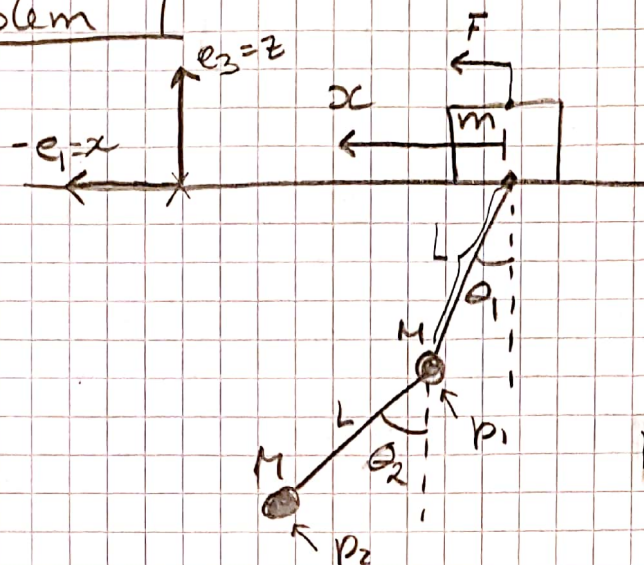


## Problem



$$m = 1 \text{ kg}$$

$$M = 1 \text{ kg}$$

$$L = 1 \text{ m}$$

$$p_1 = \begin{bmatrix} x + L \sin \theta_1 \\ -L \cos \theta_1 \end{bmatrix}$$

$$a) \quad q = \begin{bmatrix} x \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$p_0 = \begin{bmatrix} x \\ 0 \end{bmatrix}$$

$$p_2 = \begin{bmatrix} x + L \sin \theta_1 + L \sin \theta_2 \\ -L \cos \theta_1 - L \cos \theta_2 \end{bmatrix}$$

$$T(q, \dot{q}) = \frac{1}{2} m \dot{x}^2 + \frac{1}{2} M \dot{p}_1^T \dot{p}_1 + \frac{1}{2} M \dot{p}_2^T \dot{p}_2$$

$$V(q) = Mg p_{1,z} + Mg p_{2,z}$$

second element, i.e. height

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q$$

$$\frac{d}{dt} p_1(q) = \frac{\partial p_1}{\partial q} \cdot \dot{q}, \quad \frac{d}{dt} p_2(q) = \frac{\partial p_2}{\partial q} \cdot \dot{q}$$

$$b) \quad F = -10x - \dot{x} \rightarrow Q = \begin{bmatrix} -10x - \dot{x} \\ 0 \\ 0 \end{bmatrix}$$

The simulation shows very sensible results. The pendulum swing in the manner one would expect, and the box glides back and forth on the board.



```

1 %% Symbolic variables
2 % Generalized coordinates
3 syms x real;
4 syms theta1 real;
5 syms theta2 real;
6
7 % Time derivative of generalized coords
8 syms dx real;
9 syms dtheta1 real;
10 syms dtheta2 real;
11
12 q = [x; theta1; theta2]; % generalized coords. vector
13 dq = [dx; dtheta1; dtheta2];
14
15 % Parameters
16 syms m real;
17 syms M real;
18 syms L real;
19
20 % Constants
21 g = 9.81; % acceleration due to gravity
22
23 % Position of masses M
24 p0 = [x; 0];
25 p1 = p0 + [L*sin(theta1); -L*cos(theta1)];
26 p2 = p1 + [L*sin(theta2); -L*cos(theta2)];
27
28 % Time derivaitve of position of masses M
29 dp1 = jacobian(p1, q) * dq;
30 dp2 = jacobian(p2, q) * dq;
31
32
33 %% Lagrange
34 % Kinetic energy
35 T = 1/2 * m * dx^2 + 1/2 * M * (dp1') * dp1 + 1/2 * M * (dp2') * dp2;
36 T = simplify(T);
37
38 % Potential energy
39 V = M * g * p1(2) + M * g * p2(2);
40 V = simplify(V);
41
42 % External force
43 F = -10*x - dx;
44 Q = [F; 0; 0];
45
46 % Lagrange function
47 L = T - V;
48
49
50 % Lagrange equation (see (2) in task text)
51 W_hessian = hessian(L, dq);
52 other_vector = Q + jacobian(L,q)' - jacobian(W_hessian * dq, q) * dq;
53
54
55 %% Export function
56 state = [x; theta1; theta2; dx; dtheta1; dtheta2];
57 params = {L, M, m};
58

