F O U R

Time Response

SOLUTIONS TO CASE STUDIES CHALLENGES

**Antenna Control: Open-Loop Response**

The forward transfer function for angular velocity is,

G(s) = =

**a.** 0(t) = A + Be-150t + Ce-1.32t

**b.** G(s) = . Therefore, 2n =151.32, n = 14.07, and  = 5.38.

**c.** 0(s) = =



Therefore, 0(t) = 0.12121 + .0010761 e-150t - 0.12229e-1.32t.

**d.** Using G(s),

****

Defining,

****

Thus, the state equations are,



In vector-matrix form,



**e.**

**Program:**

'Case Study 1 Challenge (e)'

num=24;

den=poly([-150 -1.32]);

G=tf(num,den)

step(G)

**Computer response:**

ans =

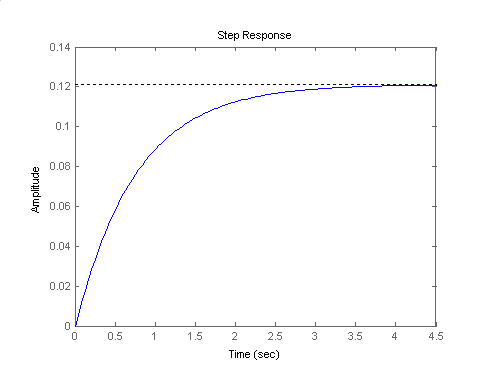
Case Study 1 Challenge (e)

Transfer function:

24

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

s^2 + 151.3 s + 198



**Ship at Sea: Open-Loop Response**

**a.** Assuming a second-order approximation: n2 = 2.25, 2n = 0.5. Therefore  = 0.167, n = 1.5. Ts = = 16; TP = = 2.12 ;   
%OS = e- / x 100 = 58.8%; nTr = 1.169 therefore, Tr = 0.77.

**b.**  **=** 

**=** 

**=** 

Taking the inverse Laplace transform,

(t) = 1 - e-0.25t ( cos1.479t +0.16903 sin1.479t)

**c.**

**Program:**

'Case Study 2 Challenge (C)'

'(a)'

numg=2.25;

deng=[1 0.5 2.25];

G=tf(numg,deng)

omegan=sqrt(deng(3))

zeta=deng(2)/(2\*omegan)

Ts=4/(zeta\*omegan)

Tp=pi/(omegan\*sqrt(1-zeta^2))

pos=exp(-zeta\*pi/sqrt(1-zeta^2))\*100

t=0:.1:2;

[y,t]=step(G,t);

Tlow=interp1(y,t,.1);

Thi=interp1(y,t,.9);

Tr=Thi-Tlow

'(b)'

numc=2.25\*[1 2];

denc=conv(poly([0 -3.57]),[1 2 2.25]);

[K,p,k]=residue(numc,denc)

'(c)'

[y,t]=step(G);

plot(t,y)

title('Roll Angle Response')

xlabel('Time(seconds)')

ylabel('Roll Angle(radians)')

**Computer response:**

ans =

Case Study 2 Challenge (C)

ans =

(a)

Transfer function:

2.25

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

s^2 + 0.5 s + 2.25

omegan =

1.5000

zeta =

0.1667

Ts =

16

Tp =

2.1241

pos =

58.8001

Tr =

0.7801

ans =

(b)

K =

0.1260

-0.3431 + 0.1058i

-0.3431 - 0.1058i

0.5602

p =

-3.5700

-1.0000 + 1.1180i

-1.0000 - 1.1180i

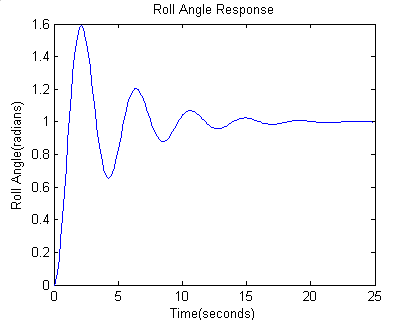
0

k =

[]

ans =

(c)



ANSWERS TO REVIEW QUESTIONS

**1.**.Time constant

**2.** The time for the step response to reach 67% of its final value

**3.** The input pole

**4.** The system poles

**5.** The radian frequency of a sinusoidal response

**6.** The time constant of an exponential response

**7.** Natural frequency is the frequency of the system with all damping removed; the damped frequency of

oscillation is the frequency of oscillation with damping in the system.

**8.** Their damped frequency of oscillation will be the same.

**9.** They will all exist under the same exponential decay envelop.

**10.** They will all have the same percent overshoot and the same shape although differently scaled in time.

**11.**  ., n, TP, %OS, Ts

**12.** Only two since a second-order system is completely defined by two component parameters

**13.** (1) Complex, (2) Real, (3) Multiple real

**14.** Pole's real part is large compared to the dominant poles, (2) Pole is near a zero

**15.** If the residue at that pole is much smaller than the residues at other poles

**16.** No; one must then use the output equation

**17.** The Laplace transform of the state transition matrix is (sI -**A**)-1

**18.** Computer simulation

**19.** Pole-zero concepts give one an intuitive feel for the problem.

**20.**  State equations, output equations, and initial value for the state-vector

**21**. Det(s**I**-**A**) = 0

SOLUTIONS to Problems

**1.**

**a.** Overdamped Case:

C(s) =

Expanding into partial fractions,



Taking the inverse Laplace transform,

c(t) = 1 + 0.171 e-7.854t - 1.171 e-1.146t

**b.** Underdamped Case:

|  |
| --- |
|  |

K2 and K3 can be found by clearing fractions with K1 replaced by its value. Thus,

9 = (s2 + 3s + 9) + (K2s + K3)s

or

9 = s2 + 3s +9 + K2s2 + K3s

Hence K2 = -1 and K3 = -3. Thus,

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

c(t) = 1 - e-3t/2 cos(t - 

= 1 - 1.155 e -1.5t cos (2.598t - )

where

 = arctan () = 30o

**c.** Oscillatory Case:

|  |
| --- |
|  |

|  |
| --- |
|  |

The evaluation of the constants in the numerator are found the same way as they were for the underdamped case. The results are K2 = -1 and K3 = 0. Hence,

|  |
| --- |
|  |

Therefore,

c(t) = 1 - cos 3t

**d.** Critically Damped

|  |
| --- |
|  |

|  |
| --- |
|  |

The constants are then evaluated as

|  |
| --- |
|  |

Now, the transform of the response is

|  |
| --- |
|  |

c(t) = 1 - 3t e-3t - e-3t

**2.**

**a.** C(s) = = - . Therefore,

Also, , sec, sec.

