MAE 279 FINAL 2020 BANSEOK LEE 605351891

Problem (a)

Final 2020 Banseok Lee 605351891

(a) analytical expressions

$$y=\phi,\ x=\begin{bmatrix}\phi\\ \dot\phi\end{bmatrix},\ \phi=B^T\theta$$

$$\dot x=\mathbb{A}x+\mathbb{B}u,\ y=\mathbb{C}x$$

$$\dot\phi=B^T\dot\theta,\ \ddot\phi=B^T\ddot\theta,\ \text{then } \ddot\theta=(B^T)^{-1}\ddot\phi \to J(B^T)^{-1}\ddot\phi+D(B^T)^{-1}\dot\phi+\nu\Lambda(B^T)^{-1}\phi = Bu$$

$$\ddot\phi=B^TJ^{-1}Bu-B^TJ^{-1}D(B^T)^{-1}\dot\phi-B^TJ^{-1}\nu_0\Lambda(B^T)^{-1}\phi$$

$$\therefore \begin{bmatrix}\dot\phi\\ \ddot\phi\end{bmatrix}=\underbrace{\begin{bmatrix}O_5 & I_5\\ -B^TJ^{-1}\nu_0\Lambda(B^T)^{-1} & -B^TJ^{-1}D(B^T)^{-1}\end{bmatrix}}_{\to\mathbb{A}}\begin{bmatrix}\phi\\ \dot\phi\end{bmatrix}+\underbrace{\begin{bmatrix}O_5\\ B^TJ^{-1}B\end{bmatrix}}_{\to\mathbb{B}}u$$

$$,\ y=\underbrace{\begin{bmatrix}I_5 & O_5\end{bmatrix}}_{\to\mathbb{C}}\begin{bmatrix}\phi\\ \dot\phi\end{bmatrix}$$

To change $x\to x$,
$$P\triangleq\begin{bmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{bmatrix}$$ then.

$$Px=x=\begin{bmatrix}\phi_1\\ \dot\phi_1\\ \phi_2\\ \dot\phi_2\\ \vdots\\ \phi_5\\ \dot\phi_5\end{bmatrix},\ \text{rank}(P)=10 \to \text{invertible}.$$

For system
$$\dot x=\mathbb{A}x+\mathbb{B}u$$
$$y=\mathbb{C}x \qquad \to x$$

$$x=P^{-1}x,\ \dot x=P^{-1}\dot x$$

$$P^{-1}\dot x=\mathbb{A}P^{-1}x+\mathbb{B}u,\ y=\mathbb{C}P^{-1}x.$$

$$\therefore \begin{cases}\dot x=P\mathbb{A}P^{-1}x+P\mathbb{B}u\\ y=\mathbb{C}P^{-1}x.\end{cases} \quad \therefore \begin{bmatrix}A_p=P\mathbb{A}P^{-1}\\ B_p=P\mathbb{B}\\ C_p=\mathbb{C}P^{-1}\end{bmatrix}$$