```

```
59 syms t real; % supporting variable for function creation
60 matlabFunction(W_hessian, 'File', 'probl_W_hessian');
61 matlabFunction(other_vector, 'File', 'probl_other_vector');
62 % Can't get parameters to work :( , 'Vars', {t, state, params}
63
```

```

1 clear all
2 close all
3 clc
4
5 %% Declarations
6 % Parameters
7 L = 1;
8 M = 1;
9 m = 1;
10
11 % Initial position
12 x0 = 0;
13 theta1_0 = pi/4;
14 theta2_0 = pi/2;
15 q0 = [x0; theta1_0; theta2_0];
16
17 % Initial velocity
18 dq0 = zeros(3,1);
19
20 % Initial state
21 %      1  2      3      4  5      6
22 % x = [x; theta1; theta2; dx; dtheta1; dtheta2];
23 state = [q0; dq0];
24
25
26 %% Simulation
27 tf = 45;
28
29 % Function declarations
30 W = @(x) prob1_W_hessian(L,M,m,x(2),x(3));
31 other = @(x) prob1_other_vector(L,M,x(5),x(6),x(4),x(2),x(3),x(1));
32 simFunc = @(t, x) [x(4:6); W(x) \ other(x)];
33
34 [time, statetrajectory] = ode45(simFunc, [0 tf], state);
35
36
37 %% 3D animation
38 DoublePlot = true;
39 FS = 30;
40 scale = 0.1;
41
42 % Create Objects
43 % Cube
44 vert{1} = 3*[ -1, -1, 0; %1
45              1, -1, 0; %2
46              1, 1, 0; %3
47              -1, 1, 0; %4
48              -1, -1, 2; %5
49              1, -1, 2; %6
50              1, 1, 2; %7
51              -1, 1, 2]/2; %8
52 fac{1} = [1 2 3 4;
53           5 6 7 8;
54           1 4 8 5;
55           1 2 6 5;
56           2 3 7 6;
57           3 4 8 7];
58 Lrail = 1.2*max(abs(statetrajectory(:,1)))/scale;

