Problem Sheet 11

Problem 11.1 Implicit Euler Method for DAE of Order 1

We consider the following DAE

$$\begin{cases} \dot{x}_1 &= x_3\,, \\ \dot{x}_2 &= x_4\,, \\ \dot{x}_3 &= -2x_1x_5\,, \\ \dot{x}_4 &= -2x_2x_5 + 9\,, \\ 0 &= x_3^2 + x_4^2 - 2x_5 + 9x_2\,. \end{cases} \tag{11.1.1}$$

with initial conditions

$$x_1(0) = 1\,, \quad x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0.$$

(11.1a) Check if the initial conditions are consistent with the algebraic equation.([NUMODE, Def. 3.8.7])

Solution:

$$x_3(0)^2 + x_4(0)^2 - 2x_5(0) + 9x_2(0) = 0^2 + 0^2 - 2\cdot 0 + 9\cdot 0 = 0.$$

(11.1b) Show that, the DAE (11.1.1) has index 1 and state the equivalent ODE explicitly.

Solution: The DAE has index 1, because

$$\frac{d}{dx_5}\big(x_3^2 + x_4^2 - 2x_5 + 9x_2\big) = -2 \neq 0.$$

The equivalent ODE can be determined by a simple substitution

$$x_5 = \frac{1}{2}\big(x_3^2 + x_4^2 + 9x_2\big)$$

in (11.1.1).

(11.1c) Apply the implicit Euler method to (11.1.1) and explicitly state the system of equations.

Solution:

$$\begin{cases} x_1^{(1)} &= x_1^{(0)} + hx_3^{(1)}\,, \\ x_2^{(1)} &= x_2^{(0)} + hx_4^{(1)}\,, \\ x_3^{(1)} &= x_3^{(0)} - 2hx_1^{(1)}x_5^{(1)}\,, \\ x_4^{(1)} &= x_4^{(0)} - 2hx_2^{(1)}x_5^{(1)} + 9h\,, \\ 0 &= \left(x_3^{(1)}\right)^2 + \left(x_4^{(1)}\right)^2 - 2x_5^{(1)} + 9x_2^{(1)}\,. \end{cases}$$

(11.1d) Complete the MATLAB-Template `firstDAE.m`, in which the system of equations from subproblem (11.1c) is solved using `nNewton` iterations of the Newton method. Apply the function for different numbers of time steps `N` and different values of `nNewton`. Plot the evolution of the algebraic equations. What do you observe?

Solution: We observe that, even if the numerical approximation is qualitatively wrong (see for example the evolution of x_5 in Figure 11.1), given enough Newton iterations, the numerical approximation stays on the manifold (see the axis scales in Figure 11.2, Figure 11.3 and Figure 11.4).

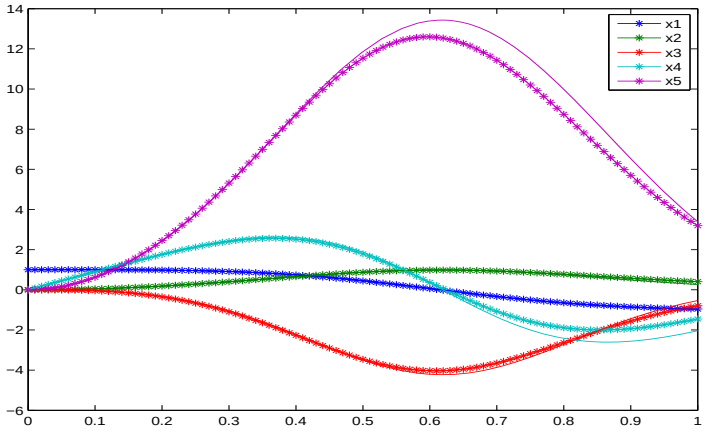

Figure 11.1: Evolution of the components with `N=100` and `nNewton=3`.

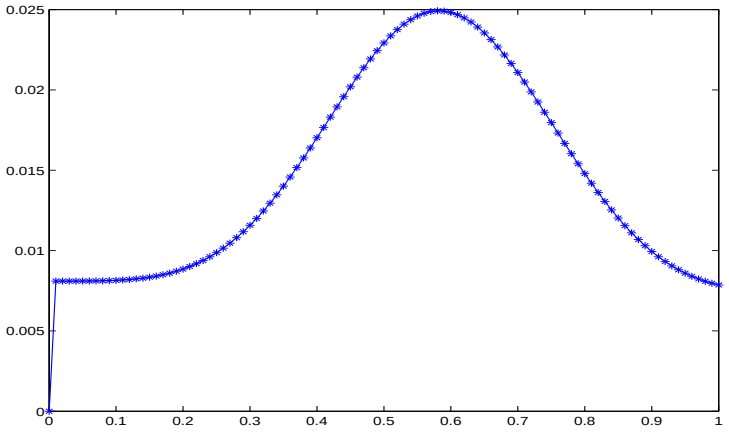

Figure 11.2: Evolution of the algebraic conditions with `N=100` and `nNewton=1`.

