Laboratory Work

Algorithms for Stabilization of Motion Trajectories of Dynamic Systems

Student

Zein Alabedeen Barhoum (345094)

Question 1:

In this task, we want the robot (material point) to follow three trajectory sequentially, first follow the first one, then follow the second then the third. The trajectories are given implicitly as equations of $q=[x,y]^T$ as follows:

$$\phi_1(q)=(x+7)^2+(y+5)^2-6.25=0$$
$$\phi_2(q)=-\sin(\frac{\pi}{3})x+\cos(\frac{\pi}{3})y+1.4=0$$
$$\phi_3(q)=(x-10)^2+(y-10)^2-36=0$$

The Jacobian matrix J corresponding to each on is as follows,

$$J_i(x,y)=\begin{bmatrix}\frac{\delta\psi_i}{\delta x} & \frac{\delta\psi_i}{\delta y}\\ \frac{\delta\phi_i}{\delta x} & \frac{\delta\phi_i}{\delta y}\end{bmatrix}$$

And can be rewritten as

$$J_i(x,y)=\begin{bmatrix}\frac{\delta\phi_i}{\delta y} & \frac{-\delta\phi_i}{\delta x}\\ \frac{\delta\phi_i}{\delta x} & \frac{\delta\phi_i}{\delta y}\end{bmatrix}$$

Therefore, the Jacobian matrices for the three trajectories are:

$$J_1(x,y)=\begin{bmatrix}2(y+5) & -2(x+7)\\ 2(x+7) & 2(y+5)\end{bmatrix}$$
$$J_2(x,y)=\begin{bmatrix}\cos(\frac{\pi}{3}) & \sin(\frac{\pi}{3})\\ -\sin(\frac{\pi}{3}) & \cos(\frac{\pi}{3})\end{bmatrix}$$
$$J_3(x,y)=\begin{bmatrix}2(y-10) & -2(x-10)\\ 2(x-10) & 2(y-10)\end{bmatrix}$$

The control law applied is the following:

$$F=m(\dot{\bar{v}}-k_q(\dot{q}-\bar{v}))$$

The control signal $\bar{v}$ represents the vector of desired velocities.

This vector is constructed of two control signals, one to stabilize the trajectory and one to control the desired speed.

$$\bar{v}=u_s+u_e$$

The signal u_s is of the form

$$u_s = R_O^I J^{-1}(q) \begin{bmatrix} \dot{s}_d \\ 0 \end{bmatrix}$$

In my case, the desired velocity os $\dot{s_d} = 2$; moreover,

$$R_O^I = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{4}) & \sin(\frac{\pi}{4}) \\ -\sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

By applying the Jacobian matrices we get:

$$u_{s1} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} 2(y+5) & -2(x+7) \\ 2(x+7) & 2(y+5) \end{bmatrix}^{-1} \times \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \frac{-\sqrt{2}}{2(x^2+14x+y^2+10y+74)} \begin{bmatrix} x-y+2 \\ x+y+12 \end{bmatrix}$$

$$u_{s2} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} \cos(\frac{\pi}{3}) & \sin(\frac{\pi}{3}) \\ -\sin(\frac{\pi}{3}) & \cos(\frac{\pi}{3}) \end{bmatrix}^{-1} \times \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.9319 \\ 0.5176 \end{bmatrix}$$

$$u_{s3} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} 2(y-10) & -2(x-10) \\ 2(x-10) & 2(y-10) \end{bmatrix}^{-1} \times \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \frac{-\sqrt{2}}{2(x^2-20x+y^2-20y+200)} \begin{bmatrix} x-y \\ x+y-20 \end{bmatrix}$$

The second term of the control signal is u_e and can be calculated as:

$$u_e = -k_e \phi(q) \begin{bmatrix} \frac{\delta \phi(q)}{\delta x} \\ \frac{\delta \phi(q)}{\delta y} \end{bmatrix}$$

By substituting the given trajectories , we get the following:

$$u_{e1} = -k_e((x+7)^2+(y+5)^2-6.25) \begin{bmatrix} 2(x+7) \\ 2(y+5) \end{bmatrix}$$

$$u_{e2} = -k_e(-\sin(\frac{\pi}{3})x+\cos(\frac{\pi}{3})y+1.4) \begin{bmatrix} -\sin(\frac{\pi}{3}) \\ \cos(\frac{\pi}{3}) \end{bmatrix}$$

$$u_{e3} = -k_e((x-10)^2+(y-10)^2-36) \begin{bmatrix} 2(x-10) \\ 2(y-10) \end{bmatrix}$$

Therefore, the control signal of required velocities is calculated for the three trajectories as:

$$\bar{v}_1 = u_{s1} + u_{e1} = \frac{-\sqrt{2}}{2\left(x^2+14x+y^2+10y+74\right)}\begin{bmatrix} x-y+2 \\ x+y+12 \end{bmatrix} - k_e\left((x+7)^2+(y+5)^2-6.25\right)\begin{bmatrix} 2(x+7) \\ 2(y+5) \end{bmatrix}$$

$$\bar{v}_2 = u_{s2} + u_{e2} = \begin{bmatrix} 1.9319 \\ 0.5176 \end{bmatrix} - k_e\left(-\sin\left(\frac{\pi}{3}\right)x + \cos\left(\frac{\pi}{3}\right)y + 1.4\right)\begin{bmatrix} -\sin\left(\frac{\pi}{3}\right) \\ \cos\left(\frac{\pi}{3}\right) \end{bmatrix}$$

$$\bar{v}_3 = u_{s3} + u_{e3} = \frac{-\sqrt{2}}{2\left(x^2-20x+y^2-20y+200\right)}\begin{bmatrix} x-y \\ x+y-20 \end{bmatrix} - k_e\left((x-10)^2+(y-10)^2-36\right)\begin{bmatrix} 2(x-10) \\ 2(y-10) \end{bmatrix}$$

