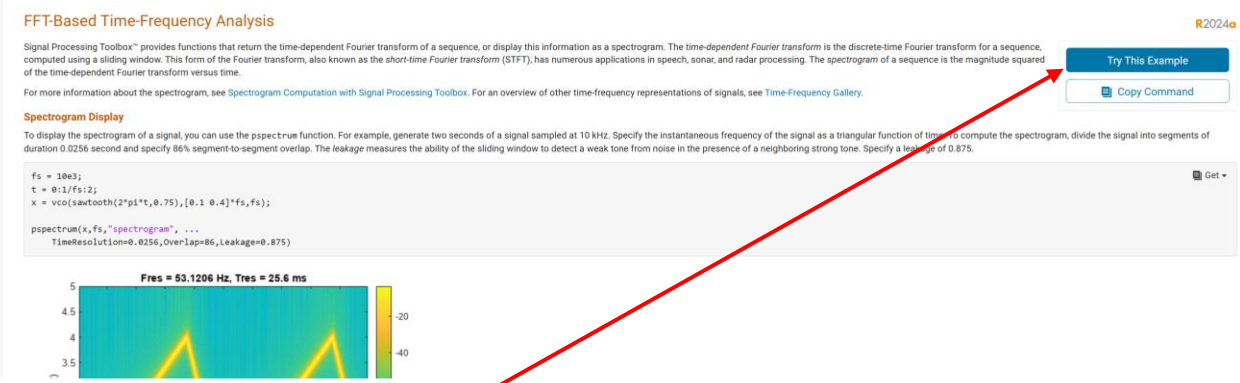
If you do not have access to MATLAB, try the online examples such as:

FFT-Based Time-Frequency Analysis - MATLAB & Simulink (mathworks.com)



FFT-Based Time-Frequency Analysis

Signal Processing Toolbox™ provides functions that return the time-dependent Fourier transform of a sequence, or display this information as a spectrogram. The *time-dependent Fourier transform* is the discrete-time Fourier transform for a sequence, computed using a sliding window. This form of the Fourier transform, also known as the *short-time Fourier transform* (STFT), has numerous applications in speech, sonar, and radar processing. The *spectrogram* of a sequence is the magnitude squared of the time-dependent Fourier transform versus time.

For more information about the spectrogram, see Spectrogram Computation with Signal Processing Toolbox. For an overview of other time-frequency representations of signals, see Time-Frequency Gallery.

Spectrogram Display

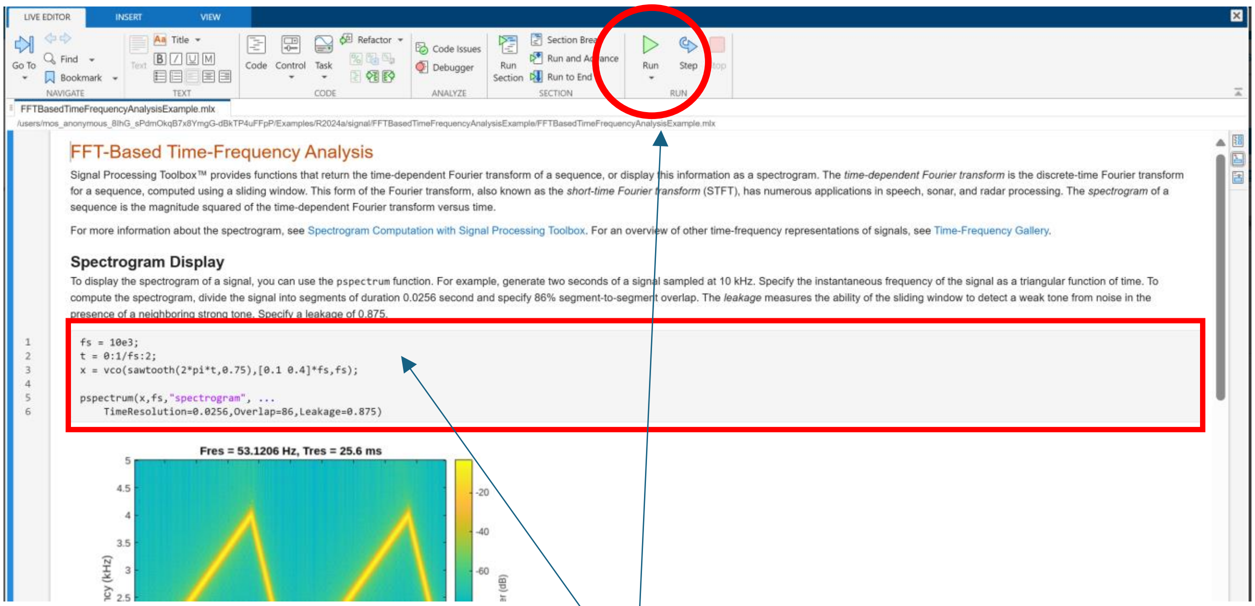
To display the spectrogram of a signal, you can use the `pspectrum` function. For example, generate two seconds of a signal sampled at 10 kHz. Specify the instantaneous frequency of the signal as a triangular function of time. To compute the spectrogram, divide the signal into segments of duration 0.0256 second and specify 86% segment-to-segment overlap. The *leakage* measures the ability of the sliding window to detect a weak tone from noise in the presence of a neighboring strong tone. Specify a leakage of 0.875.

```
fs = 10e3;
t = 0:1/fs:2;
x = vco(sawtooth(2*pi*t,0.75),[0.1 0.4]*fs,fs);

pspectrum(x,fs,"spectrogram", ...
    TimeResolution=0.0256,Overlap=86,Leakage=0.875)
```

