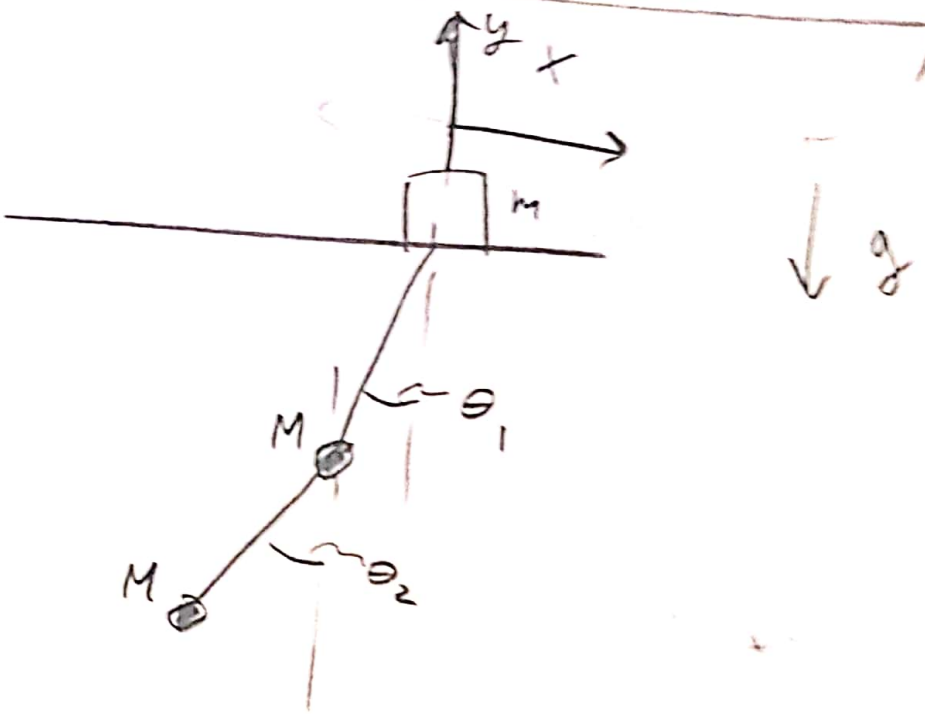
TTK4130 MODELING AND SIMULATION | A 4

ODIN A. SEVERINSEN | MTK

1



a) We see from the figure that the two masses have locations

$$P_1 = \begin{bmatrix} x - L\sin\theta_1 \\ -L\cos\theta_1 \end{bmatrix} \text{ and } P_2 = P_1 + \begin{bmatrix} -L\sin\theta_1 \\ -L\cos\theta_2 \end{bmatrix}$$

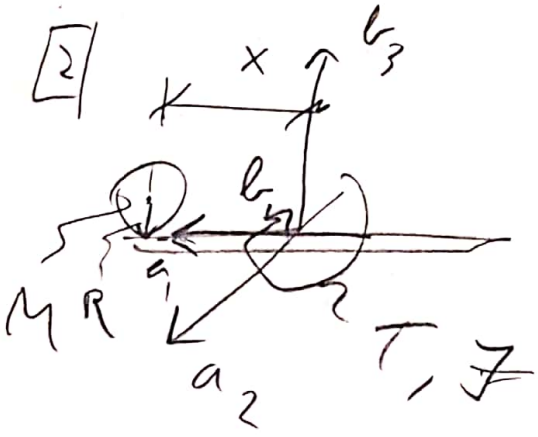
b) See attached code.

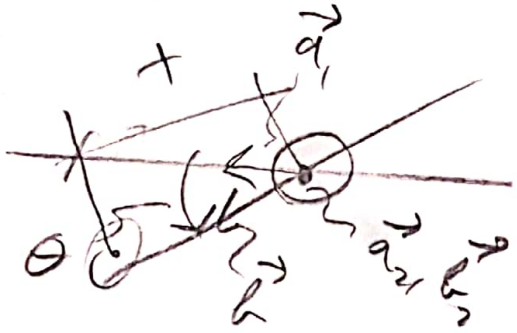
c) The results are reasonable, the pendulum moves

reasonable.

