Multibody Dynamics B - Assignment 8

July 4, 2019

Short problem statement

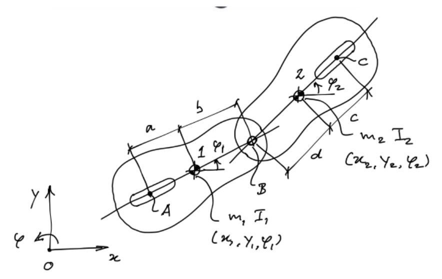

Figure 1: Sketch of EzyRoller

On figure one we can see a sketch of the EzyRoller system that we are working with. It consists of 2 rigid bodies who each have one wheel, body 1 has its wheel at A and body 2 has its wheel at C. The wheels are considered to act as a skates. Furthermore, The two bodies are connected by a hinge in B. The system therefor consists of 2 holonomic constraints and 2 non-holonomic constraints. Our object is to first determine the holonomic constraints and its jacobian, as well as the non holonomic constraints and its jacobian. We then are asked to derive the DAE of the system and formulate 2 functions, one for the coordinate projection method for the constraints on the coordinates and one for the constraints on the velocities. We then are asked to implement a numerical integration method, the Runge-Kutta 4th order method, to determine the motion of the system. Lastly we are asked to implement a reaction-torque acting in the hinge at B that resembles what we would observe when the EzyRoller system is driven in real life.

a)

First we observe that the system is expressed by 6 coordinates: where $x_i = [x_1, y_1, \varphi_1, x_2, y_2, \varphi_2]$, $\dot{x}_i = [\dot{x}_1, \dot{y}_1, \dot{\varphi}_1, \dot{x}_2, \dot{y}_2, \dot{\varphi}_2]$ and $\ddot{x}_i = [\ddot{x}_1, \ddot{y}_1, \ddot{\varphi}_1, \ddot{x}_2, \ddot{y}_2, \ddot{\varphi}_2]$. To determine our equation of motion we use the virtual power law and d'Alembert inertia forces for unconstrained system, and for the constraints we utilize the Lagrange multipliers in the virtual power expression. Now we know that there exist 2 holonomic constraints in the hinge at B. We observe that following conditions must be met

$$x_B = x_1 + bcos(\varphi_1) - x_2 + dcos(phi_2) = 0; \quad y_B = y_1 + bsin(\varphi_1) - y_2 + dsin(\varphi_2) = 0$$

Now we can easily obtain the jacobian matrix, $C_{k,i}$, where $C_{k,i} = \dfrac{\partial C_k}{\partial x_i}$ which gives us

$$C_{k,i} = \begin{bmatrix} 1 & 0 & -bsin(\varphi_1) & -1 & 0 & -dsin(\varphi_2) \\ 0 & 1 & bcos(\varphi_1) & 0 & 1 & dcos(\varphi_2) \end{bmatrix}$$

Now the convective acceleration terms were found by following calculations

$$h_k = C_{k,ij}\dot{x}_i\dot{x}_j = \frac{\partial C_{k,i}\dot{x}_i}{\partial X_j}\dot{x}_j$$

which gives

$$h_k = \begin{bmatrix} -b * \dot{\varphi}_1^2 cos(\check{}varphi_1) - d\dot{\varphi}_2^2 * cos(\varphi_2) \\ -b * \dot{\varphi}_1^2 sin(\check{}varphi_1) - d\dot{\varphi}_2^2 * sin(\varphi_2) \end{bmatrix}$$

b)

Now we have mentioned that the wheels are modeled as skates, so the non-holonomic constraints are concerned with no slip in the parallel direction, i.e. no sideslip. So we have

$$V_{A\perp} = 0; \qquad V_{C\perp} = 0$$

where

$$V_A = V_1 + \omega_1 \times r_1 \qquad V_C = V_2 + \omega_2 \times r_2$$

$$V_A = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\varphi}_1 \end{bmatrix} \times \begin{bmatrix} -acos(\varphi_1) \\ -asin(\varphi_1) \\ 0 \end{bmatrix} \qquad V_c = \begin{bmatrix} \dot{x}_2 \\ \dot{y}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\varphi}_2 \end{bmatrix} \times \begin{bmatrix} ccos(\varphi_2) \\ csin(\varphi_2) \\ 0 \end{bmatrix}$$

$$V_A = \begin{bmatrix} \dot{x}_1 + a\dot{\varphi}_1 sin(\varphi_1) \\ \dot{y}_1 + a\dot{\varphi}_1 cos(\varphi_1) \\ 0 \end{bmatrix} \qquad V_c = \begin{bmatrix} \dot{x}_2 - c\dot{\varphi}_2 sin(\varphi_2) \\ \dot{y}_2 + c\dot{\varphi}_2 cos(\varphi_2) \\ 0 \end{bmatrix}$$

$$V_{A\perp} = V_A \begin{bmatrix} -sin(\varphi_1) \\ cos(\varphi_1) \\ 0 \end{bmatrix} = 0 \qquad V_{C\perp} = V_C \begin{bmatrix} -sin(\varphi_2) \\ cos(\varphi_2) \\ 0 \end{bmatrix} = 0$$

which finally gives the non holonomic constraint equations as

$$V_{A\perp} = -\dot{x}_1 sin(\varphi_1) + \dot{y}_1 cos(\varphi_1) - a\dot{\varphi}_1 = 0$$
$$V_{C\perp} = -\dot{x}_2 sin(\varphi_2) + \dot{y}_2 cos(\varphi_2) + c\dot{\varphi}_2 = 0$$

Now in the same matter as the holonomic constraint we find the jacobian matrix which is

$$S_{m,i} = \begin{bmatrix} -sin(\varphi_1) & cos(\varphi_1) & -a & 0 & 0 & 0 \\ 0 & 0 & 0 & -sin(\varphi_2) & cos(\varphi_2) & c \end{bmatrix}$$

