

Task 1

a)

For this task, I got a lot of help from <https://www.math24.net/double-pendulum/>.

```
1 clear all
2 clc
3
4 % Parameters
5 syms m M g L F real
6 % Variables
7 syms x theta_1 theta_2 real
8 syms dx dtheta_1 dtheta_2 real
9
10 % Define symbolic variable q for the generalized coordinates
11 % x, theta1 and theta2
12 q = [x; theta_1; theta_2];
13 % Define symbolic variable dq for the derivatives
14 % of the generalized coordinates
15 dq = [dx; dtheta_1; dtheta_2];
16 % Write the expressions for the positions of the masses
17 p{1} = [q(1)+sin(theta_1)*L ;
18         -cos(theta_1)*L];
19 p{2} = p{1} + [sin(theta_2)*L ;
20               -cos(theta_2)*L];
21
22 % Kinetic energy of the cart
23 T = (m/2)*dq(1)^2;
24 % For loop that adds the kinetic energies of the masses
25 for k = 1:length(p)
26     dp{k} = jacobian(p{k},q)*dq; % velocity of mass k
27     T = T + (M/2)*(dp{k}(1)^2+dp{k}(2)^2); % add kinetic energy of mass k
28 end
29 T = simplify(T);
30
31 % Potential energy of the cart
32 V = 0;
33 % For loop that adds the potential energies of the masses
34 for k = 1:length(p)
35     V = V + m*g*p{k}(2); % add potential energy of mass k
36 end
37 V = simplify(V);
38
39 % Generalized forces
40 Q = [F; 0; 0];
41
42 % Lagrangian
43 Lag = T - V;
44
45 Lag_q = simplify(jacobian(Lag,q)).';
46 Lag_qdq = simplify(jacobian(Lag_q.',dq));
47 Lag_dq = simplify(jacobian(Lag,dq)).';
48 Lag_dq dq = simplify(jacobian(Lag_dq.',dq));
49
50 % The equations have the form W*q_dotdot = RHS, with
51 W = Lag_dq dq;
52 RHS = Q + simplify(Lag_q - Lag_qdq*dq);
53
```

```

54 state = [q;dq];
55 param = [m;M;L;g];
56
57 matlabFunction(p{1},p{2}, 'file','PendulumPosition','vars',{state, param});
58 matlabFunction(W,RHS, 'file','PendulumODEMatrices','vars',{state,F,param});

```

b)

```

1 function [dstate] = PendulumDynamics(t, x, parameters)
2 %PENDULUMODEMATRICES
3 % [W,RHS] = PENDULUMODEMATRICES(IN1,F,IN3)
4
5 % This function was generated by the Symbolic Math Toolbox version 8.5.
6 % 22-Feb-2021 19:18:47
7 % state = [q;dq];
8 % param = [m;M;L;g];
9 q = x(1:3);
10 dq = x(4:6);
11 F= -10*x(1) - x(4);
12 [W, RHS] = PendulumODEMatrices(x, F, parameters);
13 dstate = [dq; W\RHS];
14 end

```

c)

After a lot of time spent debugging, I finally got the simulation to work. The issue was my inconsistent reference frame, sometimes y was up sometimes it was down.

The simulation looks a bit weird as there are two pendulums seen from different angles, I don't know if that was the intention.

The system behaves as expected. It is chaotic in the beginning, but as there is a dampening part it goes toward equilibrium after a while.

```

1 clear all
2 close all
3 clc
4
5 % Parameters and initial states
6 tf = 45;
7 parameters = [1; 0.1; 1; 9.81];
8 state = [1; pi/4; 0; 0; 0; 0];
9
10 % Simulation
11 try
12
13     %%%%% MODIFY THE CODE AS YOU SEE FIT
14
15     [tsim,xsim] = ode45(@(t,x)PendulumDynamics(t, x, parameters),[0,tf],state)
16     ;
17 catch message
18     display('Your simulation failed with the following message:')
19     display(message.message)
20     display(' ')
21
22     % Assign dummy time and states if simulation failed
23     tf = 0.1;