**b.** C(s) = = - . Therefore,

Also, , sec, sec.

**3. I-1 Instructor**

**Program:**

'(a)'

num=5;

den=[1 5];

Ga=tf(num,den)

subplot(1,2,1)

step(Ga)

title('(a)')

'(b)'

num=20;

den=[1 20];

Gb=tf(num,den)

subplot(1,2,2)

step(Gb)

title('(b)')

**Computer response:**

ans =

(a)

Transfer function:

5

-----

s + 5

ans =

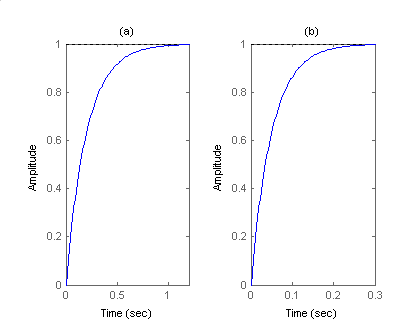
(b)

Transfer function:

20

------

s + 20



**3.**

Using voltage division, . Since 

.

Therefore: .

Also, **.**

**4.**

**Program:** clf

num=1;

den=[1 1];

G=tf(num,den);

G;

step(5\*G)

**Computer response:**

G =

1

-----

s + 1

Continuous-time transfer function.



**5.**

Writing the equation of motion,



Thus, the transfer function i*s*,



Differentiating to yield the transfer function in terms of velocity,



Thus, the settling time, *Ts*, and the rise time, *Tr,* are given by



**6.**

**a.** Pole: -2; c(t) = A + Be-2t ; first-order response.

**b.** Poles: -3, -6; c(t) = A + Be-3t + Ce-6t; overdamped response.

**c.** Poles: -10, -20; Zero: -7; c(t) = A + Be-10t + Ce-20t; overdamped response.

**d.** Poles: (-3+j3), (-3-j3) ; c(t) = A + Be-3t cos (3t + ); underdamped.

**e.** Poles: j3, -j3; Zero: -2; c(t) = A + B cos (3t + ); undamped.

**f.** Poles: -10, -10; Zero: -5; c(t) = A + Be-10t + Cte-10t; critically damped.

**7.**

**Program:**

p=roots([1 7 3 -6 2])

**Computer response:**

p =

-6.3739 + 0.0000i

-1.4551 + 0.0000i

0.4145 + 0.2094i

0.4145 - 0.2094i

**8.**

Where

So

The poles are found by obtaining the roots in the denominator of G(s) and are -2.796, 6.796 and 15.0.

**9.**

>> A=[3 -4 2;-2 0 1;4 7 -5];

>> B=[-1;-2;3];

>> C=[1 7 1];

>> D=0;

>> G=ss2tf(A,B,C,D,1);

>> [nG,dG]=ss2tf(A,B,C,D,1);

>> G=tf(nG,dG)

G =

-12 s^2 - 11 s + 288

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

s^3 + 2 s^2 - 38 s + 25

Continuous-time transfer function.

>> poles=roots(dG)

poles =

-7.5062

4.8144

0.6918

**10.**

The equation of motion is: (Ms2+fvs+Ks)X(s) = F(s).

Hence, = = .

The step response is now evaluated:

X(s) = .

Taking the inverse Laplace transform,

7. •413-Fa-72-1.1 (Chap 4)

**11.**

**a.** N/A

**b.** s2+9s+18, n2 = 18, 2n = 9, Therefore  = 1.06, n = 4.24, overdamped.

**c.** s2+30s+200, n2 = 200, 2n = 30, Therefore  = 1.06, n = 14.14, overdamped.

**d.** s2+6s+144, n2 = 144, 2n = 6, Therefore  = 0.25, n = 12, underdamped.

**e.** s2+9, n2 = 9, 2n = 0, Therefore  = 0, n = 3, undamped.

**f.** s2+20s+100, n2 = 100, 2n = 20, Therefore  = 1, n = 10, critically damped.

**12.**

**23.**

**a.** n2 = 16 r/s, 2n = 3. Therefore  = 0.375, n = 4. Ts = = 2.667 s; TP = = 0.8472 s; %OS = e- / x 100 = 28.06 %; nTr = (1.763 - 0.4172 + 1.039+ 1) = 1.4238; therefore, Tr = 0.356 s**.**

**b.** n2 = 0.04 r/s, 2n = 0.02. Therefore  = 0.05, n = 0.2. Ts = = 400 s; TP = = 15.73 s; %OS = e- / x 100 = 85.45 %; nTr = (1.763 - 0.4172 + 1.039+ 1); therefore, Tr = 5.26 s.

**c.** n2 = 1.05 x 107 r/s, 2n = 1.6 x 103. Therefore  = 0.247, n = 3240. Ts = = 0.005 s; TP = = 0.001 s; %OS = e- / x 100 = 44.92 %; nTr = (1.763 - 0.4172 + 1.039+ 1); therefore, Tr = 3.88x10-4s.