Again the connective term for the non-holonomic constraints can be determined in the same matter as for the holonomic constraints,

$$s_m = S_{ki,j}\dot{x}_i\dot{x}_j$$

which gives us

$$s_m = \begin{bmatrix} -\dot{\varphi}_1(\dot{x}_1 cos(\varphi_1) + \dot{y}_1 sin(\varphi_1)) \\ -\dot{\varphi}_2(\dot{x}_2 cos(\varphi_2) + \dot{y}_2 sin(\varphi_2)) \end{bmatrix}$$

c)

Now we can find our DAE, in matrix form it is given as, see virtual power law with d'Lambert inertia forces and Lagrange multipliers.

$$\begin{bmatrix} M_{ij} & C_{k,i} & S_{m,i} \\ C_{k,j} & 0_{kk} & 0_{km} \\ S_{m,j} & 0_{mk} & 0_{mm} \end{bmatrix} \begin{bmatrix} \ddot{x}_j \\ \lambda_k \\ \lambda_m \end{bmatrix} = \begin{bmatrix} F_i \\ -h_k \\ -s_m \end{bmatrix}$$

The mass matrix is defined as

$$M_{ij} = diag(\begin{bmatrix} m_1 & m_1 & I_1 & m_2 & m_2 & I_2 \end{bmatrix})$$

and the force as

$$F_i = \begin{bmatrix} \sum F_{x1} \\ \sum F_{y1} \\ \sum M_1 \\ \sum F_{x2} \\ \sum F_{y2} \\ \sum M_2 \end{bmatrix}$$

and the other terms, the constraints and their convective terms, can be seen above, the DAE Matlab function can be seen in Apendix A.

d)

Since we have constraints we have created a problem of drift during the numerical integration, in the state spaces. To remove these drifts we introduce the coordinate projection method. We use the Gauss-Newton method to implement our coordinate projection. Now we want to correct the positions such that

$$q_{n+1} = \bar{q}_{n+1} + \delta q_{n+1}$$

where

$$y_{n+1} = \begin{bmatrix} \bar{q}_{n+1} \\ \dot{\bar{q}}_{n+1} \end{bmatrix}$$

where we use

$$C_c(\bar{q}_{n+1}) + C_{c,k}(\bar{q}_{n+1})\delta q_{n+1} = 0$$

which gives us

$$\delta q_{n+1} = -C_k^T (C_k C_k^T)^{-1} C$$

where

$$C_k = C_{c,k}(\bar{q}_{n+1})$$
$$C = C_c(\bar{q}_{n+1})$$

Therefor we have found the error and implement this in the first equation above. This is done until the constraint equations are less then the defined tolerance, we iterate this in a while loop, or until you hit the maximum iteration count, here it is 10^{-12} and 10 respectively.
For velocities this is similar except we don't need to iterate that process since these are linear equations, so we have a linear least square problem, we define

$$\dot{q}_{n+1} = \dot{\bar{q}}_{n+1} + \delta \dot{q}_{n+1}$$

where

$$C_{c,k}(\bar{q}_{n+1})\dot{\bar{q}}_{n+1} + C_{c,k}(\bar{q}_{n+1})\delta \dot{q}_{n+1} = 0$$

which gives

$$\delta \dot{q}_{n+1} = -C_k^T (C_k C_k^T)^{-1} C_k * \dot{\bar{q}}_{n+1}$$

where

$$C_k = C_{c,k}(\bar{q}_{n+1})$$

We have now corrected both the velocities and positions for the constraints. Note that C and C_k are representitive of all the constraints, both holonomic and non-holonomic. The Matlab functions for the coordinate projection of positions and velocities can be found in appendix A.

e)

Now we look at the motion of the EzyRoller when it has no external forces or torques, only initial velocities. The numerical integration method that is used is the Runge-Kutta 4th order method which follows the scheme

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + h/2, y_n + h/2 * k_1)$$
$$k_3 = ff(t_n + h/2, y_n + h/2 * k_2)$$
$$k_4 = ff(t_n + h, y_n + h * k_3)$$
$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Our parameters are defined as follows $a = 0.2, b = 0.6, c = 0.1, d = 0.1, m_1 = 46, I_1 = 10,, m_2 = 1, I_2 = 0.005$ and our initial conditions are set as $(x_1, y_1, \varphi_1, x_2, y_2, \varphi_2) = (a, 0, 0, a+b, d, \pi/2)$ and the velocities are $\dot{x}_1 = 1$ and $\dot{\varphi}_1 = 0$. The other initial conditions are found by satisfying the constraint equations i.e. we have four constraint equations for the velocities which are

$$-\dot{x}_1 sin(\varphi_1) + \dot{y}_1 cos(\varphi_2) - a\dot{\varphi}_1$$
$$-\dot{x}_2 sin(\varphi_2) + \dot{y}_2 cos(\varphi_2) + c\dot{\varphi}_2$$
$$\dot{x}_1 - \dot{x}_2 - b\dot{\varphi}_1 sin(\varphi_1) - c\dot{\varphi}_2 sin(\varphi_2)$$
$$\dot{y}_1 - \dot{y}_2 + b\dot{\varphi}_1 cos(\varphi_1) + c\dot{\varphi}_2 cos(\varphi_2)$$

Solving these 4 equations for the 4 unknowns gave us $\dot{x}_2 = 0.5, \dot{y}_2 = 0, \dot{y}_1 = 0 and \dot{\varphi}_2 = 5$. Now plotting the motion for these condition for one second gave us

