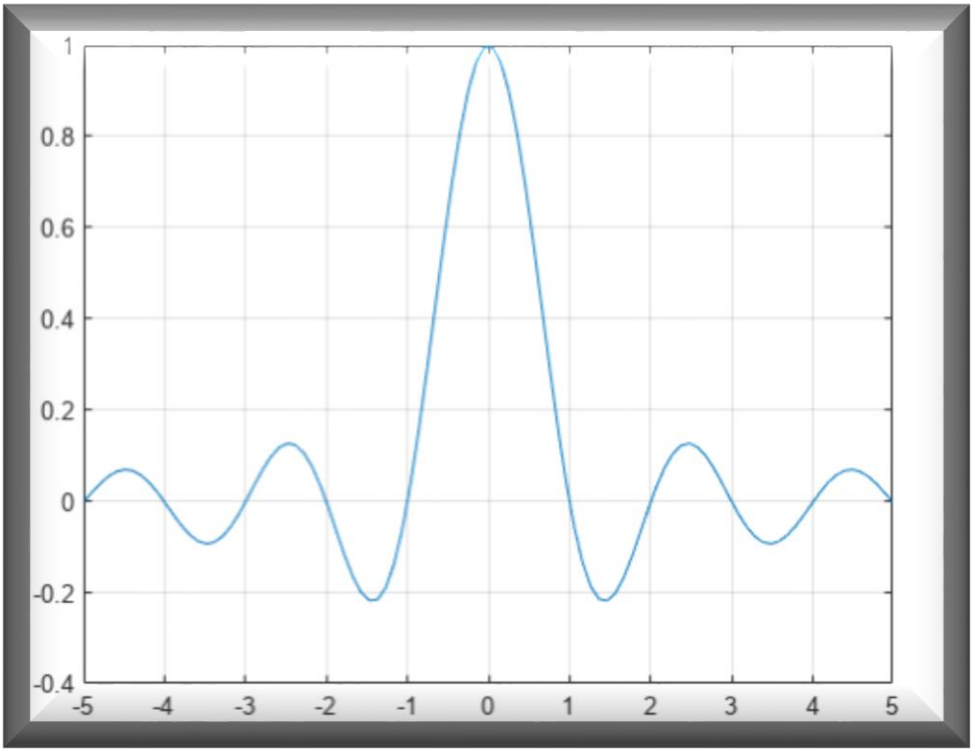


سیگنال های پر کاربرد در نرم افزار MATLAB

ارائه دهنده و استاد درس: دکتر هادی عزمی

با همکاری: علی اکبر سامانی، پارسا وطن پور، سارینا زیرک سیما، عرفان یعقوبی و عرفان احمدی

توابع در MATLAB:

در نرم افزار متلب، توابع یا همان سیگنال ها به دو صورت عددی(numeric) و سمبولیک(symbolic) تعریف می‌شوند.

تفاوت توابع عددی و سمبولیک:

۱. سرعت پردازش و اجرای توابع عددی بیشتر از توابع سمبولیک است.
۲. توابع عددی قابل ساده سازی نیستند ولی توابع سمبولیک با استفاده از دستور simplify()، قابل ساده سازی هستند.
۳. در هنگام استفاده از تبدیل ها(تبدیل لاپلاس و تبدیل z)، باید حتما از توابع سمبولیک استفاده کرد.
۴. توابع سمبولیک را می‌توان بدون استفاده از طول گام و توسط دستور fpolt()، رسم کرد.
۵. برای رسم توابع عددی حتما باید طول گام تعریف و از دستور plot() استفاده کرد.

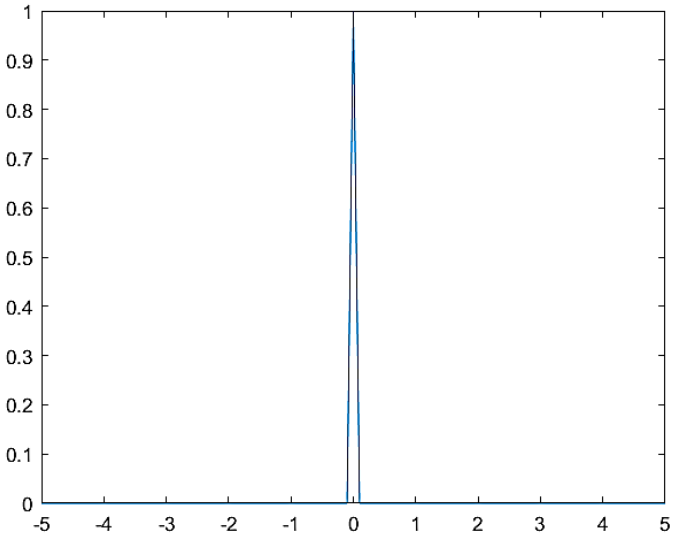
سیگنال ضربه:

```
d = dirac(t)
```

همانطور که می‌دانید تابع ضربه به ازاء ورودی صفر، مقدار بینهایت و به ازاء بقیه ورودی ها مقدار صفر دارد. از آن جایی که رسم مقدار بینهایت غیر ممکن است، باید با استفاده از دستور plot() یا stem() مقدار محدوده شده‌ای به آن نسبت دهیم. همانگونه که قبلا اشاره شد، دستور plot() برای رسم سیگنال های پیوسته و دستور stem() برای رسم سیگنال های گسسته استفاده می‌شود. برای ایجاد تابع ضربه را می‌توان به دو روش دستی یا استفاده از دستور dirac() عمل کرد. برای مثال، اجازه دهید یک تابع ضربه ایجاد کرده و سپس آن را با استفاده از دستور plot() رسم کنیم.

