به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

با عرض سلام و احترام به دوستان عزیز

بهنام ورمقانی هستم دانشجوی دوره کارشناسی ارشد دانشگاه صنعتی امیرکبیر

آموزش کاربرد متلب در کنترل خطی و کنترل مدرن رو در خدمتتون هستم امیدوارم مباحثی که مطرح میشه مفید واقع بشه.

آموزش در دو دوره برگزار خواهد شد.

در دوره نخست روال آموزش این گونه خواهد بود که تمامی دستورات مربوط به سیستم‌های کنترل خطی و مدرن همراه با مثال‌های آموزشی بررسی می‌شوند.(دوره اول)

در دوره دوم که مکمل دوره اول است به بررسی خواص سیستم‌های خطی مانند اثر صفر و قطب‌ها، مینیمم فاز بودن و نامینیمم فاز بودن و... و طراحی کنترل کننده و نیز بررسی سیستم‌ها در محیط سیمولینک و دیگر محیط‌های گرافیکی متلب مانند pidtool و sisotool خواهیم پرداخت.



Telegram.me/expertonline
مرکز شبیه سازی مقالات مرجع و معتبر

فهرست مطالب

جلسه اول: 4
قسمت اول: یافتن صفرها و قطب های سیستم از روی تابع تبدیل 4
قسمت دوم: یافتن تابع تبدیل سیستم از روی صفرها و قطب ها و بهره سیستم 8
جلسه دوم: 11
دستور tf : 11
دستور ss : 12
سری کردن بلوک ها: 14
موازی کردن بلوک ها: 16
اتصال فیدبک بلوک ها : 18
جلسه سوم: 21
دستور tf2ss : 21
دستور ss2tf : 22
یادآوری : 25
جلسه چهارم: 33
بررسی پاسخ پله سیستم تک متغیره (تک ورودی-تک خروجی) 33
دستور step : 33
جلسه پنجم: 45
بررسی پاسخ پله سیستم تک متغیره (قسمت دوم) 45
حالت ماندگار Steady state 46
فراجهش Overshoot 47
زمان اوج Peak time 48
زمان نشست Settling time 49
زمان صعود Rise time 50
زمان تاخیر Delay time 50
جلسه ششم: 52
بررسی پاسخ پله سیستم تک متغیره (قسمت سوم) 52
دستور stepinfo 52
جلسه هفتم: 60
قسمت اول: 60
بررسی پاسخ ضربه سیستم 60
قسمت دوم : 62
بررسی پاسخ شیب سیستم 62

قسمت سوم: 63
پاسخ به ورودی دلخواه: 63
قسمت چهارم: 68
یافتن پاسخ به شرایط اولیه 68
قسمت پنجم: 71
استفاده از ode45 برای یافتن پاسخ یک سیستم به ورودی و شرایط اولیه دلخواه 71
جلسه هشتم: 76
رسم مکان هندسی ریشه‌ها (قسمت اول) 76
جلسه نهم : 86
رسم مکان هندسی ریشه ها (قسمت دوم) 86
پیدا کردن بهره در نقاط مختلف 88
جلسه دهم : 92
رسم نمودار بوده (قسمت اول) 92
جلسه یازدهم 102
دیاگرام بود (قسمت دوم) 102
دستور semilogx 105
استخراج برخی از ویژگی های سیستم به کمک دیاگرام بود 106
رابطه بین منحنی فاز و منحنی دامنه در سیستم های مینیمم فاز 106
ارتباط بین نمودار بود و نوع (تیپ) سیستم 107
رابطه دیاگرام بود با ثابت خطای مکان، سرعت و شتاب 109
روش دیگر برای بدست آوردن kv و ka 114
یک برنامه کلی برای بدست آوردن ثابت های خطا 116

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه اول:

برای نوشتن ضرایب چند جمله ای ها باید برای همه توان ها ضریب در نظر بگیریم(ولو صفر باشد). مثال:

$$\frac{5\,s^3 - 4s + 3}{7\,s^4 + 4s^3 + 3s^2 - 3s}$$

$$num = [5 \quad 0 \quad -4 \quad 3]$$

$$den = [7 \quad 4 \quad 3 \quad -3 \quad 0]$$

گاهی بهتر است تعداد درایه های num و den را برابر کنیم:

$$num = [0 \quad 5 \quad 0 \quad -4 \quad 3]$$

$$den = [7 \quad 4 \quad 3 \quad -3 \quad 0]$$

قسمت اول: یافتن صفرها و قطب های سیستم از روی تابع تبدیل

برای یافتن صفرها و قطب های سیستم از روی تابع تبدیل ابتدایی ترین راه استفاده از دستور roots به صورت جداگانه برای صورت و مخرج تابع تبدیل میباشد.

مثال:

```
>> num=[7 0 -7];
>> den=[1 5 6];
>> Z=roots(num)

z =

    -1
     1

>> P=roots(den)

p =

   -3.0000
   -2.0000
```

در متلب برای بدست آوردن صفر ها و قطب ها تابع tf2zp وجود دارد این تابع چند جمله ای های صورت و مخرج تابع تبدیل را به عنوان ورودی دریافت میکند و صفرها و قطب ها و بهره سیستم را محاسبه میکند.

مثال :

```
>> num=[7 0 -7];
>> den=[1 5 6];
>> [z,p,k] = tf2zp(num,den)

z =

    -1
     1


p =

   -3.0000
   -2.0000


k =

     7
```

برای نمایش صفرها وقطب ها به صورت نمودار از ابزار fvtool استفاده میکنیم(این ابزار برای بررسی رفتار فیلترها میباشد) استفاده از آن برای رسم صفرها وقطب ها به صورت زیر است:

```
fvtool(num,den,'polezero')
```

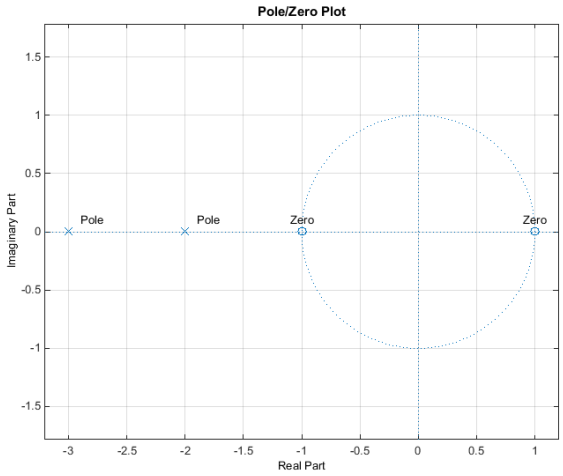
مثال:

```
>> num=[7 0 -7];
>> den=[1 5 6];
>> fvtool(num,den,'polezero')
```

با اجرای دستورات فوق نمودار صفر و قطب سیستم رسم خواهد شد.

حال میتوانیم با استفاده از دستورtext در صفحه نمایش در کنار قطب ها و صفرها متن بنویسیم:

```
>> text(real(z)-0.1,imag(z)+0.1,'Zero')
>> text(real(p)+0.1,imag(p)+0.1,'Pole')
```



همچنین برای بدست آوردن صفر ها وقطب میتوان از دستور pzmap نیز استفاده کرد تفاوت این تابع با تابع tf2zp در ورودی آن هاست زیرا ورودی این تابع خود تابع تبدیل میباشد همچنین این تابع بهره سیستم را محاسبه نمیکند.به علاوه این تابع ابتدا قطب ها و سپس صفرها را بر میگرداند.

مثال:

```
>> s=tf('s');
>> G=(7*s^2-7)/(s^2+5*s+6);
>> [P,Z]=pzmap(G)

P =

   -3.0000
   -2.0000

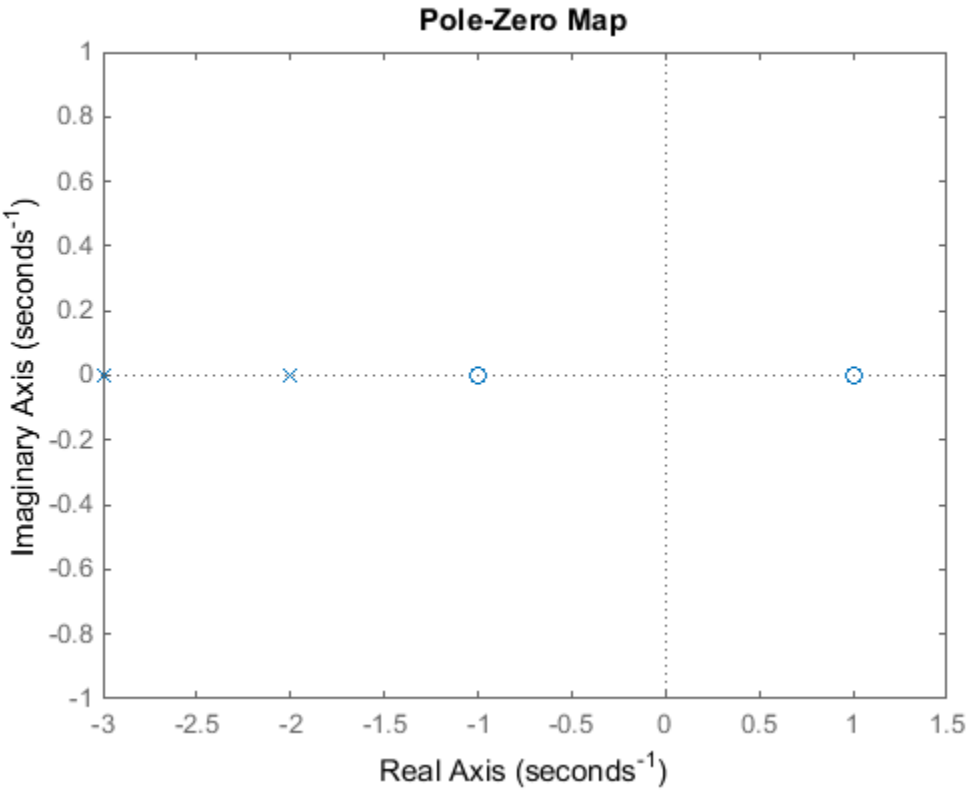

Z =

   -1
    1
```

(در جلسات آینده بطور کامل در مورد تابع tf بحث خواهیم کرد.)

از این تابع برای رسم نمودار صفر ها و قطب ها نیز میتوان استفاده کرد.

```
>> pzmap(G)
```



باید توجه داشت که تابع pzmap میتواند چند جمله ای های صورت و مخرج را نیز به عنوان ورودی دریافت کند:

```
>> num=[7 0 -7];
>> den=[1 5 6];
>> [P,Z]=pzmap(num,den)

P =

   -3.0000
   -2.0000


Z =

    -1
     1
```

قسمت دوم: یافتن تابع تبدیل سیستم از روی صفرها و قطب ها و بهره سیستم

یک راه استفاده از دستور poly میباشد این دستور با گرفتن ریشه ها چند جمله ای متناظر را برمیگرداند.

مثال:

```
>> R=[1;2;3];
>> P=poly(R)

P =

     1    -6    11    -6
```

همان طور که میبینید ضرایب چند جمله ای در بردار P قرار میگیرند.

مثال : (استفاده از دستور poly برای یافتن تابع تبدیل یک سیستم از روی صفرها و قطب ها و بهره سیستم)

برای این کار پس از بدست آوردن چند جمله ای های (مونیک) صورت و مخرج از دستور tf برای نمایش تابع تبدیل سیستم استفاده میکنیم:

```
>> k=7;
Z=[-1;1];
P=[-2;-3];
 num=poly(Z);
den=poly(P);
G=tf(k*num,den)

G =

    7 s^2 - 7
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

از دستور tf برای نمایش تابع تبدیل استفاده میکنیم ورودی های آن چند جمله ای های صورت مخرج میباشند.(این دستور را به صورت کامل در جلسات بعدی بررسی خواهیم کرد.)

به جای دستور tf میتوانیم از دستور printsys استفاده کنیم:

```
>> printsys(k*num,den)

num/den =

     7 s^2 - 7
   -------------
   s^2 + 5 s + 6
```

در این دستور میتوان پارامتر s را تغییر داد:

```
>> printsys(k*num,den,'w')

num/den =

     7 w^2 - 7
   -------------
   w^2 + 5 w + 6
```

در متلب برای بدست آوردن تابع تبدیل سیستم از روی مکان صفرها و قطب ها و بهره سیستم میتوان از دستور zp2tf استفاده کرد این دستور بردارهای صفر و قطب و بهره سیستم را به عنوان ورودی گرفته و چند جمله ای های صورت و مخرج تابع تبدیل سیستم را برمیگرداند.

مثال:

```
>> [num,den]=zp2tf(z,p,k);
>> tf(num,den)

ans =

    7 s^2 - 7
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

همچنین برای یافتن تابع تبدیل سیستم از روی مکان صفر ها وقطب ها و بهره سیستم میتوان از دستور zpk نیز استفاده کرد.

مثال:

```
>> Z=[-1;1];
>> k=7;
>> Z=[-1;1];
>> P=[-2;-3];
>> G=zpk(Z,P,k)

G =

  7 (s+1) (s-1)
  -------------
   (s+2) (s+3)

Continuous-time zero/pole/gain model.
```

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه دوم:

در این جلسه یاد خواهیم گرفت که با داشتن نمودار بلوک دیاگرام یک سیستم کنترلی چگونه با سری و موازی کردن و ساده کردن حلقه های موجود آن را به یک تابع تبدیل واحد تبدیل کنیم.

ابتدا یاد میگیریم که با داشتن صورت و مخرج تابع تبدیل و یا با داشتن فرم فضای حالت یک تابع تبدیل چگونه آن را به شکل قابل قبول برای متلب تبدیل کنیم.

دستور tf :

با داشتن صورت و مخرج یک تابع تبدیل میتوان از دستور tf برای بدست آوردن تابع تبدیل آن استفاده نمود.

```
num=[1 1];
den=[1 5 6];
G=tf(num,den)
```

```
G =

     s + 1
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

یک روش دیگر می تواند به صورت زیر باشد:

```
s=tf('s');
G=(s+1)/(s^2+5*s+6)
```

```
G =

      s + 1
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

دستور SS :

اگر مدل فضای حالت سیستم را داشته باشیم از این دستور برای تبدیل این ماتریس ها به فرم قابل استفاده در متلب استفاده میکنیم:

```
A=[-5 -6;1 0];
B=[1;0];
C=[1 1];
D=0;
G=ss(A,B,C,D)
```

```
G =

  a =
       x1  x2
   x1  -5  -6
   x2   1   0

  b =
       u1
   x1   1
   x2   0

  c =
       x1  x2
   y1   1   1

  d =
       u1
   y1   0

Continuous-time state-space model.
```

کاربرد دیگر دستور SS تبدیل فرم تابع تبدیل به مدل فضای حالت می باشد:

```
s=tf('s');
G=(s+1)/(s^2+5*s+6)

Gss=ss(G)

G =

      s + 1
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.


Gss =

  a =
       x1  x2
   x1  -5  -3
   x2   2   0

  b =
       u1
   x1   1
   x2   0

  c =
        x1   x2
   y1    1  0.5

  d =
       u1
   y1   0

Continuous-time state-space model.
```

همچنین از دستور tf میتوان برای تبدیل فرم فضای حالت به فرم تابع تبدیل نیز استفاده کرد:

```
A=[-5 -6;1 0];
B=[1;0];
C=[1 1];
D=0;
G=ss(A,B,C,D)
```

```
Gsys=tf(G)

G =

  a =
       x1  x2
   x1  -5  -6
   x2   1   0

  b =
       u1
   x1   1
   x2   0

  c =
       x1  x2
   y1   1   1

  d =
       u1
   y1   0

Continuous-time state-space model.


Gsys =

       s + 1
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

سری کردن بلوک ها:

فرض کنید دو بلوک سری داریم می‌خواهیم آن را به یک تابع تبدیل مشخص تبدیل کنیم برای این کار در متلب از دستور series استفاده می‌کنیم:



```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];
G1=tf(num1,den1);
G2=tf(num2,den2);

G=series(G1,G2)

G =

        s^2 + 3 s + 2
  -----------------------
  s^3 + 10 s^2 + 31 s + 30

Continuous-time transfer function.
```

همانطور که می بینیم دستور series عامل های مشترک صورت و مخرج را حذف نمیکند برای ساده کردن تابع تبدیل از دستور mineral استفاده می‌کنیم.

```
Gmin=minreal(G)

Gmin =

      s + 1
  -------------
  s^2 + 8 s + 15

Continuous-time transfer function.
```

هم چنین ورودی های تابع series می‌تواند صورت و مخرج توابع تبدیل باشد :

```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];

[num,den]=series(num1,den1,num2,den2)
```

پس از آن میتوان از دستور tf استفاده کرد:

```
[num,den]=series(num1,den1,num2,den2);
G=tf(num,den);
```

```
Gmin=minreal(G)

Gmin =

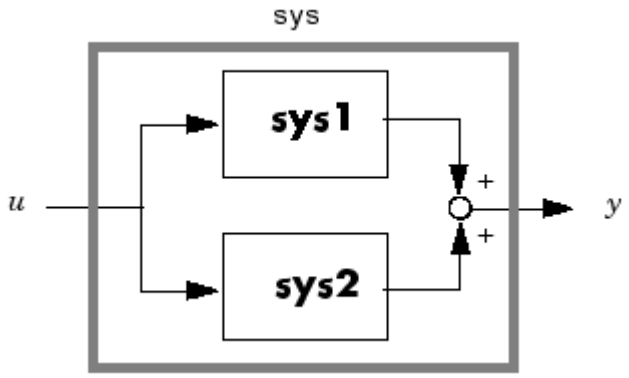
      s + 1
  --------------
  s^2 + 8 s + 15

Continuous-time transfer function.
```

موازی کردن بلوک ها:

دو بلوک را موازی گوییم اگر ورودی آن ها یکی بوده و خروجیشان با هم جمع شود.

در متلب برای محاسبه سیستم حاصل از موازی شدن دو بلوک از دستور parallel استفاده میکنیم.




```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];
sys1=tf(num1,den1);
sys2=tf(num2,den2);

sys=parallel(sys1,sys2)
```

```
sys =

  s^3 + 8 s^2 + 22 s + 17
  -----------------------
  s^3 + 10 s^2 + 31 s + 30

Continuous-time transfer function.
```

هم چنین ورودی های تابع parallel می‌تواند صورت و مخرج توابع تبدیل باشد :

```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];

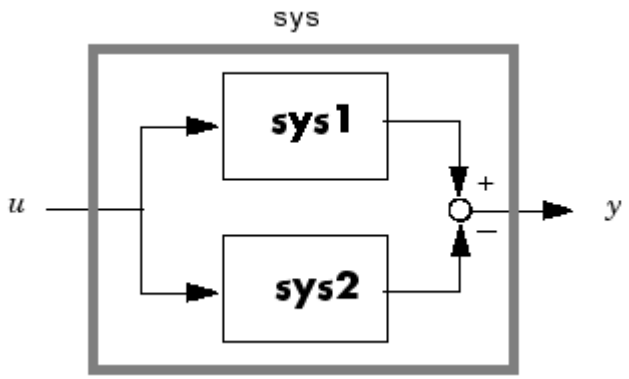
[num,den]=parallel(num1,den1,num2,den2);
sys=tf(num,den)
```

```
sys =

  s^3 + 8 s^2 + 22 s + 17
  -----------------------
  s^3 + 10 s^2 + 31 s + 30

Continuous-time transfer function.
```

توجه کنید که اگر بلوک های موازی شده از هم کم شوند کافی است که منفی را به تابع تبدیل متناظر منتقل کنیم :




```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];

sys1=tf(num1,den1)
sys2=tf(num2,den2)

sys=parallel(sys1,-sys2)
```

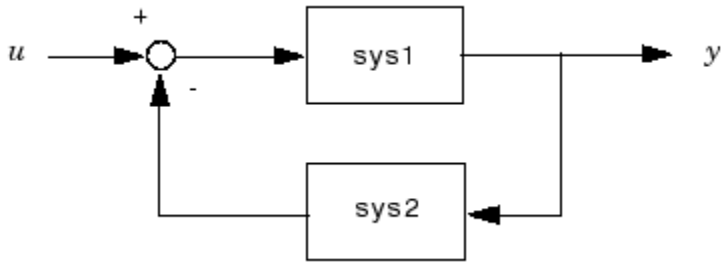
```
sys =

  -s^3 - 6 s^2 - 10 s - 7
  -----------------------
  s^3 + 10 s^2 + 31 s + 30

Continuous-time transfer function.
```

اتصال فیدبک بلوک ها :

اتصال فیدبک دار در شکل زیر نشان داده شده است.



در متلب برای ساده سازی اتصال فیدبک دار از دستور feedback استفاده می‌کنیم.

```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];

sys1=tf(num1,den1);
sys2=tf(num2,den2);

sys= feedback(sys1,sys2)

sys =

       s^2 + 6 s + 5
  -----------------------
  s^3 + 11 s^2 + 34 s + 32

Continuous-time transfer function.
```

همچنین میتوان به صورت زیر نیز استفاده کرد:

```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];

[num,den]=feedback(num1,den1,num2,den2);
sys=tf(num,den)
```

```
sys =

        s^2 + 6 s + 5
  -----------------------
  s^3 + 11 s^2 + 34 s + 32

Continuous-time transfer function.
```

پیش فرض متلب در محاسبه فیدبک ، فیدبک منفی است اگر بخواهیم فیدبک مثبت را ساده کنیم باید ورودی سومی برای تابع feedback در نظر گرفته شود این ورودی عدد 1+ میباشد:

```
num1=[1 1];
den1=[1 5 6];
num2=[1 2];
den2=[1 5];

sys1=tf(num1,den1);
sys2=tf(num2,den2);

sys= feedback(sys1,sys2,+1)
```

```
sys =

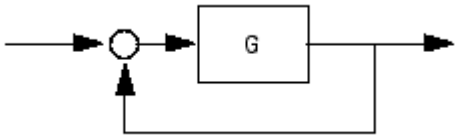
        s^2 + 6 s + 5
  ----------------------
  s^3 + 9 s^2 + 28 s + 28

Continuous-time transfer function.
```

همچنین میتوانیم منفی بودن فیدبک را به تابع تبدیل مسیر فیدبک انتقال دهیم:

```
sys= feedback(sys1,-sys2)
```

اگر فیدبک واحد داریم از دستور زیر استفاده میکنیم:




```
sys= feedback(sys1,[1])
```

مثال:

```
num1=[1 1];
den1=[1 5 6];

sys1=tf(num1,den1);


sys=feedback(sys1,[1])
```

```
 sys =

      s + 1
  -------------
  s^2 + 6 s + 7

 Continuous-time transfer function.
```

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه سوم:

در این جلسه دو دستور برای تبدیل مدل فضای حالت به تابع تبدیل و برعکس را یاد خواهیم گرفت.

دستور tf2ss :

این دستور برای تبدیل مدل تابع تبدیل به مدل فضای حالت استفاده میشود شیوه به کار بردن آن به صورت زیر است :

$$[A,B,C,D] = tf2ss(num,den)$$

مثال :

```
num=[-7 7];
den=[1 5 6];

[A,B,C,D]=tf2ss(num,den)
```

```
A =

    -5    -6
     1     0


B =

     1
     0


C =

    -7     7


D =

     0
```

میدانیم که مدل فضای حالت یک سیستم یکتا نیست و درواقع بیشمار مدل فضای حالت هم ارز داریم که میتوانیم آن ها را با تبدیل های همانندی به هم تبدیل کنیم.

دستور tf2ss تنها یکی از این تحقق ها را میدهد که به فرم کانونیکال کنترل پذیر است.

دستور ss2tf :

از این دستور برای تبدیل مدل فضای حالت به تابع تبدیل استفاده می‌کنیم شیوه استفاده برای سیستم های siso به صورت زیر است:

$$[num, den] = ss2tf(A, B, C, D)$$

مثال:

```
A=[-5 -6;1 0];
B=[1;0];
C=[-7 7];
D=0;

[num,den]=ss2tf(A,B,C,D)
G=tf(num,den)
```

```
num =

     0     -7      7

den =

     1      5      6

G =

    -7 s + 7
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

اما برای سیستم های چند ورودی چند خروجی (MIMO) باید ورودی را مشخص کنیم. با این کار در واقع به صورت ستونی توابع تبدیل را بدست می‌آوریم.

یادآوری:

یک سیستم MIMO با m ورودی و l خروجی دارای ماتریس تابع تبدیل l در m خواهد بود..

$$Y = G_{l\times m}\, U$$

هر درایه (i,j) ماتریس G تابع تبدیل ورودی $j-$ ام به خروجی $i-$ ام می‌باشد.

برای به دست آوردن ماتریس تابع تبدیل یک سیستم چند متغیره از روی مدل فضای حالت نیز از دستور ss2tf استفاده میکنیم اما ورودی سومی برای آن در نظر می‌گیریم که در واقع نشان دهنده شماره ورودی و یا متناظراً نشان دهنده ستون مورد نظر است. فرم کلی به صورت زیر است:

$$[NUM, DEN] = ss2tf(A, B, C, D, k)$$

که k نشان دهنده ورودی k ام است. با اجرای این دستور NUM یک بردار ستونی متناظر با ستوم K- ام از ماتریس G میباشد که هر درایه آن خود یک بردار سطری است که ضرایب صورت تابع تبدیل را نشان میدهد.

و den یک بردار سطری است که در واقع مخرج همه توابع تبدیل های ستون k –ام از G را نشان میدهد.

مثال زیر توضیحات فوق را شفاف خواهد کرد:

```
A=[-1 0;-2 -3];
B=[1 2;0 3];
C=[-1 1;0 1];
D=zeros(2);

[NUM,DEN]=ss2tf(A,B,C,D,1)
```

```
NUM =

     0    -1    -5
     0     0    -2


DEN =

     1     4     3
```

با این دستور در واقع ستون اول تابع تبدیل را بدست آوردیم:

$$G = \begin{bmatrix} \dfrac{-s-5}{s^2+4s+3} & * \\ \dfrac{-2}{s^2+4s+3} & * \end{bmatrix}$$

حال برای یافتن ستون دوم از دستور زیر استفاده میکنیم:

```
[NUM,DEN]=ss2tf(A,B,C,D,2)
```

```
NUM =

         0    1.0000   -7.0000
         0    3.0000   -1.0000

DEN =

     1     4     3
```

و در نتیجه ماتریس G به صورت زیر تکمیل می شود:

$$G = \begin{bmatrix} \frac{-s-5}{s^2+4s+3} & \frac{s-7}{s^2+4s+3} \\ \frac{-2}{s^2+4s+3} & \frac{3s-1}{s^2+4s+3} \end{bmatrix}$$

در این مثال به صورت اتفاقی مخرج های ستون اول و دوم یکی شده است.ولی مخرج های هر ستون همواره یکسان میباشد.

یاد گرفتیم که با داشتن فرم فضای حالت یک سیستم MIMO ماتریس تابع تبدیل آن را بیابیم و هم چنین طرز بدست آوردن فرم فضای حالت یک سیستم SISO از روی تابع تبدیل آن را یاد گرفتیم حال با داشتن ماتریس تابع تبدیل یک سیستم MIMO طریقه بدست آوردن فرم فضای حالت آن را یاد میگیریم:

یادآوری :

فرض کنید که ماتریس تابع تبدیل یک سیستم MIMO را داریم و می خواهیم فرم فضای حالت آن را بدست آوریم (این کار اصطلاحا تحقق تابع تبدیل نامیده میشود).

