F I V E

Reduction of Multiple

Subsystems

SOLUTIONS TO CASE STUDIES CHALLENGES

**Antenna Control: Designing a Closed-Loop Response**

**a.** Drawing the block diagram of the system:



Thus, T(s) =

**b.**  Drawing the signal flow-diagram for each subsystem and then interconnecting them yields:



1 = x2

2 = - 1.32x2 + 0.8x3

3 = -150x3 +150K(= -95.49Kx1 - 150x3 + 477.46Ki

o = 0.2x1

In vector-matrix notation,



**c.** 

****

Nontouching loops:

GL1GL2 =

 = 1 - [GL1 + GL2 + GL3] + [GL1GL2] = 1 + + + +

1 = 1

T(s) = =

**d.** The equivalent forward path transfer function is G(s) = .

Therefore,

T(s) =

The poles are located at -0.66 ± j1.454. n = = 1.597 rad/s; 2n = 1.32, therefore,  = 0.413.

; Ts = = = 6.06 seconds; Tp = = = 2.16 seconds; Using Figure 4.16, the normalized rise time is 1.486. Dividing by the natural frequency, Tr = = 0.93 seconds.

**e.**









**f.** Since G(s) = , T(s) = . Also,  = = 0.517 for 15% overshoot; n = ; and 2n = 1.32. Therefore, n = = = 1.277 = .   
Solving for K, K=3.2.

**UFSS Vehicle: Pitch-Angle Control Representation**

**a.** Use the observer canonical form for the vehicle dynamics so that the output yaw rate is a state variable.



**b.** Using the signal flow graph to write the state equations:



In vector-matrix form:



**c.**

**Program:**

numg1=-0.25\*[1 0.437];

deng1=poly([-2 -1.29 -0.193 0]);

'G(s)'

G=tf(numg1,deng1)

numh1=[-1 0];

denh1=[0 1];

'H(s)'

H=tf(numh1,denh1)

'Ge(s)'

Ge=feedback(G,H)

'T(s)'

T=feedback(-1\*Ge,1)

[numt,dent]=tfdata(T,'V');

[Acc,Bcc,Ccc,Dcc]=tf2ss(numt,dent)

**Computer response:**

ans =

G(s)

Transfer function:

-0.25 s - 0.1093

--------------------------------------

s^4 + 3.483 s^3 + 3.215 s^2 + 0.4979 s

ans =

H(s)

Transfer function:

-s

ans =

Ge(s)

Transfer function:

-0.25 s - 0.1093

--------------------------------------

s^4 + 3.483 s^3 + 3.465 s^2 + 0.6072 s

ans =

T(s)

Transfer function:

0.25 s + 0.1093

-----------------------------------------------

s^4 + 3.483 s^3 + 3.465 s^2 + 0.8572 s + 0.1093

Acc =

-3.4830 -3.4650 -0.8572 -0.1093

1.0000 0 0 0

0 1.0000 0 0

0 0 1.0000 0

Bcc =

1

0

0

0

Ccc =

0 0 0.2500 0.1093

Dcc =

0

Answers to Review Questions

**1.** Signals, systems, summing junctions, pickoff points

**2.** Cascade, parallel, feedback

**3.** Product of individual transfer functions, sum of individual transfer functions, forward gain divided by one plus the product of the forward gain times the feedback gain

**4.** Equivalent forms for moving blocks across summing junctions and pickoff points

**5.** As K is varied from 0 to ∞, the system goes from overdamped to critically damped to underdamped. When the system is underdamped, the settling time remains constant.

**6.** Since the real part remains constant and the imaginary part increases, the radial distance from the origin is increasing. Thus the angle  is increasing. Since = cos  the damping ratio is decreasing.

**7.** Nodes (signals), branches (systems)

**8.** Signals flowing into a node are added together. Signals flowing out of a node are the sum of signals flowing into a node.

**9.** One

**10.** Phase-variable form, cascaded form, parallel form, Jordan canonical form, observer canonical form

**11.** The Jordan canonical form and the parallel form result from a partial fraction expansion.

**12.** Parallel form

**13.** The system poles, or eigenvalues

**14.** The system poles including all repetitions of the repeated roots

**15.** Solution of the state variables are achieved through decoupled equations. i.e. the equations are solvable individually and not simultaneously.

**16.**  State variables can be identified with physical parameters; ease of solution of some representations

**17.**  Systems with zeros

**18.** .State-vector transformations are the transformation of the state vector from one basis system to another. i.e. the same vector represented in another basis.

**19.** A vector which under a matrix transformation is collinear with the original. In other words, the length of the vector has changed, but not its angle.

**20.** An eigenvalue is that multiple of the original vector that is the transformed vector.

**21.** Resulting system matrix is diagonal.

SOLUTIONS to Problems

**1.**

**a.** Combine the inner feedback and the parallel pair.



Multiply the blocks in the forward path and apply the feedback formula to get,

T(s) = .

**b.**

**Program:**

'G1(s)'

G1=tf(1,[1 0 0])

'G2(s)'

G2=tf(50,[1 1])

'G3(s)'

G3=tf(2,[1 0])

'G4(s)'

G4=tf([1 0],1)

'G5(s)'

G5=2

'Ge1(s)=G2(s)/(1+G2(s)G3(s))'

Ge1=G2/(1+G2\*G3)

'Ge2(s)=G4(s)-G5(s)'

Ge2=G4-G5

'Ge3(s)=G1(s)Ge1(s)Ge2(s)'

Ge3=G1\*Ge1\*Ge2

'T(s)=Ge3(s)/(1+Ge3(s))'

T=feedback(Ge3,1);

T=minreal(T)

**Computer response:**

ans =

G1(s)

Transfer function:

1

---

s^2

ans =

G2(s)

Transfer function:

50

-----

s + 1

ans =

G3(s)

Transfer function:

2

-

s

ans =

G4(s)

Transfer function:

s

ans =

G5(s)

G5 =

2

ans =

Ge1(s)=G2(s)/(1+G2(s)G3(s))

Transfer function:

50 s^2 + 50 s

-------------------------

s^3 + 2 s^2 + 101 s + 100

ans =

Ge2(s)=G4(s)-G5(s)

Transfer function:

s - 2

ans =

Ge3(s)=G1(s)Ge1(s)Ge2(s)

Transfer function:

50 s^3 - 50 s^2 - 100 s

-------------------------------

s^5 + 2 s^4 + 101 s^3 + 100 s^2

ans =

T(s)=Ge3(s)/(1+Ge3(s))

Transfer function:

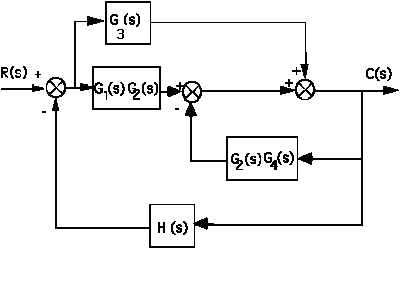
50 s - 100

-----------------------

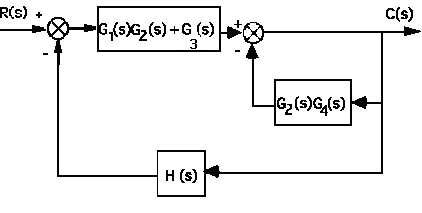
s^3 + s^2 + 150 s - 100

**2.**

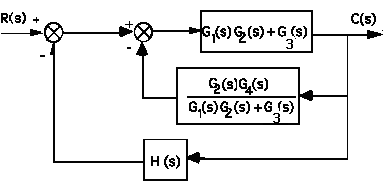
Push G2(s) to the left past the summing junction.