dirac_test.m

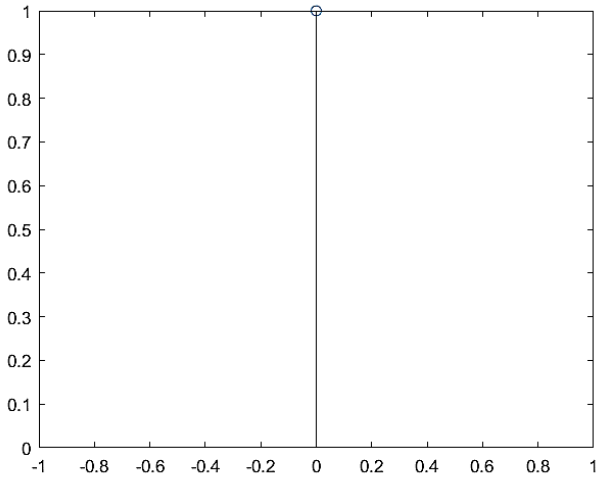
```
1  close all
2  clear
3  clc
4
5  t = -5:0.1:5;
6  d = double(t==0);
7  plot(t,d)
```



در تکه کد بالا، از دســـتور double() برای تعریف مقدار "یک" در نقطه t == 0 و مقدار "صـــفر" برای دیگر نقاط ا ستفاده می‌کنیم. همچنین می‌توان از د ستور stem() برای تابع ضربه‌ای که با د ستور dirac() ساخته شده استفاده کرد.

dirac_test.m

```
1  close all
2  clear
3  clc
4
5  t = 0:1;
6  d = dirac(t);
7  stem(d,t)
```



سیگنال پله:

```
H = heaviside(t - n)
```

با ا ستفاده از د ستور heaviside() می‌توان یک سیگنال پله با گام x ایجاد کرد. اگر به این تابع ورودی عددی نسبت دهیم، خروجی عددی و اگر ورودی سمبولیک نسبت دهیم، خروجی سمبولیک دریافت خواهیم کرد. تابع

heaviside یک تابع ناپیوسته است که برای $n < 0$ مقدار صفر، برای $n = 0$ مقدار $\frac{1}{2}$، و برای $n > 0$ مقدار 1 را بر می‌گرداند.

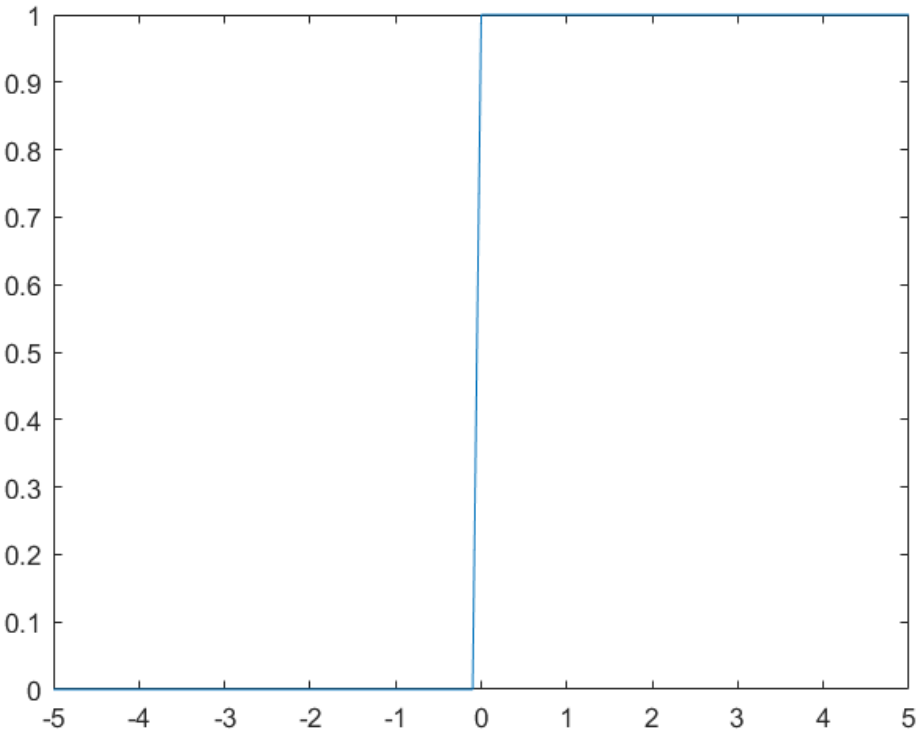
نکته: در صورتی که $n = 0$، نیازی به نوشتن 0 درون پرانتز نیست.