Then the applied force is calculated as:

$$F = m\left(\dot{\bar{v}} - k_q\left(\dot{q} - \bar{v}\right)\right)$$

The control constant parameters were set to $k_e = 10, k_q = 7$

To check the quality of control, the error is calculated as $e = \phi(q)$

The following Simulink model was created to check the performance of the control

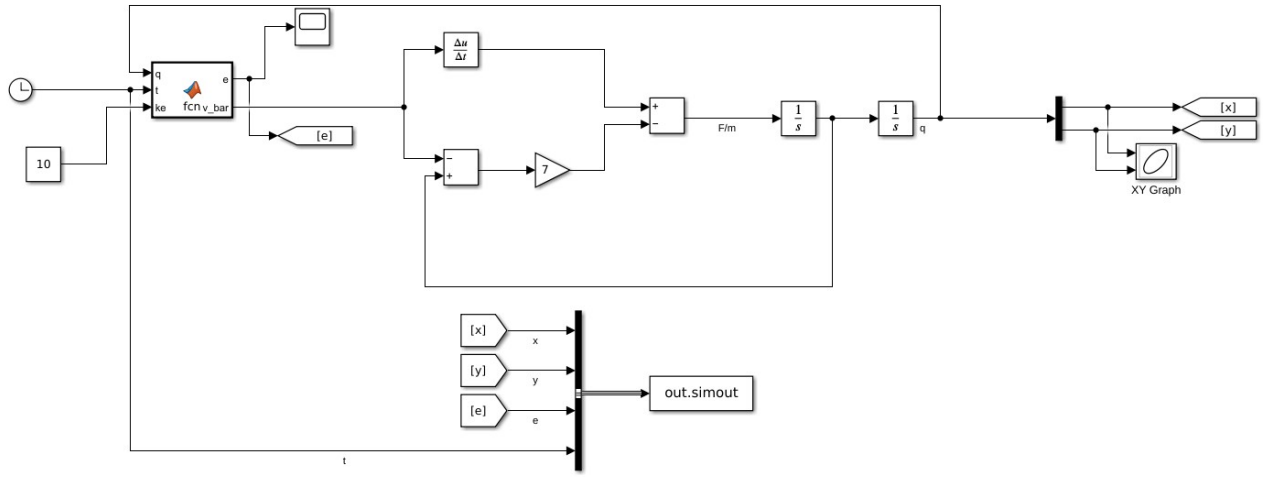


Figure 1: Simulink Model - Question 1

The function used to calculate the desired velocity control signal $\bar{v}$ and the error $e = \phi(q)$ uses the first control law for the first 60 seconds, the second for 20 seconds and the third for the rest. We can see that in the following code

```
function [e, v_bar] = fcn(q, t, ke)

x = q(1);
y = q(2);

if(t<60)
    u_s = [(sqrt(2.0).*(x-y+2.0).*(-1.0./2.0))./(x.*1.4e+1+y.*1.0e+1+x.^2+y.^2+7.4e+1);
           (sqrt(2.0).*(x+y+1.2e+1).*(-1.0./2.0))./(x.*1.4e+1+y.*1.0e+1+x.^2+y.^2+7.4e+1)];

    u_e = [-ke.*(x.*2.0+1.4e+1).*((x+7.0).^2+(y+5.0).^2-2.5e+1./4.0);
           -ke.*(y.*2.0+1.0e+1).*((x+7.0).^2+(y+5.0).^2-2.5e+1./4.0)];

    e = (x+7)^2 + (y+5)^2 - 6.25;
elseif (t<80)
    u_s = [sqrt(2.0)./2.0+sqrt(6.0)./2.0;
          sqrt(2.0).*(-1.0./2.0)+sqrt(6.0)./2.0];

    u_e = [(sqrt(3.0).*ke.*(y./2.0-(sqrt(3.0).*x)./2.0+7.0./5.0))./2.0;
           ke.*(y./2.0-(sqrt(3.0).*x)./2.0+7.0./5.0).*(-1.0./2.0)];

    e = -sin(pi/3) * x + cos(pi/3) * y + 1.4;
else
    u_s = [(sqrt(2.0).*(x-y).*(-1.0./2.0))./(x.*-2.0e+1-y.*2.0e+1+x.^2+y.^2+2.0e+2);
           (sqrt(2.0).*(x+y-2.0e+1).*(-1.0./2.0))./(x.*-2.0e+1-y.*2.0e+1+x.^2+y.^2+2.0e+2)];

    u_e = [-ke.*(x.*2.0-2.0e+1).*((x-1.0e+1).^2+(y-1.0e+1).^2-3.6e+1);
           -ke.*(y.*2.0-2.0e+1).*((x-1.0e+1).^2+(y-1.0e+1).^2-3.6e+1)];

    e = (x-10)^2 + (y-10)^2 - 36 ;
end


v_bar = u_s + u_e;
```

The following Code was used to plot the results

```
out = sim('LAB5.slx');

x = reshape(out.simout.x.Data, [], 1);
y = reshape(out.simout.y.Data, [], 1);
e = reshape(out.simout.e.Data, [], 1);
t = reshape(out.simout.t.Data, [], 1);

r1 = sqrt(6.25);
x0_1 = -7;
y0_1 = -5;

r2 = 6;
x0_2 = 10;
```

```
y0_2 = 10;

theta = 0:0.01:2*pi;
x_c1 = r1 * cos(theta) + x0_1;
y_c1 = r1 * sin(theta) + y0_1;

x_c2 = r2 * cos(theta) + x0_2;
y_c2 = r2 * sin(theta) + y0_2;

x_line = -5:0.01:20;
y_line = tan(pi/3)*x_line - 1.4/(cos(pi/3));

plot(x,y, x_c1, y_c1,'--',x_c2, y_c2,'--', x_line, y_line,'--','LineWidth',2);
xlabel('x')
ylabel('y')
legend('Robot Trajectory', 'Traj1', 'Traj2', 'Traj3')
axis equal

figure;
plot(t,e)
xlabel('t')
ylabel('e')
legend('Error')
```

The results are shown below

in Figure 2 we see the robot trajectory compared to the three reference trajectories

In Figure 3 we see the tracking error over time.

