Question 1

PART i)

State equation matrices :

In[ ]:= $A = \begin{pmatrix} 10 & 1 \\ 1 & -2 \end{pmatrix}$; $B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$; $Cm = \begin{pmatrix} 1 & 1 \end{pmatrix}$;

Controllability :

In[ ]:=
```
n = Dimensions[A][[1]];
r = Dimensions[B][[2]];
m = Dimensions[Cm][[1]];
Print["n = ", n, "\t r = ", r, "\t m = ", m]
P = Join[B, A.B, 2];
Print["P = ", MatrixForm[P]]
Print["Rank of P: ", MatrixRank[P]]
If[MatrixRank[P] == n, Print["System is controllable."],
 Print["System is uncontrollable."]]
```

n = 2 r = 1 m = 1

$P = \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}$

Rank of P: 2

System is controllable.

Observability :

In[ ]:=
```
Q = Join[Cm^T, A^T.Cm^T, 2];
Print["Q = ", MatrixForm[Q]]
Print["rank(Q) = ", MatrixRank[Q]]
If[MatrixRank[Q] == n, Print["System is observable."], Print["System is not observable."]]
```

$Q = \begin{pmatrix} 1 & 11 \\ 1 & -1 \end{pmatrix}$

rank(Q) = 2

System is observable.

In[∘]:=
```
Eigenvalues[A] // N
```
Out[∘]=
```
{10.0828, -2.08276}
```

One of the eigenvalue is positive. Thus, the system is unstable.

PART ii)

I will use the following approach:

$x(0) = Q^{-1} y(0) - Q^{-1}\, T\, u(0)$

In[∘]:=
```
Print["Cm.b equals to", Cm.B]
TT = {{0, 0}, {1, 0}}; (*T matrix in the course notes. *)
TT // MatrixForm
```
```
Cm.b equals to{{1}}
```
Out[∘]//MatrixForm=

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

$y(0) = \begin{pmatrix} y[0] \\ y'[0] \end{pmatrix}$, $u0 = \begin{pmatrix} u[0] \\ u'[0] \end{pmatrix}$

In[∘]:=
```
Y0 = {{3}, {1}};
U0 = {{1}, {0}};
```

In[∘]:=
```
X0 = Inverse[Q].Y0 - Inverse[Q].TT.U0;
Print["x[0]=", X0 // MatrixForm]
```

$x[0] = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \end{pmatrix}$

PART iii) Transfer Function

In[∘]:=
```
Inn = IdentityMatrix[n];
```

In[∘]:=
```
G[s] = (Cm.Inverse[s Inn - A].B // Flatten // Simplify)[[1]] // Simplify;
Print["G(s) = ", G[s]]
```

$G(s) = \dfrac{-9 + s}{-21 - 8 s + s^2}$

In[∘]:=
```
Solve[Denominator[G[s]] == 0, s] // Flatten
```
Out[∘]=

$\{s \to 4 - \sqrt{37},\ s \to 4 + \sqrt{37}\}$

System is unstable!

Part iv) Controllable and Observable Canonical Form

I did this part by hand . Please check the papers given to you (Page 1).

Question 2

In[ ]:= Quit[]

In[ ]:= $G[s] = \begin{pmatrix} \frac{1}{s+2} & \frac{s-1}{s+1} \\ \frac{3s}{s+2} & \frac{2}{s+1} \end{pmatrix};$

```
Print["G(s) = ", G[s], "\nlim_{s→∞} G(s) = ", Limit[G[s], s → ∞]]
```

$G(s) = \left\{\left\{\frac{1}{2+s}, \frac{-1+s}{1+s}\right\}, \left\{\frac{3s}{2+s}, \frac{2}{1+s}\right\}\right\}$

$\lim_{s\to\infty} G(s) = \{\{0, 1\}, \{3, 0\}\}$

$G(s)$ is not strictly proper. We need to convert that before continue.

In[ ]:= $DD = \begin{pmatrix} 0 & 1 \\ 3 & 0 \end{pmatrix};$

```
H[s] = DD - G[s] // Simplify;
Print["H(s) = ", H[s] // MatrixForm, "\nlim_{s→∞} H(s) = ", Limit[H[s], s → ∞] // MatrixForm]
```

$H(s) = \begin{pmatrix} -\frac{1}{2+s} & \frac{2}{1+s} \\ \frac{6}{2+s} & -\frac{2}{1+s} \end{pmatrix}$

$\lim_{s\to\infty} H(s) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Now H (s) is strictly proper!

ExtendedHermiteForm and other helpful functions given in course documents:

Part i) Minimal MFD

In[•]:=
```
N1[s] = Numerator[H[s]]; D1[s] = {{s + 2, 0}, {0, s + 1}};
Print["N(s) = ", N1[s], "\tD(s) = ", D1[s],
 "\tH(s) = N(s)D^-1(s) = ", N1[s].Inverse[D1[s]] // Simplify]
```

N(s) = {{-1, 2}, {6, -2}}   D(s) = {{2 + s, 0}, {0, 1 + s}}

$H(s) = N(s)D^{-1}(s) = \left\{\left\{-\frac{1}{2+s}, \frac{2}{1+s}\right\}, \left\{\frac{6}{2+s}, -\frac{2}{1+s}\right\}\right\}$

In[•]:=
```
DN[s_] := Join[D1[s], N1[s]];
Print[{{"D(s)"}, {"N(s)"}}, " = ", DN[s]]
{h[s], U[s]} = ExtendedHermiteForm[DN[s], s];
Wg = Take[h[s], Min[Dimensions[h[s]]]];
Print["W_g = ", Wg]
```

{{D(s)}, {N(s)}} = {{2 + s, 0}, {0, 1 + s}, {-1, 2}, {6, -2}}

W_g = {{1, 0}, {0, 1}}

Since $W_g = I$, the realization is minimal.