تابع Heaviside به صورت عددی

$n = 0$

heaviside_test.m

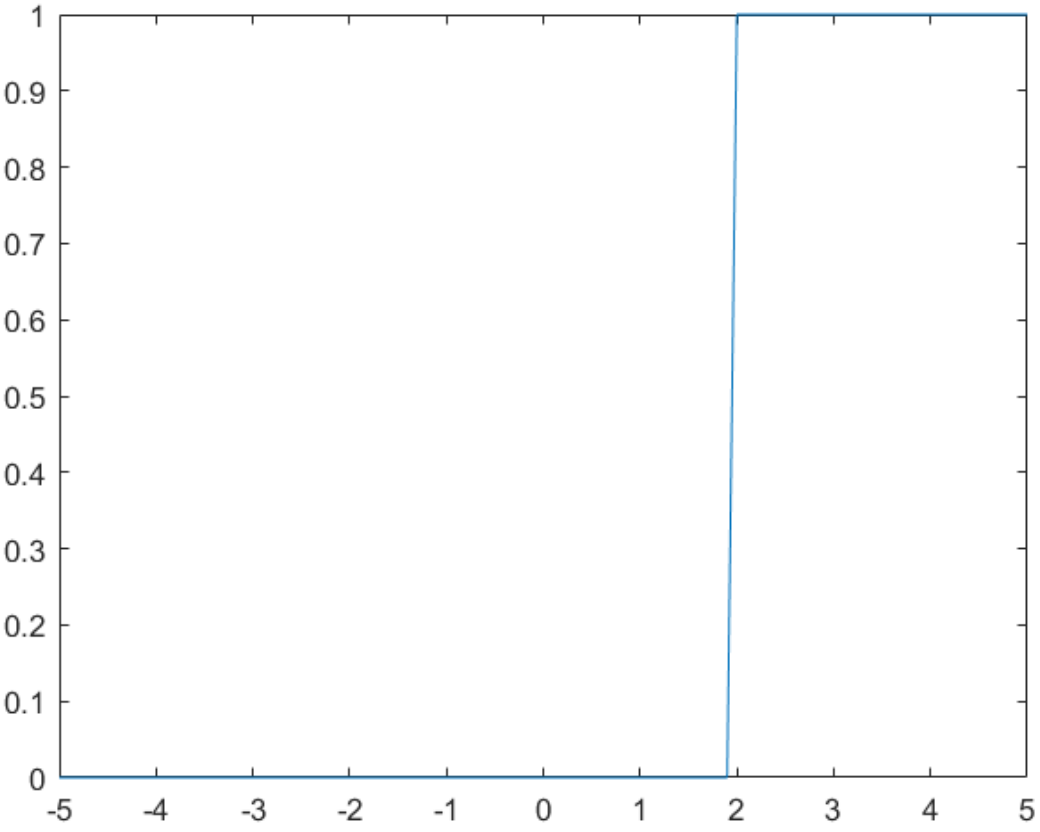
```
1  close all
2  clear
3  clc
4
5  t = -5:0.1:5;
6  u = heaviside(t);
7  plot(t,u)
```



n = 2

heaviside_test.m

```
1   close all
2   clear
3   clc
4
5   t = -5:0.1:5;
6   u = heaviside(t-2);
7   plot(t,u)
```