```

```

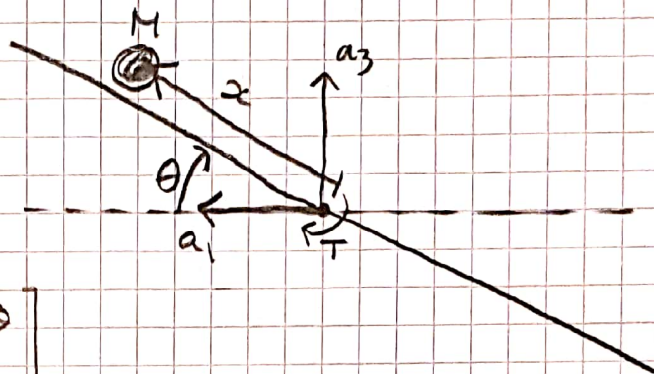
59 % Rail
60 a = 1.5;
61 vert{2} = [-Lrail,-a,-0.1;
62           -Lrail, a,-0.1;
63           Lrail, a,-0.1;
64           Lrail,-a,-0.1];
65 fac{2} = [1,2,3,4];
66 % Sphere
67 [X,Y,Z] = sphere(20);
68 [fac{3},vert{3},c] = surf2patch(3*X/2,3*Y/2,3*Z/2);
69 % Animation
70 tic
71 t_disp = 0;
72 SimSpeed = 1;
73 while t_disp < tf/SimSpeed
74     % Interpolate state
75     state_animate = interp1(time,statetraj,SimSpeed*t_disp)';
76
77     x = state_animate(1);
78     theta1 = state_animate(2);
79     theta2 = state_animate(3);
80
81     p0 = [x; 0]; % box
82     p1 = p0 + [L*sin(theta1); -L*cos(theta1)]; % sphere 1
83     p2 = p1 + [L*sin(theta2); -L*cos(theta2)]; % sphere 2
84
85     % shift coords
86     p0_3d = [-p0(1); 0; p0(2)];
87     p1_3d = [-p1(1); 0; p1(2)];
88     p2_3d = [-p2(1); 0; p2(2)];
89
90     % Input argument for DrawPendulum
91     pos_disp = [p0_3d(1); p1_3d; p2_3d];
92
93     figure(1);clf;hold on
94     if DoublePlot
95         subplot(1,2,1);hold on
96         DrawPendulum( pos_disp, vert, fac, scale);
97         campos(scale*[15 15 -70])
98         camtarget(scale*[0,0,1.5])
99         camva(30)
100        camproj('perspective')
101        subplot(1,2,2);hold on
102    end
103    DrawPendulum( pos_disp, vert, fac, scale);
104    campos(scale*[1 70 20])
105    camtarget(scale*[0,0,1.5])
106    camva(30)
107    camproj('perspective')
108    drawnow
109    if t_disp == 0
110        display('Hit a key to start animation')
111        pause
112        tic
113    end
114    t_disp = toc;
115 end
116

```

## Problem 2

$$J = 1 \text{ kg m}^2, \quad M = 10 \text{ kg}, \quad R = 0,25 \text{ m}$$

$$q = \begin{bmatrix} x \\ \theta \end{bmatrix}$$



$$p = \begin{bmatrix} x \cos \theta \\ 0 \\ x \sin \theta \end{bmatrix}$$

$$T_{\text{beam}} = \frac{1}{2} J \dot{\theta}^2$$

$$T_{\text{ball}, x} = \frac{1}{2} M \dot{x}^2$$

Rotation velocity of ball

$$\omega = \dot{\theta} + \frac{\dot{x}}{R}, \quad J_{\text{ball}} = \frac{2}{5} MR^2$$

$$T_{\text{ball}, \omega} = \frac{1}{2} \frac{2}{5} MR^2 \left( \dot{\theta} + \frac{\dot{x}}{R} \right)^2$$

$$\Rightarrow T(q, \dot{q}) = \frac{1}{2} M \dot{x}^2 + \frac{1}{2} J \dot{\theta}^2 + \frac{1}{2} \frac{2}{5} MR^2 \left( \dot{\theta} + \frac{\dot{x}}{R} \right)^2$$

$$V(q) = Mg p_3 = Mg x \sin \theta$$

$$\Rightarrow \mathcal{L} = T - V$$

$$\text{Torque} = 200(x - \theta) + 70(\dot{x} - \dot{\theta})$$

$$\Rightarrow Q = \begin{bmatrix} 0 \\ \text{Torque} \end{bmatrix}$$



The simulation shows how the PD-controller tries to keep the ball on the beam by rotating the beam.