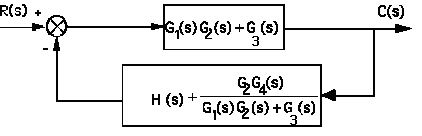
Collapse the summing junctions and add the parallel transfer functions.



Push G1(s)G2(s) + G5(s) to the right past the summing junction.



Collapse summing junctions and add feedback paths.



Applying the feedback formula,



**3.**

Combine G6 and G7 yielding G6G7. Add G4 and obtain the following diagram:



Next combine G3 and G4+G6G7.



Push G5 to the left past the pickoff point.



Notice that the feedback is in parallel form. Thus the equivalent feedback, Heq(s) = + G3(G4+G6G7) + G8. Since the forward path transfer function is G(s) = Geq(s) = G1G5, the closed-loop transfer function is

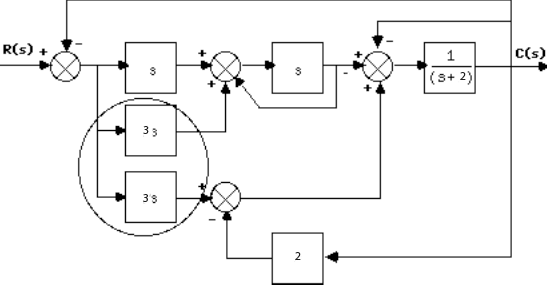
T(s) = .

Hence,

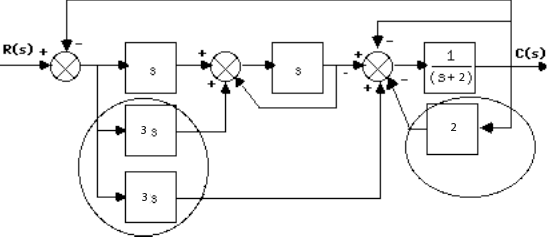


**4.**

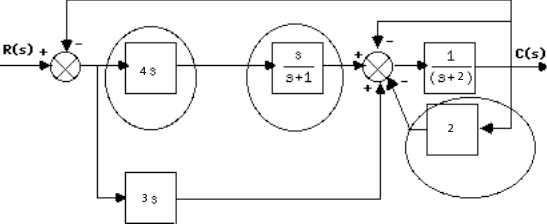
Push 3*s* to the right past the pickoff point.



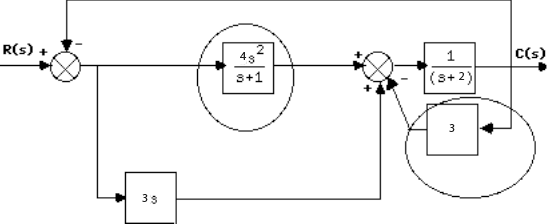
Combine summing junctions.



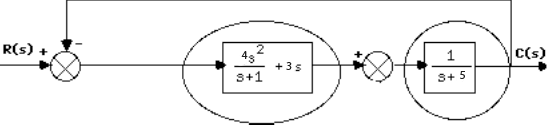
Combine parallel 3s and s. Apply feedback formula to unity feedback with G(s) = s.



Combine cascade pair and add feedback around 1/(s+1).



Combine parallel pair and feedback in forward path.



The cascade pair gives and 

**5.**

Combine the feedback with G6 and combine the parallel G2 and G3.



Move G2+G3 to the left past the pickoff point.



Combine feedback and parallel pair in the forward path yielding an equivalent forward-path transfer

function of

Ge(s) =

But, T(s) = . Thus,



**6.**

. Therefore, and = 11. Hence,= 0.5<1 so the system is underdamped. ; Ts = =0.7273 sec; Tp = =1.0938 sec.

**7.**

**8.**

The forward paths are and . The loops are , , , and .

There are no non-touching loops. So . When either of the forward paths is removed there are no loops left, so . The closed loop transfer function is

Solving for the roots of the denominator one gets -0.3219 and -1.5531.

**9.**

Since, . Therefore, 2*n* = 20. Thus,  = 10/*n* .For a 15% overshoot . Hence, *n* = 52.95 = . Therefore K = 2803.3.

**10.**

;

;

.

Therefore, *n*= 39.8; *K* = *n*2 = 1584; ** = *2n* = 47.06.

**11.**

We first find ξ, necessary for the specifications. We have and . Eliminating from both equations we get . Cross-multiplying, squaring both sides and solving, we get . . The closed loop transfer function of the system is:

From which we get that or and or .

.

**12.**

**a.** For the inner loop, Ge(s) = , and He(s) = 0.2s. Therefore, Te(s) = = . Combining with the equivalent transfer function of the parallel pair, Gp(s) = 20, the system is reduced to an equivalent unity feedback system with G(s) = Gp(s) Te(s) = . Hence, T(s) = = .

**b.** n2 = 400; 2n = 16. Therefore, n = 20, and ;   
Ts = =0.5; Tp = =0.171. From Figure 4.16, nTr = 1.463. Hence, Tr = 0.0732.   
d = Im = n= 18.33.

**13.**

The closed loop transfer function of the system is

From which we get and . The %OS==13.3%.

sec. sec. From the figure we get that from which we get sec.

14.

For the generator, *Eg(s)* = *Kf If (s)*. But, *If (s)*= . Therefore, = . For the motor, consider Ra = 2 the sum of both resistors. Also, Je = Ja+JL()2 = 0.75+1x= 1; De = DL()2 = 1. Therefore,

=  = .

But, = . Thus, = . Finally,

= = .

**15.**

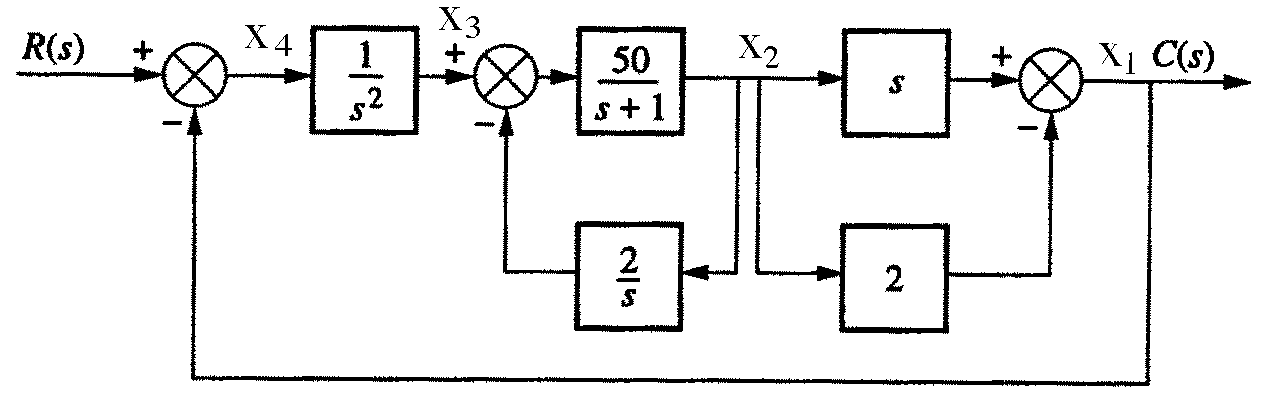
For the mechanical system, J()2s22(s) = T() . For the potentiometer, Ei (s)= 10 , or

2(s) = Ei(s). For the network, Eo(s) = Ei(s) = Ei(s) , or Ei(s) = Eo(s) .