```

```

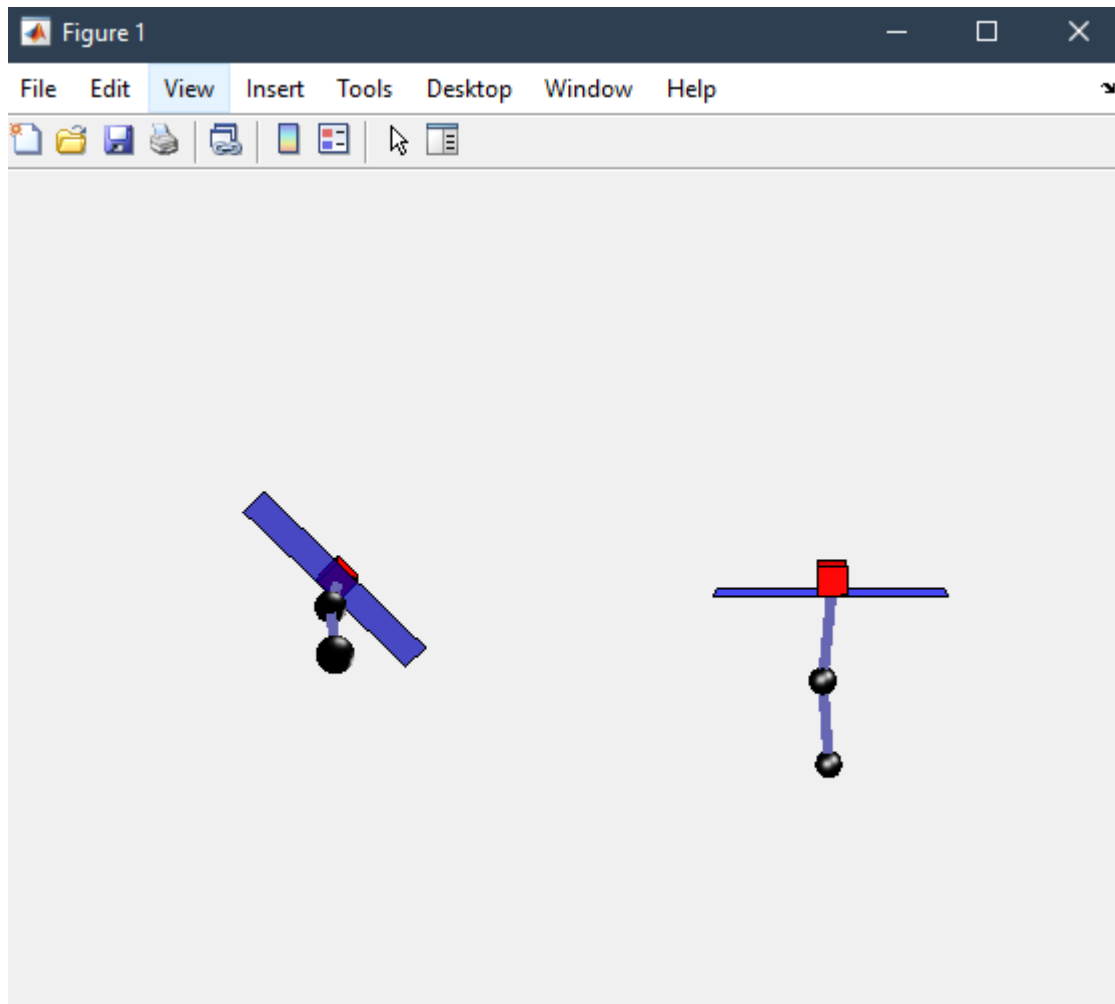
24     tsim = [0,tf];
25     xsim = 0;
26 end
27
28 %% 3D animation
29 DoublePlot = true;
30 FS = 30;
31 scale = 0.1;
32
33 % Create Objects
34 % Cube
35 vert{1} = 3*[ -1, -1, 0; %1
36              1, -1, 0; %2
37              1, 1, 0; %3
38              -1, 1, 0; %4
39              -1, -1, 2; %5
40              1, -1, 2; %6
41              1, 1, 2; %7
42              -1, 1, 2]/2; %8
43 fac{1} = [1 2 3 4;
44          5 6 7 8;
45          1 4 8 5;
46          1 2 6 5;
47          2 3 7 6;
48          3 4 8 7];
49 Lrail = 1.2*max(abs(xsim(:,1)))/scale;
50 % Rail
51 a = 1.5;
52 vert{2} = [-Lrail,-a,-0.1;
53            -Lrail, a,-0.1;
54            Lrail, a,-0.1;
55            Lrail,-a,-0.1];
56 fac{2} = [1,2,3,4];
57 % Sphere
58 [X,Y,Z] = sphere(20);
59 [fac{3},vert{3},c] = surf2patch(3*X/2,3*Y/2,3*Z/2);
60 % Animation
61 tic
62 t_disp = 0;
63 SimSpeed = 1;
64 while t_disp < tf/SimSpeed
65     % Interpolate state
66     x_disp = interp1(tsim,xsim,SimSpeed*t_disp)';
67
68     % Unwrap state. MODIFY
69     x = x_disp(1); % position cart
70     p1 = 1*[x+sin(x_disp(2));-cos(x_disp(2))]; % position 1st ball
71     p2 = p1 + 1*[sin(x_disp(3));-cos(x_disp(3))]; % position 2nd ball
72
73     % Input argument for DrawPendulum
74     pos_disp = [x(1);p1(1);0;p1(2);p2(1);0;p2(2)];
75
76     figure(1);clf;hold on
77     if DoublePlot
78         subplot(1,2,1);hold on
79         DrawPendulum( pos_disp, vert, fac, scale);
80         campos(scale*[15 15 -70])