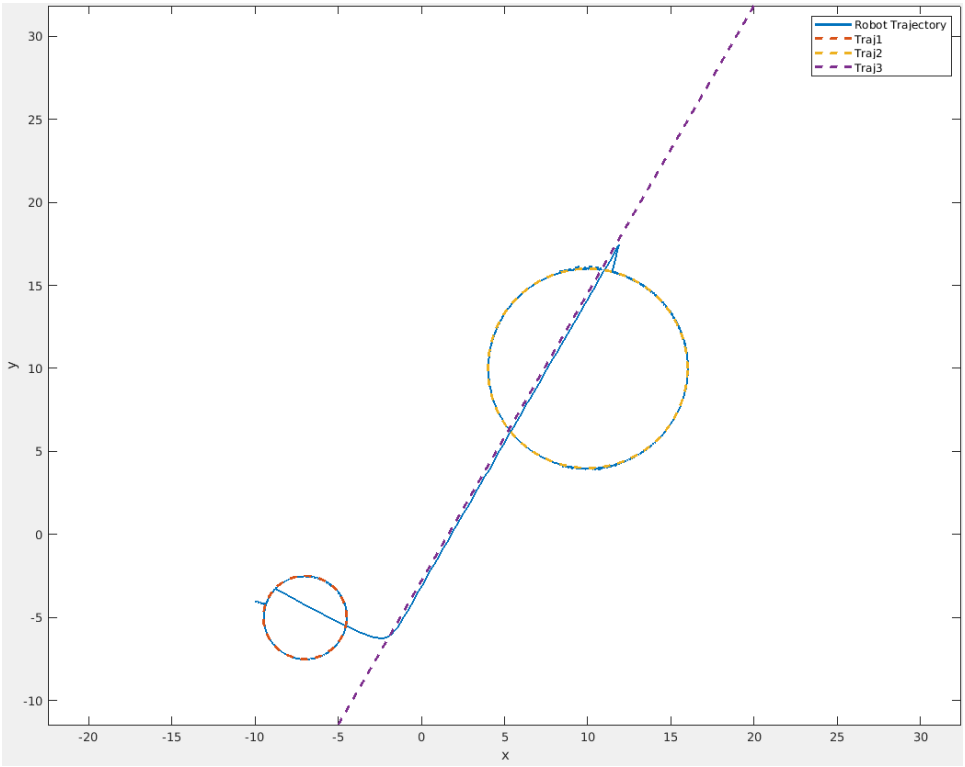


Figure 2: Reference Trajectories vs Actual trajectory at desired velocity 2, sim time = 450 s

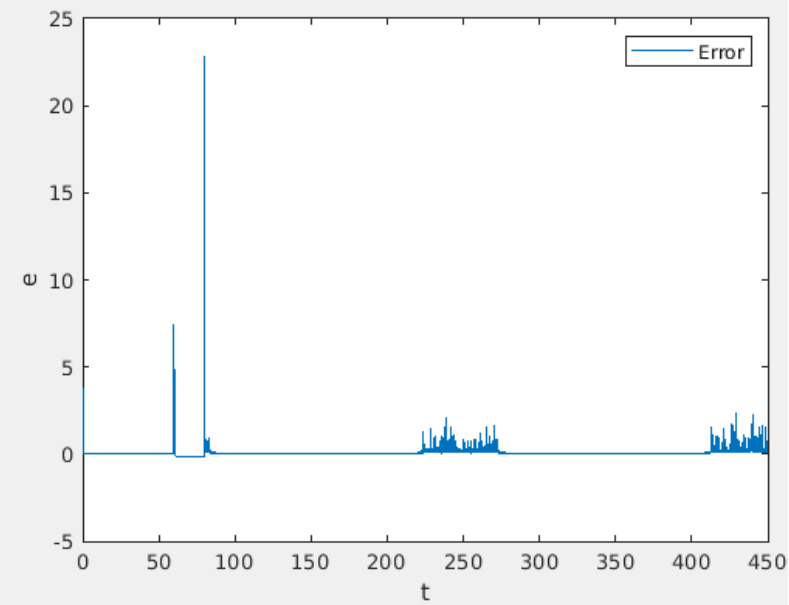


Figure 3: Tracking error at desired velocity 2

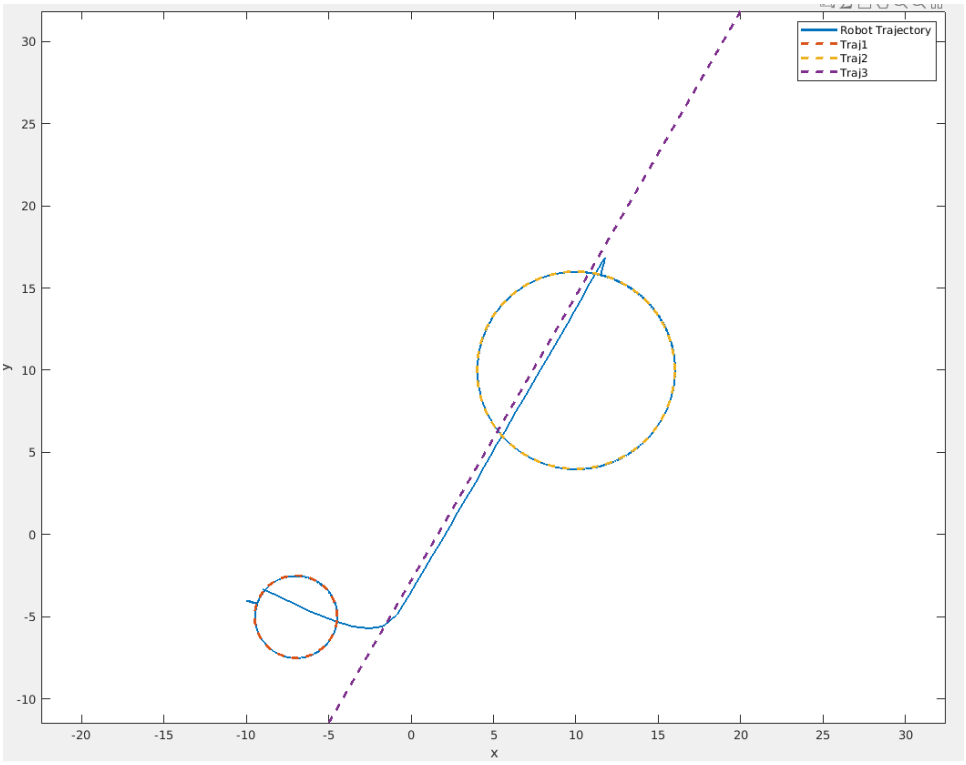


Figure 4: Reference Trajectories vs Actual trajectory at desired velocity 5, sim time = 300 s

