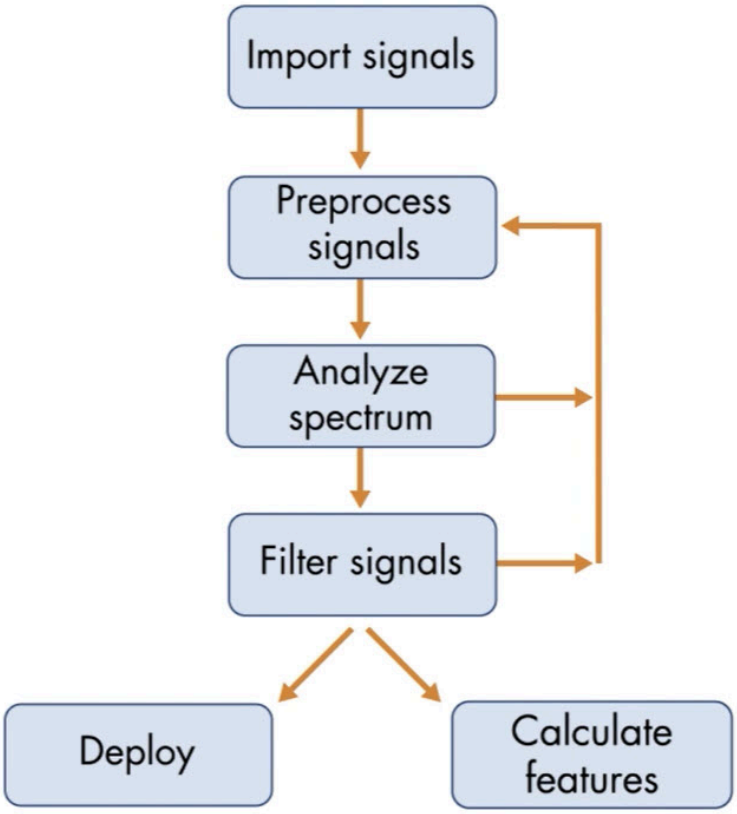
Signal Processing Onramp

Conclusion

Summary

Summary

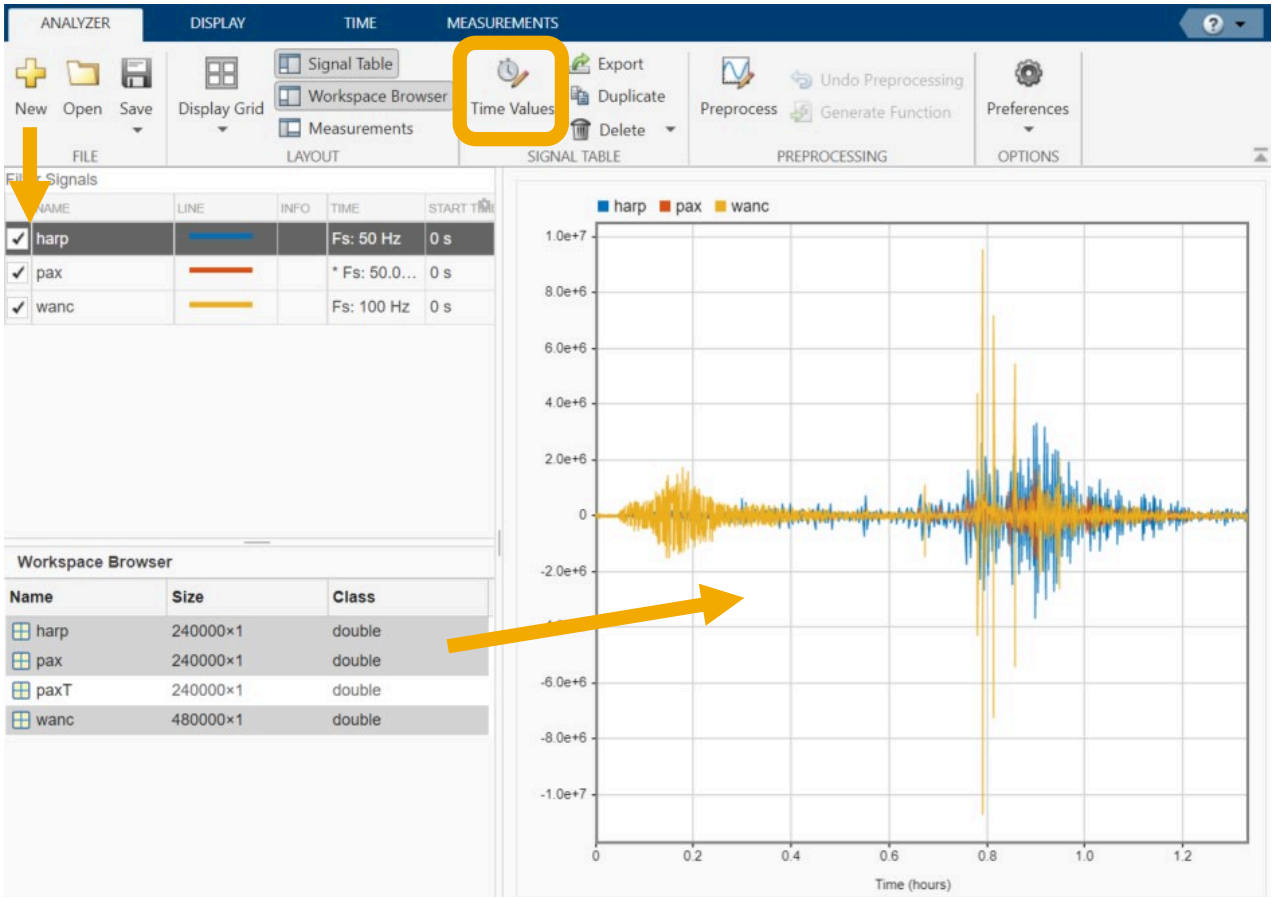
Signal Processing Onramp Workflow



1. Import the signals into MATLAB
2. Preprocess the signals in the time domain to prepare them for analysis
3. Look at the power spectrum in the frequency domain
4. Filter the signals to keep only relevant frequency content
5. Revisit earlier steps in the process to refine your algorithm
6. Extract information from the signals or deploy your algorithm to hardware