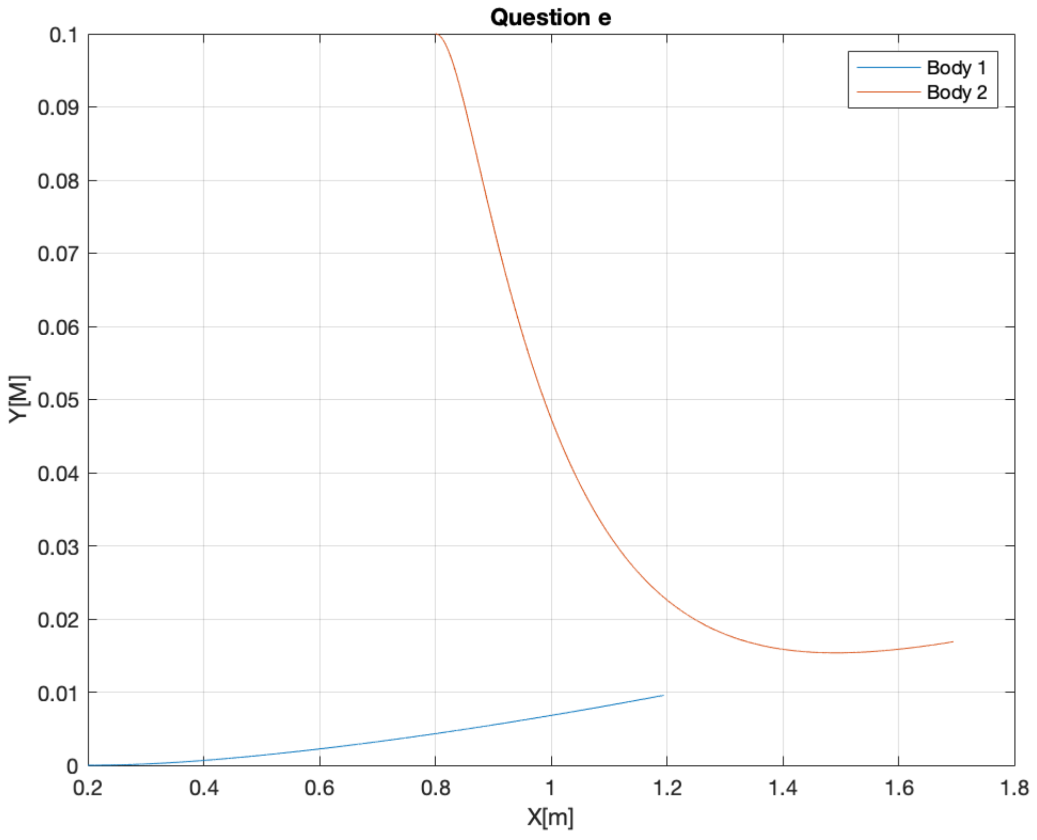

Figure 2: Motion of the EzyRoller for one second

Like we can see body 2 on the EzyRoller starts at a 90 degree angle and body 1 is horizontal. But since there is no vertical velocity body 2 kinda corrects itself and starts moving horizontally, however in that motion the whole body of the EzyRoller aligns itself such that now it is heading a little diagonally which is what you would expect, because the tires try to align them self to be on the same line, which causes the EzyRoller to move a little bit up.

f)

Now we implement a action-reaction torque, $M = M_0 cos(\omega t)$ where $M_0 = 2$ [Nm] and $\omega = \pi$ [rad/s]. The initial conditions also change such that the two bodies are at rest and aligned on the x axis, i.e. $(x_1, y_1, \varphi_1, x_2, y_2, \varphi_2) = (0, 0, 0, 0, 0, \pi)$ and all velocities are zero. We only add to the force vector the torques $-M$ and M on body 1 and body 2 respectively, i.e. the third and sixth position of the force vector. Now the path of body A and body C can be seen in figure 3 below, with the addition of the torques.