روش های مختلفی برای انجام این کار وجود دارد اما دو روش (دو تحقق) از اهمیت ویژه ای برخوردارند یکی تحقق کانونیکال کنترل پذیر است و دیگری تحقق کانونیکال رؤیت پذیر. در این جا چون در تحقق به کمک متلب از تحقق کانونیکال کنترل پذیر استفاده می شود بنابراین تحقق کانونیکال کنترل پذیر را مرور میکنیم:

ماتریس تابع تبدیل $G_{l\times m}$ را در نظر میگیریم l تعداد خروجی ها و m تعداد ورودی هاست.

ابتدا حد تمامی درایه های آن را در بی نهایت حساب میکنیم این ماتریس همان ماتریس D می باشد یعنی $D = \lim_{S\to\infty} G_{l\times m}(s)$ پس از بدست آوردن ماتریس D با کم کردن ماتریس D از ماتریس $G_{l\times m}$ ماتریس $G_{SP_{l\times m}}$ را به دست میاوریم که یک ماتریس تابع تبدیل اکیدا سره است یعنی درجه صورت همه درایه های آن از درجه مخرج متناظر کم تر است.

حال تمامی درایه های هر ستون از ماتریس $G_{SP_{l\times m}}$ را هم مخرج میکنیم(در ستون ها مخرج مشترک میگیریم) با این کار داریم:

$$G_{SP_{l\times m}} = [g_1 , g_2 , \dots , g_m]$$

که g_i ستون $i-$ ام ماتریس $G_{SP_{l\times m}}$ میباشد. همچنین تمامی درایه های این ستون ها دارای مخرج یکسان هستند.

حال در واقع باید همه درایه های ماتریس $G_{SP_{l\times m}}$ به صورت جداگانه تحقق دهیم اما این کار ساده خواهد یود زیرا همه درایه های یک ستون مخرج یکسان دارند و بنابراین ماتریس های Aو B یکسان خواهند داشت و فقط ماتریس C آن ها متفاوت خواهد بود.

پس از بدست آوردن هریک از ماتریس های A_j و B_j برای هرستون ماتریس A وB سیستم اصلی به صورت زیر خواهد بود:

$$A = diag(A_1 , A_2 , \dots , A_m)$$

یعنی ماتریس های A_j که خود هرکدام یک بلوک هستند را روی قطر اصلی قرار داده و بقه درایه ها را صفر میگذاریم.

همچنین :

$$B = diag(B_1 , B_2 , \dots , B_m)$$

و ماتریس C نیز به صورت زیر به دست می آید:

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{l1} & C_{l2} & \cdots & C_{lm} \end{bmatrix}$$

که هر کدام از C_{ij} ها تحقق درایه (i,j) ماتریس $G_{SP_{l\times m}}$ میباشند.

حال سوال این است که هریک از A_j و B_j و C_{ij} به چه شکل هستند؟

فرض کنیم $d_j(s)$ مخرج مشترک همه درایه های ستون $j-$ام ماتریس $G_{SP_{l\times m}}$ باشد و مرتبه آن برابر δ_j باشد فرم کلی ان به صورت زیر است:

$$d_j(s) = s^{\delta_j} + \alpha^j_{\delta_j-1}s^{\delta_j-1} + \cdots + \alpha^j_1 s + \alpha^j_0$$

بنابراین فرم کلی هریک از درایه های ماتریس $G_{SP_{l\times m}}$ به صورت زیر خواهد بود:

$$g_{ij}(s) = \frac{\beta^{ij}_{\delta_j-1}s^{\delta_j-1} + \beta^{ij}_{\delta_j-2}s^{\delta_j-2} + \cdots + \beta^{ij}_1 s + \beta^{ij}_0}{s^{\delta_j} + \alpha^j_{\delta_j-1}s^{\delta_j-1} + \cdots + \alpha^j_1 s + \alpha^j_0}$$

با توجه به این فرم ماتریس های A_j و B_j و C_{ij} به صورت زیر خواهند بود:

$$A_j = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha^j_0 & -\alpha^j_1 & \cdots & -\alpha^j_{\delta_j-2} & -\alpha^j_{\delta_j-1} \end{bmatrix}$$

$$B_j = [0 \quad 0 \quad \cdots \quad 0 \quad 1]^T$$

$$C_{ij} = \begin{bmatrix} \beta^{ij}_0 & \beta^{ij}_1 & \cdots & \beta^{ij}_{\delta_j-1} \end{bmatrix}$$

توجه کنید که ممکن است تعدادی از β^{ij}_r صفر باشند (مخصوصا اولی ها).

بنابراین تحقق دادن یک ماتریس تابع تبدیل چند متغیره را به صورت دستی یاد گرفتیم حال میخواهیم یاد بگیریم که این کار چگونه توسط متلب انجام میشود:

برای این کار از دستور tf2ss استفاده میکنیم.
ابتدا در ستون ها مخرج مشترک میگیریم. ضرایب مخرج مشترک را در بردار DENj قرار میدهیم و ضرایب صورت ها در هر ستون را به ترتیب در سطهای ماتریس NUMj قرار میدهیم.(توجه کنید که برخلاف روش دستی نیازی به جدا کردن ماتریس D نداریم) سپس از دستور tf2ss استفاده میکنیم:

$$[A_j\ , B_j\ , C_j\ , D_j\ ] = tf2ss(NUMj, DENj)$$

و سپس ماتریس A برابر ماتریس بلوکی قطری خواهد بود که بلوک های قطراصلی آن ماتریس های A_j خواهند بود و همچنین ماتریس B برابر ماتریس بلوکی قطری خواهد بود که بلوک های قطراصلی آن ماتریس های B_j خواهند بود. و ماتریس های C و D به صورت زیر میباشند:

$$C = [C_1 \quad C_2 \quad \cdots \quad C_m]$$

$$D = [D_1 \quad D_2 \quad \cdots \quad D_m]$$

شاید مطالب فوق اندکی گنگ به نظر اید اما در هنگام حل یک مثال خواهیم دید که کار بسیار روتین میباشد:

مثال :

$$G(s) = \begin{bmatrix} \dfrac{s+6}{s^2+5s+6} & \dfrac{1}{s+3} \\ \dfrac{1}{s+2} & 1 \\ \dfrac{1}{s^2+5s+6} & \dfrac{s+7}{s+3} \end{bmatrix}$$

ابتدا در ستون ها مخرج مشترک میگیریم:

$$G(s) = \begin{bmatrix} \frac{s+6}{s^2+5s+6} & \frac{1}{s+3} \\ \frac{s+3}{s^2+5s+6} & \frac{s+3}{s+3} \\ \frac{1}{s^2+5s+6} & \frac{s+7}{s+3} \end{bmatrix}$$

اگر بخواهیم دستی تحقق را به دست آوریم به صورت زیر عمل میکنیم:

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$G_{sp}(s) = \begin{bmatrix} \frac{s+6}{s^2+5s+6} & \frac{1}{s+3} \\ \frac{s+3}{s^2+5s+6} & 0 \\ \frac{1}{s^2+5s+6} & \frac{4}{s+3} \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix} \quad ; \quad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$A_2 = -3 \quad ; \quad B_2 = 1$$

$$c_{11} = [6 \quad 1] \; ; \quad c_{21} = [3 \quad 1] \; ; \quad c_{31} = [1 \quad 0]$$

$$c_{12} = [1] \; ; \quad c_{22} = [0] \; ; \quad c_{32} = [4]$$

بنابراین :

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -6 & -5 & 0 \\ 0 & 0 & -3 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 6 & 1 & 1 \\ 3 & 1 & 0 \\ 1 & 0 & 4 \end{bmatrix}$$

حال میخواهیم به کمک متلب این کار را انجام دهیم:

```
num11=[1 6];
num21=[1 3];
num31=[0 1];
DEN1=[1 5 6];
num12=[0 1];
num22=[1 3];
num32=[1 7];
DEN2=[1 3];

NUM1=[num11;num21;num31];
NUM2=[num12;num22;num32];

[A1,B1,C1,D1]=tf2ss(NUM1,DEN1);
[A2,B2,C2,D2]=tf2ss(NUM2,DEN2);

za1=zeros(size(A1,1),size(A2,2));
za2=zeros(size(A2,1),size(A1,2));

A=[A1 za1;za2 A2]

zb1=zeros(size(A1,1),1);
zb2=zeros(size(A2,1),1);

B=[B1 zb1;zb2 B2]

C=[C1 C2]
D=[D1 D2]
```

```
A =

    -5    -6     0
     1     0     0
     0     0    -3

B =

     1     0
     0     0
     0     1

C =

     1     6     1
     1     3     0
     0     1     4

D =

     0     0
     0     1
     0     1
```

همانطور که میبینیم جواب حاصل از متلب با جواب روش دستی متفاوت شد.

همان طور که قبلا گفتیم مدل فضای حالت یکتا نیست!

هر دو جواب صحیح است!(میتوانید امتحان کنید)

برای امتحان کردن جواب ها میتوانید از دستور ***ss2tf*** همانگونه که توضیح داده شد استفاده کنید (ستون ها را جداگانه بدست آورید)

همچنین میتوانید از برنامه زیر استفاده کنید:

```
syms s

G=C*inv(s*eye(size(A))-A)*B+D
```

در این برنامه ابتدا متغیر s را به صورت سمبلیک تعریف میکنیم سپس از رابطه

$G = C(s\,I - A)^{-1}B + D$ استفاده کرده ایم.

به نام خدا

جلسه چهارم:

بررسی پاسخ پله سیستم تک متغیره (تک ورودی-تک خروجی)

در متلب برای بدست آوردن پاسخ پله یک سیستم دو روش وجود دارد یکی استفاده از دستور step و دیگری استفاده از دستور stepinfo میباشد. در این جلسه دستور step را بررسی خواهیم نمود.

تفاوت دستور step و دستور stepinfo در این است که دستور step صرفاً بردار خروجی (و حالت ها) در پاسخ به ورودی پله را میدهد و برای بدست آوردن مشخصات پاسخ پله نیاز به کدنویسی و یا استفاده از نمودار پاسخ پله خواهیم داشت اما در دستور stepinfo مشخصات پاسخ پله را میتوان مستقیما استخرج کرد. با این حال ابتدا به توضیح دستور step و کدنویسی مورد نیاز برای بدست آوردن مشخصات پاسخ پله خواهیم پرداخت.

دستور step :

برای رسم نمودار پاسخ پله از یکی از دستورهای زیر استفاده میکنیم:

```
Step(sys)

Step(num,den)

Step(a,b,c,d)

Step(sys,t)

Step(num,den,t)

Step(a,b,c,d,u,t)
```

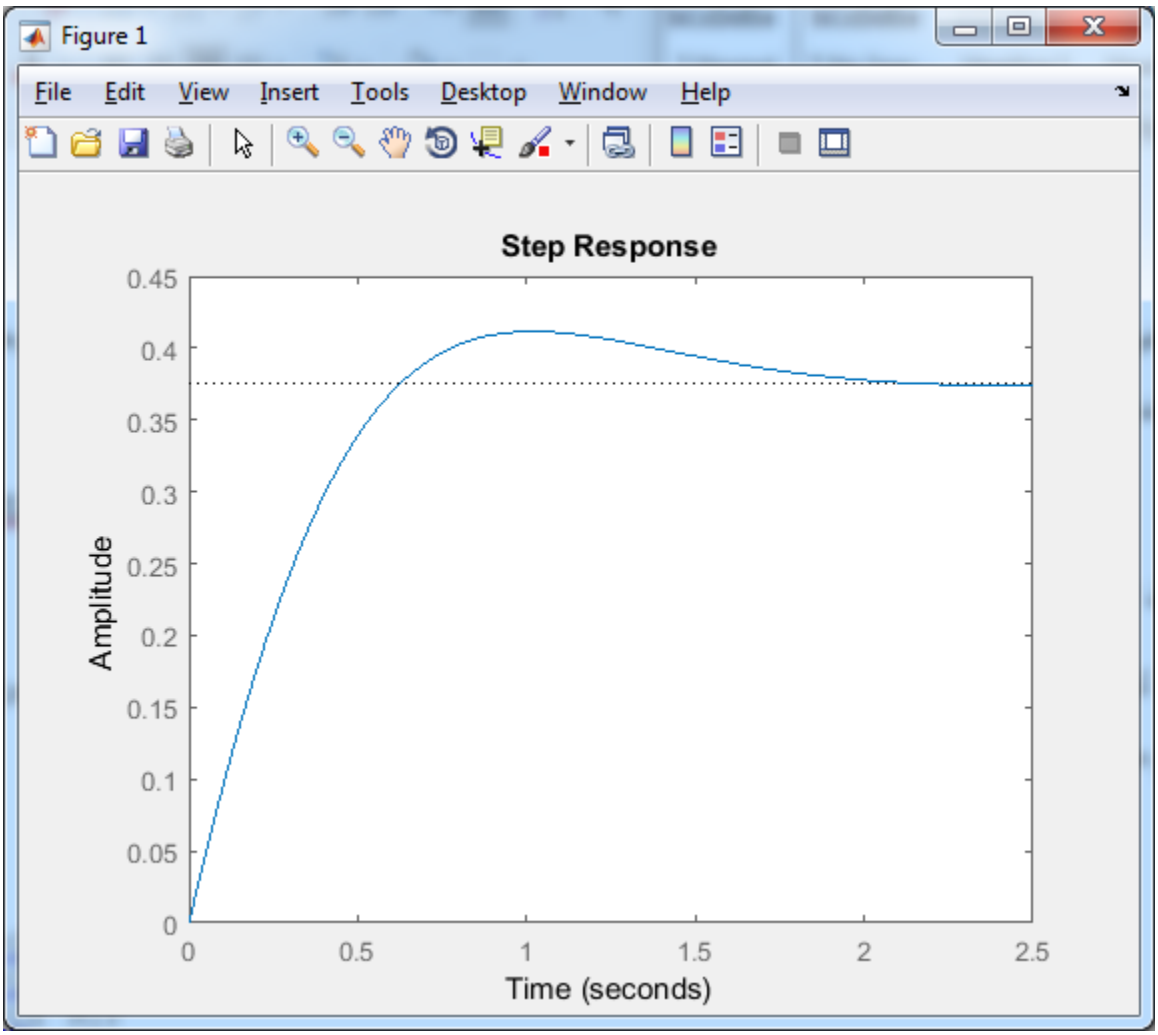
در دستور آخر u یک عدد است که ورودی مورد نظر را مشخص میکند در بحث فعلی ما چون سیستم تک متغیره میباشد بنابراین آن را برابر 1 قرار میدهیم.

Step(a,b,c,d,1,t)

مثال:

```
a=[-4 -8;1 0];
b=[1;0];
c=[1 3];
d=0;

step(a,b,c,d)
```



حال بردار زمان را خودمان وارد میکنیم:

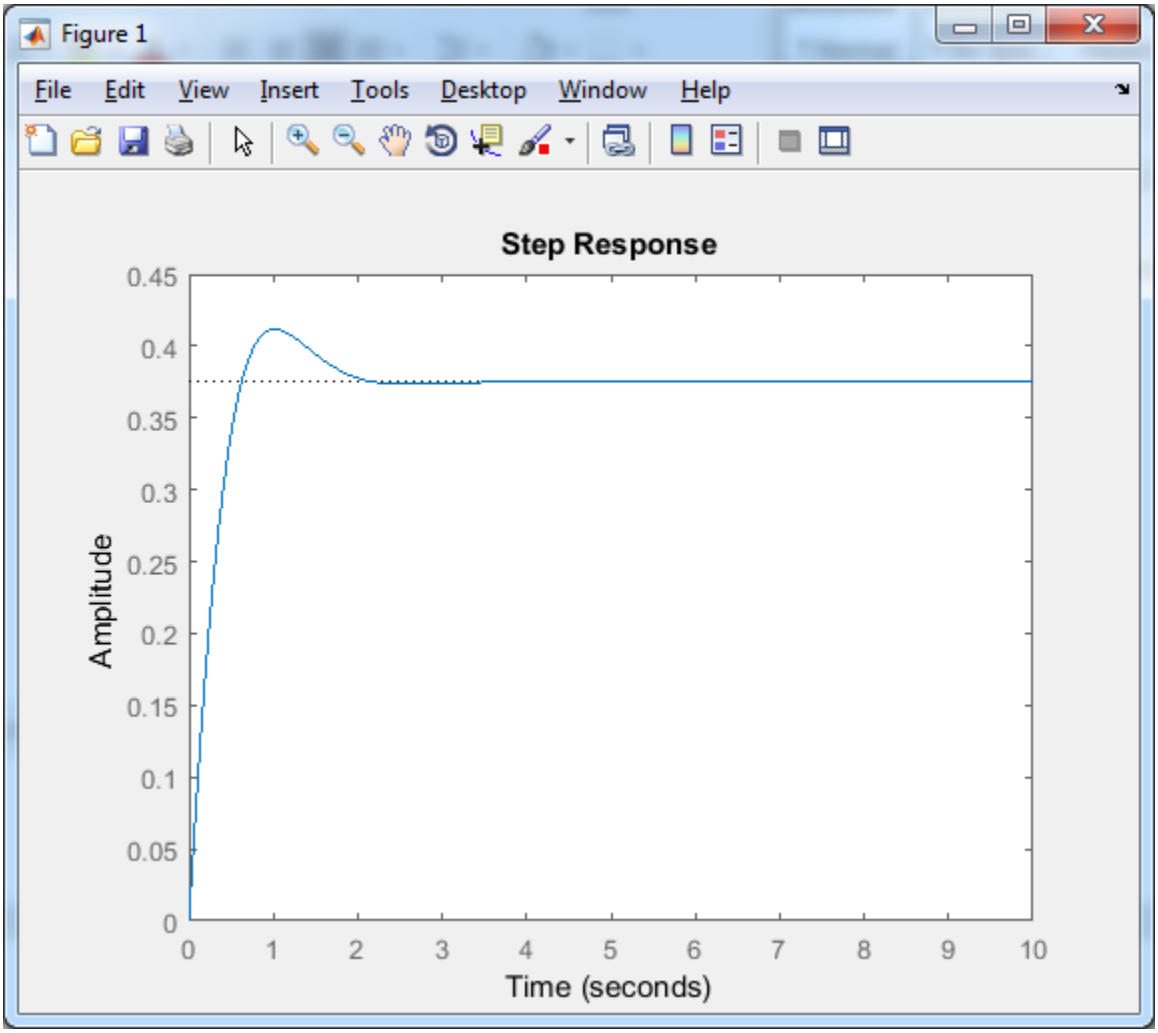
در این صورت ابتدا از دستور **SS** استفاده کنیم:

```
a=[-4 -8;1 0];
b=[1;0];
c=[1 3];
d=0;

t=0:0.01:10;

plant=ss(a,b,c,d);

step(plant,t)
```



میتوانستیم از برنامه زیر نیز استفاده کنیم که البته همان نتیجه قبل را خواهد داشت:

```
clc
clear
a=[-4 -8;1 0];
b=[1;0];
c=[1 3];
d=0;

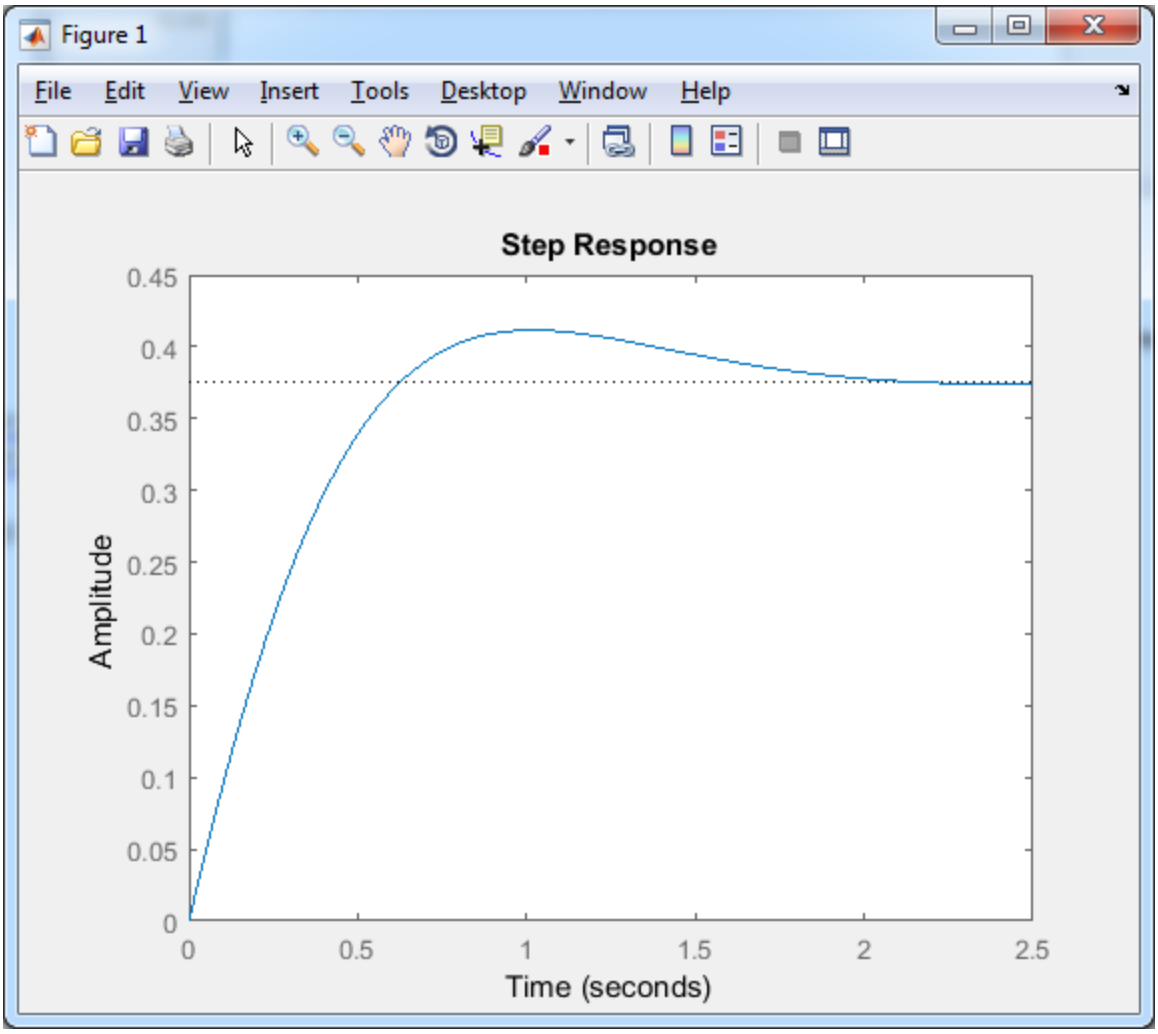
t=0:0.01:10;


step(a,b,c,d,1,t)
```

همچنین میتوان از فرم های دیگر استفاده کرد مثلا:

```
num=[1 3];
den=[1 4 8];

step(num,den)
```

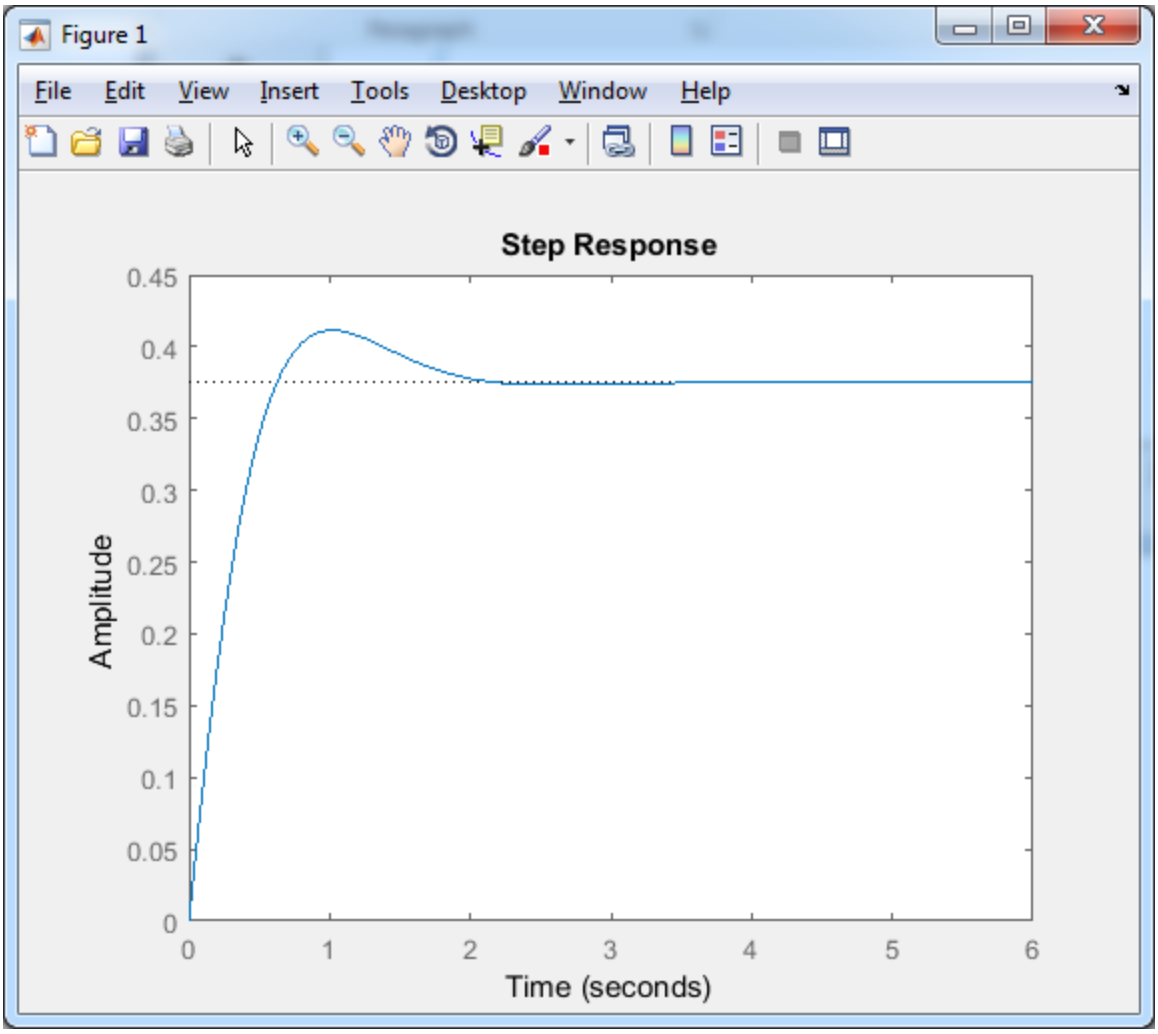


مثال :

```
num=[1  3];
den=[1  4  8];

t=0:0.05:6;

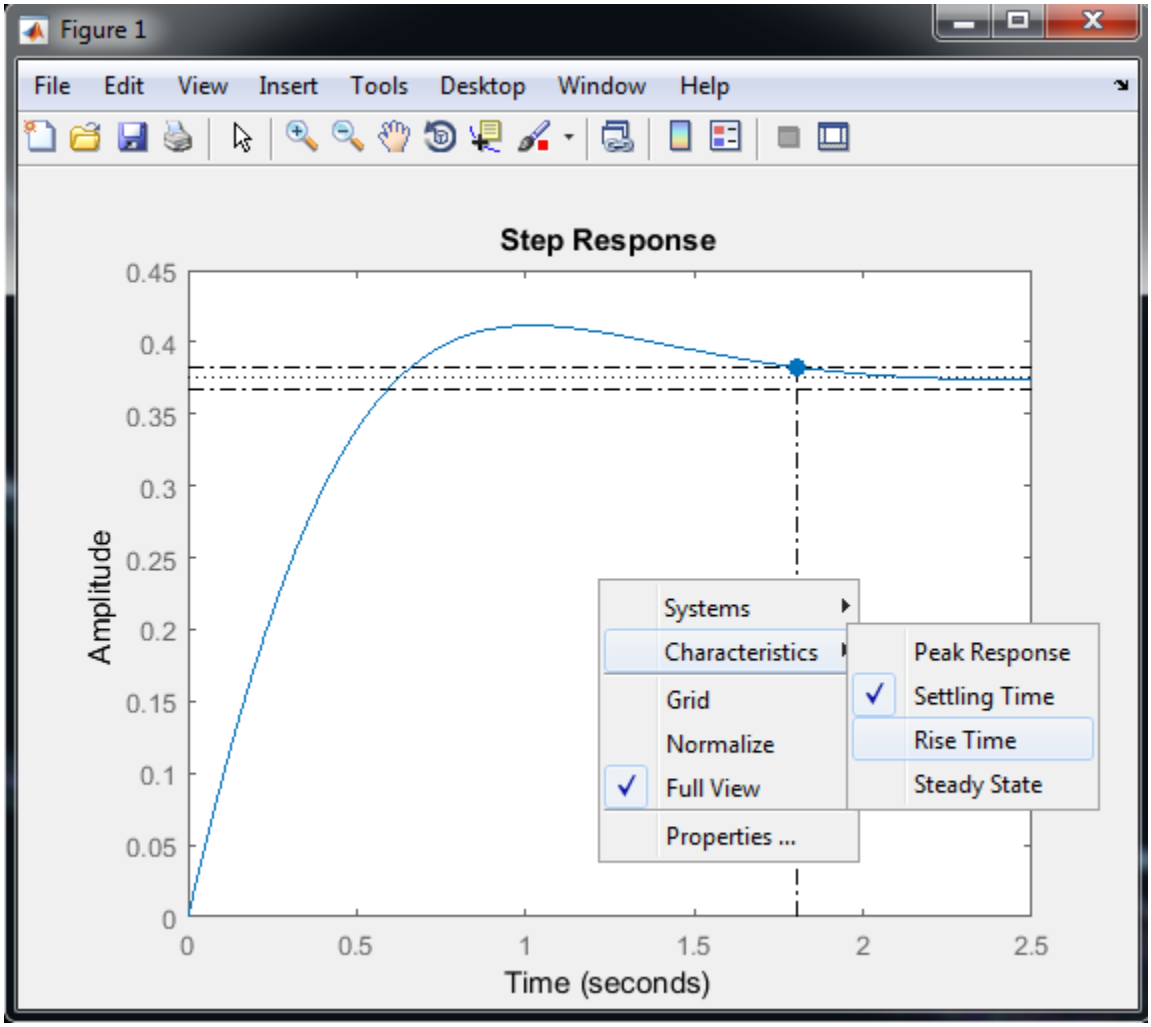
step(num,den,t)
```