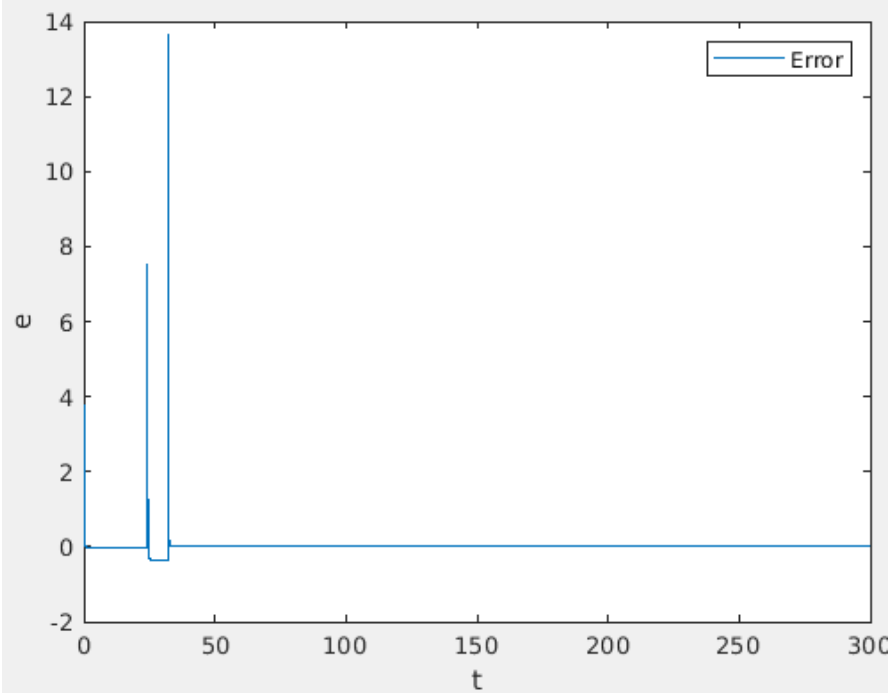


Figure 5: racking error at desired velocity 5

Question 2:

In this task, we want a robot with mass $m=2.3$ and inertia $J=1.7$ to follow a 3D trajectory defined as the intersection of two surfaces implicitly defined as functions of $q=[x,y,z]^T$ as,

$$\phi_1(q)=x^2+y^2-400=0$$
$$\phi_2(q)=z+y-10=0$$

The intersection of these two surfaces is a eclipse as shown in the following figure

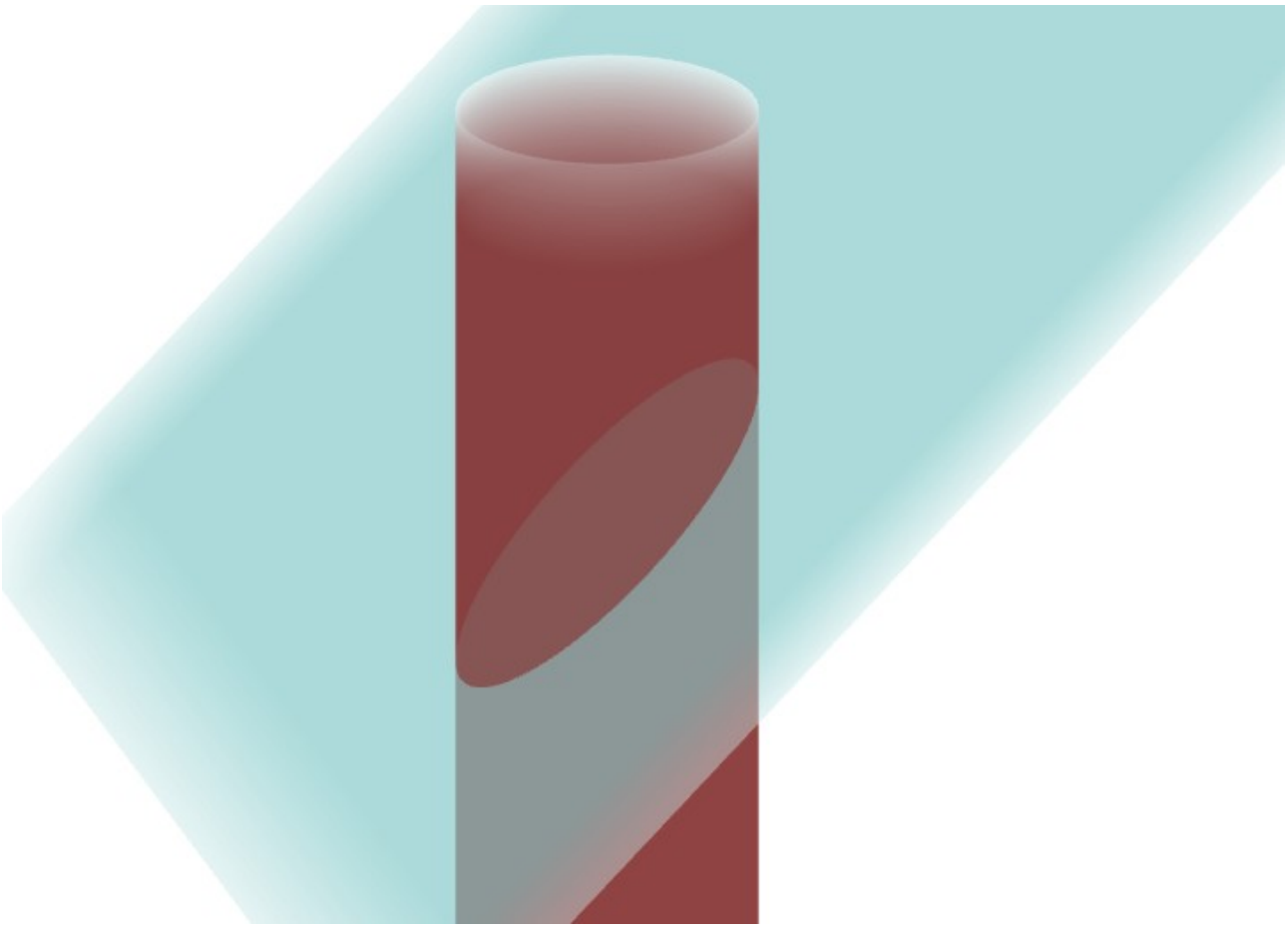


Figure 6: Surfaces Intersection

We define the error in position in two component, the error from the first surface, and the error from the second surface. When these two error components are 0 then the robot is on the desired trajectory.

$$e_1=\phi_1=x^2+y^2-400$$
$$e_2=\phi_2=z+y-10$$

The derivatives of these errors are:

$$\dot{e_1}=2x\dot{x}+2y\dot{y}=[2x,2y,0]v$$
$$\dot{e_2}=\dot{y}+\dot{z}=[0,1,1]v$$