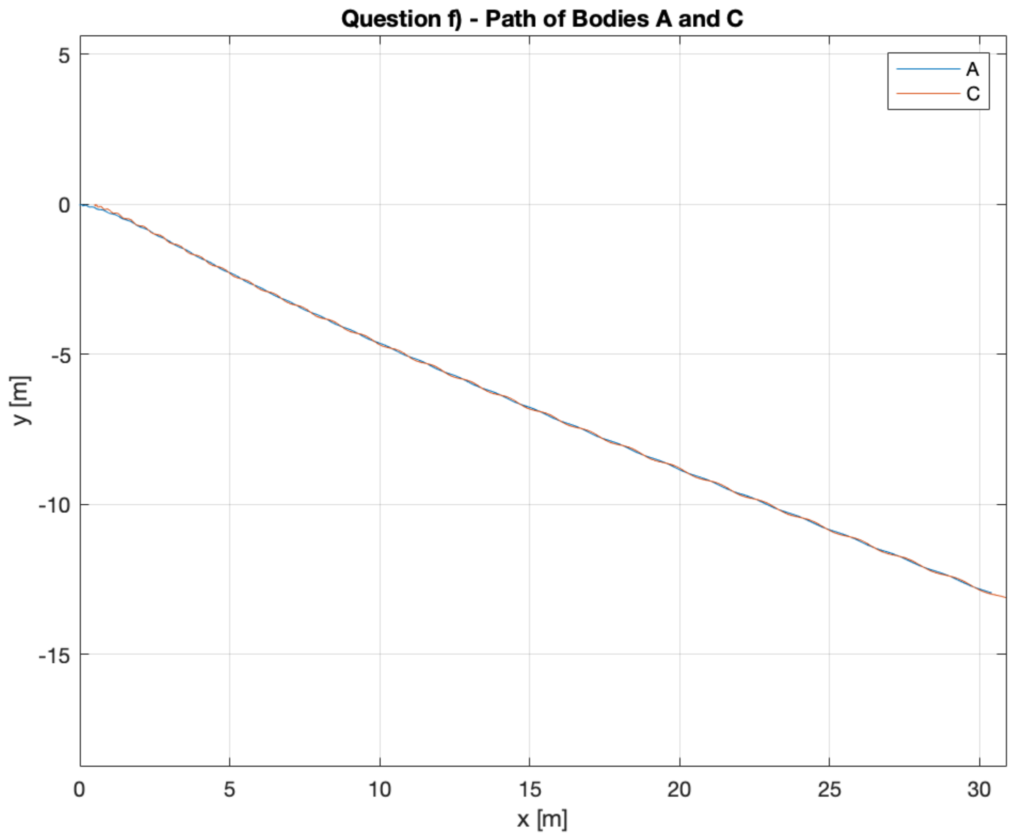

Figure 3: Motion of the EzyRoller for 60 seconds

Though we can't see it very well at this figure but if looked closely it can be observed that the bodies move like a sinusoidal, due to the torque, and are exactly out of phase with one another in the beginning but converge to a more straight path the longer it goes. We see that the body moves downwards, this is due to how we define the torques, i.e. we defin a negative torque on body 2 and a positive reactive torque on body 1. We also observe that the though the initial conditions are strictly horizontal the body moves diagonally, similarly like in results for f). However, like in f, it looks to align the tires, however this takes some time, so we get a little bit of a transient response, this results in the body turning diagonally, however the transient response fades, and eventually disappears, which results in the system moving straight, i.e. it doesn't deviate from the direction.

g)

Now we plotted the speed and angular velocity the CoM of body 1, this can be seen in figure 4 below

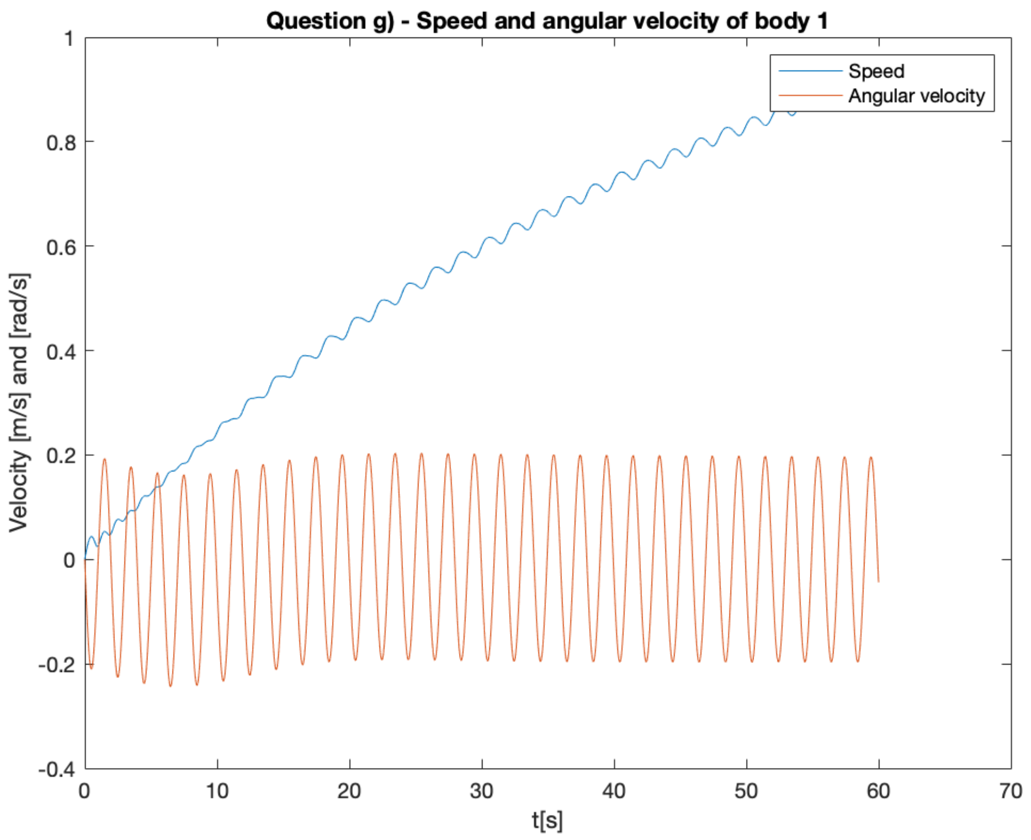

Figure 4: Speed and angular velocity for body 1 of the EzyRoller system.

We observe that the angular velocity has a small shift in the first 20 seconds, but gets to its initial rhythm again. This is likely the effect of the transient response of the system, when it is trying to first move forwards. Another interesting aspect is that the speed acts in the same manner as the torque, i.e it is increasing but has a sinusoidal pattern, this is of course a direct consequence to how the system is moving forwards, i.e. with the torques. Now we see that the speed seems to be reaching itðs peek, i.e. the maximum speed the EzyRoller is able to get with the applied torque, since the velocity is tapering off at the end.

h)

Now we look at the kinetic energy of the system, this can be seen in figure 5 below.