Scanned by CamScanner

[2]

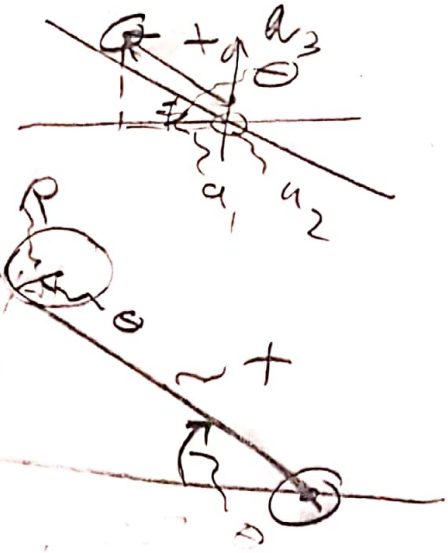




A rotates about $\vec{a}_2$ & $\vec{b}_2$

a) A rotates ~~about~~ in a-frame:

$$P = \begin{bmatrix} x\cos\theta - R\sin\theta \\ 0 \\ x\sin\theta + R\cos\theta \end{bmatrix}$$



b) The angular velocity is

$$\omega = \dot{\theta} + \dot{x}/R$$

c) The kinetic energy is a sum of ~~translative~~ translation and rotation energy, where rotation is both about its own axis and the planets joint.

Scanned by CamScanner

$$T_{ball} = \frac{1}{2} M \dot{\bar{p}}^T \dot{\bar{p}} + \frac{1}{2}\left(\overbrace{\frac{2}{5}MR^2 + M\bar{p}^T\bar{p}}^{\text{Steiners' theorem}}\right)\dot{\theta}$$

$$+ \frac{1}{2}\left(\frac{2}{5}MR^2\right)\cdot\left(\frac{\dot{+}}{\dot{-}}\right)$$

d) $T_{beam} = \frac{1}{2} J_{beam} \cdot \dot{\theta}^2$

e) See attached code

f) —— 11 ——

g) The results are reasonable, the ball doesn't glitch out or do anything funny.

Scanned by CamScanner

```
function state_dot = BallAndBeamDynamics(state, parameters)

    q = state(1:2);
    q_dot = state(3:4);
    T = 200*(q(1) - q(2)) + 70*(q_dot(1) - q_dot(2));
    [W, RHS] = BallAndBeamODEMatrices(state, T, parameters);

    state_dot = [q_dot; W\RHS];


end
```

```
Not enough input arguments.

Error in BallAndBeamDynamics (line 3)
    q = state(1:2);
```

Published with MATLAB® R2019a

```
clear all
close all
clc

% Parameters and initial states
tf = 15;
x0 = 1;
theta0 = 0;
q = [x0; theta0];
dq = zeros(2, 1);
J = 1;
M = 10;
R = 0.25;
g = -9.81;
state = [q;dq];
parameters = [J; M; R; g];

% Simulation
try

    %%%%%% MODIFY THE CODE AS YOU SEE FIT

    [tsim,xsim] = ode45(@(t,x)BallAndBeamDynamics(x, parameters),
[0,tf],state);

catch message
    display('Your simulation failed with the following message:')
    display(message.message)
    display(' ')

    % Assign dummy time and states if simulation failed
    tf = 0.1;
    tsim = [0,tf];
    xsim = 0;
end
```

3D animation

```
DoublePlot = true;
scale = 0.25;
FS = 30;
ball_radius = 0.25;

% Create Objects
% Rail
Lrail = 2;
a = ball_radius;
vert{1} = [-Lrail,-a, 0;
           -Lrail, a, 0;
            Lrail, a, 0;
            Lrail,-a, 0];
fac{1} = [1,2,3,4];
```

```
% Sphere
[X,Y,Z] = sphere(20);
[fac{2},vert{2},c] = surf2patch(X,Y,Z);

% Animation
tic
t_disp = 0;
SimSpeed = 1;
if run_sim
while t_disp < tf/SimSpeed
    % Interpolate state
    x_disp   = interp1(tsim,xsim,SimSpeed*t_disp)';

    % Unwrap state. MODIFY
    theta = x_disp(2); % beam angle
    pos = BallPosition(x_disp, parameters);

    figid = figure(1);clf;hold on
    if DoublePlot
        subplot(1,2,1);hold on
        DrawBallAndBeam(pos, theta, vert, fac, xsim, ball_radius);
        campos(scale*[10    10     20])
        camtarget(scale*[0,0,1.5])
        camva(30)
        camproj('perspective')
        subplot(1,2,2);hold on
    end
    DrawBallAndBeam(pos, theta, vert, fac, xsim, ball_radius);
    campos(0.4*scale*[1    70     20])
    camtarget(scale*[0,0,1.5])
    camva(30)
    camproj('perspective')
    drawnow

    if t_disp == 0
        display('Hit a key to start animation')
        pause
        tic
    end
    t_disp = toc;
end
end
```

```
Undefined function or variable 'run_sim'.

Error in BallAndBeamSimulation (line 59)
if run_sim
```