Numerical verification

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (a)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
AA = [zeros(n) eye(n);-B'*inv(J)*v0*Lamda*inv(B') -B'*inv(J)*D*inv(B')];
BB = [zeros(n) ; B'*inv(J)*B];
CC = [eye(n) zeros(n)];
P = [1 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 1 0 0 0 0;
    0 1 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 1 0 0 0;
    0 0 1 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 1 0 0;
    0 0 0 1 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 1;];
AAA = P*AA*inv(P);
BBB = P*BB;
CCC = CC*inv(P);
load('finaldata.mat')
CheckAp = norm(AAA-Ap)/norm(Ap);
CheckBp = norm(BBB-Bp)/norm(Bp);
CheckCp = norm(CCC-Cp)/norm(Cp);
CheckAp
CheckBp
CheckCp
```

Checking the difference between (A, B, C) from the analytical results with the (Ap, Bp, Cp) by using normalized norm function.

CheckAp = 2.6818e-14

CheckBp = 7.0453e-15

CheckCp = 0

This result shows that the analytical calculation is verified.

Problem (b)

(b)

target: $\phi = Im[\hat{\phi} e^{j\omega t}]$,

State trajectory : $x(t) = \mathcal{X} e^{\Omega t} \eta$

Control input : $u(t) = \mathbb{U} e^{\Omega t} \eta$

Find analytical expressions for $\mathcal{X}, \mathbb{U}, \Omega, \eta$ interms of $\hat{\phi}, \omega, \nu_0$, plant params,

$x = P\chi = P\begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix}$, from phasor $\hat{x} = P\begin{bmatrix} I_5 \\ j\omega I_5 \end{bmatrix}\hat{\phi}$

From (1) : $J\ddot{\theta} + D\dot{\theta} + \nu\Lambda\theta = Bu$

phasor : $\hat{u} = B^{-1}(-J\omega^2 + j\omega D + \nu_0 \Lambda)(B^T)^{-1}\hat{\phi}$

Let $\Gamma = \frac{1}{2}\begin{bmatrix} -j & 1 \\ j & 1 \end{bmatrix}$, then $\mathcal{X} = [\hat{x} \;\; \bar{\hat{x}}]\Gamma$

$\mathbb{U} = [\hat{u} \;\; \bar{\hat{u}}]\Gamma$

$\Omega = \Gamma^{-1}\begin{bmatrix} j\omega & 0 \\ 0 & -j\omega \end{bmatrix}\Gamma$

$= \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix}$

from note 15

@ $t=0$, $x(0) = \mathcal{X}\eta = P\begin{bmatrix} \phi(0) \\ \dot{\phi}(0) \end{bmatrix} = P\begin{bmatrix} Im[\hat{\phi}] \\ Im[j\omega\hat{\phi}] \end{bmatrix} \Rightarrow \eta = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Numerical verification

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (b)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gamma = 0.5*[-1i 1; 1i 1];
xhat  = P*[eye(5);1i*w*eye(5)]*phih;
uhat  = inv(B)*(-J*w^2+1i*w*D+v0*Lamda)*inv(B')*phih;
X = [xhat conj(xhat)]*Gamma;
U = [uhat conj(uhat)]*Gamma;
O = inv(Gamma)*[1i*w 0;0 -1i*w]*Gamma;
Checkreg = norm(Ap*X+Bp*U-X*O);
Checkreg
```

Implement analytical results to the Matlab codes, and by using the norm function, verified the regulator equation.

Checkreg = 5.2538e-11

This result shows that the analytical calculation is verified.

Problem (c)

(c) Design a network of coupled n-AHO w state $\xi_i(t) \in \mathbb{R}^2$ ($i=1,2 \cdots n$)

$y_i = h_i$ with $h_i(t) = \begin{bmatrix} \sin(\omega t+\beta_i) \\ \cos(\omega t+\beta_i) \end{bmatrix}$, $\beta_i = \angle \hat{\phi}_i$

$(i-1) - i - (i+1)$ nearest neighbor

So, I designed $\underset{\text{Simplest}}{\wedge}$ Δ based on $\begin{cases} \Delta \mathbb{1} = 0 \\ -\Delta \in \mathbb{H}_0 \end{cases}$, $\Delta \triangleq \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$,

Let's define rotation matrix Ω_i, $f_0(\cdot)$, $\sigma(\cdot)$

$\Omega_i \triangleq \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix}$, $f_0(\cdot) = \begin{bmatrix} \sigma(\cdot) & -\omega \\ \omega & \sigma(\cdot) \end{bmatrix}$, $\sigma(\cdot) = \mu(\alpha^2 - \|\cdot\|^2)$

then

$$\dot{\xi}_i = f_0(\xi_i) + \sum_{j=1}^{n} \delta_{ij}(\xi_i - \Omega_{ij}\xi_j), \quad \Omega_{ij} = \Omega_i^T \Omega_j$$

$$\begin{cases} \dot{\xi}_1 = f_0(\xi_1) + {+1}(\cancel{\xi_1 - \Omega_{11}\xi_1})^{0} - 1(\xi_1 - \Omega_{12}\xi_2) \\ \dot{\xi}_2 = f_0(\xi_2) - 1(\xi_2 - \Omega_{21}\xi_1) - 1(\xi_2 - \Omega_{23}\xi_3) \\ \vdots \end{cases}$$

$$\rightarrow \dot{\xi} = \begin{bmatrix} \begin{matrix} \sigma(\xi_1) & -\omega \\ \omega & \sigma(\xi_1) \end{matrix} & 0 & \cdots & 0 \\ 0 & \ddots & & 0 \\ 0 & \cdots & 0 & \begin{matrix} \sigma(\xi_5) & -\omega \\ \omega & \sigma(\xi_5) \end{matrix} \end{bmatrix} + \underbrace{\begin{bmatrix} -I_2 & \Omega_{12} & O_2 & O_2 & O_2 \\ \Omega_{21} & -2I_2 & \Omega_{23} & O_2 & O_2 \\ O_2 & \Omega_{32} & -2I_2 & \Omega_{34} & O_2 \\ O_2 & O_2 & \Omega_{43} & -2I_2 & \Omega_{45} \\ O_2 & O_2 & O_2 & \Omega_{54} & -I_2 \end{bmatrix}}_{\rightarrow R} \xi$$