In addition, we will add a error that describes the difference between the current velocity an desired velocity in traversing the trajectory, because if we doesn't include this component , the robot will reach one point on the desired path and stop. This error is defined as :

$$\Delta V = \dot{s_{des}} - \dot{s}$$

In this task, $\dot{s_{des}}$ was set to 5. The trajectory traversing velocity is defined as:

$$\dot{s} = \frac{\nabla \phi_1 \times \nabla \phi_2}{\|\nabla \phi_1 \times \nabla \phi_2\|} v$$

In this case

$$\nabla \phi_1 = \begin{bmatrix} \frac{\delta \phi_1}{\delta x} & \frac{\delta \phi_1}{\delta y} & \frac{\delta \phi_1}{\delta z} \end{bmatrix} = [2x, 2y, 0]$$
$$\nabla \phi_2 = \begin{bmatrix} \frac{\delta \phi_2}{\delta x} & \frac{\delta \phi_2}{\delta y} & \frac{\delta \phi_2}{\delta z} \end{bmatrix} = [0, 1, 1]$$

Therefore,

$$\nabla \phi_1 \times \nabla \phi_2 = [2y, -2x, 2x]$$

Therefore,

$$\dot{s} = \frac{1}{\sqrt{(2x^2 + y^2)}} [y, -x, x] v$$

The Jacobian is defined as:

$$J(q) = \begin{bmatrix} \frac{\nabla \phi_1 \times \nabla \phi_2}{\|\nabla \phi_1 \times \nabla \phi_2\|} \\ \frac{\nabla \phi_1}{\|\nabla \phi_1\|} \\ \frac{\nabla \phi_2}{\|\nabla \phi_2\|} \end{bmatrix} = \begin{bmatrix} \frac{y}{\sqrt{(2x^2 + y^2)}} & \frac{-x}{\sqrt{(2x^2 + y^2)}} & \frac{x}{\sqrt{(2x^2 + y^2)}} \\ \frac{x}{\sqrt{(x^2 + y^2)}} & \frac{y}{\sqrt{(x^2 + y^2)}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

The control forces are defined as:

$$\frac{F_c}{m} = J(q)^{-1} \left(\begin{bmatrix} u_s \\ u_{e1} \\ u_{e2} \end{bmatrix} - \dot{J}(q) v \right)$$

The control signals are defined as:

$$u_s = K_s \Delta V$$
$$u_{e1} = -K_{1e1} \dot{e}_1 - K_{2e1} e_1$$
$$u_{e2} = -K_{1e2} \dot{e}_2 - K_{2e2} e_2$$

These equations are expressed in the following part of the Simulink model:

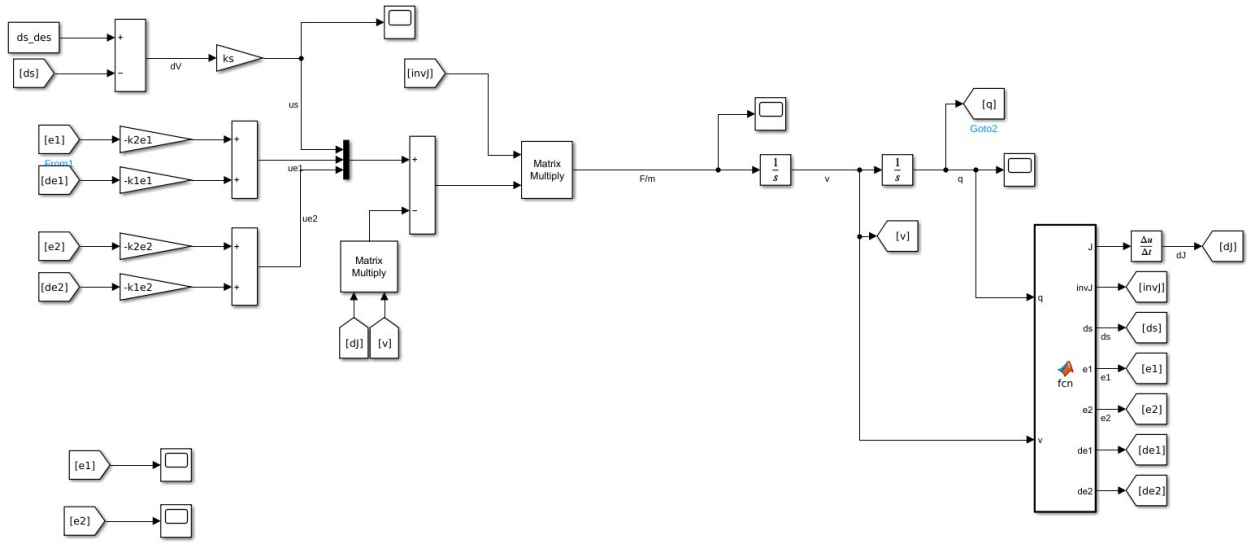


Figure 7: Translational control - Question 2

The Function for calculating errors and their derivatives, Jacobian and its inverse and current traverse speed is as follows:

```
function [J, invJ, ds, e1, e2, de1, de2] = fcn(q, v)

x = q(1);
y = q(2);
z = q(3);

J = [[y.*1.0./sqrt(x.^2.*2.0+y.^2);x.*1.0./sqrt(x.^2+y.^2);0.0], ...
     [-x.*1.0./sqrt(x.^2.*2.0+y.^2);y.*1.0./sqrt(x.^2+y.^2);sqrt(2.0)./2.0],...
     [x.*1.0./sqrt(x.^2.*2.0+y.^2);0.0;sqrt(2.0)./2.0]];
invJ = inv(J);

ds = J(1,:)* v;

e1 = x^2 +y^2 -400;

e2 = z+y-10;

grad_1 = [x.*2.0,y.*2.0,0.0];
grad_2 = [0.0,1.0,1.0];

de1 = grad_1 * v;
de2 = grad_2 * v;
```

In terms of angular control, we will define two error, error in angular position and error in angular velocity. Since we don't have a desired angular velocity , the angular velocity error is :

$$e_w = w$$

The error in angular position is defined as:

$$e_r = 0.5\left(R(\delta) - R(\delta)^T\right)^v$$

where $A^v = [A_{23}, A_{31}, A_{12}]^T$ if A is a skew-symmetric matrix. The error rotation matrix is defined as:

$$R(\delta) = R(\alpha)R(\alpha_{des})^T$$

Where $R(\alpha)$ is the current rotation matrix and $R(\alpha_{des})$ is the desired rotation matrix.