Listing 11.1: Implicit Euler method for DAE of index 1

```
%Number of steps
N=100;

%Number of Newton iterations
nNewton=1;
```

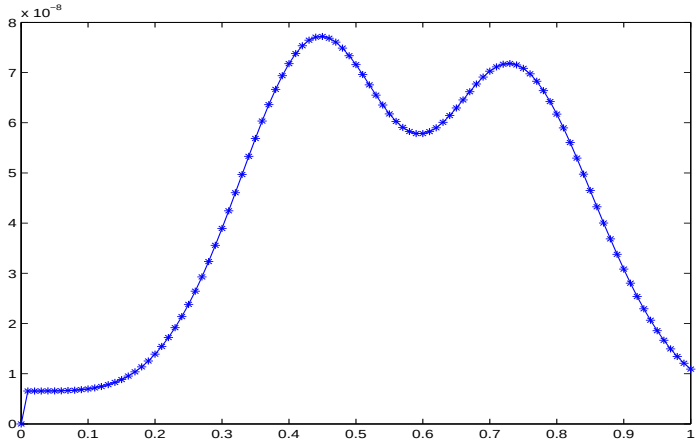

Figure 11.3: Evolution of the algebraic conditions with `N=100` and `nNewton=2`.

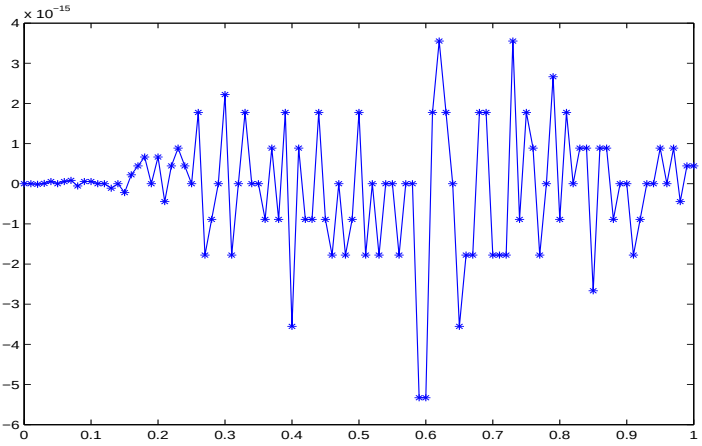

Figure 11.4: Evolution of the algebraic conditions with `N=100` and `nNewton=3`.

```
6
7   %End point
8   T=1;
9
10  %Step size
11  h=T/N;
12
13  %algebraic condition function handle
14  c= @(y) y(3)^2+y(4)^2-2*y(5)+9*y(2);
15
16  %root-search formulation of the implicit Euler method
17  F= @(y,y0) [y(1)-y0(1)-h*y(3);...
18              y(2)-y0(2)-h*y(4);...
19              y(3)-y0(3)+2*h*y(1)*y(5);...
20              y(4)-y0(4)+2*h*y(2)*y(5)-9*h;...
21              c(y)];
```

```
22
23 DF = @(y) [1 0 -h 0 0;
24             0 1 0 -h 0;
25             2*h*y(5) 0 1 0 2*h*y(1);
26             0 2*h*y(5) 0 1 2*h*y(2);
27             0 9 2*y(3) 2*y(4) -2];
28
29 %initial value
30 y=[1; 0; 0; 0; 0];
31
32 %distance to the manifold
33 ZB=zeros(N+1,1);
34 ZB(1)=c(y);
35
36 %values of the different components
37 X=zeros(N+1,5);
38 X(1,:)=y';
39
40 for ii=1:N
41
42     y0=y;
43
44     for jj=1:nNewton
45         y=y-DF(y)\F(y,y0);
46     end
47
48     %values of the different components
49     X(ii+1,:)=y';
50
51     %distance from the manifold
52     ZB(1+ii)=c(y);
53 end
54
55 fprintf('Maximal distance from the manifold: %e\n',
      max(abs(ZB)));
56
57 %Plot of the algebraic conditions evolution
58 t=0:h:N*h;
59 figure;
60 plot(0:h:N*h, ZB, '*-');
61
62 %Plot of the component evolution
63 figure;
64 plot(t, X(:,1), '*-',t, X(:,2), '*-',t, X(:,3), '*-',t,
      X(:,4), '*-',t, X(:,5), '*-');
65 hold on;
66
```

```
%Referential solution from ode23t
f = @(t,y) [y(3);...
            y(4);...
            -2*y(1)*y(5);...
            -2*y(2)*y(5)+9;...
            y(3)^2+y(4)^2-2*y(5)+9*y(2)];
J = @(t,y) [0 0 1 0 0;...
            0 0 0 1 0;...
            -2*y(5) 0 0 0 -2*y(1);...
            0 -2*y(5) 0 0 -2*y(2);...
            0 9 2*y(3) 2*y(4) -2];
M=eye(5);
M(5,5)=0;
M = @(t,y) M;
opts = odeset('Mass',M,'Jacobian',J);
y0=[1;0;0;0;0];
[t,y] = ode23t(f,[0 N*h],y0,opts);

%figure;
plot(t,y(:,1),t,y(:,2),t,y(:,3),t,y(:,4),t,y(:,5));
```

Problem 11.2 Mechanical Arm

This problem is long and complex. It would be great if you could try your best and finish as much as you can.

A mechanical arm is built up from $n > 1$ arms of length $\ell = 1$ (see Figure 11.5). The movement of the mechanical arm is described by the change in the angle θ_i (with respect to the horizontal direction) of its arms.

