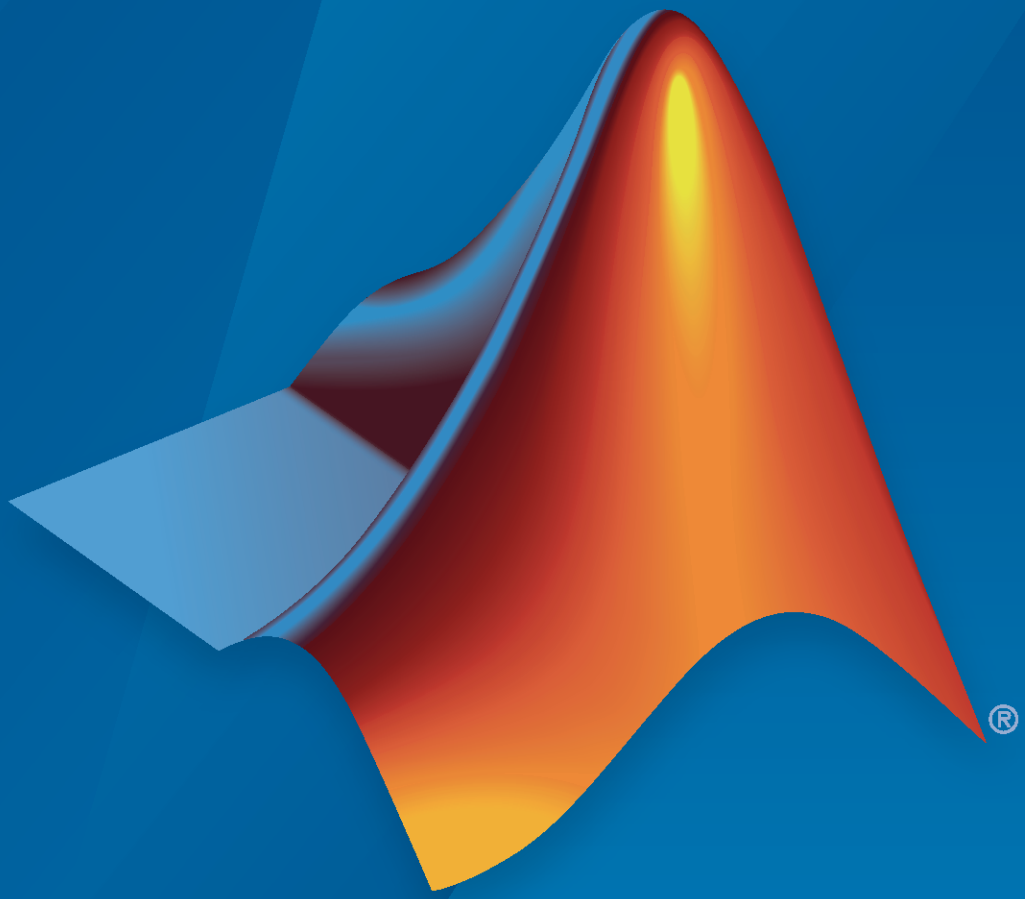
Signal Processing Toolbox™

Getting Started Guide



MATLAB®

R2026a

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Signal Processing Toolbox™ Getting Started Guide

Revision History

September 2006	First printing	New for Version 6.6 (Release 2006b)
March 2007	Online only	Revised for Version 6.7 (Release 2007a)
September 2007	Online only	Revised for Version 6.8 (Release 2007b)
March 2008	Online only	Revised for Version 6.9 (Release 2008a)
October 2008	Online only	Revised for Version 6.10 (Release 2008b)
March 2009	Online only	Revised for Version 6.11 (Release 2009a)
September 2009	Online only	Revised for Version 6.12 (Release 2009b)
March 2010	Online only	Revised for Version 6.13 (Release 2010a)
September 2010	Online only	Revised for Version 6.14 (Release 2010b)
April 2011	Online only	Revised for Version 6.15 (Release 2011a)
September 2011	Online only	Revised for Version 6.16 (Release 2011b)
March 2012	Online only	Revised for Version 6.17 (Release 2012a)
September 2012	Online only	Revised for Version 6.18 (Release 2012b)
March 2013	Online only	Revised for Version 6.19 (Release 2013a)
September 2013	Online only	Revised for Version 6.20 (Release 2013b)
March 2014	Online only	Revised for Version 6.21 (Release 2014a)
October 2014	Online only	Revised for Version 6.22 (Release 2014b)
March 2015	Online only	Revised for Version 7.0 (Release 2015a)
September 2015	Online only	Revised for Version 7.1 (Release 2015b)
March 2016	Online only	Revised for Version 7.2 (Release 2016a)
September 2016	Online only	Revised for Version 7.3 (Release 2016b)
March 2017	Online only	Revised for Version 7.4 (Release 2017a)
September 2017	Online only	Revised for Version 7.5 (Release 2017b)
March 2018	Online only	Revised for Version 8.0 (Release 2018a)
September 2018	Online only	Revised for Version 8.1 (Release 2018b)
March 2019	Online only	Revised for Version 8.2 (Release 2019a)
September 2019	Online only	Revised for Version 8.3 (Release 2019b)
March 2020	Online only	Revised for Version 8.4 (Release 2020a)
September 2020	Online only	Revised for Version 8.5 (Release 2020b)
March 2021	Online only	Revised for Version 8.6 (Release 2021a)
September 2021	Online only	Revised for Version 8.7 (Release 2021b)
March 2022	Online only	Revised for Version 9.0 (Release 2022a)
September 2022	Online only	Revised for Version 9.1 (Release 2022b)
March 2023	Online only	Revised for Version 9.2 (Release 2023a)
September 2023	Online only	Revised for Version 23.2 (R2023b)
March 2024	Online only	Revised for Version 24.1 (R2024a)
September 2024	Online only	Revised for Version 24.2 (R2024b)
March 2025	Online only	Revised for Version 25.1 (R2025a)
September 2025	Online only	Revised for Version 25.2 (R2025b)
March 2026	Online only	Revised for Version 26.1 (R2026a)

