In[ ]:= Quit[]

# Cagatay Duygu
# 48369962

## i. Find the controller-form state space realization using a suitable MFD.

In[1]:= $G[s] = \begin{pmatrix} \frac{s}{s^2-9} & \frac{3}{s+1} \\ \frac{3}{s+3} & \frac{0.5}{s+2} \end{pmatrix};$

G (s) is strictly proper!

In[2]:= H[s] = G[s]

Out[2]= $\left\{\left\{\frac{s}{-9+s^2}, \frac{3}{1+s}\right\}, \left\{\frac{3}{3+s}, \frac{0.5}{2+s}\right\}\right\}$

### Useful functions (run the cell):

In[3]:=
```
BlockDiagonalMatrix[b : {__?MatrixQ}] :=
  Module[{r, c, n = Length[b], i, j}, {r, c} = Transpose[Dimensions /@ b];
   ArrayFlatten[
    Table[If[i == j, b[[i]], ConstantArray[0, {r[[i]], c[[j]]}]], {i, n}, {j, n}]]];
FindMin[at_, kt_, xt_, pt_] :=
  Module[{a = at, k = kt, x = xt, p = pt, i, rgmin, degmin}, For[i = k;
    rgmin = k;
    degmin = +Infinity, i ≤ p, i++,
    If[a[[i]] =!= 0 && Exponent[a[[i]], x] < degmin, rgmin = i;
     degmin = Exponent[a[[i]], x]]];
   rgmin];

Variabile[at_] := Module[{a = at, xt}, xt = Variables[a];
   If[xt === {}, d, xt[[1]]]];

ExtPolQ[lt_List, xt_ : {}] := Module[{x = xt},
   If[x === {} && Length[Variables[lt]] > 1, Message[General::badarg];
    Return[$Failed], If[x === {}, x = Variabile[lt]]];
```

```
(*If all the components of lt are polynomials in x*)
Apply[And, PolynomialQ[#, x] & /@ Flatten[lt]] || (*or in x^-1,
 the function returns True,False otherwise.(Substituting all the symbols x^-n→
     x^n we can use PolynomialQ also in this case)*)Apply[And,
  PolynomialQ[#, x] & /@ Flatten[Expand[lt] /. {(x^(n : _)) → (x^-n), x → x^-1}]]];

ExtendedHermiteForm[at_, xt_ : {}] :=
  Module[{a = at, p, m, x = xt, u, v, k, kcl, i, j, q, deg, esp, coef, rg, rmin, upc, lpc,
     ch = False}, If[x === {} && Length[Variables[a]] > 1, Message[General::badarg];
     Return[$Failed], If[x === {}, x = Variabile[a]]];
   If[Not[ExtPolQ[a, x]], Message[General::pol]; Return[$Failed]];
   If[Not[Apply[And, PolynomialQ[#, x] & /@ Flatten[a]]],
    a = a /. {(x^(n : _)) → (x^-n)};
    ch = True];
   (*Initialize the variables*){p, m} = Dimensions[a];
   u = IdentityMatrix[p];
   (*Main loop on k (and kcl)*)For[k = 1;
    kcl = 1, k ≤ p && kcl ≤ m, k++, (*Find the first non-zero element in row k*)
    While[a〚Range[k, p], kcl〛 === Table[0, {p - k + 1}] && kcl < m, kcl++];
    (*With this loop we eliminate all the elements in column kcl below position
     k*)While[If[k == p, False, a〚Range[k + 1, p], kcl〛 =!= Table[0, {p - k}]],
     (*Put the least degree element within column kcl in position (k,kcl)*)
     rmin = FindMin[Transpose[a]〚kcl〛, k, x, p];
     If[rmin ≠ k, a = a /. {a〚rmin〛 → a〚k〛, a〚k〛 → a〚rmin〛};
      u = u /. {u〚rmin〛 → u〚k〛, u〚k〛 → u〚rmin〛}];
     (*Lower the degree of non-
      zero polynomials in column kcl below a〚k,kcl〛*)For[i = k + 1, i ≤ p, i++,
      If[a〚i, kcl〛 =!= 0, lpc = PolynomialQuotient[a〚i, kcl〛, a〚k, kcl〛, x];
       a = ReplacePart[a, Together[a〚i〛 - a〚k〛 * lpc], i];
       u = ReplacePart[u, Expand[u〚i〛 - u〚k〛 * lpc], i]]]];
    (*Endwhile*)(*In column kcl lower the degree of polynomials above
     a〚k,kcl〛 having the degree higher than that of a〚k,kcl〛*)If[a〚k, kcl〛 =!= 0,
     For[i = 1, i < k, i++, If[a〚i, kcl〛 =!= 0, If[Exponent[a〚i, kcl〛, x] ≥
         Exponent[a〚k, kcl〛, x], lpc = PolynomialQuotient[a〚i, kcl〛, a〚k, kcl〛, x];
        a = ReplacePart[a, Together[a〚i〛 - a〚k〛 * lpc], i];
        u = ReplacePart[u, Expand[u〚i〛 - u〚k〛 * lpc], i]]]]];
    (*Make a〚k,kcl〛 monic*)
    If[a〚k, kcl〛 =!= 0, lpc = Coefficient[a〚k, kcl〛, x, Exponent[a〚k, kcl〛, x]];
     If[lpc =!= 1, a = ReplacePart[a, Expand[a〚k〛 / lpc], k];
      u = ReplacePart[u, Expand[u〚k〛 / lpc], k]]];
    kcl++];
   (*Endfor*)(*Put all the zero rows in the last positions*)
   For[k = 1, k < p, k++, If[a〚k〛 === Table[0, {m}], i = p;
     While[a〚i〛 === Table[0, {m}] && i > k, i = i - 1];
```

```
    If[i > k, For[j = k, j ≤ i, j++, a = ReplacePart[a, a[[j + 1]], j];
      u = ReplacePart[u, u[[j + 1]], j]]];] (*endif*)];
  (*endfor*) If[ch, {a, u} = {a, u} /. {x^(n : _) → x^-n, x → x^-1}];
  {a, u}] /; MatrixQ[at] || Message[General::mtrx, "ExtendedHermiteForm"];
```

SetDelayed : Tag BlockDiagonalMatrix in BlockDiagonalMatrix [b : {__?MatrixQ }] is Protected.

## Find the $N(s)$ and $D(s)$ for H(s) using the column common denominator method:

In[8]:=
```
N1[s] = ( s          3 (s + 2)
          3 (s - 3)  0.5 (s + 1) ) // Chop;
D1[s] = ( s^2 - 9  0
          0        (s + 1) (s + 2) ) // Chop;
N1[s].Inverse[D1[s]] // Simplify // Chop
```

Out[10]=

$$\left\{\left\{\frac{s}{-9+s^2}, \frac{3}{1+s}\right\}, \left\{\frac{3}{3+s}, \frac{0.5}{2.+s}\right\}\right\}$$

In[11]:=
```
H[s] == N1[s].Inverse[D1[s]] // Simplify
```

Out[11]=

True

## Get the minimal MFD :

In[12]:=
```
DN[s] = Join[D1[s], N1[s]];
{h[s], U[s]} = ExtendedHermiteForm[DN[s], s];
h[s] // MatrixForm
```

Out[14]//MatrixForm=

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

In[15]:=
```
Wg[s] = Take[h[s], 2];
N2[s] = N1[s].Inverse[Wg[s]] // Simplify // Chop // Rationalize;
N2[s] // TraditionalForm
```

Out[17]//TraditionalForm=

$$\begin{pmatrix} s & 3 s + 6 \\ 3 s - 9 & \frac{s}{2} + \frac{1}{2} \end{pmatrix}$$

In[18]:=
```
D2[s] = D1[s].Inverse[Wg[s]] // Simplify // Rationalize;
D2[s] // TraditionalForm
```

Out[19]//TraditionalForm=

$$\begin{pmatrix} s^2 - 9 & 0 \\ 0 & (s + 1)(s + 2) \end{pmatrix}$$

## Check the transfer function :

In[20]:=
```
H2[s] = N2[s].Inverse[D2[s]] // Simplify
H[s] == H2[s]
```

Out[20]=

$$\left\{\left\{\frac{s}{-9+s^2}, \frac{3}{1+s}\right\}, \left\{\frac{3}{3+s}, \frac{1}{4+2s}\right\}\right\}$$

Out[21]=

$$\left\{\left\{\frac{s}{-9+s^2}, \frac{3}{1+s}\right\}, \left\{\frac{3}{3+s}, \frac{0.5}{2+s}\right\}\right\} == \left\{\left\{\frac{s}{-9+s^2}, \frac{3}{1+s}\right\}, \left\{\frac{3}{3+s}, \frac{1}{4+2s}\right\}\right\}$$

Now the MFD realization will be minimal.