In[•]:=
```
Print["det(D(s)) = ", Det[D1[s]],
 "\nThe minimum number of states is ", Exponent[Det[D1[s]], s]]
```

det(D(s)) = $2 + 3 s + s^2$
The minimum number of states is 2

Controller form of right MFD

Build up $S(s)$ and D_{hc} :

In[•]:=
```
Print["D(s) = ", D1[s] // Expand]
```

D(s) = {{2 + s, 0}, {0, 1 + s}}

In[•]:=
```
n = Length[D1[s]];
k = Table[Max[Exponent[D1[s]^T[[i]], s]], {i, 1, n}];
S[s] = DiagonalMatrix[s^k];
Dhc = Coefficient[D1[s]^T, s^k]^T;
Print["k = ", k, "\tS(s) = ", S[s], "\t D_hc = ", Dhc]
```

k = {1, 1}   S(s) = {{s, 0}, {0, s}} D_{hc} = {{1, 0}, {0, 1}}

Build up $\psi(s)$, $L(s)$, and D_{lc} :

In[•]:=
```
L[s] = D1[s] - Dhc.S[s] // Expand; Print["L(s) = ", L[s] // MatrixForm]
```

$L(s) = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$

```
In[∘]:= p = Table[{Reverse[s^Range[0,k[[i]]-1]]}, {i, 1, n}]
        ψ[s] = myBlockDiagonalMatrix[p]^T;
        ψ[s] // MatrixForm

Out[∘]= {{{1}}, {{1}}}

Out[∘]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
In[∘]:= d2 = Array[d_## &, {n, Total[k]}];
        sol1 = SolveAlways[Thread[Flatten[L[s]] == Flatten[d2.ψ[s]]], s] // Flatten;
        Dlc = d2 /. sol1;
        Print["D_1c = ", d2, " = ", Dlc]
```

$D_{1c} = \{\{d_{1,1}, d_{1,2}\}, \{d_{2,1}, d_{2,2}\}\} = \{\{2, 0\}, \{0, 1\}\}$

Build up A_c^0 and B_c^0:

```
In[∘]:= Ai = Table[DiagonalMatrix[ConstantArray[1, Max[k[[i]] - 1, 1]], -1, k[[i]]], {i, 1, n}];
        Ac0 = myBlockDiagonalMatrix[Ai];
        Bi = Table[{UnitVector[k[[i]], 1]}, {i, 1, n}];
        Bc0 = myBlockDiagonalMatrix[Bi]^T;
        Print["A_c^0 = ", Ac0 // MatrixForm, "\t B_c^0 = ", Bc0 // MatrixForm]
```

$A_c^0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad B_c^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Obtain the matrices for the controller - form state space realization :

```
In[∘]:= Ac = Ac0 - Bc0.Inverse[Dhc].Dlc;
        Bc = Bc0.Inverse[Dhc];
        Print["A_c = ", Ac // MatrixForm, "\t B_c = ", Bc // MatrixForm]
```

$A_c = \begin{pmatrix} -2 & 0 \\ 0 & -1 \end{pmatrix} \qquad B_c = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

```
In[∘]:= nlc2 = Array[nlc_## &, {Length[N1[s]], Total[k]}];
        sol2 = SolveAlways[Thread[Flatten[N1[s]] == Flatten[nlc2.ψ[s]]], s] // Flatten;
        Cc = nlc2 /.
          sol2;
        Print["N_1c = C_c = ", Cc // MatrixForm]
```

$N_{1c} = C_c = \begin{pmatrix} -1 & 2 \\ 6 & -2 \end{pmatrix}$

Part ii) Pole Placement

```
In[∘]:= Clear[p2]
        αd[s] = (s - p1) (s - p2) /. {p1 → -10, p2 → -10} // Expand;
        Print["α(s) = ", αd[s]]
```

$\alpha(s) = 100 + 20\, s + s^2$

In[∘]:=
```
n = Exponent[αd[s], s];
α[s] = s^n + α1 s^(n-k[[1]]) + α2 s^(n-k[[1]]-k[[2]])
```

Out[∘]= $s^2 + s\,\alpha1 + \alpha2$

In[∘]:=
```
solα = SolveAlways[α[s] == αd[s] /. {p1 → -10, p2 → -10}, {s}] // Flatten
αM[s] = {{α1, α2}, {-1, 0}} /. solα
```

Out[∘]= {α1 → 20, α2 → 100}

Out[∘]= {{20, 100}, {-1, 0}}

In[∘]:=
```
Kc = Dhc.αM[s] - Dlc;
Print["Gain Matrix: K_c = ", Kc // MatrixForm]
```

Gain Matrix: $K_c = \begin{pmatrix} 18 & 100 \\ -1 & -1 \end{pmatrix}$

In[∘]:=
```
Inn = IdentityMatrix[n];
```

In[∘]:=
```
Det[s Inn - Ac + Bc.Kc]
Eigenvalues[Ac - Bc.Kc]
```

Out[∘]= $100 + 20\,s + s^2$

Out[∘]= {-10, -10}

Part iii) Simulation

In[∘]:=
```
Clear[v];
A = Ac;
b = Bc;
c = Cc;
K = Kc;
x[t_] := Array[x_#[t] &, Length[A]]
y[t_] := c.x[t];
v[t_] := {0, 0}
u[t_] = v[t] - K.x[t];
ClosedLoopEq = Thread[D[x[t], t] == (A - b.K).x[t] + b . v[t] // Chop];
ColumnForm[ClosedLoopEq]
```