Published with MATLAB® R2019a

```
clear all
clc

% Parameters
syms J M R g To real
% Variables
syms x theta  real
syms dx dtheta real

% Define symbolic variable q for the generalized coordinates
% x and theta
q  = [x; theta];
% Define symbolic variable dq for the derivatives
% of the generalized coordinates
dq = [dx; dtheta];
% Write the expressions for the position of
% the center of the ball:
p = [x*cos(theta) - R*sin(theta); 0; x*sin(theta) + R*cos(theta)];

% Kinetic energy
T = 0.5*J*dtheta^2; % kinetic energy of beam

dp = jacobian(p,q);
T  = T + 0.5*M*dx^2; %((dx+dtheta*R)^2 + (dtheta*x)^2) ; % add linear
 kinetic energy of ball

% I     = 0.4*M*R^2; % inertia in rotation of ball
% omega = dtheta + dx/R; % angular velocity of ball

T  = T + 0.5*(2/5*M*R+M*p'*p)*dtheta^2 + 0.5*(2/5*M*R)*(dx/R)^2; % add
 rotational kinetic energy of ball

T = simplify(T);

% Potential energy
V = M*g*(R*cos(theta)-x*sin(theta));

% Generalized forces
Q = [0; To];

% Lagrangian
Lag = T - V;

Lag_q = simplify(jacobian(Lag,q)).';
Lag_qdq = simplify(jacobian(Lag_q.',dq));
Lag_dq = simplify(jacobian(Lag,dq)).';
Lag_dqdq = simplify(jacobian(Lag_dq.',dq));

% The equations have the form W*q_dotdot = RHS, with
W = Lag_dqdq;
RHS = Q + simplify(Lag_q - Lag_qdq*dq);
```

```
state = [q;dq];
param = [J; M; R; g];

matlabFunction(p, 'file','BallPosition','vars',{state,param});
matlabFunction(W,RHS, 'file','BallAndBeamODEMatrices','vars',
{state,To,param});
```

Published with MATLAB® R2019a

```
function state_dot = PendulumDynamics(state, parameters)

    q = state(1:3);
    q_dot = state(4:6);
    F = -10*q(1) - q_dot(1);
    [W, RHS] = PendulumODEMatrices(state, F, parameters);

    state_dot = [q_dot; W\RHS];

end
```

```
Not enough input arguments.

Error in PendulumDynamics (line 3)
    q = state(1:3);
```

Published with MATLAB® R2019a

```
clear all
close all
clc

% Parameters and initial states
tf = 45;

m = 1;
M = 1;
L = 1;
g = 9.81;
x0 = 0;
theta1_0 = pi/4;
theta2_0 = pi/2;
q = [x0; theta1_0; theta2_0];
dq = zeros(3, 1);

state = [q;dq];
parameters = [m;M;L;g];

% Simulation
try

    %%%%%% MODIFY THE CODE AS YOU SEE FIT

    [tsim,xsim] = ode45(@(t,x)PendulumDynamics(x, parameters),
[0,tf],state);

catch message
    display('Your simulation failed with the following message:')
    display(message.message)
    display(' ')

    % Assign dummy time and states if simulation failed
    tf = 0.1;
    tsim = [0,tf];
    xsim = 0;
end
```