Therefore, . Substitute into mechanical equation and obtain, .

**16.**

**a.**

****

****

**b.**





**c.**





**G(s) =**

**171117.**

**a.**

The transfer function is

The corresponding differential equation is:

The state variables are defined as: ; and . From which we get

The corresponding signal flow diagram:



**b.**

The transfer function can be expressed as from which the signal flow diagram is obtained.



The state space representation is

**18.**

**a.** Since G(s) = = ,



Let,



Therefore,





**b.** G(s) = () () () (). Hence,



From which,



19.

 = 1 + [G2G3G4 + G3G4 + G4 + 1] + [G3G4 + G4]; T1 = G1G2G3G4; 1 = 1. Therefore,

T(s) = =

**20.**

Closed-loop gains: -s2; - ; - ; -s2

Forward-path gains: T1 = s; T2 =

Nontouching loops: None

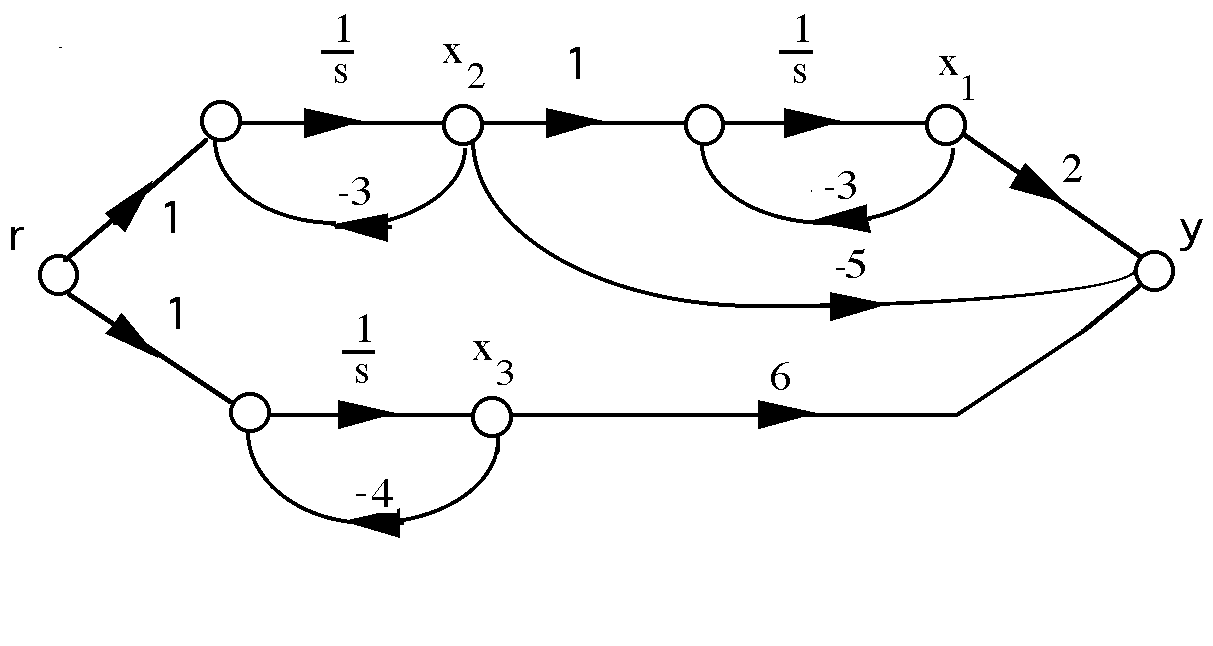
 = 1 - (-s2 - - - s2)

1 = 2 = 1

G(s) = = =

**21.**

**a. **

****

Writing the state and output equations,

1 = -3x1 + x2

2 = -3x2 + r

3 = -4x3 + r

y = 2x1 - 5x2 + 6x3

In vector-matrix form,



**b.** G(s) = 



Writing the state and output equations,

1 = -5x1 + x2

2 = -5x2 + r

3 = -7x3 + x4

4 = -7x4 + r

y = - x1 + x2 - x3 - x4

In vector matrix form,



**c.**



Writing the state and output equations,

1 = - 2x1 + x2

2 = - 2x2 + r

3 = - 5x3 + r

4 = - 6x4 + r

y = x1 - x2 - x3 + x4

In vector-matrix form,





**22.**

**a.**



Writing the state equations,

1 = x2

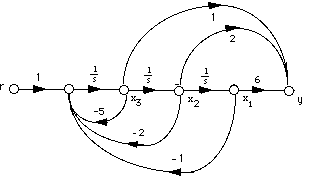
2 = - 7x1 - 2x2 + r

y = 3x1 + x2

In vector matrix form,



**b.**

****

Writing the state equations**,**

****

In vector matrix form,

****

**c.**



1 = x2

2 = x3

3 = x4

4 = - 4x1 - 6x2 - 5x3 - 3x4 + r

y = x1 + 7x2 + 2x3 + x4

In vector matrix form,



**23.**

**a.** Controller canonical form:

From the phase-variable form in Problem 5.31(a), reverse the order of the state variables and obtain,

2 = x1

1 = - 7x2 - 2x1 + r

y = 3x2 + x1

Putting the equations in order,

1 = - 2x1 - 7x2 + r

2 = x1

y = x1 + 3x2

In vector-matrix form,



Observer canonical form:

G(s) = . Divide each term by and get



Cross multiplying,

(+ ) R(s) = (1 + + ) C(s)

Thus,

+ = C(s)

Drawing the signal-flow graph,



Writing the state and output equations,

1 = - 2x1 + x2 + r

2 = - 7x1 + 3r

y = x1

In vector matrix form,



**b.** Controller canonical form:

From the phase-variable form in Problem 5.31(b), reverse the order of the state variables and obtain,

****

Putting the equations in order,

****

In vector-matrix form,



Observer canonical form:

. Divide each term by  and get



Cross-multiplying,



Thus,



Drawing the signal-flow graph,



Writing the state and output equations,



In vector-matrix form,



**c.** Controller canonical form:

From the phase-variable form in Problem 5.31(c), reverse the order of the state variables and obtain,

4 = x3

3 = x2

2 = x1

1 = - 4x4 - 6x3 - 5x2 - 3x1 + r

y = x4 + 7x3 + 2x2 + x1

Putting the equations in order,

1 = - 3x1 - 5x2 - 6x3 - 4x4 + r

2 = x1

3 = x2

4 = x3

y = x1 + 2x2­ +7x3 + x4

In vector-matrix form,



Observer canonical form:

G(s) = . Divide each term by and get



Cross multiplying,

(+ + + ) R(s) = (1 + + + + ) C(s)

Thus,

+ + + = C(s)

Drawing the signal-flow graph,



Writing the state and output equations,

1 = - 3x1 + x2 + r

2 = - 5x1 + x3 + 2r

3 = - 6x1 + x4 +7r

4 = - 4x1 + r

y = x1

In vector matrix form,



**24.**

**a.**



**-2**

**-8**

**-9**

Writing the state equations,



In vector-matrix form,



**b.**



**-6**

**-24**

Writing the state equations,



In vector-matrix form,



**c.**



1 = x2

2 = -x2 - x2 + 160(r-x1) = -160x1 -2x2 +160r

y = x1

In vector-matrix form,





**d.** Since = , we draw the signal-flow as follows: 

Writing the state equations,

1 = x2

2 = -x1 - 2x2 + 16(r-c) = -x1 - 2x2 + 16(r - (2x1+x2) = -33x1 - 18x2 + 16r

y = 2x1 + x2

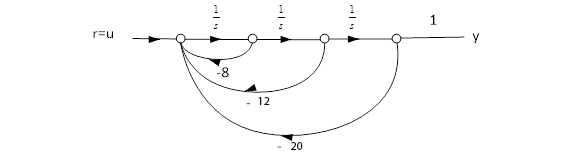
In vector-matrix form,

 .



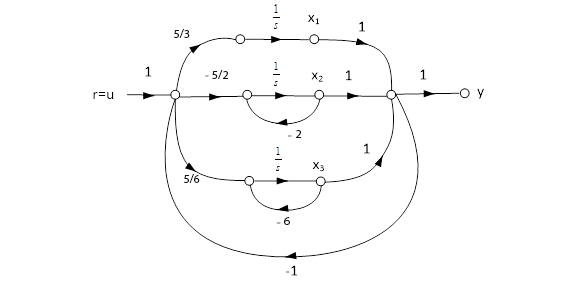
**25.**

**a.** Phase Variable form



Or in matrix form

1. Parallel form



The state equations are:

In matrix form:

**26.**

**a. **

Drawing the signal-flow diagram,



Writing the state and output equations,



In vector-matrix form,



**b. **

Drawing the signal-flow diagram and including the unity-feedback path,



Writing the state and output equations,



In vector-matrix form,

****

**27.**

**Program:**

'(a)'

'G(s)'

G=zpk([-2 -3],[-1 -4 -5 -6],10)

'T(s)'

T=feedback(G,1,-1)

[numt,dent]=tfdata(T,'v');

'Find controller canonical form'

[Acc,Bcc,Ccc,Dcc]=tf2ss(numt,dent)

A1=flipud(Acc);

'Transform to phase-variable form'

Apv=fliplr(A1)

Bpv=flipud(Bcc)

Cpv=fliplr(Ccc)

'(b)'

'G(s)'

G=zpk([-2 -3],[-1 -4 -5 -6],10)

'T(s)'

T=feedback(G,1,-1)

[numt,dent]=tfdata(T,'v');

'Find controller canonical form'

[Acc,Bcc,Ccc,Dcc]=tf2ss(numt,dent)

'Transform to modal form'

[A,B,C,D]=canon(Acc,Bcc,Ccc,Dcc,'modal')

**Computer response:**

ans =

(a)

ans =

G(s)

Zero/pole/gain:

10 (s+2) (s+3)

-----------------------

(s+1) (s+4) (s+5) (s+6)

ans =

T(s)

Zero/pole/gain:

10 (s+2) (s+3)

------------------------------------------

(s+1.264) (s+3.412) (s^2 + 11.32s + 41.73)

ans =

Find controller canonical form

Acc =

-16.0000 -99.0000 -244.0000 -180.0000

1.0000 0 0 0

0 1.0000 0 0

0 0 1.0000 0

Bcc =

1

0

0

0

Ccc =

0 10.0000 50.0000 60.0000

Dcc =

0

ans =

Transform to phase-variable form

Apv =

0 1.0000 0 0

0 0 1.0000 0

0 0 0 1.0000

-180.0000 -244.0000 -99.0000 -16.0000

Bpv =

0

0

0

1

Cpv =

60.0000 50.0000 10.0000 0

ans =

(b)

ans =

G(s)

Zero/pole/gain:

10 (s+2) (s+3)

-----------------------

(s+1) (s+4) (s+5) (s+6)

ans =

T(s)

Zero/pole/gain:

10 (s+2) (s+3)

------------------------------------------

(s+1.264) (s+3.412) (s^2 + 11.32s + 41.73)

ans =

Find controller canonical form

Acc =

-16.0000 -99.0000 -244.0000 -180.0000

1.0000 0 0 0

0 1.0000 0 0

0 0 1.0000 0

Bcc =

1

0

0

0

Ccc =

0 10.0000 50.0000 60.0000

Dcc =

0

ans =

Transform to modal form

A =

-5.6618 3.1109 0 0

-3.1109 -5.6618 0 0

0 0 -3.4124 0

0 0 0 -1.2639

B =

-4.1108

1.0468

1.3125

0.0487

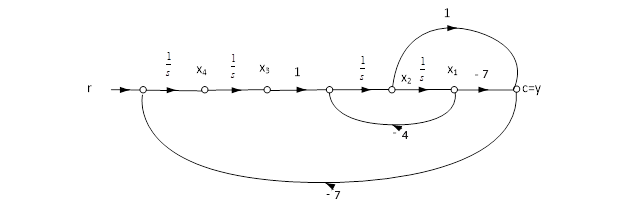
C =

0.1827 0.6973 -0.1401 4.2067

D =

0

**28.**

****

Or in Matrix form





**29.**

**a.**

1 + 51 + 61 - 32 - 42 = 0

-1 - 41 + 2 + 52 + 52 = T

or

1 = - 51 - 61 + 32 + 42

2 = 1 + 41 - 52 - 52 + T

Letting, 1 = x1 ; 1 = x2 ; 2 = x3 ; 2 = x4 ,



where x = .