Out[∘]=
$x_1'[t] == -20\,x_1[t] - 100\,x_2[t]$
$x_2'[t] == x_1[t]$

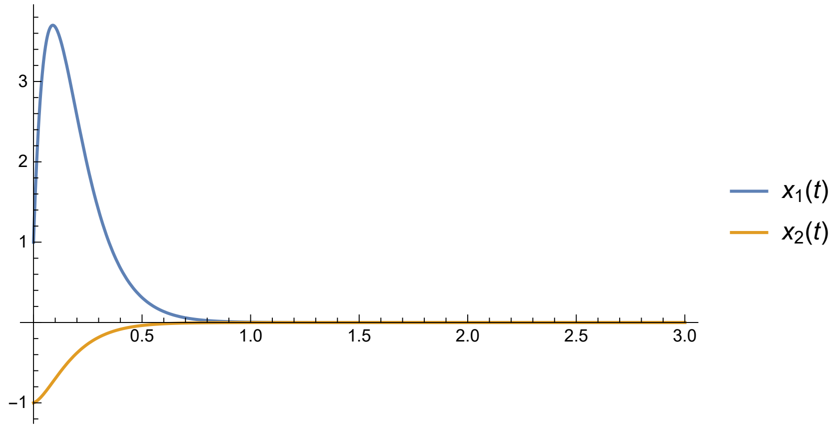
I will only use 2 initial condition since deg[det[D (s)]] = 2 and realization is minimal

```
IC = Thread[x[0] == {1, -1}];
tmax = 3;
CLSol = NDSolve[{ClosedLoopEq, IC} // Flatten, x[t], {t, 0, tmax}];
```

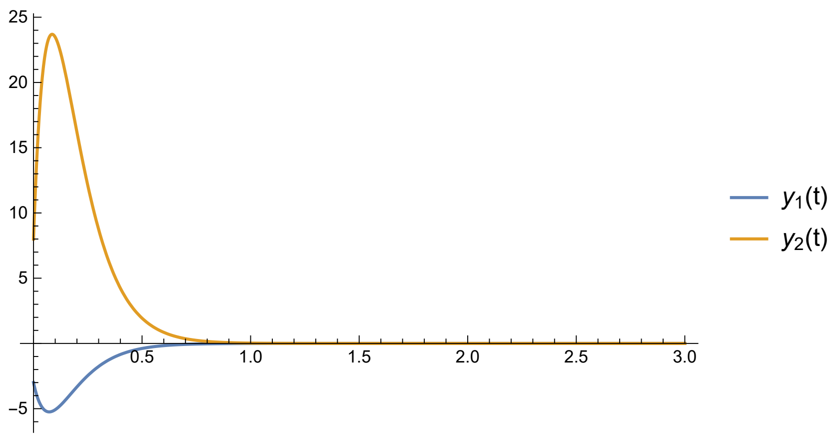
In[ ]:=

```
Plot[{Evaluate[x[t] /. CLSol]}, {t, 0, tmax}, PlotRange → All, PlotLegends → {x[t]}]
Plot[{Evaluate[c . x[t] /. CLSol]}, {t, 0, tmax},
 PlotRange → All, PlotLegends → {"y₁(t)", "y₂(t)"}]
Plot[{Evaluate[-K.x[t] /. CLSol]}, {t, 0, tmax},
 PlotRange → All, PlotLegends → {"u₁(t)", "u₂(t)"}]
```

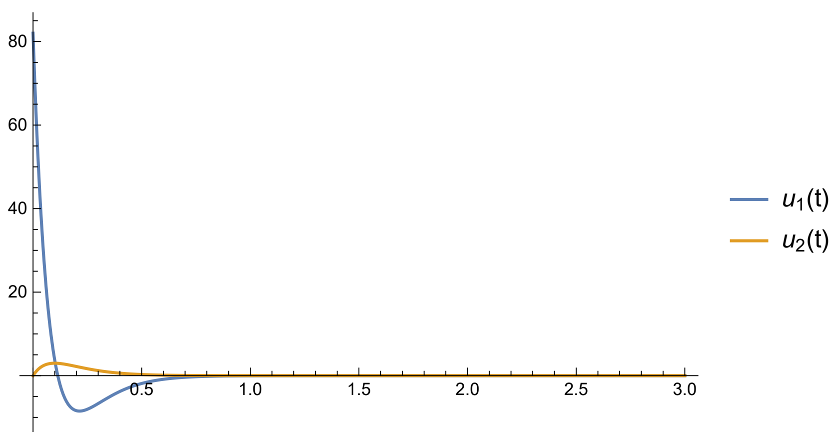
Out[ ]=



Out[ ]=



Out[ ]=



Question 3

In[∘]:=
```
Quit[]
```

Part i) State Space Realization

Construct N and D matrices.

In[∘]:=
```
N1[s] = (1, s-1, s^2); D1[s] = {{s (s-1) (s+1)}};
Print["N(s) = ", N1[s], "\tD(s) = ", D1[s]]
N1[s].Inverse[D1[s]] == G[s] // ReleaseHold
```

N(s) = $\{\{1\}, \{-1+s\}, \{s^2\}\}$ D(s) = $\{\{(-1+s)\, s\, (1+s)\}\}$

Out[∘]= $\left\{\left\{\frac{1}{(-1+s)\, s\, (1+s)}\right\}, \left\{\frac{1}{s\,(1+s)}\right\}, \left\{\frac{s}{(-1+s)\,(1+s)}\right\}\right\} == G[s]$

In[∘]:=
```
Print["det(D(s)) = ", Det[D1[s]],
 "\nThe minimum number of states is ", Exponent[Det[D1[s]], s]]
```

det(D(s)) = (−1 + s) s (1 + s)
The minimum number of states is 3

Find the GRCD:

Make: $DN(s) = \begin{pmatrix} D(s) \\ N(s) \end{pmatrix}$

In[∘]:=
```
DN[s] = Join[D1[s], N1[s]]; TraditionalForm[DN[s]]
```

Out[∘]//TraditionalForm=

$$\begin{pmatrix} (s-1)\, s\, (s+1) \\ 1 \\ s-1 \\ s^2 \end{pmatrix}$$

ExtendedHermiteForm and other helpful functions given in course documents:

Minimal Order State Space Realization

In[∘]:=
```
Clear[H, U]
{H[s], U[s]} = ExtendedHermiteForm[DN[s], s];
H[s] // TraditionalForm
 U[s] // Simplify // TraditionalForm;
Wg[s] = Take[H[s], Length[D1[s]]];
Print["Wg(s) = ", Wg[s]]
```

Out[∘]//TraditionalForm=

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$W_g(s)$ = {{1}}

$W_g(s) = I.$

Realization already has the minimal number of states.