Contents

1 Overview

2 Basic Signal Processing Concepts

Overview

Signal Processing Toolbox Product Description

Perform signal processing and analysis

Signal Processing Toolbox provides functions and apps to manage, analyze, preprocess, and extract features from uniformly and nonuniformly sampled signals. The toolbox includes tools for filter design and analysis, resampling, smoothing, detrending, and power spectrum estimation. You can use the Signal Analyzer app to visualize and process signals simultaneously in time, frequency, and time-frequency domains. With the Filter Designer app, you can design and analyze FIR and IIR digital filters.

Using toolbox functions and the Signal Feature Extractor app, you can prepare signal datasets for AI model training by engineering features that reduce dimensionality and improve the quality of signals. With the Signal Labeler app, you can annotate signals in time and time-frequency domains to create labeled signal sets for training AI models. The toolbox supports GPU acceleration in addition to C/C++ and CUDA® code generation for desktop prototyping and embedded system deployment.

Basic Signal Processing Concepts

Representing Signals

In this section...

Numeric Arrays

The central data construct in the MATLAB environment is the *numeric array*, an ordered collection of real or complex numeric data with two or more dimensions. The basic data objects of signal processing (one-dimensional signals or sequences, multichannel signals, and two-dimensional signals) are all naturally suited to array representation.

Vector Representation

MATLAB represents ordinary one-dimensional sampled data signals, or sequences, as *vectors*. Vectors are 1-by-*n* or *n*-by-1 arrays, where *n* is the number of samples in the sequence. One way to introduce a sequence is to enter it as a list of elements at the command prompt. The statement

```
x = [4 3 7 -9 1];
```

creates a simple five-element real sequence in a row vector. Transposition turns the sequence into a column vector

```
x = x';
```

```
x =
     4
     3
     7
    -9
     1
```

Column orientation is preferable for single channel signals because it extends naturally to the multichannel case. For multichannel data, each column of a matrix represents one channel. Each row of such a matrix then corresponds to a sample point. A three-channel signal that consists of x, 2x, and x/π is

```
y = [x 2*x x/pi]
```

```
y =
    4.0000    8.0000    1.2732
    3.0000    6.0000    0.9549
    7.0000   14.0000    2.2282
   -9.0000  -18.0000   -2.8648
    1.0000    2.0000    0.3183
```

If the sequence has complex-valued elements, the transpose operator takes the conjugate of the sequence elements. To transform a complex-valued row vector into a column vector without taking conjugates, use the `.'` or non-conjugate transpose:

```
x = [1-i 3+i 2+3*i 4-2*i]; % 1-by-4 vector
x = x.';                   % 4-by-1 vector
```

Waveform Generation: Time Vectors and Sinusoids

Most toolbox functions require you to begin with a vector representing a time base. Consider generating data with a 1000 Hz sample frequency, for example. An appropriate time vector is

```
t = (0:0.001:1)';
```