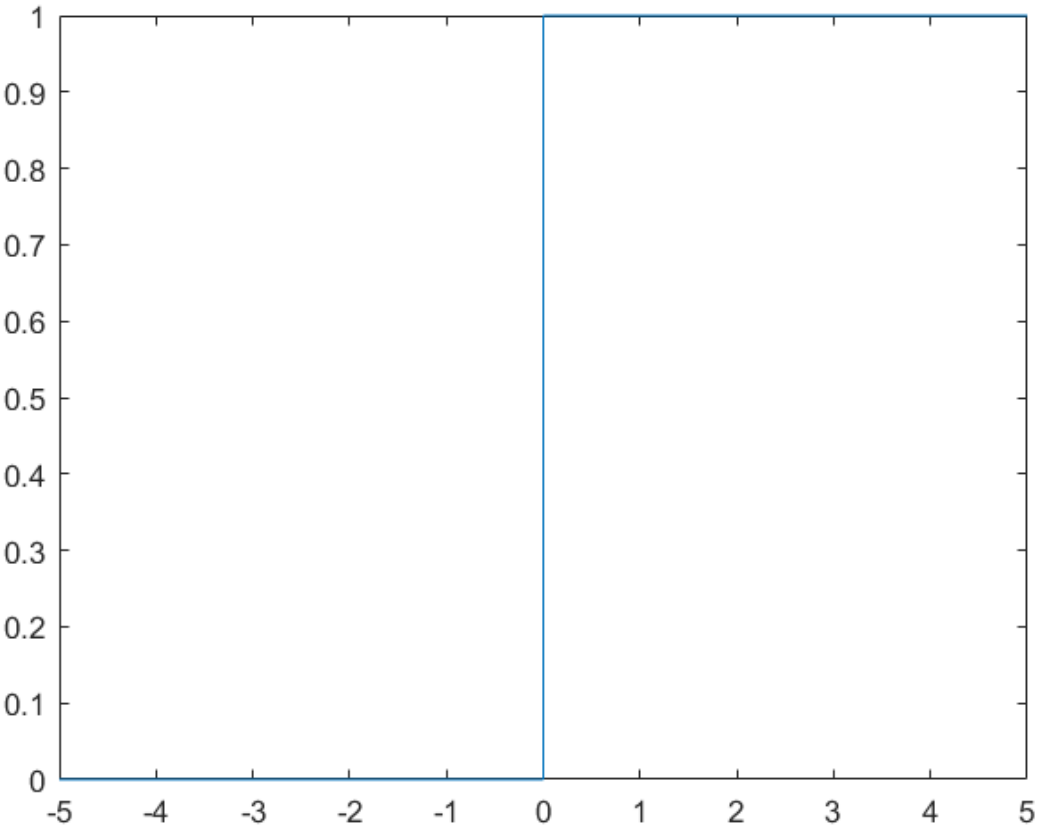


تابع Heaviside به صورت سمبولیک

n = 0

heaviside_test.m

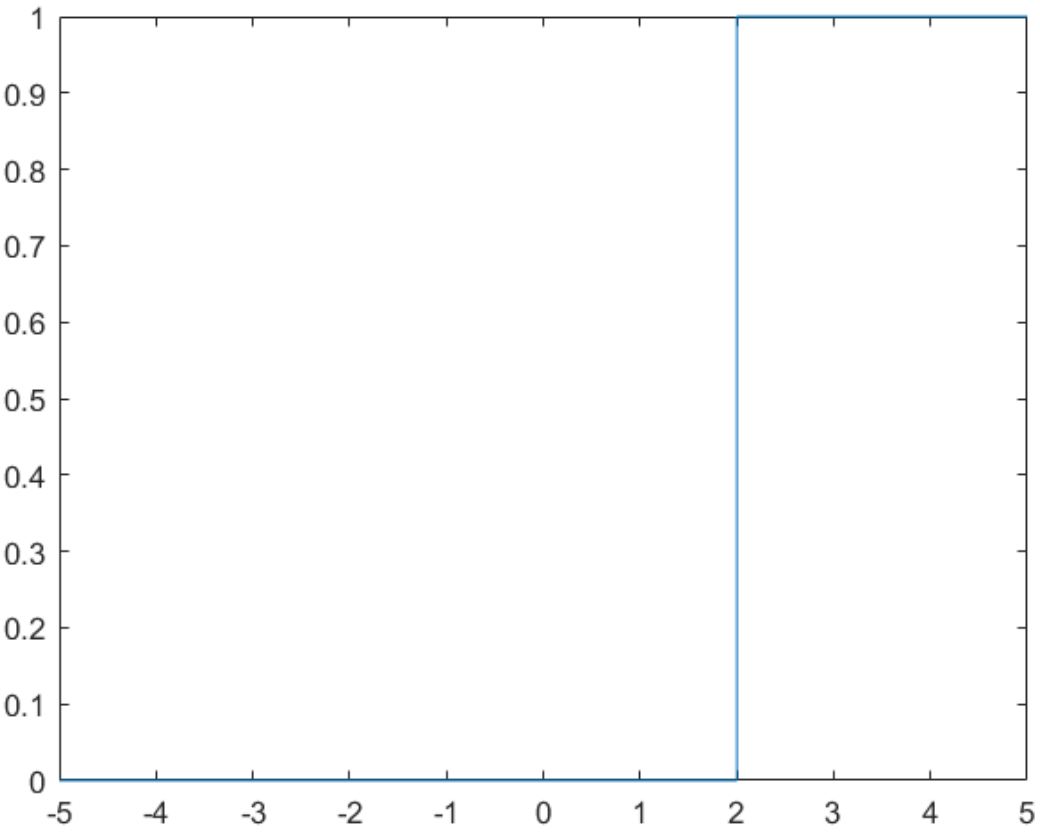
```
1  close all
2  clear
3  clc
4
5  syms t;
6  y = heaviside(t);
7  fplot(y)
```



n = 2

heaviside_test.m

```
1   close all
2   clear
3   clc
4
5   syms t;
6   y = heaviside(t-2);
7   fplot(y)
```