```

```

81         camtarget(scale*[0,0,1.5])
82         camva(30)
83         camproj('perspective')
84         subplot(1,2,2);hold on
85     end
86     DrawPendulum( pos_disp, vert, fac, scale);
87     campos(scale*[1      70      20])
88     camtarget(scale*[0,0,1.5])
89     camva(30)
90     camproj('perspective')
91     drawnow
92     if t_disp == 0
93         display('Hit a key to start animation')
94         pause
95         tic
96     end
97     t_disp = toc;
98 end

```



Task 2

a)

The position of the ball is:

$$\mathbf{p} = x\vec{b}_1 + R\vec{b}_2 = \begin{bmatrix} x \cos(\theta) - R \sin(\theta) \\ -x \sin(\theta) + R \cos(\theta) \end{bmatrix}$$

b)

Inspired by the previous task, the total movement of the ball can be written as:

$$\text{jac}(\mathbf{p}, \mathbf{q}) * \dot{\mathbf{q}}$$

The angular velocity can be written as:

$$\left(\begin{bmatrix} \frac{1}{R} & 1 \end{bmatrix} * \dot{\mathbf{q}}\right)^2$$

c)

The kinetic energy from translation of the mass center is:

$$T_{trans}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} M (\text{jac}(\mathbf{p}, \mathbf{q}) * \dot{\mathbf{q}})^T * (\text{jac}(\mathbf{p}, \mathbf{q}) * \dot{\mathbf{q}})$$

The kinetic energy from rotation of the mass center is:

$$T_{rot}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{5} M R^2 \left(\begin{bmatrix} \frac{1}{R} & 1 \end{bmatrix} * \dot{\mathbf{q}}\right)^2$$

d)

The kinetic energy of the rail is:

$$T_{rail}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} J \dot{\theta}^2$$

Finally the total kinetic energy is $T = T_{trans} + T_{rot} + T_{rail}$.

e)

```
1 clear all
2 clc
3
4 % Parameters
5 syms J M R g To real
6 % Variables
7 syms x theta real
8 syms dx dtheta real
9
10 % Define symbolic variable q for the generalized coordinates
11 % x and theta
12 q = [x; theta];
13 % Define symbolic variable dq for the derivatives
14 % of the generalized coordinates
15 dq = [dx; dtheta];
16 % Write the expressions for the position of
17 % the center of the ball:
18 p = [x*cos(theta)-R*sin(theta);
19      x*sin(theta)+R*cos(theta)];
```

```

20
21 % Kinetic energy
22 T = 0.5*J*dtheta^2; % kinetic energy of beam
23
24 dp = jacobian(p,q)*dq; % linear velocity of ball
25 T = T + 0.5*M*(dp'*dp); % add linear kinetic energy of ball
26
27 I = (2/5)*M*R^2; % inertia in rotation of ball
28 omega = [1/R 1]*dq; % angular velocity of ball
29
30 T = T + 0.5*I*omega^2; % add rotational kinetic energy of ball
31
32 T = simplify(T);
33
34 % Potential energy
35 V = p(2);
36
37 % Generalized forces
38 Q = [0; To];
39
40 % Lagrangian
41 Lag = T - V;
42
43 Lag_q = simplify(jacobian(Lag,q)).';
44 Lag_qdq = simplify(jacobian(Lag_q.',dq));
45 Lag_dq = simplify(jacobian(Lag,dq)).';
46 Lag_dq dq = simplify(jacobian(Lag_dq.',dq));
47
48 % The equations have the form W*q_dotdot = RHS, with
49 W = Lag_dq dq;
50 RHS = Q + simplify(Lag_q - Lag_qdq*dq);
51
52 state = [q;dq];
53 param = [J; M; R; g];
54
55 matlabFunction(p, 'file','BallPosition','vars',{state,param});
56 matlabFunction(W,RHS, 'file','BallAndBeamODEMatrices','vars',{state,To,param})
    ;