For simulation, I set $\alpha = 1$, $\mu = 200$ (Increase μ → AHO converges to SS fast)

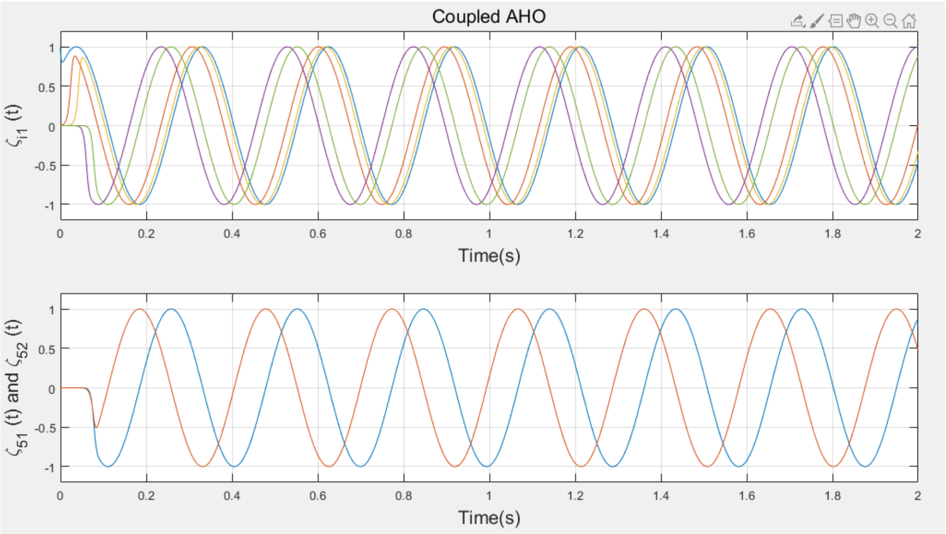
Simulation codes

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (c)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
options=odeset('RelTol',1e-4,'Refine',5);
simtime=[0 2];
xinit=[1;1;0;0;0;0;0;0;0;0];
Beta1 = unwrap(angle(phih))*(360/(2*pi)); % get optmal phase angle

[t,x]=ode45(@(t,x)cdx(t,x,Beta1,0),simtime,xinit);
```

ODE function

```
function cdx = cdx(t,x,Beta, 0)