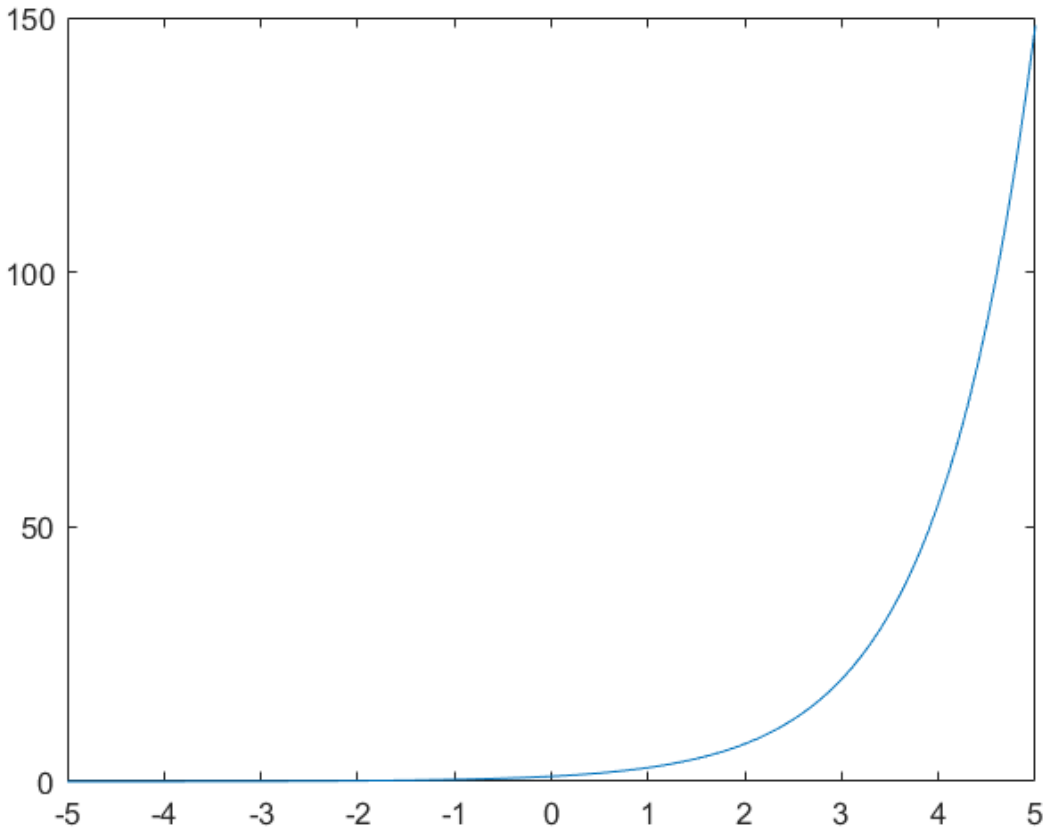
سیگنال نمایی (exp):

```
y = exp(t)
```

تابع exp به صورت عددی

exp_test.m

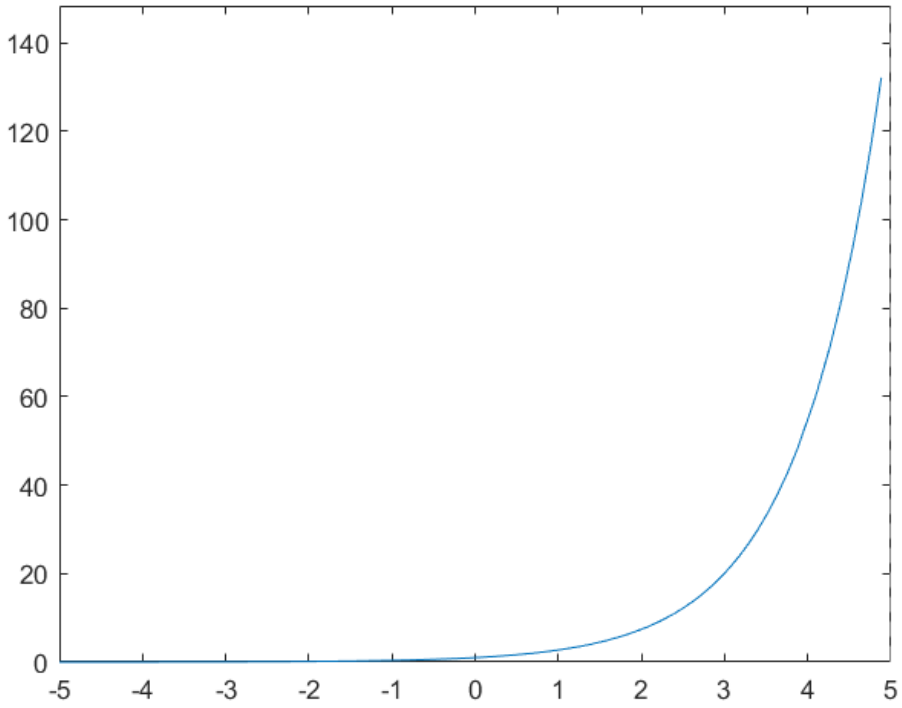
```
1  close all
2  clear
3  clc
4
5  t = -5:0.1:5;
6  y = exp(t);
7  plot(t,y)
```



تابع exp به صورت سمبولیک

exp_test.m

```
1  close all
2  clear
3  clc
4
5  syms t;
6  y = exp(t);
7  fplot(y)
```



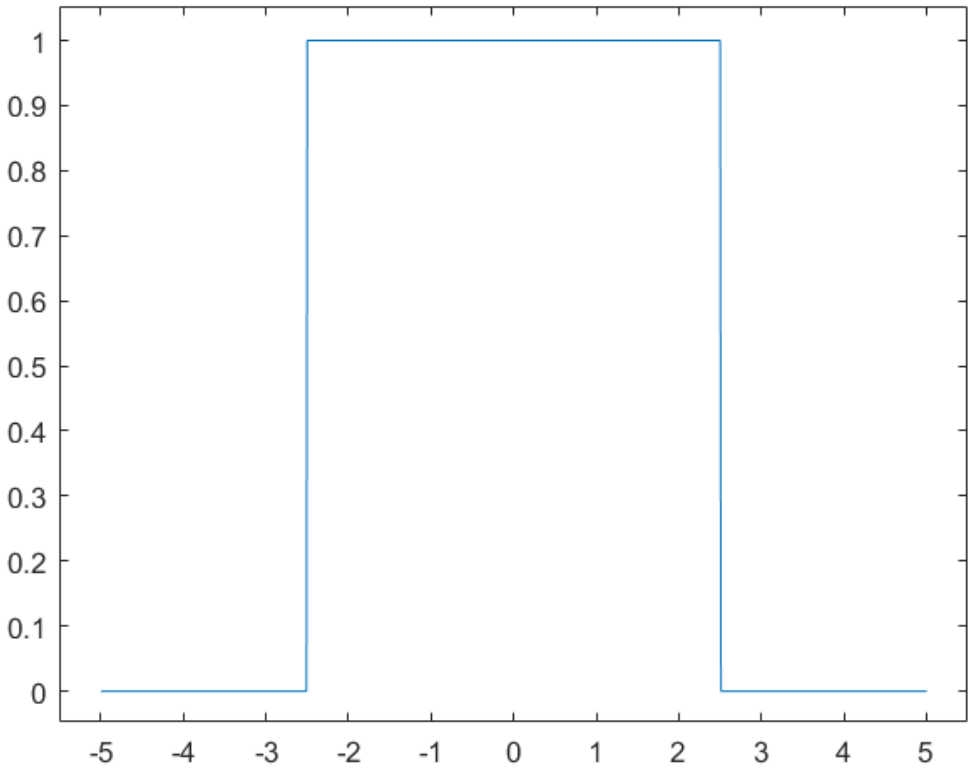
سیگنال مستطیلی(rect):

```
rectangularPulse(t/2*a)
```