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

In[22]:=
```
D2[s] // Expand
```

Out[22]=

$$\{\{-9+s^2, 0\}, \{0, 2+3s+s^2\}\}$$

In[23]:=
```
n = Length[D2[s]];
k = Table[Max[Exponent[D2[s]ᵀ[[i]], s]], {i, 1, n}]
S[s] = DiagonalMatrix[s^k]
```

Out[24]=

{2, 2}

Out[25]=

$$\{\{s^2, 0\}, \{0, s^2\}\}$$

In[26]:=
```
Dhc = Coefficient[D2[s]ᵀ, s^k]ᵀ
```

Out[26]=

{{1, 0}, {0, 1}}

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

In[27]:=
```
p2 = Table[{Reverse[s^Range[0,k[[i]]-1]]}, {i, 1, n}]
```

Out[27]=

{{{s, 1}}, {{s, 1}}}

In[28]:=
```
ψ[s] = BlockDiagonalMatrix[p2]ᵀ
```

Out[28]=

```
Transpose[BlockDiagonalMatrix[{{{s, 1}}, {{s, 1}}}]]
```

# BlockDiagonalMatrix function does not work in my file (it says it is protected.). Thus I will write those myself.

In[29]:=

```
ψ[s] = (s 0
        1 0
        0 s
        0 1)
```

Out[29]=

```
{{s, 0}, {1, 0}, {0, s}, {0, 1}}
```

In[30]:=

```
L[s] = D2[s] - Dhc.S[s] // Expand
```

Out[30]=

```
{{-9, 0}, {0, 2 + 3 s}}
```

In[31]:=

```
d2 = Array[d## &, {n, Total[k]}];
sol1 = SolveAlways[Thread[Flatten[L[s]] == Flatten[d2.ψ[s]]], s] // Flatten;
Dlc = d2 /. sol1
```

Out[33]=

```
{{0, -9, 0, 0}, {0, 0, 3, 2}}
```

## Build up $A_c^0$:

In[34]:=

```
Ai = Table[DiagonalMatrix[ConstantArray[1, Max[k[[i]] - 1, 1]], -1, k[[i]]], {i, 1, n}]
Ac0 = (0 0 0 0
       1 0 0 0
       0 0 0 0
       0 0 1 0)
```

Out[34]=

```
{{{0, 0}, {1, 0}}, {{0, 0}, {1, 0}}}
```

Out[35]=

```
{{0, 0, 0, 0}, {1, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 1, 0}}
```

## Build up $B_c^0$:

In[36]:=

```
Bi = Table[{UnitVector[k[[i]], 1]}, {i, 1, n}]
Bc0 = (1 0
       0 0
       0 1
       0 0)
```

Out[36]=

```
{{{1, 0}}, {{1, 0}}}
```

Out[37]=

```
{{1, 0}, {0, 0}, {0, 1}, {0, 0}}
```

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

In[38]:=
```
Ac = Ac0 - Bc0.Inverse[Dhc].Dlc;
Ac // MatrixForm
```

Out[39]//MatrixForm=

$$\begin{pmatrix} 0 & 9 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -3 & -2 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

In[40]:=
```
Bc = Bc0.Inverse[Dhc];
Bc // MatrixForm
```

Out[41]//MatrixForm=

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

In[42]:=
```
nlc2 = Array[nlc## &, {Length[N2[s]], Total[k]}];
sol2 = SolveAlways[Thread[Flatten[N2[s]] == Flatten[nlc2.ψ[s]]], s] // Flatten;
Cc = nlc2 /. sol2
```

Out[44]=

$$\left\{\{1, 0, 3, 6\}, \left\{3, -9, \frac{1}{2}, \frac{1}{2}\right\}\right\}$$

## State Space Representation :

In[45]:=
```
x[t] = Array[x#[t] &, Total[k]]
y[t] = Array[y#[t] &, Length[N2[s]]]
u[t] = Array[u#[t] &, Length[Bc^T]]
```

Out[45]=

$\{x_1[t], x_2[t], x_3[t], x_4[t]\}$

Out[46]=

$\{y_1[t], y_2[t]\}$

Out[47]=

$\{u_1[t], u_2[t]\}$

In[48]:=
```
SEQ = Thread[D[x[t], t] == Ac.x[t] + Bc.u[t]]; ColumnForm[SEQ]
OEQ = Thread[y[t] == Cc.x[t]];
ColumnForm[OEQ]
```

Out[48]=

$x_1'[t] == u_1[t] + 9\, x_2[t]$
$x_2'[t] == x_1[t]$
$x_3'[t] == u_2[t] - 3\, x_3[t] - 2\, x_4[t]$
$x_4'[t] == x_3[t]$

Out[49]=

$y_1[t] == x_1[t] + 3\, x_3[t] + 6\, x_4[t]$
$y_2[t] == 3\, x_1[t] - 9\, x_2[t] + \frac{x_3[t]}{2} + \frac{x_4[t]}{2}$

- Check for controllability and observability

In[50]:=
```
P = Join[Bc, Ac.Bc, Ac.Ac.Bc, 2];
P // TraditionalForm
MatrixRank[P]
```

Out[51]//TraditionalForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -3 & 0 & 7 \\ 0 & 0 & 0 & 1 & 0 & -3 \end{pmatrix}$$

Out[52]=

4

- Yes! The system is controllable

In[53]:=
```
Q = Join[Cc, Cc.Ac, Cc.Ac.Ac];
Q // TraditionalForm
MatrixRank[Q]
```

Out[54]//TraditionalForm=

$$\begin{pmatrix} 1 & 0 & 3 & 6 \\ 3 & -9 & \frac{1}{2} & \frac{1}{2} \\ 0 & 9 & -3 & -6 \\ -9 & 27 & -1 & -1 \\ 9 & 0 & 3 & 6 \\ 27 & -81 & 2 & 2 \end{pmatrix}$$

Out[55]=

4

- Yes! The system is observable!

In[56]:=
```
A = Ac
```

Out[56]=

```
{{0, 9, 0, 0}, {1, 0, 0, 0}, {0, 0, -3, -2}, {0, 0, 1, 0}}
```

In[57]:=
```
b = Bc
```
Out[57]=
```
{{1, 0}, {0, 0}, {0, 1}, {0, 0}}
```

In[58]:=
```
c = Cc
```
Out[58]=

$$\left\{\{1, 0, 3, 6\}, \left\{3, -9, \frac{1}{2}, \frac{1}{2}\right\}\right\}$$

# ii)

In[59]:=
```
Eigenvalues[A]
```
Out[59]=
```
{-3, 3, -2, -1}
```

One of the eigenvalues is not stable.

In[ ]:=
```
I4 = IdentityMatrix[4];
```

In[ ]:=
```
α[s] = Det[s I4 - (A - b.K)]
```

In[60]:=

$$-18 - 27 s - 7 s^2 + 3 s^3 + s^4 + 2 \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K + 16 s \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K + 18 s^2 \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K + 4 s^3 \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K$$

Out[60]=

$$-18 - 27 s - 7 s^2 + 3 s^3 + s^4 + 2 \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K + 16 s \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K + 18 s^2 \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K + 4 s^3 \{\{1, 0\}, \{0, 0\}, \{0, 1\}, \{0, 0\}\}.K$$

In[61]:=

$$\alpha[s] = (s - (-5))^2 (s - (-4))^2 \text{ // Expand}$$

Out[61]=

$$400 + 360 s + 121 s^2 + 18 s^3 + s^4$$

In[62]:=

$$hf = \text{HornerForm}\left[\alpha[s] - s^{\text{Exponent}[\alpha[s],s]}, s\right]$$

Out[62]=

$$400 + s (360 + s (121 + 18 s))$$

In[63]:=
```
hf[[1]]
```
Out[63]=
```
400
```

In[64]:=
```
hf[[2]]
```
Out[64]=
```
s (360 + s (121 + 18 s))
```

In[65]:=
```
α1[s] = hf[[2]] / s; α2[s] = hf[[1]];
```

In[66]:=
```
d3 = Array[d_## &, {n, Total[k]}]
sol2 =
  SolveAlways[Thread[Flatten[( α1[s] α2[s] ; -1 0 )] == Flatten[d3.ψ[s]]], s] // Flatten;
D3 = d3 /. sol2
```

Out[66]=

$\{\{d_{1,1}, d_{1,2}, d_{1,3}, d_{1,4}\}, \{d_{2,1}, d_{2,2}, d_{2,3}, d_{2,4}\}\}$

Out[68]=

$\{\{d_{1,1}, d_{1,2}, d_{1,3}, d_{1,4}\}, \{d_{2,1}, d_{2,2}, d_{2,3}, d_{2,4}\}\}$

In[74]:=
```
hf[[3]]
```

••• Part : Part 3 of 400 + s (360 + s (121 + 18 s)) does not exist.