Controller Form Realization of MFD

In[∘]:=
```
Print["D(s) = ", D1[s] // Expand]
```

D(s) = $\{\{-s + s^3\}\}$

In[∘]:=
```
j = Length[D1[s]];
k = Table[Max[Exponent[D1[s]ᵀ[[i]], s]], {i, 1, j}];
S[s] = DiagonalMatrix[s^k];
Dhc = Coefficient[D1[s]ᵀ, s^k]ᵀ;
Print["k = ", k // MatrixForm, ";\tS(s) = ", S[s], ";\t Dhc = ", Dhc]
```

k = (3); S(s) = $\{\{s^3\}\}$; D_{hc} = {{1}}

Build up $\psi(s)$, $L(s)$, and D_{lc} :

In[∘]:=
```
L[s] = D1[s] - Dhc.S[s] // Expand; Print["L(s) = ", L[s]]
```

L(s) = {{-s}}

In[∘]:=
```
p2 = Table[{Reverse[s^Range[0,k[[i]]-1]]}, {i, 1, j}]
ψ[s] = myBlockDiagonalMatrix[p2]ᵀ
```

Out[∘]= $\{\{\{s^2, s, 1\}\}\}$

Out[∘]= $\{\{s^2\}, \{s\}, \{1\}\}$

In[∘]:=
```
d2 = Array[d## &, {j, Total[k]}];
sol1 = SolveAlways[Thread[Flatten[L[s]] == Flatten[d2.ψ[s]]], s] // Flatten;
Dlc = d2 /. sol1;
Print["Dlc = ", Dlc]
```

D_{lc} = {{0, -1, 0}}

In[∘]:=
```
Dlc.ψ[s] + Dhc.S[s]
```

Out[∘]= $\{\{-s + s^3\}\}$

Build up A_c^0 and B_c^0:

In[∘]:=
```
Ai = Table[DiagonalMatrix[ConstantArray[1, Max[k[[i]] - 1, 1]], -1, k[[i]]], {i, 1, j}];
Ac0 = myBlockDiagonalMatrix[Ai];
Bi = Table[{UnitVector[k[[i]], 1]}, {i, 1, j}];
Bc0 = myBlockDiagonalMatrix[Bi]ᵀ;
Print["A_c^0 = ", Ac0, "\t B_c^0 = ", Bc0]
```

A_c^0 = {{0, 0, 0}, {1, 0, 0}, {0, 1, 0}} B_c^0 = {{1}, {0}, {0}}

Obtain the matrices for the controller - form state space realization :

In[∘]:=
```
A = Ac0 - Bc0.Inverse[Dhc].Dlc;
B = Bc0.Inverse[Dhc];
```

In[∘]:=
```
nlc2 = Array[nlc## &, {Length[N1[s]], Total[k]}];
sol2 = SolveAlways[Thread[Flatten[N1[s]] == Flatten[nlc2.ψ[s]]], s] // Flatten;
Cm = nlc2 /. sol2;
```

In[∘]:=
```
Print["A_c = ", A // TraditionalForm, "\t B_c = ",
 B // TraditionalForm, "\t C = ", Cm // TraditionalForm]
```

$A_c = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ $B_c = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ $C = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \end{pmatrix}$

State Space Realization is found!

State Space Eqns

In[∘]:=
```
x[t_] := Array[x#[t] &, Total[k]]
y[t_] := Array[y#[t] &, Length[N1[s]]]
u[t_] := Array[u#[t] &, Length[Bcᵀ]]
```

In[∘]:=
```
SSEQ = Thread[D[x[t], t] == A.x[t] + B.u[t]] // Chop; ColumnForm[SSEQ]
OEQ = Thread[y[t] == Cm.x[t]] // Chop;
ColumnForm[OEQ]
```

Out[∘]=
$x_1'[t] == u_1[t] + x_2[t]$
$x_2'[t] == x_1[t]$
$x_3'[t] == x_2[t]$

Out[∘]=
$y_1[t] == x_3[t]$
$y_2[t] == x_2[t] - x_3[t]$
$y_3[t] == x_1[t]$

Part ii)

Find Q, Sf and R matrices
Sf=0; Q=100 I and R= 0.01 I

In[∘]:=
```
Iy = IdentityMatrix[Length[y[t]]]
Iu = IdentityMatrix[Length[u[t]]]
n = Length[A];
```

Out[∘]= {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}

Out[∘]= {{1}}

In[∘]:=
```
Sf = ConstantArray[0, Dimensions[Iy]]; Q = 100 Iy; R = 0.01 Iu; tf = 10;
Print["Sf = ", Sf // MatrixForm, "\t Q = ", Q // MatrixForm, "\t R = ", R // MatrixForm]
```

$$S_f = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad Q = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{pmatrix} \qquad R = \begin{pmatrix} 0.01 \end{pmatrix}$$

Riccati Equation (tf is finite)

In[∘]:=
```
S[t_] := Array[Subscript[S, Sequence @@ Through[{Min, Max}[##]]][t] &, {n, n}]
S[t]
LHRE = D[S[t], t] + S[t].A - S[t].B.Inverse[R].B^T.S[t] + Cm^T.Q.Cm + A^T.S[t]
```

Out[∘]= $\{\{S_{1,1}[t], S_{1,2}[t], S_{1,3}[t]\}, \{S_{1,2}[t], S_{2,2}[t], S_{2,3}[t]\}, \{S_{1,3}[t], S_{2,3}[t], S_{3,3}[t]\}\}$