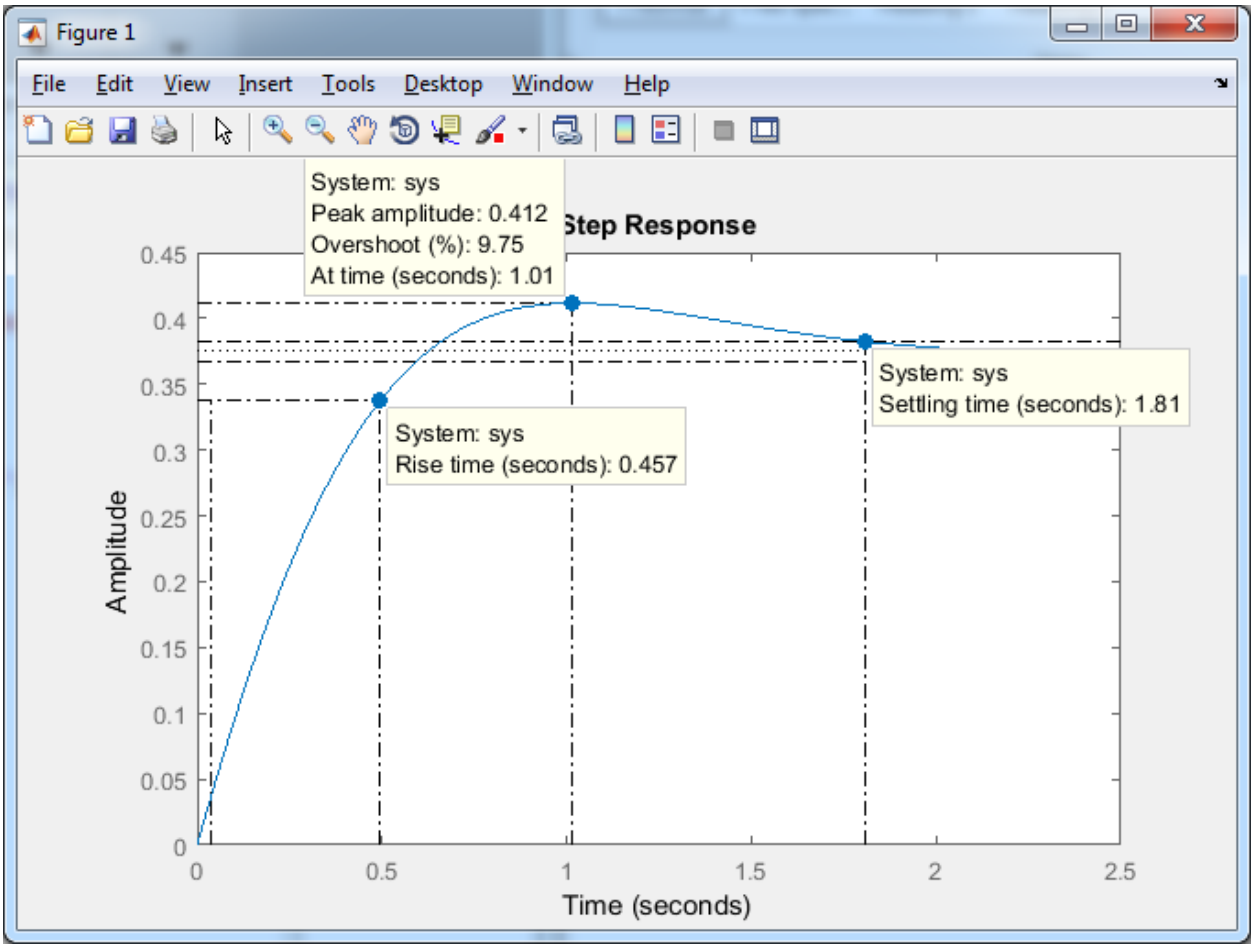
پس از رسم نمودار پاسخ پله از همه دستورات ترسیم میتوانید استفاده کنید.

مثلا متن اضافه کنید یا legend بگذارید و ... (برای یادگیری در مورد تنظیمات صفحه ترسیم میتوانید فایل های آموزش مقدماتی متلب را از اینترنت دانلود کنید)

با کلیک راست کردن در صفحه ترسیم میتوانیم مشخصات پاسخ پله را اضافه کنیم به این صورت :



هر کدام از مشخصات را انتخاب کنیم به شکل اضافه میشود. سپس با کلیک کردن بر روی نقطه مشخص شده میتوان مقدار عددی آن را دید:



میتوان بردارهای خروجی پاسخ پله و نیز زمان را ذخیره کرد و از آن ها استفاده کرد برای این کار به صورت زیر هریک از دستورات فوق را به کار میبریم.

[y,x,T]=step(.)

در این حالت y بردار خروجی و x بردار حالت های سیستم و T نیز بردار زمان است.

با استفاده از این دستور هیچ نموداری ترسیم نمیشود و برای رسم پاسخ پله باید از دستور plot استفاده کنیم.

توجه کنید که اگر بردار t را خودمان به عنوان ورودی دستور step ندهیم در این صورت خود متلب یک بردار با گام های متغیر برای آن در نظر میگیرد.

همچنین دقت کنید که اگر سیستم را در فضای حالت ندهیم در این صورت بردار x تهی خواهد بود.

مثال های زیر نکات فوق را مرور میکنند.

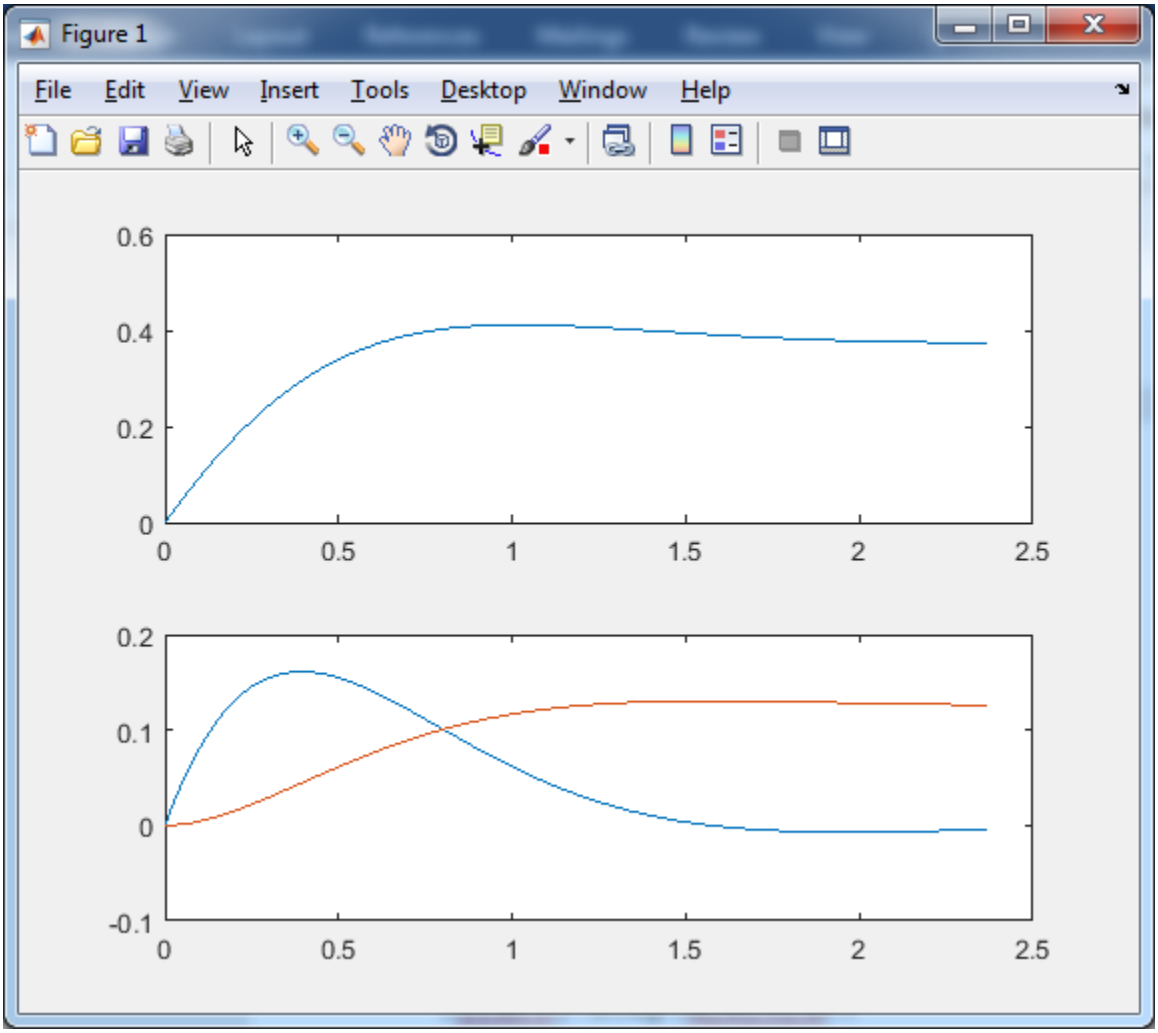
مثال :

```
a=[-4 -8;1 0];
b=[1;0];
c=[1 3];
d=0;

[y,x,T]=step(a,b,c,d);

subplot(2,1,1)
plot(T,y)

subplot(2,1,2)
plot(T,x)
```



توجه شود که چون دو متغیر حالت داریم پس در تصویر پایینی دو نمودار خواهیم داشت.

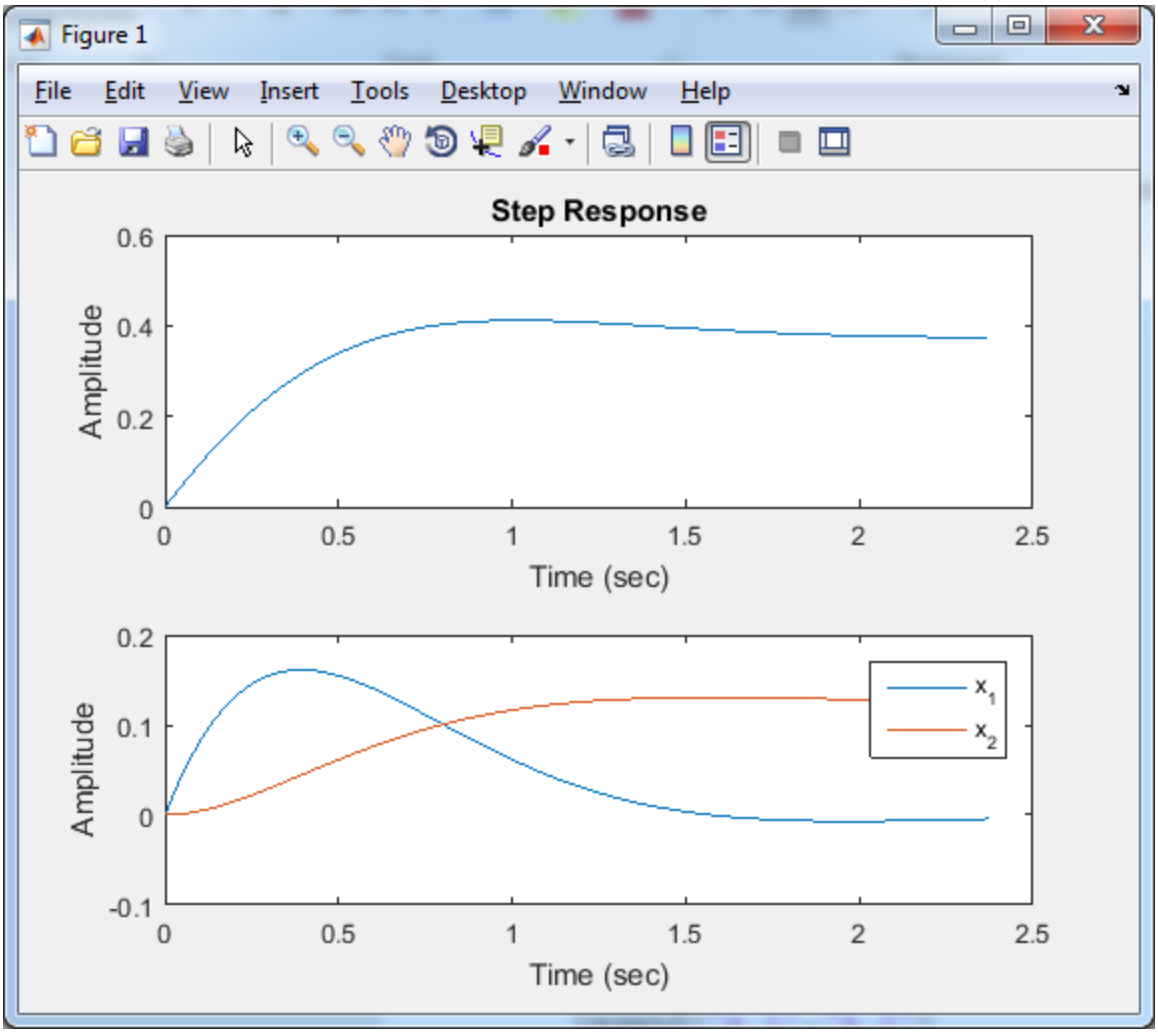
همچنین میتوان label ها و ... را اضافه کرد.

```
a=[-4 -8;1 0];
b=[1;0];
c=[1 3];
d=0;

[y,x,T]=step(a,b,c,d);

subplot(2,1,1)
plot(T,y)
xlabel('Time (sec)')
ylabel('Amplitude')
title('Step Response')


subplot(2,1,2)
plot(T,x)
xlabel('Time (sec)')
ylabel('Amplitude')
legend('x_1','x_2')
```



توجه کنید که در این حالت نمیتوان مشخصات پاسخ پله را اضافه کرد.

مثال :

```
num=[1  3];
den=[1  4  8];

[y,x,T]=step(num,den);
size(x)
```

```
ans =

     0     0
```

همانطور که میبینید بردار **X** تهی میباشد.

در جلسه آینده در مورد کدنویس برای بدست آوردن مشخصات پاسخ پله بحث خواهیم کرد.

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه پنجم:

بررسی پاسخ پله سیستم تک متغیره (قسمت دوم)

جلسه پیش در مورد چگونگی بدست آوردن نمودار پاسخ پله و همچنین ذخیره کردن داده های مربوط به پاسخ پله بحث کردیم در این جلسه به کدنویسی برای بدست آوردن مشخصات پاسخ پله می‌پردازیم.

مشخصات اساسی پاسخ پله عبارتند از:

فراجهش Overshoot

زمان اوج Peak time

زمان نشست Settling time

زمان صعود Rise time

زمان تاخیر Delay time

حالت ماندگار Steady state

این ویژگی ها، ویژگی های اساسی برای پاسخ پله می‌باشند و معیار عملکردی سیستم هستند. برای سیستم های مرتبه دوم استاندارد برحسب فرکانس طبیعی نامیرا و نیز ضریب میرایی سیستم به صورت تقریبی میتوان این مشخصات را بدست آورد اما بدست آوردن آنها برای سیستم های مرتبه بالا یا باید مستقیما از روی پاسخ پله مشخص شود و یا با کاهش مرتبه به صورت تقریبی بدست آید. در این جا ما از روش اول یعنی از روی داده های پاسخ پله این مشخصات را استخراج می کنیم.

حال به کدنویسی برای بدست آوردن هریک از مشخصه های فوق می‌پردازیم:

حالت ماندگار Steady state

مقدار نهایی و ماندگار پاسخ پله را حال ماندگار گویند برای محاسبه آن کافی است مقادیر نهایی خرجی پاسخ پله را بدست آوریم برنامه زیر این کار را انجام می‌دهد:

```
clc
clear

num=[7];
den=[1,5,6];

[y,x,T]=step(num,den);

y_ss=y(numel(T))
```

خروجی برنامه به صورت زیر است:

```
y_ss =

    1.1631
```

برای بدست آوردن مقدار دقیق تر مقدار نهایی بهتر است زمان را نیز خودمان وارد کنیم زیرا اگر زمان را خودمان وارد نکنیم متلب به صورت خودکار معمولا زمان کوچکی برای آن در نظر می‌گیرد و بنابراین ممکن است مقداری خطا داشته باشیم این خطا مخصوصا در محاسبه مقدار درصد فراجهش بیشتر خود را نشان میدهد.

در برنامه زیر زمان را خودمان وارد می‌کنیم:

```
clc
clear

num=[7];
den=[1,5,6];

t=0:0.005:10;

[y,x,T]=step(num,den,t);
```

```
y_ss=y(numel(T))
```

فراجهش Overshoot

با توجه به این که این مشخصات پاسخ پله معمولا برای سیستم های مرتبه دو استاندارد محاسبه می‌شود بنابراین مقدار فراجهش را نسبت به مقدار یک اندازه میگیرند یعنی حالت نهایی را برابر یک در نظر میگیرند اما ما به جای ماکزیمم فراجهش از درصد فراجهش استفاده می‌کنیم و آن را به صورت زیر محاسبه میکنیم.

$$\text{Maximum percent overshoot} = \frac{c(t_p) - c(\infty)}{c(\infty)} \times 100\%$$

برنامه زیر مقدار درصد فراجهش را محاسبه می‌کند:

```
clc
clear

num=[1];
den=[1 1 1];

[y,x,T]=step(num,den);

y_ss=y(numel(T));
[ymax,Tmax]=max(y);

ovs_per_cent=(ymax(1)-y_ss)/y_ss*100
```

خروجی به صورت زیر است:

```
ovs_per_cent =

   15.8324
```

برای دقت بیشتر بهتر است زمان را نیز خودمان وارد کنیم:

```
clc
clear

num=[1];
den=[1 1 1];
```

```
t=0:0.005:15;

[y,x,T]=step(num,den,t);

y_ss=y(numel(T));
[ymax,Tmax]=max(y);

ovs_per_cent=(ymax(1)-y_ss)/y_ss*100
```

خروجی به صورت زیر است:

```
ovs_per_cent =

   16.3773
```

زمان اوج Peak time

زمان لازم برای رسیدن پاسخ پله به مقدار ماکزیمم خود.

برنامه زیر برای بدست آوردن زمان اوج می‌باشد:

```
clc
clear

num=[1];
den=[1 1 1];

dt=0.005;
t=0:dt:15;

[y,x,T]=step(num,den,t);

y_ss=y(numel(T));
[ymax,Tmax]=max(y);

Peak_Time=(Tmax-1)*dt
```

خروجی به صورت زیر می‌باشد:

```
Peak_Time =

    3.6300
```

زمان نشست Settling time

زمان نشت نخستین زمانی است که منحنی پاسخ پله وارد باند مشخصی حول مقدار نهایی خود می‌شود و از آن خارج نمی‌شود یعنی از این زمان به بعد قدرمطلق مقدار اختلاف پاسخ پله از مقدار نهایی آن کمتر از نسبت معینی از مقدار نهایی می‌باشد.

معمولا معیار 2 درصد را برای زمان نشست در نظر میگیرند (هرچند مقادیر 3 و 5 درصد نیز رایجند) برنامه زیر با در نظر گرفتن معیار 2 درصد زمان نشست پاسخ پله را بدست می‌دهد:

```
clc
clear

num=[1];
den=[1 1 1];

dt=0.005;
t=0:dt:15;

[y,x,T]=step(num,den,t);

y_ss=y(numel(T));

k=numel(T);

while(true)

    k=k-1;
    if y(k)<0.98*y_ss || y(k)>1.02*y_ss
        break;
    end

end

settling_time=k*dt
```

خروجی به صورت زیر است:

```
settling_time =

    8.0300
```

زمان صعود Rise time

زمان صعود زمانی می‌باشد که طول می‌کشد تا منحنی پاسخ پله از 10 به 90 درصد مقدار نهایی خود برسد.

برنامه زیر مقدار زمان صعود را محاسبه می‌کند:

```
clc
clear

num=[1];
den=[1 1 1];

dt=0.005;
t=0:dt:15;

[y,x,T]=step(num,den,t);

y_ss=y(numel(T));

k1=1;
while y(k1)<0.10*y_ss
    k1=k1+1;
end

k2=1;
while y(k2)<0.90*y_ss
    k2=k2+1;
end

Rise_Time=(k2-k1)*dt
```

خروجی به صورت زیر است:

```
Rise_Time =

    1.6350
```

زمان تاخیر Delay time

زمانی است که طول می‌کشد تا پاسخ پله برای اولین بار به 50 درصد مقدار نهاییش برسد.

برنامه زیر برای محاسبه زمان تاخیر می‌تواند بکار رود:

```
clc
clear

num=[1];
den=[1 1 1];

dt=0.005;
t=0:dt:15;

[y,x,T]=step(num,den,t);

y_ss=y(numel(T));

k=1;

while y(k)<0.50*y_ss
    k=k+1;
end

Delay_Time=(k-1)*dt
```

خروجی برنامه به صورت زیر است:

```
Delay_Time =

    1.2950
```

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه ششم:

بررسی پاسخ پله سیستم تک متغیره (قسمت سوم)

دستور stepinfo

در این جلسه به بررسی دستور stepinfo می‌پردازیم:

از این دستور برای استخراج ویزگی های پاسخ پله یک سیستم استفاده میکنیم. ورودی این تابع هم میتواند سیستم باشد و هم پاسخ پله یک سیستم.

```
S = stepinfo(y,t,yfinal)
S = stepinfo(y,t)
S = stepinfo(sys)
```

در حالت اول یعنی `S = stepinfo(y,t,yfinal)` فرض بر این است که پاسخ پله سیستم را به نحوی داریم مثلا از دستور `[y,x,t] = step(sys)` پاسخ پله سیستم را داریم و فقط میخواهیم مشخصات آن را بیابیم. همچنین اگر مقدار نهایی را وارد نکنیم (یعنی استفاده از دستور `S = stepinfo(y,t)`) در این صورت در محاسبات مقدار نهایی برابر با آخرین نمونه y در نظر خواهد گرفته خواهد شد.

در حالت دوم (یعنی `S = stepinfo(sys)`) خود تابع `stepinfo` مستقیما مشخصات پاسخ پله را بدست می آورد.

خروجی این تابع از نوع structure می باشد.

مثال :

```
clc
clear

num=[1 5];
den=[1 3 5];

[y,x,t]=step(num,den);
s=stepinfo(y,t,1)
```

نتیجه به صورت زیر میباشد:

```
s =


RiseTime: 0.8042

SettlingTime: 2.4750

SettlingMin: 0.9134

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674

PeakTime: 1.6272
```

همچنین برای نشان دادن هریک از مشخصات میتوانیم از دستور s.name استفاده کنیم مثلا :

```
s.RiseTime


ans =


0.8042
```

مثال:

در مثال زیر مقدار نهایی را وارد نمیکنیم:

```
num=[1 5];
den=[1 3 5];

[y,x,t]=step(num,den);
s=stepinfo(y,t)
```

نتیجه بصورت زیر خواهد بود:

```
s =


RiseTime: 0.7995

SettlingTime: 2.5336

SettlingMin: 0.9134

SettlingMax: 1.0674

Overshoot: 7.0751

Undershoot: 0

Peak: 1.0674

PeakTime: 1.6272
```

همانطور که دیده میشود در بعضی از مشخصات تفاوت هایی دیده میشود که بدلیل تفاوت در مقدار نهایی در نظر گرفته شده است.

مثال:

```
clc
clear

num=[1 5];
den=[1 3 5];
G=tf(num,den);
s=stepinfo(G)
```

نتیجه بصورت زیر است:

```
s =
```

```
RiseTime: 0.8042

SettlingTime: 2.4750

SettlingMin: 0.9134

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674

PeakTime: 1.6272
```

همچنین میتوان از مدل فضای حالت نیز استفاده کرد.

مثال:

```
clc
clear

num=[1 5];
den=[1 3 5];
[A,B,C,D]=tf2ss(num,den);
sys=ss(A,B,C,D);
s=stepinfo(sys)
```

نتیجه :

```
s =


RiseTime: 0.8042

SettlingTime: 2.4750

SettlingMin: 0.9134

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674
```

```
PeakTime: 1.6272
```

همانطور که میدانیم برای زمان برخاست و زمان نشست تعاریف (معیارهای) مختلفی وجود دارد.