Out[74]=

(400 + s (360 + s (121 + 18 s)))[[3]]

In[◦]:=
```
n
```

Out[◦]=

2

In[70]:=
```
k
```

Out[70]=

{2, 2}

In[71]:=
```
K = Dhc.D3 - Dlc
```

Out[71]=

$\{\{d_{1,1}, 9 + d_{1,2}, d_{1,3}, d_{1,4}\}, \{d_{2,1}, d_{2,2}, -3 + d_{2,3}, -2 + d_{2,4}\}\}$

It did not work somehow. I’ll try to type it myself to save time.

In[72]:=
```
α1 = 18 s + 121
```

Out[72]=

121 + 18 s

In[73]:=
```
α2 = 360 s + 400
```

Out[73]=

400 + 360 s

In[76]:=
```
αM = ( α1 α2 ; -1 0 )
```

Out[76]=

{{121 + 18 s, 400 + 360 s}, {-1, 0}}

In[77]:=
```
ψ[s]
```

Out[77]=

{{s, 0}, {1, 0}, {0, s}, {0, 1}}

In[81]:=

```
αM2 = (18 121 360 400
       0  -1   0   0)
```

Out[81]=

```
{{18, 121, 360, 400}, {0, -1, 0, 0}}
```

In[83]:=

```
αM == αM2 . ψ[s]
```

Out[83]=

```
True
```

In[86]:=

```
K = Dhc . αM2 - Dlc;
K // MatrixForm
```

Out[87]//MatrixForm=

$$\begin{pmatrix} 18 & 130 & 360 & 400 \\ 0 & -1 & -3 & -2 \end{pmatrix}$$

In[112]:=

```
Eigenvalues[A - b.K]
```

Out[112]=

```
{-5, -5, -4, -4}
```

**Eigenvalues are placed correctly!**

In[141]:=

```
A - b.K // MatrixForm
```

Out[141]//MatrixForm=

$$\begin{pmatrix} -18 & -121 & -360 & -400 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

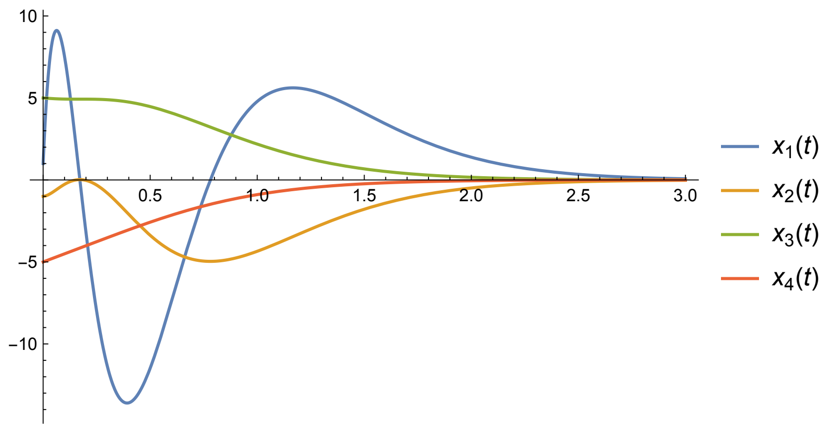
In[381]:=

```
Clear[v];
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]
IC = Thread[x[0] == {1, -1, 5 , -5}];
tmax = 3;
CLSol = NDSolve[{ClosedLoopEq, IC} // Flatten, x[t], {t, 0, tmax}];
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)"}]
```

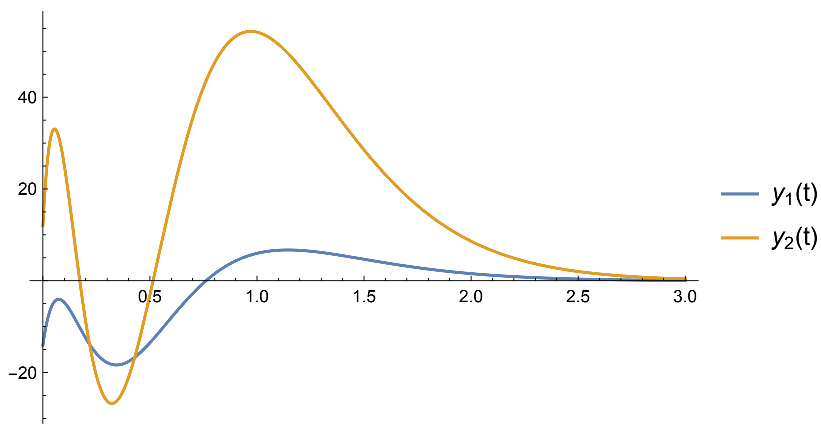
Out[386]=

$$x_1'[t] == -18\,x_1[t] - 121\,x_2[t] - 360\,x_3[t] - 400\,x_4[t]$$
$$x_2'[t] == x_1[t]$$
$$x_3'[t] == x_2[t]$$
$$x_4'[t] == x_3[t]$$

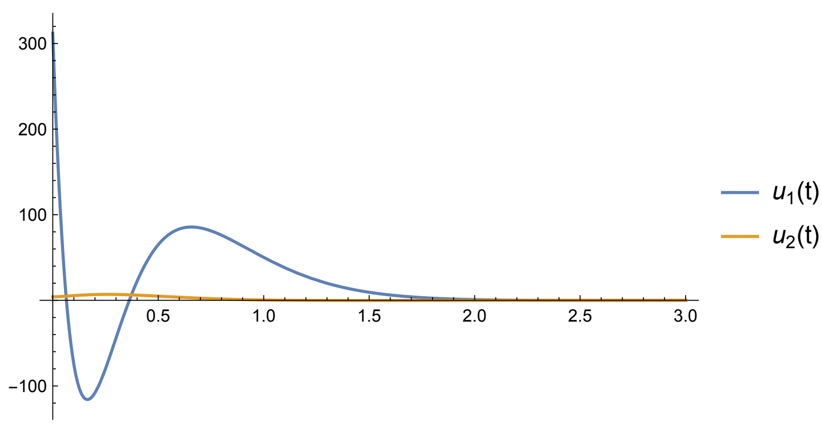
Out[390]=




Out[391]=




Out[392]=



# iii)

In[393]:=

$$Q = \begin{pmatrix} 40 & 40 & 0 & 0 \\ 40 & 100 & 0 & 0 \\ 0 & 0 & 160 & 32 \\ 0 & 0 & 32 & 200 \end{pmatrix}; \ Sf = \begin{pmatrix} 1000 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 \\ 0 & 0 & 800 & 0 \\ 0 & 0 & 0 & 1000 \end{pmatrix}; \ R = \begin{pmatrix} 4 & 0 \\ 0 & 10 \end{pmatrix};$$

In[427]:=

```
B = b; C1 = c;
n = Length[A];
```

In[429]:=

```
Sr = RiccatiSolve[{A, B}, {Q, R}] // N // Chop;
Sr // MatrixForm
```

Out[430]//MatrixForm=

$$\begin{pmatrix} 27.883 & 77.1825 & 0 & 0 \\ 77.1825 & 247.073 & 0 & 0 \\ 0 & 0 & 25.4959 & 28.9898 \\ 0 & 0 & 28.9898 & 179.873 \end{pmatrix}$$

In[431]:=

```
PositiveDefiniteMatrixQ[Sr]
```

Out[431]=

True

In[432]:=

```
Eigenvalues[Sr] // N
```

Out[432]=

{271.523, 185.138, 20.2316, 3.43242}

In[433]:=

```
Ko = (Inverse[R].Bᵀ.Sr) // FullSimplify;
Ko // MatrixForm
```

Out[434]//MatrixForm=

$$\begin{pmatrix} 6.97074 & 19.2956 & 0. & 0. \\ 0. & 0. & 2.54959 & 2.89898 \end{pmatrix}$$