    x1 = x(1:2,:); x2 = x(3:4,:); x3 = x(5:6,:); x4 = x(7:8,:); x5 = x(9:10,:);
    beta1 = Beta(1); beta2 = Beta(2); beta3 = Beta(3); beta4 = Beta(4); beta5 = Beta(5);
    mu = 200;
    A1 = -0 + eye(2)*(mu)*(1-norm(x1)^2);
    A2 = -0 + eye(2)*(mu)*(1-norm(x2)^2);
    A3 = -0 + eye(2)*(mu)*(1-norm(x3)^2);
    A4 = -0 + eye(2)*(mu)*(1-norm(x4)^2);
    A5 = -0 + eye(2)*(mu)*(1-norm(x5)^2);
    R1 = [cos(beta1) -sin(beta1);sin(beta1) cos(beta1)];
    R2 = [cos(beta2) -sin(beta2);sin(beta2) cos(beta2)];
    R3 = [cos(beta3) -sin(beta3);sin(beta3) cos(beta3)];
    R4 = [cos(beta4) -sin(beta4);sin(beta4) cos(beta4)];
    R5 = [cos(beta5) -sin(beta5);sin(beta5) cos(beta5)];
    R = [-eye(2) R1'*R2 zeros(2) zeros(2) zeros(2);
        R2'*R1 -2*eye(2) R2'*R3 zeros(2) zeros(2);
        zeros(2) R3'*R2 -2*eye(2) R3'*R4 zeros(2);
        zeros(2) zeros(2) R4'*R3 -2*eye(2) R4'*R5;
        zeros(2) zeros(2) zeros(2) R5'*R4 -eye(2)];
    cdx = (blkdiag(A1,A2,A3,A4,A5) + R)*x;
end
```

Figure (c)



From trial and error, found that larger mu make the oscillator converges to steady-state faster. Thus, the designed AHO converges almost immediately.

Problem (d)

(d) Design a decentralized output regulator of the form.

$$u = U\xi + Q_1(y - CX\xi), \quad \dot{\xi} = \varphi(\xi)$$

optimal gait $x(t) \to \mathcal{X}e^{\Omega t}\eta$

$$(\mathbb{X}, \mathbb{U}) \in \mathcal{X} = XH, \ \mathbb{U} = UH$$

$$h = He^{\Omega t}\eta = \begin{bmatrix} \sin(\omega t+\beta_1) \\ \cos(\omega t+\beta_1) \\ \vdots \\ \sin(\omega t+\beta_5) \\ \cos(\omega t+\beta_5) \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha_1\sin\gamma_1 & \alpha_1\cos\gamma_1 \\ \vdots & \vdots \\ \alpha_{10}\sin\gamma_1 & \alpha_{10}\cos\gamma_1 \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \cos\omega t & -\sin\omega t \\ \sin\omega t & \cos\omega t \end{bmatrix}}_{e^{\Omega t}} \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\eta}$$

let $\alpha_1 \sim \alpha_{10} = 1$, then

$$\left.\begin{cases} \sin(\omega t+\beta_1) = \sin(\omega t+\gamma_1) \\ \cos(\omega t+\beta_1) = \sin(\omega t+\gamma_2) \\ \vdots \\ \cos(\omega t+\beta_5) = \sin(\omega t+\gamma_{10}) \end{cases}\right\} \Rightarrow \therefore H = \begin{bmatrix} \sin\beta_1 & \cos\beta_1 \\ \sin(\beta_1+\frac{\pi}{2}) & \cos(\beta_1+\frac{\pi}{2}) \\ \vdots & \vdots \\ \sin(\beta_5+\frac{\pi}{2}) & \cos(\beta_5+\frac{\pi}{2}) \end{bmatrix} \in \mathbb{R}^{2n \times 2}$$

$$\mathcal{X}_i = X_iH_i = X_i\begin{bmatrix} \sin\beta_i & \cos\beta_i \\ \sin(\beta_i+\frac{\pi}{2}) & \cos(\beta_i+\frac{\pi}{2}) \end{bmatrix}$$

$$X_i = \mathcal{X}_iH_i^{-1} \Rightarrow X = \begin{bmatrix} X_1 & & & & \\ & X_2 & & & \\ & & X_3 & & \\ & & & X_4 & \\ & & & & X_5 \end{bmatrix}$$

$$\mathbb{U}_i = U_iH_i = U_i\begin{bmatrix} \sin\beta_i & \cos\beta_i \\ \sin(\beta_i+\frac{\pi}{2}) & \cos(\beta_i+\frac{\pi}{2}) \end{bmatrix}$$

$$U_i = \mathbb{U}_iH_i^{-1} \Rightarrow U = \begin{bmatrix} U_1 & & & & \\ & U_2 & & & \\ & & U_3 & & \\ & & & U_4 & \\ & & & & U_5 \end{bmatrix}$$

$$J\ddot{\theta} + D\dot{\theta} + \nu\Lambda\theta = Bu, \quad y = C\theta$$

① $C = B^T$

② B is non singular

③ $u = -ky, \ k = \begin{bmatrix} k_1 & & \\ & \ddots & \\ & & k_n \end{bmatrix}, \ k_i > 0 \Rightarrow J\ddot{\theta} + D\dot{\theta} + BKB^T\theta = 0$

From the mechanical equation analysis, the control gain K derived a diagonal matrix and has positive components. Then in the simulation, I found seems proper K by trial and error process.

K = diag(0.0002, 0.0002, 0.0002, 0.0002, 0.0002)

Simulation codes

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (d)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
beta1 = Beta1(1); beta2 = Beta1(2); beta3 = Beta1(3); beta4 = Beta1(4); beta5 = Beta1(5);
X1 = X(1:2,:); X2 = X(3:4,:); X3 = X(5:6,:); X4 = X(7:8,:); X5 = X(9:10,:);
U1 = U(1,:); U2 = U(2,:); U3 = U(3,:); U4 = U(4,:); U5 = U(5,:);
h1 = [sin(beta1) cos(beta1);sin(beta1+90) cos(beta1+90)];
h2 = [sin(beta2) cos(beta2);sin(beta2+90) cos(beta2+90)];
h3 = [sin(beta3) cos(beta3);sin(beta3+90) cos(beta3+90)];
h4 = [sin(beta4) cos(beta4);sin(beta4+90) cos(beta4+90)];
h5 = [sin(beta5) cos(beta5);sin(beta5+90) cos(beta5+90)];
H = [h1;h2;h3;h4;h5];
XX = blkdiag(X1*inv(h1),X2*inv(h2),X3*inv(h3),X4*inv(h4),X5*inv(h5));
UU = blkdiag(U1*inv(h1),U2*inv(h2),U3*inv(h3),U4*inv(h4),U5*inv(h5));
YY = Cp*XX;
options=odeset('RelTol',1e-4,'Refine',5);
simtime=[0 2];
xinit=[sin(beta1);cos(beta1);sin(beta2);cos(beta2);sin(beta3);
    cos(beta3);sin(beta4);cos(beta4);sin(beta5);cos(beta5);
    0;0;0;0;0;0;0;0;0;0;0];
Beta = unwrap(angle(phih))*(360/(2*pi));
K = blkdiag(0.00009,0.00009,0.00009,0.00009,0.00009);

[tt,xx]=ode45(@(tt,xx) ddx(tt,xx,K,P,Bp,Cp,XX,UU,Beta,0,phih),simtime,xinit);
```