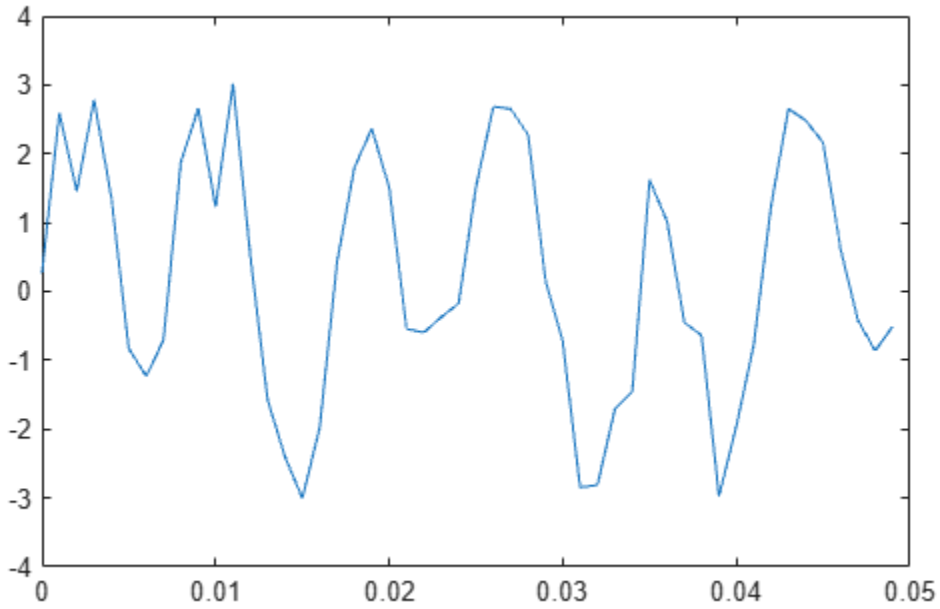
where the MATLAB® colon operator (`:`) creates a 1001-element row vector that represents time running from 0 to 1 seconds in steps of 1 ms. The transpose operator (`'`) changes the row vector into a column; the semicolon (`;`) tells MATLAB to compute, but not display, the result.

Given `t`, you can create a sample signal y consisting of two sinusoids, one at 50 Hz and one at 120 Hz with twice the amplitude.

```
y = sin(2*pi*50*t) + 2*sin(2*pi*120*t);
```

The new variable `y`, formed from vector `t`, is also 1001 elements long. You can add normally distributed white noise to the signal and plot the first 50 points:

```
yn = y + 0.5*randn(size(t));
plot(t(1:50),yn(1:50))
```



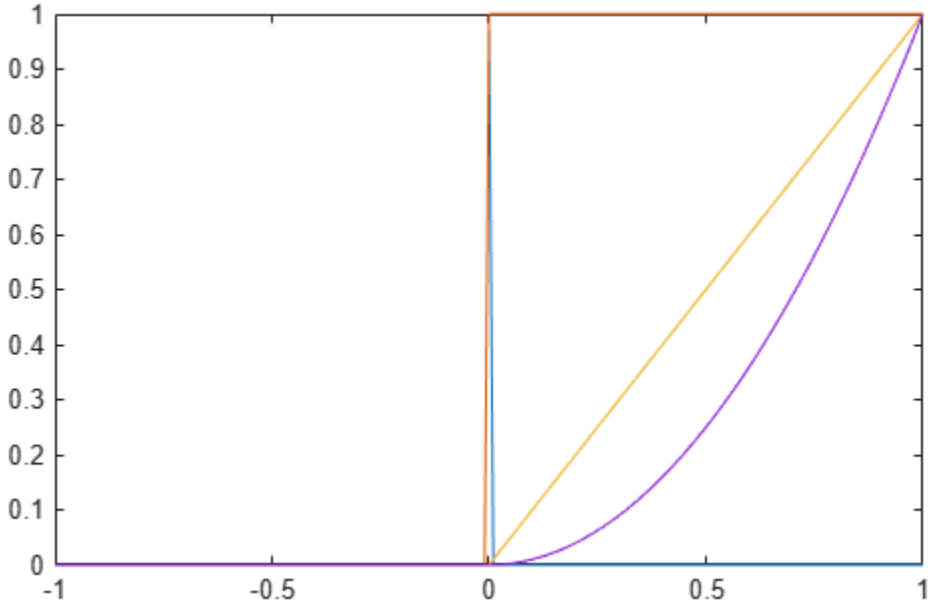
Impulse, Step, and Ramp Functions

Since MATLAB® is a programming language, an endless variety of different signals is possible. Here are some statements that generate a unit impulse, a unit step, a unit ramp, and a unit parabola.

```
t = (-1:0.01:1)';

impulse = t==0;
unitstep = t>=0;
ramp = t.*unitstep;
quad = t.^2.*unitstep;
```