در دستور stepinfo میتوان این معیارها را تغییر داد:

```
S = stepinfo(sys,'SettlingTimeThreshold',ST)
```

در این دستور معیار زمان نشست را تغییر میدهیم به طور پیش فرض مقدار 0.02 یعنی 2 درصد برای زمان نشت در نظر گرفته میشود اما در این دستور مقدار ST برای ان در نظر گرفته شده است.

مثال:

میخواهیم معیار زمان نشست را به 5 درصد تغییر دهیم:

```
clc
clear

num=[1 5];
den=[1 3 5];
[A,B,C,D]=tf2ss(num,den);
sys=ss(A,B,C,D);
S = stepinfo(sys,'SettlingTimeThreshold',0.05)
```

نتیجه:

```
S =


RiseTime: 0.8042

SettlingTime: 2.0241

SettlingMin: 0.9134

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674
```

```
PeakTime: 1.6272
```

میخواهیم معیار زمان برخاست را تغییر دهیم:

```
S = stepinfo(sys,'RiseTimeLimits',RT)
```

در این دستور RT بازه مورد نظر را میدهد. به طور پیش فرض ستور stepinfo برای زمان نشست معیار 10 درصد تا 90 درصد پاسخ نهایی را در نظر میگیرد.

در مثال زیر میخواهیم این معیار را به 5 درصد تا 95 درصد تغییر دهیم.

مثال:

```
clc
clear

num=[1 5];
den=[1 3 5];
[A,B,C,D]=tf2ss(num,den);
sys=ss(A,B,C,D);
S = stepinfo(sys,'RiseTimeLimits',[0.05,0.95])
```

نتیجه :

```
S =


RiseTime: 0.9471

SettlingTime: 2.4750

SettlingMin: 0.9583

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674

PeakTime: 1.6272
```

در مثال زیر هر دو معیار زمان نشست و زمان برخاست را تغییر میدهیم:

مثال:

```
clc
clear

num=[1 5];
den=[1 3 5];
[A,B,C,D]=tf2ss(num,den);
sys=ss(A,B,C,D);
S =
stepinfo(sys,'RiseTimeLimits',[0.05,0.95],'SettlingTimeThreshold
',0.05)
```

نتیجه:

```
S =


RiseTime: 0.9471

SettlingTime: 2.0241

SettlingMin: 0.9583

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674

PeakTime: 1.6272
```

توجه داشته باشید که در دستورهای اخیر به جای sys میتوانیم از اطلاعات پاسخ پله استفاده کنیم:

مثال:

```
clc
clear
```

```
num=[1 5];
den=[1 3 5];
[y,x,t]=step(num,den);

S =
stepinfo(y,t,1,'RiseTimeLimits',[0.05,0.95],'SettlingTimeThreshold',0.05)
```

نتیجه :

```
S =


RiseTime: 0.9471

SettlingTime: 2.0241

SettlingMin: 0.9583

SettlingMax: 1.0674

Overshoot: 6.7417

Undershoot: 0

Peak: 1.0674

PeakTime: 1.6272
```

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه هفتم:

قسمت اول:

بررسی پاسخ ضربه سیستم

منظور از پاسخ ضربه پاسخ سیستم با شرایط اولیه صفر به ورودی ضربه است.

همانطور که میدانیم پاسخ ضربه یک سیستم خطی مشتق پاسخ پله است. و بنابراین با توجه به روابط زیر میتوان برای بدست آوردن پاسخ ضربه G(s) از پاسخ پله s G(s) استفاده کرد:

$Y(s) = G(s)\, R(s)$

پاسخ پله: $R(s)= 1/s$: $Y(s)=G(s) / s$

پاسخ ضربه : $R(s)= 1$: $Y(s)=G(s) = s\, G(s) / s$

مثال :

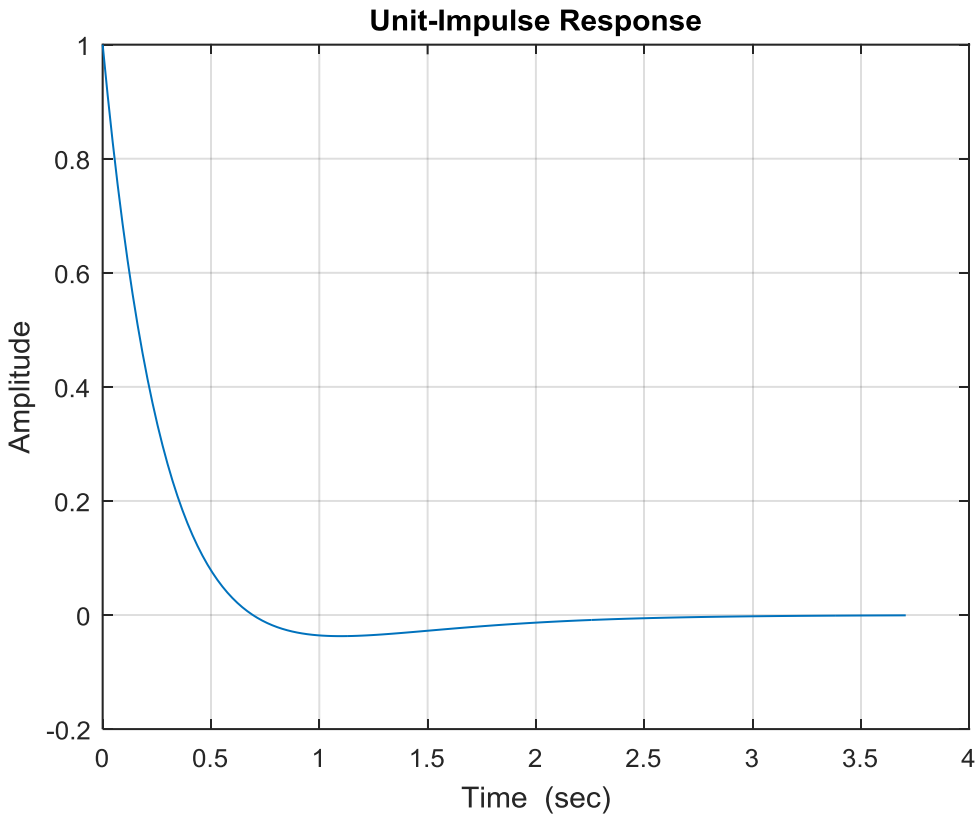
پاسخ ضربه سیستم زیر را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

```
clc
clear

s=tf('s');
G=(s+1)/(s+3)/(s+2);
[Y,t,x]=step(s*G);
plot(t,Y)
grid on
```

```
title('Unit-Impulse Response')
xlabel('Time  (sec)')
ylabel('Amplitude')
```



اما در متلب برای بدست آوردن پاسخ ضربه یک سیستم تابع impulse وجود دارد.

```
impulse(sys)
impulse(sys,Tfinal)
impulse(sys,t)
[y,t] = impulse(sys)
[y,t] = impulse(sys,Tfinal)
y = impulse(sys,t)
[y,t,x] = impulse(sys)
[y,t,x,ysd] = impulse(sys)
```

مثال :

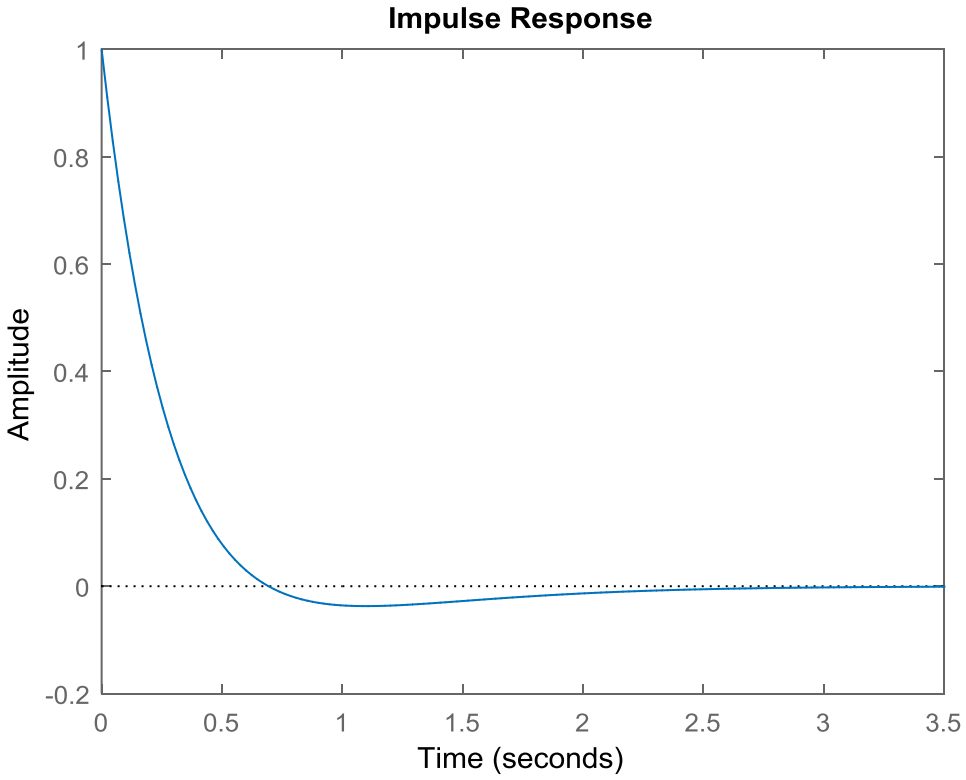
پاسخ ضربه سیستم زیر را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

```
clc
```

```
clear

s=tf('s');
G=(s+1)/(s+3)/(s+2);
impulse(G)
```



قسمت دوم :

بررسی پاسخ شیب سیستم

برای یافتن پاسخ شیب با توجه به این که پاسخ شیب انتگرال پاسخ پله است بنابراین میتوان برای بدست آوردن پاسخ شیب از پاسخ پله سیستم G(s) / s استفاده کرد.

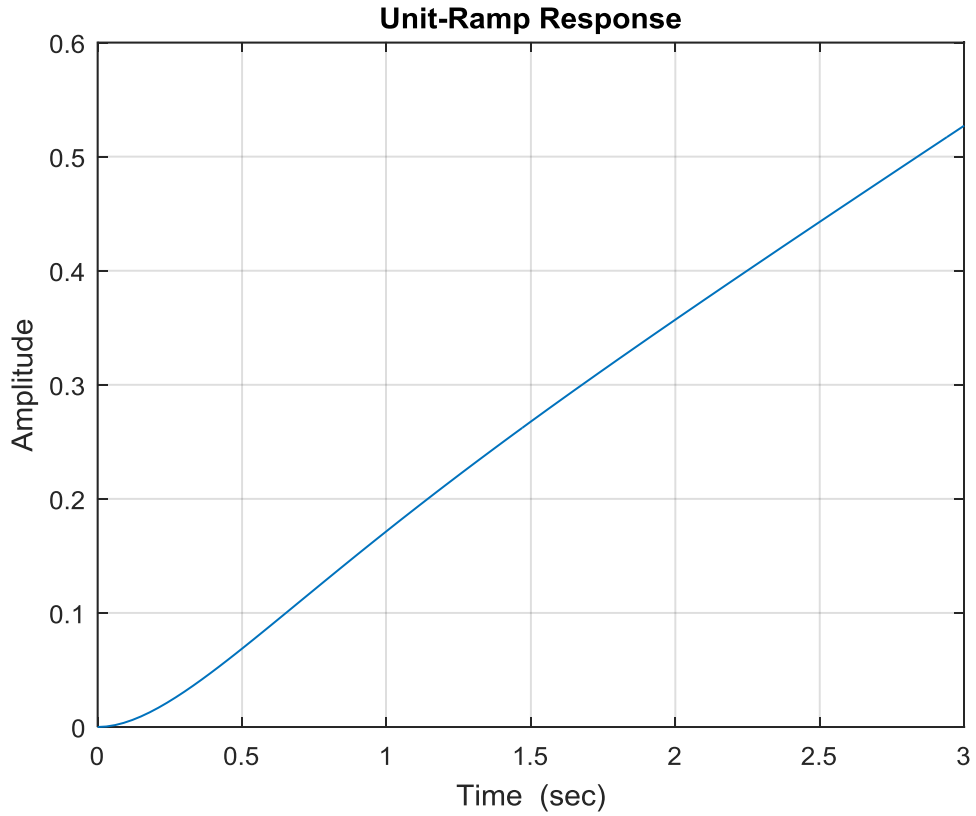
مثال :

پاسخ شیب سیستم زیر را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

```
clc
clear

s=tf('s');
G=(s+1)/(s+3)/(s+2);
[Y,t,x]=step(G/s,3);
plot(t,Y)
grid on
title('Unit-Ramp Response')
xlabel('Time  (sec)')
ylabel('Amplitude')
```



قسمت سوم:

پاسخ به ورودی دلخواه:

دستور lsim در متلب برای بدست آوردن پاسخ سیستم به ورودی دلخواه است:

```
lsim(sys,u,t)
lsim(sys,u,t,x0)
```

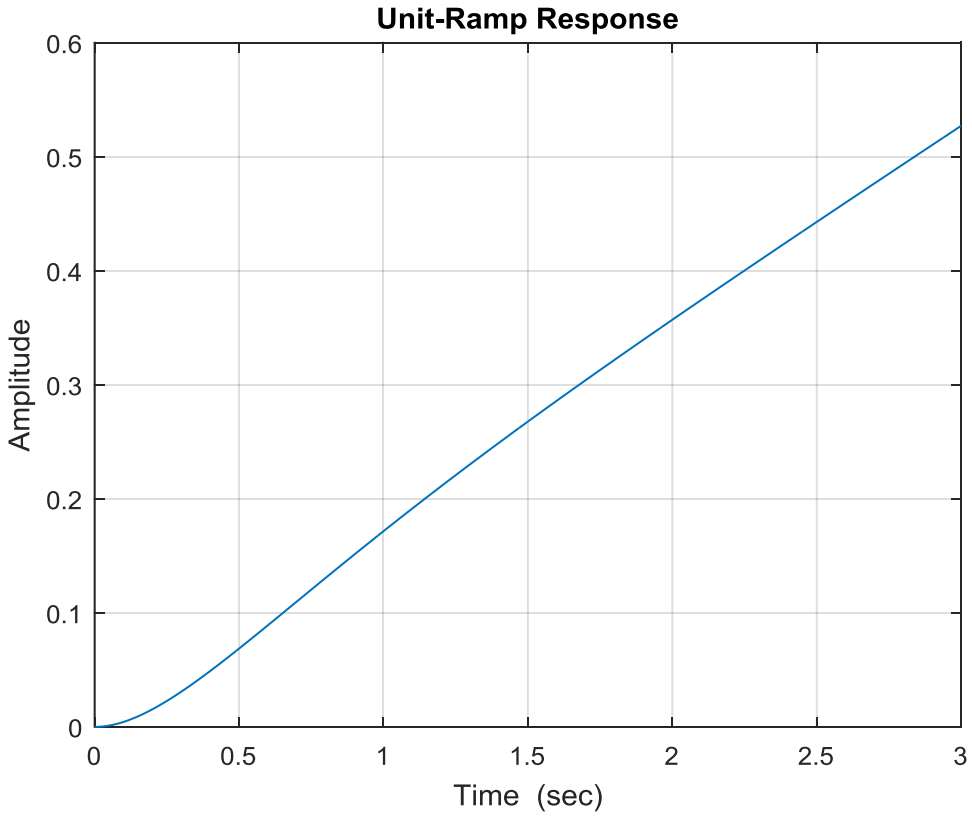
مثال :

پاسخ شیب سیستم زیر را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

```
clc
clear

s=tf('s');
G=(s+1)/(s+3)/(s+2);
t=0:0.01:3;
u=t;
[Y,T,X]=lsim(G,u,t);
plot(T,Y)
grid on
title('Unit-Ramp Response')
xlabel('Time  (sec)')
ylabel('Amplitude')
```



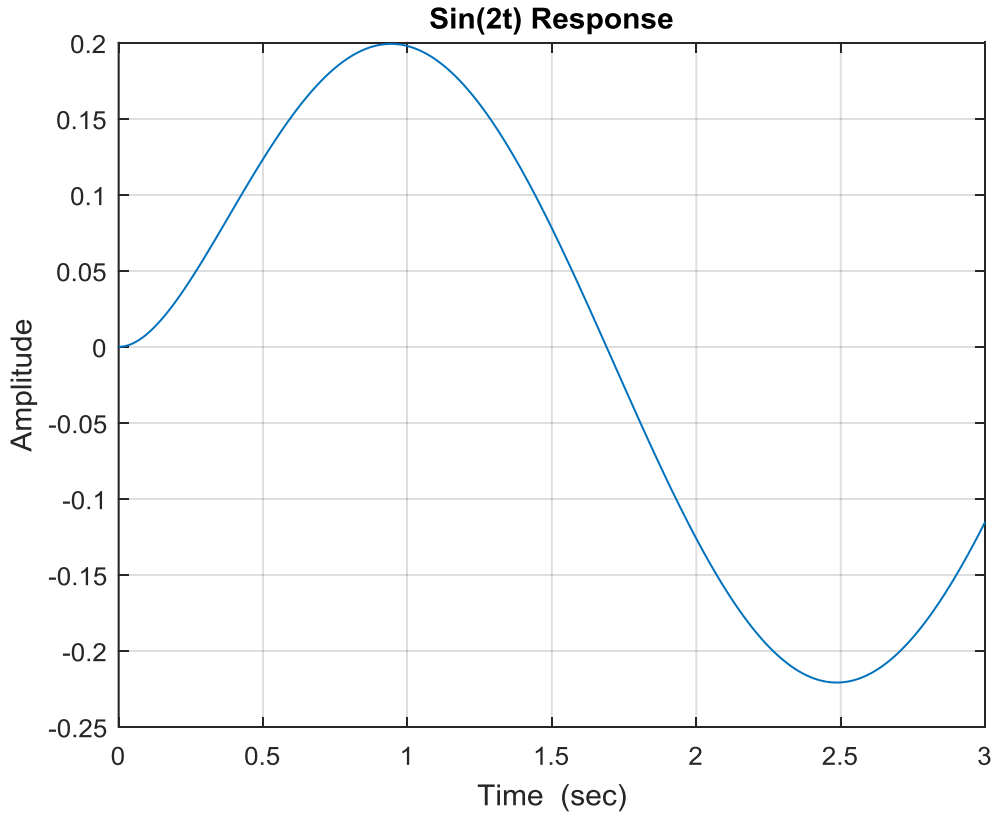
مثال:

پاسخ سیستم زیر به ورودی sin(2t) را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

```
clc
clear

s=tf('s');
G=(s+1)/(s+3)/(s+2);
t=0:0.01:3;
u=sin(2.*t);
[Y,T,X]=lsim(G,u,t);
plot(T,Y)
grid on
title('Sin(2t) Response')
xlabel('Time  (sec)')
ylabel('Amplitude')
```



مثال:

پاسخ سیستم زیر به ورودی sin(2t) و شرایط اولیه y=0.1 را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

پاسخ:

ابتدا باید یک تحقق از سیستم بدست آوریم:

```
 [A,B,C,D]=tf2ss([1 1],[1 5 6])
```

```
A =

    6-     5-
    0      1


B =

    1
    0


C =

    1     1
```

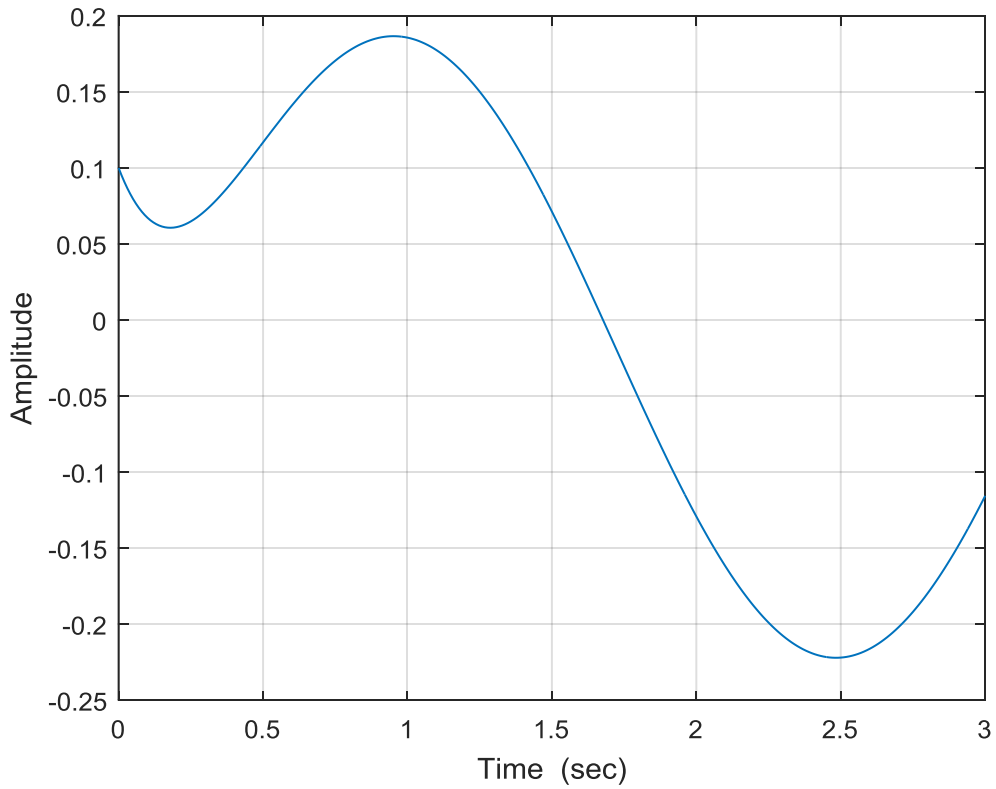
```
D =


0
```

بنابراین شرایط اولیه تمام پاسخ های دستگاه x1+x2=0.1 می تواند باشد.

```
clc
clear

[A,B,C,D]=tf2ss([1 1],[1 5 6]);
sys=ss(A,B,C,D);
t=0:0.01:3;
u=sin(2.*t);
[Y,T,X]=lsim(sys,u,t,[0.05,0.05]);
plot(T,Y)
grid on
xlabel('Time  (sec)')
ylabel('Amplitude')
```



تذکر مهم :

در مثال فوق یک تحقق از سیستم را بدست آوردیم و همچنین شرایط اولیه را برابر جواب دلخواهی از دستگاه x1+x2=0.1 در نظر گرفتیم. با تغییر تحقق و یا با در نظر گرفتن جوابی دیگر از دستگاه x1+x2=0.1 پاسخ سیستم (پاسخ به شرایط اولیه و در نتیجه پاسخ کلی) تغییر خواهد کرد. توجه به این نکته لازم است که ما در عمل شرایط اولیه را برای یک تحقق مشخص داریم و بنابراین در عمل شرایط اولیه اگر مشخص باشد باید در مدل فضای حالت باشیم.

قسمت چهارم:

یافتن پاسخ به شرایط اولیه

از دستور initial برای بدست آوردن پاسخ به شرایط اولیه استفاده میکنیم:

```
initial(sys,x0)
initial(sys,x0,Tfinal)
initial(sys,x0,t)
[y,t,x] = initial(sys,x0)
[y,t,x] = initial(sys,x0,Tfinal)
[y,t,x] = initial(sys,x0,t)
```

مثال :

پاسخ سیستم زیر به شرایط اولیه y=0.1 را بیابید:

$$G(s) = \frac{s+1}{(s+3)(s+2)}$$

پاسخ:

ابتدا باید یک تحقق از سیستم بدست آوریم:

```
[A,B,C,D]=tf2ss([1 1],[1 5 6])
```

```
A =
```

```
6-     5-

0      1



B =


1

0



C =


1      1



D =

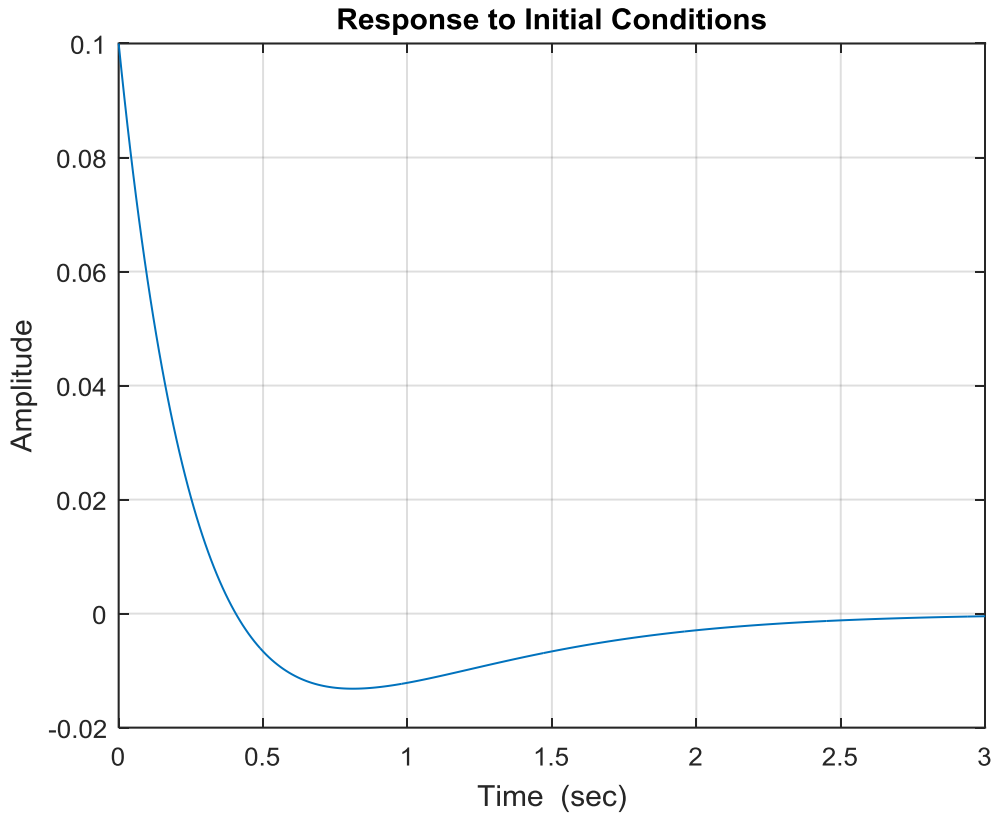

0
```

بنابراین شرایط اولیه تمام پاسخ های دستگاه x1+x2=0.1 می تواند باشد.

```
clc
clear

[A,B,C,D]=tf2ss([1 1],[1 5 6]);
sys=ss(A,B,C,D);
t=0:0.01:3;
[Y,T,X]=initial(sys,[0.05,0.05],t);
plot(T,Y)
grid on
title('Response to Initial Conditions')
xlabel('Time  (sec)')
```

```
ylabel('Amplitude')
```



مثال:

قاعده جمع آثار را برای سیستم زیر و نیز ورودی sin(2t) و شرایط اولیه X0=[0.1,0.3] بررسی کنید.