At t = 0, set the initial zeta to the desired oscillator state H*eta.

Then, Zeta(0) = H*eta.

ODE function

```
function ddx = ddx(tt,xx,K,P,Bp,Cp,XX,UU,Beta1,0,phih)

    % Define Constants
    n = 5;
    m = 0.001; % kg
    l = 0.1; % m
    cni = 0.009; % N/(m/s)
    cti = 0.0006; % N/(m/s)
    ct0 = 0.0006; % N/(m/s)
    li = l/(2*n+2); mi = m/(n+1);
    l0 = l/(2*n+2); m0 = m/(n+1);
    v0 = -0.15; % m/s
    % Define matrices
    B = diag(-ones(n,1))+diag(ones(n-1,1),1);
    A = diag(ones(n,1))+diag(ones(n-1,1),-1);
    e = ones(n,1);
    L = diag(li*ones(n,1)); M = diag(mi*ones(n,1));
    Ct = diag(cti*ones(n,1)); Cn = diag(cni*ones(n,1));
    C0 = Cn-Ct;
    F = inv(B')*A*L;
    h = F'*M*e;
    Lamda = F'*C0 + diag((F'-e*h'/m)*Ct*e);
    J = L*M*L/3 + F'*M*F;
    D = L*Cn*L/3 + F'*Cn*F;

   ddx = [];
   x1 = xx(1:2,:);  % zeta1
   x2 = xx(3:4,:);  % zeta2
   x3 = xx(5:6,:);  % zeta3
   x4 = xx(7:8,:);  % zeta4
   x5 = xx(9:10,:); % zeta5
   x0 = xx(1:10,:); % Zeta
   x6 = xx(11:20,:);% state x
   v = xx(21,:);    % v
```

Set parameters and state of the ODE function.

```
% AHO
beta1 = Beta1(1); beta2 = Beta1(2); beta3 = Beta1(3); beta4 = Beta1(4); beta5 = Beta1(5);
mu = 200;
A1 = -0 + eye(2)*(mu)*(1-norm(x1)^2);
A2 = -0 + eye(2)*(mu)*(1-norm(x2)^2);
A3 = -0 + eye(2)*(mu)*(1-norm(x3)^2);
A4 = -0 + eye(2)*(mu)*(1-norm(x4)^2);
A5 = -0 + eye(2)*(mu)*(1-norm(x5)^2);
R1 = [cos(beta1) -sin(beta1);sin(beta1) cos(beta1)];
R2 = [cos(beta2) -sin(beta2);sin(beta2) cos(beta2)];
R3 = [cos(beta3) -sin(beta3);sin(beta3) cos(beta3)];
R4 = [cos(beta4) -sin(beta4);sin(beta4) cos(beta4)];
R5 = [cos(beta5) -sin(beta5);sin(beta5) cos(beta5)];
R = [-eye(2) R1'*R2 zeros(2) zeros(2) zeros(2);
    R2'*R1 -2*eye(2) R2'*R3 zeros(2) zeros(2);
    zeros(2) R3'*R2 -2*eye(2) R3'*R4 zeros(2);
    zeros(2) zeros(2) R4'*R3 -2*eye(2) R4'*R5;
    zeros(2) zeros(2) zeros(2) R5'*R4 -eye(2)];

% get Zeta dot
ddx(1:10,:) = (blkdiag(A1,A2,A3,A4,A5) + R)*x0;
```