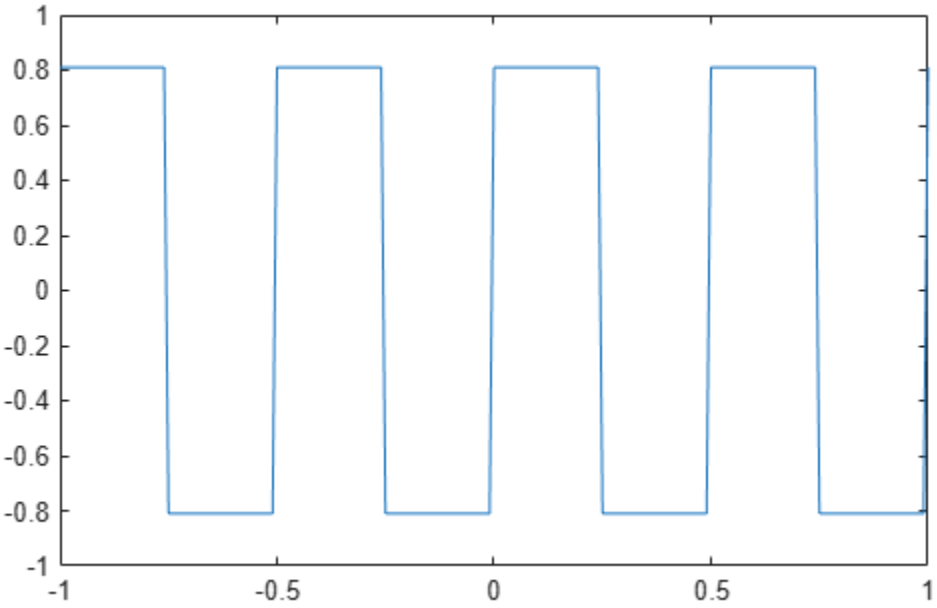
All of these sequences are column vectors that inherit their shapes from `t`. Plot the sequences.

```
plot(t,[impulse unitstep ramp quad])
```



Generate and plot a square wave with period 0.5 and amplitude 0.81.

```
sqwave = 0.81*square(4*pi*t);
plot(t,sqwave)
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

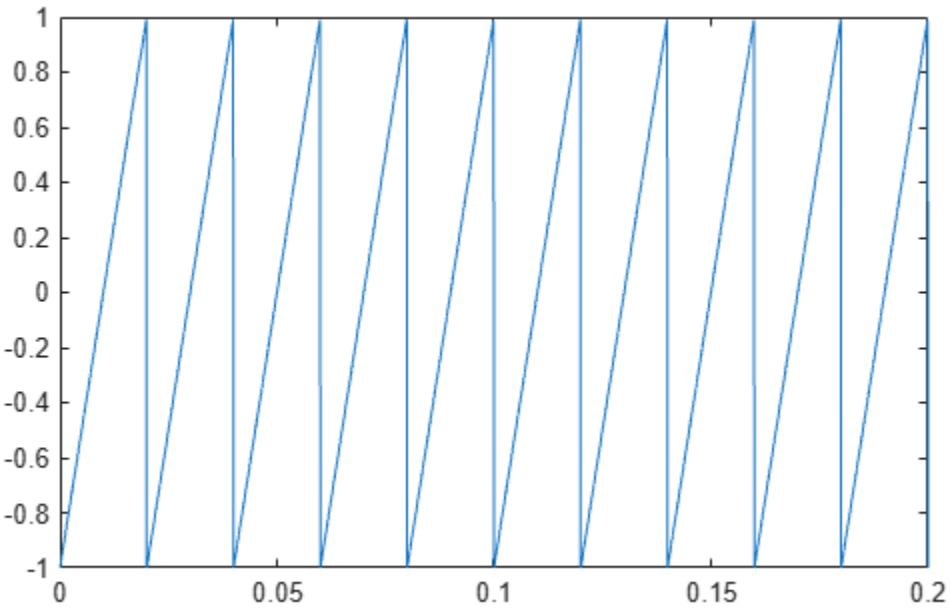
Common Periodic Waveforms

Signal Processing Toolbox™ provides functions for generating widely used periodic waveforms.

- `sawtooth` generates a sawtooth wave with peaks at ±1 and a period of 2π. An optional width parameter specifies a fractional multiple of 2π at which the signal maximum occurs.
- `square` generates a square wave with a period of 2π. An optional parameter specifies the *duty cycle*, the percent of the period for which the signal is positive.

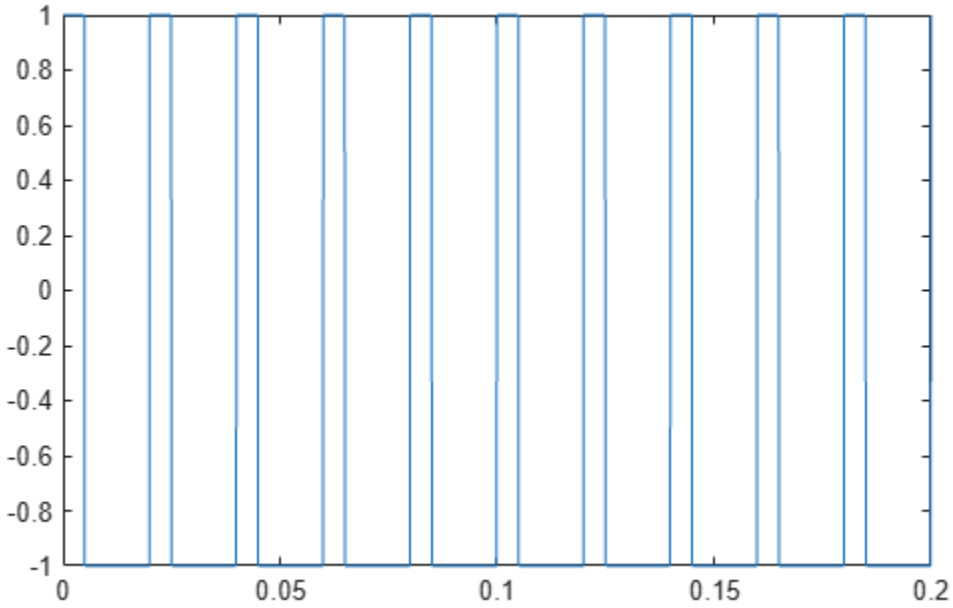
Generate 1.5 seconds of a 50 Hz sawtooth wave with a sample rate of 10 kHz. Plot 0.2 seconds of the generated waveform.

