P. Grohs
W. Wu

Spring Term 2016

Numerical Analysis II

ETH Zürich
D-MATH

Problem Sheet 1

Introduction. Templates and `.p` files regarding the MATLAB problems can be found in the `.zip` file, which is on the website.

Problem 1.1 Collapsing Initial Value Problem

We will look at the following initial value problem from the lecture

$$\dot{y} = \sin(1/y) - 2, \quad y(0) = 1.$$

Show, using the integral's monotonicity, that the solution $y \in \mathcal{C}^1([0, t_+[, \mathbb{R})$ satisfies the inequality

$$y(t) \leq 1 - t \text{ for } 0 \leq t < t_+.$$

Explain why the solution necessarily experiences a collapse.

Solution: We have

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \sin(1/y) - 2 \leq -1.$$

From this and the monotonicity of the integral, it follows that

$$\begin{aligned} y(t) &= y(0) + \int_0^t (\sin(1/y) - 2) ds \\ &\leq y(0) - \int_0^t ds. \end{aligned}$$

We use the initial value condition and obtain

$$y(t) \leq 1 - t.$$

Furthermore, $0 < y(t)$, as the right hand is not defined at $y = 0$ and $y(0) = 1$. Thereby $t_+ \leq 1$ and the solution is restricted, i.e. only a collapse remains (cf. [NODE, Ex. 1.3.3]).

Problem 1.2 Cannon

Situated at the coordinate origin of the (x_1, x_2)-plane is a cannon which fires projectiles with initial velocity v_0 at an angle α with respect to the x_1-axis. The goal is to hit a target placed at the point $(250, 0)$. To model the trajectory of the projectile of mass m we assume that it is affected by the gravitational force mg, acting in negative x_2-direction, as well as the air resistance of magnitude ρv^2, which is directed opposite to the instantaneous flight direction at all times. Newton's second law yields the autonomous [NODE, Def. 1.1.5] ordinary differential equation of second order.

$$\ddot{\mathbf{x}} = g\begin{bmatrix} 0 \\ -1 \end{bmatrix} - \frac{\rho}{m}\|\dot{\mathbf{x}}\|\dot{\mathbf{x}} \tag{1.2.1}$$

describing the projectile's trajectory $\mathbf{x}(t)$. In addition, we obtain the initial values

$$\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \dot{\mathbf{x}}(0) = v_0 \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix}.$$

In the following, we will assume that $v_0 = 100$, $g = 10$, $m = 20$ and $\rho = 0.1$.

(1.2a) Convert (1.2.1) into an equivalent autonomous first order system $\dot{\mathbf{y}} = f(\mathbf{y})$, see [NODE, Rem. 1.1.6]. Determine the corresponding initial values for the first order system.

Solution: Let $y = \left[\begin{smallmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{smallmatrix}\right]$. We obtain

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{x}} \\ g\left[\begin{smallmatrix} 0 \\ -1 \end{smallmatrix}\right] - \frac{\rho}{m}\|\dot{\mathbf{x}}\|\dot{\mathbf{x}} \end{bmatrix} =: f(\mathbf{x}, \dot{\mathbf{x}}) = f(\mathbf{y}),$$

where we have omitted the time dependence of $\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$ for clarity. The initial values for the converted system can be read off the definition of $\mathbf{y}$:

$$\mathbf{y}(0) = \begin{bmatrix} \mathbf{x}(0) \\ \dot{\mathbf{x}}(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ v_0\cos\alpha \\ v_0\sin\alpha \end{bmatrix}.$$

(1.2b) Implement a MATLAB function

`function cannon_trajectory`

which calculates the trajectories of the projectile

- for a firing angle $\alpha = 20°$ in the time interval $t \in [0, 5]$,
- for a firing angle $\alpha = 40°$ in the time interval $t \in [0, 8.5]$,

using the standard integrator `ode45` in MATLAB and which plots both solutions in a single figure.

Solution: See `cannon_trajectory.m`.