Add the oscillator that designed in problem (c) and by using this, zeta dot is calculated.

```
% get feedback u
w = 0(2,1);
h1 = [sin(beta1) cos(beta1);sin(beta1+90) cos(beta1+90)];
h2 = [sin(beta2) cos(beta2);sin(beta2+90) cos(beta2+90)];
h3 = [sin(beta3) cos(beta3);sin(beta3+90) cos(beta3+90)];
h4 = [sin(beta4) cos(beta4);sin(beta4+90) cos(beta4+90)];
h5 = [sin(beta5) cos(beta5);sin(beta5+90) cos(beta5+90)];
uhat = inv(B)*(-J*w^2+1i*w*D+v*Lamda)*inv(B')*phih;
Gamma = 0.5*[-1i 1; 1i 1];
U = [uhat conj(uhat)]*Gamma;
U1 = U(1,:); U2 = U(2,:); U3 = U(3,:); U4 = U(4,:); U5 = U(5,:);
UU = blkdiag(U1*inv(h1),U2*inv(h2),U3*inv(h3),U4*inv(h4),U5*inv(h5));
rho = 0.05*sin(300*tt); % noise
y = (1+rho)*Cp*x6;
u = UU*x0-K*(y-Cp*XX*x0);
```

Because velocity v(t) is not a constant value anymore and U has v inside, newly update U.

Then, the sensor noise rho is calculated and added feedback control equation.

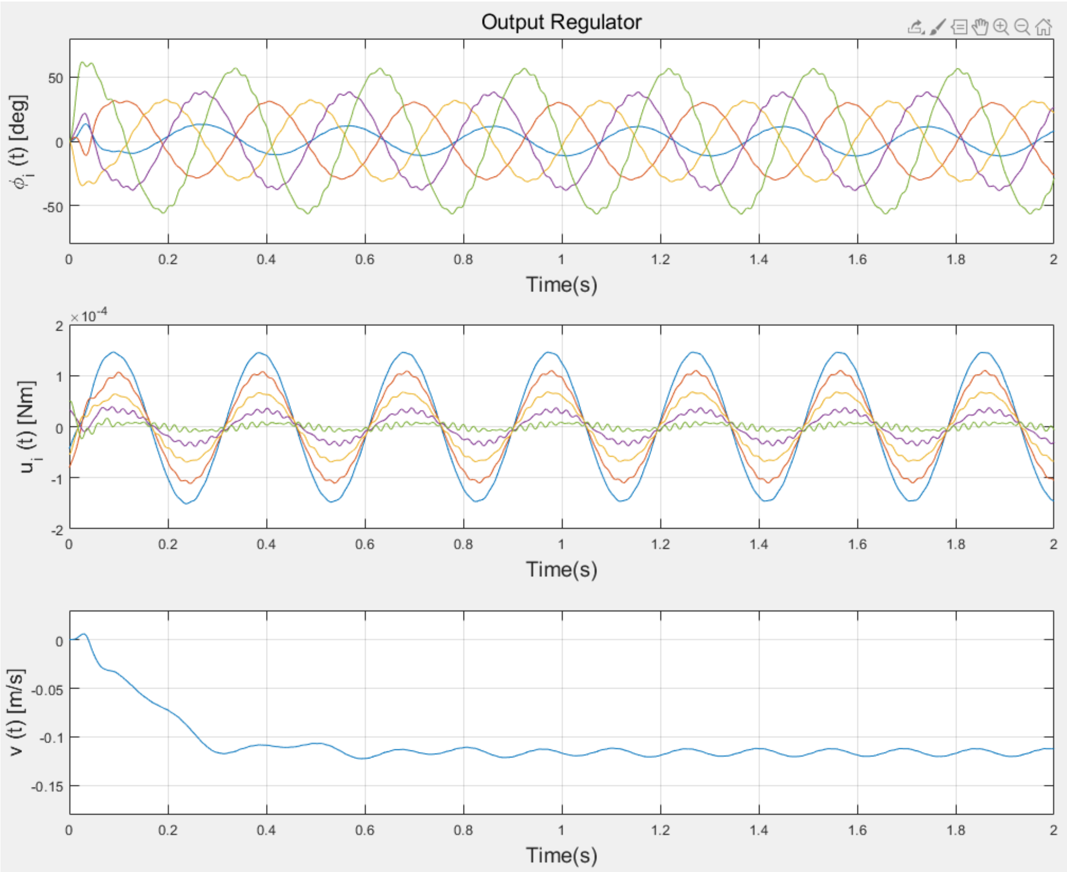
```
% get state dot from nonlinear plant dynamics
AA = [zeros(n) eye(n);-B'*inv(J)*v*Lamda*inv(B') -B'*inv(J)*D*inv(B')];
AAA = P*AA*inv(P);
ddx(11:20,:) = AAA*x6 +Bp*u;
```

