frac{\partial}{\partial \mathbf{p}} H_P(\mathbf{p}, \mathbf{q})$$

HINT: Write the symplectic Euler method in individual components $\mathbf{p}$ and $\mathbf{q}$.

Solution: The symplectic Euler

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{p}_0 - h\frac{\partial}{\partial \mathbf{q}} H(\mathbf{p}_1, \mathbf{q}_0) \\ \mathbf{q}_1 &= \mathbf{q}_0 + h\frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}_1, \mathbf{q}_0) \end{aligned}$$

in components

$$\begin{aligned} p_{r,1} &= p_{r,0} - h(-r_0^{-3}p_{\phi,1}^2 - \sin(\phi_0) + (r_0 - 1)) \\ p_{\phi,1} &= p_{\phi,0} - hr_0\cos(\phi_0) \\ r_1 &= r_0 + hp_{r,1} \\ \phi_1 &= \phi_0 + hr_0^{-2}p_{\phi,1}. \end{aligned}$$

Using this $p_{r,1}$ and $p_{\phi,1}$ can be explicitly determined by first calculating $p_{\phi,1}$ and then $p_{r,1}$.

(13.3b) Implement the symplectic Euler method from subproblem (13.3a) in MATLAB and solve the equation of motion for $\mathbf{p}_0 = 0$, $r_0 = 1$, $\varphi_0 = \pi/6$ for the time period $[0, 1000]$.

Solution: The MATLAB code illustrates the solutions of the symplectic solver for the Hamilton function in Polar and Cartesian coordinates (notice the differing definition of the Polar coordinates $x_2 = -r\sin(\phi)$ from subproblem (13.3a)).

Listing 13.4: subproblem (13.3b)

```
% NUMODE: ex:springpend
% evolution of spring pendulum
%

% Right hand side in Cartesian coordinates
Hq = @(q) ((norm(q)-1)*q/norm(q) + [0;1]);
Hp = @(p) (p);

f = @(y) [-Hq(y(3:4));Hp(y(1:2))];

% Right hand side in Polar coordinates
HP=@(p,q)1/2*(p(1)^2+q(1)^-2*p(2)^2)-q(1)*sin(q(2))+1/2*(q(1)-1)^2;
HPq_1=@(p,q) -1/q(1)^3*p(2)^2-sin(q(2))+(q(1)-1);
HPq_2=@(p,q) -q(1)*cos(q(2));

HPp_1=@(p,q) p(1);
HPp_2=@(p,q) 1/q(1)^2*p(2);

g = @(y) [-HPq_1(y(1:2),y(3:4));...
    -HPq_2(y(1:2),y(3:4)) ;...
    HPp_1(y(1:2),y(3:4));...
    HPp_2(y(1:2),y(3:4))];

% initial data in Cartesian coordinates
p0 = [0;0];
q0 = [cos(pi/6);-sin(pi/6)];

% initial data in Polar coordinates
Pp0=[0;0];
Pq0=[1;pi/6];

% timestep
h = 0.3;

% initial values
P = p0;
Q = q0+0.5*h*p0;
PP = Pp0;
PQ = Pq0+0.5*h*Pp0;

% energy
maxEn=HP(Pp0,Pq0);
minEn=HP(Pp0,Pq0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ANIMATION
```

```
k = 3;
spt = 1/(4*k)+1/(2*k)*(0:2*k-1);
spv = [[0,spt,1];[0,0.1*sin(2*k*pi*spt),0]];

figsv = figure('name','Spring Pendulum','position',[0 0 1600 800]);
subplot(1,2,1);
t = 0;

for i=1:1000
  subplot(1,2,1); q = Q; plotspring;
  subplot(1,2,2); q = [PQ(1)*cos(PQ(2)) -PQ(1)*sin(PQ(2))]; plotspring;

  % symplectic Euler, Cartesian coordinates
  P = P - h*Hq(Q);
  Q = Q + h*Hp(P);

  % symplectic Euler, Polar coordinates
  PP(2)=PP(2)-h*HPq_2(PP,PQ);
  PP(1)=PP(1)-h*HPq_1(PP,PQ);
  PQ(2)=PQ(2)+h*HPp_2(PP,PQ);
  PQ(1)=PQ(1)+h*HPp_1(PP,PQ);

  maxEn=max([maxEn,HP(PP,PQ)]);
  minEn=min([minEn,HP(PP,PQ)]);

  t = t+h;
end

[maxEn minEn]
```

Listing 13.5: subproblem (13.3b)

```
% plotspring.m
  lq = norm(q);
  sp = 1/lq*[q(1),-q(2);...
          q(2),q(1)]*[lq,0;0,1]*spv;
  plot(q(1),q(2),'ro','markersize',8); hold on;
  plot(sp(1,:),sp(2,:),'b-');
  plot([-1.5 1.5],[0 0],'k-');
  plot([0 0],[-4 1],'k-'); hold off;
  xlabel('{\bf q_1}','fontsize',14);
  ylabel('{\bf q_2}','fontsize',14);
  title(sprintf('t = %f',t));
  axis([-1.5 1.5 -4 1]);
  drawnow;
```

(13.3c) What observations do you make for large step sizes? Try to explain the observation.

Solution: For large time steps the symplectic Euler becomes unstable. The reason for this is the explicit part of the method.

(13.3d) Analyse the size of the oscillations of the total energy $H(\mathbf{p}, \mathbf{q})$ of the system during the numerical integration in dependence of the uniform time step size.

Solution: Variation of the energy for different time step sizes throughout 100 iterations:

h	min	max
0.6	0.0398	0.7636
0.4	0.2322	0.6816
0.2	0.3797	0.5955

Published on 23 May 2016.

To be submitted by 31 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 20, 2016