Out[∘]=
$$\begin{aligned}
&\{\{100. - 100.\, S_{1,1}[t]^2 + 2\, S_{1,2}[t] + S_{1,1}'[t],\\
&\quad 0. + S_{1,1}[t] - 100.\, S_{1,1}[t]\, S_{1,2}[t] + S_{1,3}[t] + S_{2,2}[t] + S_{1,2}'[t],\\
&\quad 0. - 100.\, S_{1,1}[t]\, S_{1,3}[t] + S_{2,3}[t] + S_{1,3}'[t]\},\\
&\{0. + S_{1,1}[t] - 100.\, S_{1,1}[t]\, S_{1,2}[t] + S_{1,3}[t] + S_{2,2}[t] + S_{1,2}'[t],\\
&\quad 100. + 2\, S_{1,2}[t] - 100.\, S_{1,2}[t]^2 + 2\, S_{2,3}[t] + S_{2,2}'[t],\\
&\quad -100. + S_{1,3}[t] - 100.\, S_{1,2}[t]\, S_{1,3}[t] + S_{3,3}[t] + S_{2,3}'[t]\},\\
&\{0. - 100.\, S_{1,1}[t]\, S_{1,3}[t] + S_{2,3}[t] + S_{1,3}'[t],\\
&\quad -100. + S_{1,3}[t] - 100.\, S_{1,2}[t]\, S_{1,3}[t] + S_{3,3}[t] + S_{2,3}'[t],\ 200. - 100.\, S_{1,3}[t]^2 + S_{3,3}'[t]\}\}
\end{aligned}$$

Due to symmetry there are duplicate equations :

In[∘]:=
```
Clear[i, j]
```

Eliminate duplicate equations:

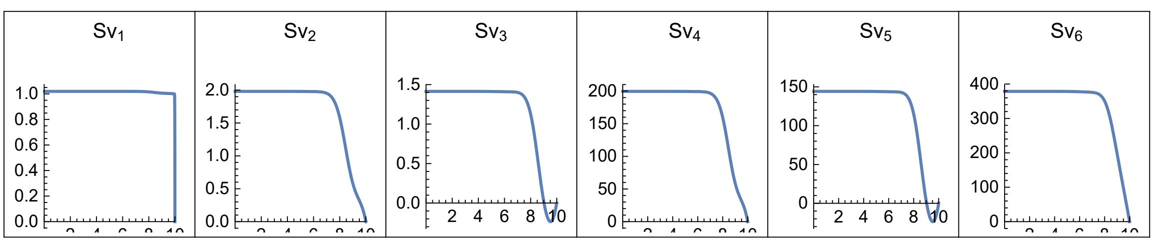
In[∘]:=
```
UpperElements[M_] := Flatten[Table[M[[i, j]], {i, Length[M]}, {j, i, Length[M]}]]
LHREup = UpperElements[LHRE];
Ov = ConstantArray[0, Length[LHREup]];
RE = Thread[LHREup == Ov];
Sv[t_] := UpperElements[S[t]]
Print["RE: ", RE // MatrixForm]
```

RE:
$$\begin{pmatrix} 100. - 100.\, S_{1,1}[t]^2 + 2\, S_{1,2}[t] + S_{1,1}'[t] == 0 \\ 0. + S_{1,1}[t] - 100.\, S_{1,1}[t]\, S_{1,2}[t] + S_{1,3}[t] + S_{2,2}[t] + S_{1,2}'[t] == 0 \\ 0. - 100.\, S_{1,1}[t]\, S_{1,3}[t] + S_{2,3}[t] + S_{1,3}'[t] == 0 \\ 100. + 2\, S_{1,2}[t] - 100.\, S_{1,2}[t]^2 + 2\, S_{2,3}[t] + S_{2,2}'[t] == 0 \\ -100. + S_{1,3}[t] - 100.\, S_{1,2}[t]\, S_{1,3}[t] + S_{3,3}[t] + S_{2,3}'[t] == 0 \\ 200. - 100.\, S_{1,3}[t]^2 + S_{3,3}'[t] == 0 \end{pmatrix}$$

Solve RE :

In[∘]:=
```
solS =
  NDSolve[{RE, Thread[Sv[tf] == UpperElements[Cmᵀ.Sf.Cm]]}, Sv[t], {t, 0, tf}] // Quiet //
   Flatten;
(*Table[Plot[Evaluate[Sv[t][[i]]/.SolS],{t,0,tf}, PlotRange→{{0,tf},All},
  PlotLabel→"Sv"i],{i,1,Length[Sv[t]]}]*)
```

In[∘]:=
```
(*For[i=1,i<=Length[Sv[t]],i++,Print[Plot[Evaluate[Sv[t][[i]]/.solS],
   {t,0,tf}, PlotRange→{{0,tf},All},PlotLabel→"Sv"i]]]*)
GraphicsGrid[{Table[Plot[Evaluate[Sv[t][[i]] /. solS], {t, 0, tf},
    PlotRange → {{0, tf}, All}, AspectRatio → Full, PlotLabel →
     ToString[Subscript["Sv", i], StandardForm]], {i, 1, Length[Sv[t]]}]}, Frame → All]
```

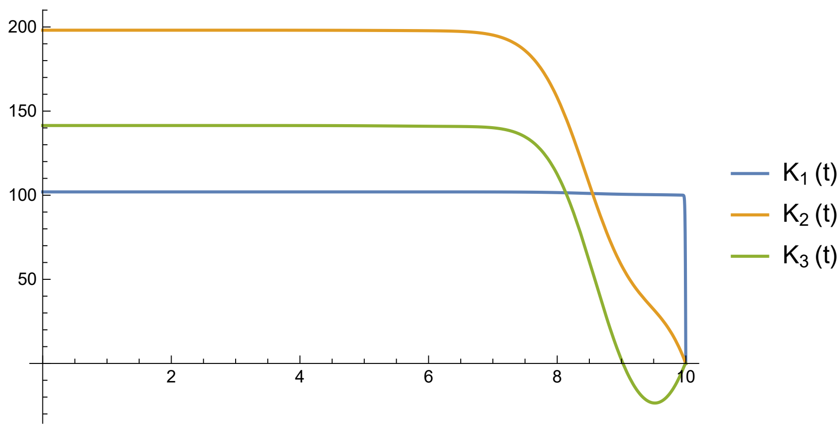
Out[∘]=


LQR gain matrix :