Listing 1.1: Calculates the trajectories of the two projectiles.

```
1 function cannon_trajectory
2
3 %Student:
4
5 %set angle alpha (in radians)
6 alpha=2*pi/360*20;
7
8 %set constants
9 v0=100;
```

```
g=10;
m=20;
rho=0.1;

%set initial condition
%(attention: this is a 4x1 vector!)
y0=[0;0;v0*cos(alpha);v0*sin(alpha)];

%specify time interval
tspan=[0 5];

%specify right hand side
%(attention: output needs to be a 4x1 vector!)
rhs=@(t,y) [y(3:4);[0;-g]-rho/m*norm(y(3:4))*y(3:4)];

%carry out integration using ode45
[tout,yout]=ode45(rhs,tspan,y0);

%plot the trajectory
figure;
plot(yout(:,1),yout(:,2));
hold on;

%overwrite alpha with new value (in radians)
alpha=2*pi/360*40;

%specify intitial condition
%(attention: this is a 4x1 vector!)
y0=[0;0;v0*cos(alpha);v0*sin(alpha)];

%specify time interval
tspan=[0 8.5];

%specify right hand side
%(attention: output needs to be a 4x1 vector!)
rhs=@(t,y) [y(3:4);[0;-g]-rho/m*norm(y(3:4))*y(3:4)];

%carry out integration using ode45
[tout,yout]=ode45(rhs,tspan,y0);

%plot the trajectory (in green)
plot(yout(:,1),yout(:,2),'g');
legend('alpha=20','alpha=40');
```

The resulting plot is given in Figure 1.1.

(1.2c) Develop a MATLAB function

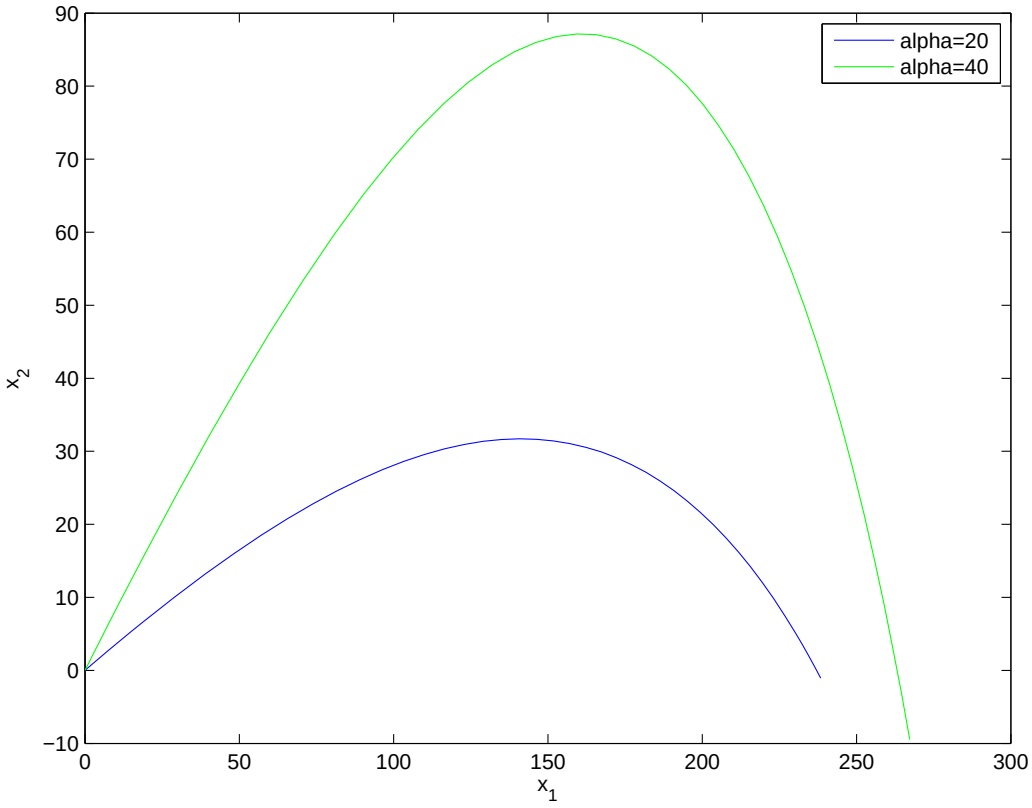

Figure 1.1: Trajectory of the projectile at firing angle $20°$ (blue) and $40°$ (red), respectively.