**b.** Using the signal-flow diagram,

1 = x2

2 = -6x1 - 5x2 + 4x3 + 3x4

3 = x4

4 = 4x1 + 3x2 - 5x3 - 5x4 + T

y = x3

In vector-matrix form,



**30.**

1 = **A1x1** + **B1**r (1)

y1 = **C1x1** (2)

2 = **A2x2** + **B2**y1 (3)

y2 = **C2x2** (4)

Substituting Eq. (2) into Eq. (3),

1 = **A1x1** + **B1**r

2 = **B2C1x1 + A2x2**

y2 = **C2x2**

In vector-matrix notation,





**31.**

1 = **A1x1** + **B1**e (1)

y = **C1x1** (2)2 = **A2x2** + **B2**y (3)p = **C2x2** (4)

Substituting e = r - p into Eq. (1) and substituting Eq. (2) into (3), we obtain,

1 = **A1x1** + **B1**(r - p) (5)y = **C1x1** (6)2 = **A2x2** + **B2C1x1** (7)p = **C2x2** (8)

Substituting Eq. (8) into Eq. (5),

**1 = A1x1 - B1C2x2 + B1r**

**2 = B2C1x1 + A2x2y = C1x1**

In vector-matrix form,





**32.**







**33.**

Eigenvalues are 1, -2, and 3 since,

|**I** - **A** | = ( - 3) ( + 2) ( - 1)

Solving for the eigenvectors, **Ax** = **x**

or,







For  = 1, x1 = x2 = . For  = -2, x1 = 2x3, x2 = -3x3. For  = 3, x1 = x3 , x2 = -2x3 . Thus,

**= P**-1**APz** + **P**-1**B**u; y = **CPz**, where



**34.**

**Program:**

A=[-10 -3 7;18.25 6.25 -11.75;-7.25 -2.25 5.75];

B=[1;3;2];

C=[1 -2 4];

[P,d]=eig(A);

Ad=inv(P)\*A\*P

Bd=inv(P)\*B

Cd=C\*P

**Computer response:**

Ad =

-2.0000 0.0000 0.0000

-0.0000 3.0000 -0.0000

0.0000 0.0000 1.0000

Bd =

1.8708

-3.6742

3.6742

Cd =

3.2071 3.6742 2.8577

**35.**

Push Pitch Gain to the right past the pickoff point.



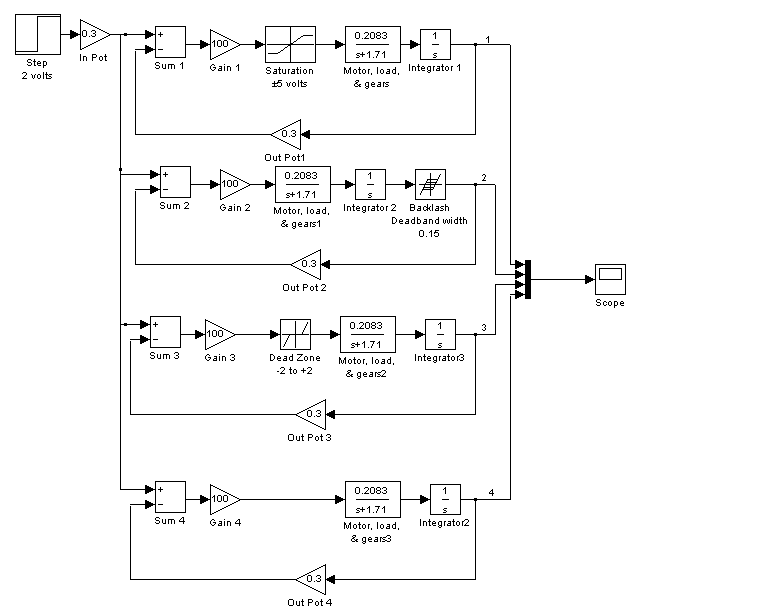
Collapse the summing junctions and add the feedback transfer functions.

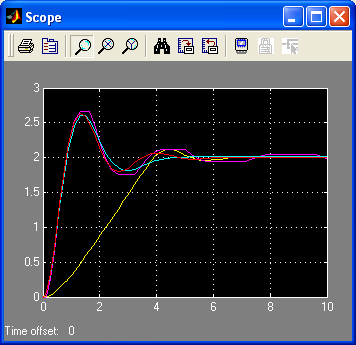


Apply the feedback formula and obtain,



**36.**

****



Linear

Deadzone

Linear

Backlash

**37.**

1. The first equation follows from the schematic. The second equation is obtained by applying the voltage divider rule at the op-amp’s inverting terminal, noting that since the op-amp considered is ideal, there is no current demand there.
2. ; ; ; 



**c.**  

**38.**

* 1. The three equations follow by direct observation from the small signal circuit.
  2. The block diagram is given by

-

+





vo

vgs

Vo

Vi

+

-

vi

1. From the block diagram we get



**39.**

**a.**  Using Mason’s rule

; Loops  and , no non-touching loops. 



1. From part (a)



**40.**

**a.**

>> A=[-100.2 -20.7 -30.7 200.3; 40 -20.22 49.95 526.1;...

0 10.22 -59.95 -526.1; 0 0 0 0];

>> B=[208; -208; -108.8; -1];

>> C = [0 1570 1570 59400];

>> D = -6240;

>> [n,d]=ss2tf(A,B,C,D)

n =

1.0e+009 \*

Columns 1 through 3

-0.00000624000000 -0.00168228480000 -0.14206098728000

Columns 4 through 5

-3.91955218234127 -9.08349454230472

d =

1.0e+005 \*

Columns 1 through 3

0.00001000000000 0.00180370000000 0.09562734000000

Columns 4 through 5

1.32499100000000 0

>> roots(n)

ans =

1.0e+002 \*

-1.34317654991673

-0.78476212102923

-0.54257777928519

-0.02545278053809

>> roots(d)

ans =

0

-92.38329312886714

-66.38046756013043

-21.60623931100260

Note that , follows that



**b.**

>> [r,p,k]=residue(n,d)