**14.**

**Program:**

'(a)'

clf

numa=16;

dena=[1 3 16];

Ta=tf(numa,dena)

omegana=sqrt(dena(3))

zetaa=dena(2)/(2\*omegana)

Tsa=4/(zetaa\*omegana)

Tpa=pi/(omegana\*sqrt(1-zetaa^2))

Tra=(1.76\*zetaa^3 - 0.417\*zetaa^2 + 1.039\*zetaa + 1)/omegana

percenta=exp(-zetaa\*pi/sqrt(1-zetaa^2))\*100

subplot(221)

step(Ta)

title('(a)')

'(b)'

numb=0.04;

denb=[1 0.02 0.04];

Tb=tf(numb,denb)

omeganb=sqrt(denb(3))

zetab=denb(2)/(2\*omeganb)

Tsb=4/(zetab\*omeganb)

Tpb=pi/(omeganb\*sqrt(1-zetab^2))

Trb=(1.76\*zetab^3 - 0.417\*zetab^2 + 1.039\*zetab + 1)/omeganb

percentb=exp(-zetab\*pi/sqrt(1-zetab^2))\*100

subplot(222)

step(Tb)

title('(b)')

'(c)'

numc=1.05E7;

denc=[1 1.6E3 1.05E7];

Tc=tf(numc,denc)

omeganc=sqrt(denc(3))

zetac=denc(2)/(2\*omeganc)

Tsc=4/(zetac\*omeganc)

Tpc=pi/(omeganc\*sqrt(1-zetac^2))

Trc=(1.76\*zetac^3 - 0.417\*zetac^2 + 1.039\*zetac + 1)/omeganc

percentc=exp(-zetac\*pi/sqrt(1-zetac^2))\*100

subplot(223)

step(Tc)

title('(c)')

**Computer response:**

ans =

(a)

Transfer function:

16

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

s^2 + 3 s + 16

omegana =

4

zetaa =

0.3750

Tsa =

2.6667

Tpa =

0.8472

Tra =

0.3559

percenta =

28.0597

ans =

(b)

Transfer function:

0.04

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

s^2 + 0.02 s + 0.04

omeganb =

0.2000

zetab =

0.0500

Tsb =

400

Tpb =

15.7276

Trb =

5.2556

percentb =

85.4468

ans =

(c)

Transfer function:

1.05e007

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

s^2 + 1600 s + 1.05e007

omeganc =

3.2404e+003

zetac =

0.2469

Tsc =

0.0050

Tpc =

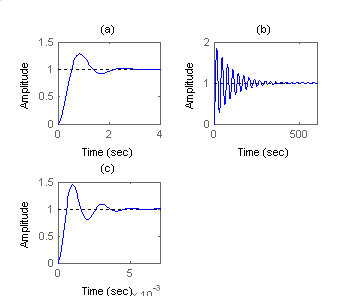
0.0010

Trc =

3.8810e-004

percentc =

44.9154



**15.**

**Program:**

T1=tf(16,[1 3 16])

T2=tf(0.04,[1 0.02 0.04])

T3=tf(1.05e7,[1 1.6e3 1.05e7])

ltiview

**Computer response:**

Transfer function:

16

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

s^2 + 3 s + 16

Transfer function:

0.04

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

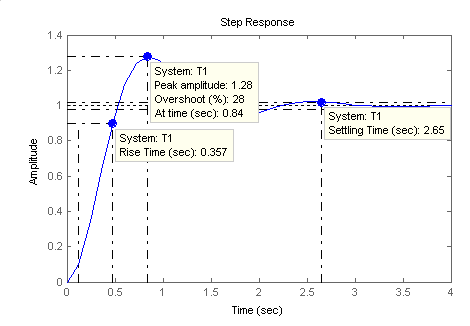
s^2 + 0.02 s + 0.04

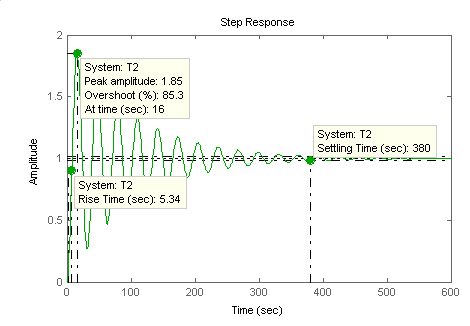
Transfer function:

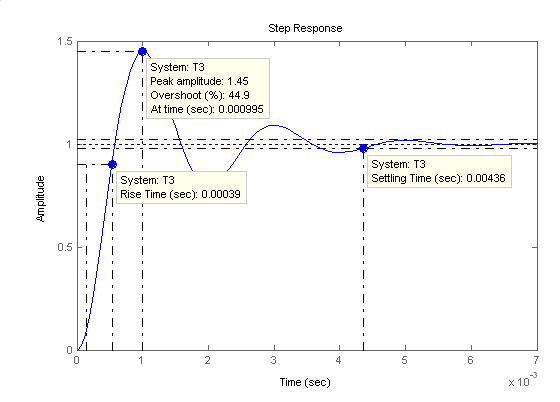
1.05e007

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

s^2 + 1600 s + 1.05e007







**16.**

**a.**  = = 0.517, n = = 15.474. Therefore, poles = -n ± jn   
 = -8 ± j13.246.

**b.**  = = 0.627, n = = 0.4031.

Therefore, poles = -n ± jn = -0.2527 ± j0.314.

**c.** n = = 4, n= = 2.856. Therefore, poles = -4 ± j2.856.

**17.**

**a.** Adding impedances . So the transfer function is .

**b.** Follows that , or , so %OS = . sec. sec. To obtain the rise time Figure 4.16 is used , or sec. The dc gain of the system is .

**18.**

**a**. The impedance equations are:

|  |
| --- |
|  |
|  |

Solving for

So

**b.**  , or . sec. sec and %OS = .

**19.**

**a.** Applying the input to the transfer function, and using a partial fraction expansion

Substituting we have

To equate coefficients the above equation is multiplied on both sides by so

from which it is readily apparent that . So

The inverse Laplace transform of this equation is

**b.** The settling time will be achieved when or or

**c.**

>> x=linspace(0,7,2000);

>> y=exp(-x).\*(1+x);

>> plot(x,y)



So, . We have that .

**20.**

Since the third pole is more than five times the real part of the dominant pole, s2+2.4s+9 determines the transient response.

Since 2n = 2.4, and n = = 3 →  = 0.4,  
, Ts = = 3.33 sec,

Tp =  = 1.143 sec;

From Figure , so sec. For a unit step input .

**21.**

**a.** Measuring the time constant from the graph, T = 0.0244 seconds.



Estimating a first-order system, G(s) = . But, *a* = 1/T = 40.984, and = 2. Hence, K = 81.967. Thus,

G(s) =

**b.** Measuring the percent overshoot and settling time from the graph: %OS = (13.82-11.03)/11.03 = 25.3%,



and Ts = 2.62 seconds. Estimating a second-order system, we use Eq. (4.39) to find  = 0.4 , and Eq. (4.42) to find n = 3.82. Thus, G(s) = . Since Cfinal = 11.03, = 11.03. Hence, K = 160.95. Substituting all values,

G(s) =

**c.** From the graph, %OS = 40%. Using Eq. (4.39),  = 0.28. Also from the graph,  Substituting  = 0.28, we find n = 0.818.

Thus,

G(s) = =.

**(c) C3(s) = ; (d) C4(s) =**

**22.**

**a.** 

The amplitude of residue of the pole at -2 is larger than the amplitude of the sinusoid, so a pole-zero cancellation cannot be assumed.

**b.** 

Since the amplitudes of the sinusoids are of the same order of magnitude as the residue of the pole at -2, pole-zero cancellation cannot be assumed.

**c.**

****

Since the amplitudes of the sinusoids much larger than the residue of the pole at -4.5, a pole-zero cancellation can be assumed. Since 2n = 2, and n = = 4.47211,  = 0.224,  
, Ts = = 4 sec, Tp = = 0.72 sec; nTr = 1.23, therefore, Tr = 0.275 sec.

**d.**

****

Since the amplitude of the sinusoids are several orders of magnitude larger than the residue of the pole at -4.9, a pole-zero cancellation can be assumed. Since 2n = 5, and n = = 3.7081,  = 0.6742,  
, Ts = = 1.6 sec, Tp = = 1.1471 sec; nTr = 2.0503, therefore, Tr = 0.553 sec.

**23.**

**Program:**

%Form sC(s) to get transfer function

clf

num=[1 3];

den=conv([1 3 10],[1 2]);

T=tf(num,den)

step(T)

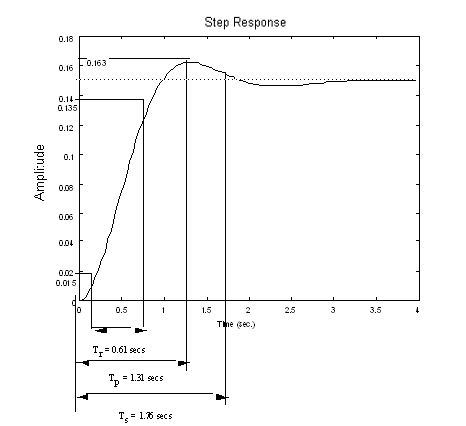
**Computer response:**

Transfer function:

s + 3

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

s^3 + 5 s^2 + 16 s + 20



%OS = = 8.67%

**24.**

Only part **c** can be approximated as a second-order system. From the exponentially decaying cosine the poles are located at . Thus,



Also,  and . Hence, , yielding 52.66 percent overshoot.

**25.**

and . Solving for the roots the two poles are -1.5±j3.428.

**26.**

**a.**

**b.** Solving gives poles at 4.2242 and -0.1121±j0.967

**27.**

***x*** *= (s****I*** *-* ***A*** *)* -1 *(****x0*** + ***B*** *u )*









**28.**

***x*** *= (s****I*** *-* ***A*** *)* -1 *(****x0*** + ***B*** *u )*







**29.**

Obtaining the inverse Laplace transform

**30.**









**31.**

**Program:**

A=[-3 1 0;0 -6 1;0 0 -5];

B=[0;1;1];

C=[0 1 1];

D=0;

S=ss(A,B,C,D)

step(S)

**Computer response:**

a =

x1 x2 x3

x1 -3 1 0

x2 0 -6 1

x3 0 0 -5

b =

u1

x1 0

x2 1

x3 1

c =

x1 x2 x3

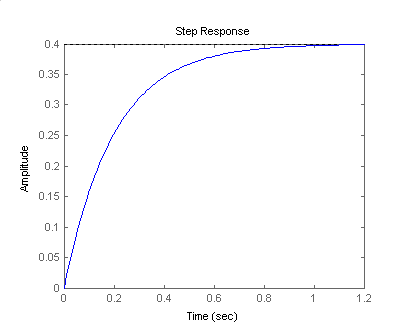
y1 0 1 1

d =

u1

y1 0

Continuous-time model.



**32.**

**Program:**

syms s %Construct symbolic object for

%frequency variable 's'.

'a' %Display label

A=[-3 1 0;0 -6 1;0 0 -5] %Create matrix A.

B=[0;1;1]; %Create vector B.

C=[0 1 1]; %Create C vector

X0=[1;1;0] %Create initial condition vector,X(0).

U=1/s; %Create U(s).

I=[1 0 0;0 1 0;0 0 1]; %Create identity matrix.

X=((s\*I-A)^-1)\*(X0+B\*U); %Find Laplace transform of state vector.

x1=ilaplace(X(1)) %Solve for X1(t).

x2=ilaplace(X(2)) %Solve for X2(t).

x3=ilaplace(X(3)) %Solve for X3(t).

y=C\*[x1;x2;x3] %Solve for output, y(t).

y=simplify(y) %Simplify y(t).

'y(t)' %Display label.

pretty(y) %Pretty print y(t).

**Computer response:**

ans =

a

A =

-3 1 0

0 -6 1

0 0 -5

X0 =

1

1

0

x1 =

7/6\*exp(-3\*t)-1/3\*exp(-6\*t)+1/15+1/10\*exp(-5\*t)

x2 =

exp(-6\*t)+1/5-1/5\*exp(-5\*t)

x3 =

1/5-1/5\*exp(-5\*t)

y =

2/5+exp(-6\*t)-2/5\*exp(-5\*t)

y =

2/5+exp(-6\*t)-2/5\*exp(-5\*t)

ans =

y(t)

2/5 + exp(-6 t) - 2/5 exp(-5 t)

**33.**

|**I** - **A** | = 2 + 5 +1

|**I** - **A** | = ( + 0.20871) ( + 4.7913)













Therefore,





Solving for Ai's two at a time, and substituting into the state-transition matrix



To find **x**(t),







To find the output,







**34.**

|**I** - **A** | = ( + 2) ( + 0.5 - 2.3979i) ( + 0.5 + 2.3979i)

Let the state-transition matrix be



Since (0) = **I**, = **A**, and = **A**2, we can evaluate the coefficients, Ai's. Thus,







Solving for the Ai's taking three equations at a time,







= - e-2t

**35.**

**Program:**

syms s t tau %Construct symbolic object for

%frequency variable 's', 't', and 'tau.

'a' %Display label.

A=[-2 1 0;0 0 1;0 -6 -1] %Create matrix A.

B=[1;0;0] %Create vector B.

C=[1 0 0] %Create vector C.

X0=[1;1;0] %Create initial condition vector,X(0).

I=[1 0 0;0 1 0;0 0 1]; %Create identity matrix.

'E=(s\*I-A)^-1' %Display label.

E=((s\*I-A)^-1) %Find Laplace transform of state

%transition matrix, (sI-A)^-1.

Fi11=ilaplace(E(1,1)); %Take inverse Laplace transform

Fi12=ilaplace(E(1,2)); %of each element

Fi13=ilaplace(E(1,3));

Fi21=ilaplace(E(2,1));

Fi22=ilaplace(E(2,2));

Fi23=ilaplace(E(2,3));

Fi31=ilaplace(E(3,1));

Fi32=ilaplace(E(3,2)); %to find state transition matrix.

Fi33=ilaplace(E(3,3)); %of (sI-A)^-1.

'Fi(t)' %Display label.

Fi=[Fi11 Fi12 Fi13 %Form Fi(t).

Fi21 Fi22 Fi23

Fi31 Fi32 Fi33];

pretty(Fi) %Pretty print state transition matrix, Fi.

Fitmtau=subs(Fi,t,t-tau); %Form Fi(t-tau).

'Fi(t-tau)' %Display label.

pretty(Fitmtau) %Pretty print Fi(t-tau).

x=Fi\*X0+int(Fitmtau\*B\*1,tau,0,t);

%Solve for x(t).

x=simple(x); %Collect terms.

x=simplify(x); %Simplify x(t).

x=vpa(x,3);

'x(t)' %Display label.

pretty(x) %Pretty print x(t).

y=C\*x; %Find y(t)

y=simplify(y);

y=vpa(simple(y),3);

y=collect(y);

'y(t)'

pretty(y) %Pretty print y(t).

**Computer response:**

ans =

a

A =

-2 1 0

0 0 1

0 -6 -1

B =

1

0

0

C =

1 0 0

X0 =

1

1

0

ans =

E=(s\*I-A)^-1

E =

[ 1/(s+2), (s+1)/(s+2)/(s^2+s+6), 1/(s+2)/(s^2+s+6)]

[ 0, (s+1)/(s^2+s+6), 1/(s^2+s+6)]

[ 0, -6/(s^2+s+6), s/(s^2+s+6)]

ans =

Fi(t)

[ 13

[exp(-2 t) , - 1/8 exp(-2 t) + 1/8 %1 + --- %2 ,

[ 184

]

1/8 exp(-2 t) - 1/8 %1 + 3/184 %2]

]

[

[0 , 1/23 %2 + %1 , - 1/23

1/2 1/2 1/2

(-23) (exp((-1/2 + 1/2 (-23) ) t) - exp((-1/2 - 1/2 (-23) ) t))

]

]

[

[0 , 6/23

1/2 1/2 1/2

(-23) (exp((-1/2 + 1/2 (-23) ) t) - exp((-1/2 - 1/2 (-23) ) t))

]

, - 1/23 %2 + %1]

1/2

%1 := exp(- 1/2 t) cos(1/2 23 t)

1/2 1/2

%2 := exp(- 1/2 t) 23 sin(1/2 23 t)

ans =

Fi(t-tau)

[

[exp(-2 t + 2 tau) ,

[

13 1/2

- 1/8 exp(-2 t + 2 tau) + 1/8 %2 cos(%1) + --- %2 23 sin(%1) ,

184

1/2 ]

1/8 exp(-2 t + 2 tau) - 1/8 %2 cos(%1) + 3/184 %2 23 sin(%1)]

]

[ 1/2 1/2

[0 , 1/23 %2 23 sin(%1) + %2 cos(%1) , - 1/23 (-23) (

1/2

exp((-1/2 + 1/2 (-23) ) (t - tau))

1/2 ]

- exp((-1/2 - 1/2 (-23) ) (t - tau)))]

[ 1/2 1/2

[0 , 6/23 (-23) (exp((-1/2 + 1/2 (-23) ) (t - tau))

1/2

- exp((-1/2 - 1/2 (-23) ) (t - tau))) ,

1/2 ]

- 1/23 %2 23 sin(%1) + %2 cos(%1)]

1/2

%1 := 1/2 23 (t - tau)

%2 := exp(- 1/2 t + 1/2 tau)

ans =

x(t)

[.375 exp(-2. t) + .125 exp(-.500 t) cos(2.40 t)

+ .339 exp(-.500 t) sin(2.40 t) + .500]

[.209 exp(-.500 t) sin(2.40 t) + exp(-.500 t) cos(2.40 t)]

[1.25 i (exp((-.500 + 2.40 i) t) - 1. exp((-.500 - 2.40 i) t))]

ans =

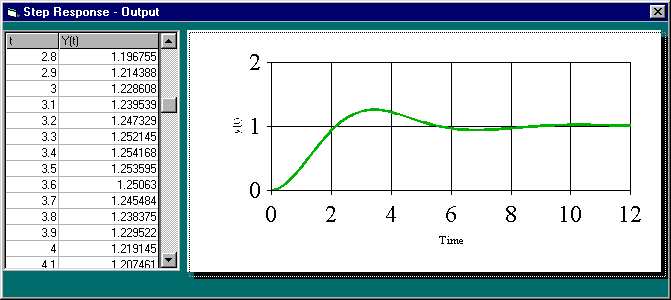
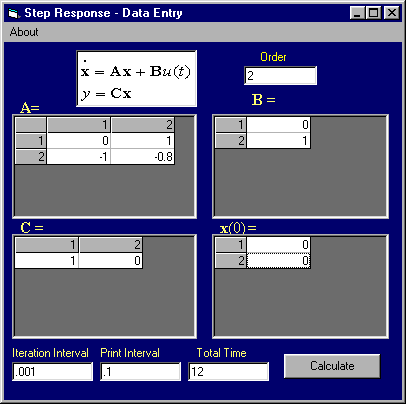
y(t)

.375 exp(-2. t) + .125 exp(-.500 t) cos(2.40 t)

+ .339 exp(-.500 t) sin(2.40 t) + .500

**36.**

The state-space representation used to obtain the plot is,  Using the Step Response software,

 Calculating % overshoot, settling time, and peak time,

2n = 0.8, n = 1,  = 0.4. Therefore, , Ts = = 10 sec,

Tp = = 3.43 sec.

**37.**

**a.** P(s) = = + - . Therefore, p(t) = + e-2t - e-5t.

**b.** To represent the system in state space, draw the following block diagram.



For the first block,



Let x1 = y, and x2 = . Therefore,

1 = x2

2 = -10x1 - 7x2 + v(t)

Also,

p(t) = 0.5y + = 0.5x1 + x2

Thus,



**c.**

**Program:**

A=[0 1;-10 -7];

B=[0;1];

C=[.5 1];

D=0;

S=ss(A,B,C,D)

step(S)

**Computer response:**

a =

x1 x2

x1 0 1

x2 -10 -7

b =

u1

x1 0

x2 1

c =

x1 x2

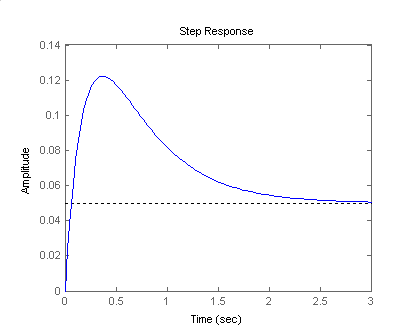
y1 0.5 1

d =

u1

y1 0

Continuous-time model.



**rely on part (c) in general? If your results do not match the plot, explain why not?**

**35. (a)**

****

****

**Therefore, u(t) = - 2.6158 - 0.06454e-1.23t + e-0.113t (2.6804 cos 0.0643t + 3.4776 sin 0.0643)  
= - 2.6158 - 0.06454e-1.23t + 4.39 e-0.113t cos (0.0643t - 52.38o).**

**(b)**

****

**383838.**

Consider the un-shifted Laplace transform of the output





















thus



Obtaining the inverse Laplace transform of the latter and delaying the equation in time domain we get



**39.**

**a.** The transfer function can be written as

****

It has poles at s=-0.36±j1.145 and s=-1. A zero at s=-3.33

The ‘far away’ pole at -1 is relatively close to the complex conjugate poles as 0.36\*5>1 so a dominant pole approximation can’t be applied.

b) In time domain the input can be expressed as:



Obtaining Laplace transforms this can be expressed as



We first obtain the response to an unshifted unit step:

****

****

****

We will get C and D by equating coefficients. Substituting these two values and multiplying both sides by the denominator we get.







We immediately get C=-2.4 and D=-5.128

So

****

Obtaining inverse Laplace transform we get





So the actual (shifted) unit step response is given by



The response to the pulse is given by:





**40.**

The oscillation period is

 and from the figure 

Thus T=0.0338sec from which we get 

The peaks of the response occur when the ‘cos’ term of the step response is ±1 thus from the figure we have:

 and 

From which we get

 or  or 

Substituting this result we get 

or  or or 

Finally 

**41.**

The step input amplitude is the same for both responses so it will just be assumed to be unitary.

For the ‘control’ response we have:

, from which we get







Leading a transfer function



Similarly for the ‘hot tail’:

, 









Using MATLAB:

>> syms s

>> s=tf('s')

Transfer function:

s

>> Gc = 1108.89/(s^2+22\*s+1108.89);

>> Gh = 1167.6/(s^2+26.9\*s+1167.6);

>> step(Gc,Gh)



Both responses are equivalent if error tolerances are considered.

**42.**

The original transfer function has zeros at 

And poles at ; 

With 

The dominant poles are those with real parts at -120, so a real pole is added at

-1200 giving the following approximation:





Using MATLAB:

>> syms s

>> s=tf('s');

>>G=9.7e4\*(s^2-14400\*s+106.6e6)…

/(s^2+3800\*s+23.86e6)/(s^2+240\*s+2324.8e3);

>> Gdp=4.8782\*(s^2-14400\*s+106.6e6)/(s^2+240\*s+2324.8e3)/(s+1200);

>> step(G,Gdp)



Both responses differ because the original non-dominant poles are very close to the complex pair of zeros.

**43.**

M(s) requires at least 4 ‘far away’ poles that are added a decade beyond all original poles and zeros. This gives



**44.**





****

**45.**

**a.**  Let the impulse response of T(s) be h(t). We have that



; 

. Obtaining the inverse Laplace transform we get



1. Let the step response of the system be g(t). We have that





**c.**  

; ; 

Leading . After the inverse Laplace we get



**46.**

1. The poles given by  have an and 

The poles given by  have an and  Thus the former represent the Phugoid and the latter the Short Period modes.

1. In the original we have  so the Phugoid approximation is given by:



**c.**

>> syms s

>> s=tf('s');

>>G=-26.12\*(s+0.0098)\*(s+1.371)/(s^2+8.99e-3\*s+3.97e-3)/(s^2+4.21\*s+18.23);

>> Gphug=-1.965\*(s+0.0098)/(s^2+8.99e-3\*s+3.97e-3);

>> step(G,Gphug)



Both responses are indistinguishable.

**47.**

a.

Program

numg=[33 202 10061 24332 170704];

deng=[1 8 464 2411 52899 167829 913599 1076555];

G=tf(numg,deng)

[K,p,k]=residue(numg,deng)

**Computer Response**

K =

0.0018 + 0.0020i

0.0018 - 0.0020i

-0.1155 - 0.0062i

-0.1155 + 0.0062i

0.0077 - 0.0108i

0.0077 + 0.0108i

0.2119

p =

-1.6971 +16.4799i

-1.6971 -16.4799i

-0.5992 +12.1443i

-0.5992 -12.1443i

-1.0117 + 4.2600i

-1.0117 - 4.2600i

-1.3839

k =

[]

**b.**

Therefore, an approximation to *G(s)/* is:



**c.**

**Program**

numg=[33 202 10061 24332 170704];

deng=[1 8 464 2411 52899 167829 913599 1076555];

G=tf(numg,deng);

numga=0.2119;

denga=[1 1.3839];

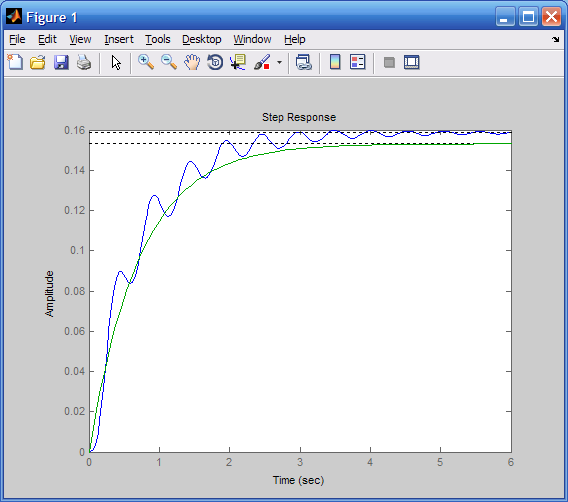
Ga=tf(numga,denga);

step(G)

hold on

step(Ga)

**Computer Response**

****

Approximation does not show oscillations and is slightly off of final value.

**48.**

**Computer Response**

Transfer function:

s^15 + 1775 s^14 + 1.104e006 s^13 + 2.756e008 s^12 + 2.272e010 s^11

+ 7.933e011 s^10 + 1.182e013 s^9 + 6.046e013 s^8 + 1.322e014 s^7

+ 1.238e014 s^6 + 3.977e013 s^5 + 5.448e012 s^4 + 3.165e011 s^3

+ 6.069e009 s^2 + 4.666e007 s + 1.259e005

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

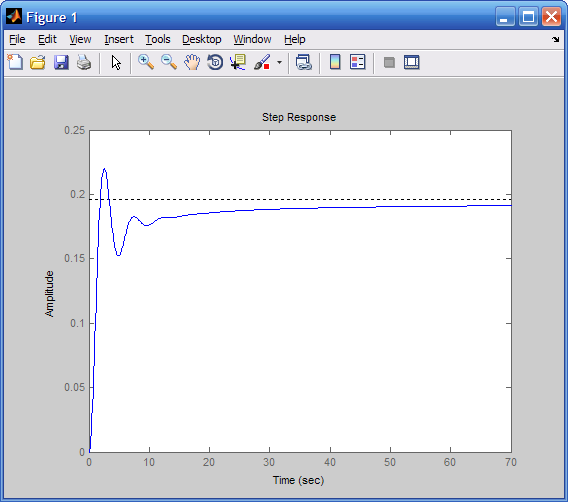
31.62 s^17 + 4.397e004 s^16 + 1.929e007 s^15 + 2.941e009 s^14

+ 1.768e011 s^13 + 4.642e012 s^12 + 5.318e013 s^11 + 2.784e014 s^10

+ 7.557e014 s^9 + 1.238e015 s^8 + 1.356e015 s^7 + 8.985e014 s^6

+ 2.523e014 s^5 + 3.179e013 s^4 + 1.732e012 s^3 + 3.225e010 s^2

+ 2.425e008 s + 6.414e005



**49.**

1. To find the step responses for these two processes, *ya(t)* and *yp(t)*, we consider first the un-shifted Laplace transform of their outputs for *Xa(s)* = *Xp(s)* = 1/s:

 (1),

where  and (2)

Substituting the values of *A* and *B* into equation (1) gives:

 (3)

Taking the inverse Laplace transform of ** and delaying the resulting response in the time domain by 3.3 seconds, we get:

 (4)

Noting that the denominator of ** can be factored into , we have:

 (5),

where: ;

;

. (6)

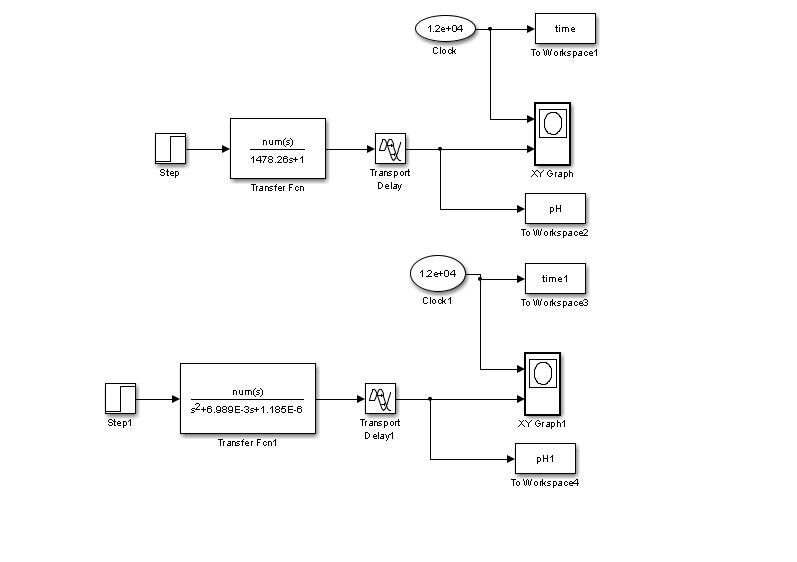
Substituting the values of *C*, *D* and *E* into equation (5) and simplifying gives:

 (7)

Taking the inverse Laplace transform of ** and delaying the resulting response in the time domain by 25 seconds, we get:

 (8)

1. Using Simulink to model the two processes described above, *ya(t)* and *yp(t)* were output to the “workspace.” MATLAB plot commands were then utilized to plot *ya(t)* and *yp(t)* on a single graph. The model and graph are shown below.





**50.**

1. We have  and



We first find 



;  so



Follows that





And



Finally the state transition matrix is given by:



**b.**





Since 









**c.**

>> A=[-8.34 -2.26; 1 0];

>> B = [1; 0];

>> C = [12.54 2.26];

>> D = 0;

>> t = linspace(0,15,1000);

>> y1 = step(A,B,C,D,1,t);

>> y2 = 1.014+0.568\*exp(-0.28.\*t)-1.582\*exp(-8.06.\*t);

>> plot(t,y1,t,y2)



**51.**

From the problem statement

 (1)

 (2)

 (3)

Substituting from equations (2) and (3) into (1) gives:

(4)

The Laplace transforms of the inputs are:  and  (5)

First, we find the response to , *ωc1(t)*:

 (5) →  (6)

Next we find the response the un-shifted step in torque: , *ωc2(t)*:

 (7) →  (8)

So the actual response to a shifted unit step in torque is given by:

(9)

Therefore, the final analytical expression for *ωc*(t)is:

 (10)

**52.**

From the problem statement

 (1)

Substituting the parameters given in the problem into equation (1) yields:

 (2)

or  (3)

Since *Tem*(t) = 50 u(t), the Laplace transform of the shaft speed at no-load is:

 (4)

With a load torque, *TL* = 0.2 *ω**(t)*, N.m., the speed, *ΩL*(s), is given by:

 (5)

Solving equation (5) for *ΩL*(s), substituting for *G(s)* from (3), and simplifying, we have:

 (6)

The MATLAB M-file is:

num1=1478.88;

num2=[1 1.1655 13209];

den1=[1 0];

den2=[1 5.6013 63481.7];

num=conv(num1, num2);

den=conv(den1, den2);

Omega\_nl=tf(num, den);

step(Omega\_nl, 0:0.01:3);

hold on;

numL=num;

denL=[1 11.52 63489 78139];

Omega\_L=tf(numL, denL);

step(Omega\_L, 0:0.01:3);

hold off;

grid;

The MATLAB figure, shown below, illustrates the two step-responses obtained: the blue curve corresponds to the *ωn.l.*, while the green one shows that the angular speed drops markedly when a load torque, *TL* = 0.2 *ω (t)*, N.m., is applied (The steady-state value of *ωL*s.s. = 250 rad/second.).



**53.**

From the problem in chapter 3

; 

; D = 0.

The MATLAB M-file is:

A=[0 1 0 0;29.8615 0 0 0;0 0 0 1;-0.9401 0 0 0];

B=[0;-1.1574;0; 0.4167];

C=[0.36 0 1 0];

D=0;

S=ss(A, B, C, D);

impulseplot(S,0:0.1:11.0);

The impulse response, *x*G(t), is shown below:



Note: As would be clear in chapter 6, such a response to a unit impulse indicates that this system is unstable and needs to be stabilized.

**SOLUTIONS TO DESIGN PROBLEMS**

**54.**

The transfer function is, . Now, . Thus, . Substituting the value of M in the denominator of the transfer function yields, . Identify the roots . Using the imaginary part and substituting into the peak time equation yields , from which .

**55.**

Writing the equation of motion, . Thus, the transfer function is

. Since . But, Also, from Eq. (4.39) 17% overshoot implies * =* 0.491. Hence, *n* = 0.815. Now, 1/M = *n*2 = 0.664. Therefore, M 1.51. Since .

**15. Find J and K in the rotational system shown in Figure P.4.7 to yield a 30% overshoot and a settling time of 4 seconds for a step input in torque.**

****

Figure P.4.7

16. •312-Sp-76-M.3 (Chap 4)

**16. Given the system shown in Figure P.4.8. Find the damping, D, to yield a 30% overshoot in output angular displacement for a step input in torque.**

****

Figure P.4.8

**56.**

Writing the equation of motion

[s2+D(5)2s+2](s) = T(s)

The transfer function is

=

Also,

 = = 0.358

and

2n = 2(0.358)(5) = 25D

Therefore D = 0.14.

**57.**

Let the rotation of the shaft with gear *N*2 be *L(s).*  Assuming that all rotating load has been reflected to the *N*2 shaft, , where F(s) is the force from the translational system, *r* = 2 is the radius of the rotational member, *J*eqL is the equivalent inertia at the *N*2 shaft, and *D*eqL is the equivalent damping at the *N*2 shaft. Since *J*eqL = 1(2)2 + 1 = 5 and *D*eqL = 1(2)2 = 4, the equation of motion becomes, . For the translational system . Substituting *F(s)* into the rotational equation of motion, . But, Substituting these quantities in the equation above yields .

Thus, the transfer function is .

Now, . Hence, *M* = 15/4.

For 16% overshoot,  = 0.504 from Eq. (4.39).

Therefore, . Solving for *n* yields *n* = 0.3968.

But, . Thus, *K =* 3.15.

**58.**

The transfer function for the capacitor voltage is = = .

For 20% overshoot,  = = 0.456. Therefore, 2n = R = 2(0.456)(103) = 912.

**59.**

Solving for the capacitor voltage using voltage division, . Thus, the transfer function is . Since . Thus . Also, since 20% overshoot implies a damping ratio of 0.46 and . Hence, .

**60.**

Using voltage division the transfer function is,



Also,. Thus, . Using Eq. (4.39) with 15% overshoot, ** = 0.5169. But, 2*n* = *R/L.* Thus, . Therefore, L = 81.8 mH and R = 98.5 .

**61.**

1. In Problem 3.31 we had





When the equations are equivalent to





Substituting parameter values one gets





**b.**





To obtain the adjoint matrix we calculate the cofactors:



















Then we have



Finally



1. 100% effectiveness means that  or , so by the final value theorem

 (virus copies per mL of plasma)

The closest poles to the imaginary axis are  so the approximate settling time will be days.

**62.**

**a.**

Substituting  into the transfer function and solving for ΔV(s) gives:

**

Here:  and 

Substituting we have:

**

Taking the inverse Laplace transform, we have:



**b.**

>> s=tf('s');

>> G=1/(1908\*s+10);

>> t=0:0.1:1000;

>> y1=2650\*step(G,t);

>> y2=265\*(1-exp(-5.24e-3.\*t));

>> plot(t,y1,t,y2)

>> xlabel('sec')

>> ylabel('m/s')



Both plots are identical.

**63.**

Since no overshoot is observed in the response, for simplicity, we postulate a first order system with some delay, namely .

The steady state change in temperature is of 8°C, so the transfer function’s dc gain is -8 since the step is negative. The time delay observed is approximately 0.017h=60sec. To find the systems time constant we find the point at which the temperature reaches 0.63(8°C)=5.04°C.

This happens approximately 0.02hours = 75 seconds after the observed response delay. So the approximate transfer function is .