Because the plant dynamic has v inside, update Ap for every iteration.

```
% get v dot
thetadot = inv(B')*[xx(12,:);xx(14,:);xx(16,:);xx(18,:);xx(20,:)];
theta = inv(B')*[xx(11,:);xx(13,:);xx(15,:);xx(17,:);xx(19,:)];
dtheta = ct0 +e'*Ct*e+theta'*C0*theta;
ddx(21,:) = (-dtheta*v-thetadot'*Lamda*theta)/m;

end
```

v dot is calculated by second line of the equation (1).

Figure (d)



The simulator sufficiently compensated noise and reached steady-state quickly.

Problem (e)

Modification of the simulation codes from (d)

```
LL = blkdiag([100;100],[100;100],[100;100],[100;100],[100;100]);
eps = 0.1;
```

Designed block diagonal matrix L (LL in the code) and epsilon by trial and error.

```
% With disturbance without Q2
[ttt,xxx]=ode45(@(ttt,xxx) edx1(ttt,xxx,K,P,Bp,Cp,XX,UU,Beta,0,phih,LL,eps),simtime2,xinit);

% With disturbance with Q2
[tttt,xxxx]=ode45(@(tttt,xxxx) edx2(tttt,xxxx,K,P,Bp,Cp,XX,UU,Beta1,0,phih,LL,eps),simtime2,xinit);
```

Modification of the ODE function from (d)

```
    % add disturbance
    d = 20;
    if 2 <= tttt
        if tttt <= 2.1
            u = d*(UU*x0-K*(y-Cp*XX*x0));
        else
            u = UU*x0-K*(y-Cp*XX*x0);
        end
    else
        u = UU*x0-K*(y-Cp*XX*x0);
    end
```

Add disturbance to the system from 2s to 2.1s.

```
% get Zeta dot with feedback L
edx2(1:10,:) = (blkdiag(A1,A2,A3,A4,A5) + R)*x0 + eps*LL*(y-Cp*XX*x0);
```

For the feedback case, add the Q2 in the oscillator equation.

Figure1 (e)