r =

1.0e+005 \*

-0.73309459854184

-0.51344619392820

-3.63566779304453

-0.68555141448543

p =

-92.38329312886714

-66.38046756013043

-21.60623931100260

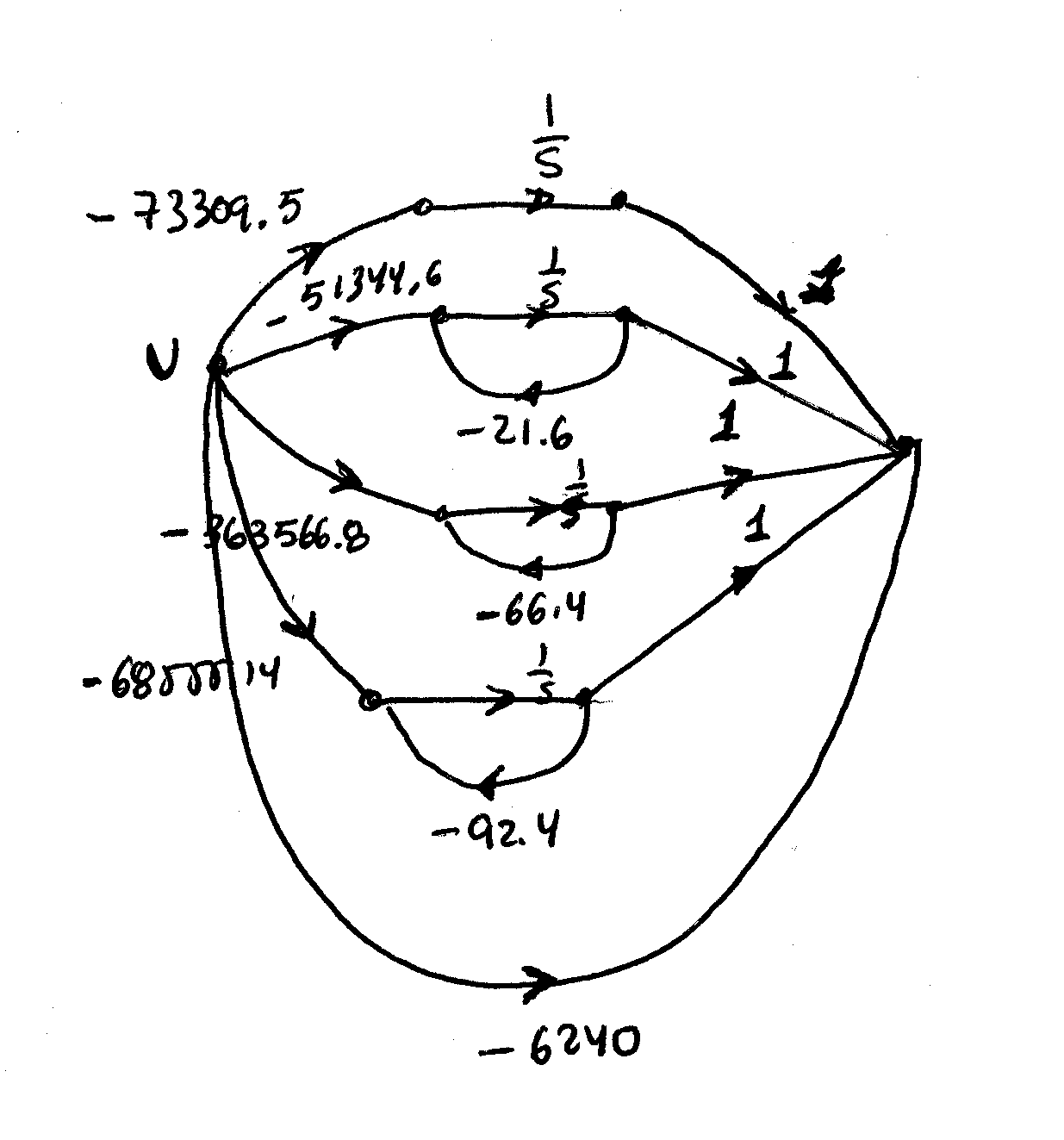
0

k =

-6240

or 

**c.**



1. The corresponding state space representation is:





**41.**

**a.**

>> A = [0 1 0; 0 -68.3 -7.2; 0 3.2 -0.7]

A =

0 1.0000 0

0 -68.3000 -7.2000

0 3.2000 -0.7000

>> [V,D]=eig(A)

V =

1.0000 0.0147 -0.1016

0 -0.9988 0.1059

0 0.0475 -0.9892

D =

0 0 0

0 -67.9574 0

0 0 -1.0426

Matrix V is the sought similarity transformation.

**b.**

>> Ad = inv(V)\*A\*V

Ad =

0 -0.0000 -0.0000

0 -67.9574 0.0000

0 -0.0000 -1.0426

>> B = [0;425.4;0]

B =

0

425.4000

0

>> Bd = inv(V)\*B

Bd =

4.2030

-428.1077

-20.5661

The diagonalized system is:



**42.**

**a.**



1. There is only one forward path 

There are three loops: ; and 

 and  are non-touching loops so



When  is eliminated only  is left so 

Finally 

**43.**

1. There are two forwards paths:

= and =

The loops are:

There are no non-touching loops. Therefore

Also

**b.**

There is only one forward path

The loops and are the same as in part a. Also . It follows that



**44.**

1. Assuming there are two forward paths, and

The loops are

There are two non-touching loops and .

+=

We also have that by eliminating or

1. The system can be described by means of the following diagram:



It follows that

**45.**

1. Substituting the values given above into the block diagram, when Δ*vp* = 0, we have:

*ΔHm(s)*



*X(s)*

*YC(s)*

*E(s)*

*R(s)*

\_

+

*GC(s)*

*GV(s)*

*GX(s)*

*Gm(s)*

*β*

**The Mould Level Block Diagram for Δ*vp* = 0**

Thus, the closed-loop transfer function is:

 =

=  =

=  =

= 

1. Simulink was used to simulate the system. The model of that system is shown in Figure P5.x-4\*. The parameters of the PID controller were set to: *Kp* = 2, *Kd* = 1.6, and *KI* = 0.4. The reference step, *r(t)* = 5 u(t), and thecasting speed step, *vp* (t) = 0.97 u(t) were set to start at t = 0. An adder was used to add the initial value,   
   *Hm* (0—) = – 75 mm, at the output, to the change in mould level, *ΔHm*.

The time and mould level (in array format) were output to “workspace ” sinks, each of which was given the respective variable name. After the simulation ended, Matlab plot commands were used to obtain and edit the graph of *hm(t)* from 0 to *t* = 80 seconds.



**Simulink Model of the Mould Level Control System**

****

**Response of the Mould Level to Simultaneous Step Changes in Reference Input, *r(t)* = 5 u(t), and Casting Speed, *vp* (t) = 0.97 u(t) at an Initial Level, *Hm* (0—) = – 75**

**46.**

-

-



C(s)

R(s)

C(s)

R(s)

+

+

-



It can be easily verified that the closed loop transfer function for this system is identical to the original.

**47.**

The closed loop transfer function is where

and

Substituting

**48.**

The MATLAB M-file is:

num1=25;

num2=[1 1.2 12500];

den1=[1 0];

den2=[1 5.6 62000];

num=conv(num1, num2);

den=conv(den1, den2);

G=tf(num, den);

D=feedback(G,0.1);

[numd,dend]=tfdata(D,'v');

numcm=[40 5];

dencm=[1 0];

numOL=conv(numd, numcm);

denOL=conv(dend, dencm);

Omega\_OL=tf(numOL, denOL);

Omega\_CL=260\*feedback(Omega\_OL,1);

step(Omega\_CL, 0:0.002:0.2);

grid;

After the above file is run, MATLAB’s command window may be used to obtaintherequested minor-loop transfer function:

D =

25 s^2 + 30 s + 312500

----------------------------------------

s^3 + 8.1 s^2 + 62003 s + 31250

Continuous-time transfer function.

The MATLAB figure, shown below, illustrates the step-response obtainedwith all of the requested important characteristics noted on it.

****

**49.**

**a.**

Note that due to the topology, the loop on the top should have no influence whatsoever on output . Applying Mason’s: There are two forward paths and . There are two loops and ; both loops are non-touching. Thus . Eliminating forward path 1: . Eliminating forward path 2: . The closed loop transfer function is:

**b.**

In this case there are three forward paths ; ; . The loops and are just as in part a. Eliminating forward path 1 Eliminating either paths 2 or 3 . The closed loop transfer function is:

**50.**

There are three forward paths ; and . There are two loops: and . There are no non-touching loops. So . Eliminating forward paths 1 or 2 leaves no loops so . Eliminating forward path 3 leaves one loop so . The closed loop transfer function is:

**51.**

|  |  |
| --- | --- |
| Showing the equivalent circuit here for reference we have:  (1)  Substituting the equation given in the problem into (1), re-arranging, and simplifying gives: |  |

 (2)