```
fs = 10e3;
t = 0:1/fs:1.5;
x = sawtooth(2*pi*50*t);

plot(t,x)
axis([0 0.2 -1 1])
```



Generate 1.5 seconds of a 50 Hz square wave with a sample rate of 10 kHz. Specify a duty cycle of 25%. Plot 0.2 seconds of the generated waveform.

```
fs = 10e3;
t = 0:1/fs:1.5;
x = square(2*pi*50*t,25);

plot(t,x)
axis([0 0.2 -1 1])
```

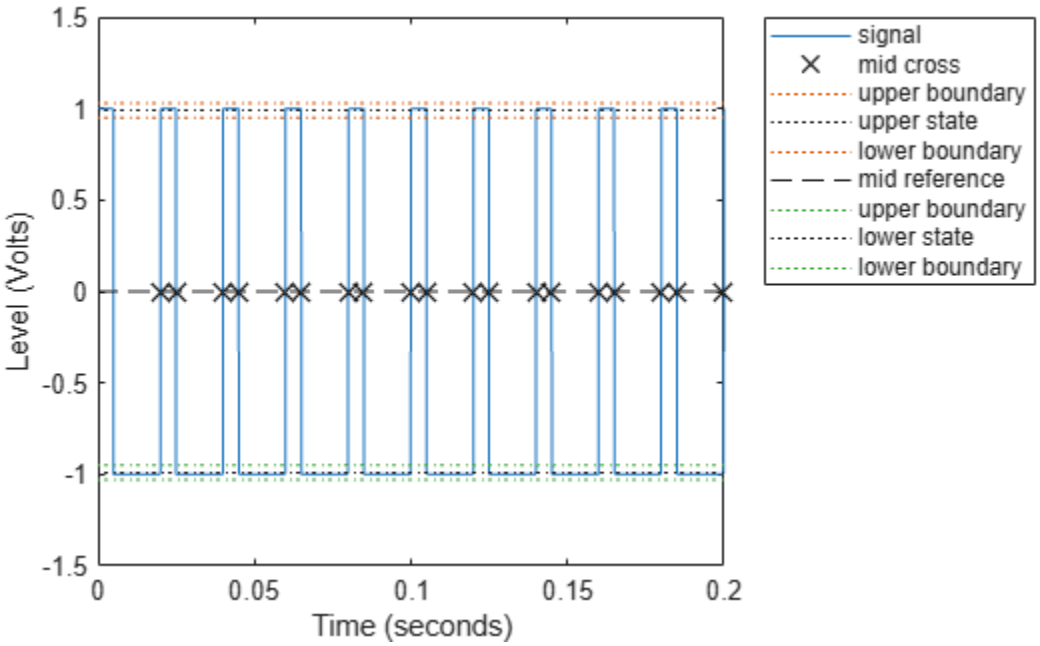


Use the `dutycycle` function to verify that the duty cycle of the square wave is the specified value. Use the function with no output arguments to plot the waveform, the location of the mid-reference level instants, the associated reference levels, the state levels, and the associated lower and upper state boundaries.