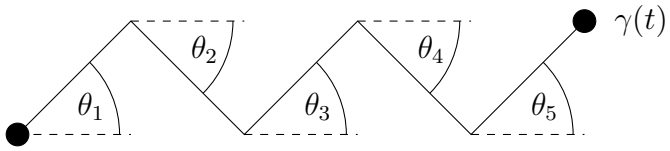

Figure 11.5: Mechanical Arm

(11.2a) Write a function $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^2$, which gives the position of the end of the arm as a function of the angles $\theta_1, \cdots, \theta_n$.

Solution: One such function is

$$\mathbf{g}(\theta_1, \theta_2, \ldots, \theta_n) = \sum_{j=1}^{n} \begin{pmatrix} \cos\theta_j \\ \sin\theta_j \end{pmatrix}.$$

Now let $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^\top$. A natural restriction is that the arm moves along a given path $\boldsymbol{\gamma} : [0, T] \to \mathbb{R}^2$. We want to rewrite the problem as a DAE.

(11.2b) The algebraic constraint for this problem is $\mathbf{g}(\boldsymbol{\theta}) = \gamma(t)$. Prove that when $n > 2$, i.e. there are more than 2 arms, the solution $\boldsymbol{\theta}$ to constraint equation cannot be unique. We will take the one with the smallest Euclidean norm $\|\dot{\boldsymbol{\theta}}(t)\|$. Prove that in this case $\dot{\boldsymbol{\theta}} \in \operatorname{Im} \mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta})^T$, so there exists $\lambda \in \mathbb{R}^2$ such that $\dot{\boldsymbol{\theta}} = \mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta})^T\lambda$.

Solution: Taking differentiation on both sides, we get

$$\mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta}(t))\dot{\boldsymbol{\theta}}(t) = \dot{\gamma}(t) \tag{11.2.1}$$

We have n unknown variables $\theta_1, \cdots, \theta_n$ with only 2 constraints, so the equation is underdetermined when $n > 2$.

Keep in mind that $\boldsymbol{\theta} \in \mathbb{R}^{n\times 1}$, $\mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta}) \in \mathbb{R}^{2\times n}$. For any fixed t, any solution $\boldsymbol{\theta}$, decompose $\dot{\boldsymbol{\theta}}$ in the form of $\mathbf{p} + \mathbf{q}$, where $\mathbf{p} \in \operatorname{Ker} \mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta}(t))$, $\mathbf{q} \in \operatorname{Im} \mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta}(t))^T$. Then $\mathbf{q}$ is also a solution to (11.2.1), and since $\operatorname{Im} \mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta}(t))^T \perp \operatorname{Ker} \mathrm{D}_{\boldsymbol{\theta}}\mathbf{g}(\boldsymbol{\theta}(t))$, we have $\|\mathbf{q}\| \leqslant \|\mathbf{p} + \mathbf{q}\| = \|\dot{\boldsymbol{\theta}}\|$.

As a result of previous problem, we have the following DAE equivalent to the equation we were discussing

$$\dot{\boldsymbol{\theta}} = \mathbf{d}(\boldsymbol{\theta}, \boldsymbol{\lambda}), \tag{11.2.2}$$
$$0 = c(t, \boldsymbol{\theta}, \boldsymbol{\lambda}), \tag{11.2.3}$$

with

$$c(t, \boldsymbol{\theta}, \boldsymbol{\lambda}) := \mathbf{g}(\boldsymbol{\theta}) - \boldsymbol{\gamma}(t),$$
$$\mathbf{d}(\boldsymbol{\theta}, \boldsymbol{\lambda}) := \mathrm{D}_{\boldsymbol{\theta}}c(t, \boldsymbol{\theta})^\top \boldsymbol{\lambda},$$

and $\boldsymbol{\lambda} \in \mathbb{R}^2$. Since c is independent of $\boldsymbol{\lambda}$, we may just write $c(t, \boldsymbol{\theta})$ instead of $c(t, \boldsymbol{\theta}, \boldsymbol{\lambda})$.

(11.2c) Derive the DAE (11.2.2) for $n = 5$ and

$$\boldsymbol{\gamma}(t) := \begin{pmatrix} 4 + \frac{4}{5}\sin^3(t) \\ \frac{13}{16}\cos(t) - \frac{5}{16}\cos(2t) - \frac{1}{8}\cos(3t) - \frac{1}{16}\cos(4t) - \frac{5}{16} \end{pmatrix}.$$

Solution: We have

$$c(t, \boldsymbol{\theta}) = \sum_{j=1}^{5} \begin{pmatrix} \cos\theta_j \\ \sin\theta_j \end{pmatrix} - \begin{pmatrix} 4 + \frac{4}{5}\sin^3(t) \\ \frac{13}{16}\cos(t) - \frac{5}{16}\cos(2t) - \frac{1}{8}\cos(3t) - \frac{1}{16}\cos(4t) - \frac{5}{16} \end{pmatrix}.$$

Hence

$$\mathrm{D}_{\boldsymbol{\theta}}c(t, \boldsymbol{\theta}) = \begin{pmatrix} -\sin(\theta_1) & -\sin(\theta_2) & -\sin(\theta_3) & -\sin(\theta_4) & -\sin(\theta_5) \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) & \cos(\theta_4) & \cos(\theta_5) \end{pmatrix}.$$