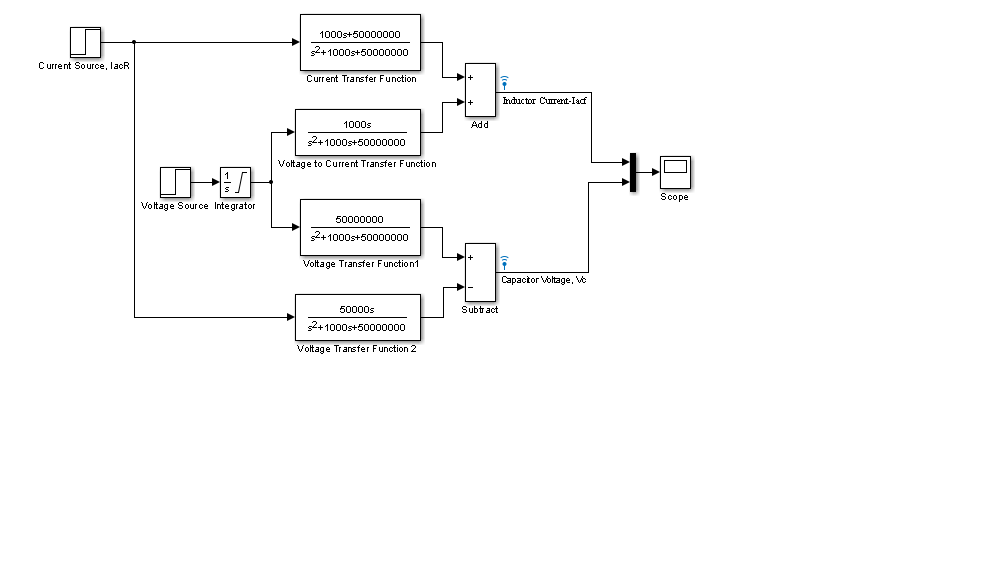


Substituting circuit parameters into the equation given in the problem statement & (2), and assuming zero initial conditions, we have:

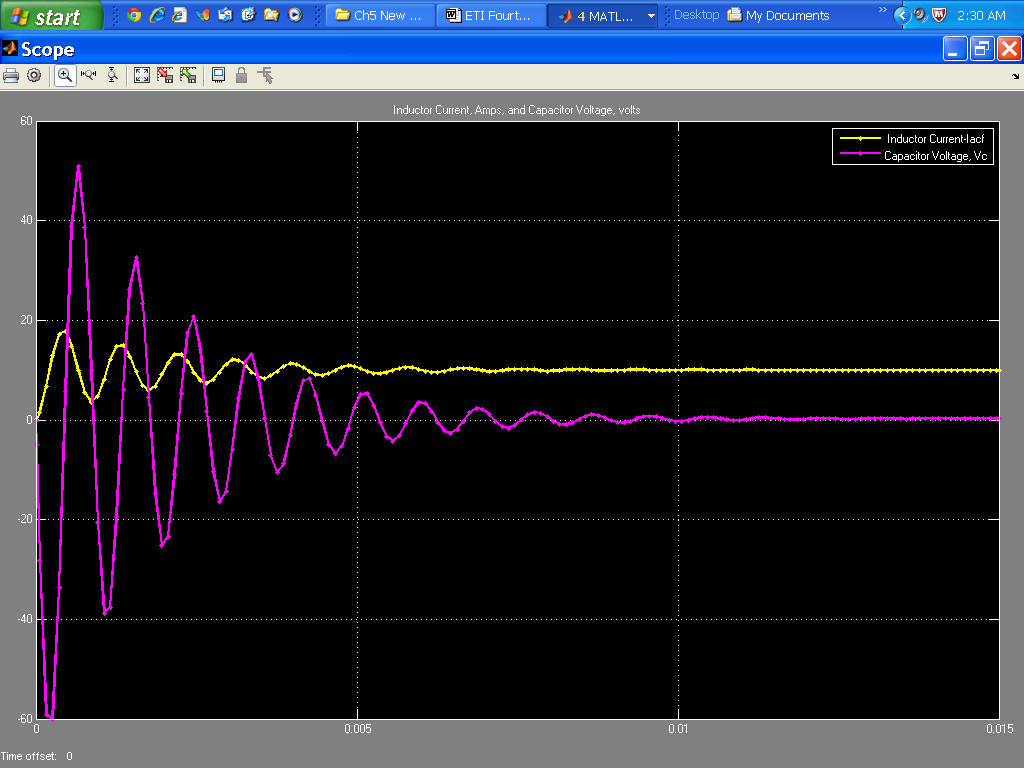
 (3)

(4)

The Simulink model is shown below. The current source, *IacR*, (a step input) was set to model *iacR* (t) = 10 u(t), amps, and the voltage ramp, *va* (t) = 20 t u(t), was created using a step of 20 volts and an integrator with the upper saturation limit set to 20 volts and the lower to 0. The result of the simulation was captured on a “scope” that was set to display the two input variables (using a mux) from t = 0 to 15 ms, with the y-axis range set from – 60 to +60.



**Simulink model of the system**



**Inductor Current, *iacF(t)* in amps, and Capacitor Voltage, *vC(t)* in volts, as Displayed on the scope at the end of simulation (set to 15 ms).**

**SOLUTIONS TO DESIGN PROBLEMS**

**52.**

Je = Ja+JL()2 = 2+2 = 4; De = Da+DL()2 = 2+DL()2. Therefore, the forward-path transfer function is,

G(s) = (1000) . Thus, T(s) = = .   
Hence,  = = 0.456; n = ; 2n = . Therefore De = 10.9; from which DL = 3560.

**53.**

The equivalent forward path transfer function is Ge(s) = . Thus, T(s) = = . Prior to tachometer compensation (K2 = 0), T(s) = . Therefore K = n2 = 100. Thus, after tachometer compensation, T(s) = . Hence, n = 10; 2n = 1+K2. Therefore, K2 = 2n - 1 = 2(0.69)(10) - 1 = 12.8.

**54.**

At the *N*2 shaft, with rotation,



Thus,



But, . Hence,



where



Thus, the total load inertia and load damping is



Reflecting *JL* and *DL* to the motor yields,



Thus, the motor transfer function is



The gears are (10/20)(1) = 1/2. Thus, the forward-path transfer function is



Finding the closed-loop transfer function yields,



For *Ts* = 2, . For 20% overshoot,  = 0.456. Thus,



Or, ; from which  and hence, . But, . Thus, M = 11.75 and D = 196.