`function x1 = cannon_hit(alpha)`

which determines the x_1-coordinate of the point of impact of the projectile as a function of the firing angle α, using the MATLAB functions `ode45` and `odeset('Events',...)`.

Solution: We make use of the event mechanism of MATLAB-ODE-solvers to detect the impact of the projectile on the ground. We have impact if the x_2-coordinate of the projectile attains the value zero. We therefore return this coordinate as `value` inside the event function. `isterminal` is set back to 1, so that the integration is aborted after impact (the subsequent development of the trajectory would only be hypothetical anyways). Since the projectile's x_2-coordinate changes from positive to negative at the moment of impact, we set `direction = -1`.

The complete code is contained in `cannon_hit.m`.

Listing 1.2: Determine the x_1-coordinate of the projectile's point of impact as a function of α

```
function x1 = cannon_hit(alpha)

% given data
v0 = 100;
g = 10;
m = 20;
rho = 0.1;

%define right hand side
odefun = @(tt, xx) [xx(3:4); -rho/m*norm(xx(3:4))*xx(3:4) +
   [0; -g]];

% specify integration interval
tspan = [0 100];

% determine initial conditions
x0 = [0; 0; v0*cos(alpha); v0*sin(alpha)];

% activate event-function
opts = odeset('Events', @cannon_event);

% carry out integration
[t, x, te, xe, ie] = ode45(odefun, tspan, x0, opts);

% return point of impact
x1 = xe(1);

function [value, isterminal, direction] = cannon_event(t, y)

% return height above ground of projectile
value = y(2);

% stop integration, if height = 0 is attained
isterminal = 1;
```

```
34
35 % specify direction through zero
36 direction = -1;
```

(1.2d) We wish to determine the firing angle $20° < \alpha^* < 40°$ for which the projectile will hit the above mentioned target. Formulate this problem in terms of a root-finding problem and solve it numerically by implementing a secant method in the MATLAB-function

`function alpha = cannon_angle.`

Solution: See `cannon_angle.m`.

Listing 1.3: Determine the firing angle α

```
1  function alpha=cannon_angle
2  %determine firing angle for which
3  %the target is hit
4
5  %Student:
6
7  %set initial values for secant method
8  %careful: the values eventually need
9  %to be written in radiants!
10 alpha(1)=20*2*pi/360;
11 alpha(2)=40*2*pi/360;
12 alpha(3)=0;
13
14 %write a function for the secant method
15 f=@(alpha) 250-cannon_hit(alpha);
16
17 %stops the while-loop as soon as the absolute
18 %value of the function 'f' is smaller than 1e-5
19 while abs(f(alpha(2)))>1e-5
20     %carry out one step of the secant method
21     alpha(3)= alpha(2)-f(alpha(2))*(alpha(2)-alpha(1))/...
22         (f(alpha(2))-f(alpha(1)));
23     %update the values
24     alpha(1)=alpha(2);
25     alpha(2)=alpha(3);
26 end
27 %write output in degrees
28 alpha=alpha(2)/2/pi*360;
```

The required firing angle is given by $\alpha = 0.4149$, or $23.77°$.

(1.2e) Show that the energy of the projectile is a conserved quantity if the air resistance is negligible.