Consequently

$$\dot{\boldsymbol{\theta}} = \begin{pmatrix} -\sin(\theta_1) & -\sin(\theta_2) & -\sin(\theta_3) & -\sin(\theta_4) & -\sin(\theta_5) \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) & \cos(\theta_4) & \cos(\theta_5) \end{pmatrix}^\top \boldsymbol{\lambda},$$
$$0 = \sum_{j=1}^{5} \begin{pmatrix} \cos\theta_j \\ \sin\theta_j \end{pmatrix} - \begin{pmatrix} 4 + \frac{4}{5}\sin^3(t) \\ \frac{13}{16}\cos(t) - \frac{5}{16}\cos(2t) - \frac{1}{8}\cos(3t) - \frac{1}{16}\cos(4t) - \frac{5}{16} \end{pmatrix},$$

(11.2d) Write a straightforward MATLAB function `robotsolve1.m` that approximates the solution of the DAE (11.2c) using the MATLAB integrator `ode23t` (with default tolerance) on the interval $[0, 2\pi]$. Use the initial condition

$$\big(\boldsymbol{\theta}(0), \boldsymbol{\lambda}(0)\big)^\top = (0, 0, 0, \frac{\pi}{3}, -\frac{\pi}{3}, 0, 0)^\top.$$

Why (and where) does this approach fail?

Solution: See implementation Listing 11.2. We notice that the ODE solver `ode23t` breaks off the calculation with the error message saying that the index of our DAE is > 1.

Listing 11.2: Solve (11.2c) with `ode23t`

```
function robotsolve1

% set integration interval
tspan = [0, 2*pi];

% initial value
y0=[0;0;0;pi/3;-pi/3;0;0];

%arm end position function
g=@(theta) sum([cos(theta)';sin(theta)'],2);

%path function of the arm
gamma=@(t) [4+4/5*sin(t).^3;...
     13/16*cos(t)-5/16*cos(2*t)-1/8*cos(3*t)-1/16*cos(4*t)-5/16];

% right hand side
Dc=@(theta) [-sin(theta)';cos(theta)'];

fun = @(t,y) [Dc(y(1:5))'*y(6:7); g(y(1:5))-gamma(t)];

% construct mass matrix
M = vertcat([eye(5),zeros(5,2)],zeros(2,7));

% call-up ode23t
opts = odeset('Mass', M);
[t,y] = ode23t(fun, tspan, y0, opts);

%compute the arm's exact path
exact_path=gamma(t');

%compute the approximated path as well as the
%distance between the approximated and the exact path
approx_path=zeros(2,length(t));
err=zeros(1,length(t));
for i=1:length(t)
    approx_path(:,i)=g(y(i,1:5)');
```

```
    err(i)=norm(g(y(i,1:5)')-gamma(t(i)));
end

%plot of the exact and approximated paths
figure;
plot(exact_path(1,:),
   exact_path(2,:),'b',approx_path(1,:),approx_path(2,:),'ro');
legend('exact path','approximated path');

%Plot of the paths distance
figure;
plot(t,err,'*-');
xlabel('time');
ylabel('error');
title('distance between the exact and approximated path');
```

(11.2e) Write a MATLAB function `robotimpe.m` that approximates the solution of the DAE (11.2c) using `N`=100 steps of the implicit Euler method on the interval $[0, 2\pi]$. Use the initial conditions

$$\big(\boldsymbol{\theta}(0), \boldsymbol{\lambda}(0)\big)^\top = (0, 0, 0, \frac{\pi}{3}, -\frac{\pi}{3}, 0, 0)^\top.$$

Your MATLAB function should also plot the exact path or the arm $\boldsymbol{\gamma}(t)$ and the approximated path of the arm $\mathbf{g}\big(\theta(t)\big)$ as well as the distance between the two. What do you observe?

HINT: Implicit means that $(\boldsymbol{\theta}_{k+1}, \boldsymbol{\lambda}_{k+1})^\top$ is calculated using one Newton iteration with initial value $(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k)^\top$.

Solution: Let $\boldsymbol{\theta}_1 = \boldsymbol{\theta}(t_0 + h)$, $\boldsymbol{\theta}_0 = \boldsymbol{\theta}(0)$. The implicit Euler method applied to (11.2.2) gives

$$\begin{pmatrix} \boldsymbol{\theta}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\theta}_0 + h\mathbf{d}(\boldsymbol{\theta}_1, \boldsymbol{\lambda}_1) \\ c(t_0 + h, \boldsymbol{\theta}_1) \end{pmatrix}.$$

This implies that $(\boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ is a root of the following function

$$\mathbf{F}(\boldsymbol{\theta}_1, \boldsymbol{\lambda}_1) := \begin{pmatrix} \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0 - h\mathbf{d}(\boldsymbol{\theta}_1, \boldsymbol{\lambda}_1) \\ c(t_0 + h, \boldsymbol{\theta}_1) \end{pmatrix}.$$

One Newton iteration results in

$$\begin{pmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\lambda}_1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\theta}_0 \\ \boldsymbol{\lambda}_0 \end{pmatrix} - \big(\mathrm{D}\mathbf{F}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_0)\big)^{-1} * \mathbf{F}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_0),$$

with

$$\mathrm{D}\mathbf{F}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_0) := \begin{pmatrix} \mathbf{I} - h\mathrm{D}_{\boldsymbol{\theta}}\mathbf{d}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_0) & -h\mathrm{D}_{\boldsymbol{\lambda}}\mathbf{d}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_0) \\ \mathrm{D}_{\boldsymbol{\theta}}c(t_0 + h, \boldsymbol{\theta}_1) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I} \\ \mathrm{D}_{\boldsymbol{\theta}}c(t_0 + h, \boldsymbol{\theta}_1) \end{pmatrix} - \begin{pmatrix} h\mathrm{D}_{\boldsymbol{\theta}}\mathbf{d}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_0) \\ 0 \end{pmatrix}.$$

See Listing 11.3, Figure 11.6 and Figure 11.7. We observe that, the solution is qualitatively correct even though the constraints are not satisfied any more.