Click on “Try this example”.

A “live” code window will open, that you can modify.



LIVE EDITOR
INSERT
VIEW
Go To
Find
Bookmark
NAVIGATE
Title
Text
TEXT
Code
Control
Task
Refactor
CODE
Code Issues
Debugger
ANALYZE
Run Section
Section Brea
Run and Ad ance
Run to End
SECTION
Run
Step
top
RUN
FFTBasedTimeFrequencyAnalysisExample.mlx
/users/mos_anonymous_8lhG_sPdmOkqB7x8YmgG-dBkTP4uFFpP/Examples/R2024a/signal/FFTBasedTimeFrequencyAnalysisExample/FFTBasedTimeFrequencyAnalysisExample.mlx

FFT-Based Time-Frequency Analysis

Signal Processing Toolbox™ provides functions that return the time-dependent Fourier transform of a sequence, or display this information as a spectrogram. The *time-dependent Fourier transform* is the discrete-time Fourier transform for a sequence, computed using a sliding window. This form of the Fourier transform, also known as the *short-time Fourier transform* (STFT), has numerous applications in speech, sonar, and radar processing. The *spectrogram* of a sequence is the magnitude squared of the time-dependent Fourier transform versus time.

For more information about the spectrogram, see Spectrogram Computation with Signal Processing Toolbox. For an overview of other time-frequency representations of signals, see Time-Frequency Gallery.

Spectrogram Display

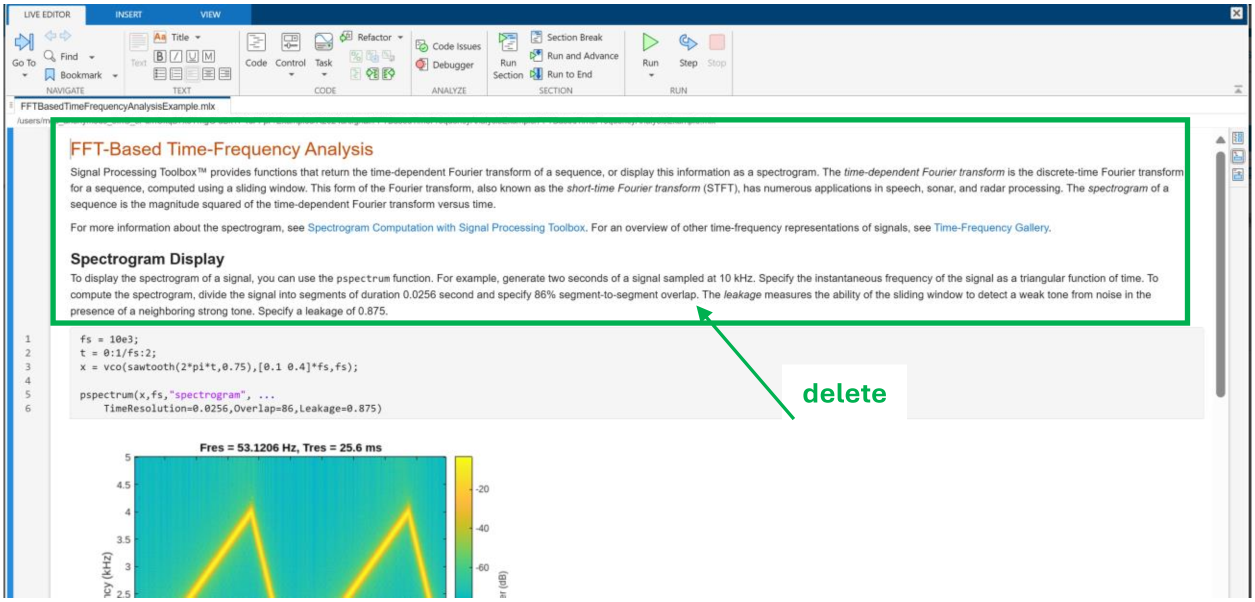
To display the spectrogram of a signal, you can use the `pspectrum` function. For example, generate two seconds of a signal sampled at 10 kHz. Specify the instantaneous frequency of the signal as a triangular function of time. To compute the spectrogram, divide the signal into segments of duration 0.0256 second and specify 86% segment-to-segment overlap. The *leakage* measures the ability of the sliding window to detect a weak tone from noise in the presence of a neighboring strong tone. Specify a leakage of 0.875.

```
fs = 10e3;
t = 0:1/fs:2;
x = vco(sawtooth(2*pi*t,0.75),[0.1 0.4]*fs,fs);

pspectrum(x,fs,"spectrogram", ...
    TimeResolution=0.0256,Overlap=86,Leakage=0.875)
```