Solution: The energy of the projectile is given by

$$E(t) = mg\begin{bmatrix}0\\1\end{bmatrix}\mathbf{x} + \frac{1}{2}m\dot{\mathbf{x}}^2.$$

Differentiation with respect to time and substitution of $\ddot{\mathbf{x}}$ using (1.2.1) with $\rho = 0$ yields

$$\begin{aligned}\frac{d}{dt}E(t) &= mg\begin{bmatrix}0\\1\end{bmatrix}\dot{\mathbf{x}} + \frac{1}{2}m2\dot{\mathbf{x}}\ddot{\mathbf{x}}\\ &= mg\begin{bmatrix}0\\1\end{bmatrix}\dot{\mathbf{x}} + m\dot{\mathbf{x}}\left(g\begin{bmatrix}0\\-1\end{bmatrix}\right) = 0.\end{aligned}$$

Problem 1.3 Cat and Mouse

A cat chases after a mouse in the (x, y)-plane. In doing so, the cat invariably runs directly toward the mouse at a constant speed $v_C = 2$. The mouse on its part tries to escape to its mouse hole at velocity $v_M = 1$. The mouse hole is situated at the point $(0, 1)$. Let the mouse be at the point $(0, 0)$ at time $t = 0$ and let the cat be at the point $(1, 0)$.

(1.3a) Determine the IVP (see [NODE, Def. 1.1.2]) describing the trajectory of the mouse.

Solution: Let $\mathbf{y}_M(t)$ be the position vector of the mouse at time t. Then $\mathbf{y}_M(t)$ satisfies

$$\dot{\mathbf{y}_M}(t) = \begin{pmatrix}0\\1\end{pmatrix}, \qquad \mathbf{y}_M(0) = \begin{pmatrix}0\\0\end{pmatrix}.$$

(1.3b) Determine the IVP describing the trajectory of the cat. What is the augmented state space Ω in this case (see [NODE, Def. 1.1.1])?

Solution: Let $\mathbf{y}_C(t)$ be the position vector of the cat at time t. Then $\mathbf{y}_C(t)$ satisfies

$$\dot{\mathbf{y}}_C(t) = 2 \cdot \frac{\mathbf{y}_M(t) - \mathbf{y}_C(t)}{\|\mathbf{y}_M(t) - \mathbf{y}_C(t)\|}, \qquad \mathbf{y}_C(0) = \begin{pmatrix}1\\0\end{pmatrix}.$$

For the state space we define the following continuous function

$$\begin{aligned} f: \ [0\,1] \times \mathbb{R} \times \mathbb{R} &\to \mathbb{R}\\ (t, x, y) &\mapsto x^2 + (t - y)^2.\end{aligned}$$

Let $f^{-1}(0)$ be the pre image of the origin under f, then the augmented state space is given by $\Omega = [0\,1] \times \mathbb{R}^2 \setminus f^{-1}(0)$.

(1.3c) Write a MATLAB function

```
function trajectories
```

which calculates the evolution of the trajectories of both animals up to time $T = 0.5$ using the MATLAB function `ode45`. Plot the trajectories in the same figure.

Solution: The trajectories can be calculated using the following code. The resulting plot is shown in Figure 1.2.

Listing 1.4: Calculate and plot the trajectories of both animals.

```
function trajectories

% initial value for mouse
y_M_0=[0;0];

% right hand side for mouse
y_M_Abl=@(t,x) [0;1];

% solve mouse-IVP (initial value problem)
[t_M,y_M] = ode45(y_M_Abl,[0 0.5],y_M_0);

% plot mouse's trajectory
figure;
hold on; % allows multiple plots in same figure
plot(y_M(:,1),y_M(:,2),'r*'); % 'r' is the color red

% initial value for cat
y_K_0=[1;0];

% analytical trajectory of mouse
y_M_analytic=@(t) [0;0] + t.*[0; 1];

% right hand side for cat
y_K_Abl=@(t,x) 2*(y_M_analytic(t)-x)/norm(y_M_analytic(t)-x);

% solve cat-IVP
[t_K,y_K] = ode45(y_K_Abl,[0 0.5],y_K_0);

% plot trajectory of cat
plot(y_K(:,1),y_K(:,2),'b*'); % 'b' stands for blue
grid on;
legend('Mouse trajectory', 'Cat trajectory');
title('Trajectories of animals for t in [0 0.5]')
return;
```

(1.3d) Determine the IVP which describes the trajectory of the displacement vector between the mouse and the cat. Write a MATLAB function

```
function te = rendezvous
```

which calculates whether, and when, the cat and the mouse have approached each other up to 10^{-5} using the MATLAB function `ode45` and `odeset('Events', ...)`.