The control moment is defined as:

$$M = w \times Jw - K_w e_w - k_r e_r$$

The following Figure shows the angular control part.

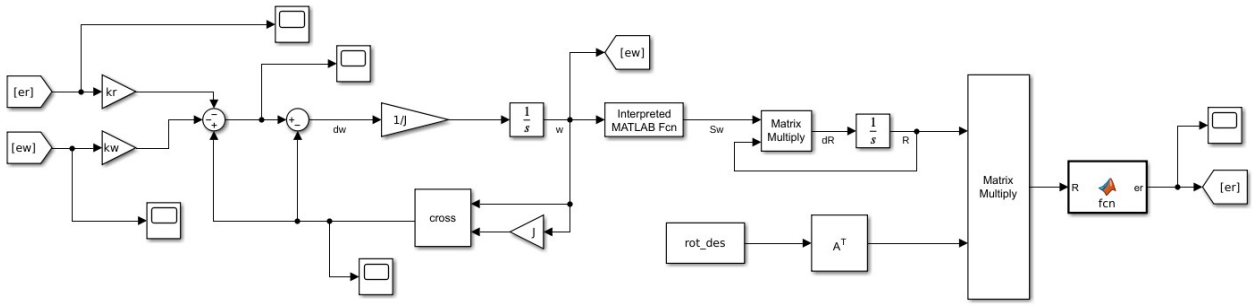


Figure 8: Angular control - Question 2

Experiment:

The initial rotation is defined using Euler vector as: $n_0 = [0,1,0]$ which maps to the rotation matrix:

$$R_{init} = \begin{bmatrix} 0.5403 & 0 & 0.8415 \\ 0 & 1.0000 & 0 \\ -0.8415 & 0 & 0.5403 \end{bmatrix}$$

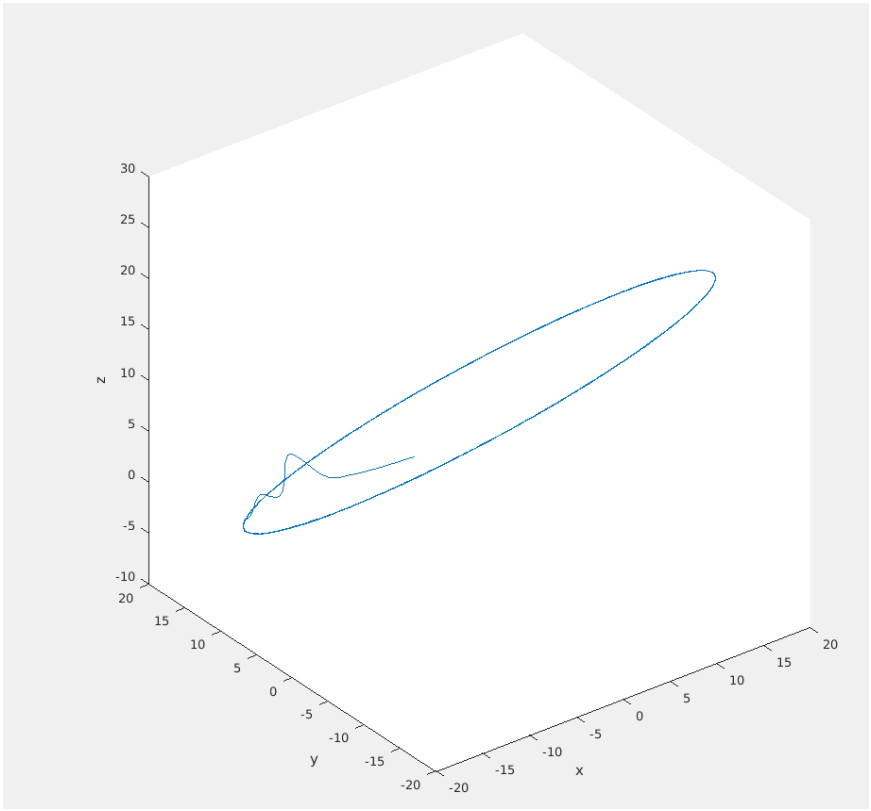
The desired rotation is defined using the vector $n_{des} = \left[\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]$ which maps to the rotation matrix:

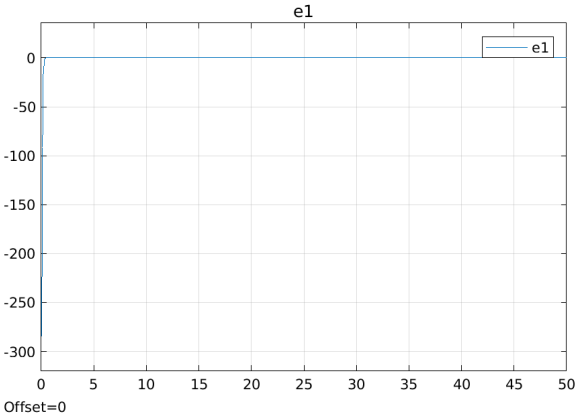
$$R_{des} = \begin{bmatrix} 0.6935 & -0.3326 & 0.6391 \\ 0.6391 & 0.6935 & -0.3326 \\ -0.3326 & 0.6391 & 0.6935 \end{bmatrix}$$

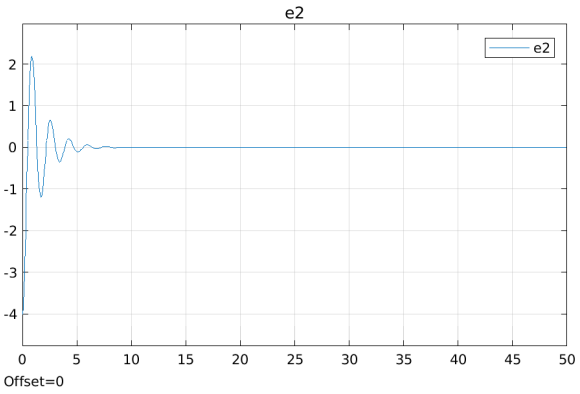
The desired traversing speed was set to $\dot{s_{des}} = 5$ and the control coefficients were set to:

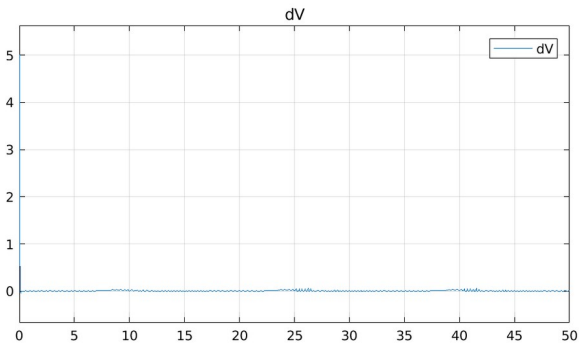
$$k_s = 50, k_{1e1} = 1, k_{2e1} = 10, k_{1e2} = 1, k_{2e2} = 10, k_r = 50, k_w = 20$$

The results are shown in the Figures below:









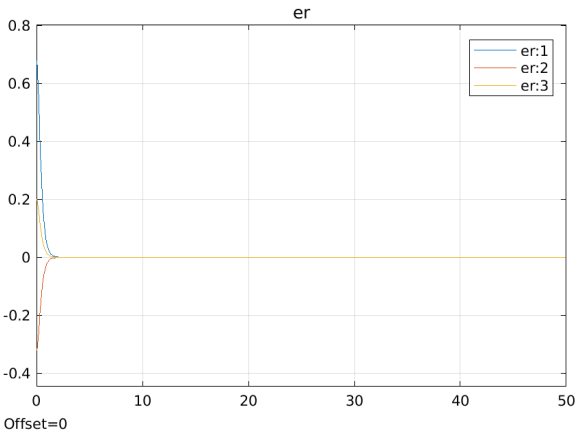


Figure 9: Simulation Results - Question 2

Question 3:

In this task, the passification method was used to achieve the same trajectory discussed in the previous question.

The control forces and moments are defined as:

$$F = u_{pf} + u_f$$
$$M = u_{pm} + u_m$$

where $u_{pf} = \dot{v}_d, u_{pm} = w \times Jw + J\dot{w}_d$

It turns out that the system is passive in terms of the input $u^T = [u_f^T, u_m^T]$ and output $y^T = [(\dot{q} - v_d)^T, (w - w_d)^T]$

We can use feedback gain as $u = -Ky$.

The desired velocities are defined as:

$$v_d = u_\tau + u_\phi = J(q)^{-1}[\dot{s_{des}}, 0, 0]^T - K_\phi \left(\phi_1 \frac{\delta \phi_1}{\delta q} + \phi_2 \frac{\delta \phi_2}{\delta q} \right)$$

We can see that here as with the previous question, we have a term to drive the error of position to 0 and a term to drive the velocity error to 0.

Bu substituting, we get:

$$v_d = \begin{bmatrix} \frac{y}{\sqrt{(2x^2+y^2)}} & \frac{-x}{\sqrt{(2x^2+y^2)}} & \frac{x}{\sqrt{(2x^2+y^2)}} \\ \frac{x}{\sqrt{(x^2+y^2)}} & \frac{y}{\sqrt{(x^2+y^2)}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{-1} \begin{bmatrix} \dot{s_{des}} \\ 0 \\ 0 \end{bmatrix} - K_\phi \begin{bmatrix} 2x(x^2+y^2-400) \\ 2y(x^2+y^2-400) + y + z - 10 \\ y + z - 10 \end{bmatrix}$$

The desired angular velocity is defined as:

$$w_d = -S(n_c) n k_w (1 - n^T n_c)$$

The Simulink model used for this task is shown below:

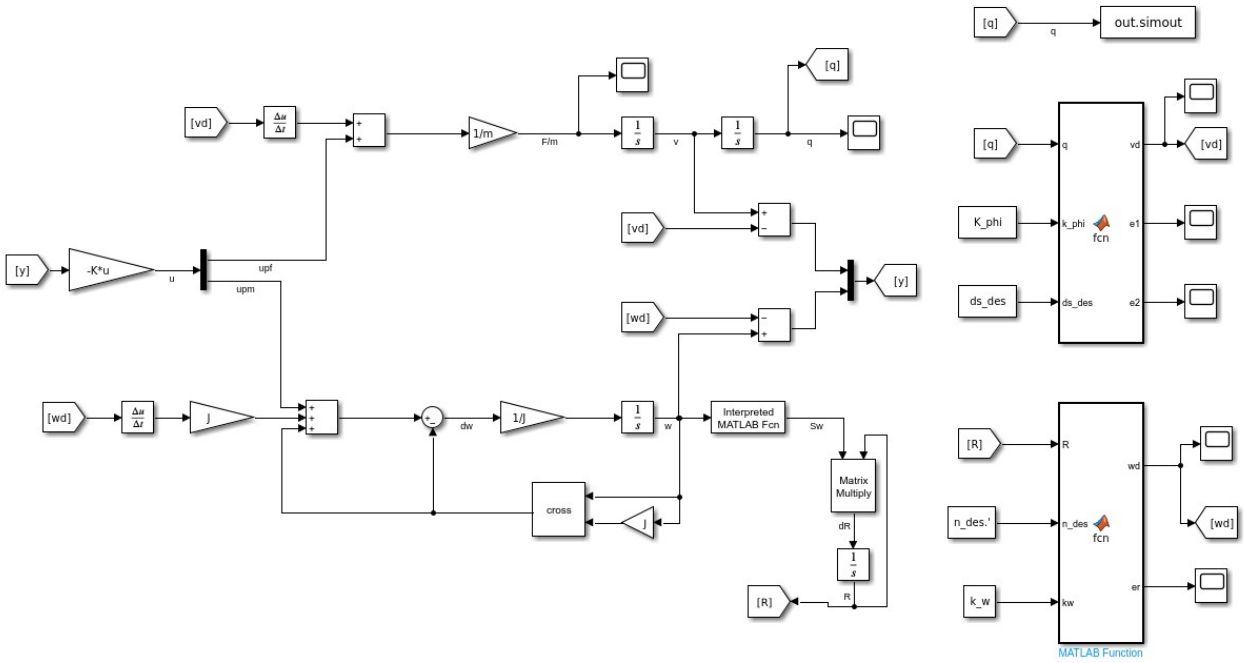
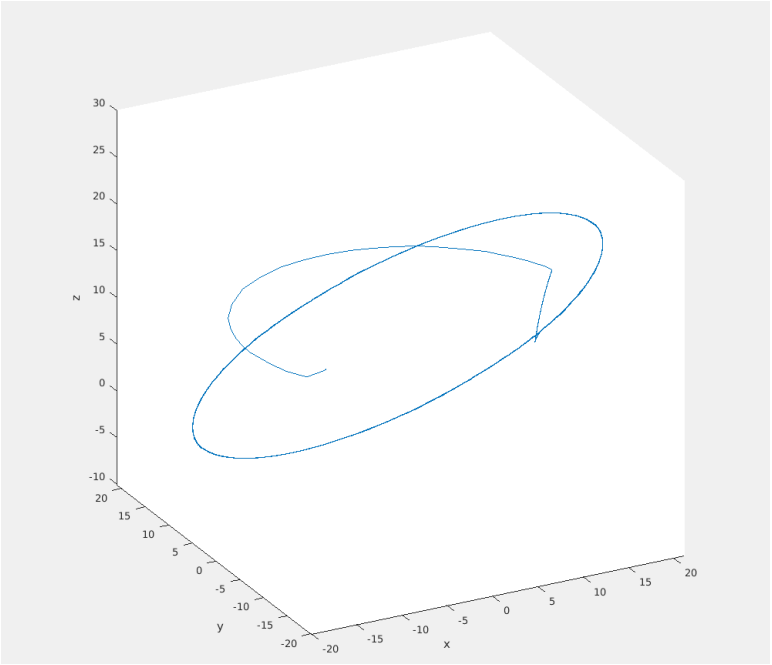