3D animation

```
DoublePlot = true;
FS = 30;
scale = 0.1;

% Create Objects
% Cube
vert{1} = 3*[ -1, -1, 0;  %1
               1, -1, 0;  %2
               1,  1, 0;  %3
              -1,  1, 0;  %4
              -1, -1, 2;  %5
```

```
                1, -1, 2;  %6
                1,  1, 2;  %7
               -1,  1, 2]/2; %8
fac{1} = [1 2 3 4;
          5 6 7 8;
          1 4 8 5;
          1 2 6 5;
          2 3 7 6;
          3 4 8 7];
Lrail = 1.2*max(abs(xsim(:,1)))/scale;
% Rail
a = 1.5;
vert{2} = [-Lrail,-a,-0.1;
           -Lrail, a,-0.1;
            Lrail, a,-0.1;
            Lrail,-a,-0.1];
fac{2} = [1,2,3,4];
% Sphere
[X,Y,Z] = sphere(20);
[fac{3},vert{3},c] = surf2patch(3*X/2,3*Y/2,3*Z/2);
% Animation
tic
t_disp = 0;
SimSpeed = 1;
if run_sim
 while t_disp < tf/SimSpeed
    % Interpolate state
    x_disp   = interp1(tsim,xsim,SimSpeed*t_disp)';

    % Unwrap state. MODIFY
    x = x_disp(1); % position cart
    [p1, p2] = PendulumPosition(x_disp, parameters);

    % Input argument for DrawPendulm
    pos_disp = [x(1);p1(1);0;p1(2);p2(1);0;p2(2)];

    figure(1);clf;hold on
    if DoublePlot
        subplot(1,2,1);hold on
        DrawPendulum( pos_disp, vert, fac, scale);
        campos(scale*[15    15     -70])
        camtarget(scale*[0,0,1.5])
        camva(30)
        camproj('perspective')
        subplot(1,2,2);hold on
    end
    DrawPendulum( pos_disp, vert, fac, scale);
    campos(scale*[1    70     20])
    camtarget(scale*[0,0,1.5])
    camva(30)
    camproj('perspective')
    drawnow
    if t_disp == 0
        display('Hit a key to start animation')
```

```
        pause
        tic
    end
    t_disp = toc;
 end
end
```

```
Undefined function or variable 'run_sim'.

Error in PendulumSimulation (line 75)
if run_sim
```

Published with MATLAB® R2019a

```
clear all
clc

% Parameters
syms m M g L F real
% Variables
syms x  theta1  theta2 real
syms dx dtheta1 dtheta2 real

% Define symbolic variable q for the generalized coordinates
% x, theta1 and theta2
q = [x; theta1; theta2];
% Define symbolic variable dq for the derivatives
% of the generalized coordinates
dq = [dx; dtheta1; dtheta2];
% Write the expressions for the positions of the masses
p{1} = [x - L*sin(theta1); -L*cos(theta1)];
p{2} = p{1} + [-L*sin(theta2); -L*cos(theta2)];

% Kinetic energy of the cart
T = 0.5 * m * dx^2;
% For loop that adds the kinetic energies of the masses
dp = cell(length(p), 1);

for k = 1:length(p)
    dp{k} = jacobian(p{k},q)*dq; % velocity of mass k
    T = T + 0.5 * dp{k}' * M * dp{k}; % add kinetic energy of mass k
end
T = simplify(T);

% Potential energy of the cart
V = 0;
% For loop that adds the potential energies of the masses
for k = 1:length(p)
    V = V + M*g*p{k}(2); % add potential energy of mass k
end
V = simplify(V);

% Generalized forces
Q = [F; 0; 0];

% Lagrangian
Lag = T - V;

Lag_q = simplify(jacobian(Lag,q)).';
Lag_qdq = simplify(jacobian(Lag_q.',dq));
Lag_dq = simplify(jacobian(Lag,dq)).';
Lag_dqdq = simplify(jacobian(Lag_dq.',dq));

% The equations have the form W*q_dotdot = RHS, with
W = Lag_dqdq;
RHS = Q + simplify(Lag_q - Lag_qdq*dq);
```

```
state = [q;dq];
param = [m;M;L;g];

matlabFunction(p{1},p{2}, 'file','PendulumPosition','vars',{state,
 param});
matlabFunction(W,RHS, 'file','PendulumODEMatrices','vars',
{state,F,param});
```

Published with MATLAB® R2019a