In[◦]:=
```
Clear[Kop]
Kop[t_] := (Inverse[R].Bᵀ.S[t])
Kop[t] // Chop
Plot[Evaluate[Kop[t] /. solS], {t, 0, tf},
 PlotRange → All, PlotLegends → Array["K"# "(t)" &, 3]]
```

Out[◦]= $\{\{100.\ S_{1,1}[t],\ 100.\ S_{1,2}[t],\ 100.\ S_{1,3}[t]\}\}$

Out[◦]=



State Equation :

In[◦]:=
```
uop[t_] := -Kop[t].x[t]
StEqn = Thread[D[x[t], t] == A.x[t] + B.uop[t]] // Chop;
ColumnForm[StEqn]
```

Out[◦]=

$x_1'[t] == x_2[t] - 100.\ x_1[t]\ S_{1,1}[t] - 100.\ x_2[t]\ S_{1,2}[t] - 100.\ x_3[t]\ S_{1,3}[t]$

$x_2'[t] == x_1[t]$

$x_3'[t] == x_2[t]$

Tracking eqns:

In[◦]:=
```
ξ[t_] := Array[ξ#[t] &, n];
w[t_] := Array[w#[t] &, n];
η[t_] := Array[η#[t] &, n];
ξEq = Thread[-D[ξ[t], t] + S[t].B.Inverse[R].Bᵀ.ξ[t] + S[t].w[t] - Cmᵀ.Q.η[t] - Aᵀ.ξ[t] ==
      Table[0, Length[ξ[t]]]] /. Thread[w[t] → {0, 0, 0}] // Chop;
(*S[t] yerine Sr koyarsan olur zamanla degismeyen K icin.*)
ColumnForm[ξEq]
```

Out[◦]=

$-100\ \eta_3[t] - \xi_2[t] + 100.\ \xi_1[t]\ S_{1,1}[t] - \xi_1'[t] == 0$

$-100\ \eta_2[t] - \xi_1[t] - \xi_3[t] + 100.\ \xi_1[t]\ S_{1,2}[t] - \xi_2'[t] == 0$

$-100\ \eta_1[t] + 100\ \eta_2[t] + 100.\ \xi_1[t]\ S_{1,3}[t] - \xi_3'[t] == 0$

Desired tracking

In[ ]:=
```
yd[t_] := ao Cos[ω t]
η[t_] := {yd[t], yd'[t], yd''[t]}; Print["η(t)  = ", η[t] // MatrixForm]
ξEq = Thread[-D[ξ[t], t] + S[t].B.Inverse[R].Bᵀ.ξ[t] + S[t].w[t] - Cmᵀ.Q.η[t] - Aᵀ.ξ[t] ==
      Table[0, Length[ξ[t]]]] /. Thread[w[t] → {0, 0, 0}] // Chop;
ColumnForm[ξEq]
```

$$\eta(t) \;=\; \begin{pmatrix} \text{ao} \operatorname{Cos}[t\,\omega] \\ -\text{ao}\,\omega \operatorname{Sin}[t\,\omega] \\ -\text{ao}\,\omega^2 \operatorname{Cos}[t\,\omega] \end{pmatrix}$$

Out[ ]=

$100\,\text{ao}\,\omega^2 \operatorname{Cos}[t\,\omega] - \xi_2[t] + 100.\ \xi_1[t]\ S_{1,1}[t] - \xi_1'[t] == 0$

$100\,\text{ao}\,\omega \operatorname{Sin}[t\,\omega] - \xi_1[t] - \xi_3[t] + 100.\ \xi_1[t]\ S_{1,2}[t] - \xi_2'[t] == 0$

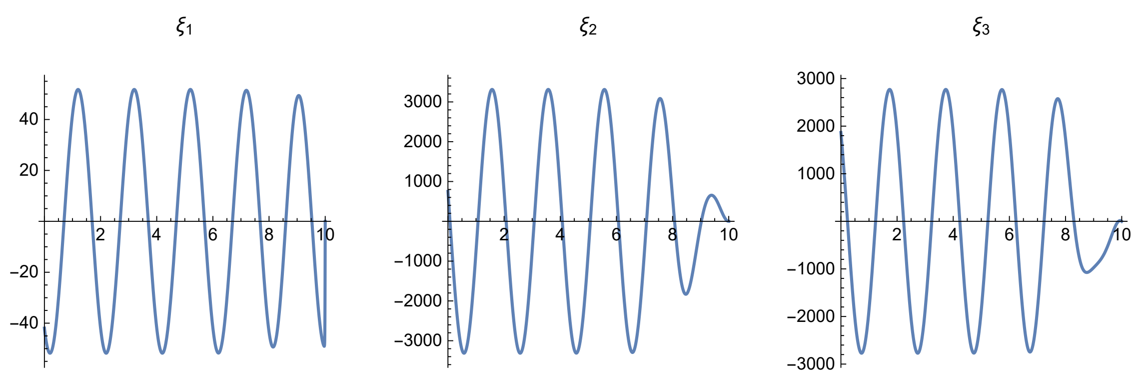
$-100\,\text{ao} \operatorname{Cos}[t\,\omega] - 100\,\text{ao}\,\omega \operatorname{Sin}[t\,\omega] + 100.\ \xi_1[t]\ S_{1,3}[t] - \xi_3'[t] == 0$

Part iii) Simulation

Given values:

In[ ]:=
```
Vals = {ao → 5, ω → π};
Solξ = NDSolve[{ξEq /. solS, Thread[ξ[tf] == Cmᵀ.Sf.η[tf]]} /. Vals // Flatten,
    ξ[t], {t, 0, tf}] // Flatten;
GraphicsGrid[{Table[Plot[Evaluate[ξ[t][[i]] /. Solξ], {t, 0, tf},
    PlotRange → All, PlotLabel → ξi, AspectRatio → 1], {i, 1, Length[ξ[t]]}]}]
```

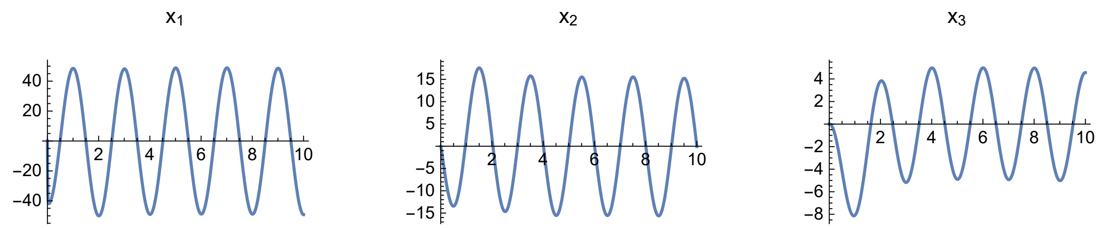
Out[ ]=



State Responses

In[∘]:=
```
u[t_] := -Kop[t].x[t] + Inverse[R].Bᵀ.ξ[t]
Solx = NDSolve[{(D[x[t], t] == A.x[t] + B.u[t] /. Vals /. solS /. Solξ), x[0] == {0, 0, 0}},
    x[t], {t, 0, tf}][[1]];
GraphicsGrid[{Table[Plot[Evaluate[x[t][[i]] /. Solx], {t, 0, tf},
    PlotRange → All, PlotLabel → "x"i], {i, 1, Length[x[t]]}]}]
```

Out[∘]=

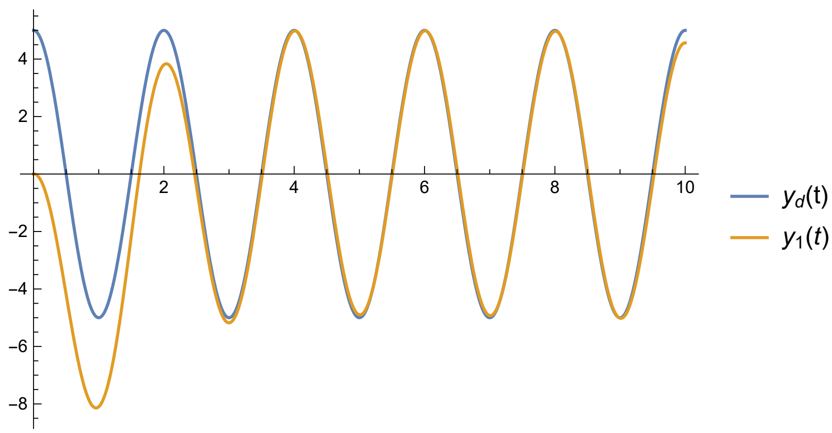


Output Plot :

In[∘]:=
```
yo[t_] := (Cm.x[t]) /. Solx
```

In[∘]:=
```
Plot[{yd[t] /. Vals, yo[t][[1]]}, {t, 0, tf}, PlotLegends → {"yd(t)", y[t][[1]]}]
```

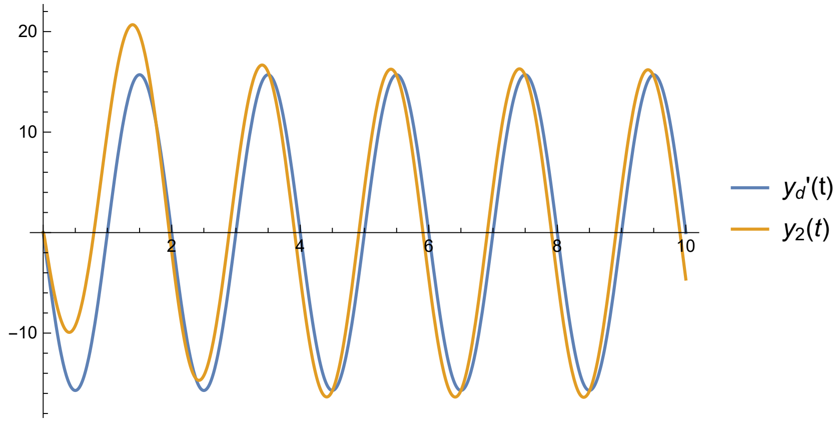
Out[∘]=



First Derivative

In[ ]:=
```
Plot[{yd'[t] /. Vals // Chop, yo[t][[2]]}, {t, 0, tf}, PlotLegends → {"yd'(t)", y[t][[2]]}]
```

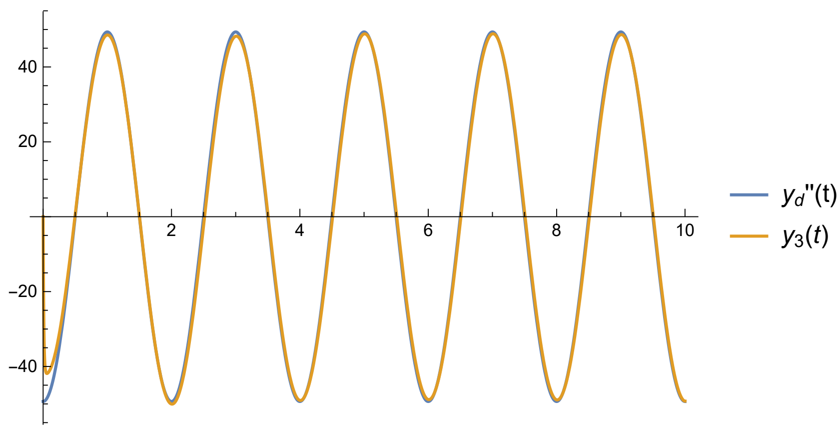
Out[ ]=



Second Derivative

In[ ]:=
```
Plot[{yd''[t] /. Vals // Chop, yo[t][[3]]},
 {t, 0, tf}, PlotLegends → {"yd''(t)", y[t][[3]]}]
```