In[435]:=

```
Eigenvalues[A - B.Ko] // Simplify // N
```

Out[435]=

{-4.84632, -4.44827, -2.12442, -1.10132}

In[436]:=

```
Clear[v];
x[t_] := Array[x#[t] &, Length[A]]
y[t_] := c.x[t];
v[t_] := {0, 0}
u[t_] = v[t] - Ko.x[t];
ClosedLoopEq = Thread[D[x[t], t] == (A - b.Ko).x[t] + b . v[t] // Chop];
ColumnForm[ClosedLoopEq]
IC = Thread[x[0] == {1, -1, 5 , -5}];
tmax = 3;
CLSol = NDSolve[{ClosedLoopEq, IC} // Flatten, x[t], {t, 0, tmax}];
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[441]=

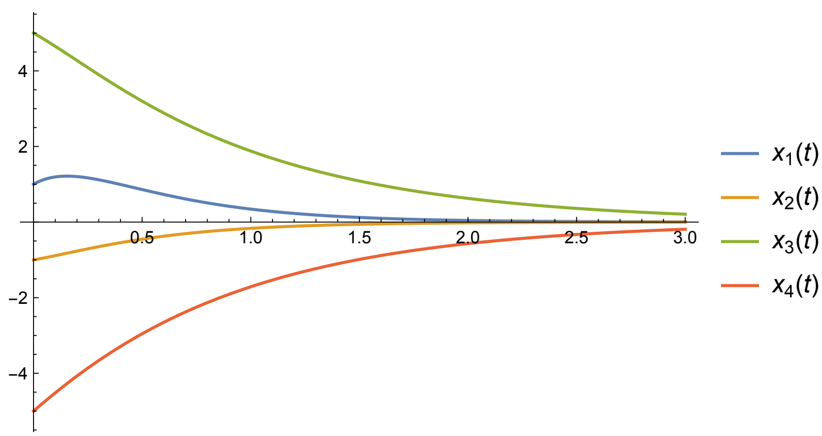
$x_1'[t] == -6.97074\, x_1[t] - 10.2956\, x_2[t]$

$x_2'[t] == 1.\, x_1[t]$

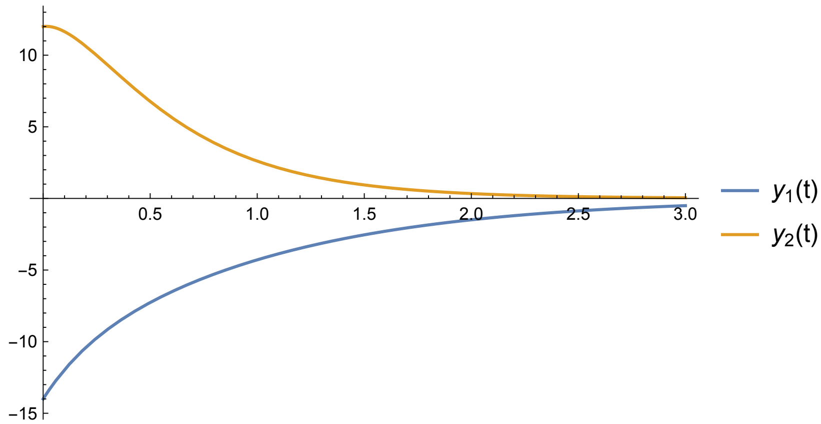
$x_3'[t] == -5.54959\, x_3[t] - 4.89898\, x_4[t]$

$x_4'[t] == 1.\, x_3[t]$

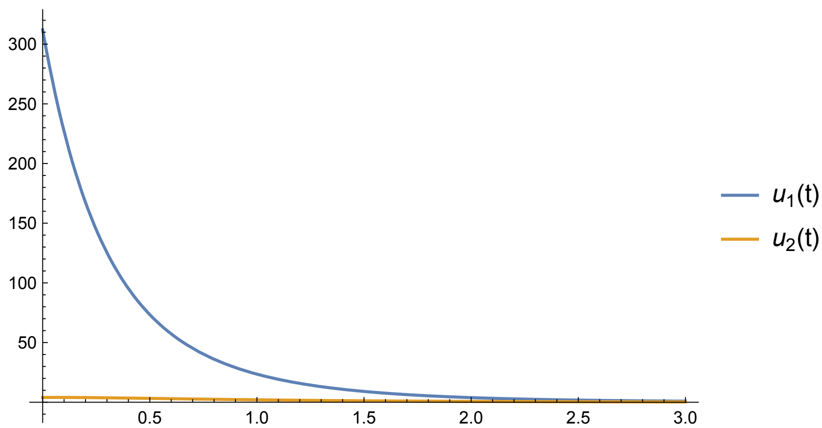
Out[445]=



Out[446]=




Out[447]=




Way better solution than part ii! Transient response is improved!

# iv)

This time K will change with time. (tf --/-->∞)

In[653]:=

```
A
```

Out[653]=

```
{{0, 9, 0, 0}, {1, 0, 0, 0}, {0, 0, -3, -2}, {0, 0, 1, 0}}
```

In[654]:=

```
C1 = C1 // N
```

Out[654]=

```
{{1., 0., 3., 6.}, {3., -9., 0.5, 0.5}}
```

In[655]:=

```
Clear[S]
```

In[656]:=

```
S[t_] := Array[
  Subscript[S, Sequence @@ Through[{Min, Max}[##]]][t] &, {Length[A], Length[A]}]
S[t]
(*LHRE=D[S[t],t]+S[t].A-S[t].B.Inverse[R].Bᵀ.S[t]+C1ᵀ.Q.C1+Aᵀ.S[t] *)
```

Out[657]=

$\{\{S_{1,1}[t], S_{1,2}[t], S_{1,3}[t], S_{1,4}[t]\}, \{S_{1,2}[t], S_{2,2}[t], S_{2,3}[t], S_{2,4}[t]\}, \{S_{1,3}[t], S_{2,3}[t], S_{3,3}[t], S_{3,4}[t]\}, \{S_{1,4}[t], S_{2,4}[t], S_{3,4}[t], S_{4,4}[t]\}\}$

In[658]:=

```
LHRE = D[S[t], t] + S[t].A - S[t].B.Inverse[R].Bᵀ.S[t] + Q + Aᵀ.S[t]
```

Out[658]=

$$\left(\begin{array}{cc} S_{1,1}'(t) - \frac{1}{4} S_{1,1}(t)^2 - \frac{1}{10} S_{1,3}(t)^2 + 2 S_{1,2}(t) + 40 & S_{1,2}'(t) - \frac{1}{4} S_{1,2}(t) S_{1,1}(t) + 9 S_{1,1}(t) + S_{2,2}(t) - \frac{1}{10} S_{1,3}( \\ S_{1,2}'(t) - \frac{1}{4} S_{1,2}(t) S_{1,1}(t) + 9 S_{1,1}(t) + S_{2,2}(t) - \frac{1}{10} S_{1,3}(t) S_{2,3}(t) + 40 & S_{2,2}'(t) - \frac{1}{4} S_{1,2}(t)^2 + 18 S_{1,2}(t) - \frac{1}{10} S_{2,3}(t)^2 + \\ S_{1,3}'(t) - \frac{1}{4} S_{1,1}(t) S_{1,3}(t) - \frac{1}{10} S_{3,3}(t) S_{1,3}(t) - 3 S_{1,3}(t) + S_{1,4}(t) + S_{2,3}(t) & S_{2,3}'(t) - \frac{1}{4} S_{1,2}(t) S_{1,3}(t) + 9 S_{1,3}(t) - 3 S_{2,3}(t) + S_{2,4}(t) - \\ S_{1,4}'(t) - \frac{1}{10} S_{3,4}(t) S_{1,3}(t) - 2 S_{1,3}(t) - \frac{1}{4} S_{1,1}(t) S_{1,4}(t) + S_{2,4}(t) & S_{2,4}'(t) - \frac{1}{4} S_{1,2}(t) S_{1,4}(t) + 9 S_{1,4}(t) - 2 S_{2,3}(t) - \frac{1}{10} S \end{array}\right.$$

In[659]:=

```
UpperElements[M_] := Flatten[Table[M〚i, j〛, {i, Length[M]}, {j, i, Length[M]}]]
LHREnD = UpperElements[LHRE];
Ov = ConstantArray[0, Length[LHREnD]];
```

In[662]:=