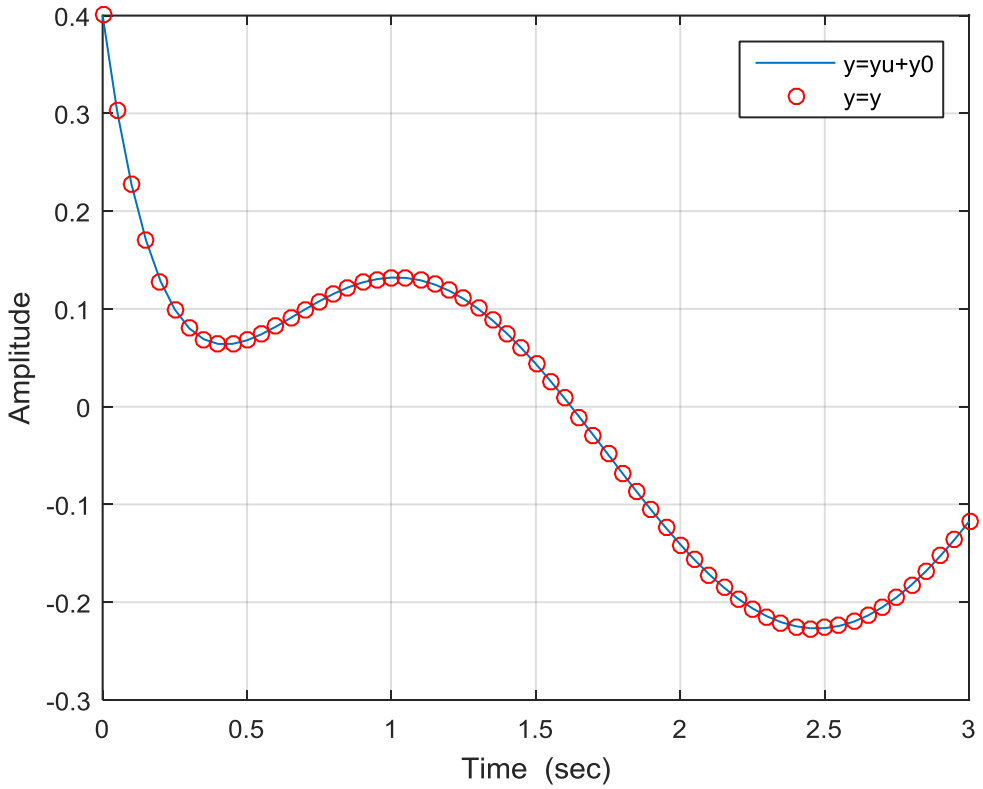
$$A=\begin{bmatrix}-5 & -6\\ 1 & 0\end{bmatrix};\ B=\begin{bmatrix}1\\0\end{bmatrix};\ C=[1 \quad 1];\ D=0$$

```
clc
clear

A=[-5 -6;1 0];
B=[1;0];
C=[1 1];
D=0;
sys=ss(A,B,C,D);
t=0:0.05:3;
u=sin(2.*t);
[Y0,T0,X0]=initial(sys,[0.1,0.3],t);
[Yu,Tu,Xu]=lsim(sys,u,t);
Y=Yu+Y0;
```

```
T=Tu;
plot(T,Y)

hold on
[Yy,Tt,Xx]=lsim(sys,u,t,[0.1,0.3]);
plot(Tt,Yy,'or')
grid on
xlabel('Time  (sec)')
ylabel('Amplitude')
legend('y=yu+y0','y=y')
```



همانظور که میبینیم هر دو نمودار روی هم واقع شده اند.

قسمت پنجم:

استفاده از ode45 برای یافتن پاسخ یک سیستم به ورودی و شرایط اولیه دلخواه:

در این قسمت میخواهیم پاسخ یک سیستم کلی (خطی – غیر خطی – متغیر با زمان) به ورودی و شرایط اولیه دلخواه را بدست آوریم.

برای این کار سیستم را توسط مدل فضای حالت مدل میکنیم. در واقع به طور ساده معدلات دیفرانسیل حاکم بر سیستم را مینویسیم. سپس از دستور ode45 برای حل این معادلات استفاده میکنیم.

مثال:

پاسخ سیستم زیر به ورودی sin(2t) و شرایط اولیه X0=[0.1,0.3] را بیابید.

$$A = \begin{bmatrix} -5 & -6 \\ 1 & 0 \end{bmatrix};\ B = \begin{bmatrix} 1 \\ 0 \end{bmatrix};\ C = [1 \quad 1];\ D = 0$$

حل:

ابتدا یک تابع مینویسیم که سیستم را مدل میکند:

```
function dX=sys(t,X)

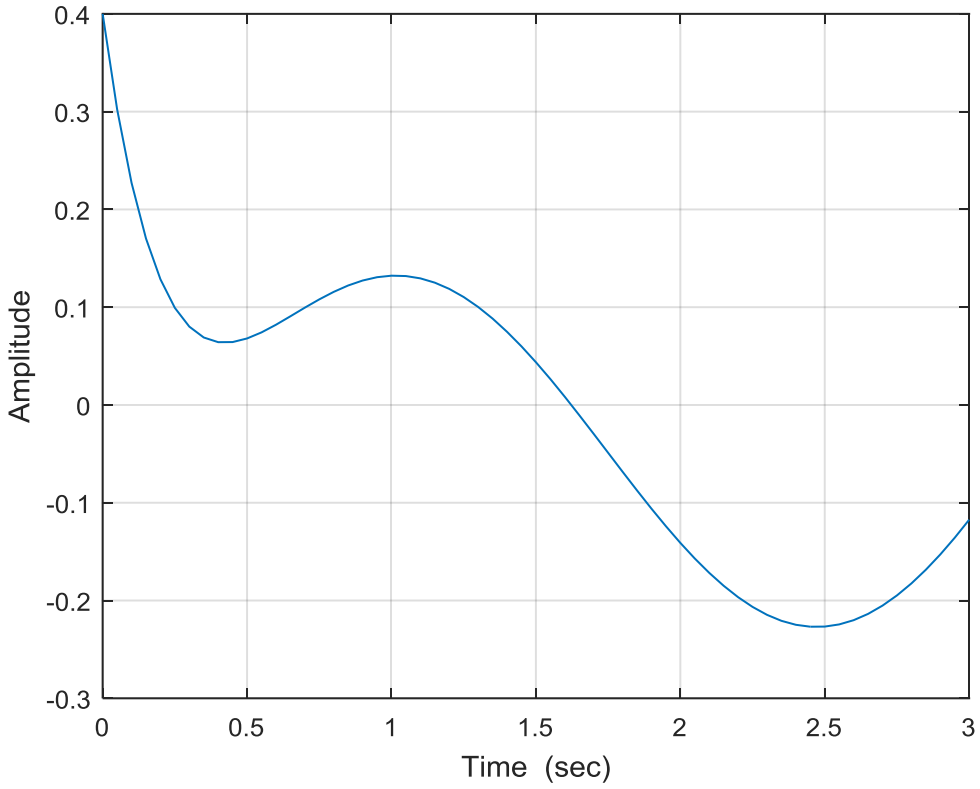
dX=zeros(2,1);

u=sin(2*t);

dX(1)=-5*X(1)-6*X(2)+u;
dX(2)=X(1);
```

سپس از دستور ode45 استفاده میکنیم:

```
X0=[0.1,0.3];
t=0:0.05:3;
[T,X]=ode45(@sys,t,X0);
y=X(:,1)+X(:,2); % y=c*x+d*u
plot(T,y)
grid on
xlabel('Time  (sec)')
ylabel('Amplitude')
```



تذکر:

برتری روش اخیر این است که هیچ محدودیتی در نوع سیستم نداریم. یعنی سیستم میتواند غیرخطی نیز باشد.

مثال:

سیستمی با معادله دیفرانسیل غیرخطی زیر توصیف میشود. پاسخ آن را به ورودی sin(2t) بیابید. فرض کنید شرایط اولیه درصورت در نظر گرفتن x و مشتق آن به عنوان حالت های سیستم برابر [1,-2] باشد. و خروجی سیستم x میباشد.

$$\ddot{x} = -x^2\dot{x} + u$$

پاسخ:

ابتدا مدل فضای حالت آن را میابیم:

با فرض متغیرهای حالت به صورت $x_1 = x$ و $x_2 = \dot{x}$ داریم.

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -x_1^2 x_2 + u$$

حال که مدل فضای حالت را داریم مانند مثال قبل یک تابع برای توصیف آن نوشته و سپس از دستور ode45 استفاده میکنیم:

```
function dX=sys(t,X)

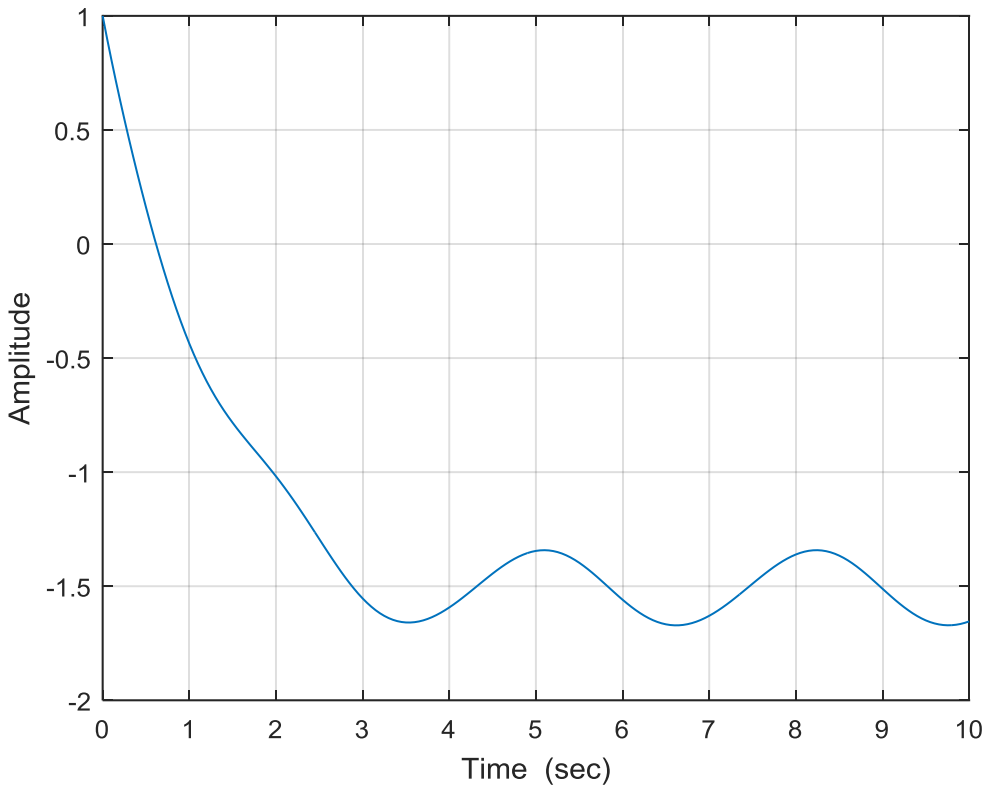
dX=zeros(2,1);

u=sin(2*t);

dX(1)=X(2);
dX(2)=-X(1)^2*X(2)+u;
```

حال از دستور ode45 استفاده میکنیم:

```
X0=[1,-2];
t=0:0.05:10;
[T,X]=ode45(@sys,t,X0);
y=X(:,1); %y=x
plot(T,y)
grid on
xlabel('Time  (sec)')
ylabel('Amplitude')
```


1
0.5
0
-0.5
-1
-1.5
-2
Amplitude
0
1
2
3
4
5
6
7
8
9
10
Time (sec)

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه هشتم:

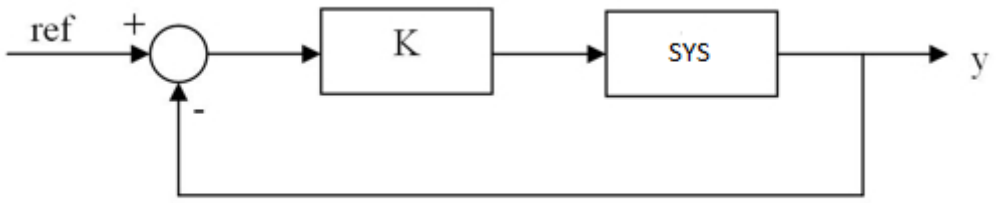
رسم مکان هندسی ریشه‌ها (قسمت اول)

در این جلسه رسم مکان هندسی ریشه ها را یاد خواهیم گرفت(قسمت اول)

برای رسم مکان هندسی ریشه ها از دستور rlocus استفاده میکنیم:

```
rlocus(sys)
rlocus(sys,k)
[r,k] = rlocus(sys)
r = rlocus(sys,k)
```

شکل استاندارد فیدبک برای رسم مکان هندسی ریشه ها به شکل زیر است:



Standard Closed Loop Feedback Configuration

که در آن K بهره سیستم میباشد. با مشخص بودن K به صورت یک عدد ثابت سیستم حلقه بسته دارای n قطب خواهد بود. با تغییر مقدار K و بدست آوردن قطب های متناظر و سپس رسم این قطب ها در صفحه مختلط نمودار مکان هندسی ریشه ها بدست می آید.

مثال :

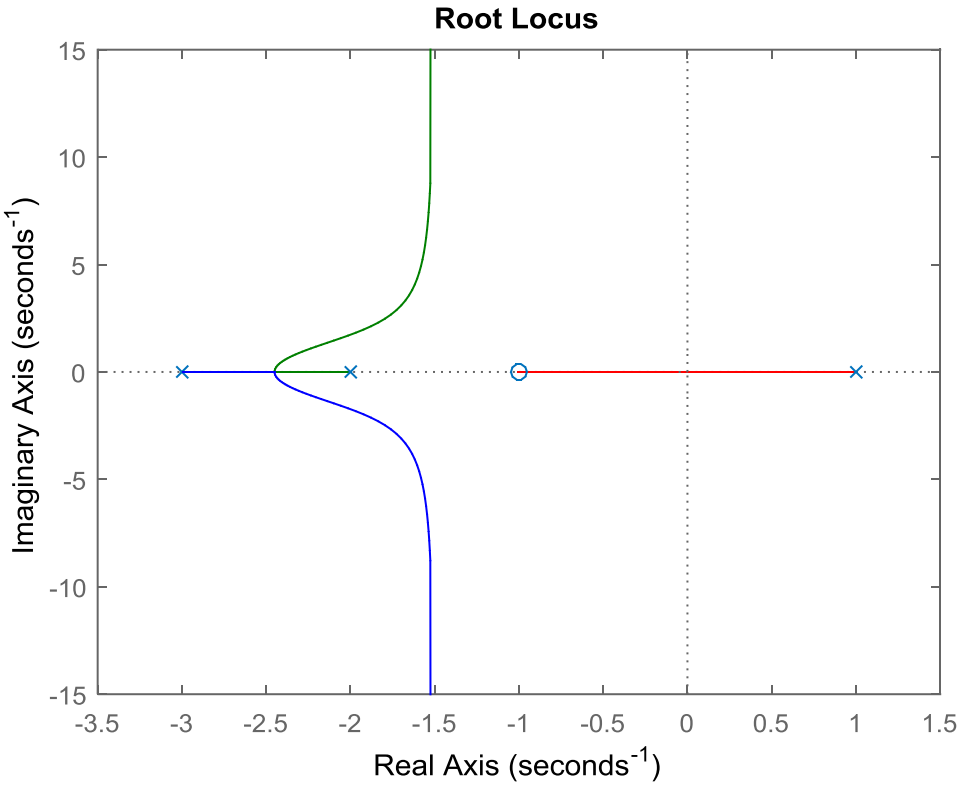
مکان هندسی ریشه های سیستمی با تابع تبدیل حلقه باز زیر را رسم کنید:

$$P(s) = \frac{s+1}{(s+3)(s+2)(s-1)}$$

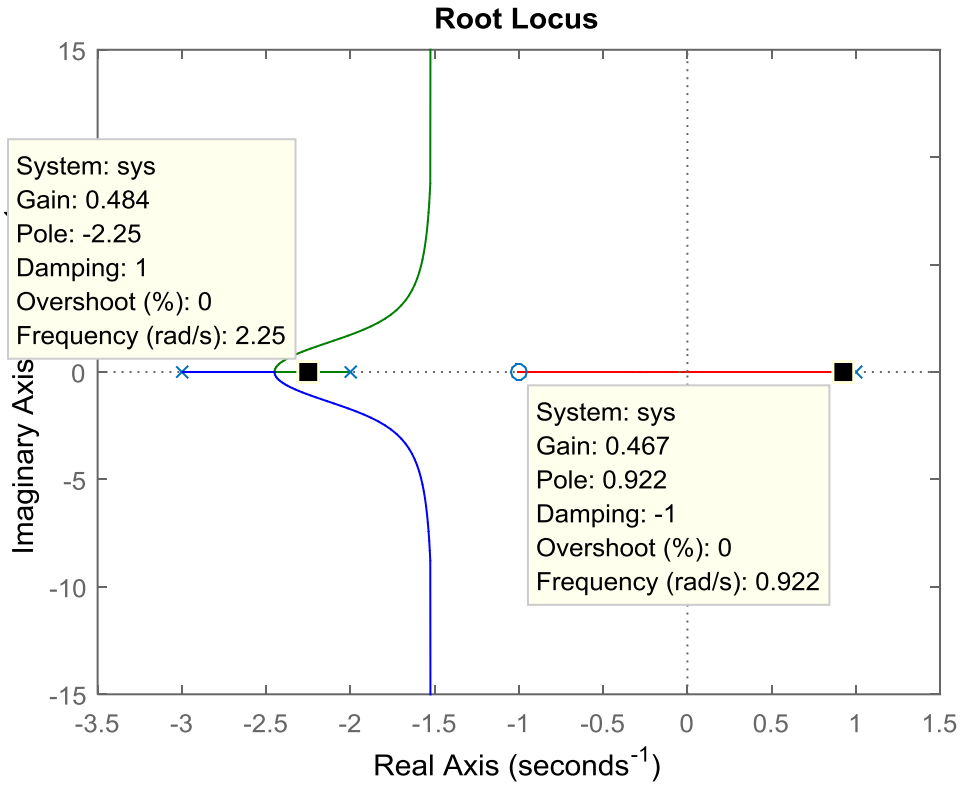
```
clc
clear
close all

num=[1 1];
den=[1 4 1 -6];

rlocus(num,den)
```



با کلیک کردن روی هر نقطه از نمودار میتوان بهره متناظر و نیز قطب را دید:



در این حالت متلب خود به صورت اتوماتیک بردار بهره k را تعیین میکند اما میتوانیم خودمان به طور مستقیم این بردار را تعیین کنیم:

مثال:

مکان هندسی ریشه های سیستمی با تابع تبدیل حلقه باز زیر را برای بردار بهره K=[0 10] با گام های 0.1 رسم کنید:

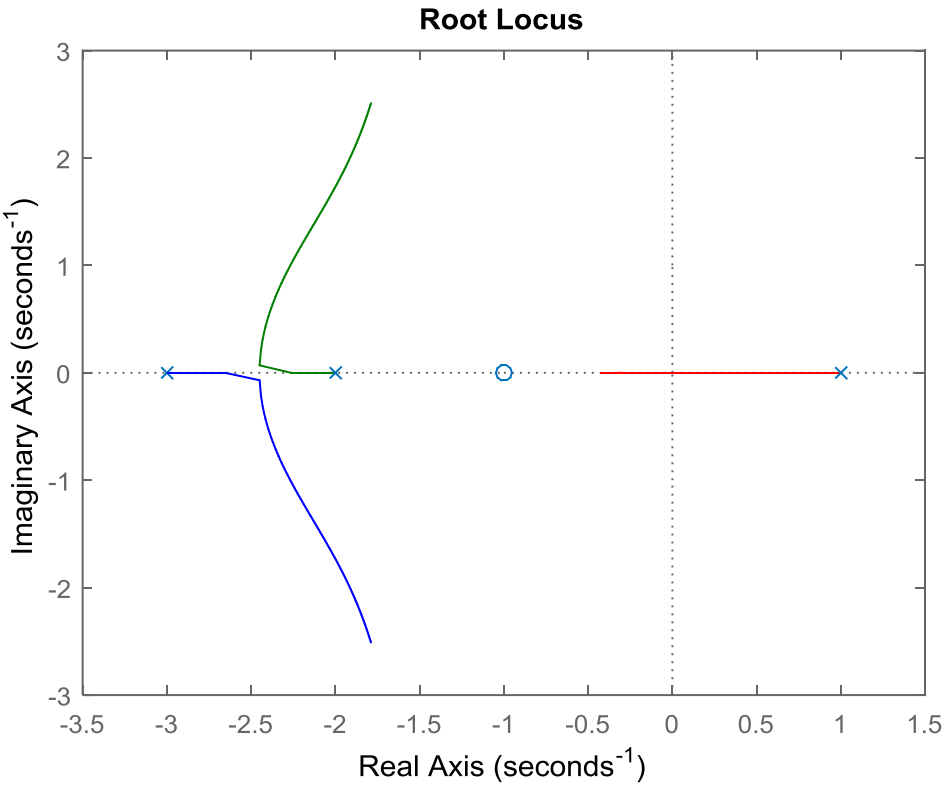
$$P(s) = \frac{s+1}{(s+3)(s+2)(s-1)}$$

```
clc
clear
close all

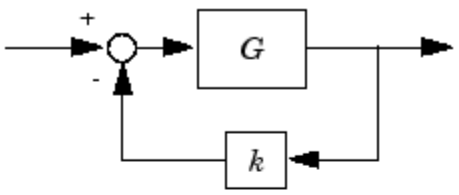
num=[1 1];
den=[1 4 1 -6];
```

```
k=0:0.1:10;

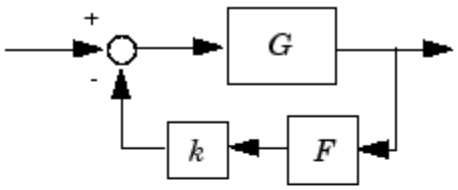
rlocus(num,den,k)
```



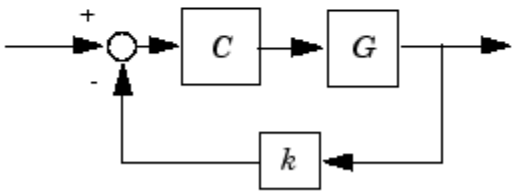
اگر سیستم در قالب استاندارد فوق نباشد باید آنرا به گونه ای به قالب استاندارد تبدیل کنیم:



sys = G



sys = F * G



sys = G * C

مثلا اگر فیدبک مثبت باشد میتوان F=-1 در نظر گرفت یعنی مکان هندسی ریشه های -G(s) را رسم میکنیم.

یا مثلا بخواهیم مکان هندسی ریشه ها را برای بهره های منفی رسم کنیم باز هم مکان هندسی ریشه های -G(s) را رسم میکنیم.

مثال:

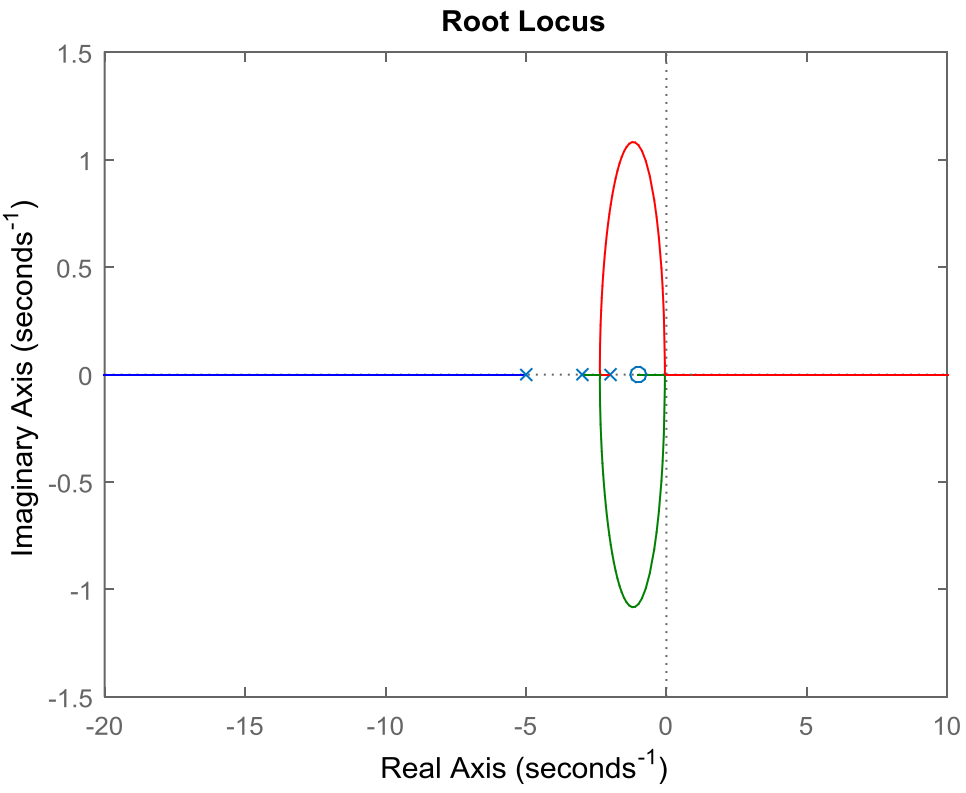
مکان هندسی ریشه های سیستمی با تابع تبدیل حلقه باز زیر را برای بهره ها منفی رسم کنید:

$$P(s) = \frac{s+1}{(s+3)(s+2)(s+5)}$$

```
clc
clear
close all

num=-[1 1]; % -G(s)
den=[1 10 31 30];

rlocus(num,den)
```



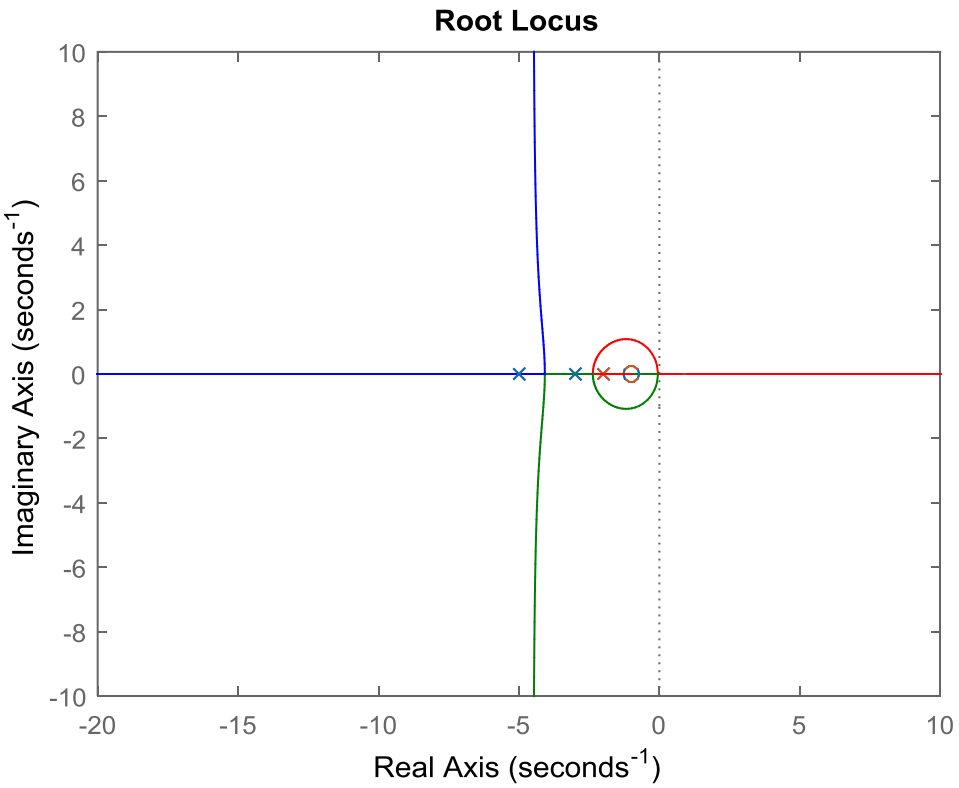
مثال:

مکان هندسی ریشه های سیستمی با تابع تبدیل حلقه باز زیر را برای بهره ها منفی و مثبت رسم کنید:

$$P(s) = \frac{s+1}{(s+3)(s+2)(s+5)}$$

```
clc
clear
close all

for i=[-1 1]
num=i*[1 1];
den=[1 10 31 30];
rlocus(num,den)
hold on
end
```



اگر بخواهیم ریشه ها را به ازای مقادیر مختلف بهره بدانیم میتوانیم به صورت زیر از دستور rlocus استفاده کنیم.

```
[r,k] = rlocus(sys)
```

در این حالت K یک بردار است با m درایه و r یک ماتریس با سایز m در n میباشد که n در واقع مرتبه معادله مشخصه (تعداد قطب های حلقه بسته) میباشد.