همانند تعریف سیگنال rect که در جزوه آموختیم، در نرم افزار متلب از تابع rectangularPulse() استفاده می‌کنیم. بدین صورت که لبه‌ی بالارونده‌ی آن در نقطه‌ی a- و لبه‌ی پایین رونده‌ی آن در نقطه‌ی a قرار گرفته است.

rect_test.m

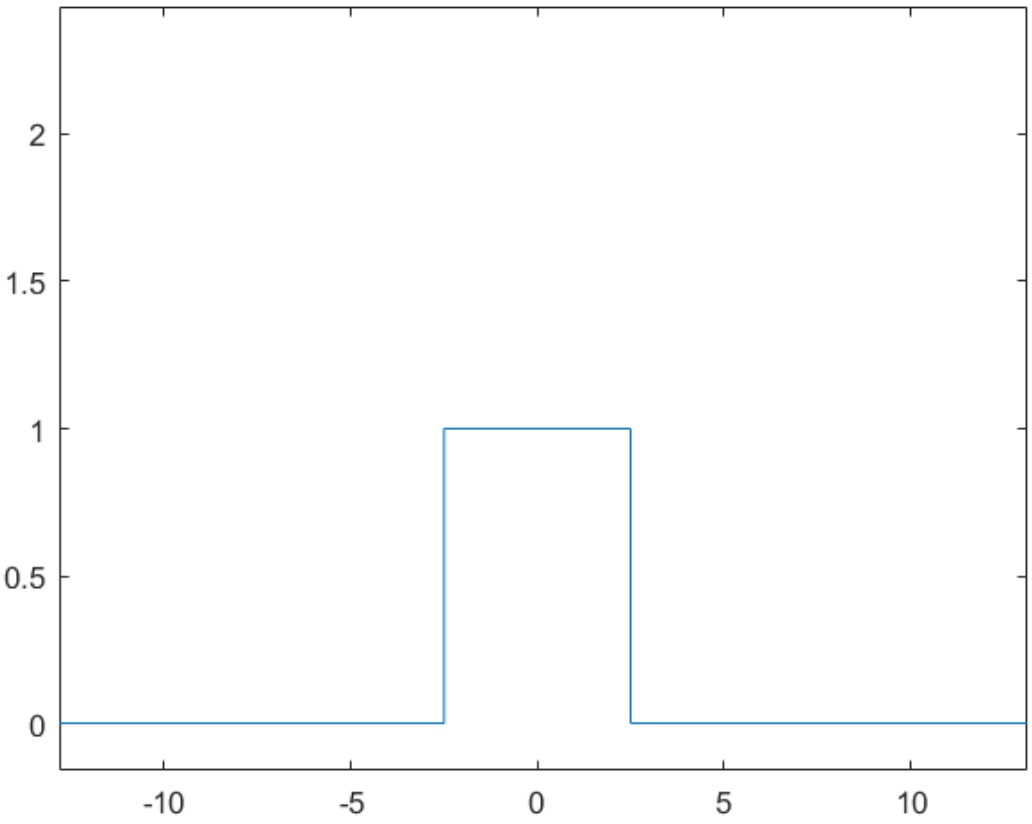
```
close all
clear
clc

t = -5:0.01:5;
f = rectangularPulse(t/5);
plot(t,f)
```



برای تعریف سیگنال rect به صورت سمبولیک، کافیست متغیر t را به صورت سمبولیک تعریف کرده و به ورودی تابع rectangularPulse() اعمال کنیم.

rect_test.m

```
1  close all
2  clear
3  clc
4
5  syms t;
6  f = rectangularPulse(t/5);
7  fplot(f)
```



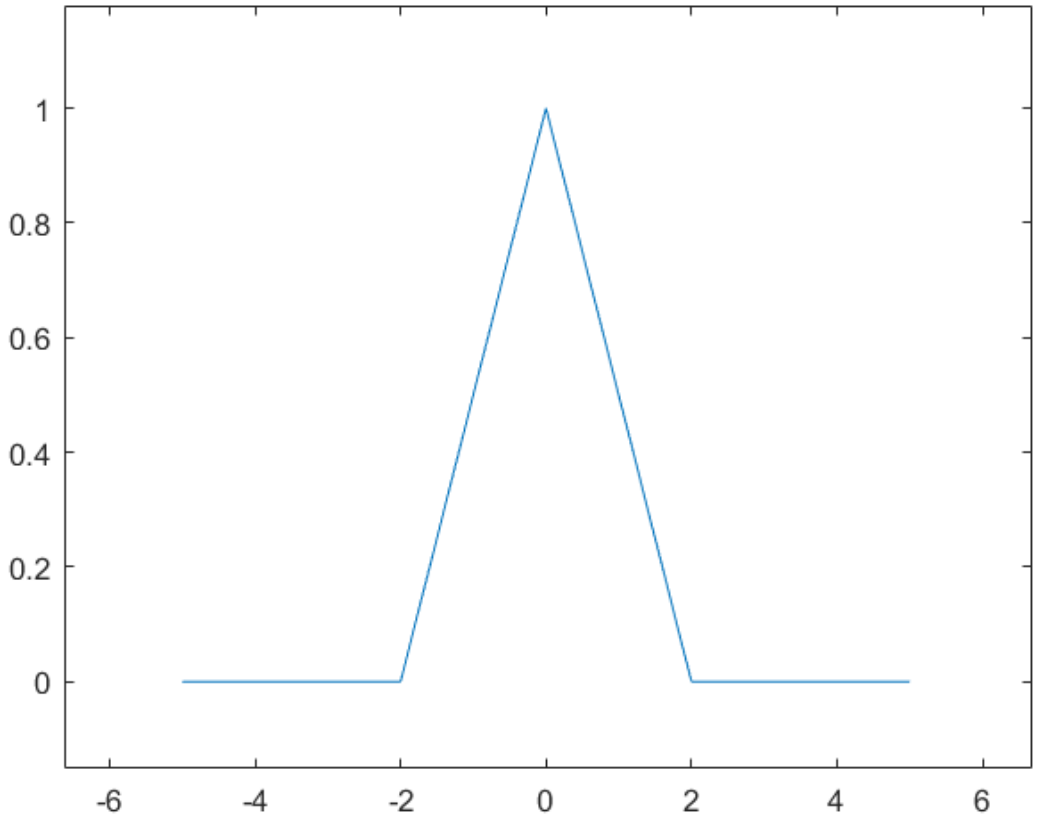
سیگنال مثلثی(tri):

```
triangularPulse(t/2*a)
```