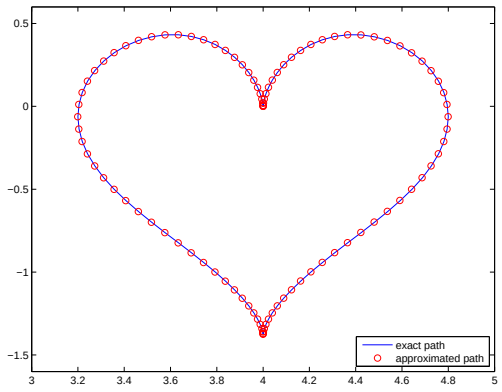

Figure 11.6: Exact and approximated paths

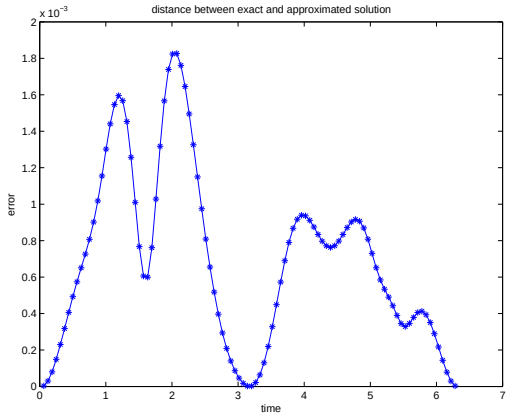

Figure 11.7: Distance between the exact and approximated paths

Listing 11.3: Implicit Euler for (11.2c)

```
function robotimpe
close all;

% set number of iterations
N=100;

% set integration interval
tspan = [0, 2*pi];

% compute stepsize
h=tspan(2)/N;

% set initial value
y0=[0;0;0;pi/3;-pi/3;0;0];

%arm end position function
g=@(theta) sum([cos(theta)';sin(theta)'],2);
```

```
18
19  %arm's path function
20  gamma=@(t) [4+4/5*sin(t).^3;...
21       13/16*cos(t)-5/16*cos(2*t)-1/8*cos(3*t)-1/16*cos(4*t)-5/16];
22
23  % right hand side
24  Dc=@(theta) [-sin(theta)';cos(theta)'];
25
26  F = @(t,y) [-h*Dc(y(1:5))'*y(6:7);...
27                g(y(1:5))-gamma(t)];
28
29  %derivation of the right hand side for Newton step
30  DF=@(y) vertcat([eye(5), zeros(5,2)]
       +h*[diag(y(6)*cos(y(1:5))+y(7)*sin(y(1:5))),sin(y(1:5)),-cos(y(1:5))],.
31      [Dc(y(1:5)),zeros(2,2)]) ;
32
33  %Compute the evolution of y as well as the approximated path
34  % and its distance to the exact path
35  y=y0;
36  err=zeros(1,N);
37  approx_path=zeros(2,N+1);
38  approx_path(:,1)=g(y0(1:5));
39  for i=1:N
40      %compute the new value by a Newton step
41      y=y-DF(y)\F(i*h,y);
42      %compute approximated path
43      approx_path(:,i+1)=g(y(1:5));
44  % and its distance to the exact path
45      err(i)=norm(g(y(1:5))-gamma(h*i));
46
47  end
48  % compute the path's exact path
49  exact_path=gamma(0:h:2*pi);
50
51  % plot of the exact and approximated path
52  figure;
53  plot(exact_path(1,:),
       exact_path(2,:),'b',approx_path(1,:),approx_path(2,:),'ro');
54  xlim([3 5]);
55  ylim([-1.6 0.6]);
56  legend('exact path','approximated
       path','Location','SouthEast');
57
58  % Plot the distance between the paths
59  figure;
60  plot(h:h:2*pi,err,'*-');
61  xlabel('time');
```

```
62 ylabel('error');
63 title('distance between exact and approximated solution');
```

(11.2f) Transform (11.2.2) into an equivalent DAE of index 1 by differentiating c and $\mathbf{d}$ from (11.2c) . Prove that the transformed DAE has index 1.