You can modify the code and then hit run.

You can also delete the extra text and live code windows if not needed, and type in the code you want to run to find the filter coefficients.



FFT-Based Time-Frequency Analysis

Signal Processing Toolbox™ provides functions that return the time-dependent Fourier transform of a sequence, or display this information as a spectrogram. The *time-dependent Fourier transform* is the discrete-time Fourier transform for a sequence, computed using a sliding window. This form of the Fourier transform, also known as the *short-time Fourier transform* (STFT), has numerous applications in speech, sonar, and radar processing. The *spectrogram* of a sequence is the magnitude squared of the time-dependent Fourier transform versus time.

For more information about the spectrogram, see Spectrogram Computation with Signal Processing Toolbox. For an overview of other time-frequency representations of signals, see Time-Frequency Gallery.

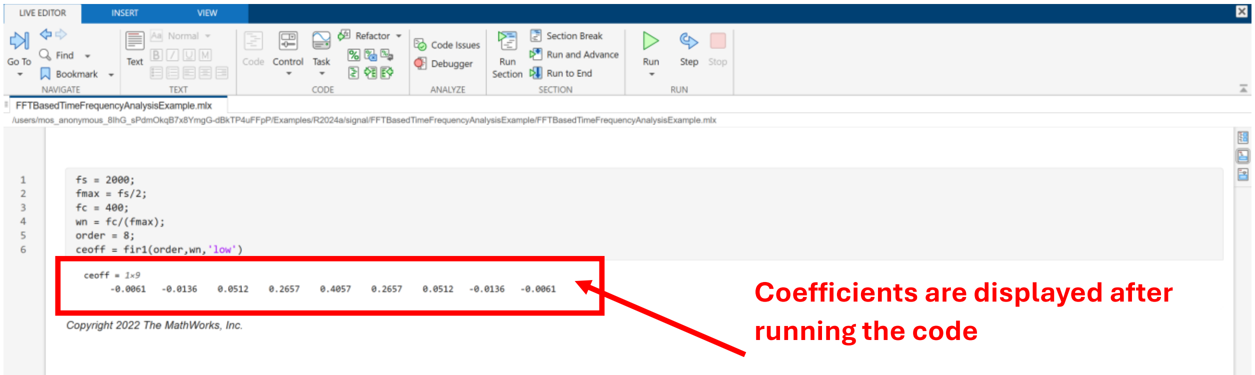
Spectrogram Display

To display the spectrogram of a signal, you can use the `pspectrum` function. For example, generate two seconds of a signal sampled at 10 kHz. Specify the instantaneous frequency of the signal as a triangular function of time. To compute the spectrogram, divide the signal into segments of duration 0.0256 second and specify 86% segment-to-segment overlap. The *leakage* measures the ability of the sliding window to detect a weak tone from noise in the presence of a neighboring strong tone. Specify a leakage of 0.875.

```
fs = 10e3;
t = 0:1/fs:2;
x = vco(sawtooth(2*pi*t,0.75),[0.1 0.4]*fs,fs);

pspectrum(x,fs,"spectrogram", ...
    TimeResolution=0.0256,Overlap=86,Leakage=0.875)
```

Fres = 53.1206 Hz, Tres = 25.6 ms
5
4.5
4
3.5
3
2.5
ncy (kHz)
-20
-40
-60
er (dB)
delete


LIVE EDITOR
INSERT
VIEW
Go To
Find
Bookmark
NAVIGATE
Normal
Text
TEXT
Code
Control
Task
Refactor
CODE
Code Issues
Debugger
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
FFTBasedTimeFrequencyAnalysisExample.mlx
/users/mos_anonymous_8lhG_sPdmOkqB7x8YmgG-dBkTP4uFFpP/Examples/R2024a/signal/FFTBasedTimeFrequencyAnalysisExample/FFTBasedTimeFrequencyAnalysisExample.mlx