```
RE = Thread[LHREnD == 0v]
Sv[t_] := UpperElements[S[t]]
```

Out[662]=

$$\left\{40 - \frac{1}{4} S_{1,1}[t]^2 + 2 S_{1,2}[t] - \frac{1}{10} S_{1,3}[t]^2 + S_{1,1}'[t] == 0,\right.$$

$$40 + 9 S_{1,1}[t] - \frac{1}{4} S_{1,1}[t] S_{1,2}[t] + S_{2,2}[t] - \frac{1}{10} S_{1,3}[t] S_{2,3}[t] + S_{1,2}'[t] == 0,$$

$$-3 S_{1,3}[t] - \frac{1}{4} S_{1,1}[t] S_{1,3}[t] + S_{1,4}[t] + S_{2,3}[t] - \frac{1}{10} S_{1,3}[t] S_{3,3}[t] + S_{1,3}'[t] == 0,$$

$$-2 S_{1,3}[t] - \frac{1}{4} S_{1,1}[t] S_{1,4}[t] + S_{2,4}[t] - \frac{1}{10} S_{1,3}[t] S_{3,4}[t] + S_{1,4}'[t] == 0,$$

$$100 + 18 S_{1,2}[t] - \frac{1}{4} S_{1,2}[t]^2 - \frac{1}{10} S_{2,3}[t]^2 + S_{2,2}'[t] == 0,$$

$$9 S_{1,3}[t] - \frac{1}{4} S_{1,2}[t] S_{1,3}[t] - 3 S_{2,3}[t] + S_{2,4}[t] - \frac{1}{10} S_{2,3}[t] S_{3,3}[t] + S_{2,3}'[t] == 0,$$

$$9 S_{1,4}[t] - \frac{1}{4} S_{1,2}[t] S_{1,4}[t] - 2 S_{2,3}[t] - \frac{1}{10} S_{2,3}[t] S_{3,4}[t] + S_{2,4}'[t] == 0,$$

$$160 - \frac{1}{4} S_{1,3}[t]^2 - 6 S_{3,3}[t] - \frac{1}{10} S_{3,3}[t]^2 + 2 S_{3,4}[t] + S_{3,3}'[t] == 0,$$

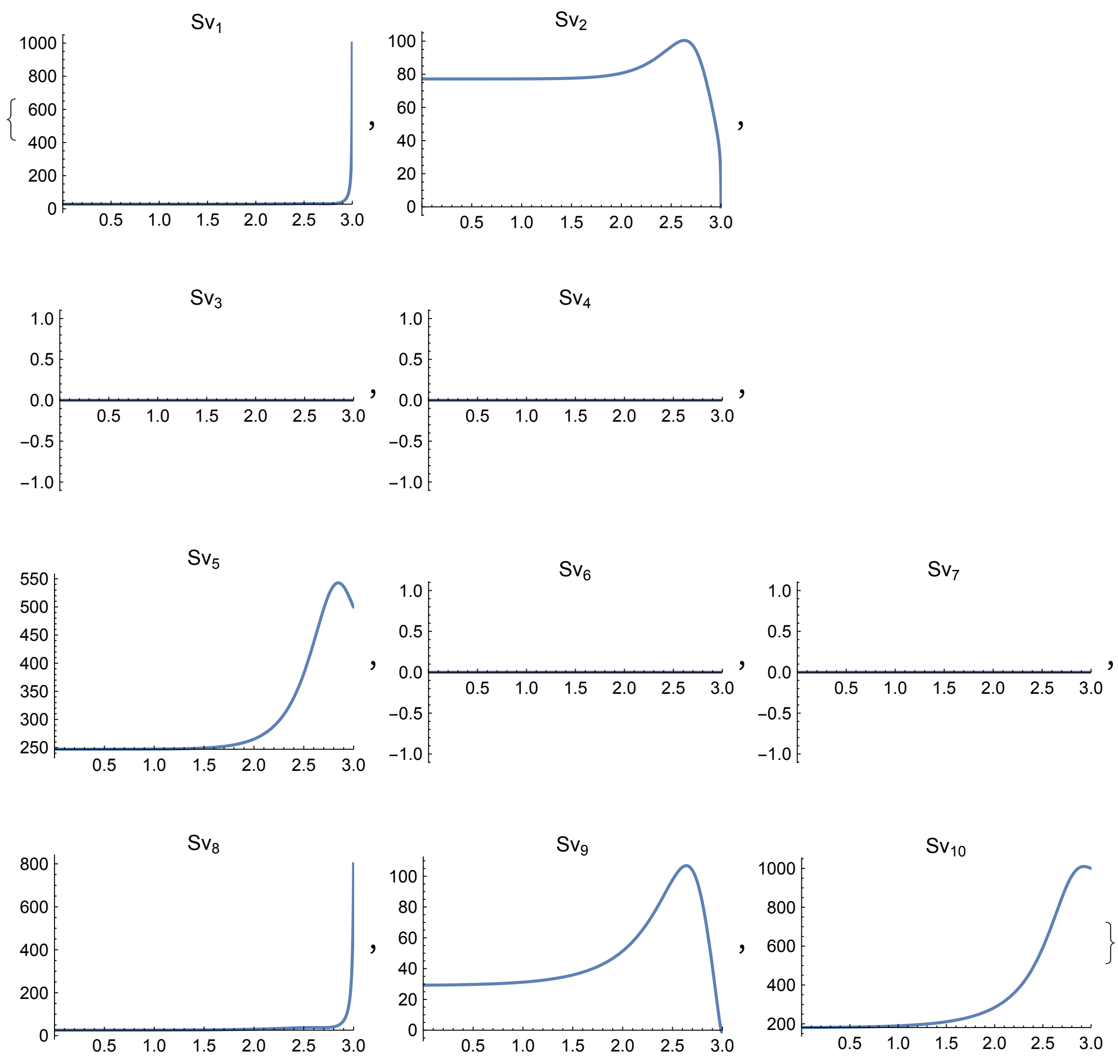
$$32 - \frac{1}{4} S_{1,3}[t] S_{1,4}[t] - 2 S_{3,3}[t] - 3 S_{3,4}[t] - \frac{1}{10} S_{3,3}[t] S_{3,4}[t] + S_{4,4}[t] + S_{3,4}'[t] == 0,$$

$$\left. 200 - \frac{1}{4} S_{1,4}[t]^2 - 4 S_{3,4}[t] - \frac{1}{10} S_{3,4}[t]^2 + S_{4,4}'[t] == 0\right\}$$

In[705]:=

```
tf = 3;
SolS = NDSolve[{RE, Thread[Sv[tf] == UpperElements[Sf]]} // Flatten,
   Sv[t], {t, 0, tf}, Method → "StiffnessSwitching"];
Table[Plot[Evaluate[Sv[t]⟦i⟧ /. SolS], {t, 0, tf},
  PlotRange → {{0, tf}, All}, PlotLabel → "Sv"_i], {i, 1, Length[Sv[t]]}]
```

Out[707]=



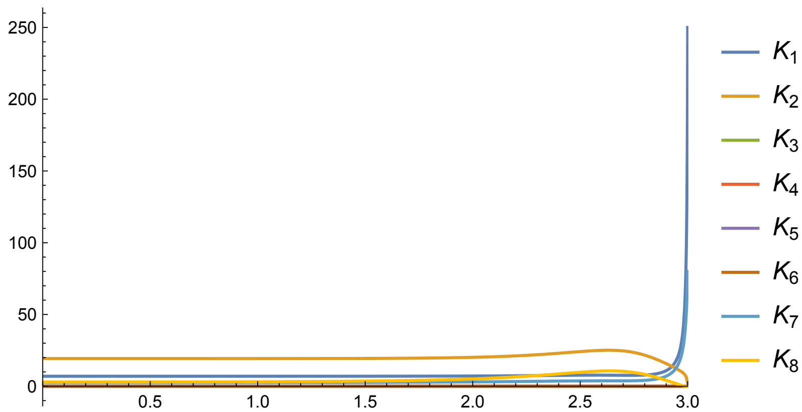
In[718]:=

```
Clear[Ko]
Ko[t_] := (Inverse[R].Bᵀ.S[t])
Ko[t]
Plot[Evaluate[Ko[t] /. SolS], {t, 0, tf}, PlotRange → {{0, tf}, All},
 PlotLegends → {"K1", "K2", "K3", "K4", "K5", "K6", "K7", "K8" }]
```

Out[720]=

$$\left\{\left\{\frac{1}{4} S_{1,1}[t], \frac{1}{4} S_{1,2}[t], \frac{1}{4} S_{1,3}[t], \frac{1}{4} S_{1,4}[t]\right\}, \left\{\frac{1}{10} S_{1,3}[t], \frac{1}{10} S_{2,3}[t], \frac{1}{10} S_{3,3}[t], \frac{1}{10} S_{3,4}[t]\right\}\right\}$$

Out[721]=




2 of the values in K matrix spikes at the last second. This will cause a diversion from optimality

In[752]:=

```
uop2[t_] := -Ko[t].x[t]
yop[t_] := C1.x[t];
```

In[739]:=

```
StEqn2 = Thread[D[x[t], t] == A.x[t] + B.uop2[t]] // Chop;
ColumnForm[StEqn2]
IC = Thread[x[0] == {1, -1, 5, -5}] // Flatten
```

Out[740]=

$$x_1'[t] == 9 x_2[t] - \frac{1}{4} x_1[t] S_{1,1}[t] - \frac{1}{4} x_2[t] S_{1,2}[t] - \frac{1}{4} x_3[t] S_{1,3}[t] - \frac{1}{4} x_4[t] S_{1,4}[t]$$
$$x_2'[t] == x_1[t]$$
$$x_3'[t] == -3 x_3[t] - 2 x_4[t] - \frac{1}{10} x_1[t] S_{1,3}[t] - \frac{1}{10} x_2[t] S_{2,3}[t] - \frac{1}{10} x_3[t] S_{3,3}[t] - \frac{1}{10} x_4[t] S_{3,4}[$$
$$x_4'[t] == x_3[t]$$

Out[741]=

$$\{x_1[0] == 1, x_2[0] == -1, x_3[0] == 5, x_4[0] == -5\}$$

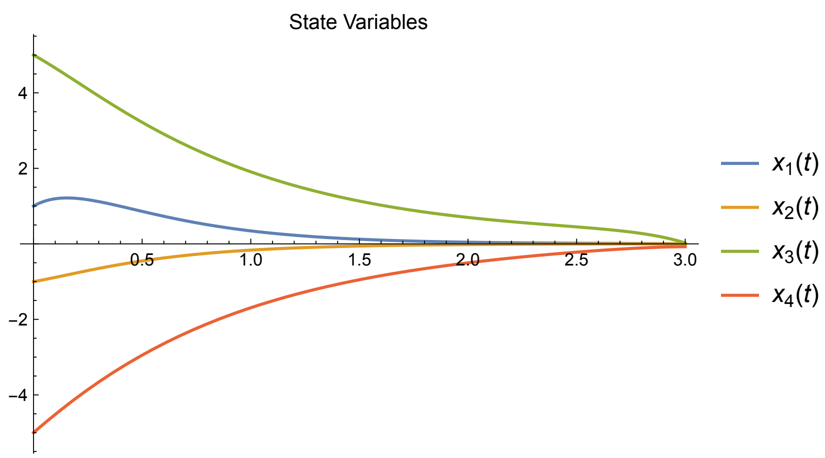
In[742]:=