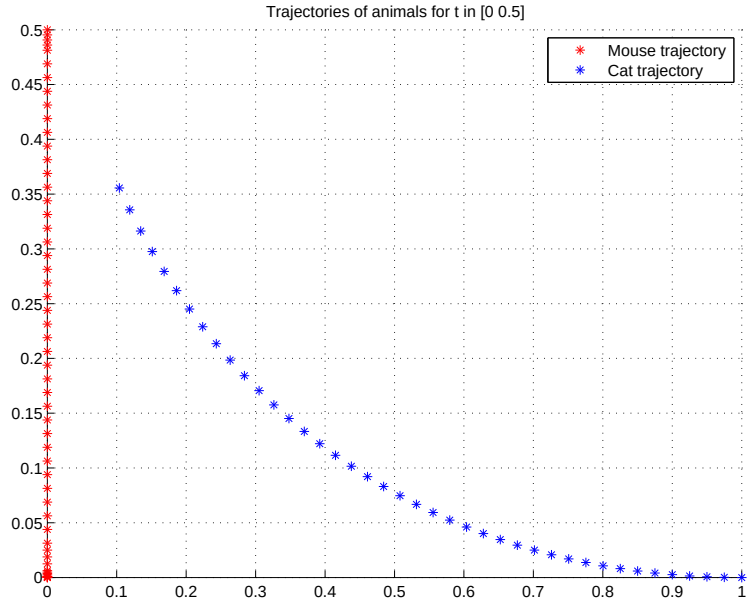

Figure 1.2: Trajectories of both animals up to time $T = 0.5$.

Solution: The displacement vector is given by $\mathbf{r}(t) = \mathbf{y}_M(t) - \mathbf{y}_C(t)$. Thus

$$\begin{aligned}\dot{\mathbf{r}}(t) &= \dot{\mathbf{y}}_M(t) - \dot{\mathbf{y}}_C(t) \\ &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 2 \cdot \frac{\mathbf{y}_M(t) - \mathbf{y}_C(t)}{\|\mathbf{y}_M(t) - \mathbf{y}_C(t)\|} \\ &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 2 \cdot \frac{\mathbf{r}(t)}{\|\mathbf{r}(t)\|}.\end{aligned}$$

For the initial value we obtain

$$\mathbf{r}(0) = \mathbf{y}_M(0) - \mathbf{y}_C(0) = \begin{pmatrix} -1 \\ 0 \end{pmatrix}.$$

The meeting time can be calculated using the following code. The function `meeting(t,x)` aborts the integration as soon as the target function $\|x\| - 10^{-5}$ crosses zero from positive to negative.

Listing 1.5: Calculates the time of rendezvous.

```
1  function te = rendezvous
2
3  % intitial value of displacement vector
4  y_r_0=[-1;0];
5
6  % right hand side for displacement vector
7  y_r_Abl=@(t,x) [0;1]-2*x/norm(x);
8
9  % activate event function; definition of function 'meeting'
10 % follows on line 18
```

```
opts = odeset('Events', @meeting);

% solve the IVP using opts
[t_r, y_r, te, ye, ie] = ode45(y_r_Abl, [0 1], y_r_0, opts);

end

function [value, isterminal, direction] = meeting(t, x)
% as soon as the length of the displacement vector
% is less than 1e-5, tell ode45 to terminate integration

% calculate deviation of distance from 1e-5
value = norm(x) - 1e-5;

% terminate integration when event occurs
isterminal = 1;

% specify direction of zero-crossing
direction =-1;
end
```

Published on 24 February 2016.

To be submitted by 1 March 2016.

References

[NUMODE] Lecture Slides for the course "Numerical Methods for Ordinary Differential Equations", SVN revision # 63606.

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

Last modified on February 29, 2016