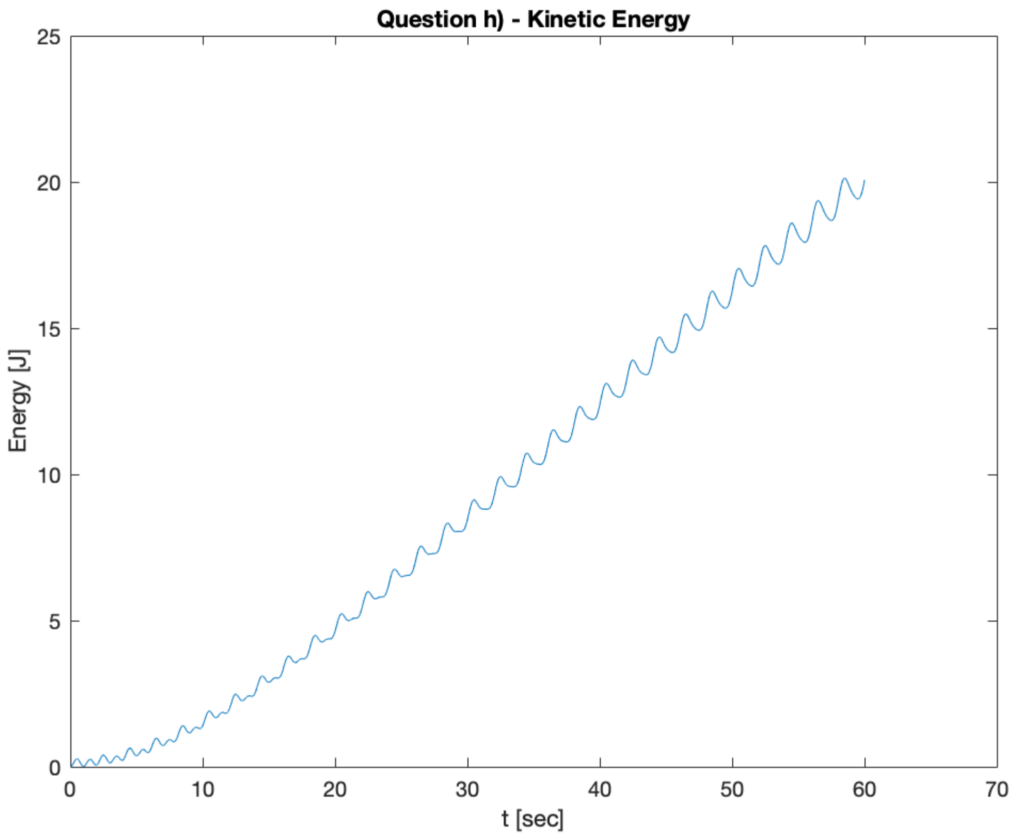

Figure 5: Kinetic energy of the EzyRoller system.

We see that the kinetic energy has the shape of an exponential function. It has, similarly to the other figures, the same pattern as the torque, i.e. the sinusoidal rhythm. This is of course expected after seeing the velocity graph, since the kinetic energy is dependent on the velocity. Now the kinetic energy won't increase endlessly, it will reach a stable point when the system will reach its maximum velocity, so if w let the simulation run longer the kinetic energy plot will taper off and have a sinusoidal rhythm around a certain value.

i)

Now we plot and look at the applied torque and the relative angular velocity in the hinge at B, this can be seen in figure 6 below.

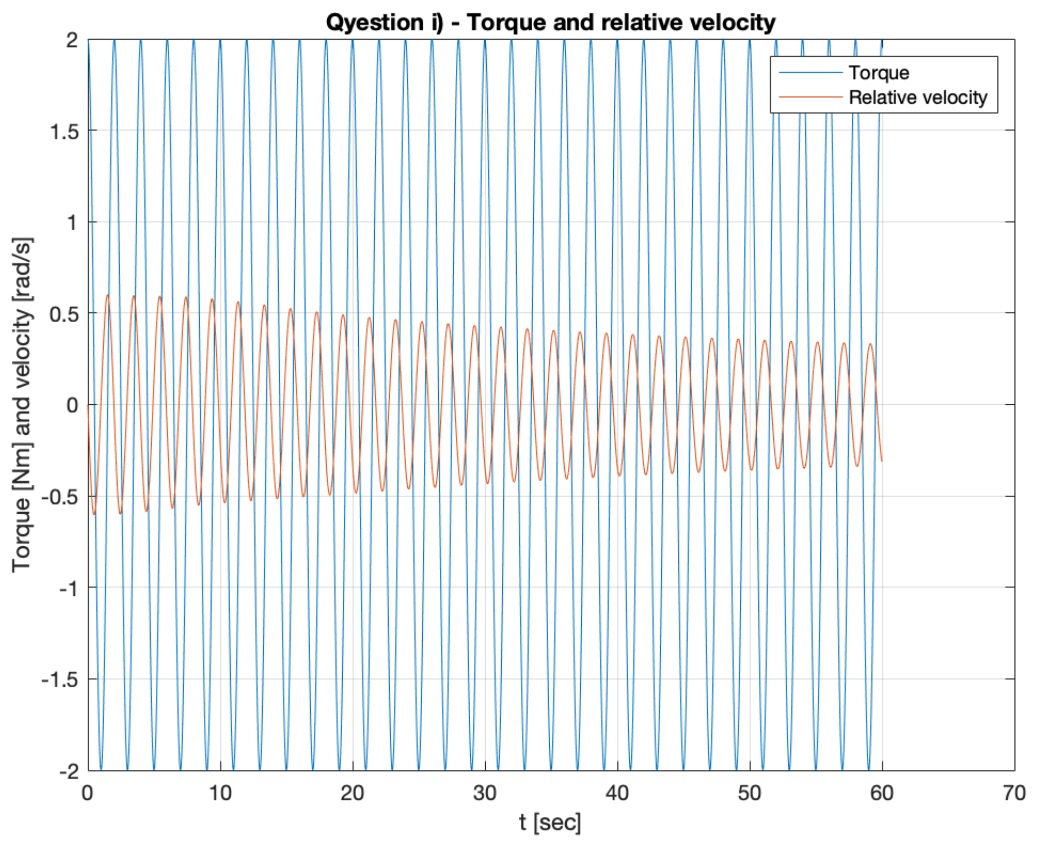

Figure 6: Applied torque on the EzyRoller system and the relative angular velocity of the hinge at B.

Now nothing particularly interesting is happening with the torque, since it always keeps its sinusoidal rhythm and never changes, in frequency or in magnitude. However we can see that the relative angular velocity is all over the place. First the frequency is changing, i.e. the time between one period is shorter in the beginning, this is due to the transient response, this is obvious in the first 5 seconds. Furthermore it looks like its damping, with the transient response dying out. This is likely what is affecting most of the other observed deviations. However the relative angular velocity is damping because the system is reaching stability, I mentioned earlier that body 1 and body 2 are completely out of phase but converge to a straighter line, i.e. the tires are trying to align, this is what we are observing with the relative angular velocity. When the transient has died out the two bodies are converging to align their tires, similarly to what we had talked about for question e, and observed with the path, i.e. the reason they make a turn at first and then become quite straight.

i)

Now lastly we plot the work done along with the kinetic energy, this can be observed in figure 7 here below.