```
SolStEqn2 = NDSolve[{StEqn2, IC} /. SolS, x[t], {t, 0, tf}] // Flatten;
```

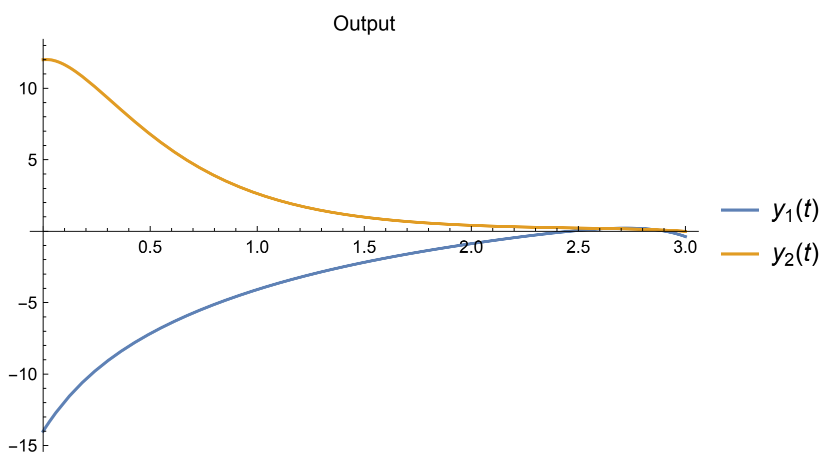
In[754]:=

```
Plot[Evaluate[x[t] /. SolStEqn2], {t, 0, tf},
 PlotRange → All, PlotLegends → x[t], PlotLabel → "State Variables"]
Plot[Evaluate[yop[t] /. SolStEqn2 /. SolS], {t, 0, tf},
 PlotRange → All, PlotLegends → y[t], PlotLabel → "Output"]
Plot[Evaluate[uop2[t] /. SolStEqn2 /. SolS], {t, 0, tf},
 PlotRange → All, PlotLegends → u[t], PlotLabel → "Input"]
```

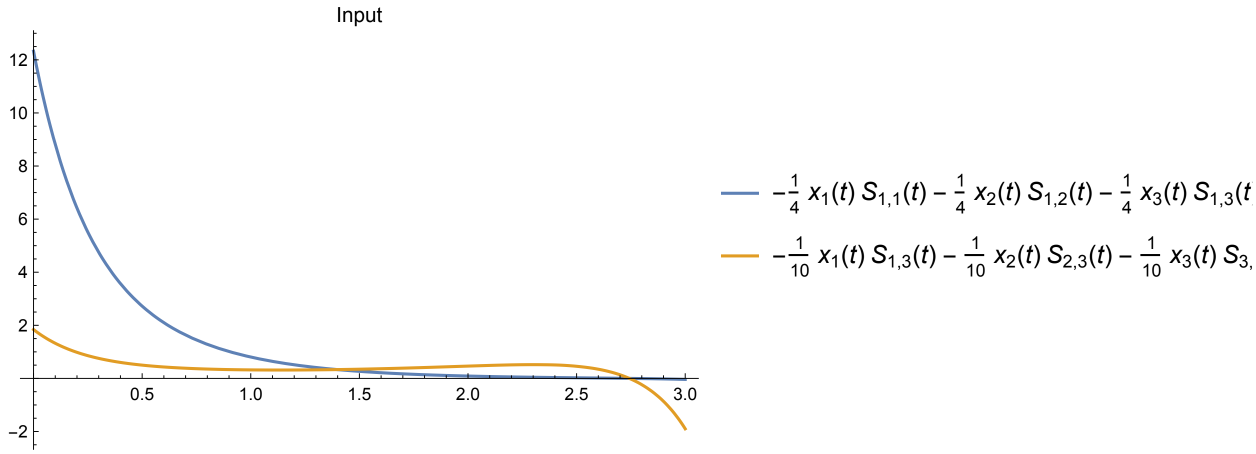
Out[754]=




Out[755]=



Out[756]=




# LQR in the part **iii)** the best solution in terms of optimality. This one loses optimality while it is getting closer to final time tf!

# v)

In[818]:=

```
Om = Join[C1ᵀ, Aᵀ.C1ᵀ, (Aᵀ.Aᵀ).C1ᵀ, (Aᵀ.Aᵀ.Aᵀ).C1ᵀ, 2]ᵀ;
Om // MatrixForm
MatrixRank[Om]
```

Out[819]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 3. & 6. \\ 3. & -9. & 0.5 & 0.5 \\ 0. & 9. & -3. & -6. \\ -9. & 27. & -1. & -1. \\ 9. & 0. & 3. & 6. \\ 27. & -81. & 2. & 2. \\ 0. & 81. & -3. & -6. \\ -81. & 243. & -4. & -4. \end{pmatrix}$$

Out[820]=

4

**I forgot that this method only works for SISO systems (need inverse[Om]). I just left it here. I attempted this way and failed.**

In[821]:=

```
Inverse[Om]
```

Inverse : Argument

{{1., 0., 3., 6. }, {3., −9., 0.5, 0.5 }, {0., 9., −3., −6.}, {−9., 27., −1., −1.}, {9., 0., 3., 6. }, {27., −81., 2., 2. }, {0., 81., −3⋱., −6.}, {−81., 243., −4., −4.}} at position 1 is not a non −empty square matrix.

Out[821]=

```
Inverse[
 {{1., 0., 3., 6.}, {3., -9., 0.5, 0.5}, {0., 9., -3., -6.}, {-9., 27., -1., -1.},
  {9., 0., 3., 6.}, {27., -81., 2., 2.}, {0., 81., -3., -6.}, {-81., 243., -4., -4.}}]
```

Observable!

In[803]:=

```
I4 = IdentityMatrix[4];
a[s] = Det[s I4 - A] // Expand
```

Out[804]=

$-18 - 27 s - 7 s^2 + 3 s^3 + s^4$

In[810]:=

$\alpha o[s] = (s + 16)^2 (s + 20)^2$ // Expand // N

Out[810]=

$102\,400. + 23\,040.\, s + 1936.\, s^2 + 72.\, s^3 + s^4$

In[811]:=

$$Ut = \begin{pmatrix} 1 & 3 & -7 & -27 \\ 0 & 1 & 3 & -7 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

In[812]:=

```
αoc = Reverse[Drop[CoefficientList[αo[s], s], -1]] // N // Chop
```

Out[812]=

```
{72., 1936., 23040., 102400.}
```

In[813]:=

```
ac = Reverse[Drop[CoefficientList[a[s], s], -1]]
```

Out[813]=

```
{3, -7, -27, -18}
```

In[814]:=

$Lt = \{(\alpha oc - ac).Inverse[Ut].Inverse[Om^{T}]\};$

In[793]:=

$Ut.Om^{T}$ // MatrixForm

Out[793]//MatrixForm=

$$\begin{pmatrix} -182. & -41. & 210. & 106. & -174. & -284. \\ -33. & -11. & 42. & 31. & -33. & -89. \\ 21. & 2. & -21. & -4. & 21. & 8. \\ 6. & 0.5 & -6. & -1. & 6. & 2. \end{pmatrix}$$

I did coefficient matching but I struggled with the mathematica syntax. I used python to do the coefficient matching by giving poles

-16,-16, -20, -20 and found the L.

In[839]:=

```
L1 = ( 1009.9678571428571428571428571429 1009.9678571428571428571428571429
       724.28293650793650793650793650794 724.28293650793650793650793650794
       54187.575                         54187.575
       -28785.975                        -28785.975 );
```

In[843]:=

```
Eigenvalues[A - L1.C1]
```

Out[843]=

```
{-20. + 0.00371062 i, -20. - 0.00371062 i, -16. + 0.00299783 i, -16. - 0.00299783 i}
```

Given L matrix placed desired position. There is a small imaginary value and they are extremely high numbers. I remember we use some matrices in the LSA class but I couldn’t get the class files. I wanted to check with the MATLAB place() function to see what is the answer there. Found as follow:

In[850]:=