مثال:

قطب های حلقه بسته سیستمی با تابع تبدیل حلقه باز زیر را بدست آورید:

$$P(s) = \frac{s+1}{(s+3)(s+2)(s-1)}$$

```
clc
clear
close all

num=[1 1];
den=[1 4 1 -6];

[r,k]=rlocus(num,den);
```

بنابراین مقادیر بهره در بردار k و ریشه های متناظر در ماتریس r قرار میگیرند.

همچنین میتوانیم بردار k را خود تعیین کنیم:

```
num=[1 1];
den=[1 4 1 -6];
k=0:0.1:10;

r=rlocus(num,den,k);
```

حال اگر بخواهیم مکان هندسی را رسم کنیم کافی است r را رسم کنیم:

```
clc
clear
close all

num=[1 1];
```

```
den=[1  4  1  -6];

[r,k]=rlocus(num,den);

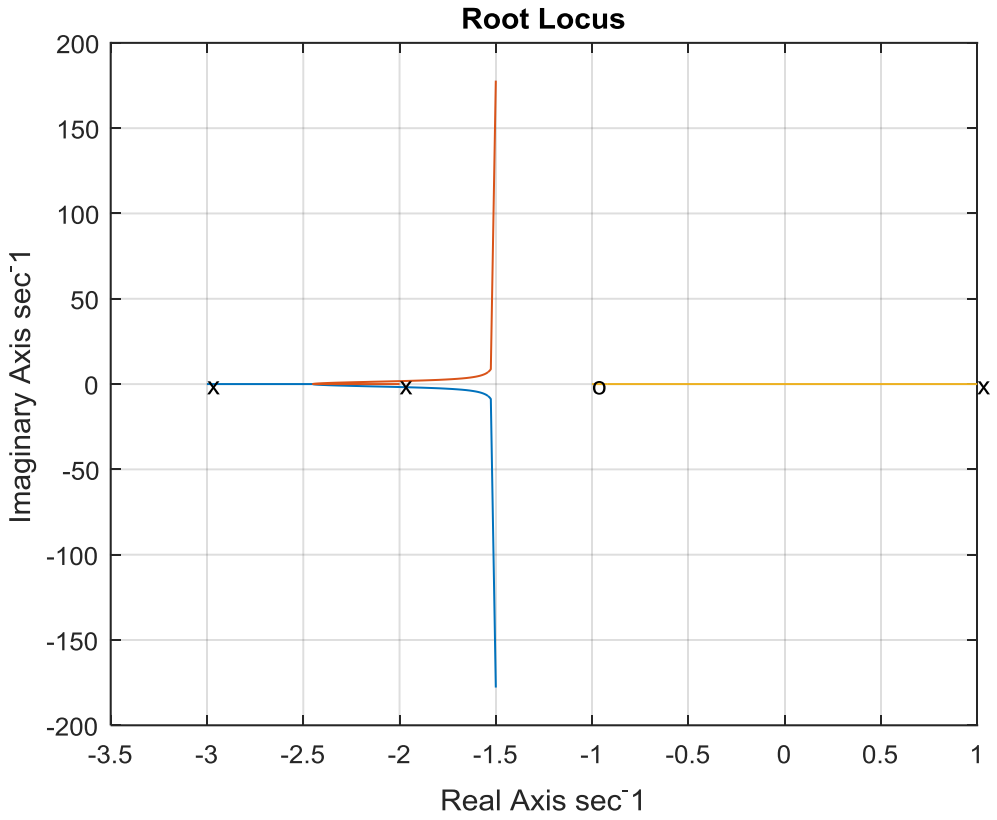
plot(r);

grid on
title('Root Locus');
xlabel('Real Axis sec^-1');
ylabel('Imaginary Axis sec^-1');

%%
hold on

k=0;
r=rlocus(num,den,k);
text(real(r),imag(r),'x')

k=inf;
r=rlocus(num,den,k);
text(real(r),imag(r),'o')
```



در نوشتن برنامه فوق برای مشخص کردن قطب ها و صفرهای حلقه باز از این ویژگی که در بهره صفر در قطب های حلقه باز و در بهره بینهایت در صفرهای حلقه باز هستیم استفاده نمودیم. که بیشتر جنبه آموزشی داشته باشد اما برای دقیق تر بودن همانطور که در جلسه اول توضیح داده شد بهتر است از دستور pzmap استفاده کنیم:

```
clc
clear
close all

num=[1 1];
den=[1 4 1 -6];


r=rlocus(num,den);

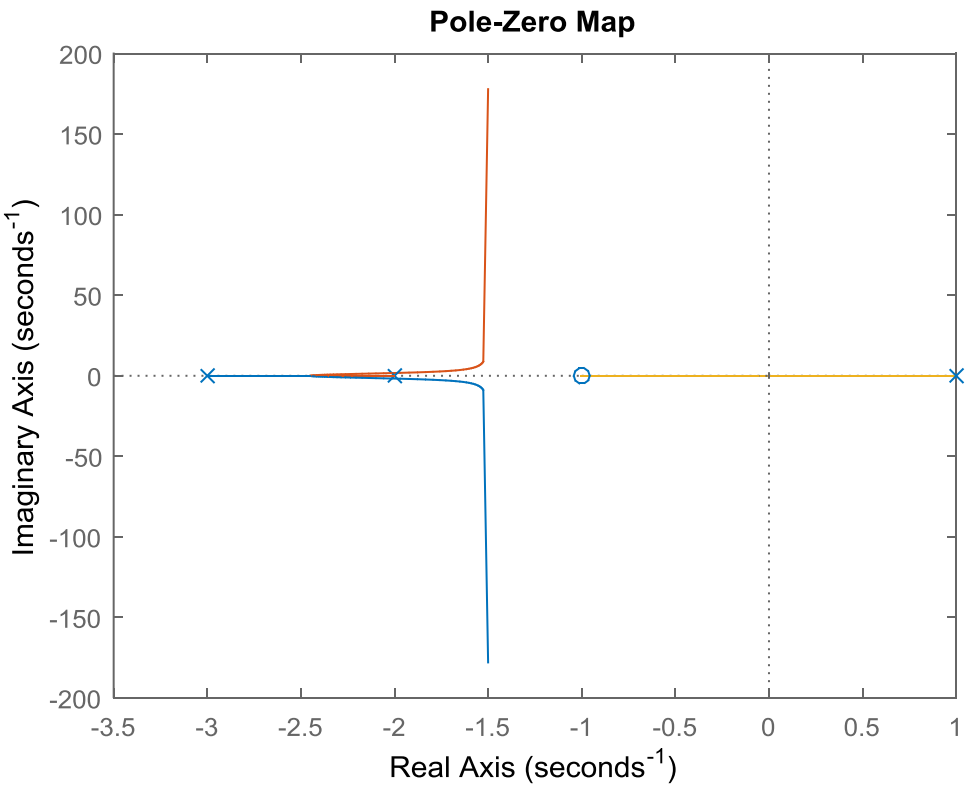
plot(r);

grid on
title('Root Locus');
```

```
xlabel('Real Axis sec^-1');
ylabel('Imaginary Axis sec^-1');

%%
hold on

pzmap(num,den)
```



به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه نهم :

رسم مکان هندسی ریشه ها (قسمت دوم)

جلسه قبل رسم مکان هندسی ریشه ها را به کمک دستور rlocus یاد گرفتیم .

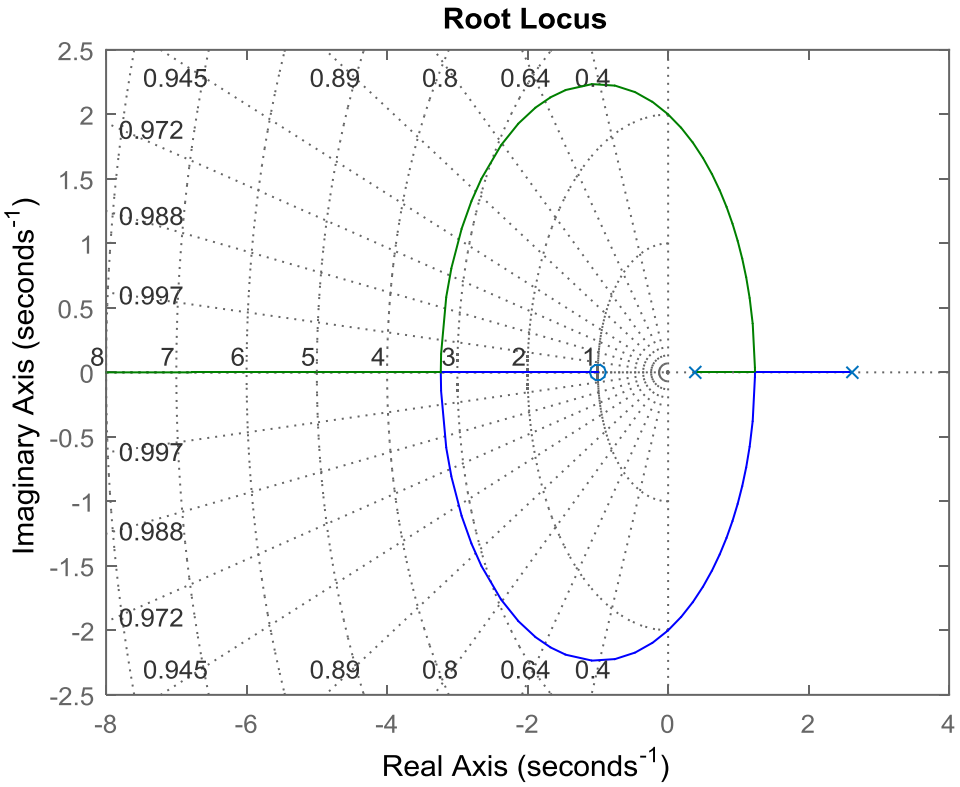
برای رسم شبکه قطبی (مکان ζ ثابت و ω_n ثابت) از دستور grid و یا sgrid استفاده میکنیم تفاوت sgrid با grid در این است که در sgrid میتوانیم نسبت میرایی و فرکانس طبیعی نامیرا را مشخص کنیم.

مثال:

```
clc
clear
close all

num=[1 1];
den=[1 -3 1];

rlocus(num,den)
sgrid % or grid
```

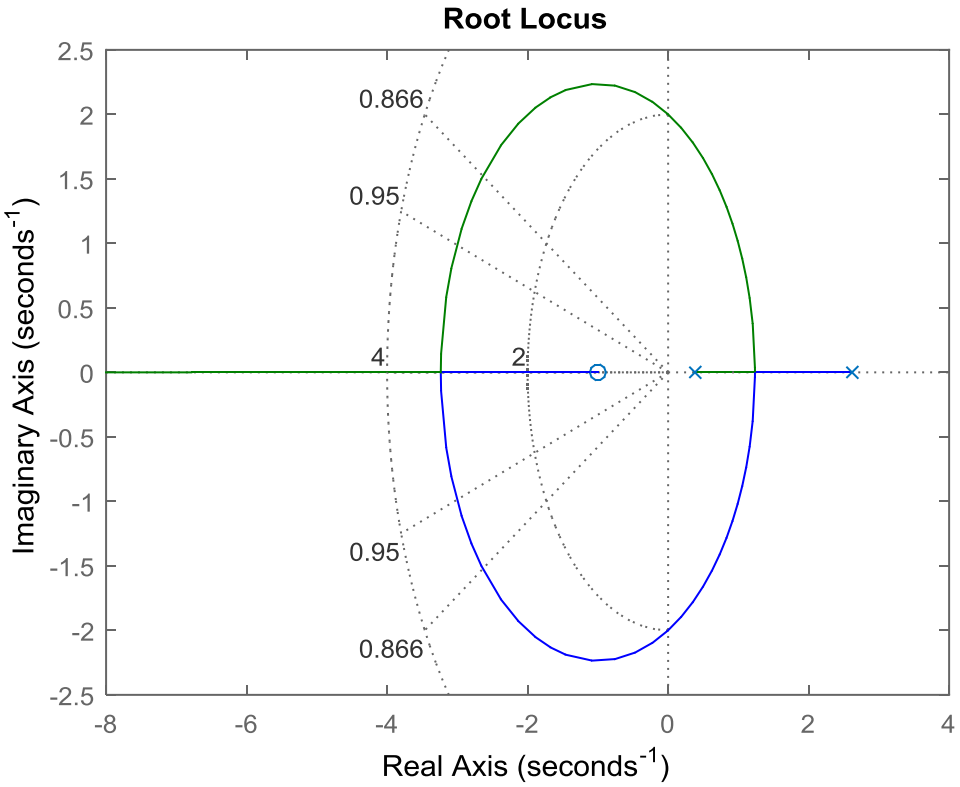


حال اگر بخواهیم فقط نسبت میرایی 0.866 و 0.95 و فرکانس نامیرایی 2 و 4 را رسم کنیم به صورت زیر عمل میکنیم:

```
clc
clear
close all

num=[1 1];
den=[1 -3 1];

rlocus(num,den)
sgrid([0.866,0.95],[2,4])
```



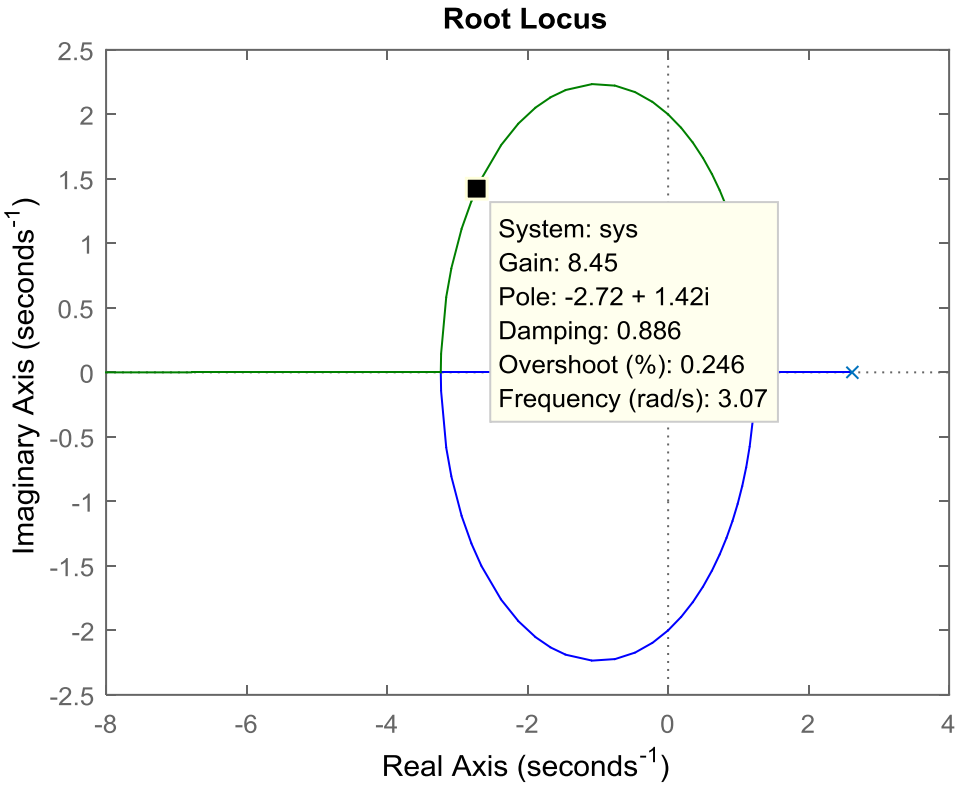
واضح است که اگر بخواهیم فقط خطوط مربوط به نسبت میرایی و یا فقط دوایر مربوط به فرکانس نامیرایی را رسم کنیم کافی است که بردار مربوط به دیگری را در ورودی تابع sgrid تهی انتخاب کنیم مثلا دستور

```
sgrid([0.866,0.95],[])
```

فقط خطوط میرایی ثابت را رسم میکند.

پیدا کردن بهره در نقاط مختلف:

با کلیک کردن روی هر نقطه از نمودار مکان هندسی میتوان بهره متناظر را یافت:



همان طور که میبینیم با کلیک کردن روی هر نقطه بهره و قطب متناظر و همچنین بعضی از مشخصه های پاسخ پله به ازی آن قطب نمایش داده میشود. اما اشکال این کار این است که همه قطب های متناظر با آن بهره را نمیدهد.

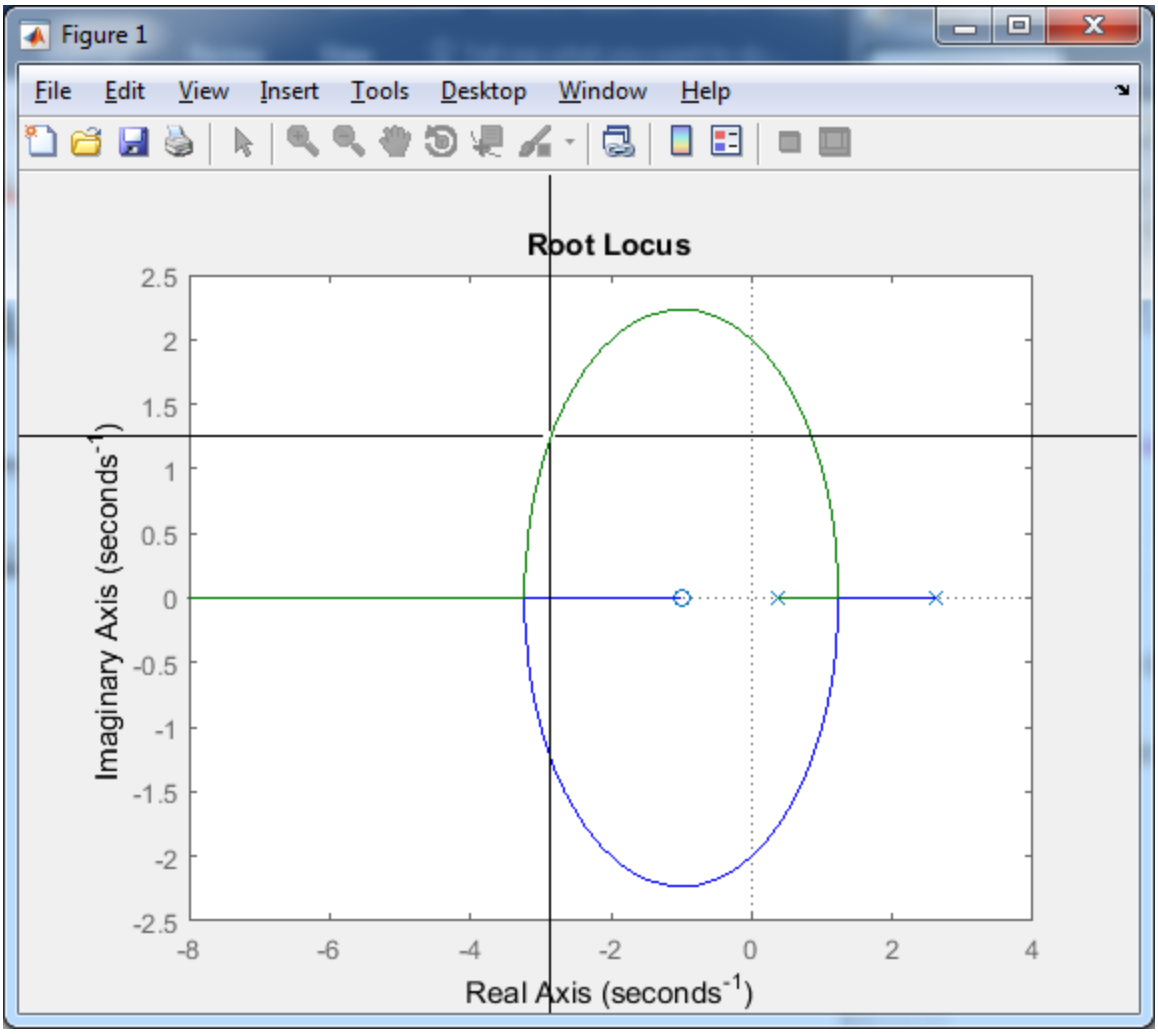
برای بدست آوردن همه قطب ها باید از دستور rlocfind استفاده کنیم:

```
[k,r]=rlocfind(num,den)
```

```
[k,r]=rlocfind(A,B,C,D)
```

این دستور را باید پس از رسم مکان هندسی ریشه ها (به کمک دستور rlocus) اجرا کنیم. با این کار به صورت گرافیکی یک نقطه از مکان هندسی را انتخاب میکنیم. در خروجی قطب های حلقه بسته و بهره متناظر با آن قطب (نقطه) انتخاب شده نمایش داده میشود.

مثال:



با کلیک کردن در هر نقطه قطب انتخاب شده و سایر قطب های همراه و همچنین بهره متناظر در خروجی نمایش داده میشود:

```
selected_point =


  -2.7251 + 1.4241i



k =
```

```
    8.4488


r =


  -2.7244 + 1.4235i

  -2.7244 - 1.4235i
```

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه دهم :

رسم نمودار بوده (قسمت اول) :

برای رسم نمودار بوده باید نمودار دامنه و فاز سیستم را بیابیم.

در متلب دستور bode برای این منظور تعبیه شده است اما برای آموزش بهتر مفاهیم نمودار بوده ابتدا به نوشتن برنامه برای رسم دیاگرام بوده می‌پردازیم:

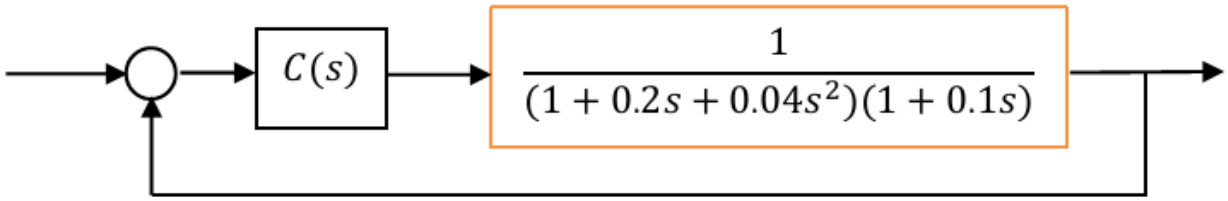
در عمل برای این منظور فرکانس های اصلی سیستم را میابیم و با دادن ورودی سینوسی با فرکانس از حدود 0.1 کوچک ترین تا 10 برابر بزرگ ترین آن ها در چندین نقطه مناسب ،بهره(نسبت دامنه خروجی به دامنه ورودی) و اختلاف فاز (اختلاف زاویه بین خروجی و ورودی) را اندازه گیری می‌کنیم.با استفاده از نقاط بدست آمده دیاگرام بود را در صفحه نیمه لگاریتمی رسم می‌کنیم.

در اینجا توابع فاز و دامنه را میابیم و سپس مقادیر آن‌ها را در فرکانس‌های مختلف یافته و رسم می‌کنیم.

مثال زیر این روش را تشریح می‌کند:

مثال:

برای دیاگرام زیر و با فرض c(s)=1 دیاگرام بوده را رسم کنید:



$$G(s) = \frac{1}{(1 + 0.2s + 0.04s^2)(1 + 0.1s)}$$

$$G(j\omega) = \frac{1}{(1 + 0.2j\omega - 0.04\omega^2)(1 + 0.1j\omega)}$$

$$20\log|G(j\omega)| = -20\ \log|1 + 0.2j\omega - 0.04\omega^2| - 20\ \log|1 + 0.1j\omega|$$

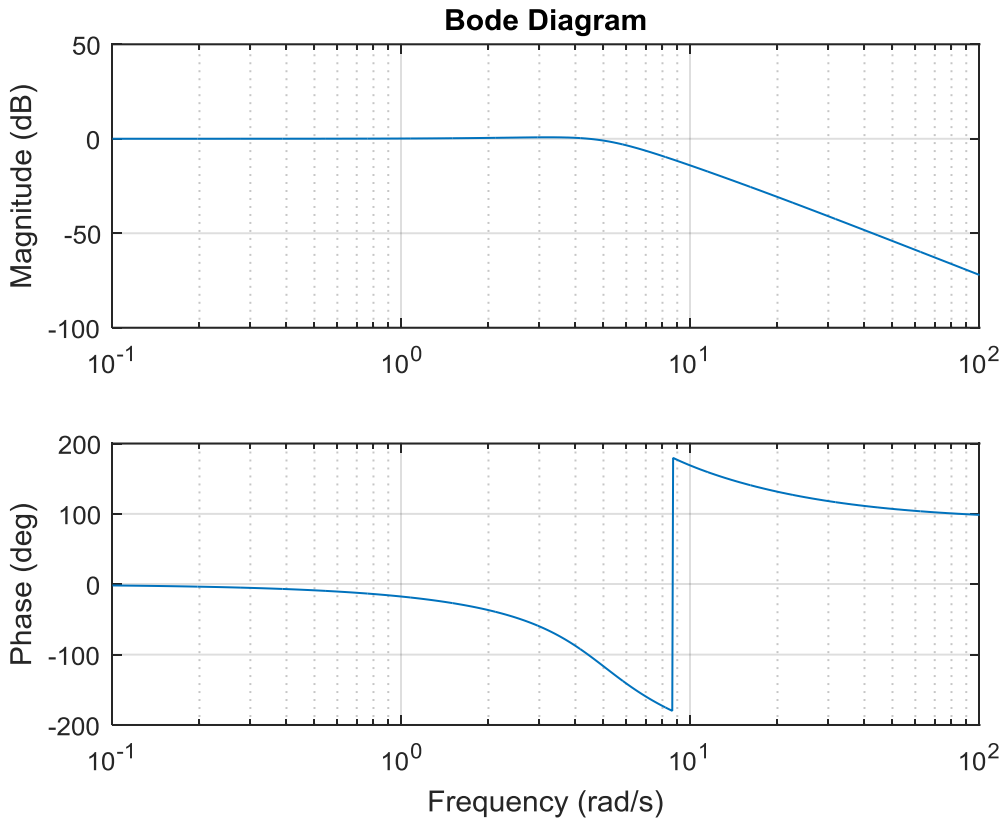
$$\measuredangle G(j\omega) = -\measuredangle(1 + 0.2j\omega - 0.04\omega^2) - \measuredangle(1 + .01j\omega)$$

برنامه متلب زیر این کار را انجام میدهد.

```
clc
clear
close all

w=logspace(-1,2,1000);
for n=1:length(w);
m=20*log10(abs(1/((1+0.2*w(n)*1i-
0.04*(w(n))^2)*(1+0.1*w(n)*1i))));
ph=angle(1/((1+0.2*w(n)*1i-
0.04*(w(n))^2)*(1+0.1*w(n)*1i)))*(180/pi);
mag(n)=m;
phase(n)=ph;
end
figure(1)
subplot(2,1,1)
semilogx(w,mag)
ylabel('Magnitude (dB)')
grid on
title('Bode Diagram')
subplot(2,1,2)
semilogx(w,phase)
xlabel('Frequency (rad/s)')
ylabel('Phase (deg)')
grid on
```

خروجی برنامه :



پرش موجود در فاز به دلیل این است که خروجی دستور angle بین $-\pi$ و π است.