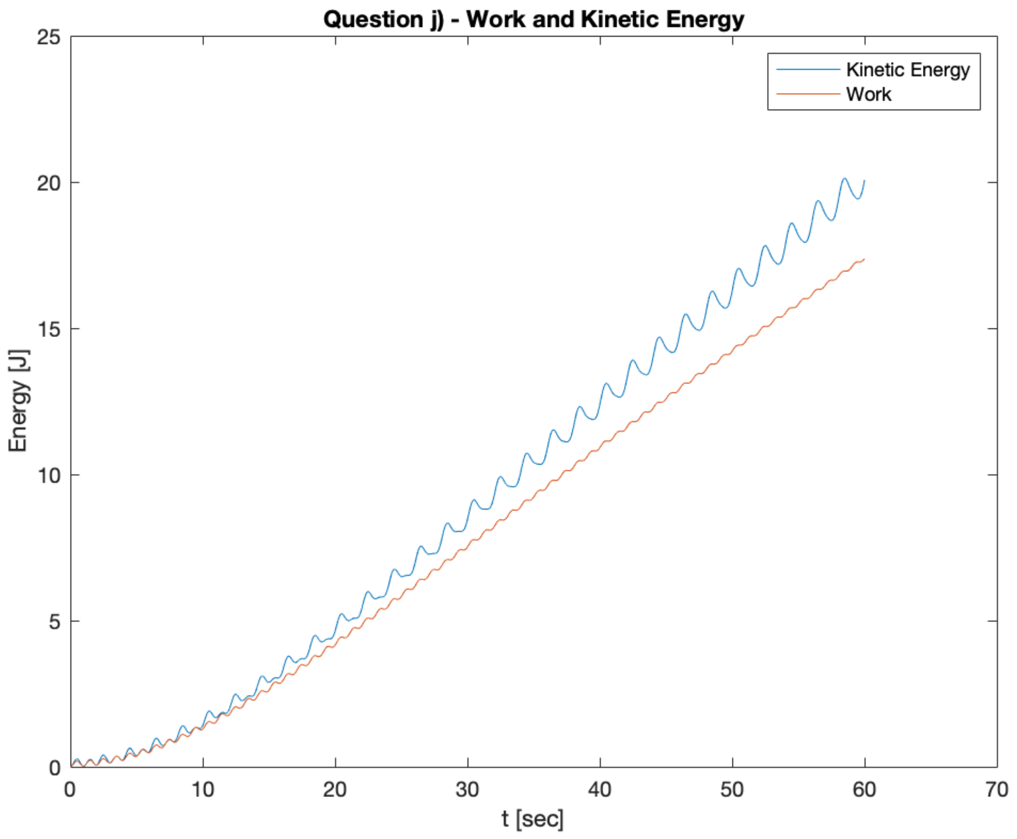

Figure 7: Kinetic energy and the work done on the system.

Now what we see is that the work and kinetic energy are not the same, however they should be. This is due to the fact that when we numerically integrate we get an error that accumulates. If we lower the step size the work converges closer and closer to the kinetic energy, though it will never really be exactly the same, you would need to have a really small step size. This is what happens also for the velocities and positions if we wouldn't use the coordinate projection method, thus we can observe the importance of integrating those functions into our solution.