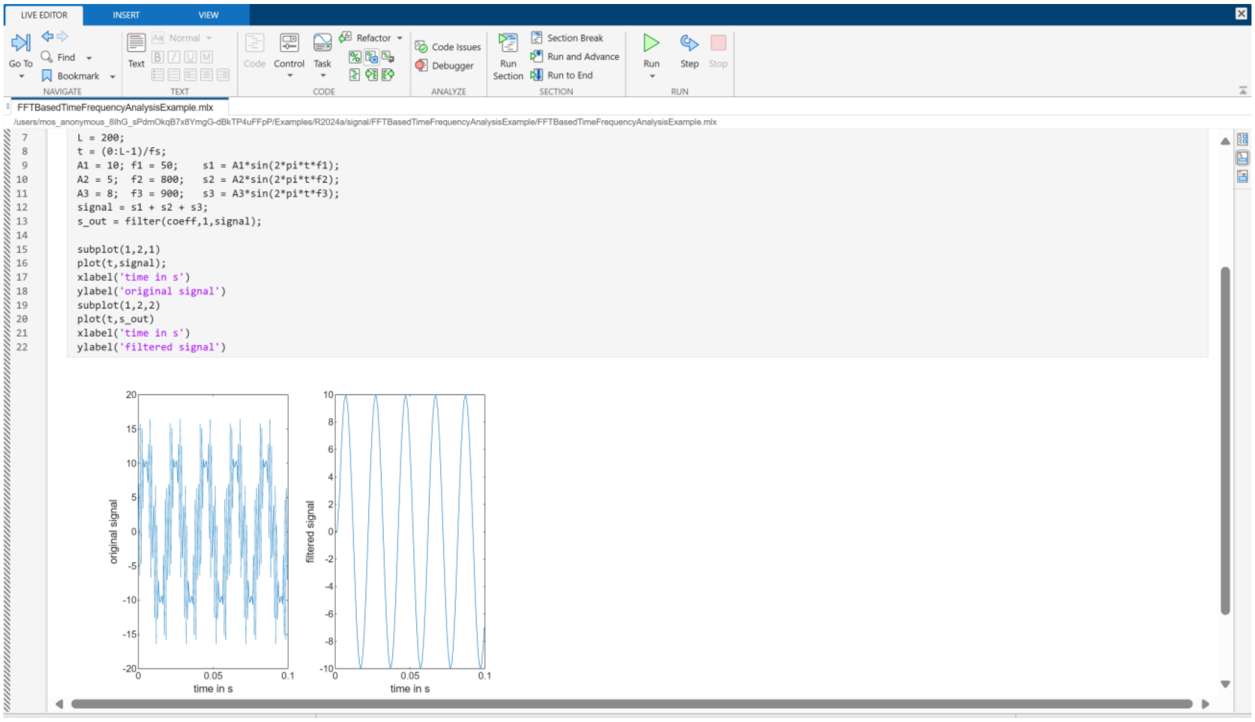
```
fs = 2000;
fmax = fs/2;
fc = 400;
wn = fc/(fmax);
order = 8;
ceoff = fir1(order,wn,'low')
```

```
ceoff = 1×9
    -0.0061   -0.0136    0.0512    0.2657    0.4057    0.2657    0.0512   -0.0136   -0.0061
```

Copyright 2022 The MathWorks, Inc.

Coefficients are displayed after running the code

You can also test your filter by adding the extra code for generating an input signal and passing it through the generated filter and then plotting the input and output signals. Then you hit 'run' to obtain the following results:



/users/mos_anonymous_8lhG_sPdmOkqB7x8YmgG-dBkTP4uFFpP/Examples/R2024a/signal/FFTBasedTimeFrequencyAnalysisExample/FFTBasedTimeFrequencyAnalysisExample.mlx

```
L = 200;
t = (0:L-1)/fs;
A1 = 10; f1 = 50;    s1 = A1*sin(2*pi*t*f1);
A2 = 5;  f2 = 800;   s2 = A2*sin(2*pi*t*f2);
A3 = 8;  f3 = 900;   s3 = A3*sin(2*pi*t*f3);
signal = s1 + s2 + s3;
s_out = filter(coeff,1,signal);

subplot(1,2,1)
plot(t,signal);
xlabel('time in s')
ylabel('original signal')
subplot(1,2,2)
plot(t,s_out)
xlabel('time in s')
ylabel('filtered signal')
```