**55.**

**a.** The leftmost op-amp equation can be obtained by superposition. Let , then the circuit is an inverting amplifier thus . Now let , the circuit is a non-inverting amplifier with an equal resistor voltage divider at its input, thus . Adding both input components 

**b.**  The two equations representing the system are:

 and



The block diagram is:

-1

-

+



Vo

Vi

+

1. From the figure



1. The system is first order so from which



**e.** For a unit step input the output will look as follows



**56.**

1. The transfer function obtained in Problem 3.32 is

 by inspection we write the phase-variable form





1. We renumber the phase-variable form state variables in reverse order





And we rearrange in ascending numerical order to obtain the controller canonical form:





1. To obtain the observer canonical form we rewrite the system’s transfer function as:



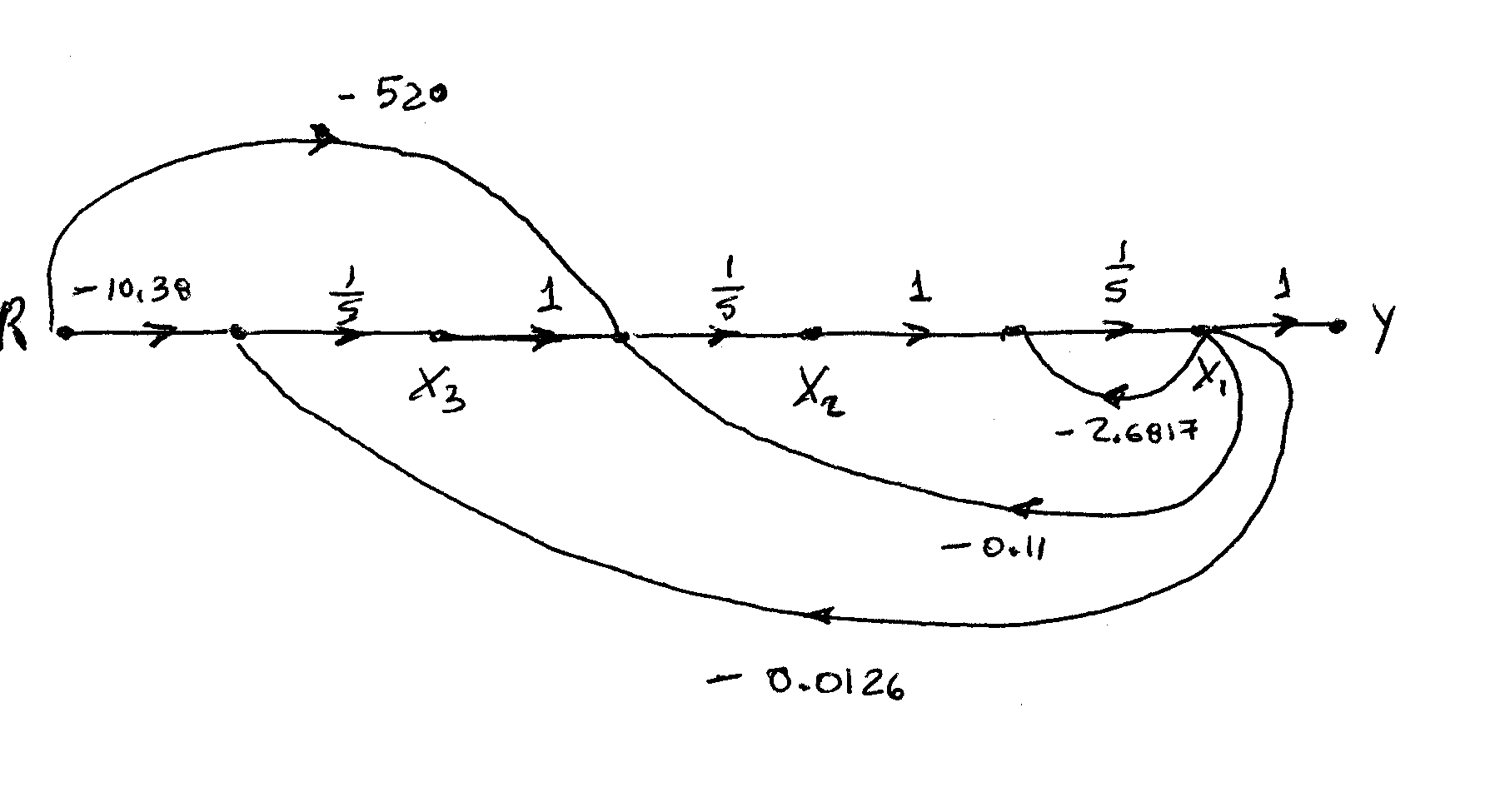
We cross-multiply to obtain



Combining terms with like powers of integration:



We draw the signal flow graph:



The following equations follow:









Which lead to observer canonical form:

; 

**d.**

>> A=[-0.04167 0 -0.0058; 0.0217 -0.24 0.0058; 0 100 -2.4];

>> B=[5.2;-5.2;0];

>> C=[0 0 1];

>> [V,D]=eig(A);

>> Bd=inv(V)\*B

Bd =

1.0e+002 \*

-0.9936 + 0.0371i

-0.9936 - 0.0371i

1.9797

>> Cd = C\*V

Cd =

0.9963 0.9963 1.0000

>> D

D =

-0.0192 + 0.0658i 0 0

0 -0.0192 - 0.0658i 0

0 0 -2.6433

So a diagonalized version of the system is





**57.**

1. Substituting all values and transfer functions into the respective blocks of the system (Figure 4), we get:

Vehicle Speed

*V(s)*

+

Torque

Controller

& Power Amplifier

Ref. Signal

*Rv(s)*

0.6154

Moving the last pick-off point to the left past the ** block and changing the position of the back-emf feedback pick-off point, so that it becomes an outer loop, we obtain the block-diagram shown below. In that diagram the  block (representing the total inertia) has two parallel feedback blocks. Reducing these two blocks into one, we have the following equivalent feedback transfer function:



Replacing that feedback loop with its equivalent transfer function, *Geq(s)*, we have:



Moving the armature current pick-off point to the right past the  and *Geq*(s) blocks, the above block-diagram becomesas shown below**.**



The latter, in turn, can be reduced to that shown next as the cascaded blocks in the feedback to the torque controller are replaced by the single block:  and the inner feedback loop is replaced by its equivalent transfer function:





Thus: 

Finally or



Hence: 

**b**. Simulink was used to model the HEV cascade control system. That model is shown below. The reference signal, *rv (t)*, was set as a step input with a zero initial value, a step time = 0 seconds, and a final value equal to 4 volts [corresponding to the desired final car speed, *v* (∞) = 60 km/h, e.g. a desired final value of the change in car speed, Δ*v* (∞) = 5.55 m/s]. The variables of interest [time, change in car speed, acceleration, and motor armature current] were output (in array format) to four “workspace” sinks, each of which was assigned the respective variable name. After the simulation ended, Matlab plot commands were utilized to obtain and edit the required three graphs. These graphs are shown below.

The simulations show that in response to such a speed reference command, car acceleration would go initially to a maximum value of 10.22 m/s2 and the motor armature current would reach a maximum value of 666.7 A. That would require an electric motor drive rated around 80 kW or using both the electric motor and gas or diesel engine, when fast acceleration is required. Most practical HEV control systems, however, use current-limiting and acceleration-limiting devices or software programs.

**Model of the HEV Cascade Control System**

**Change in car speed in response to a speed reference signal step of 4 volts**



**Car acceleration reponse to a speed reference signal step of 4 volts**



**Motor armature current reponse to a speed reference signal step of 4 volts**

**58.**

1. There is only 1 forward path. The gain of the forward path is . There are two loops and , and no non-touching loop; so . After the forward loop is eliminated The closed-loop transfer function from command input to output is:
2. There is only 1 forward path. The gain of the forward path is There are two loops and , and no non-touching loop; so . After the forward loop is eliminated one loop remains so The closed-loop transfer function from command input to output is:
3. The total output is
4. In Figure P5.54(*b*) let . The closed-loop transfer function from command input to system output is:

Similarly the transfer function from disturbance to system outputs is: