TMMS30 Lab 2

Eric Moringe (erimo668, 000228-5435, 43),
David Wiman (davwi279, 000120-8495, 67)

February 23, 2023

Task 1

(a) The plots for task 1 a),

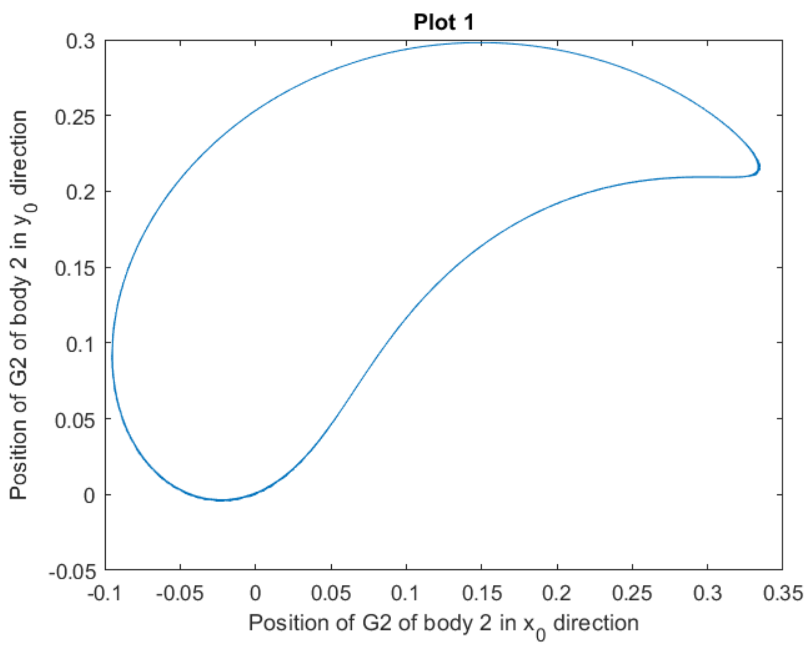

Figure 1: Position of the center of mass of body 2.

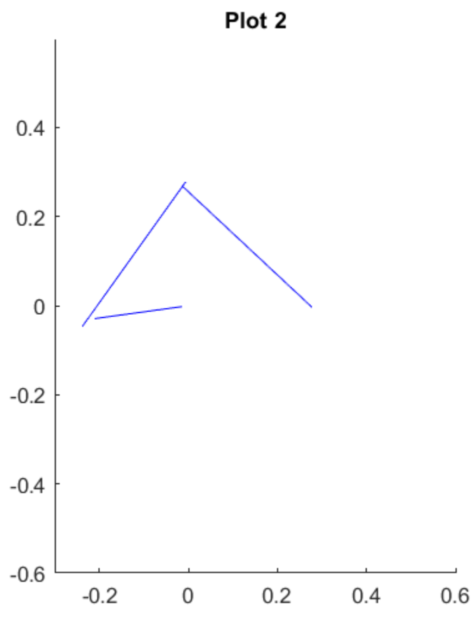

Figure 2: The mechanism at the end of the simulation with Baumgarte stabilization parameters set to zero.

With Baumgarte stabilization parameters set to zero, we see gaps in the solution. The bodies have gaps between them.

(b) The plots for task 1 b),

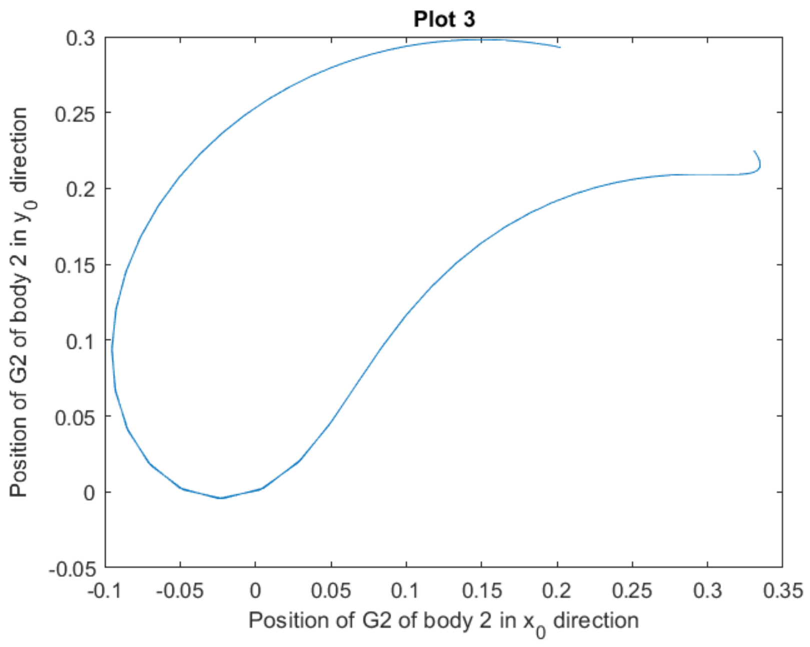

Figure 3: Position of the center of mass of body 2 when a spring is introduced.

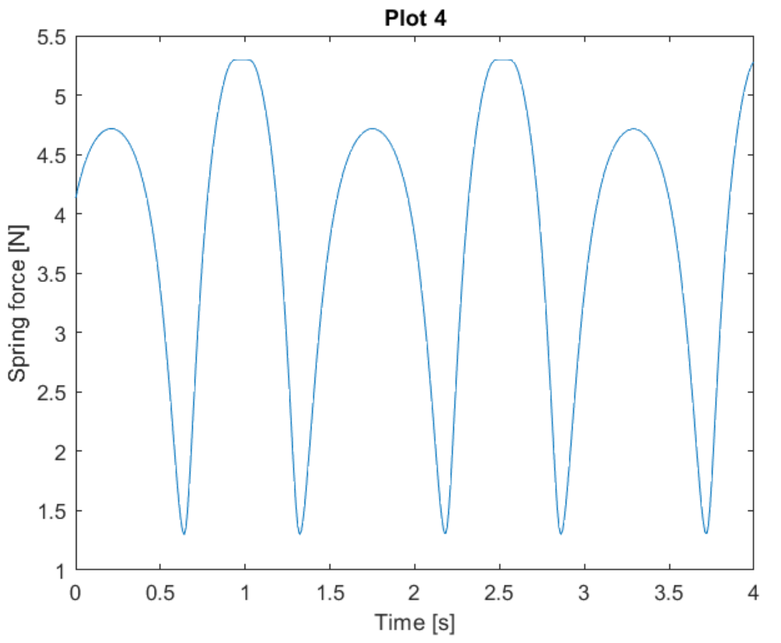

Figure 4: The spring force during 4 seconds of simulation. The maximum force is 5.3 N and the minimum force is 1.3 N.

The spring is as extended as possible when body 1 and 2 form a straight line of length 0.6 m. Since the unstretched length of the spring is 0.07 m, the elongation is 0.53 m. With the spring constant $k = 10$, the maximum force is 5.3 N, using Hooke's law, which match the maximum value of the simulation in plot 4.

The spring is the least extended when body 1 and 2 overlap, i.e. when body 1 points straight down and body 2 points straight up. The distance between the base and the top of body 2 is then $-0.2+0.4 = 0.2$. This distance, with $l_0 = 0.07$, means the elongation is 0.13 m and the spring force is therefore 1.3 N.

(c) The plots for task 1 c),

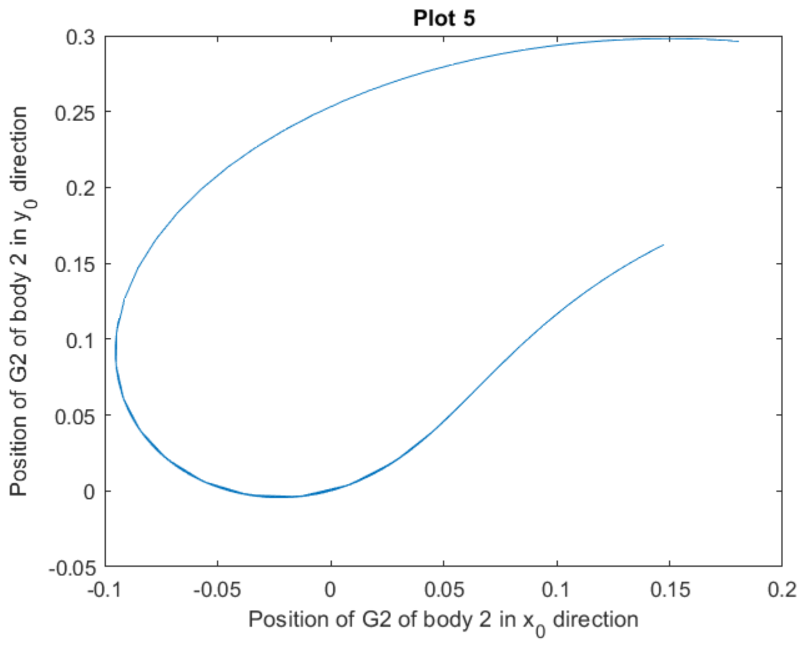

Figure 5: Position of the center of mass of body 2 when a spring and damper is introduced.

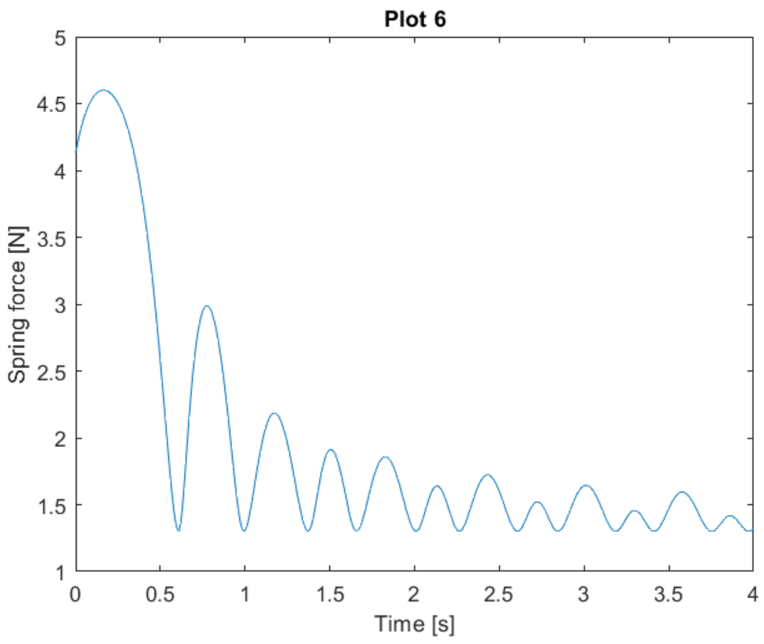

Figure 6: The spring force when a damper is present.

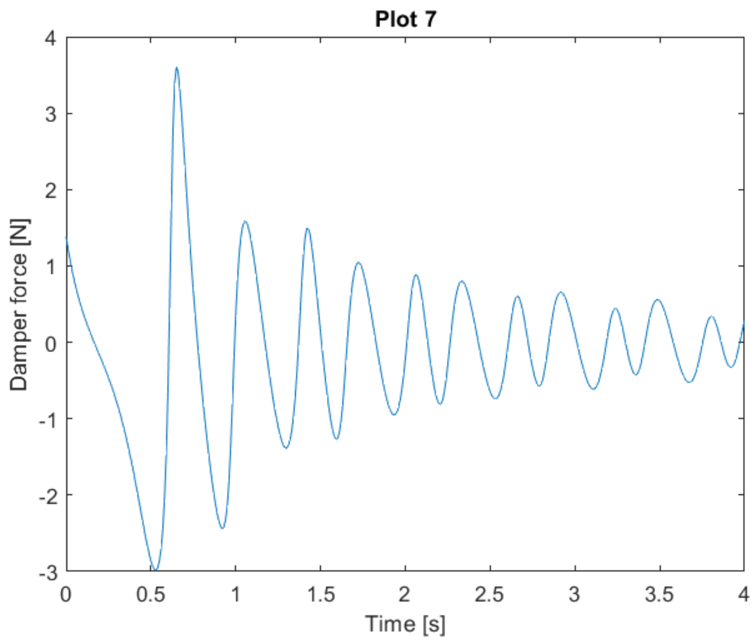

Figure 7: The damper force when a spring is present.

The code added to the file get_Q.m:

```
for i=1:length(loads.tdamper)

  %Add your stuff here!
```

```
    body_i=loads.tdamper(i).body(1);   %Body i
    q_i=q(3*body_i-2:3*body_i);         %Absolute coordinates for body i: [x_i,y_i,theta_i]
    q_dot_i = q_dot(3*body_i-2:3*body_i);
    s_pi=rot(q_i(3),loads.tdamper(i).loc{1});
    s_pi_tilde = [-s_pi(2);s_pi(1)];
    r_pi=q_i(1:2)+s_pi;
    r_dot_i=q_dot_i(1:2);

    body_j=loads.tdamper(i).body(2);   %Body j
    q_j=q(3*body_j-2:3*body_j);
    q_dot_j = q_dot(3*body_j-2:3*body_j);
    s_pj=rot(q_j(3),loads.tdamper(i).loc{2});
    s_pj_tilde = [-s_pj(2);s_pj(1)];
    r_pj=q_j(1:2)+s_pj;
    r_dot_j=q_dot_j(1:2);

    %Length of damper:
    l_thin = norm(r_pj-r_pi);

    % Vector between damper ends
    l_fat = r_pj-r_pi;

    %speed of damper:
    l_dot_fat = r_dot_j + q_dot_j(3)*s_pj_tilde - r_dot_i - q_dot_i(3)*s_pi_tilde;

    l_dot_thin = l_dot_fat'*l_fat/l_thin;

    %Unit vector along the damber from body i to body j:
    e=(r_pj-r_pi)/l;

    %Damper force:
    force=loads.tdamper(i).c*l_dot_thin;

    %Force on body i and j:
    f_i=force*e;
    f_j=-f_i;

    %These forces give the impressed loads:
    Q_f_i=get_Q_f(f_i,s_pi);
    Q_f_j=get_Q_f(f_j,s_pj);


    %Add contributions to Q:
    ind_i=3*body_i-2:3*body_i;   %Rows of Q corresponding to body i.
    ind_j=3*body_j-2:3*body_j;   %Rows of Q corresponding to body j.

    Q(ind_i)=Q(ind_i)+Q_f_i;
    Q(ind_j)=Q(ind_j)+Q_f_j;


    if (nargin == 8)
        %Store the damper force in loads_hist:
```

```
      loads_hist.tdamper(i).f(jj)=force;
  end
end
```