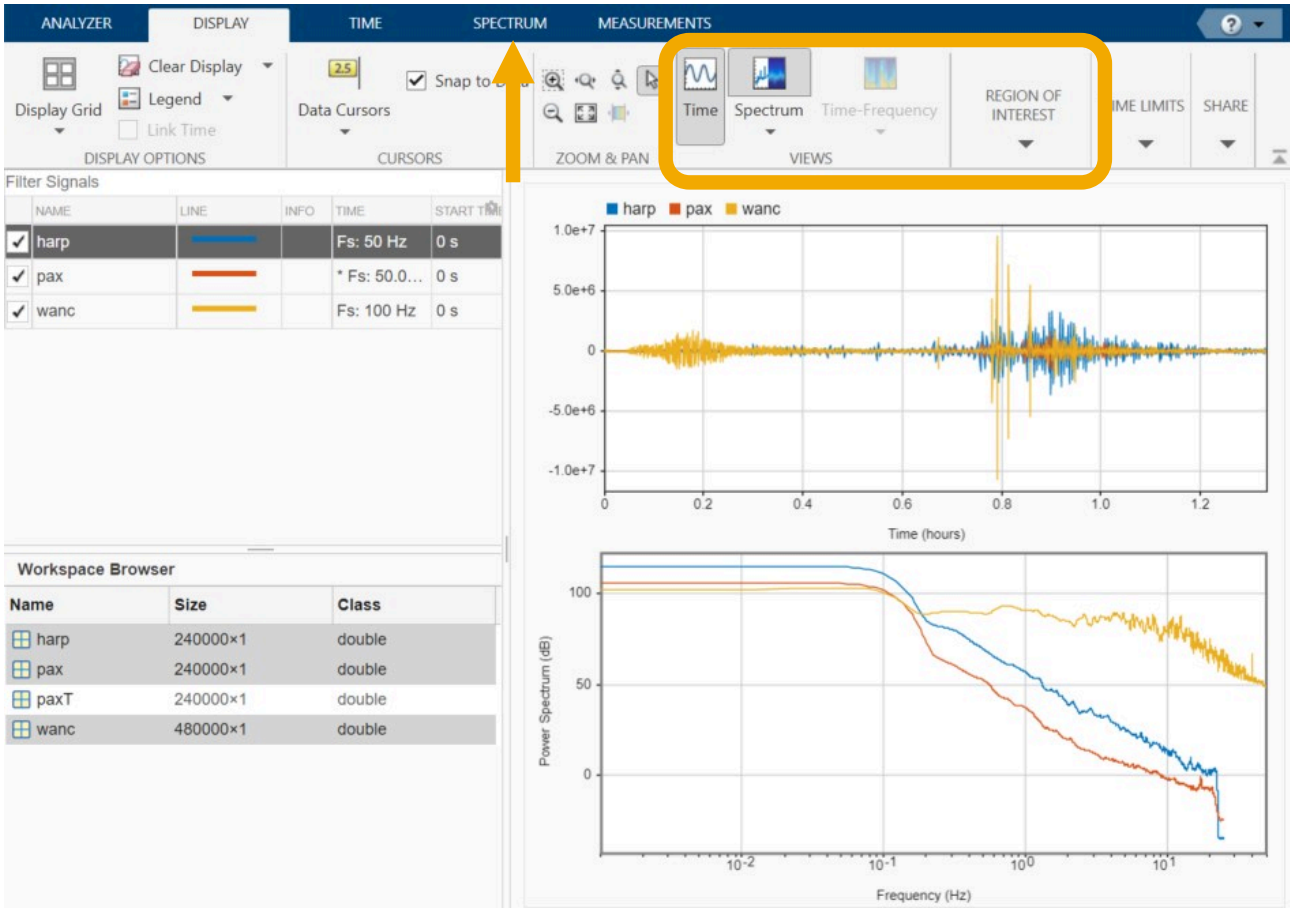
Signal Analyzer Workflow

Import Signals



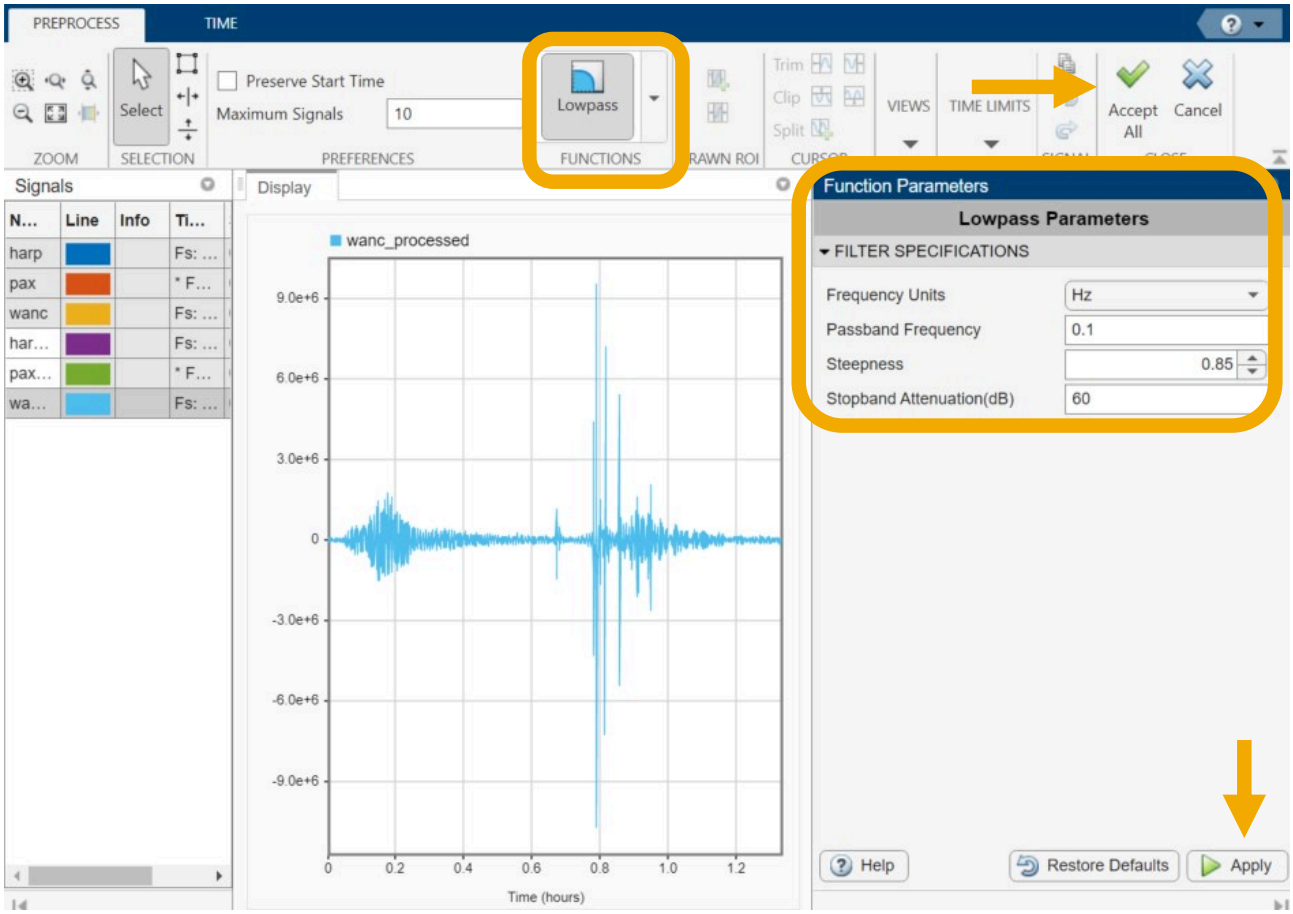
1. Import signals from files to the Workspace using the `readmatrix` function.
2. Set the time information for each signal with **Time Values**.
3. Drag signals from the **Workspace Browser** to the axes to display them.
4. Display different signals using the checkboxes in the **Filter Signals** panel.