```
L = ( 109.25   -73.25
      36.4167  0.1389
      23.75    984.25
      -23.75   -480.25 );

K // MatrixForm (* K from pole placement (part 2) *)
```

Out[824]//MatrixForm=

```
( 18  130  360  400
  0   -1   -3   -2 )
```

In[826]:=

```
Ac = A - L.C1;
```

# Eqs trial without noise

In[851]:=

```
xo[t_] := {xo1[t], xo2[t], xo3[t], xo4[t]}; u[t_] = {u1[t], u2[t]};
EqObserver = Thread[xo'[t] == Ac.xo[t] + L.y[t] + B.u[t]] // Chop // Flatten;
TableForm[EqObserver]
```

Out[853]//TableForm=

```
xo1′[t] == u1[t] + 110.5 xo1[t] - 650.25 xo2[t] - 291.125 xo3[t] - 618.875 xo4[t] + 109.25 y1[t
xo2′[t] == -35.8334 xo1[t] + 1.2501 xo2[t] - 109.32 xo3[t] - 218.57 xo4[t] + 36.4167 y1[t] + 0.
xo3′[t] == u2[t] - 2976.5 xo1[t] + 8858.25 xo2[t] - 566.375 xo3[t] - 636.625 xo4[t] + 23.75 y1[
xo4′[t] == 1464.5 xo1[t] - 4322.25 xo2[t] + 312.375 xo3[t] + 382.625 xo4[t] - 23.75 y1[t] - 48
```

In[856]:=

```
Length[x[t]]
```

Out[856]=

4

In[857]:=

```
Length[v[t]]
```

Out[857]=

2

In[858]:=

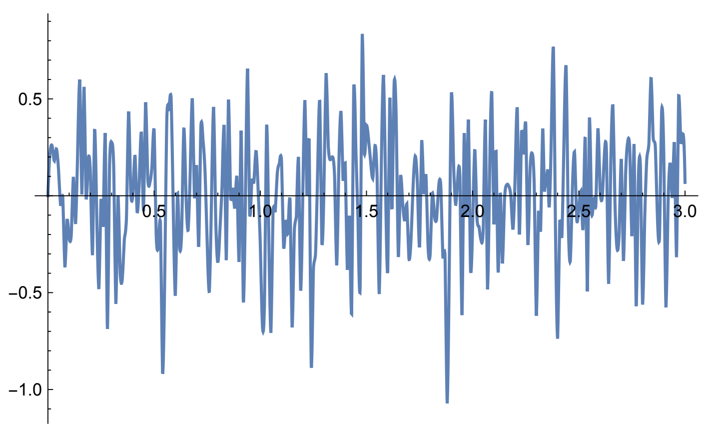
```
W = 0.1 IdentityMatrix[Length[x[t]]];
Θ = 0.01 IdentityMatrix[Length[v[t]]];
```

## Plant Noise:

In[864]:=

```
tmax = tf; step = 0.01; σW = √W[[1, 1]] ;
g = Interpolation[Thread[{Range[0, tmax, step],
      Join[{0}, RandomReal[NormalDistribution[0, σW], tmax / step]]}], t];
Plot[g, {t, 0, tmax}]
Print["Mean: ", Mean[RandomReal[NormalDistribution[0, σW], tmax / step]],
 " \nVariance: ", Variance[RandomReal[NormalDistribution[0, σW], tmax / step]]]
```

Out[866]=



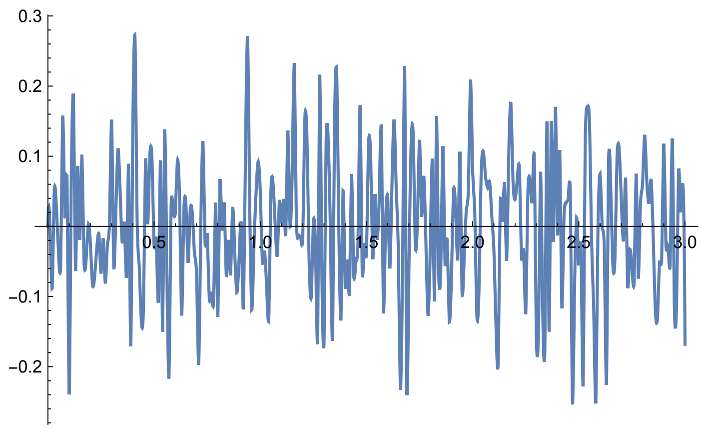

```
Mean: -0.0129999
Variance: 0.0799231
```

## Sensor Noise

In[868]:=

```
σθ = √(Θ[[1, 1]]);
θ = Interpolation[Thread[{Range[0, tmax, step],
      Join[{0}, RandomReal[NormalDistribution[0, σθ], tmax / step]]}], t];
Plot[θ, {t, 0, tmax}]
Print["Mean: ", Mean[RandomReal[NormalDistribution[0, σθ], tmax / step]],
 " \nVariance: ", Variance[RandomReal[NormalDistribution[0, σθ], tmax / step]]]
```

Out[870]=



```
Mean: 0.00685842
Variance: 0.00974751
```

In[872]:=

```
Clear[u]
```

In[1071]:=

```
xov[t_] := Array[xo#[t] &, Length[A]];
uoc[t_] := -K.xov[t]
yhat[t_] := C1.xov[t]
ynoise[t_] := C1.x[t] + ConstantArray[θ, Length[y[t]]]

EqObserverOp =
  Thread[xov'[t] == A.xov[t] + B.uoc[t] + L.(ynoise[t] - yhat[t])] // Chop // Flatten;
EqObsControllerOp = Thread[
   x'[t] == (A - B.K).x[t] + B.K.(x[t] - xov[t]) + ConstantArray[g, Length[x[t]]]];
AllEqnOp = {EqObserverOp, EqObsControllerOp} // Chop // Flatten // Simplify;
ColumnForm[AllEqnOp];
IC = Thread[x[0] == {1, -1, 5, -5}] // Flatten;
ICo = Thread[xov[0] == {0, 0, 0, 0}] // Flatten;
```

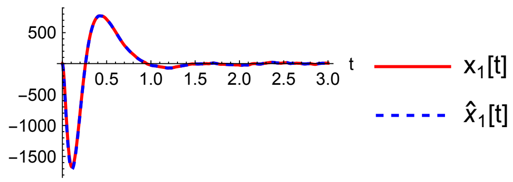
In[884]:=

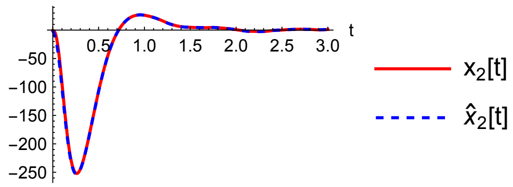
```
SolObsControllerOp = NDSolve[{AllEqnOp, IC, ICo} // Flatten,
    {x[t], xov[t]} // Flatten, {t, 0, tf}] // Flatten;
```

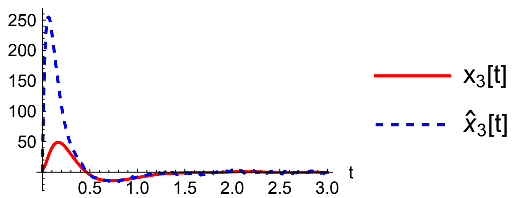
In[887]:=

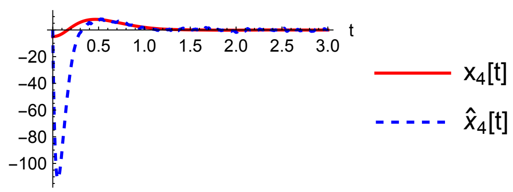
```
Table[Plot[Evaluate[{x[t][[i]], xov[t][[i]]} /. SolObsControllerOp], {t, 0, tf},
   PlotStyle → {{Red}, {Dashed, Blue}}, PlotRange → All, AxesLabel → {"t"},
   PlotLegends → {ToString[Subscript["x", i], StandardForm] <> "[t]",
     ToString[Subscript["x̂", i], StandardForm] <> "[t]"}],
  {i, 1, Length[x[t]]}] // TableForm
```

Out[887]//TableForm=




Error is very large at the beginning for states 3 and 4.

# vi)

## Solve the Filter Algebraic Riccati Equation :

The steady state error covariance matrix Σ:

In[1009]:=