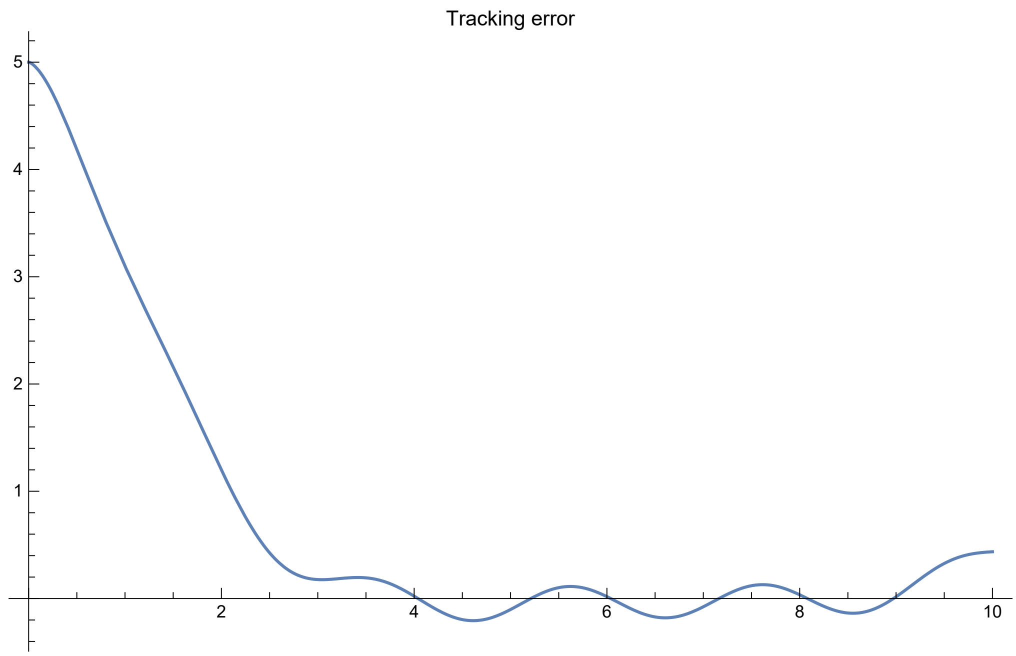
Out[ ]=



Tracking Error Plot:

In[∘]:=
```
Plot[{yd[t] - yo[t][[1]] /. Solx /. Vals},
 {t, 0, tf}, PlotLabel → "Tracking error", PlotRange → All]
```

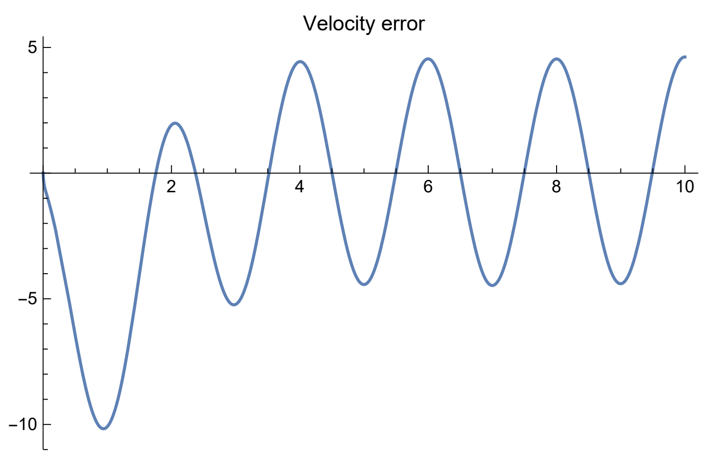
Out[∘]=



Velocity Error :

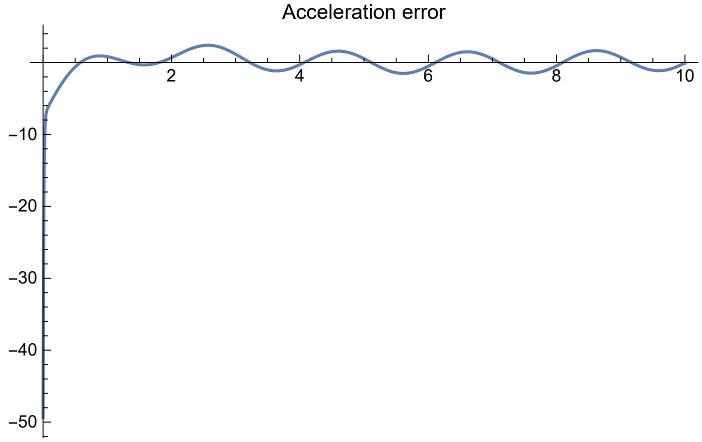
In[∘]:=
```
Plot[{yd'[t] - yo[t][[2]] /. Solx /. Vals},
 {t, 0, tf}, PlotLabel → "Velocity error", PlotRange → All]
```

Out[∘]=



Acceleration Error :

In[ ]:=
```
Plot[{yd''[t] - yo[t][[3]] /. Solx /. Vals},
 {t, 0, tf}, PlotLabel → "Acceleration error", PlotRange → All]
```

Out[ ]=


Objective Function value calculation :

Define tracking error as z :

In[ ]:=
```
z[t_] := η[t] - yo[t]
J = 1/2 NIntegrate[
   (z[t].Q.z[t] + u[t].R.u[t]) /. Vals /. solS /. Solx /. Solξ, {t, 0, tf}, AccuracyGoal → 6]
```

Out[ ]= 12 146.5

System operates as desired. Tracking can be done successfully!