Appendix A

Matlab code

```matlab
clc;clear;

%% setup

syms x1 y1 x2 y2 phi1 phi2;
syms dx1 dy1 dx2 dy2 dphi1 dphi2;
syms ddx1 ddy1 ddx2 ddy2 ddphi1 ddphi2;
%syms a b c d;
%syms m1 m2 I1 I2

global a b c d m1 m2 I1 I2

a = 0.2; b = 0.6; c = 0.1; d = 0.1; m1 = 46; m2 = 1; I1 = 10; I2 = 0.005;


x = [x1; y1; phi1; x2; y2; phi2];
dx = [dx1; dy1; dphi1; dx2; dy2; dphi2];

% holonomic constraints
C1 = x1 + b*cos(phi1) - x2 + d*cos(phi2);
C2 = y1 + b*sin(phi1) - y2 + d*sin(phi2);
C = [C1; C2];

Cd = jacobian(C, x);
Cd = simplify(Cd);

% convective terms
Ch = jacobian(Cd*dx, x)*dx;
Ch = simplify(Ch);

% nonholonomic constraints
S1 = -dx1*sin(phi1) + dy1*cos(phi2) - a*dphi1;
S2 = -dx2*sin(phi2) + dy2*cos(phi2) + c*dphi2;
S = [S1; S2];

% convective terms
Sh = [-dphi1*dx1*cos(phi1)-dphi1*dy1*sin(phi1);
      -dphi2*dx2*cos(phi2)-dphi2*dy2*sin(phi2)];

% DAE
M = diag([m1 m1 I1 m2 m2 I2]);
Sd = [-sin(phi1) 0; cos(phi1) 0; -a 0; 0 -sin(phi2); 0 cos(phi2); 0 c];
A = [M Cd.' Sd; Cd zeros(2,4); Sd.' zeros(2,4)];
B = [zeros(6,1); -Ch; -Sh];
Acc = simplify(A\B);
matlabFunction(Acc,'File','Acceleration');
matlabFunction(Cd, 'File', 'HoC');
matlabFunction(Ch, 'File', 'HoCc');
matlabFunction(Sd, 'File', 'nHoC');
matlabFunction(Sh, 'File', 'nHoCc');
matlabFunction(C, 'File', 'Cphw8');


%% Question E)-J)
% Initial conditions

x1 = 0;
```

```
y1 = 0;
phi1 = 0;
phi2 = pi;
dx1 = 0;
dy1 = 0;
dphi1 = 0;
dphi2 = 0;
x2 = x1 + b * cos(phi1) + d * cos(phi2);
y2 = y1 + b * sin(phi1) + d * sin(phi2);
dx2 = dx1 - b * dphi1 * sin(phi1) - d * dphi2 * sin(phi2);
dy2 = dy1 + b * dphi1 * cos(phi1) + d * dphi2 * cos(phi2);
y0 = [x1, y1, phi1, x2, y2, phi2, dx1, dy1, dphi1, dx2, dy2, dphi2].';

% Setup
t0 = 0;
tend = 60;
td = 0.0001; % Stepsize

% Define torque and force vector
M0 = 2;
omega = pi;
M1 = M0 * cos(omega *td);
F = [0, 0, M1, 0, 0, -M1].';
nstep = (tend - t0) / td +1 ;
t = t0;
y = y0;
T = zeros(nstep,1);
Y = zeros(nstep,12);
W = zeros(nstep,1);
Es = zeros(nstep,1);
T(1) = t;
Y(1,:) = y.';

%% Numerical integration, Runge-Kutta 4th order and coordinate projection
for i = 2 : nstep
    [t,y] = rk4step(@odedae2,t,y,td);
    %y_start = y0;
    y = projectSpeed(t,y);
    %y = projectposition(t,y);
    f = AppliedForcesE(t,y);
    T(i) = t;
    Y(i,:) = y.';
end

%% Post Processing
% Positions and velocities
X1 = Y(:,1); Y1 = Y(:,2); PHI1 = Y(:,3);
X2 = Y(:,4); Y2 = Y(:,5); PHI2 = Y(:,6);
X1D = Y(:,7); Y1D = Y(:,8); PHI1D = Y(:,9);
X2D = Y(:,10); Y2D = Y(:,11); PHI2D = Y(:,12);

% Position of wheels
XA = X1 - a * cos(PHI1);
YA = Y1 - a * sin(PHI1);
XC = X2 + c * cos(PHI2);
YC = Y2 + c * sin(PHI2);

% Work done
for j = 1:length(T)
    if j==1
        W(j) = 0;
    else
        W(j) = 2*cos(pi*T(j))*(PHI1(j)-PHI1(j-1))-2*cos(pi*T(j))*(PHI2(j)-PHI2(j-1));
    end
end

% Kinetic energy
E_kin1 = 1/2 * m1 * (X1D.^2 + Y1D.^2) + 1/2 * I1 * PHI1D.^2;
E_kin2 = 1/2 * m2 * (X2D.^2 + Y2D.^2) + 1/2 * I2 * PHI2D.^2;
E_kin = E_kin1 + E_kin2;

%% Plots
figure(1)
plot(X1,Y1)
hold on
plot(X2,Y2)

axis equal
title('Question f) - Path of Bodies A and C')
legend('A','C')
xlabel('x [m]')
ylabel('y [m]')
grid on

figure(2)
speed = sqrt(X1D.^2 + Y1D.^2);
plot(T,speed)
hold on
plot(T,PHI1D)
title('Question g) - Speed and angular velocity of body 1');
legend('Speed', 'Angular velocity');
xlabel('t[s]');
ylabel('Velocity [m/s] and [rad/s]');


figure(3)
M=2*cos(omega*T);
plot(T,M)
hold on
plot(T,PHI2D - PHI1D)
xlabel('t [sec]')
ylabel('Torque [Nm] and velocity [rad/s]')
legend('Torque','Relative velocity')
title('Qyestion i) - Torque and relative velocity')
grid on

figure(4)
plot(T, E_kin)
hold on
plot(T, cumsum(W))
title('Question j) - Work and Kinetic Energy')
xlabel('t [sec]')
ylabel('Energy [J]')
legend('Kinetic Energy','Work')

figure(5)
plot(T, E_kin)
xlabel('t [sec]')
ylabel('Energy [J]')
title('Question h) - Kinetic Energy')
```

```matlab
function q_corr = projectposition(t,q)
% setup for while loop
tol = 1e-12;
i = 0;
maxit = 10;

x1 = q(1);
y1 = q(2);
phi1 = q(3);
x2 = q(4);
y2 = q(5);
phi2 = q(6);

x1d = q(7);
y1d = q(8);
phi1d = q(9);
x2d = q(10);
y2d = q(11);
phi2d = q(12);
dxvec = reshape (q(7:12),6,1);

q_corr=q;

s = Cphw8(phi1,phi2,x1,x2,y1,y2);

% Find corrected positions
while (tol<max(abs(s)) && i<maxit)
    Dv = HoC(phi1,phi2);
    s = Cphw8(phi1,phi2,x1,x2,y1,y2);

    Δ_p = -Dv.'*((Dv*Dv.')\s);
    q_corr(1:6) = q_corr(1:6) + Δ_p;

    i = i+1;

end

end
```

```matlab
function ycorr = projectSpeed(t,ypred)
%UNTITLED8 Summary of this function goes here
%   Detailed explanation goes here
[s,Ds,h] = constraint(ypred);
dydot = -Ds.'*((Ds*Ds.')\s); % [dx1 dy1 dp1 dx2 dy2 dp2]
ycorr = ypred;
ycorr(7:12) = ypred(7:12) + dydot(1:6);
end
```

```matlab
function [t,y] = rk4step(odefun,t0,y0,h)
  t=t0;
  y=y0;
  k1=feval(odefun,t,y);
  k2=feval(odefun,t+h/2,y+(h/2).*k1);
  k3=feval(odefun,t+h/2,y+(h/2).*k2);