برای تعریف سیگنال tri کافیست همانند تعریف سیگنال rect عمل کنیم؛ با این تفاوت که این بار از تابع triangularPulse() استفاده می‌کنیم.

tri_test.m

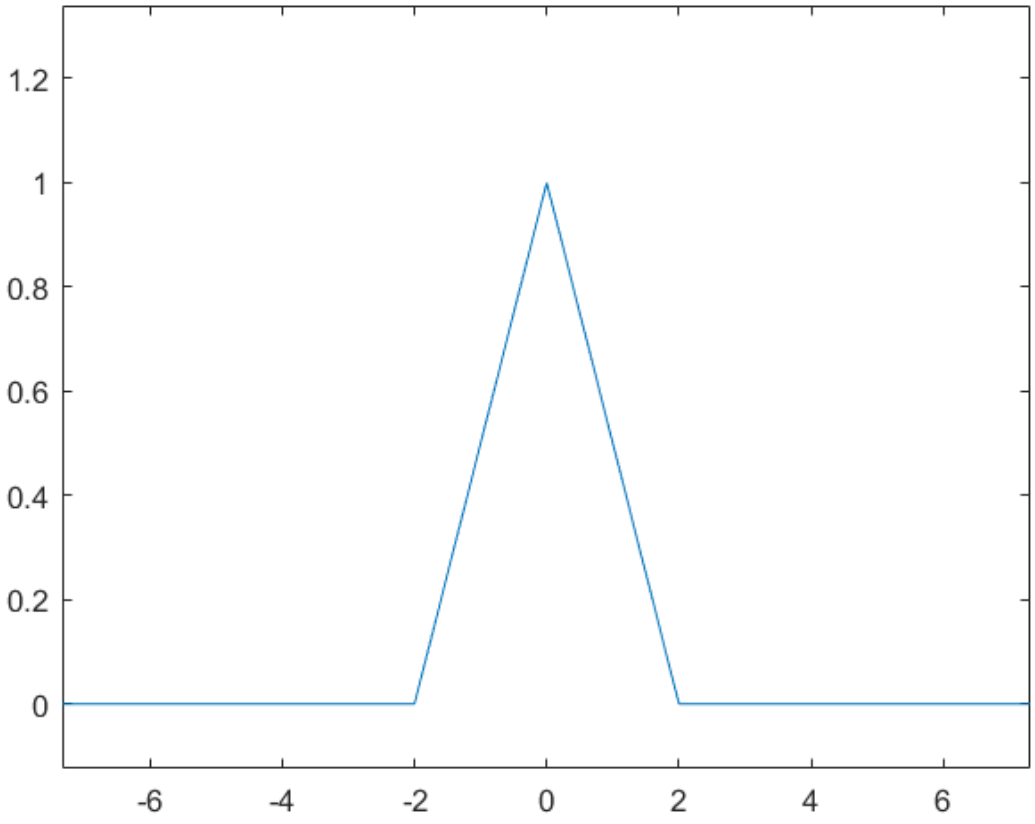
```
close all
clear
clc

t = -5:0.01:5;
f = triangularPulse(t/2);
plot(t,f)
```



برای تعریف سیگنال tri به صورت سمبولیک نیز کافیست متغیر t را به صورت سمبولیک تعریف به ورودی تابع triangularPulse() اعمال کنیم.

tri_test.m

```
close all
clear
clc

syms t;
f = triangularPulse(t/2);
fplot(f)
```



سیگنال sinc(t):