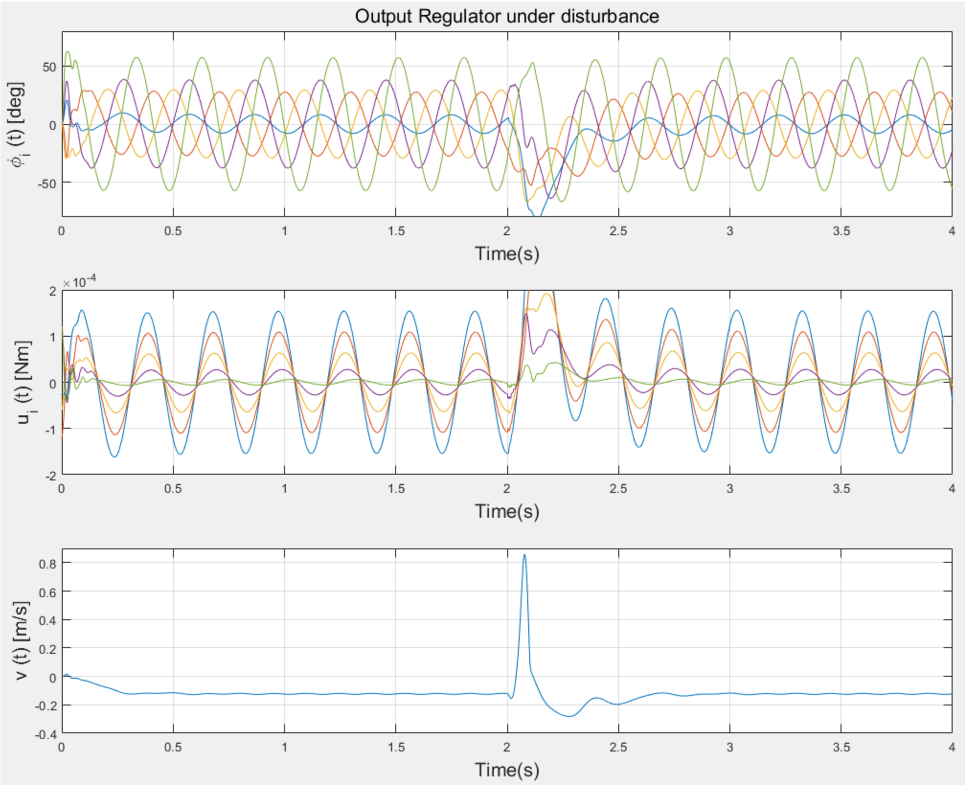
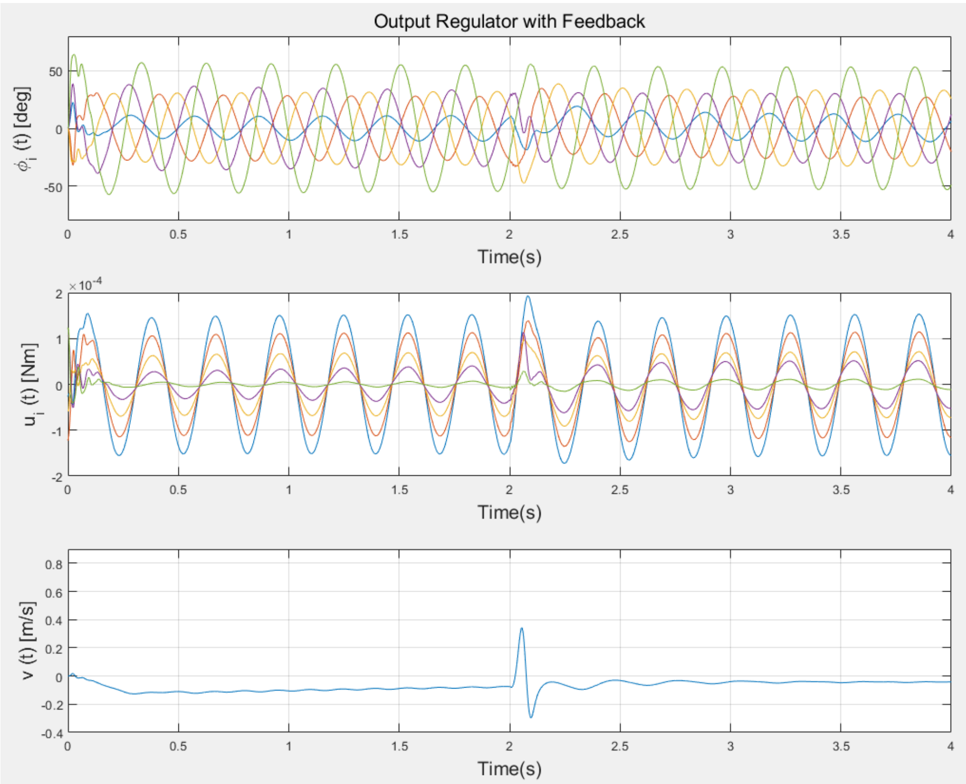


Figure2 (e)



In the first system, disturbance at 2s breaks stable gait easily. And it takes almost 0.4 seconds to recover the unstable state. The same system with proper Q2 feedback attenuates disturbance much more quickly. Indeed, the system is not much affected by disturbance. This result shows that the second system with Q2 feedback is more robust than the first one.