```
dc = dutycycle(x,fs);
dc = dc(1)
```

```
dc =
0.2500
```

```
dutycycle(x,fs);
xlim([0 0.2])
```



See Also

`dutycycle` | `sawtooth` | `square`

Common Aperiodic Waveforms

Signal Processing Toolbox™ provides functions for generating several widely used aperiodic waveforms.

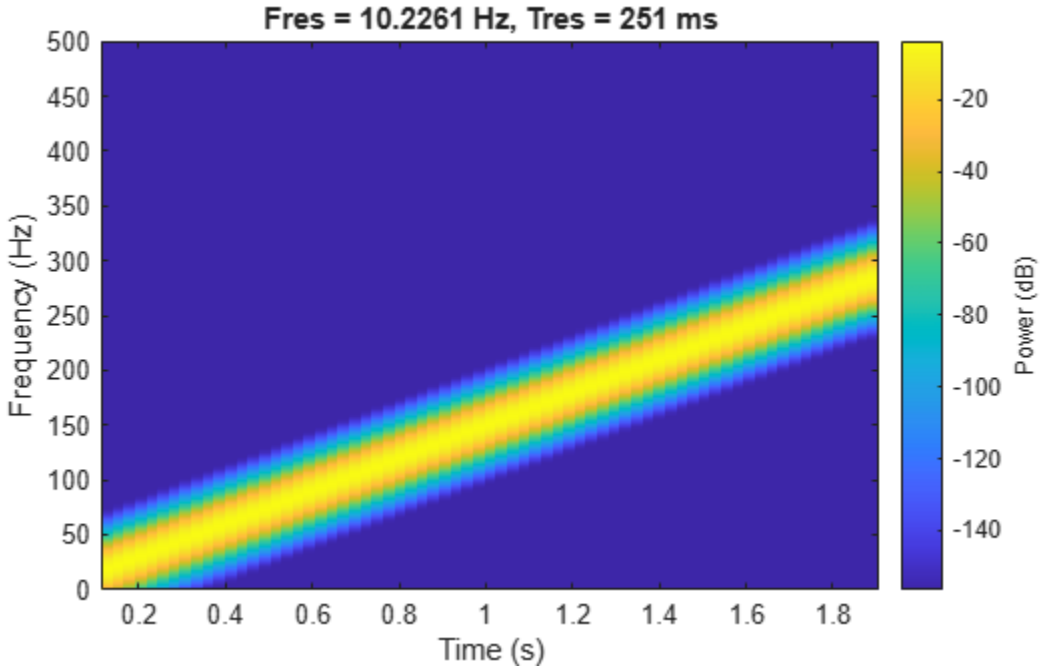
- `gauspuls` generates a Gaussian-modulated sinusoidal pulse with a specified time, center frequency, and fractional bandwidth. Optional parameters return in-phase and quadrature pulses, the RF signal envelope, and the cutoff time for the trailing pulse envelope.
- `chirp` generates a linear, logarithmic, or quadratic swept-frequency sinusoidal signal. An optional parameter specifies alternative sweep methods. An optional parameter allows an initial phase to be specified in degrees.

Compute 2 seconds of a linear chirp signal with a sample rate of 1 kHz that starts at DC and crosses 150 Hz at 1 second.

```
t = 0:1/1000:2;
y = chirp(t,0,1,150);
```

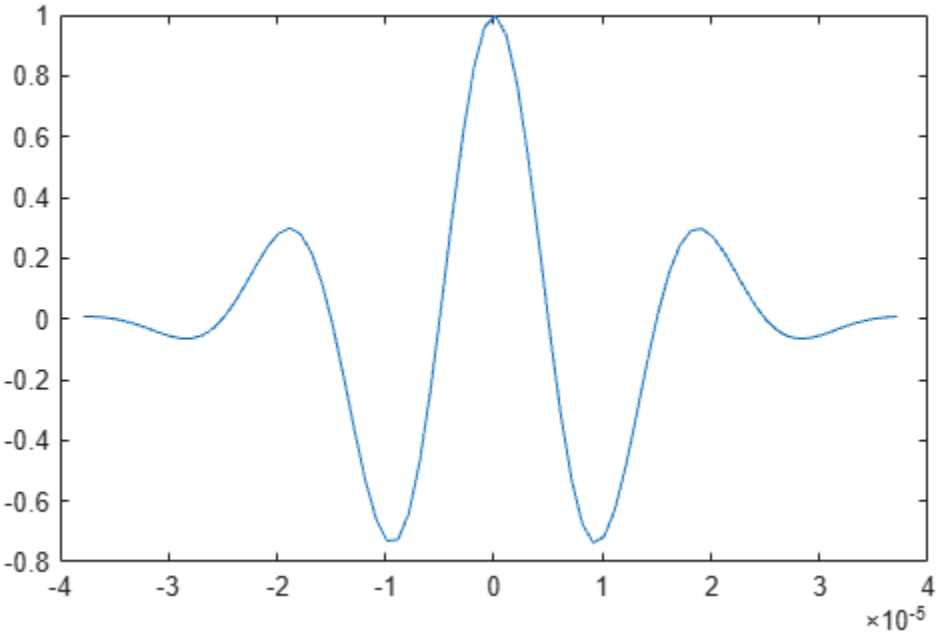
Plot the spectrogram of the chirp. Specify 90% of overlap between adjoining windowed segments.

```
pspectrum(y,t,'spectrogram','OverlapPercent',90)
```



Use `gauspuls` to plot a 50 kHz Gaussian RF pulse with 60% bandwidth, sampled at a rate of 1 MHz. Truncate the pulse where the envelope falls 40 dB below the peak.