  k4=feval(odefun,t+h,y+h.*k3);
  t=t+h;
  y=y+(h/6).*(k1+2.*(k2+k3)+k4);
```

```matlab
function f = AppliedForcesE(t,yvec)
global a b c d m1 m2 I1 I2 CD FC F M
% copy state vector into meaningful variables
x1 = yvec(1);
y1 = yvec(2);
phi1 = yvec(3);
x2 = yvec(4);
y2 = yvec(5);
phi2 = yvec(6);
x1d = yvec(7);
y1d = yvec(8);
phi1d = yvec(9);
x2d = yvec(10);
y2d = yvec(11);
phi2d = yvec(12);

M0 = 2;
omega = pi;
M1 = M0 * cos(omega *t);
f = zeros(6,1);
F = [0, 0, 0, 0, 0, 0].';  % e)
% add torques questions: f)-j)
f(1) = F(1);
f(2) = F(2);
f(3) = M1;
f(4) = F(4);
f(5) = F(5);
f(6) = -M1;
end
```

```matlab
function [s,Ds, h] = constraint(yvec)
% Holonomic and non-holonomic constraints s, Jacobian Ds, convective term h
global a b c d m1 m2 I1 I2 FC

x1 = yvec(1);
y1 = yvec(2);
phi1 = yvec(3);
x2 = yvec(4);
y2 = yvec(5);
phi2 = yvec(6);

dx1 = yvec(7);
dy1 = yvec(8);
dphi1 = yvec(9);
dx2 = yvec(10);
dy2 = yvec(11);
dphi2 = yvec(12);

dxvec = reshape(yvec(7:12),[6,1]);

Ds = [1, 0, -b*sin(phi1), -1, 0, -d*sin(phi2);
      0, 1, b*cos(phi1), 0, -1, d*cos(phi2);
      -sin(phi1), cos(phi1), -a, 0, 0, 0;
      0, 0, 0, -sin(phi2), cos(phi2), c
      ];

 % constraint equations
 s = Ds*dxvec;

 % convective terms for holonomic and non-holonomic
h = [-b*cos(phi1)*dphi1^2 - d*cos(phi2)*dphi2^2;
-b*sin(phi1)*dphi1^2 - d*sin(phi1)*dphi2^2;
-dphi1*(dx1*cos(phi1) + dy1*sin(phi1));
-dphi2*(dx2*cos(phi2) + dy2*sin(phi2))];

end
```

```
%% e)
function dyvec = odedae(t,yvec)

global a b c d m1 m2 I1 I2 M FC

x1 = yvec(1);
y1 = yvec(2);
phi1 = yvec(3);
x2 = yvec(4);
y2 = yvec(5);
phi2 = yvec(6);

dx1 = yvec(7);
dy1 = yvec(8);
dphi1 = yvec(9);
dx2 = yvec(10);
dy2 = yvec(11);
dphi2 = yvec(12);

M = diag([m1 m1 I1 m2 m2 I2]);
[s,Ds,h] = constraint(yvec);
DAE = [M Ds.';
       Ds zeros(4)];

% determine the applied forces
f = AppliedForces(t,yvec);
rhs = [f; -h];
lhs = DAE\rhs;
dyvec = yvec;

% copy velocties and accelerations into dy
dyvec(1:6) = yvec(7:12);
dyvec(7:12) = lhs(1:6);
% copy constraint force in global FC
FC = lhs(7:10);
end
```

```
%% f)-j)
function yvecd = odedae2(t,yvec)

global a b c d m1 m2 I1 I2 CD FC
% copy state vector into meaningful variables
x1 = yvec(1);
y1 = yvec(2);
phi1 = yvec(3);
x2 = yvec(4);
y2 = yvec(5);
phi2 = yvec(6);
dx1 = yvec(7);
dy1 = yvec(8);
```

```
dphi1 = yvec(9);
dx2 = yvec(10);
dy2 = yvec(11);
dphi2 = yvec(12);

A =[         46,          0,                 0,           0,          0,              0, ...
                 1,                0, -sin(phi1),          0;
             0,         46,                0,           0,          0,              0, ...
                               0,                1,  cos(phi1),          0;
             0,          0,               10,           0,          0,              0, ...
                -(3*sin(phi1))/5, (3*cos(phi1))/5,       -1/5,          0;
             0,          0,                0,           1,          0,              0, ...
                              -1,                0,          0, -sin(phi2);
             0,          0,                0,           0,          1,              0, ...
                               0,               -1,          0,  cos(phi2);
             0,          0,                0,           0,          0,          1/200,    ...
                -sin(phi2)/10,    cos(phi2)/10,          0,       1/10;
             1,          0, -(3*sin(phi1))/5,          -1,          0, -sin(phi2)/10, ...
                               0,                0,          0,          0;
             0,          1,  (3*cos(phi1))/5,           0,         -1,  cos(phi2)/10, ...
                               0,                0,          0,          0;
  -sin(phi1), cos(phi1),             -1/5,           0,          0,              0, ...
                      0,                0,          0,          0;
             0,          0,                0, -sin(phi2), cos(phi2),           1/10, ...
                               0,                0,          0,          0];
 B =[ 0;
                                                 0;
                                                 0;
                                                 2*cos(pi*t);
                                                 0;
                                                 -2*cos(pi*t);
  (3*cos(phi1)*dphi1^2)/5 + (cos(phi2)*dphi2^2)/10;
  (3*sin(phi1)*dphi1^2)/5 + (sin(phi2)*dphi2^2)/10;
         dphi1*dy1*sin(phi1) + dphi1*dx1*cos(phi1);
         dphi2*dy2*sin(phi2) + dphi2*dx2*cos(phi2)];

  DAE = A\B;
yvecd = yvec;
% copy velocties and accelerations into yd
yvecd(1:6) = yvec(7:12);
yvecd(7:12) = DAE(1:6);
% Copy constraint force in global FC
FC = DAE(4);
end
```

```
%% with odedea for e)
function f = AppliedForces(t,yvec)

global a b c d m1 m2 I1 I2 M

omega = pi;
f = zeros(6,1);
end
```