Visualize Signals



1. Use the **Display** tab to visualize your signal in the time domain, frequency domain, or time-frequency domain.
2. To customize the visualization, use the appropriate tab. For example, you can customize the power spectrum in the **Spectrum** tab.

Preprocess and Filter Signals



1. To modify signals, click **Preprocess** in the **Analyze** tab to enter preprocessing mode.
2. Duplicate signals before modifying them so that you can compare your signals before and after applying techniques.
3. Select preprocessing techniques from the **Functions** dropdown.
4. The **Function Parameters** panel will open. Set the appropriate specifications and click **Apply**.
5. Click **Accept All** to leave preprocessing mode.

Generate Signals

The sample rate is the number of samples that are generated per second.

Specify the start and end times of your signal. This time vector extends from 0 seconds to 1 second. The spacing between each point is $\frac{1}{fs}$.

Calculate a sine wave at each point in `t`. Change the value of `f` to change the signal's frequency.

To make the generated signal more realistic, you can add noise by adding a vector of random numbers.

```
fs = 100;

t = 0:1/fs:1;

sig = sin(2*pi*f*t);

noise = 0.1*randn(size(sig));
sigNoisy = sig+sigNoisy;
```

Function Reference

Function	Description	Example
resample	Resample nonuniform signal to a uniform sample rate	`sig = resample(sig,tx,fs)`
resample	Resamples the input sequence at p/q times the original sample rate	`sig = resample(sig,p,q)`
alignsignals	Align two signals by delaying the earliest signal	`[xa,ya] = alignsignals(x,y)`
pspectrum	Display the power spectrum	`pspectrum(sig)`
pspectrum	Display the spectrogram	`pspectrum(sig,"spectrogram")`
cwt	Display the continuous wavelet transform *Requires the Wavelet Toolbox*	`cwt(sig,fs)`
lowpass	Lowpass filter a signal	`lowpass(sig,10)`
bandpass	Bandpass filter a signal	`bandpass(sig,[8 12])`
lowpass	Filter with modified steepness	`lowpass(sig,10,"Steepness",0.95);`
Find Local Extrema task	Find local maxima and minima in the Live Editor	

Compare Visualizations