برای اینکه پرش در فاز را از بین ببریم میتوان برنامه را به صورت زیر اصلاح کرد:

```
clc
clear
close all

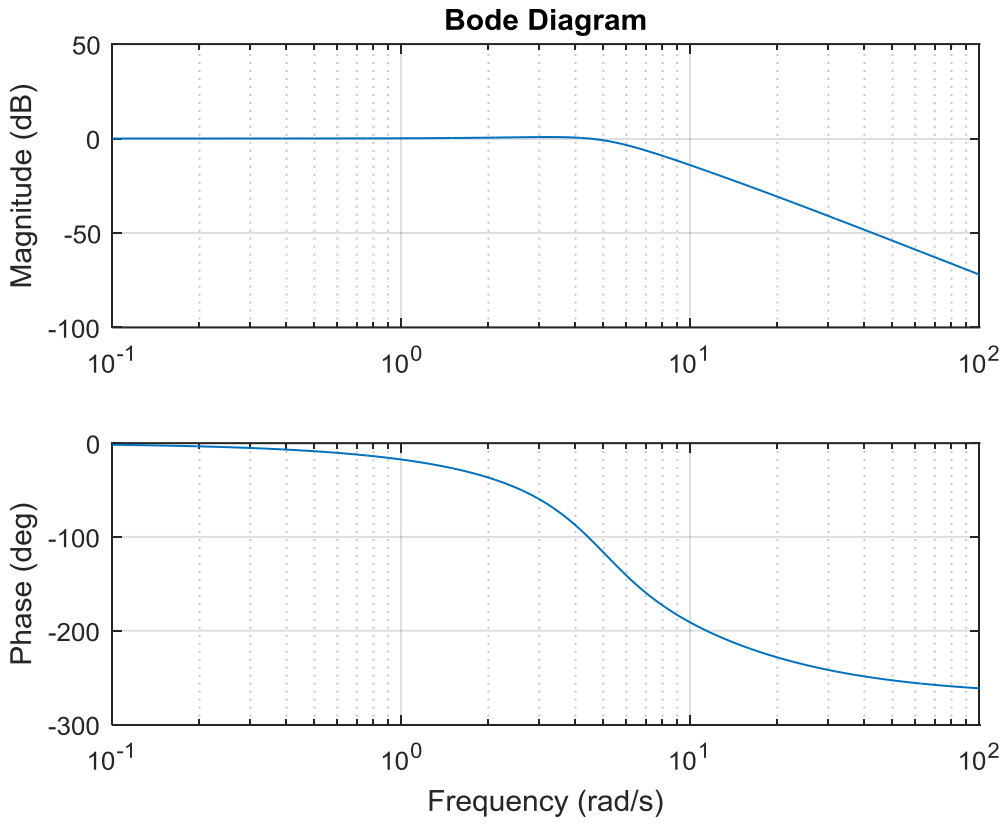
w=logspace(-1,2,1000);
for n=1:length(w);
    m=20*log10(abs(1/((1+0.2*w(n)*1i-
0.04*(w(n))^2)*(1+0.1*w(n)*1i))));
    ph=angle(1/((1+0.2*w(n)*1i-
0.04*(w(n))^2)*(1+0.1*w(n)*1i)))*(180/pi);
    mag(n)=m;
    phase(n)=ph;
end

for i=1:length(w)-1
    if phase(i+1)-phase(i)>=180
        phase(i+1)=phase(i+1)-360;
```

```
    elseif phase(i+1)-phase(i)<=-180
        phase(i+1)=phase(i+1)+360;
    end
end

figure(1)
subplot(2,1,1)
semilogx(w,mag)
ylabel('Magnitude (dB)')
grid on
title('Bode Diagram')
subplot(2,1,2)
semilogx(w,phase)
xlabel('Frequency (rad/s)')
ylabel('Phase (deg)')
grid on
```

خروجی :



رسم دیاگرام بوده به کمک دستور bode :

bode(num,den)

bode(num,den,w)
bode(num,den)
bode(A,B,C,D)
bode(A,B,C,D,iu)
bode(A,B,C,D,iu,w)
bode(sys)

اگر بردار **w** را وارد نکنیم خود متلب یک بردار برای آن در نظر میگیرد.

اگر بخواهیم ورودی را انتخاب کنیم باید از فرم فضای حالت استفاده کنیم. که در آن‌ها **iu** شماره ورودی است.

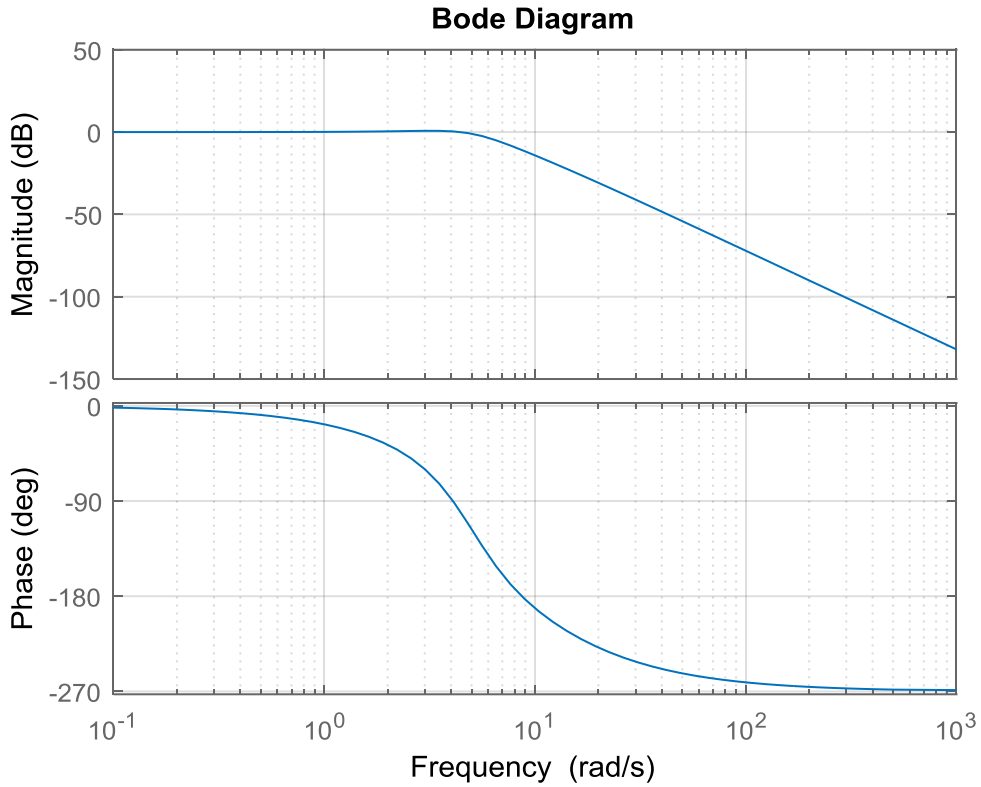
مثال :

همان مثال قبل را به کمک دستور **bode** رسم کنید:

```
clc
clear
close all

num=[1];
den=[0.004 0.06 0.3 1];
bode(num,den)
grid on
```

نتیجه:



که مطابق انتظار همان نتیجه قبلی را می‌دهد.

مثال :

نمودار بوده سیستم فوق را برای فرکانس‌های بین 0 تا 100 رادیان بر ثانیه رسم کنید.(250 فرکانس)

ابتدا باید یک بردار w ساخت که شامل فرکانس‌های مدنظر باشد برای این کار از دستور logspace استفاده می‌کنیم:

```
w = logspace(a,b)
w = logspace(a,b,n)
```

دستور اول یک بردار سطری 50 تایی ایجاد میکند که درایه‌های آن به صورت لگاریتمی بین 10^a و 10^b می‌باشد.

دستور دوم یک بردار سطری n تایی ایجاد میکند که درایه‌های آن به صورت لگاریتمی بین 10^a و 10^b می‌باشد.

بنابراین در این مثال :

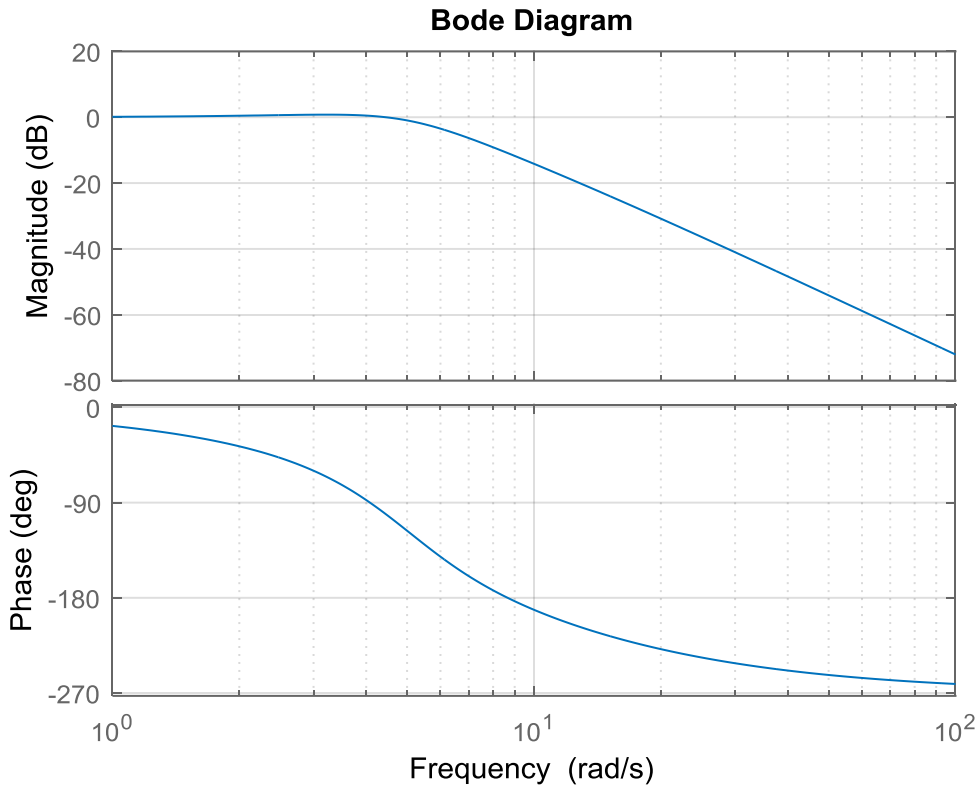
```
a=0;
```

b=2;

```
clc
clear
close all

w=logspace(0,2,250);
num=[1];
den=[0.004 0.06 0.3 1];
bode(num,den,w)
grid on
```

نتیجه :



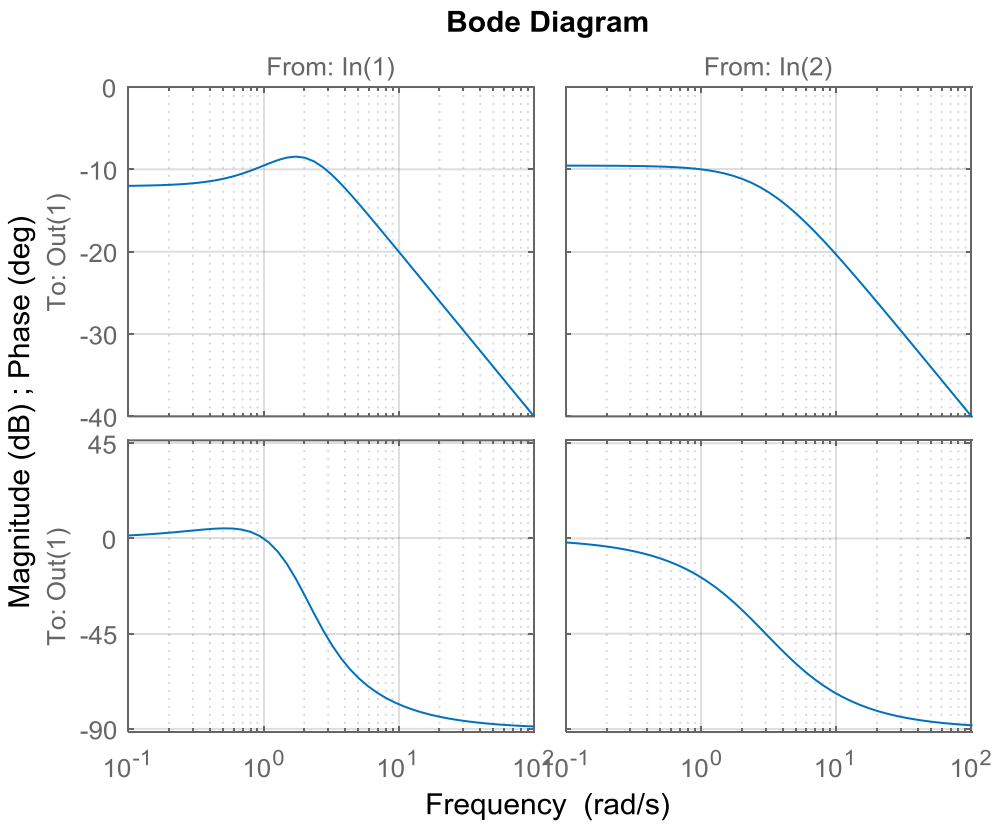
مثال دیاگرام بوده را برای سیستم زیر رسم کنید:

$$G = \left[\frac{s+1}{s^2+3s+4} \quad \frac{1}{s+3}\right]$$

پاسخ :

```
clc
clear
close all

s=tf('s');
G=[(s+1)/(s^2+3*s+4) 1/(s+3)];
bode(G)
grid on
```



توجه کنید که در مثال قبل 2 ورودی و یک خروجی داشتیم.

مثال :

دیاگرام بوده مثال قبل را فقط برای ورودی اول رسم کنید.

ابتدا باید فرم فضای حالت را بیابیم. (در جلسه سوم به طور کامل طریقه بدست آوردن فرم فضای حالت سیستم چند متغیره را بحث کردیم.)

```
clc
clear
close all

w=logspace(-1,2,250);

NUM11=[1 1];
NUM12=[1];
DEN1=[1 3 4];
DEN2=[1 3];

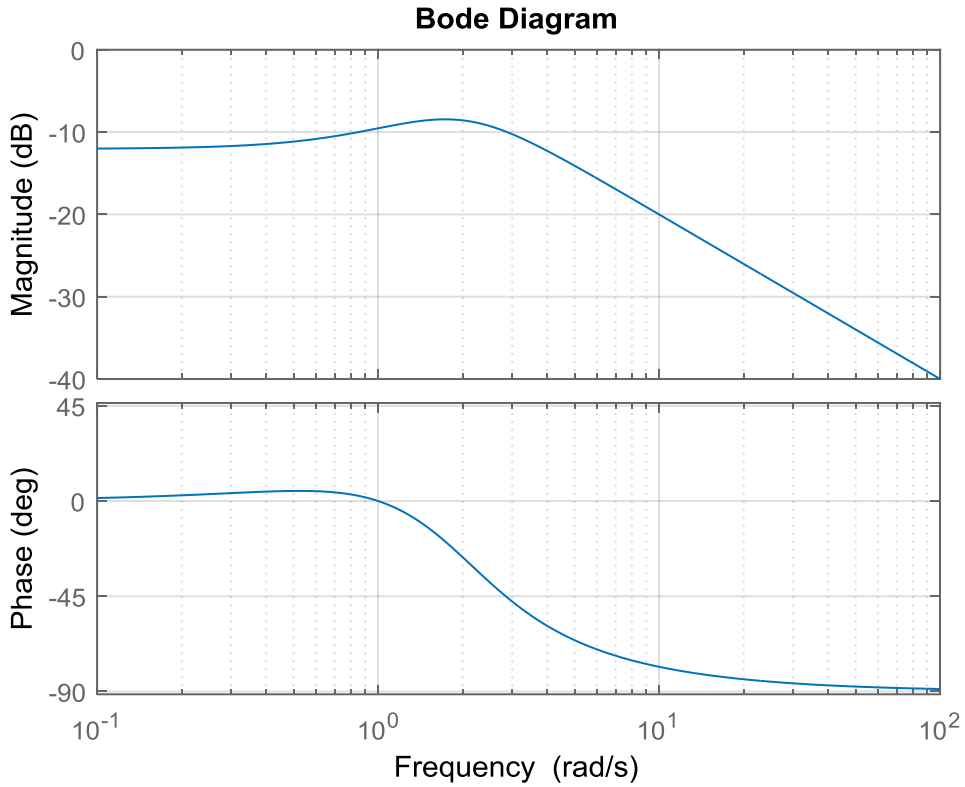
NUM1=NUM11;
NUM2=NUM12;

[A1,B1,C1,D1]=tf2ss(NUM1,DEN1);
[A2,B2,C2,D2]=tf2ss(NUM2,DEN2);

za1=zeros(size(A1,1),size(A2,2));
za2=zeros(size(A2,1),size(A1,2));
A=[A1 za1;za2 A2];
zb1=zeros(size(A1,1),1);
zb2=zeros(size(A2,1),1);
B=[B1 zb1;zb2 B2];
C=[C1 C2];
D=[D1 D2];

bode(A,B,C,D,1,w)
grid on
```

نتیجه :


Bode Diagram
Magnitude (dB)
0
-10
-20
-30
-40
Phase (deg)
45
0
-45
-90
10^-1
10^0
10^1
10^2
Frequency (rad/s)

به نام خدا

آموزش کاربرد متلب در کنترل خطی و مدرن

مدرس : بهنام ورمقانی

جلسه یازدهم :

دیاگرام بود (قسمت دوم)

برای بدست آوردن مقادیر دامنه و فاز از دستور زیر استفاده می‌کنیم :

```
[mag,phase,wout] = bode(sys)
```

در این دستور mag مقدار دامنه و phase مقدار زاویه برحسب درجه است و wout نیز بردار فرکانس های متناظر است.

اگر بخواهیم مقدار دامنه را بر حسب dB بدست آوریم از رابطه زیر استفاده میکنیم:

```
Mag_dB=20*log10(mag)
```

مثال:

برای سیستم

$$P(s) = \frac{s+1}{s^2+2s+3}$$

دیاگرام بود را رسم کنید:

```
clc
clear
close all

num=[1 1];
den=[1 2 3];

[mag,phase,w]=bode(num,den);

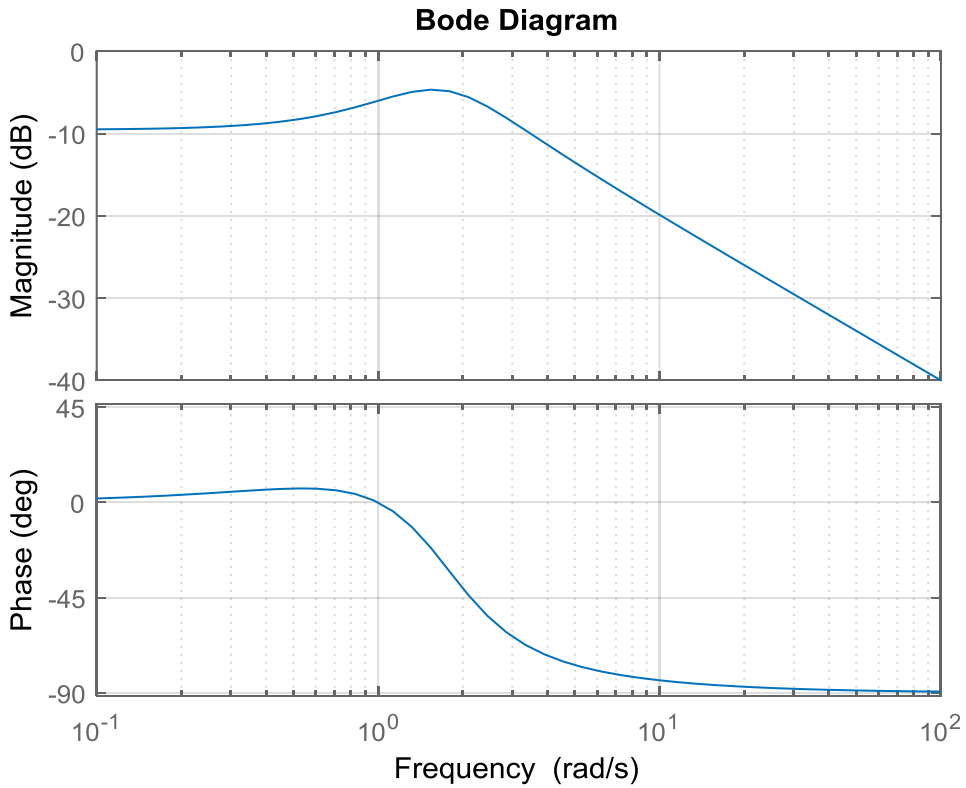
figure(1)
bode(num,den)
```

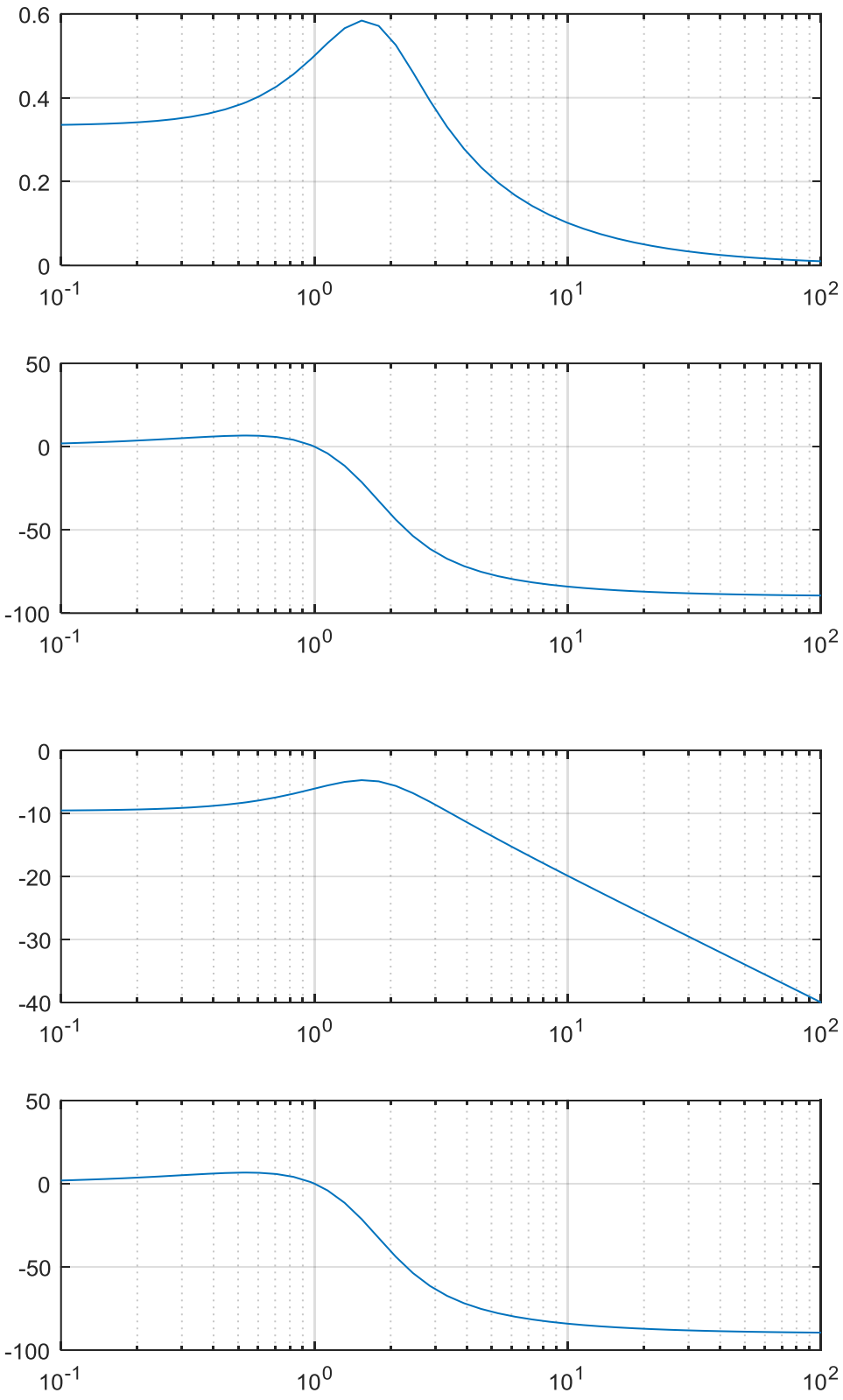
```
grid on

figure(2)
subplot(2,1,1)
semilogx(w,mag)
grid on
subplot(2,1,2)
semilogx(w,phase)
grid on

figure(3)
mag_dB=20*log10(mag);
subplot(2,1,1)
semilogx(w,mag_dB)
grid on
subplot(2,1,2)
semilogx(w,phase)
grid on
```

نتایج:




0.6
0.4
0.2
0
10^{-1}
10^{0}
10^{1}
10^{2}
50
0
-50
-100
10^{-1}
10^{0}
10^{1}
10^{2}
0
-10
-20
-30
-40
10^{-1}
10^{0}
10^{1}
10^{2}
50
0
-50
-100
10^{-1}
10^{0}
10^{1}
10^{2}

در شکل اول دیاگرام بود را توسط دستور bode رسم کردیم در شکل 2 پس از بدست آوردن مقادیر دامنه و فاز و فرکانس متناظرشان با استفاده از دستور semilogx مقادیر دامنه و فاز را رسم کردیم در این شکل مقدار دامنه بر حسب dB نیست. در شکل 3 نیز مقادیر دامنه و فاز را به کمک دستور semilogx رسم کردیم اما مقدار دامنه را برحسب dB تبدیل کردیم.

دستور semilogx :

این دستور شبیه به دستور plot عمل می‌کند با این تفاوت که محور x ها را در اسکیل لگاریتمی نمایش میدهد.