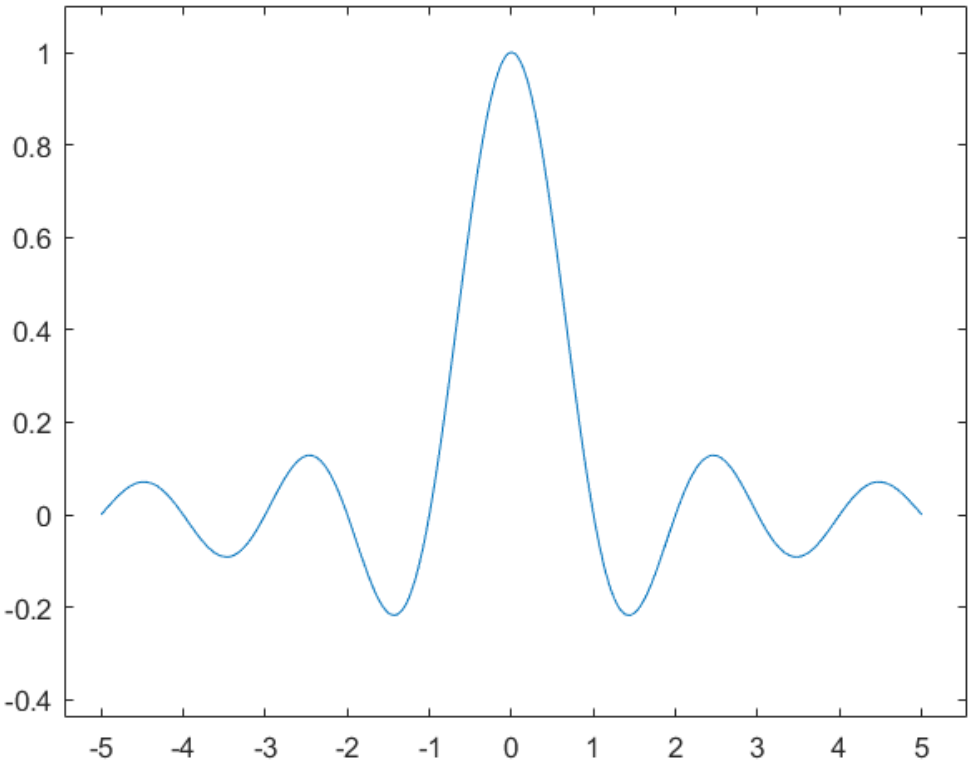
```
sinc(t)
```

همانطور که در جزوه درس خواندیم، مقدار تابع sinc در $t = 0$ مبهم است اما وقتی t به سمت صفر میل میکند، مقدار حدی برابر یک دارد.

به صورت عددی:

sinc_test.m

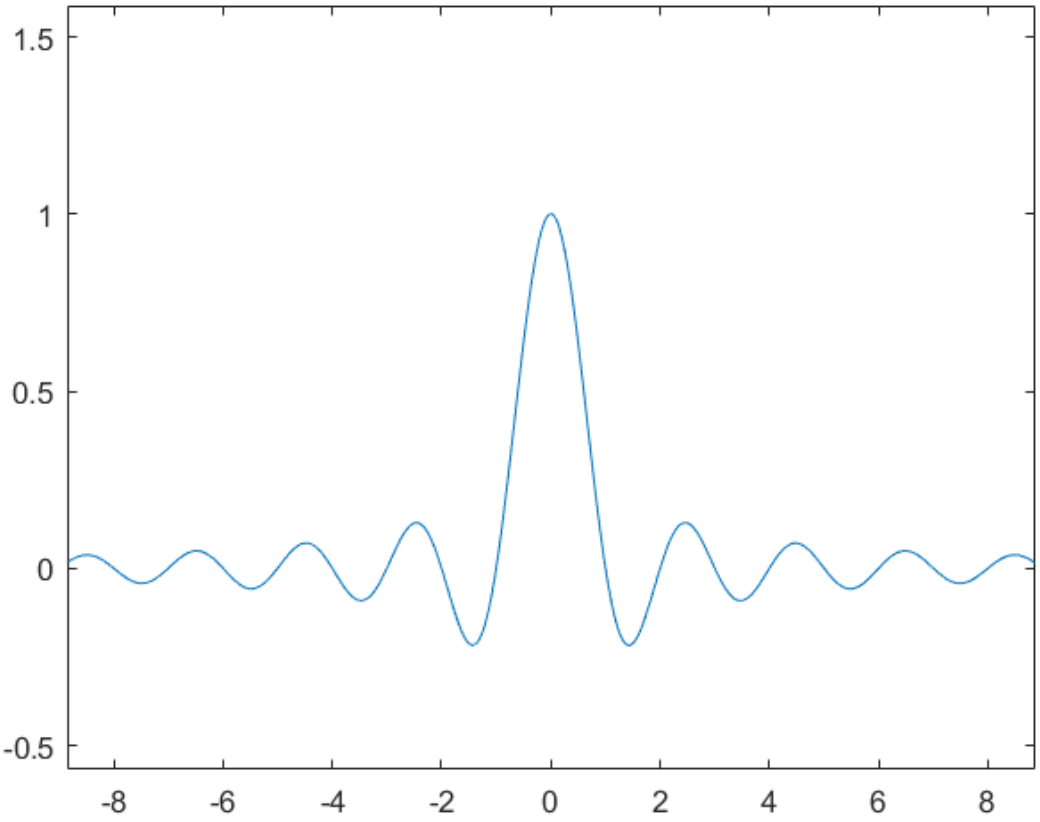
```
close all
clear
clc

t = -5:0.01:5;
f = sinc(t);
plot(t,f)
```



به صورت سمبولیک:

sinc_test.m

```
1  close all
2  clear
3  clc
4
5  syms t;
6  f = sinc(t);
7  fplot(f)
```



یک نکته کوتاه:

در نرم افزار متلب برای محاسبه انتگرال از دستور int() و برای محاسبه مشتق از دستور diff() استفاده می‌کنیم.

سیگنال شیب واحد(ramp):

در نرم افزار متلب نمی‌توان به صورت مستقیم تابع شیب واحد را تولید کرد اما همانطور که قبلا خواندیم:

$$r(t) = \int u(t)dt$$

با استفاده از معادله فوق و نکته گفته شده می‌توان تابع شیب واحد یا ramp را ایجاد کرد.

ramp_test.m

```
1  close all
2  clear
3  clc
4
5  syms t;
6  f = int(heaviside(t));
7  fplot(f)
```