```
W = 0.1 IdentityMatrix[Length[x[t]]];
Θ = 0.01 IdentityMatrix[Length[v[t]]];
Σ = RiccatiSolve[{Aᵀ, C1ᵀ}, {W, Θ}];
Print["Σ = ", Σ // MatrixForm]
```

$$\Sigma = \begin{pmatrix} 2.20641 & 0.721383 & 0.0526951 & -0.333838 \\ 0.721383 & 0.239227 & 0.0177511 & -0.109225 \\ 0.0526951 & 0.0177511 & 0.0220575 & -0.0167187 \\ -0.333838 & -0.109225 & -0.0167187 & 0.0597287 \end{pmatrix}$$

In[1013]:=

```
Lop = Σ .C1ᵀ.Inverse[Θ];
Print["Lopt = ", Lop // MatrixForm]
```

$$L_{opt} = \begin{pmatrix} 36.1468 & -1.37801 \\ 11.9284 & -3.46262 \\ 1.85555 & 0.0994942 \\ -2.56226 & 0.301816 \end{pmatrix}$$

In[1015]:=

```
P = RiccatiSolve[{A, B}, {Q, R}] // N;
Print["P = ", P // MatrixForm]
```

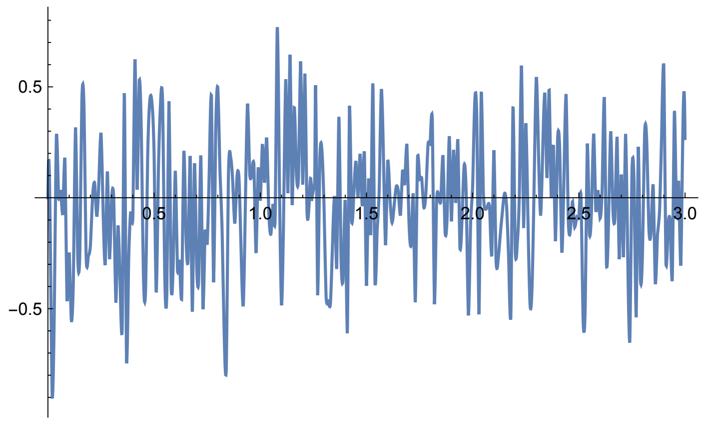
$$P = \begin{pmatrix} 27.883 & 77.1825 & 0. & 0. \\ 77.1825 & 247.073 & 0. & 0. \\ 0. & 0. & 25.4959 & 28.9898 \\ 0. & 0. & 28.9898 & 179.873 \end{pmatrix}$$

## Plant Noise:

In[1018]:=

```
tmax = tf; step = 0.01; σW = √(W[[1, 1]]);
g = Interpolation[Thread[{Range[0, tmax, step],
      Join[{0}, RandomReal[NormalDistribution[0, σW], tf / step]]}], t];
Plot[g, {t, 0, tf}]
Print["Mean: ", Mean[RandomReal[NormalDistribution[0, σW], tf / step]],
 " \nVariance: ", Variance[RandomReal[NormalDistribution[0, σW], tf / step]]]
```

Out[1020]=



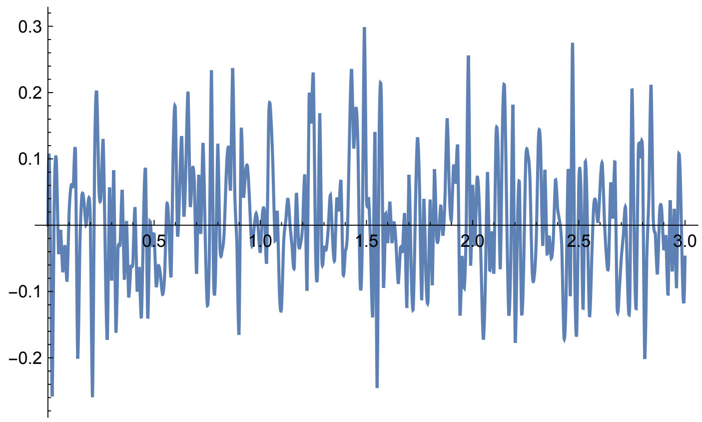

```
Mean: -0.0148233
Variance: 0.112052
```

## Sensor Noise

In[1022]:=

```
σθ = √Θ[[1, 1]];
θ = Interpolation[Thread[{Range[0, tmax, step],
     Join[{0}, RandomReal[NormalDistribution[0, σθ], tf / step]]}], t];
Plot[θ, {t, 0, tf}]
Print["Mean: ", Mean[RandomReal[NormalDistribution[0, σθ], tf / step]],
 " \nVariance: ", Variance[RandomReal[NormalDistribution[0, σθ], tf / step]]]
```

Out[1024]=




```
Mean: 0.00550301
Variance: 0.0100085
```

### Combined Observer - Controller Equations

## Build State Equation :

In[1026]:=

```
K
```

Out[1026]=

```
{{18, 130, 360, 400}, {0, -1, -3, -2}}
```

In[1027]:=

```
Ko = (Inverse[R].Bᵀ.Sr) // FullSimplify;
Ko // MatrixForm
```

Out[1028]//MatrixForm=

$$\begin{pmatrix} 6.97074 & 19.2956 & 0. & 0. \\ 0. & 0. & 2.54959 & 2.89898 \end{pmatrix}$$

In[1066]:=

```
uop[t_] := -Ko.x[t]
yop[t_] := C1.x[t];
StEqn = Thread[D[x[t], t] == A.x[t] + B.uop[t]] // Chop;
ColumnForm[StEqn]
IC = Thread[x[0] == {1, -1, 5, -5}] // Flatten
```

Out[1069]=

$x_1'[t] == -6.97074\, x_1[t] - 10.2956\, x_2[t]$
$x_2'[t] == x_1[t]$
$x_3'[t] == -5.54959\, x_3[t] - 4.89898\, x_4[t]$
$x_4'[t] == x_3[t]$

Out[1070]=

$\{x_1[0] == 1,\ x_2[0] == -1,\ x_3[0] == 5,\ x_4[0] == -5\}$

In[1044]:=

```
Clear[u]

xov[t_] := Array[xo#[t] &, Length[A]];
uoc[t_] := -Ko.xov[t]
yhat[t_] := C1.xov[t]
ynoise[t_] := C1.x[t] + ConstantArray[θ, Length[y[t]]]

EqObserverOp =
  Thread[xov'[t] == A.xov[t] + B.uoc[t] + Lop.(ynoise[t] - yhat[t])] // Chop // Flatten;
EqObsControllerOp = Thread[
   x'[t] == (A - B.Ko).x[t] + B.Ko.(x[t] - xov[t]) + ConstantArray[g, Length[x[t]]]];
AllEqnOp = {EqObserverOp, EqObsControllerOp} // Chop // Flatten // Simplify;
ColumnForm[AllEqnOp];
IC = Thread[x[0] == {1, -1, 5, -5}] // Flatten;
ICo = Thread[xov[0] == {0, 0, 0, 0}] // Flatten;
```

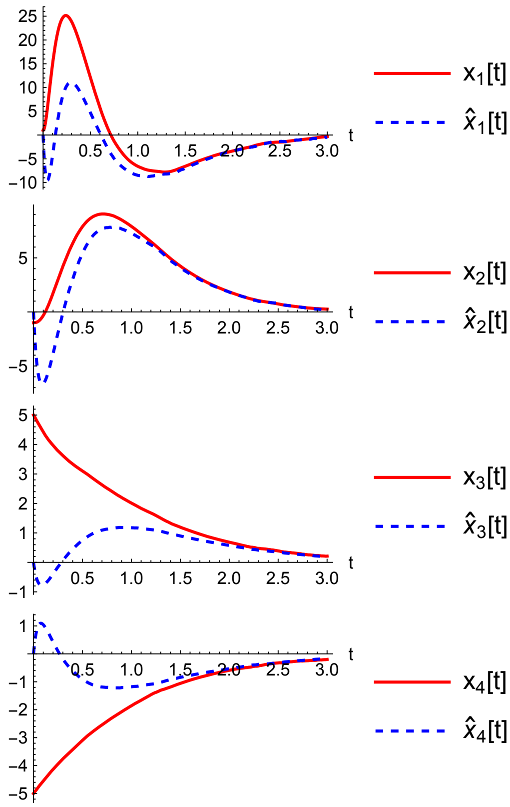
In[1064]:=

```
SolObsControllerOp = NDSolve[{AllEqnOp, IC, ICo} // Flatten,
    {x[t], xov[t]} // Flatten, {t, 0, tf}] // Flatten;
```

In[1065]:=

```
Table[Plot[Evaluate[{x[t][[i]], xov[t][[i]]} /. SolObsControllerOp], {t, 0, tf},
   PlotStyle → {{Red}, {Dashed, Blue}}, PlotRange → All, AxesLabel → {"t"},
   PlotLegends → {ToString[Subscript["x", i], StandardForm] <> "[t]",
     ToString[Subscript["x̂", i], StandardForm] <> "[t]"}],
  {i, 1, Length[x[t]]}] // TableForm
```

Out[1065]//TableForm=




This response is better than the the first one (part v, given pole placement) since we see