```

f)

```

1 function [dstate] = BallAndBeanDynamics(t, x, parameters)
2 %PENDULUMODEMATRICES
3 % [W,RHS] = PENDULUMODEMATRICES(IN1,F,IN3)
4 % This function was generated by the Symbolic Math Toolbox version 8.5.
5 % 22-Feb-2021 19:18:47
6 % state = [q;dq];
7 % param = [J; M; R; g];
8 q = x(1:2);
9 dq = x(3:4);
10 F = 200*(x(1) - x(2)) + 70*(x(3) - x(4));
11 [W, RHS] = BallAndBeamODEMatrices(x, F, parameters);
12 dstate = [dq; W\RHS];
13 end

```

g)

The simulation works, the controller appear to work, but the physics are not reasonable as the ball stay attached to the beam, even when the beam is rotated 180°.

```
1 clear all
2 close all
3 clc
4
5 % Parameters and initial states
6 tf = 15;
7 parameters = [1; 10; 0.25; 9.81];
8 state = [1;0;0;0];
9
10 % Simulation
11 try
12
13     %%%%% MODIFY THE CODE AS YOU SEE FIT
14
15     [tsim,xsim] = ode45(@(t,x)BallAndBeamDynamics(t, x, parameters),[0,tf],
16         state);
17
18 catch message
19     display('Your simulation failed with the following message:')
20     display(message.message)
21     display(' ')
22
23     % Assign dummy time and states if simulation failed
24     tf = 0.1;
25     tsim = [0,tf];
26     xsim = 0;
27
28 end
29
30 %% 3D animation
31 DoublePlot = true;
32 scale = 0.25;
33 FS = 30;
34 ball_radius = 0.25;
35
36 % Create Objects
37 % Rail
38 Lrail = 2;
39 a = ball_radius;
40 vert{1} = [-Lrail,-a, 0;
41             -Lrail, a, 0;
42             Lrail, a, 0;
43             Lrail,-a, 0];
44 fac{1} = [1,2,3,4];
45 % Sphere
46 [X,Y,Z] = sphere(20);
47 [fac{2},vert{2},c] = surf2patch(X,Y,Z);
48
49 % Animation
50 tic
51 t_disp = 0;
52 SimSpeed = 1;
53 while t_disp < tf/SimSpeed
54     % Interpolate state
55     x_disp = interp1(tsim,xsim,SimSpeed*t_disp)';
```

```

54
55 % Unwrap state. MODIFY
56 theta = x_disp(2); % beam angle
57 pos = x_disp(1)*[cos(theta);sin(theta)] + ball_radius*[-sin(theta);cos(
    theta)];
58 pos = [pos(1);0;pos(2)]; % ball position
59
60 figid = figure(1);clf;hold on
61 if DoublePlot
62     subplot(1,2,1);hold on
63     DrawBallAndBeam(pos, theta, vert, fac, xsim, ball_radius);
64     campos(scale*[10    10    20])
65     camtarget(scale*[0,0,1.5])
66     camva(30)
67     camproj('perspective')
68     subplot(1,2,2);hold on
69 end
70 DrawBallAndBeam(pos, theta, vert, fac, xsim, ball_radius);
71 campos(0.4*scale*[1    70    20])
72 camtarget(scale*[0,0,1.5])
73 camva(30)
74 camproj('perspective')
75 drawnow
76
77 if t_disp == 0
78     display('Hit a key to start animation')
79     pause
80     tic
81 end
82 t_disp = toc;
83 end

```