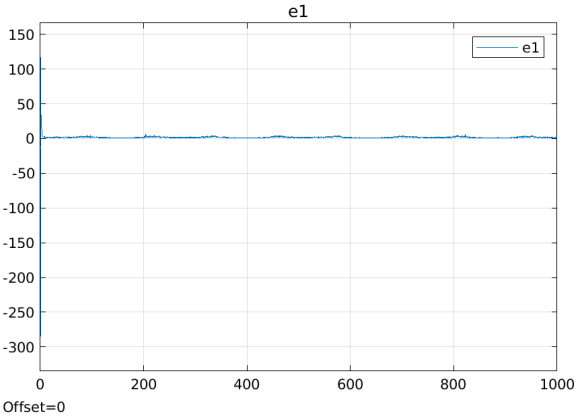
Figure 10: Simulink Model - Question 3

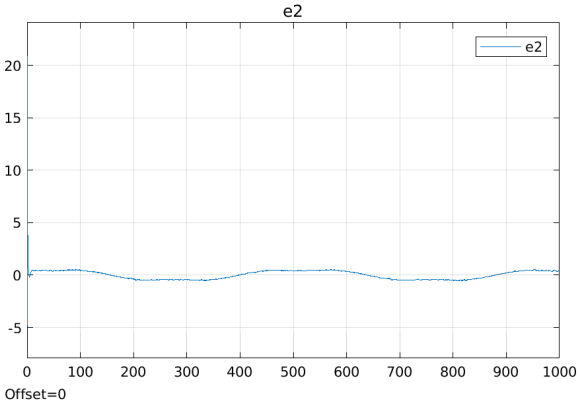
The code used to calculate the velocities are:

```
function [vd,e1,e2]  = fcn(q,k_phi,ds_des)
.
x = q(1);
y = q(2);
z = q(3);

u_phi = - k_phi * [x.*(x.^2+y.^2-4.0e+2).*2.0;y+z+y.*(x.^2+y.^2-4.0e+2).*2.0-1.0e+1;y+z-1.0e+1];

J = [[y.*1.0./sqrt(x.^2.*2.0+y.^2);x.*1.0./sqrt(x.^2+y.^2);0.0], ...
     [-x.*1.0./sqrt(x.^2.*2.0+y.^2);y.*1.0./sqrt(x.^2+y.^2);sqrt(2.0)./2.0],...
     [x.*1.0./sqrt(x.^2.*2.0+y.^2);0.0;sqrt(2.0)./2.0]];

u_tau = (J^-1) * [ds_des;0;0];

vd = u_tau + u_phi;

e1 = x^2 +y^2 -400;
e2 = z+y-10;


function [wd, er] = fcn(R, n_des, kw)

n = rotm2axang(R);
n = n(1:3)*n(4);
n = n.';


Sc = [0,n_des(3),-n_des(2); -n_des(3), 0, n_des(1); n_des(2),-n_des(1),0];

wd = -Sc * n * kw * (1-n.'*n_des);
er = (1-n.'*n_des);
```

The results with coefficients: $K = diag([1,1,1,10,10,10]), K_\phi = diag([1,1,1]), k_w = 3, \dot{s_{des}} = 1$







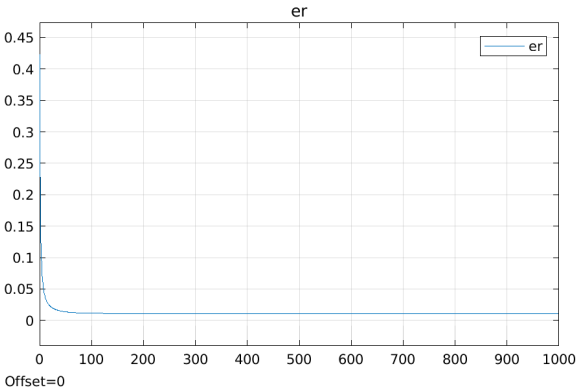


Figure 11: Simulation Results - Question 3

Conclusions:

In this laboratory work, trajectory tracking in 2D and 3D space was investigated.

We saw in the first task how we can control a robot to follow a trajectory or a sequence of trajectories which are represented implicitly. The control law used depend on driving two types of errors to zero, the first type is to minimize the normal distance between the robot and the trajectory and the second to provide a constant rate along the trajectory (to prevent just stopping on with zero normal distance).

Simulations with three consequent trajectories were conducted with different required speeds. The control performance was very good, however, with higher speeds, the normal error increases ; however, with acceptable margins.

In the second task, 3D trajectories were represented as an intersection between two 3D surfaces , we applied two parts of control in this task, translational control and angular control. The same principle was applied in this question as the first one, the control works in two parts, one part to minimize the normal distance and the other to keep constant velocity. In the angular part, we saw how to reduce the rotation matrix error to a vectorized error to be used. And the control law was built to minimize this error in addition to the error in angular velocity. The results showed very good performance and excellent tracking.

In the third question, a method was implemented for the same task as question2. This method depends on transforming the system into passive system and then control it using feedback gain control. The results showed good performance, however, it can be improvement with fine tuning of the control coefficients.