Task 2

(a) The plot for task 2 a),

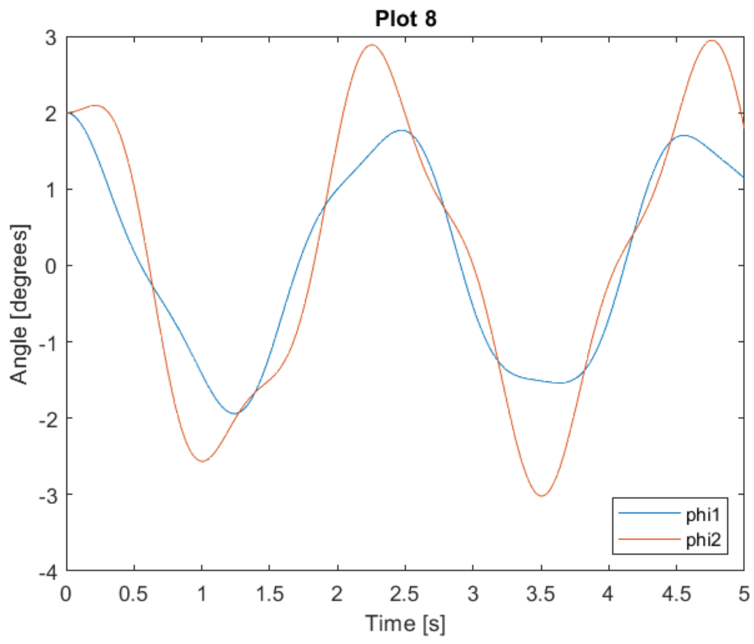

Figure 8: The angles φ of the double pendulum's bodies.

(b) The plot for task 2 b),

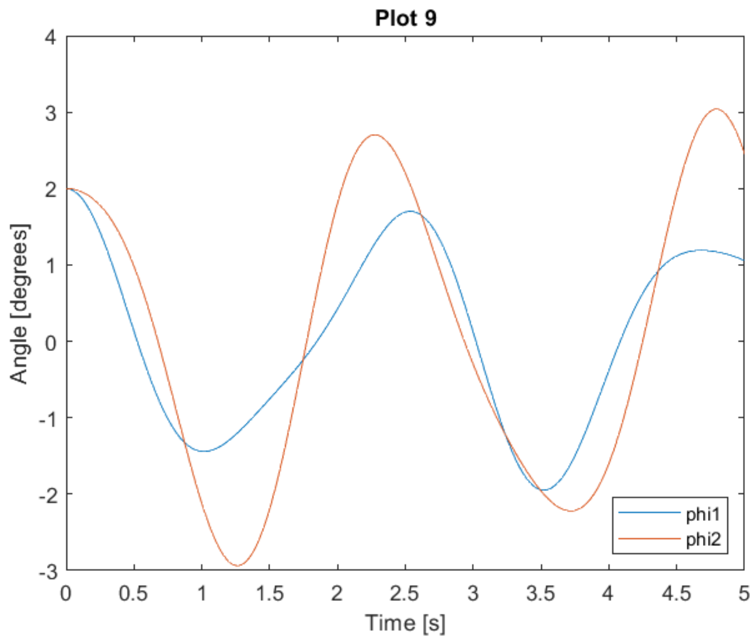

Figure 9: The angles φ of the double pendulum's bodies using Matlab's ODE45 solver.

The solutions are quite similar for the MBS code and the ODE45 solver.

The code written for the ODE solver:

```
%% ODE45 Solution

l1 = 1;
l2 = 1;
m1 = 10;
m2 = 10;
g=9.81;

phi_1=pi/90;
phi_2=pi/90;
t_max=5;

options=odeset('RelTol',1e-6,'AbsTol',1e-8);
[t_vec,Y]=ode45(@(t,y) double_equ(t,y,l1,l2,m1,m2,g),[0 t_max],[phi_1 0
    phi_2 0],options);

phi_1_vec=Y(:,1);
phi_1_dot_vec=Y(:,2);
phi_2_vec=Y(:,3);
phi_2_dot_vec=Y(:,4);

figure(1)
plot(t_vec,phi_1_vec*180/pi)
hold on
plot(t_vec,phi_2_vec*180/pi)
title('Plot 9')
xlabel('Time [s]')
ylabel('Angle [degrees]')
legend('phi1','phi2','Location','Southeast')

function ydot=double_equ(t,y,l1,l2,m1,m2,g)
phi_1=y(1);
phi_1_dot=y(2);
phi_2=y(3);
phi_2_dot=y(4);

A=[(m1*l1^2/3+m2*l1^2), m2*l1*l2/2;
l2/3,  l1/2];

b=[-g*l1*(m1+2*m2)*phi_1/2;
-g*phi_2/2];

z=A\b;

ydot=zeros(4,1);
ydot(1)=phi_1_dot;
ydot(2)=z(1);
ydot(3)=phi_2_dot;
ydot(4)=z(2);
end
```