Solution: By differentiating the constraints once with respect to time we get

$$\begin{aligned}\tilde{c}(t,\boldsymbol{\theta}) &:= \frac{d}{dt}c(t,\boldsymbol{\theta})\\ &= \frac{d}{dt}(\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\gamma}(t))\\ &= \mathrm{D}_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} - \dot{\gamma}(t).\end{aligned}$$

The new constraint satisfies

$$\mathrm{D}_{\boldsymbol{\lambda}}\tilde{c}(t,\boldsymbol{\theta}) = \mathrm{D}_{\boldsymbol{\theta}}\boldsymbol{g}(t,\boldsymbol{\theta})\mathrm{D}_{\boldsymbol{\theta}}\boldsymbol{g}(t,\boldsymbol{\theta})^\top = \begin{pmatrix} \sum_{j=1}^5 \sin^2(\theta_j) & -\sum_{j=1}^5 \sin(\theta_j)\cos(\theta_j) \\ -\sum_{j=1}^5 \sin(\theta_j)\cos(\theta_j) & \sum_{j=1}^5 \cos^2(\theta_j)\end{pmatrix}.$$

We have

$$\det(\mathrm{D}_{\boldsymbol{\lambda}}\tilde{c}(t,\boldsymbol{\theta})) = \sum_{j=1}^5 \sin^2(\theta_j)\sum_{j=1}^5 \cos^2(\theta_j) - \left(\sum_{j=1}^5 \sin(\theta_j)\cos(\theta_j)\right)^2 e.$$

Therefore, by the Cauchy-Schwarz inequality we have $\det(\mathrm{D}_{\boldsymbol{\lambda}}\tilde{c}(t,\boldsymbol{\theta})) \geq 0$ where the equality holds if and only if there exists a $\mu \in \mathbb{R}$ such that $\sin(\theta_j) = \mu\cos(\theta_j)$ for all $j = 1,\dots,5$. Let us assume that indeed is the case. Plugging in $\sum_{j=1}^5 \sin(\theta_j) = \mu\sum_{j=1}^5 \cos(\theta_j)$ into (11.2.3) yields

$$4 + \frac{4}{5}\sin^3(t) = \mu\left(\frac{13}{16}\cos(t) - \frac{5}{16}\cos(2t) - \frac{1}{8}\cos(3t) - \frac{1}{16}\cos(4t) - \frac{5}{16}\right).$$

Evaluating the last expression at $t = 0$ gives $\mu = \frac{1}{64}$, while $t = \frac{\pi}{2}$ gives $\mu = \frac{25}{384}$, which is a contradiction.

Thus, $\mathrm{D}_{\boldsymbol{\lambda}}\tilde{c}(t,\boldsymbol{\theta})$ is always invertible. The DAE of index 1, equivalent to (11.2.2), is therefore

$$\begin{aligned}\dot{\boldsymbol{\theta}} &= \mathrm{D}_{\boldsymbol{\theta}}c(t,\boldsymbol{\theta})^\top\boldsymbol{\lambda},\\ 0 &= \mathrm{D}_{\boldsymbol{\theta}}c(t,\boldsymbol{\theta})\mathrm{D}_{\boldsymbol{\theta}}c(t,\boldsymbol{\theta})^\top\boldsymbol{\lambda} - \dot{\boldsymbol{\gamma}}(t).\end{aligned}$$

(11.2g) Write a MATLAB function `robotsolve2.m` that approximates the solution of the DAE subproblem (11.2f) using the MATLAB-integrator `ode23t` (default tolerance) on the interval $[0, 2\pi]$. Use the initial conditions

$$\big(\boldsymbol{\theta}(0), \boldsymbol{\lambda}(0)\big)^\top = \Big(0, 0, 0, \frac{\pi}{3}, -\frac{\pi}{3}, 0, 0\Big)^\top.$$

Your MATLAB-script should also plot the exact path or the arm $\boldsymbol{\gamma}(t)$ and the approximated path of the arm $\mathrm{g}\big(\theta(t)\big)$ as well as the distance between the two. What do you observe?

Solution: See Listing 11.4. We observe that, the ODE-solver now gives a solutions which is qualitatively correct but does not satisfy the constraint any more.

