Contents

- MATLAB'da doğrusal sistem modelleri
- Transfer fonksiyonu gösterimi
- Transfer fonksiyonu ile ilgili bazı faydalı komutlar
- Transfer fonksiyonunun sıfırlar, kutuplar ve kazanç (zpk) olarak ifadesi
- Standart transfer fonksiyonu ile sıfır-kutup-kazanç gösterimi arasında geçiş
- Durum Uzayı Gösterimi
- Durum uzayı gösteriminin tek olmayışı ve eşdeğerlik dönüşümleri
- Durum uzayı gösterimi ile diğer gösterimler arasında geçiş
- Sistem tepkisinin grafiksel analizi
- Birim dürtü cevabı ve birim basamak cevapları
- Sistemin genel bir girişe verdiği cevap
- Durum uzayı gösterimindeki sistemler için cevaplar
- Sistemin frekans cevabı ile ilgili çizimler
- Doğrusal sistemlerin analizi için grafiksel arayüz
- Kök-yer eğrisi
- Kapalı-çevrim sistemi oluşturma
- Kapalı çevrim transfer fonksiyonunu oluşturulması ile ilgili bir not
- Çok-giriş, çok-çıkışlı sistem tanımlanması

MATLAB'da doğrusal sistem modelleri

MATLAB'dan doğrusal sistemlerin matematiksel modellemesi için transfer fonksiyonu, sıfır-kutup-kazanç, durum uzayı vs. gösterimler kullanılabilir.

Transfer fonksiyonu gösterimi

Hatırlanacağı üzere transfer fonksyionu tek-giriş tek-çıkış bir sistemin Laplace alanındaki giriş çıkış ilişkisini verir:

$$G(s) = \frac{Y(s)}{U(s)}$$

Burada $U(s)$ ve $Y(s)$ giriş ve çıkış Laplace dönüşümüdür.

MATLAB'da transfer fonksiyonu oluşturmak için **tf** komutu kullanılır. Örneğin

$$G(s) = \frac{2s+6}{s^3+5s^2+9s+5}$$

transfer fonksiyonunu oluşturalım:

```
num = [2 6]; % Pay polinomunun katsayıları
den = [1 5 9 5]; % Payda polinomunun katsayıları

G = tf(num,den) % Transfer fonksiyonunu oluştur
```

```
Transfer function:
       2 s + 6
---------------------
s^3 + 5 s^2 + 9 s + 5
```

Yüksek dereceli sistemlerde transfer fonksiyonunu yukarıdaki gibi oluşturmak kafa karıştırıcı olabilir çünkü num ve den oluşturmak için kullanılan komutlarda hangi katsayının s'nin hangi kuvvetine karşılık geldiği hemen görülmüyor.

Bunun yerine aşağıdaki yöntemi kullanarak daha doğal ve anlaşılır biçimde transfer fonksiyonu oluşturabiliriz:

```
s = tf('s') % Laplace değişkenini oluştur

G = (2*s+6)/(s^3+5*s^2+9*s+5) % Transfer fonksiyonunu oluştur
```

```
Transfer function:
s


Transfer function:
       2 s + 6
--------------------
s^3 + 5 s^2 + 9 s + 5
```

Transfer fonksiyonu ile ilgili bazı faydalı komutlar

İstenildiğinde **tfdata** komutu ile transfer fonksiyonun içinden payı ve paydası alınabilir

```
[num, den] = tfdata(G)
```

```
num =

    [1x4 double]


den =

    [1x4 double]
```

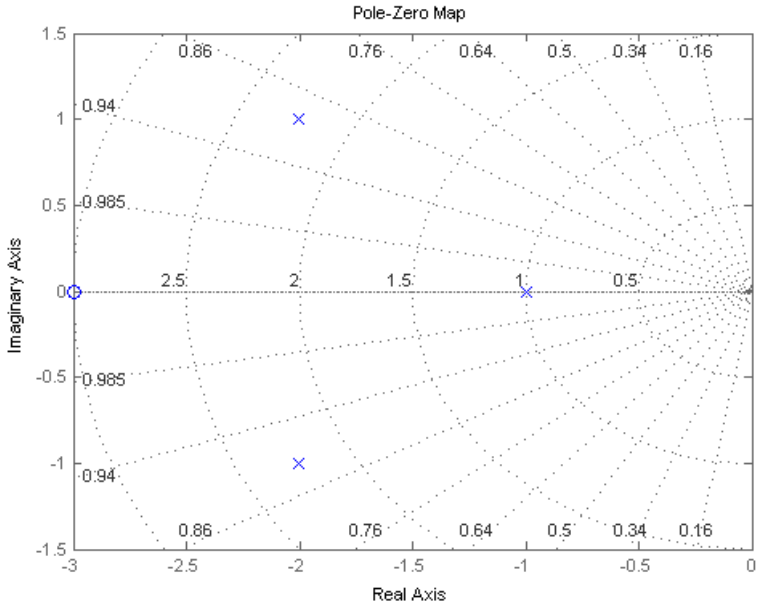
Transfer fonksiyonunun sıfırlarını (payın kökleri) ve kutuplarını (paydanın kökleri) bulmak için **pole**ve **zero** komutları kullanılır:

```
z = zero(G) % Sıfırları bul
p = pole(G) % Kutupları bul
```

```
z =

    -3


p =

  -2.0000 + 1.0000i
  -2.0000 - 1.0000i
  -1.0000
```

pzmap komutu ile sistemin sıfırları ve kutuplarını sana düzlem üzerine işaretletebiliriz:

```
pzmap(G); % Kutup ve sıfırları sanal düzlem üzerine işaretle
grid; % Kılavuz çizgileri
```



Kılavuz çizgilerinin özel bir şekilde çizildiğine dikkat ediniz. Doğrusal çizgiler sabit sönüm (damping) çizgileri olup, dairesel eğriler de sabit doğal frekans (natural frequency) eğrileridir. Bu kavramlar kontrol sistemlerinin analiz ve tasarımında sıkça karşımıza çıkan kavramlardır. İleriki derslerimizde bunlarla ilgili örnekler yapacağız.

Transfer fonksiyonunun sıfırlar, kutuplar ve kazanç (zpk) olarak ifadesi

İstesek transfer fonksiyonunu sıfır, kutup ve kazanç terimlerden oluşan çarpanlar cinsinden de ifade edebiliriz. Oluşturduğumuz transfer fonksiyonlarının bu türden gösterilmesini sağlamak için **zpk**komutu ile oluşturabiliriz. Örneğin yukarıdaki sistemi bu komutla oluşturmak için:

```
z = -3; % Sıfırlar
p = [-1 -2+j -2-j]; % Kutuplar
k = 2; % Kazanç = Pay ve paydadaki en büyük terimlerin oranı

H = zpk(z,p,k) % Transfer fonksiyonunu sıfır-kutup-kazanç
gösteriminde oluştur
```

```
Zero/pole/gain:
      2 (s+3)
--------------------
(s+1) (s^2 + 4s + 5)
```

Yukarıda yaptığımız gibi burada da önce Laplace değişkenini oluşturup, sonra o değişkeni kullanarak da tanımlama yapabiliriz:

```
s = zpk('s');
```

```
H = (2*s+6)/(s^3+5*s^2+9*s+5) % Transfer fonksiyonunu oluştur
```

```
Zero/pole/gain:
      2 (s+3)
--------------------
(s+1) (s^2 + 4s + 5)
```

Yukarıda s'yi tf yerine zpk fonksiyonu ile tanımladığımız için bu değişkeni kullanarak oluşturduğumuz transfer fonksiyonu da otomatik olarak zpk formatında oluşur.

Standart transfer fonksiyonu ile sıfır-kutup-kazanç gösterimi arasında geçiş

İstendiğinde farklı gösterimler arasında kolayca geçiş yapılabilir

```
tf(H) % H'yi standart transfer fonksiyonu formuna çevir
```

```
Transfer function:
       2 s + 6
---------------------
s^3 + 5 s^2 + 9 s + 5
```

Standart formattan da benzer şekilde sıfır-kutup-kazanç formatına çevirebiliriz:

```
zpk(G) % G'yi zpk formatına çevir
```

```
Zero/pole/gain:
      2 (s+3)
--------------------
(s+1) (s^2 + 4s + 5)
```

Durum Uzayı Gösterimi

Doğrusal sistemler için bir alternatif gösterim de aşağıdaki gibi durum uzayı gösterimidir:

$$\frac{dx}{dt}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Sistemi bu gösterimde tanımlamak için **ss** komutu kullanılır:

```
A = [ 0  1 ; -5  -2 ]; % Sistemin A matrisi
B = [ 0 ; 3 ]; % Sistemin B matrisi
C = [ 1  0 ]; % Sistemin C matrisi
D = 0; % Sistemin D matrisi

P = ss(A,B,C,D) %% Sistemin durum uzayı gösterimini oluştur
```

```
a =
       x1  x2
   x1   0   1
```

```
   x2  -5  -2

b =
       u1
   x1   0
   x2   3

c =
       x1  x2
   y1   1   0

d =
       u1
   y1   0

Continuous-time model.
```

Durum uzayı gösteriminin matrislerine ulaşmak için:

```
P.A % A matrisine ulaş
P.B % B matrisine ulaş
P.C % C matrisine ulaş
P.D % D matrisine ulaş
ans =

     0     1
    -5    -2


ans =

     0
     3


ans =

     1     0


ans =

     0
```

Alternatif olarak **ssdata** komutu ile sistemin tüm matrisleri istenilen değişkenlere atanabilir

```
[A,B,C,D] = ssdata(P) % P'nin matrislerini A, B, C ve D
değişkenlerine al
A =

     0     1
    -5    -2
```

```
B =

     0
     3


C =

     1     0


D =

     0
```

Durum uzayı gösteriminin tek olmayışı ve eşdeğerlik dönüşümleri

Durum uzayı gösteriminin tek olmadığını hatırlayınız. Yani aynı sistemi temsil eden pek çok durum uzayı gösterimi bulunabilir. Verilen bir gösterimi kullanarak, aynı sistemi temsil eden başka bir durum uzayı gösterimine geçmek için terslenebilir bir T dönüşüm matrisi kullanabiliriz. Örneğin

```
T = [1 2;2 -1] % Bir dönüşüm matrisi tanımlayalım
T =

     1     2
     2    -1
```

```
Ti = T^-1 % Matrisin tersi

% Dönüşüm işlemleri
A2 = T*A*Ti;
B2 = T*B;
C2 = C*Ti;
D2 = D;

P2 = ss(A2,B2,C2,D2) % Eşdeğer sistemi oluştur
Ti =

    0.2000    0.4000
    0.4000   -0.2000


a =
         x1    x2
   x1  -3.2  -3.4
   x2   2.6   1.2

b =
       u1
   x1   6
   x2  -3

c =
```

```
         x1    x2
   y1   0.2   0.4

d =
       u1
   y1   0

Continuous-time model.
```

Yukarıdaki dönüşümü tek komutla yapmak için **ss2ss** kullanabiliriz.

```
P2 = ss2ss(P,T)

a =
         x1    x2
   x1  -3.2  -3.4
   x2   2.6   1.2

b =
       u1
   x1   6
   x2  -3

c =
         x1   x2
   y1   0.2  0.4

d =
       u1
   y1   0

Continuous-time model.
```

Dönüşümün tersi de geçerlidir, yani P2'den de P'yi aşağıdaki gibi geçebiliriz:

```
P = ss2ss(P2,Ti)

a =
       x1  x2
   x1   0   1
   x2  -5  -2

b =
       u1
   x1   0
   x2   3

c =
       x1  x2
   y1   1   0

d =
       u1
   y1   0
```

```
Continuous-time model.
```

Yukarıdaki işlemin tersi de geçerlidir: Eğer iki sistemin eşdeğer olduğunu (yani aynı sistemi temsil ettiğini) biliyorsak, aralarında mutlaka yukarıdaki gibi bir T dönüşüm matrisi bulmak mümkündür.

Durum uzayı gösterimi ile diğer gösterimler arasında geçiş

Durum uzayı gösterimden de diğer gösterimlere karşılıklı olarak geçiş yapılabilir:

```
P2 = tf(P) % P'yi transfer fonsksiyonu gösterimine çevir

Transfer function:
      3
-------------
s^2 + 2 s + 5

P3 = zpk(P) % P'yi sıfır-kutup-kazanç gösterimine çevir

Zero/pole/gain:
      3
--------------
(s^2 + 2s + 5)

P4 = ss(P3) % P3'ü tekrar durum uzayı gösterimine çevir

a =
       x1  x2
   x1  -1   2
   x2  -2  -1

b =
          u1
   x1      0
   x2  1.414

c =
          x1     x2
   y1  1.061      0

d =
       u1
   y1   0

Continuous-time model.
```

Yukarıda da olduğu gibi, P3'ü tekrar durum uzayı gösterimine çevirdiğimizde elde ettiğimiz P4 sisteminin durum uzayı matrisleri P'nin matrisleriyle aynı çıkmayabilir. Bu durum normaldir çünkü yukarıda da belirtildiği gibi aynı sistemi temsil eden çok sayıda durum uzayı gösterimi bulmak mümkündür.

Eğer iki durum uzayı sisteminin eşdeğer olduğunu (aynı sistemi ifade ettiğini) doğrulamak istersek**canon** komutu ile gösterimleri doğal biçime çevirip kıyaslayabiliriz:

```
CP = canon(P,'companion') % P'yi doğal biçimde göster
```

```
a =
       x1  x2
   x1   0  -5
   x2   1  -2

b =
       u1
   x1   1
   x2   0

c =
       x1  x2
   y1   0   3

d =
       u1
   y1   0

Continuous-time model.
```

```
CP4 = canon(P4,'companion') % P'yi doğal biçimde göster
```

```
a =
       x1  x2
   x1   0  -5
   x2   1  -2

b =
       u1
   x1   1
   x2   0

c =
       x1  x2
   y1   0   3

d =
       u1
   y1   0

Continuous-time model.
```

Durum uzayı matrisleri incelendiğinde aynı oldukları görülebilir. **canon** komutunu iki çıkış argümanıyla çağırırsak sistemden doğal biçimine olan dönüşüm matrisini verir:

```
[CP, T] = canon(P,'companion')
```

```
a =
       x1  x2
   x1   0  -5
   x2   1  -2

b =
       u1
```

```
   x1   1
   x2   0

c =
       x1  x2
   y1   0   3

d =
       u1
   y1   0

Continuous-time model.

T =

    0.6667    0.3333
    0.3333         0
```

```
[CP4, T4] = canon(P4,'companion')
```

```
a =
       x1  x2
   x1   0  -5
   x2   1  -2

b =
       u1
   x1   1
   x2   0

c =
       x1  x2
   y1   0   3

d =
       u1
   y1   0

Continuous-time model.

T4 =

    0.3536    0.7071
    0.3536         0
```

Bu matrisleri birleştirerek de P ve P4 arasındaki geçiş matrisini bulabiliriz. P'den CP'ye T ile geçiliyor. T'yi bir fonksiyon gibi hayal edelim ve bu işlemi $T(P) = CP$ olarak gösterelim. Benzer şekilde P4'ten CP4 = CP sistemine de T4 ile geçiliyor, yani $T4(P3) = CP$ veya $T4^{-1}(CP) = P4$.

O halde $P4 = T4^{-1}(CP) = T4^{-1} * (T(P))$ yani $P4 = T4^{-1} * T(P)$ yani P'den P4'e geçiş için $T4^{-1} * T$ dönüşümünü kullanabiliriz:

```
ss2ss(P,T4^-1*T)
```

```
a =
       x1  x2
   x1  -1   2
   x2  -2  -1

b =
          u1
   x1      0
   x2  1.414

c =
          x1     x2
   y1  1.061      0

d =
       u1
   y1   0

Continuous-time model.
```

P4 ile kıyaslayalım:

```
P4
```

```
a =
       x1  x2
   x1  -1   2
   x2  -2  -1

b =
          u1
   x1      0
   x2  1.414

c =
          x1     x2
   y1  1.061      0

d =
       u1
   y1   0

Continuous-time model.
```

Sonuçların aynı olduğu açıktır.

Sistem tepkisinin grafiksel analizi

Doğrusal sistemlerin davranışlarını analiz etmek için sıkça kullanılan grafiksel çizimler, MATLAB'da kolaylıkla gerçekleştirilebilir. Bundan sonraki kısımlarda bununla ilgili bazı örnekler göreceğiz. Öncelikle örnek bir transfer fonksyonu oluşturalım:

```
s = tf('s'); % Laplace değişkeni
```

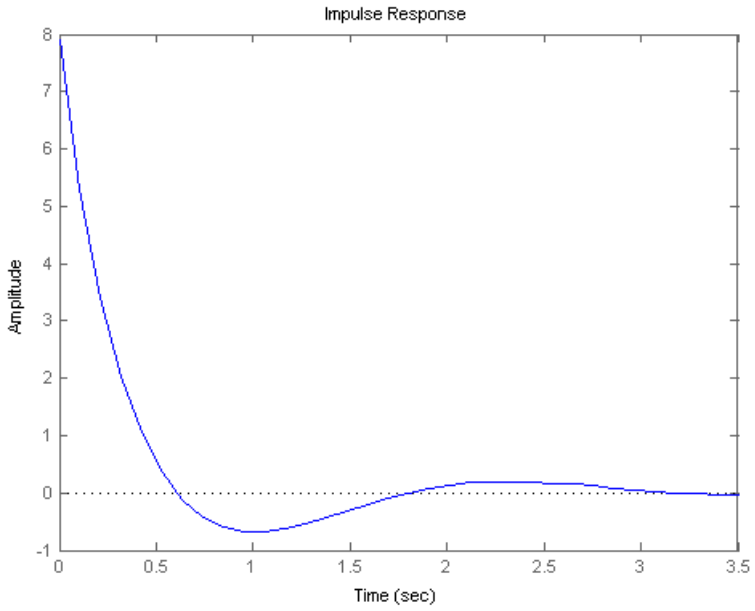
```
G = (8*s^2+18*s+32)/(s^3+6*s^2+14*s+24)
```

```
Transfer function:
   8 s^2 + 18 s + 32
----------------------
s^3 + 6 s^2 + 14 s + 24
```

Birim dürtü cevabı ve birim basamak cevapları

Sistemin dürtü cevabını çizdir:

```
impulse(G);
```



Eğer komutları çıkış argümanlarıyla çağırırsak çizim yapmaz, vermiş olduğumuz çıkış değişkenlerine sistemin tepkisini kaydeder:

```
[y,t] = impulse(G);
```

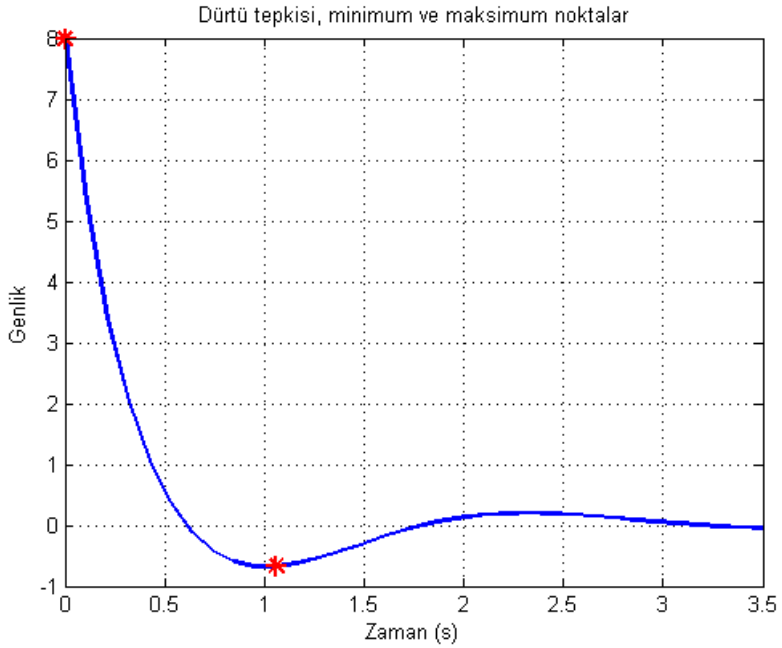
Daha sonra bu değişkenleri kullanarak kendimiz gerekli işlemleri ve çizimleri yapabiliriz. Örneğin, tepkinin minimum ve maksimum değerleri aldığı zaman anlarını bulup, bunları şekil üzerinde gösterelim:

```
[yMin,indMin] = min(y) % y'nin minimumu ve indisi
tMin = t(indMin) % y'nin minimum olduğu zaman anı
[yMax,indMax] = max(y) % y'nin maksimumu ve indisi
tMax = t(indMax) % y'nin maksimum olduğu zaman anı
```

```
yMin =

   -0.6748


indMin =
```

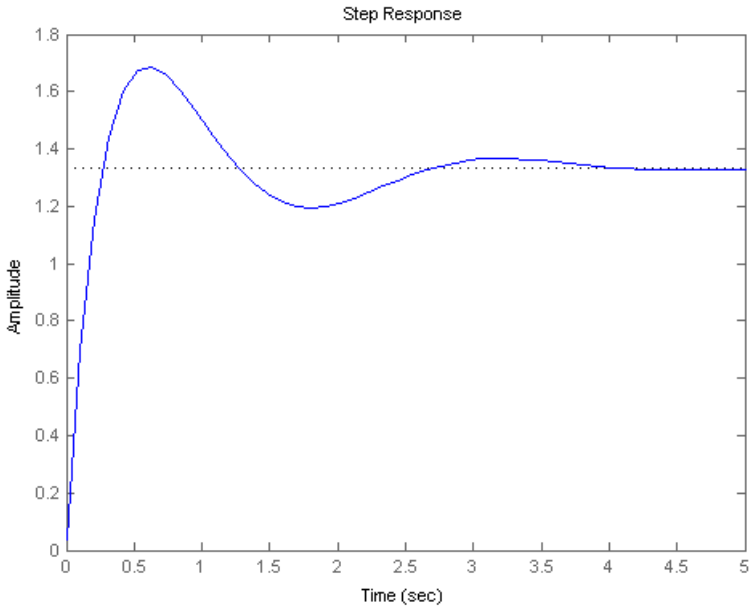
```
     11


tMin =

    1.0597


yMax =

     8


indMax =

     1


tMax =

     0
```

```
plot(t,y,'LineWidth',1.5); % Dürtü tepkisini çizdir;
hold all;
plot(tMin,yMin,'r*',tMax,yMax,'r*','LineWidth',2,'MarkerSize',10);
hold off;
xlabel('Zaman (s)');
ylabel('Genlik');
title('Dürtü tepkisi, minimum ve maksimum noktalar');
grid; % Kılavuz çizgiler
```



Sistemin birim basamak cevabı

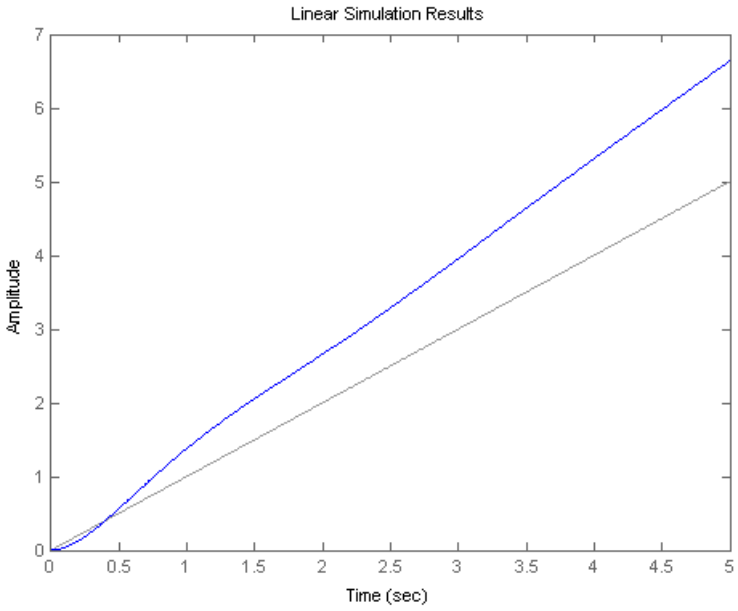
```
step(G);
```



Sistemin genel bir girişe verdiği cevap

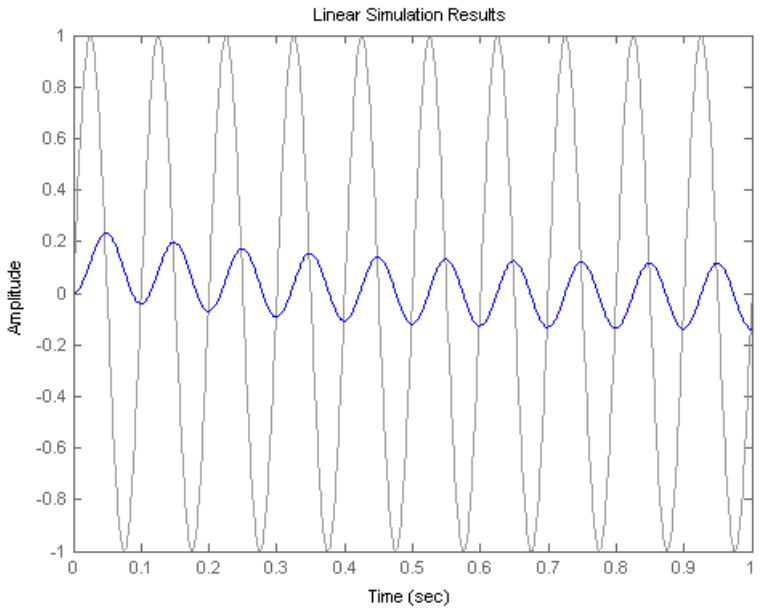
lsim komutu ile sistemin bizim verdiğimiz herhangi bir girişe olan tepkisini hesaplayabiliriz. Örneğin sistemin rampa cevabını bulalım:

```
t = linspace(0,5,1000); % 0'dan 5 saniyeye kadar bir zaman vektörü
oluşturalım.
u = t; % Giriş sinyali
% G sistemine, t zaman vektörü boyunca u sinyalinin girilmesi ile
elde
% edilen çıkışı hesapla ve çizdir:
lsim(G,u,t)
```



Başka bir örnek olarak sistemin bir sinüs sinyaline olan tepkisini bulalım:

```
fu = 10; % Sinüsün frekansı
t = linspace(0,1,1000); % 0'dan 1 saniyeye kadar bir zaman vektörü
oluşturalım.
u = sin(2*pi*fu*t); % Giriş sinyali

% G sistemine, t zaman vektörü boyunca u sinyalinin girilmesi ile
elde
% edilen çıkışı hesapla ve çizdir:
lsim(G,u,t)
```



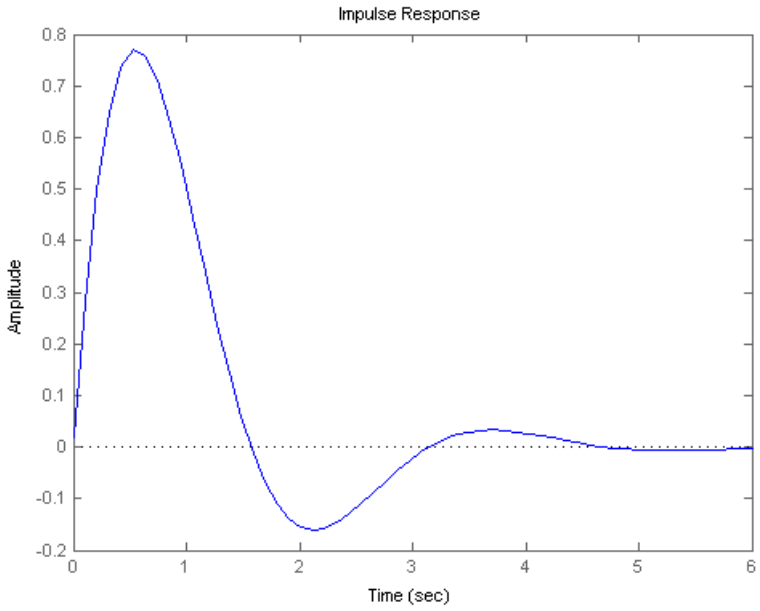
Durum uzayı gösterimindeki sistemler için cevaplar

Durum uzayı gösterimindeki bir sistem için de birim dürtü ve birim basamak cevapları aynı şekilde çizdirilebilir. Örneğin:

```
A = [ 0  1 ; -5  -2 ]; % Sistemin A matrisi
B = [ 0 ; 3 ]; % Sistemin B matrisi
C = [ 1  0 ]; % Sistemin C matrisi
D = 0; % Sistemin D matrisi

P = ss(A,B,C,D); %% Sistemin durum uzayı gösterimini oluştur
```

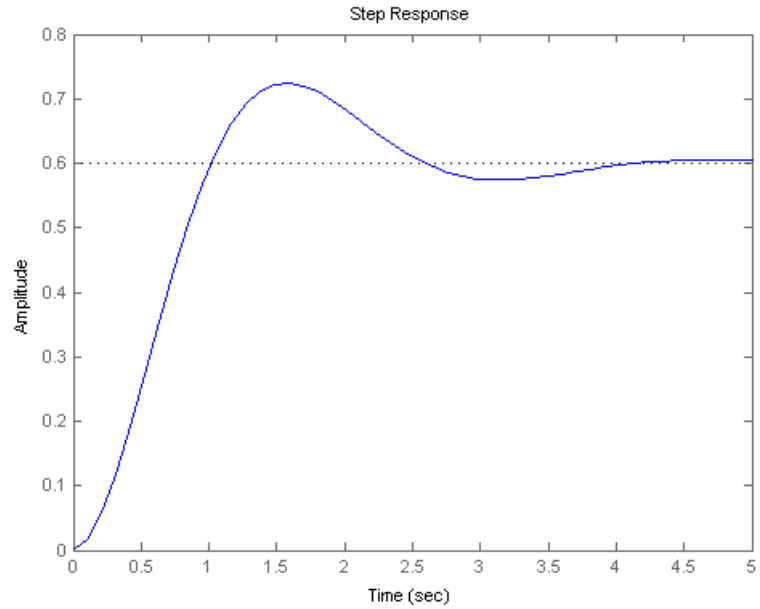
Birim dürtü cevabı:

```
impulse(P);
```



Birim basamak cevabı:

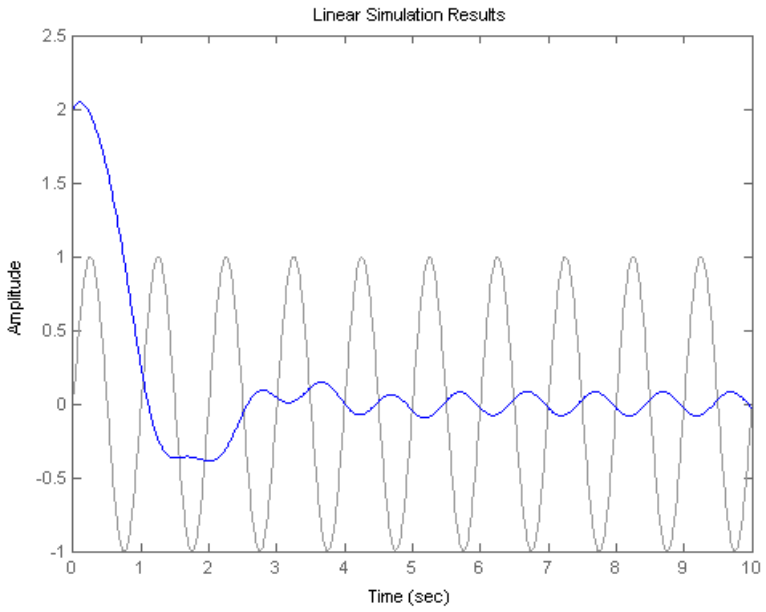
```
step(P)
```



Sistemin genel bir girişe verdiği cevap yine **lsim** ile bulunabilir ancak durum uzayı sistemlerinde ek olarak sistemin başlangıç durumunu da girmek gerekir. Örneğin:

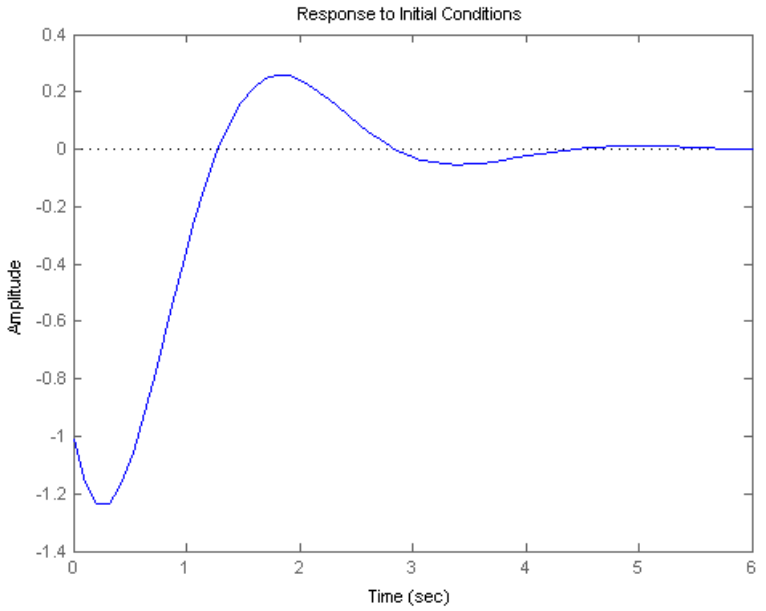
```
fu = 1; % Sinüsün frekansı
t = linspace(0,10,1000); % 0'dan 10 saniyeye kadar bir zaman vektörü
oluşturalım.
u = sin(2*pi*fu*t); % Giriş sinyali
x0 = [2;1]; % Sistemin ilk koşulu
```

```
% t=0 anındaki durumu x0 olan P sistemine, t zaman vektörü boyunca u
% sinyalinin girilmesi ile elde edilen çıkışı hesapla ve çizdir:
lsim(P,u,t,x0)
```



Durum uzayı sistemleri için ayrıca **initial** komutu kulanılarak sıfır-giriş altında sistemin bir koşuldan başlayarak verdiği tepki görülebilir:

```
x0 = [-1,-2];
initial(P,x0);
```

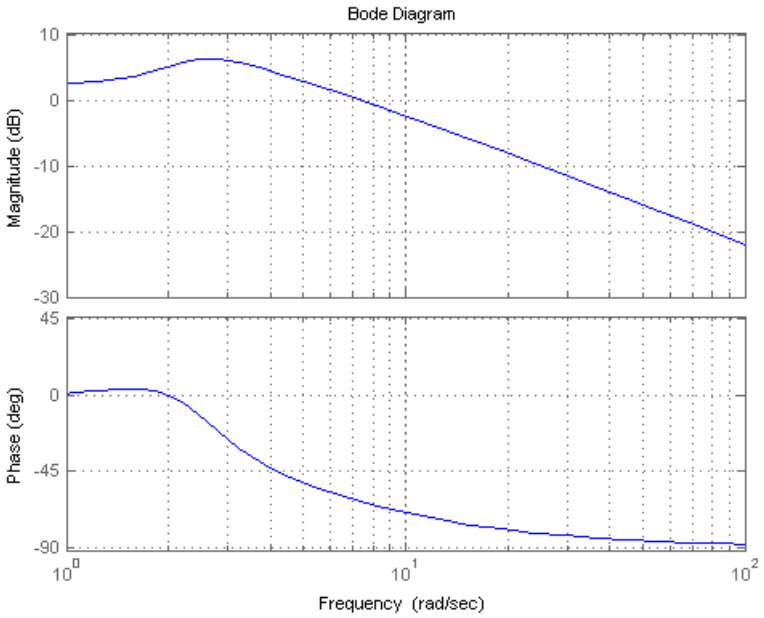


Sistemin frekans cevabı ile ilgili çizimler

Sistemin Bode çizgesini elde etmek **bode** komutu kullanılabilir:

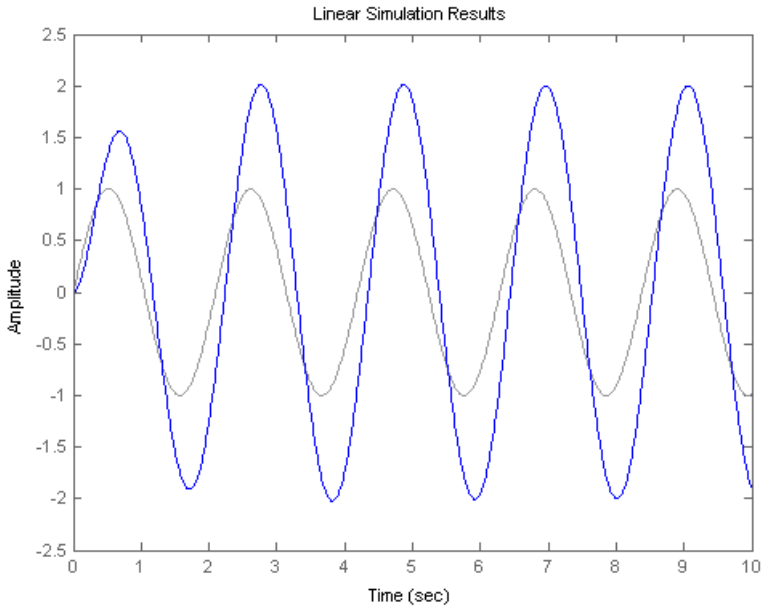
```
s = tf('s'); % Laplace değişkeni
G = (8*s^2+18*s+32)/(s^3+6*s^2+14*s+24) % Sistemi oluştur
bode(G); % Bode çizgesi
grid;
```

```
Transfer function:
   8 s^2 + 18 s + 32
----------------------
s^3 + 6 s^2 + 14 s + 24
```



Bode çizgesinin sistemin değişik frekanslardaki girişleri ne oranda etkilediğini gösterdiğini hatırlayın. Örneğin şekle bakarak f = 3 rad/s frekansındaki bir giriş için kazancın 5.98 dB (yaklaşık iki kat) olacağı görülmektedir. Bunu doğrulamak için **lsim** komutu ile böyle bir sinüsoidal girişe sistemin tepkisini çizdirelim.

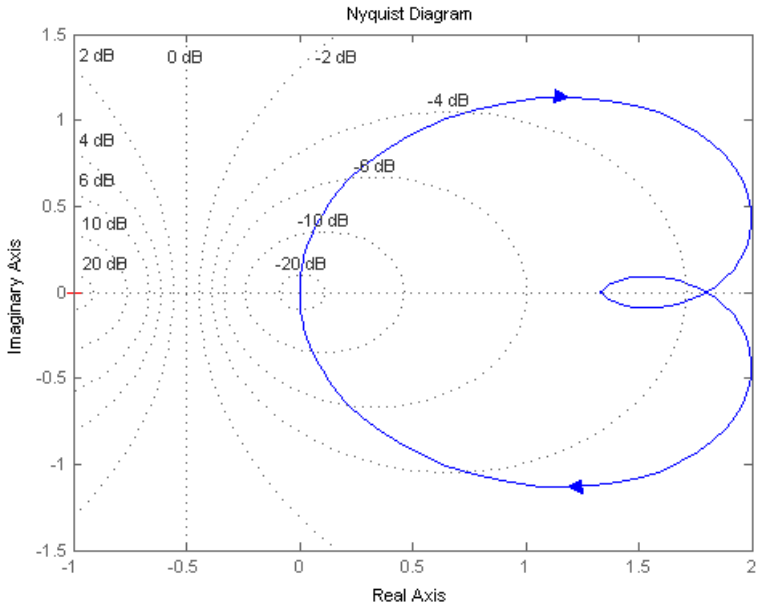
```
wu = 3;
t = linspace(0,10,1000); % 0'dan 10 saniyeye kadar bir zaman vektörü
oluşturalım.
u = sin(wu*t); % Giriş sinyali
x0 = [2;1;-1]; % Sistemin ilk koşulu
lsim(G,u,t,x0); % Sistemin tepkisini çizdir
```



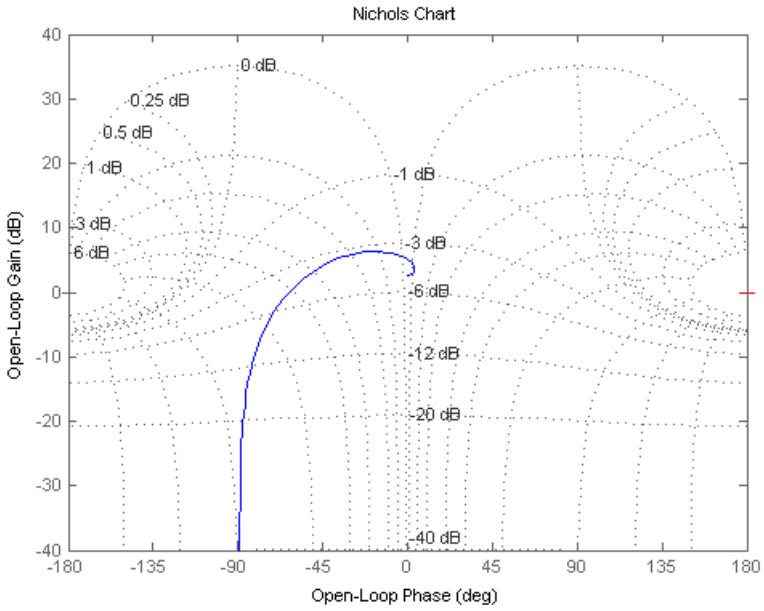
Çıkış sinyalinin genliğinin girişin iki katı olduğu şekilde açıkça görülmektedir.

Sistemin frekans cevabı ile ilgili sık kullanılan Nyquist, Nichols vb. çizgeler de MATLAB'da kolaylıkla oluşturulabilir:

```
nyquist(G);
grid;
```



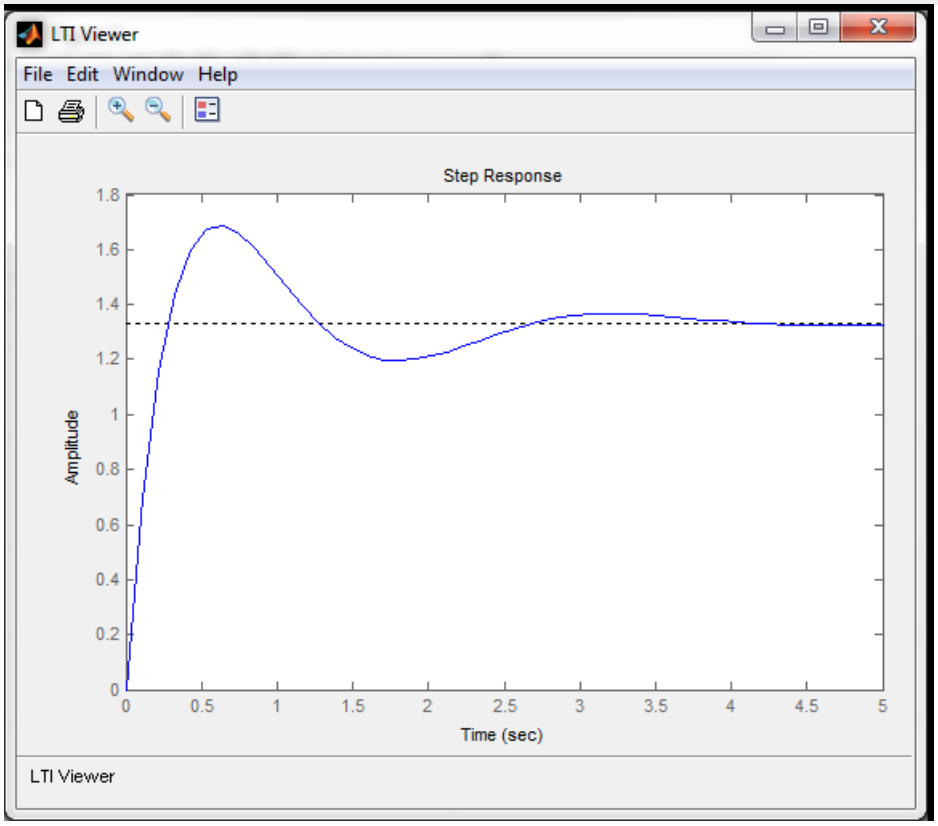

```
nichols(G);
grid
```



Doğrusal sistemlerin analizi için grafiksel arayüz

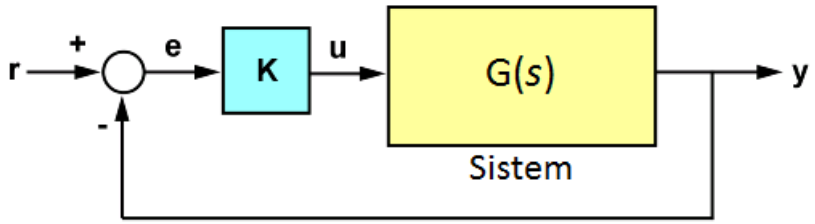
Yukarıda anlatılan çizimlerin pek çoğu **ltiview** grafikesel arayüzü kullanılarak da gerçekleştirilebilir:

```
ltiview(G);
```



Kök-yer eğrisi

Hatırlanacağı üzere kök-yer eğrisi, bir kazanç (K) parametresine bağlı olarak aşağıdaki gibi bir kapalı çevrim sisteminin kutuplarının hareketini grafiksel olarak gösteren bir araçtır.



MATLAB'da kök yer eğrisi çizdirmek için **rlocus** komutu kullanılır. İncelemek için örnek bir sistem oluşturalım:

```
s = tf('s'); % Laplace değişkeni
G = (s+6)/(s^2+4*s+8)/(s-2) % Örnek bir sistem
```

```
Transfer function:
     s + 6
---------------
s^3 + 2 s^2 - 16
```

Sistemin kararsız olduğu görülebilir:

```
isstable(G)
```

```
ans =

     0
```

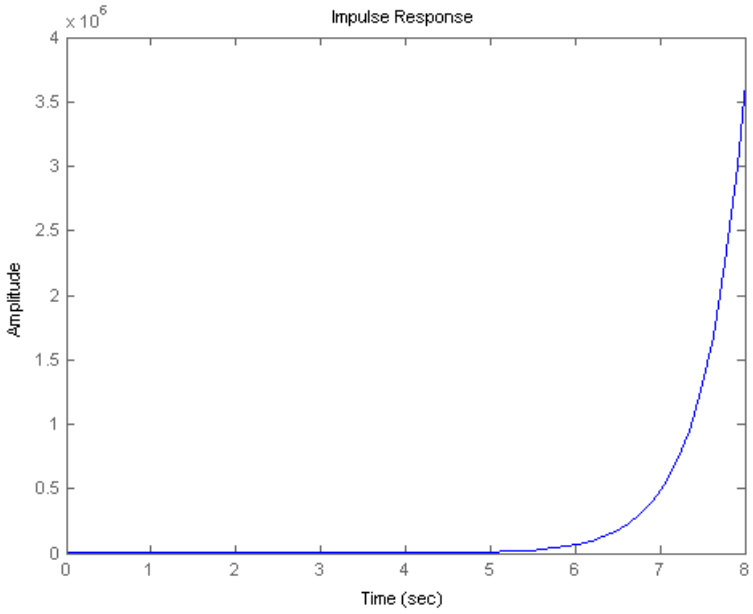
Alternatif olarak sistemin kutuplarına da bakabilirdik. Sağda kutup olduğu için sistem kararsızdır

```
pole(G)
```

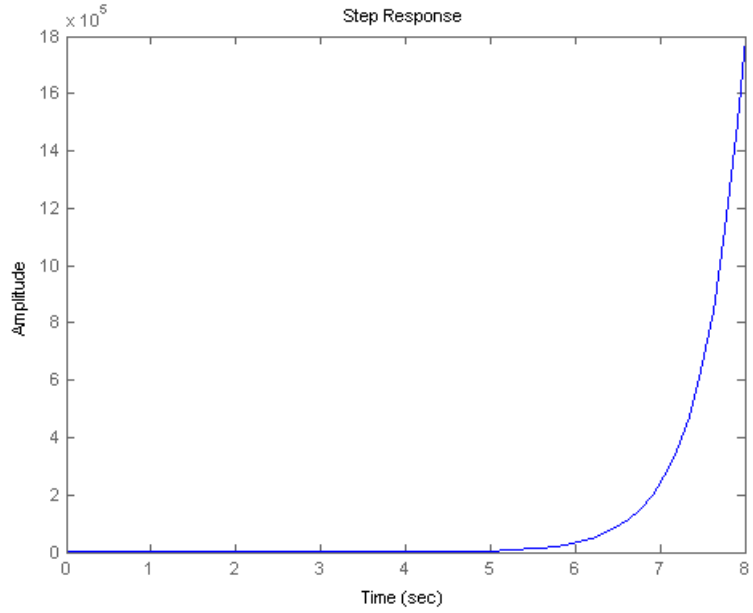
```
ans =

  -2.0000 + 2.0000i
  -2.0000 - 2.0000i
   2.0000
```

Sistemin birim basamak ve birim dürtü cevaplarından da bu durumu teyit edebiliriz

```
impulse(G); % Birim dürtü cevabı
```

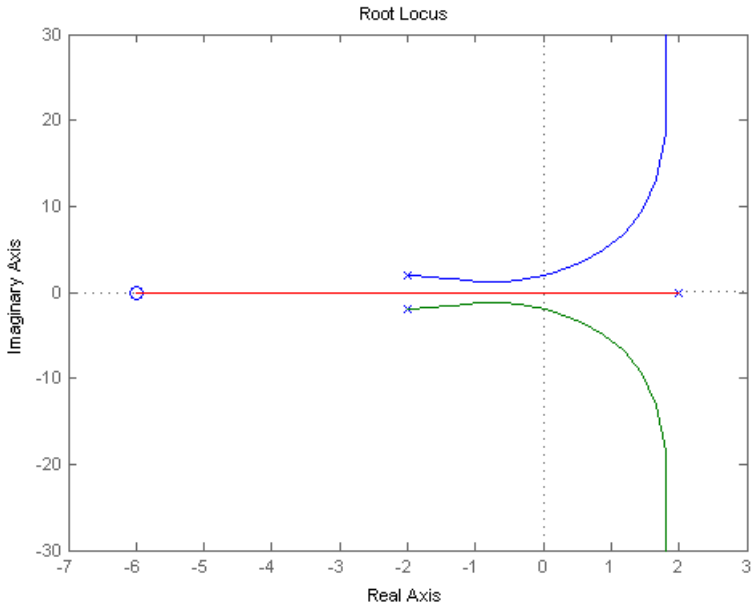



```
step(G); % Birim basamak cevabı
```



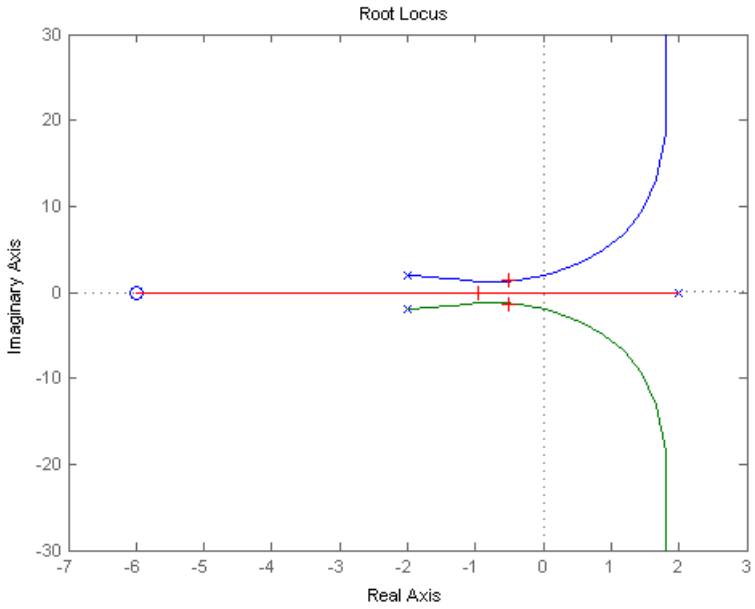
Böyle bir sistemi kararlılaştırmak için yapılabilecek en basit şey, yukarıdaki şekildeki gibi bir kapalı-çevrim sistemi oluşturup, uygun bir kazanç (K) değeri seçmeye çalışmaktır. Kazancı seçmemizde kök-yer eğrisi bize yardımcı olabilir:

```
rlocus(G); % Kök-yer eğrisini çizdir
```



Kök yer üzerindeki noktalara tıklayarak o noktadaki kutubun değerini ve sistemi o noktaya getimek için kullanılması gereken K kazanıcını bulabiliriz. Alternatif olarak kök-yer eğrisi şekli açıkken**rlocfind** komutunu çalıştırıp, daha sonra eğri üzerinde bir noktaya tıklayarak ilgili bilgileri aşağıdaki gibi elde edebiliriz:

```
rlocus(G); % Kök-yer eğrisini çizdir
[K, p] = rlocfind(G); % Kök yer eğrisi üzerinde bir nokta seç,
karşılık
                      % kutupları ve kazancı al.
Select a point in the graphics window

selected_point =

  -0.5664 + 1.3975i
```



Kapalı-çevrim sistemi oluşturma

Seçilen kazanç değeri ile kapalı çevrim sistemini oluşturmak için **feedback** komutundan faydalanabiliriz:

```
GCL = feedback(K*G,1)   % İleri yolda K*G, geri yolda 1 olan, eksi
                        % geribeslemeli kapalı çevrim sistemini
oluştur.
```

```
Transfer function:
       2.987 s + 17.92
----------------------------
s^3 + 2 s^2 + 2.987 s + 1.921
```

Elde etmiş olduğumuz kapalı çevrim sistemi kararlıdır:

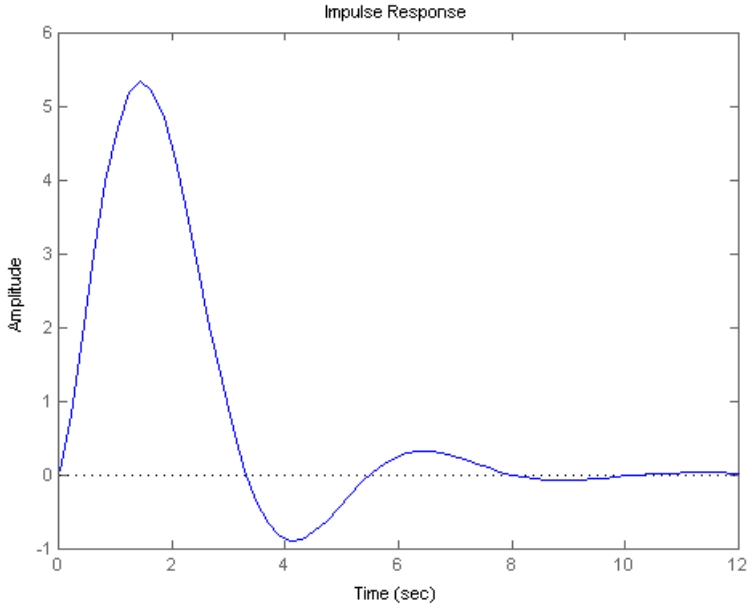
```
isstable(GCL)
ans =

     1
```

Bunu kapalı çevrim sisteminin kutuplarına bakarak da doğrulayabiliriz:

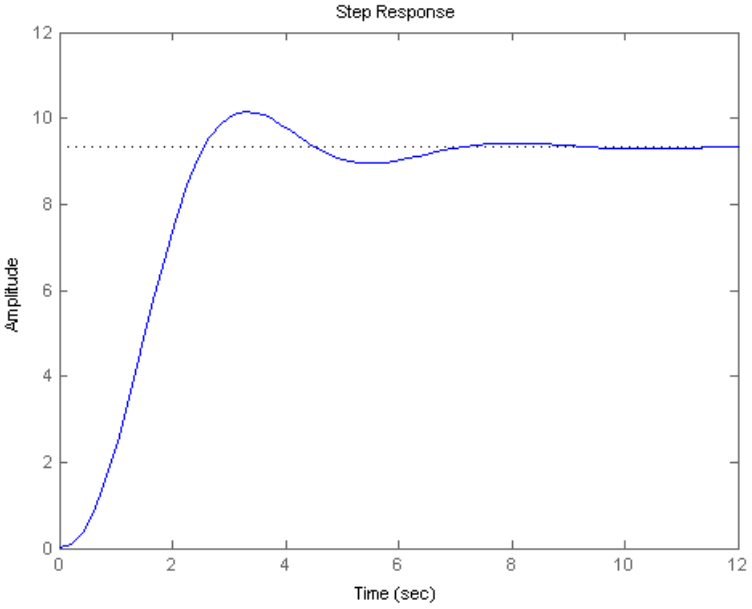
```
pole(GCL)
ans =

  -0.5168 + 1.3118i
  -0.5168 - 1.3118i
  -0.9663
```

Tüm kutuplar solda olduğundan kapalı çevrim sistemi kararlıdır. Birim dürtü ve birim basamak cevaplarına da çizdirelim:

```
impulse(GCL)
```




```
step(GCL)
```



Kapalı çevrim transfer fonksiyonunu oluşturulması ile ilgili bir not

Kapalı çevrim transfer fonksiyonun matematiksel ifadesinin

$$G_{CL}(s) = \frac{KG(s)}{1 + KG(s)}$$

olduğunu biliyoruz. Doğrudan bu ifadeyi kullanarak kapalı çevrim transfer fonksiyonunu oluşturmaya çalışalım:

```
(K*G)/(1 + K*G)
```

```
Transfer function:
       2.987 s^4 + 23.89 s^3 + 35.84 s^2 - 47.79 s - 286.7
-----------------------------------------------------------------
s^6 + 4 s^5 + 6.987 s^4 - 8.105 s^3 - 28.16 s^2 - 47.79 s - 30.74
```

Sonuç GCL ile farklı gibi görünüyor ancak bunun sebebi, MATLAB'ın sonuçtaki sadeleştirmeleri otomatik olarak yapmamasıdır. Bir transfer fonksiyonda sadeleştirmeleri yaptırmak için **minreal**komutunu kullanabiliriz:

```
minreal( (K*G)/(1 + K*G) )
```

```
Transfer function:
       2.987 s + 17.92
-----------------------------
s^3 + 2 s^2 + 2.987 s + 1.921
```

Çıkan sonuç GCL için aynıdır:

```
GCL
```

```
Transfer function:
       2.987 s + 17.92
-----------------------------
s^3 + 2 s^2 + 2.987 s + 1.921
```

Ancak, kapalı-çevrim transfer fonksiyonunu bu şekilde bulmak hem daha zahmetli, hem de nümerik hatalara karşı daha hassastır. O nedenle bu şekilde hesaplamanız tavsiye edilmez, doğru yaklaşım**feedback** komutunu kullanmaktır.

Çok-giriş, çok-çıkışlı sistem tanımlanması

Çok-giriş çok-çıkışlı sistemlerin tanımlanması, tek-giriş tek-çıkış sistem tanımına çok benzerdir. Örneğin, transfer fonksiyonları cinsinden iki giriş, üç çıkışlı bir sistem tanımlayalım:

```
s = tf('s'); % Laplace değişkeni

% Sistemi transfer fonksiyonu matrisi olarak tanımla
H = [2/(s+4) 1/(s+1); 0 (s+1)/(s^2+s+3); -1/(s^2+s+1) 2]
```

```
Transfer function from input 1 to output...
        2
 #1:  -----
      s + 4
```

```
 #2:  0

          -1
 #3:  -----------
      s^2 + s + 1

Transfer function from input 2 to output...
        1
 #1:  -----
      s + 1

         s + 1
 #2:  -----------
      s^2 + s + 3

 #3:  2
```

Sistemin boyutları

```
size(H)
Transfer function with 3 outputs and 2 inputs.
```

Sistem kararlı mı?

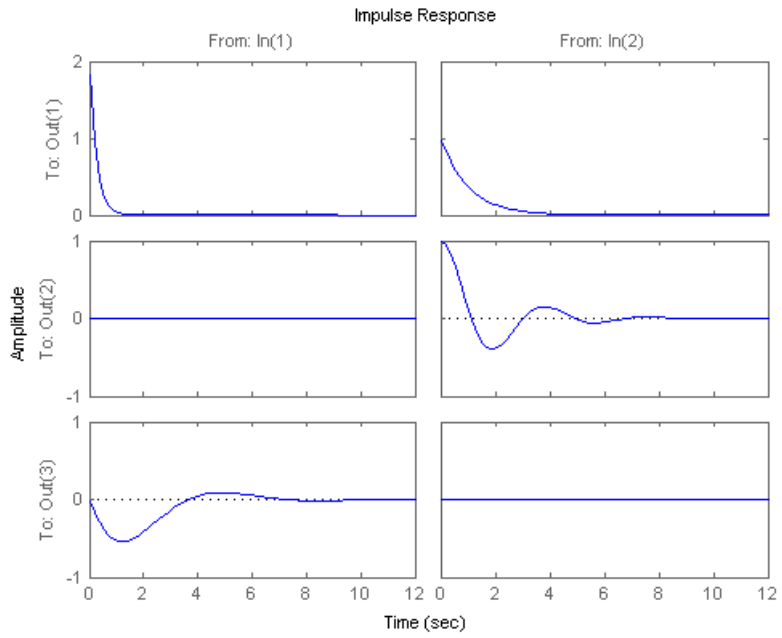
```
isstable(H)
ans =

     1
```

Sistemin kutupları

```
pole(H)
ans =

  -4.0000
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i
  -1.0000
  -0.5000 + 1.6583i
  -0.5000 - 1.6583i
```

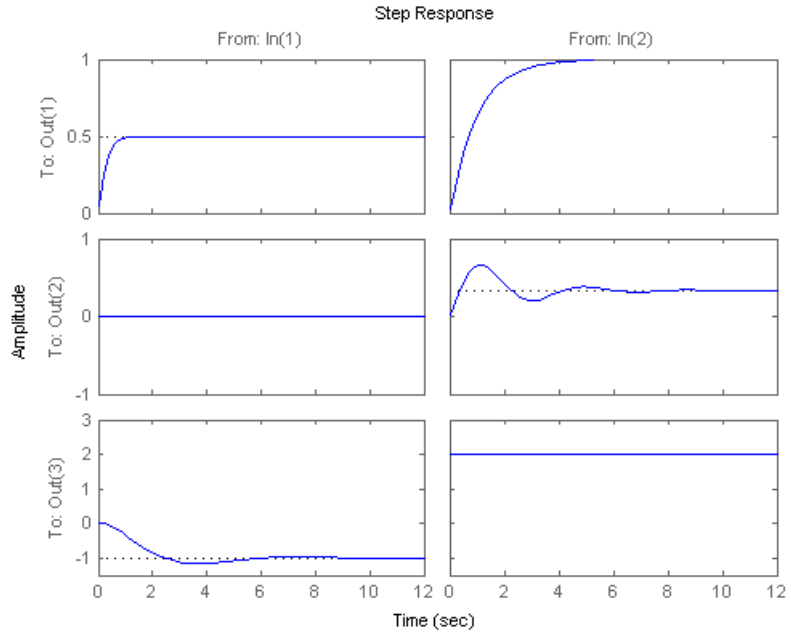
Sistemin birim dürtü cevabı

```
impulse(H)
```



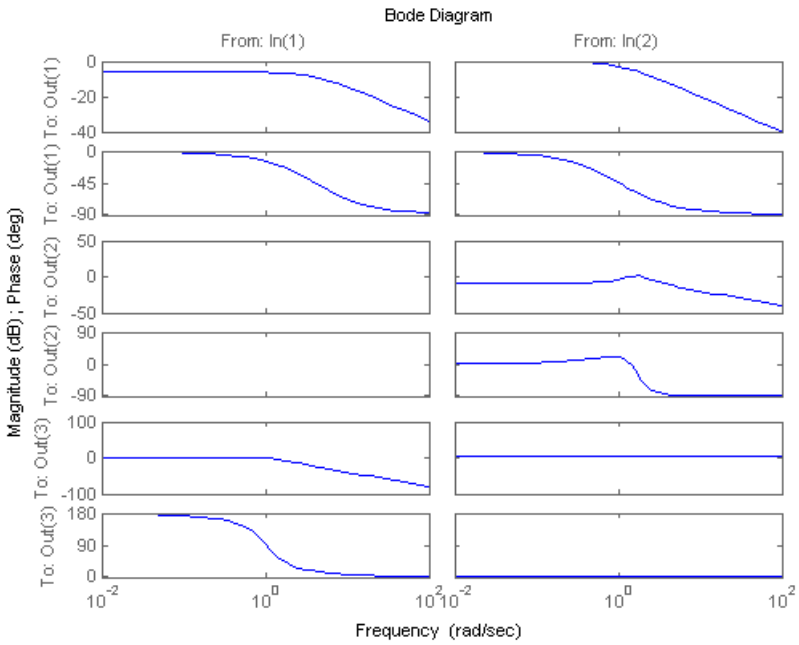
Sistemin birim basamak cevabı

```
step(H)
```



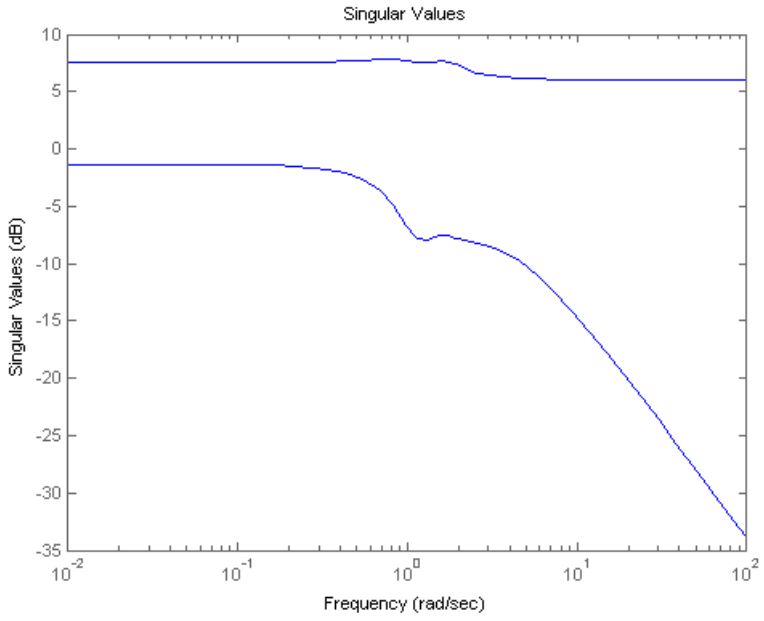
Sistemin frekans cevabı

```
bode(H)
```



Sistemin tekil değerleri

```
sigma(H)
```



Sistemi durum uzayı biçimine çevirelim:

```
HS = ss(H)
```

```
a =
       x1    x2    x3    x4    x5    x6
   x1  -4     0     0     0     0     0
   x2   0    -1    -1     0     0     0
   x3   0     1     0     0     0     0
   x4   0     0     0    -1     0     0
```

```
   x5      0      0      0      0     -1   -1.5
   x6      0      0      0      0      2      0

b =
        u1   u2
   x1    2    0
   x2    1    0
   x3    0    0
   x4    0    1
   x5    0    1
   x6    0    0

c =
        x1    x2    x3    x4    x5    x6
   y1    1     0     0     1     0     0
   y2    0     0     0     0     1   0.5
   y3    0     0    -1     0     0     0

d =
        u1   u2
   y1    0    0
   y2    0    0
   y3    0    2

Continuous-time model.
```

Published with MATLAB® 7.9