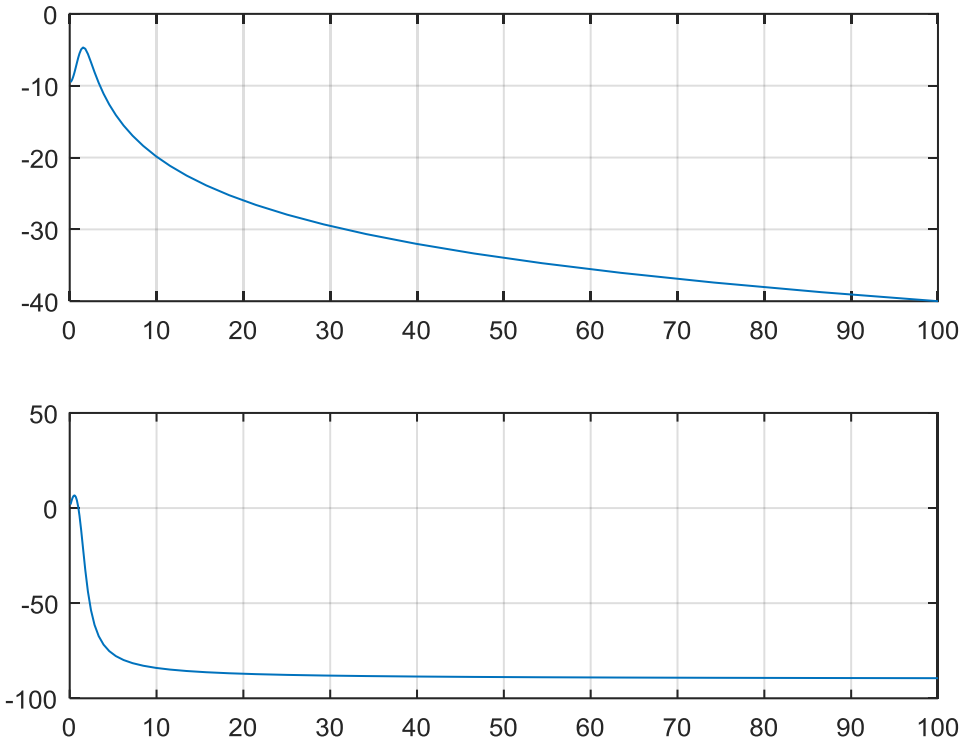
مثلا در مثال قبل اگر به جای semilogx از دستور plot استفاده کنیم داریم:

```
clc
clear
close all

num=[1 1];
den=[1 2 3];

[mag,phase,w]=bode(num,den);

subplot(2,1,1)
plot(w,20*log10(mag))
grid on
subplot(2,1,2)
plot(w,phase)
grid on
```



استخراج برخی از ویژگی های سیستم به کمک دیاگرام بود:

رابطه بین منحنی فاز و منحنی دامنه در سیستم های مینیمم فاز:

در سیستم های مینیمم فاز با مشخص بودن نمودار اندازه در همه فرکانس ها میتوان نمودار فرکانس را هم به دست آورد و برعکس. در سیستم های مینیمم فاز مقدار زاویه فاز در فرکانس بینهایت برابر است با -90(q-p) که q و p به ترتیب درجه مخرج و صورت تابع تبدیل حلقه باز سیستم است.

در همه سیستم ها چه مینیمم فاز و چه نامینیمم فاز شیب نمودار لگاریتم اندازه برابر -20(q-p) است.

مثال :

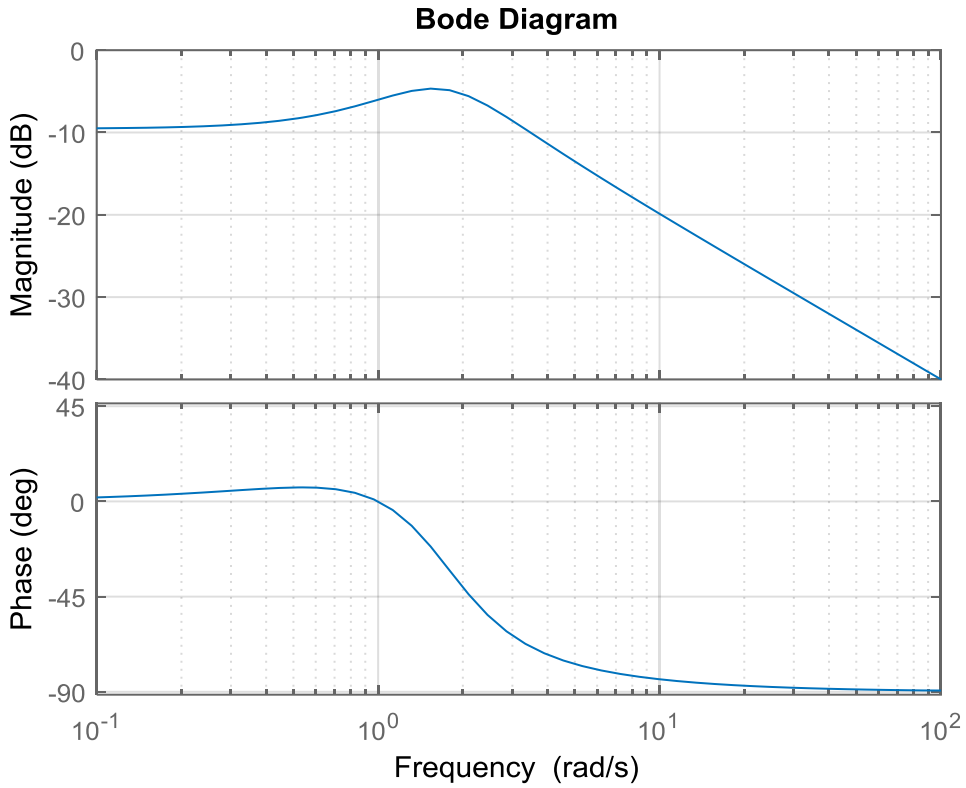
$$P(s) = \frac{s+1}{s^2+2s+3}$$

```
clc
```

```
clear
close all

num=[1 1];
den=[1 2 3];

bode(num,den);
grid on
```



در مثال فوق q=2 و p=1 بنابراین مقدار فاز نهایی باید برابر با -90(q-p)=-90 درجه باشد که از دیاگرام بود مشخص است.

همچنین مقدار شیب در فرکانس‌های نهایی برابر -20(q-p)=-2 dB/dec که از نمودار بود مشخص است.

در واقع اگر نمودار بود یک سیستم را داشته باشیم به کمک روابط اخیر میتوانیم درجه صورت و مخرج تابع تبدیل متناظرش را بیابیم. کافی است یک دستگاه دو معادله دو مجهول را حل کنیم.

ارتباط بین نمودار بود و نوع (تیپ) سیستم :

برای تعیین نوع(تیپ) سیستم به شیب نمودار اندازه در حوالی صفر توجه میکنیم برای سیستم با تیپ n این شیب برابر 20n- میباشد.

مثال :

$$P(s) = \frac{s+1}{s^2(s^2+2s+3)}$$

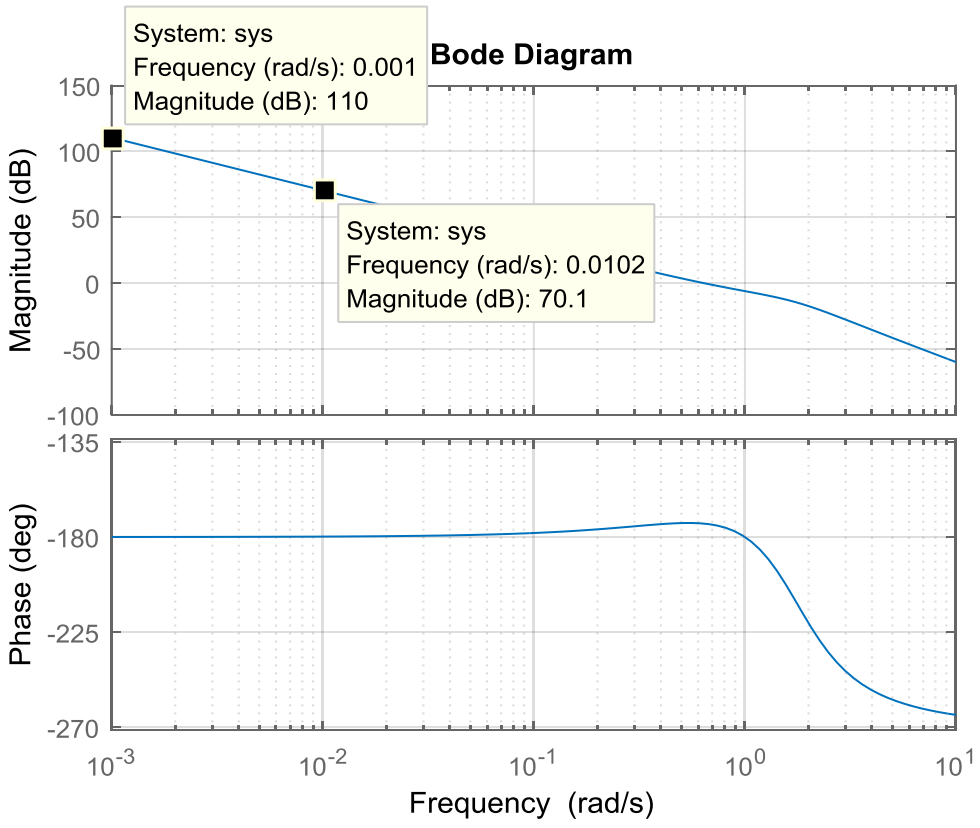
```
clc
clear
close all

s=tf('s');
sys=(s+1)/(s^2)/(s^2+2*s+3);

w=logspace(-3,1,100);

bode(sys,w);
grid on
```

نتیجه :



همانطور که از شکل فوق مشخص است شیب در حوالی فرکانس صفر برابر با 40 dB/dec است. که نشان میدهد سیستم از تیپ 2 میباشد.

در مثال فوق چون هدف تعیین تیپ سیستم بود بنابراین بردار w را به گونه‌ای انتخاب کردیم تا مقادیر دقیق‌تری در حوالی صفر داشته باشیم. یعنی فرکانس‌ها را از 0.001 تا 10 در نظر گرفتیم.

رابطه دیاگرام بود با ثابت خطای مکان، سرعت و شتاب :

با توجه به تیپ سیستم (شیب منحنی اندازه حوالی صفر) میتوان برای سیستم های نوع صفر k_p و برای سیستم های نوع یک k_v و برای سیستم های نوع دو k_a را بدست آوریم. برای سیستم نوع صفر دیاگرام اندازه با یک خط افقی شروع میشود و محل برخورد این خط با محور اندازه برابر $20\ \log k_p$ میباشد. برای سیستم با تیپ یک خط اولیه با شیب 20- را امتداد میدهیم عرض نقطه برخورد این خط با خط $\omega = 1$ برابر $20\ \log k_v$ میباشد. برای سیستم با تیپ دو خط اولیه با شیب 40- را امتداد میدهیم عرض نقطه برخورد این خط با خط $\omega = 1$ برابر $20\ \log k_a$ میباشد.

مثال:

$$G(s) = \frac{1}{(1 + 0.2s + 0.04s^2)(1 + 0.1s)}$$

همانطور که مشخص است سیستم از تیپ 0 می‌باشد. به صورت تحلیلی داریم:

$$k_p = \lim_{s \to 0} \frac{1}{(1 + 0.2s + 0.04s^2)(1 + 0.1s)} = 1$$

حال دیاگرام بود را رسم میکنیم:

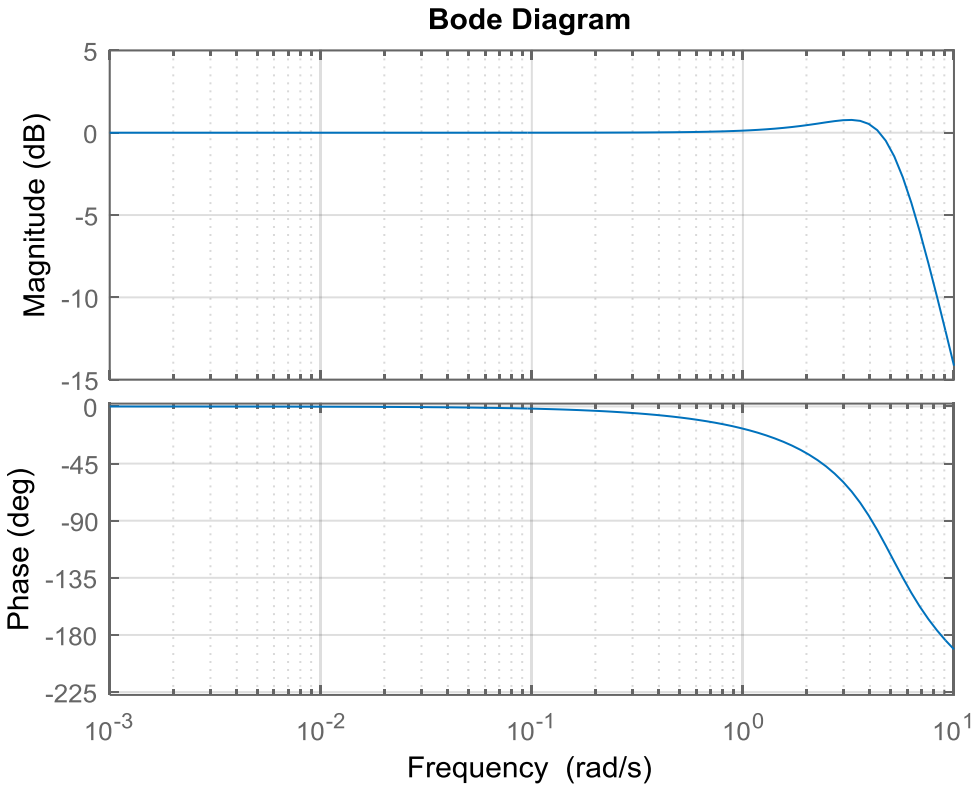
```
clc
clear
close all

s=tf('s');
sys=1/(1+0.2*s+0.04*s^2)/(1+0.1*s);

w=logspace(-3,1,100);

bode(sys,w);
```

```
grid on
```



بنابراین:

$20 \log k_p = 0$ و بنابراین $k_p = 1$ میباشد.

که همان نتیجه عملیات تحلیلی می‌باشد.

مثال :

$$G(s) = \frac{1}{s(1 + 0.2s + 0.04s^2)(1 + 0.1s)}$$

همانطور که مشخص است سیستم از تیپ 1 می‌باشد. به صورت تحلیلی داریم:

$$k_v = \lim_{s \to 0} s \frac{1}{s(1 + 0.2s + 0.04s^2)(1 + 0.1s)} = 1$$

حال به کمک دیاگرام بود :

یک خط اولیه با شیب 20- را امتداد میدهیم عرض نقطه برخورد این خط با خط $\omega = 1$ برابر $20 \log k_v$ میباشد. این کار را توسط برنامه زیر انجام میدهیم:

```
clc
clear
close all
w=logspace(-1,2,1000);

s=tf('s');
sys=1/s/(1+0.2*s+0.04*s^2)/(1+0.1*s);

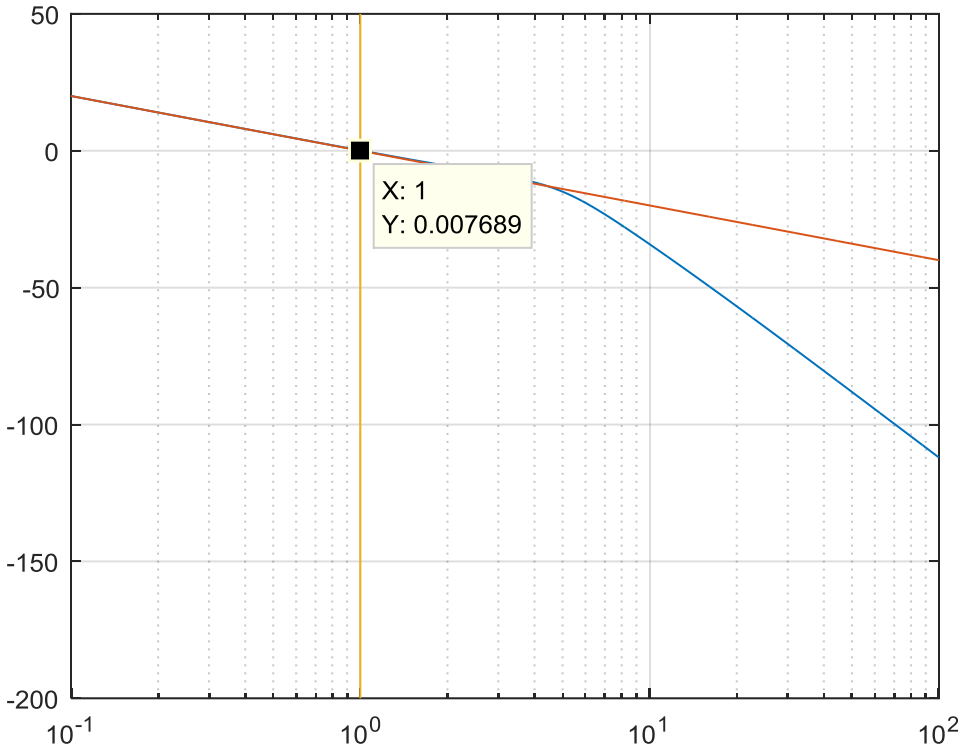
[MAG,phase,W]=bode(sys,w);

MAG_dB=20*log10(MAG);

for i=1:numel(MAG)
    mag(1,i)=MAG_dB(1,1,i);
end

semilogx(W,mag)
grid on
hold on
slope=(mag(10)-mag(1))/(log10(w(10))-log10(w(1)))
l=slope.*(log10(w)-log10(w(1)))+mag(1);
plot(w,l)
plot([1,1],[-200,50])
```

در این برنامه پس از محاسبه مقدار اندازه و تبدیل آن برحسب dB مقادیر را در یک بردار به نام mag قرار می‌دهیم. سپس باید در فرکانس‌های پایین یک خط بر نمودار اندازه مماس کنیم و عرض نقطه برخورد این خط با خط $\omega = 1$ را محاسبه کنیم. بنابراین ابتدا شیب خط مماس را بدست می‌آوریم (این شیب باید نزدیک به 20- باشد) سپس خط مورد نظر را رسم می‌کنیم. و در نهایت با رسم خط w=1 محل برخورد این دو خط را بدست می‌آوریم.



$$20 \log k_v = 0.007689$$

$$k_v = 1.0008856$$

که همان نتیجه حاصل از روش تحلیلی است.

مثال :

$$G(s) = \frac{1}{s^2(1+0.2s)(2+0.1s)}$$

همانطور که مشخص است سیستم از تیپ 2 می‌باشد. به صورت تحلیلی داریم:

$$k_a = \lim_{s\to 0} s^2 \frac{1}{s^2(1+0.2s)(2+0.1s)} = 0.5$$

حال به کمک دیاگرام بود :

خط اولیه با شیب 40- را امتداد میدهیم عرض نقطه برخورد این خط با خط $\omega = 1$ برابر $20 \log k_a$ میباشد.

همان برنامه فوق را اجرا می‌کنیم فقط با تابع تبدیل جدید:

```
clc
clear
close all
w=logspace(-1,2,1000);

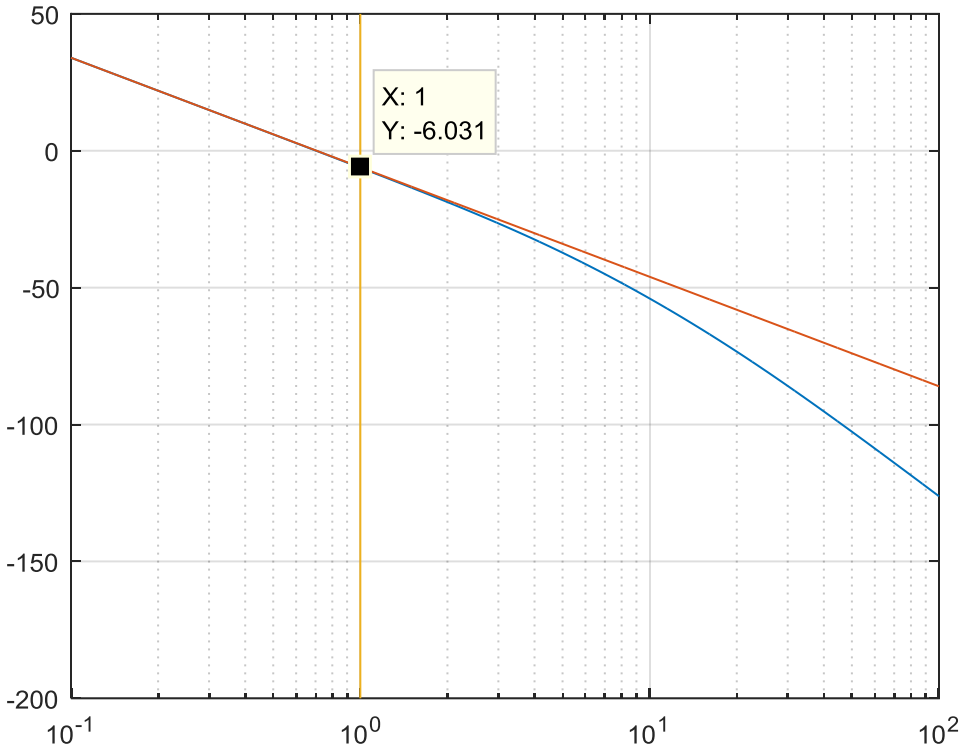
s=tf('s');
sys=1/s^2/(1+0.2*s)/(2+0.1*s);

[MAG,phase,W]=bode(sys,w);

MAG_dB=20*log10(MAG);

for i=1:numel(MAG)
    mag(1,i)=MAG_dB(1,1,i);
end

semilogx(W,mag)
grid on
hold on
slope=(mag(10)-mag(1))/(log10(w(10))-log10(w(1)))
l=slope.*(log10(w)-log10(w(1)))+mag(1);
plot(w,l)
plot([1,1],[-200,50])
```



$$20 \log k_a = -6.031$$

$$k_a = 0.4994$$

که همان نتیجه حاصل از روش تحلیلی می‌باشد.

روش دیگر برای بدست آوردن k_v و k_a

همچنین برای بدست آوردن k_v و k_a میتوان محل برخورد خط اولیه یا امتداد آن را با $0db$ به دست آورد و این فرکانس از نظر عددی برای سیستم مرتبه یک برابر k_v و برای سیستم مرتبه دو، برابر ریشه دوم k_a میباشد.

مثال :

$$G(s) = \frac{1}{s^2(1+0.2s)(2+0.1s)}$$

سیستم از نوع (تیپ) 2 می‌باشد و برای آن ثابت خطای شتاب برابر 0.5 است.(طبق محاسبات مثال قبل)

برنامه زیر که بسیار شبیه به برنامه قبلی است به کمک روش اخیر مقدار ثابت خطای شتاب (k_a) بدست می‌دهد.

```
clc
clear
close all
w=logspace(-1,2,10000);

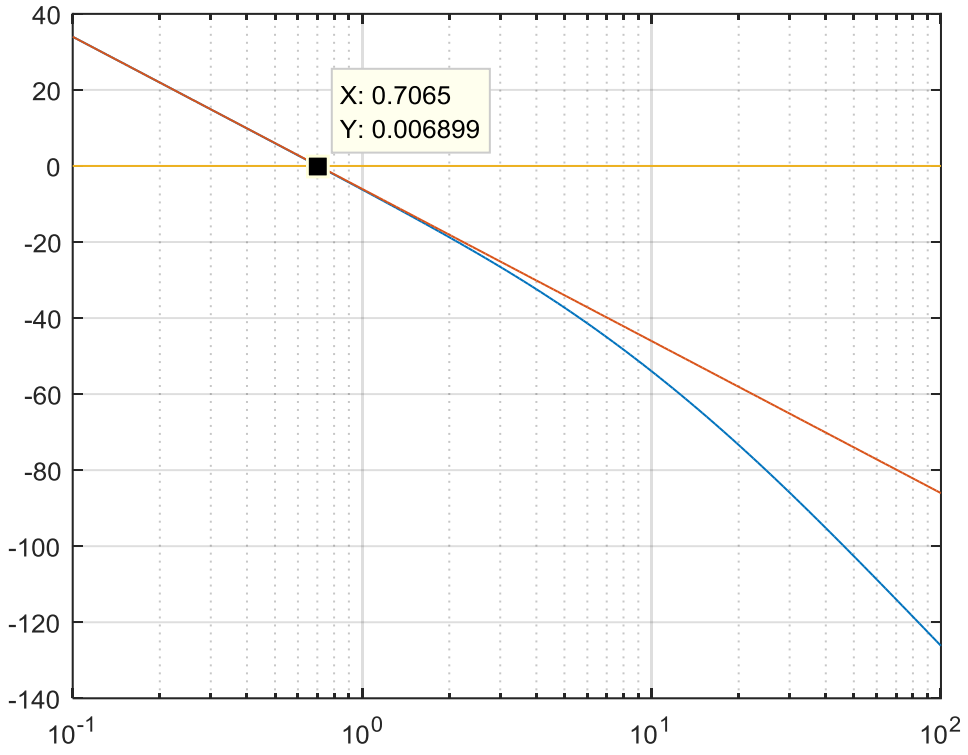
s=tf('s');
sys=1/s^2/(1+0.2*s)/(2+0.1*s);

[MAG,phase,W]=bode(sys,w);

MAG_dB=20*log10(MAG);

for i=1:numel(MAG)
    mag(1,i)=MAG_dB(1,1,i);
end

semilogx(W,mag)
grid on
hold on
slope=(mag(10)-mag(1))/(log10(w(10))-log10(w(1)))
l=slope.*(log10(w)-log10(w(1)))+mag(1);
plot(w,l)
plot([0.1,100],[0,0])
```



همانطور که ملاحظه شد تنها تفاوت این است که خط مماس را به جای قطع با x=0 با y=0 قطع می‌دهیم.

$$k_a = (0.7065)^2 = 0.49914$$

یک برنامه کلی برای بدست آوردن ثابت های خطا :

برنامه زیر که یک برنامه کلی است میتواند ثابت های خطا را با توجه نوع سیستم برای هر سیستم دلخواه به دست آورد همچنین k_v و k_a را با هر دو روش به دست می آورد.

```
clc
clear
close all
num=[1];
den=[0.02 0.5 2 0 0];
w=logspace(-3,2,1000);
[mag phase w]=bode(num,den,w);
slope=(20*log10(mag(50))-20*log10(mag(10)))/(log10(w(50))-
log10(w(10)));
K=slope*(log10(1)-log10(w(10)))+20*log10(mag(10));

k=10^(K/20);
if den(length(den))==0 && den(length(den)-1)==0;
```

```
    disp('type of system=2');disp('ba raveshe ghat ba w=1')
    ka=k
end
if den(length(den))==0 && den(length(den)-1)~=0;
    disp('type of system=1');disp('ba raveshe ghat ba w=1')
    kv=k
end
if den(length(den))~=0;
    disp('type of system=0')
    kp=k
end

kk=10^(((0-20*log10(mag(10)))/slope)+log10(w(10)));
if den(length(den))==0 && den(length(den)-1)==0;
    disp('ba raveshe ghat ba 0db')
    ka=kk^2
end
if den(length(den))==0 && den(length(den)-1)~=0;
    disp('ba raveshe ghat ba 0db')
    kv=kk
end
```

نتیجه :

```
type of system=2
ba raveshe ghat ba w=1

ka =

0.5000

ba raveshe ghat ba 0db

ka =

0.5000
```