The results are reasonable. See plot for coordinates vs. time.



```

1 %% Symbolic variables
2 % Generalized coordinates
3 syms x real;
4 syms theta real;
5
6 % Time derivative of generalized coords
7 syms dx real;
8 syms dtheta real;
9
10 q = [x; theta]; % generalized coords. vector
11 dq = [dx; dtheta];
12
13 % Parameters
14 syms R real;
15 syms M real;
16 syms J real;
17
18 % Constants
19 g = 9.81; % acceleration due to gravity
20
21
22 % Position of ball in frame a
23 p_a = [x*cos(theta); 0; x*sin(theta)];
24
25 %% Lagrange
26 % Kinetic energy
27 T = 1/2 * M * dx^2 + 1/2 * J * dtheta^2 + 1/2 * 2/5 * M * R^2 * (dtheta + dx/R) ^2;
28 T = simplify(T);
29
30 % Potential energy
31 V = M * g * p_a(3);
32 V = simplify(V);
33
34 % External torque
35 Torque = 200 * (x - theta) + 70 * (dx - dtheta);
36 Q = [0; Torque];
37
38 % Lagrange function
39 L = T - V;
40
41
42 % Lagrange equation (see (2) in task text)
43 W_hessian = hessian(L, dq);
44 other_vector = Q + jacobian(L,q)' - jacobian(W_hessian * dq, q) * dq;
45
46
47 %% Export function
48 state = [q; dq];
49 params = [J; M; R];
50
51 matlabFunction(W_hessian, 'File', 'prob2_W_hessian', 'Vars', {state, params});
52 matlabFunction(other_vector, 'File', 'prob2_other_vector', 'Vars', {state,
params});
53 % Can't get parameters to work :( , 'Vars', {state, params}
54

```

```

1 clear all
2 close all
3 clc
4
5
6 %% Declarations
7 % Parameters
8 J = 1;
9 M = 1;
10 R = 0.25;
11
12 % Initial position
13 x0 = 1;
14 theta_0 = 0;
15 q0 = [x0; theta_0];
16
17 % Initial velocity
18 dq0 = zeros(2,1);
19
20 % Initial state
21 %      1  2      3  4
22 % x = [x; theta; dx; dtheta];
23 state = [q0; dq0];
24
25
26 %% Simulation
27 tf = 15;
28
29 % Function declarations
30 W = @(x) prob2_W_hessian(x, [J,M,R]');
31 other = @(x) prob2_other_vector(x, [J,M,R]');
32 simFunc = @(t, x) [x(3:4); W(x) \ other(x)];
33
34 [tsim, xsim] = ode45(simFunc, [0 tf], state);
35
36
37 %% 3D animation
38 DoublePlot = true;
39 scale = 0.25;
40 FS = 30;
41 ball_radius = 0.25;
42
43 % Create Objects
44 % Rail
45 Lrail = 2;
46 a = ball_radius;
47 vert{1} = [-Lrail,-a, 0;
48            -Lrail, a, 0;
49            Lrail, a, 0;
50            Lrail,-a, 0];
51 fac{1} = [1,2,3,4];
52 % Sphere
53 [X,Y,Z] = sphere(20);
54 [fac{2},vert{2},c] = surf2patch(X,Y,Z);
55
56 % Animation
57 tic
58 t disp = 0;

```

```

59 SimSpeed = 1;
60 while t_disp < tf/SimSpeed
61     % Interpolate state
62     x_disp = interp1(tsim,xsim,SimSpeed*t_disp)';
63
64     % Unwrap state. MODIFY
65     x = x_disp(1);
66     theta = x_disp(2); % beam angle
67     pos = x*[cos(theta);sin(theta)] + ball_radius*[-sin(theta);cos(theta)];
68     pos = [pos(1);0;pos(2)]; % ball position
69
70     figid = figure(1);clf;hold on
71     if DoublePlot
72         subplot(1,2,1);hold on
73         DrawBallAndBeam(pos, theta, vert, fac, xsim, ball_radius);
74         campos(scale*[10 10 20])
75         camtarget(scale*[0,0,1.5])
76         camva(30)
77         camproj('perspective')
78         subplot(1,2,2);hold on
79     end
80     DrawBallAndBeam(pos, theta, vert, fac, xsim, ball_radius);
81     campos(0.4*scale*[1 70 20])
82     camtarget(scale*[0,0,1.5])
83     camva(30)
84     camproj('perspective')
85     drawnow
86
87     if t_disp == 0
88         display('Hit a key to start animation')
89         pause
90         tic
91     end
92     t_disp = toc;
93 end
94

```