```
tc = gauspuls('cutoff',50e3,0.6,[],-40);
t = -tc : 1e-6 : tc;
yi = gauspuls(t,50e3,0.6);
plot(t,yi)
```



See Also

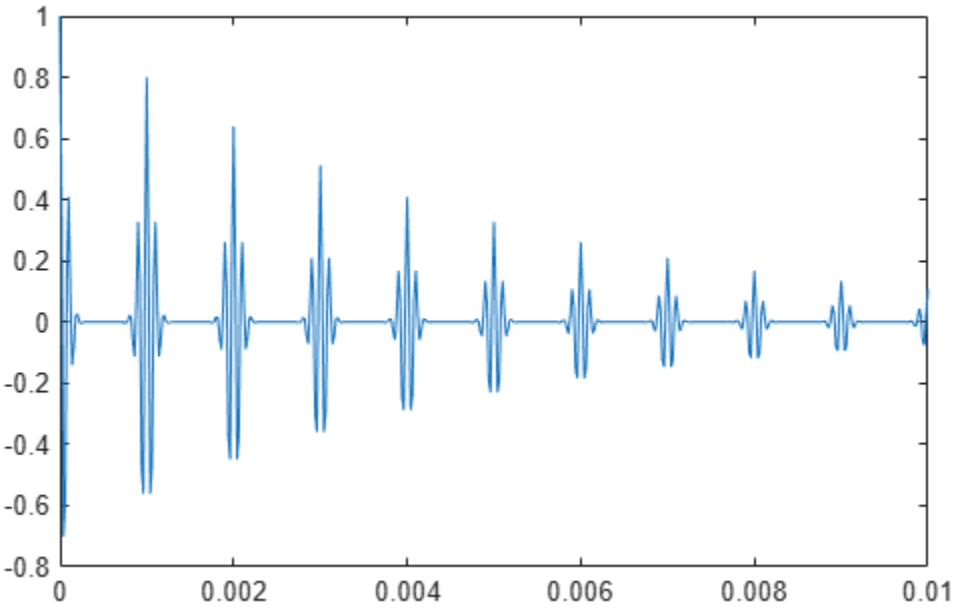
`chirp` | `gauspuls` | `pspectrum`

The pulstran Function

The `pulstran` function generates pulse trains from either continuous or sampled prototype pulses. This example generates a pulse train consisting of the sum of multiple delayed interpolations of a Gaussian pulse.

The pulse train is defined to have a sample rate of 50 kHz, a pulse train length of 10 ms, and a pulse repetition rate of 1 kHz. `T` specifies the time instants at which the pulse train is sampled. `D` specifies the delay to each pulse repetition in the first column and an optional attenuation for each repetition in the second column. To construct the pulse train, pass the name of the `gauspuls` function to `pulstran`, along with additional parameters that specify a 10 kHz Gaussian pulse with 50% bandwidth.

```
T = 0:1/50e3:10e-3;
D = [0:1/1e3:10e-3;0.8.^(0:10)]';

Y = pulstran(T,D,'gauspuls',10e3,0.5);

plot(T,Y)
```



See “Compute Envelope Spectrum of Vibration Signal” for an example that uses the `pulstran` function to generate vibration data for bearing analysis.

See Also

`pulstran`

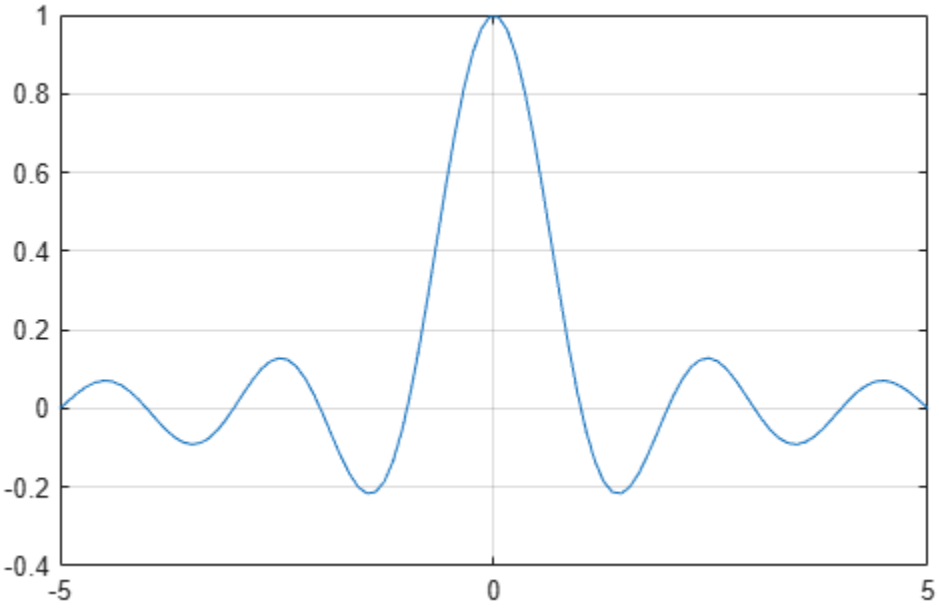
The Sinc Function

The `sinc` function computes the mathematical sinc function for an input vector or matrix x. Viewed as a function of time, or space, the sinc function is the inverse Fourier transform of the rectangular pulse in frequency centered at zero, with width 2π and unit height:

$$\operatorname{sinc} x = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega x}\, d\omega = \begin{cases} \dfrac{\sin \pi x}{\pi x}, & x \neq 0, \\ 1, & x = 0\,. \end{cases}$$

To plot the sinc function for a linearly spaced vector with values ranging from -5 to 5, use these commands:

```
x = linspace(-5,5);
y = sinc(x);
plot(x,y)
grid
```



See Also

`diric` | `sinc`

The Dirichlet Function

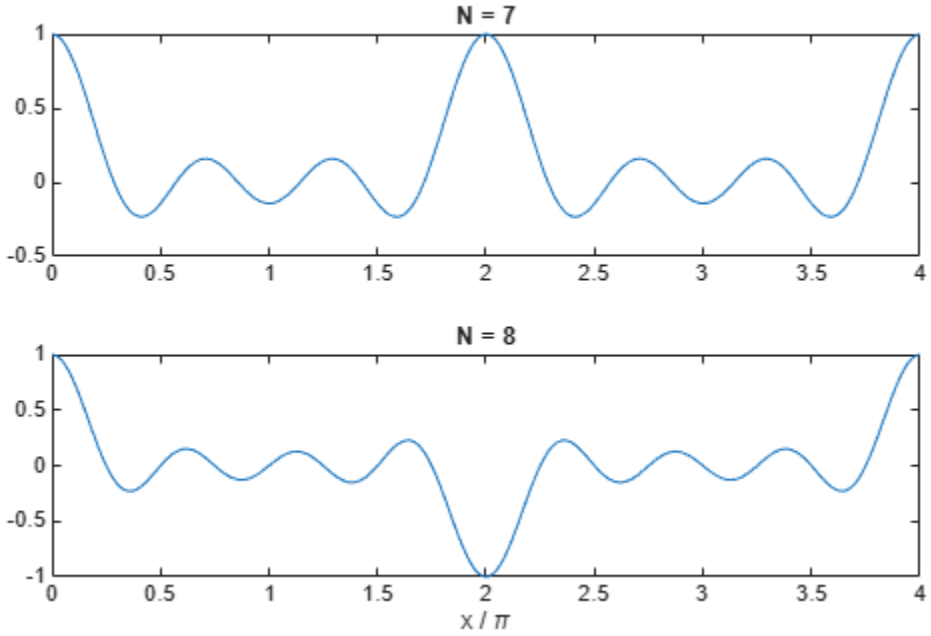
The function `diric` computes the Dirichlet function, sometimes called the *periodic sinc* or *aliased sinc* function, for an input vector or matrix `x`. The Dirichlet function is defined by

$$D(x) = \begin{cases} \frac{\sin(Nx/2)}{N\sin(x/2)}, \\ \&x \neq 2\pi k, \end{cases}$$

where N is a user-specified positive integer. For N odd, the Dirichlet function has a period of 2π; for N even, its period is 4π. The magnitude of this function is $1/N$ times the magnitude of the discrete-time Fourier transform of the N-point rectangular window.

To plot the Dirichlet function between 0 and 4π for $N = 7$ and $N = 8$, use

```
x = linspace(0,4*pi,300);

subplot(2,1,1)
plot(x/pi,diric(x,7))
title('N = 7')

subplot(2,1,2)
plot(x/pi,diric(x,8))
title('N = 8')
xlabel('x / \pi')
```



See Also

`diric` | `sinc`

Working with Data

Data Precision

All Signal Processing Toolbox functions accept double-precision inputs. If you input single-precision floating-point or integer data types, you should not expect to receive correct results and in many cases, an error will occur. DSP System Toolbox™ and Fixed-Point Designer™ products enable single-precision floating-point and fixed-point support for most `dfilt` structures.

Selected Bibliography

Algorithm development for Signal Processing Toolbox functions has drawn heavily upon the references listed below. All are recommended to the interested reader who needs to know more about signal processing than is covered in this manual.

References

[1] Crochiere, R. E., and Lawrence R. Rabiner. *Multi-Rate Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1983. pp. 88–91.

[2] IEEE. *Programs for Digital Signal Processing*. IEEE Press. New York: John Wiley & Sons, 1979.

[3] Jackson, L. B. *Digital Filters and Signal Processing*. Third Ed. Boston: Kluwer Academic Publishers, 1989.

[4] Kay, Steven M. *Modern Spectral Estimation*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[5] Oppenheim, Alan V., and Ronald W. Schafer. *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[6] Parks, Thomas W., and C. Sidney Burrus. *Digital Filter Design*. New York: John Wiley & Sons, 1987.

[7] Percival, D. B., and A. T. Walden. *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*. Cambridge: Cambridge University Press, 1993.

[8] Pratt, W. K. *Digital Image Processing*. New York: John Wiley & Sons, 1991.

[9] Proakis, John G., and Dimitris G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice Hall, 1996.

[10] Rabiner, Lawrence R., and Bernard Gold. *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1975.

[11] Welch, P. D. “The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms.” *IEEE® Transactions on Audio and Electroacoustics*. Vol. AU-15, 1967. pp. 70–73.