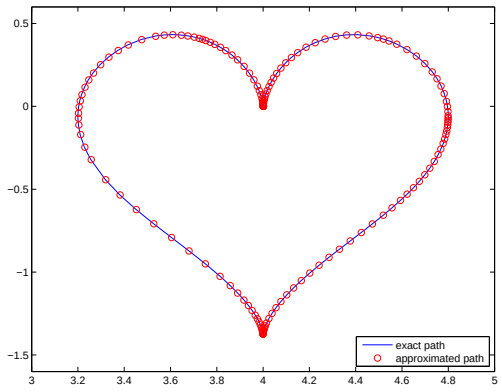

Figure 11.8: Exact and approximated path

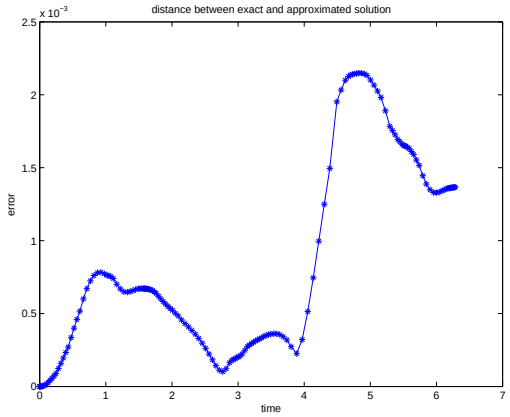

Figure 11.9: Distance between the exact and approximated path

Listing 11.4: Solve subproblem (11.2f) using `ode23t`

```
function robotsolve2
close all;

% set integration interval
tspan = [0, 2*pi];

% initial value
y0=[0;0;0;pi/3;-pi/3;0;0];

%arm end position function
g=@(theta) sum([cos(theta)';sin(theta)'],2);

%path function of the arm
gamma=@(t) [4+4/5*sin(t).^3;...
     13/16*cos(t)-5/16*cos(2*t)-1/8*cos(3*t)-1/16*cos(4*t)-5/16];

% right hand side
```

```
Dc=@(theta) [-sin(theta)';cos(theta)'];

gammadot=@(t) [ 12/5*sin(t).^2*cos(t);...
                  -13/16*sin(t)+5/8*sin(2*t)+3/8*sin(3*t)+1/4*sin(4*t)];

fun = @(t,y) [Dc(y(1:5))'*y(6:7);...
               Dc(y(1:5))*Dc(y(1:5))'*y(6:7)-gammadot(t)];

% construct mass matrix
M = vertcat([eye(5),zeros(5,2)],zeros(2,7));

% call up ode23t
opts = odeset('Mass', M);
[t,y] = ode23t(fun, tspan, y0, opts);

%compute the arm's exact path
exact_path=gamma(t');

%compute the approximated path as well as the
%distance between the approximated and the exact path
approx_path=zeros(2,length(t));
err=zeros(1,length(t));
for i=1:length(t)
    approx_path(:,i)=g(y(i,1:5)');
    err(i)=norm(g(y(i,1:5)')-gamma(t(i)));
end

% plot of the exact and approximated path
figure;
plot(exact_path(1,:),
   exact_path(2,:),'b',approx_path(1,:),approx_path(2,:),'ro');
xlim([3 5]);
ylim([-1.6 0.6]);
legend('exact path','approximated
   path','Location','SouthEast');

% Plot the distance between the paths
figure;
plot(t,err,'*-');
xlabel('time');
ylabel('error');
title('distance between exact and approximated solution');
```

(11.2h) Show that, the DAE (11.2.2) is equivalent to the following ODE

$$\dot{\boldsymbol{\theta}} = \mathrm{D}_{\boldsymbol{\theta}} c(t,\boldsymbol{\theta})^{\top} \left(\mathrm{D}_{\boldsymbol{\theta}} c(t,\boldsymbol{\theta}) \mathrm{D}_{\boldsymbol{\theta}} c(t,\boldsymbol{\theta})^{\top}\right)^{-1} \dot{\boldsymbol{\gamma}}(t). \tag{11.2.4}$$

Solution: In subproblem (11.2f) we showed that

$$\mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta}) \mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta})^{\top} \boldsymbol{\lambda} - \dot{\boldsymbol{\gamma}}(t) = 0,$$

and because $\mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta}) \mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta})^{\top}$ is invertible it follows that

$$\boldsymbol{\lambda} = \left(\mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta}) \mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta})^{\top}\right)^{-1} \dot{\boldsymbol{\gamma}}(t).$$

Plugging this in the first equation of (11.2.2) results in

$$\dot{\boldsymbol{\theta}} = \mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta})^{\top} \left(\mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta}) \mathrm{D}_{\boldsymbol{\theta}} c(t, \boldsymbol{\theta})^{\top}\right)^{-1} \dot{\boldsymbol{\gamma}}(t).$$

(11.2i) Write a MATLAB function `robotodesolve.m` that approximates the solution of the ODE (11.2.4) using an appropriate MATLAB integrator on the interval $[0, 2\pi]$. Use the initial conditions

$$\left(\boldsymbol{\theta}(0), \boldsymbol{\lambda}(0)\right)^{\top} = \left(0, 0, 0, \frac{\pi}{3}, -\frac{\pi}{3}, 0, 0\right)^{\top}.$$

Your MATLAB function should also plot the exact path or the arm $\boldsymbol{\gamma}(t)$ and the approximated path of the arm $\mathrm{g}\big(\theta(t)\big)$ as well as the distance between the two. What do you observe?

Solution: Since DAEs are equivalent to stiff ODEs we need to choose the integrator `ode23s` or `ode15s`. See Listing 11.5, Figure 11.10 and Figure 11.11. We observe again that, the solutions are qualitatively correct the constraints however not satisfied.

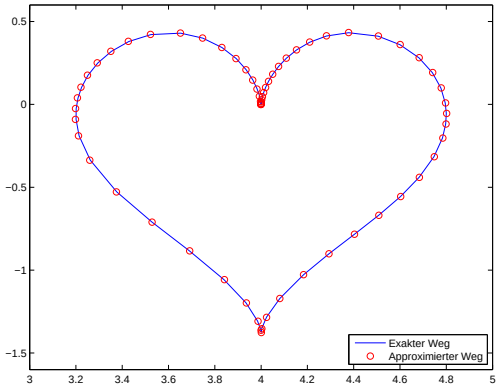

Figure 11.10: Exact and approximated path

Listing 11.5: Solve (11.2.4) using `ode23s`

```
function robotodesolve
close all;

% set integration interval
tspan = [0, 2*pi];

% set initial value
y0=[0;0;0;pi/3;-pi/3];

```

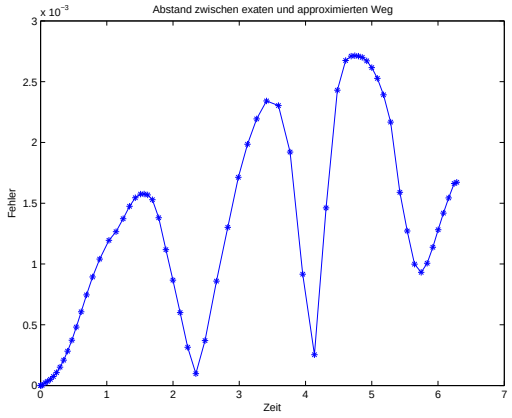

Figure 11.11: Distance between the exact and approximated path

```
%arm end position function
g=@(theta) sum([cos(theta)';sin(theta)'],2);

%goal path function
gamma=@(t) [4+4/5*sin(t).^3;...
     13/16*cos(t)-5/16*cos(2*t)-1/8*cos(3*t)-1/16*cos(4*t)-5/16];

% right hand side
Dc=@(theta) [-sin(theta)';cos(theta)'];

gammadot=@(t) [ 12/5*sin(t).^2*cos(t);...
        -13/16*sin(t)+5/8*sin(2*t)+3/8*sin(3*t)+1/4*sin(4*t)];

fun = @(t,y) [Dc(y)'*((Dc(y)*Dc(y)')\gammadot(t))];

%ode23
[t,y] = ode23s(fun, tspan, y0);

% compute exact path
exact_path=gamma(t');

%plot approximated and exact paths as well as
% the distance between them
approx_path=zeros(2,length(t));
err=zeros(1,length(t));
for i=1:length(t)
    approx_path(:,i)=g(y(i,1:5)');
    err(i)=norm(g(y(i,1:5)')-gamma(t(i)));
end

%Plot exact and approximated paths
figure;
```

```
plot(exact_path(1,:),
   exact_path(2,:),'b',approx_path(1,:),approx_path(2,:),'ro');
xlim([3 5]);
ylim([-1.6 0.6]);
legend('Exact path','Approximated
   path','Location','SouthEast');

%Plot des Abstands des Weges
figure;
plot(t,err,'*-');
xlabel('time');
ylabel('error');
title('distance between exact and approximated path');
```

Problem 11.3 Robustness of L-Stable Method

In this exercise we investigate the *robustness* of L-stable 1-step methods for the scalar linear model problem [NUMODE, Eq. (3.1.1)]. Robustness denotes the special property of a numerical integrator, namely that the integration error can be reasonably estimated independently of a parameter in the differential equation.

We apply the two step Radau-2-method (→ [NUMODE, Eq. (3.6.4)]) to the model problem

$$\dot{y} = \lambda y, \qquad y(0) = 1. \tag{11.3.1}$$

Hence we construct an equidistant mesh with step size $h = \frac{1}{N}$ on the time interval $[0, 1]$.

(11.3a) Determine an analytic expression for

$$e_N(\lambda) := |y_N - y(1)|, \tag{11.3.2}$$

by expressing the numerical solution using the stability function (→ [NUMODE, Thm. 3.1.6]) of the method.

Solution: By [NUMODE, Eq. (3.1.16)] we have

$$y_N = (\boldsymbol{\Psi}^h)^N y(0) = S(h\lambda)^N y(0) = S(\tfrac{\lambda}{N})^N,$$

where $S(\cdot)$ denotes the stability function of the Radau-2-method. This gives

$$e_N(\lambda) = \left|S(\tfrac{\lambda}{N})^N - \mathrm{e}^{\lambda}\right|.$$

For the stability function [NUMODE, Thm. 3.1.6], [NODE, Rem. 3.1.2] implies

$$S(z) = \frac{1 + \frac{1}{3}z}{1 - \frac{2}{3}z + \frac{1}{6}z^2}.$$

(11.3b) Write a MATLAB function

```
e = sup_e(N),
```

which determines a lower bound for

$$\sup_{0\le\lambda\le N} e_N(\lambda)$$

by sampling on the interval $[0, N]$.

Solution: See Listing 11.6.

Listing 11.6: subproblem (11.3b)

```
function e = sup_e(N)
% determine lower bound for the sup over e_N(lambda) for
% 0 <= lambda <= N
%
% Input:
% N: number of steps, see above
%
% Output:
% e: lower bound for the sup
%
% call:
% e = sup_e(10);

% stability function of the Radau-2-method
S = @(z) (1 + z/3) ./ (1 + (z/2 - 2) .* z/3);

% generate net for the sampling
lambda = linspace(0, N, 5000);

% find max
e = max( abs( S(lambda/N).^N - exp(lambda) ) );
```

(11.3c) Try to give a constant C, independent of h and λ, such that

$$|y_N - y(1)| \le Ch\,. \tag{11.3.3}$$

Does this C exist? Prove it.

Solution: No C with those properties exists because

$$\sup_{0\le\lambda\le N} e_N(\lambda) \ge e_N(N) = \left|S(1)^N - \mathrm{e}^N\right| \to \infty \quad \text{for} \quad N = \tfrac{1}{h} \to \infty.$$

(11.3d) State a 1-step method which when applied to (11.3.1) with uniform time step size h has the property (11.3.3) with C independent of h and λ.

Solution: The exponential Euler method has the desired property since its stability function is given by $S(z) = \exp(z)$ and hence

$$e_N(\lambda) = \left|S(\tfrac{\lambda}{N})^N - \exp(\lambda)\right| = \left|\exp(\tfrac{\lambda}{N})^N - \exp(\lambda)\right| = 0.$$

Published on 9 May 2016.

To be submitted by 17 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 9, 2016