Lab 8. Signal Analysis Using Matlab Simulink

Introduction

The Matlab Simulink software allows you to model digital signals, examine power spectra of digital signals, represent LTI systems in different formats, help turn specifications for IIR and FIR filters into working designs, and other operations.

This lab will briefly describe how to use Matlab’s Simulink software for signal analysis.

Computer Setup

You will need to make sure some examples and exercises are available on your computer before beginning the lab.

1. Start Matlab.
2. Put the CD in the CD-ROM drive.
3. Set the Matlab current directory to the CD-ROM drive.
4. Run the Matlab install script `install_training` on the CD. To do so, type the name at the Matlab command prompt:
   
   `install_training`

5. The script should open a graphical user interface (GUI) that will guide you through the installation procedure. In the GUI, you have these options:
   - Choose a class root directory for the files.
   - Choose the courses for which you want to install the files.
   - Add the course directory to your Matlab path and edit your `startup.m` file so that the files are on the path each time you start Matlab.

The installation creates a subdirectory of your chosen root directory (the default is `C:\class`) called `coursefiles`. This directory has subdirectories for each of the courses you install, labeled by course number. These individual course directories contain the examples and exercises
for the courses. You may want to change the work directory to your M: drive and put it in the path.

©2009GM

Matlab Setup for Building Signal Processing Blockset Models

When you create a model using blocks from the Signal Processing Blockset, you may want to adjust certain Simulink settings to reflect the discrete time nature of a digital model.

The Signal Processing Blockset provides an M-file, dspstartup.m, that lets you automate this configuration process so that every new model you create is preconfigured for simulating a digital (discrete) system.

Statistical Signal Processing

Repeated observations (or simulations within the Simulink environment) of a random process \( X \) under the same environmental conditions yield different results. The signal value of a simulation, \( x(n) \), at sample \( n \) is not constant among the realizations in an ensemble; rather, the ensemble is characterized by a set of probability density functions. Each individual signal value \( x(n) \) of a specific realization in the ensemble can be thought of as the outcome of an underlying random variable. The collection of these random variables, one for each sample time, comprise the random process.

Working with random signals is easier when one or both of the following properties can be shown to hold.

- A signal is said to be stationary if the statistics over the simulation do not change with a shift in time, i.e., if the random variables \( x(n) \) and \( x(n + N) \) have the same statistics for every integer value of \( N \).
- A signal is said to be ergodic if the ensemble averages are equal to the time averages of any sample function. In other words, the statistics for any given realization in the ensemble are the same as the statistics for the ensemble.

Note that an ergodic signal must be stationary, but a stationary signal is not necessarily ergodic.
Statistics

The Signal Processing Blockset has blocks to extract basic statistics from stochastic signals, such as mean, median, variance, and correlation. These are the building blocks for much of statistical signal processing.

Simple
- Mean
  \[ \mu_x = EX \]
- Standard Deviation and Variance
  \[ \sigma_x^2 = E\{X^2\} - E^2\{X\} \]
- Minimum and Maximum

Correlation
- Autocorrelation and Cross-Correlation
  \[ R_{xy}(\tau) = E\{X(t)Y^*(t + \tau)\} \]

Trending
- Detrend
- Histogram
- Sort

Example: Crosscorrelation

In a simple target ranging system, a transmitted signal \( x \) is compared with a received signal \( y \). \( y \) is modeled by the equation

\[ y(n) = \alpha x(n - d) + \beta \]

where \( \alpha \) is an attenuation factor, \( d \) is a time delay, and \( \beta \) is channel noise. If \( T \) is the return time for the signal, then \( x \) and \( y \) should be correlated at \( n = T \). The target will be located at a distance of \( vT \), where \( v \) is the channel speed of the signal.

Try:

```
multipathcorr
```

Example: Mean Block

The Mean block in the Signal Processing Blockset performs calculates the mean for a portion of the input sequence. You can use this block with the Buffer block as an averaging filter.

The Buffer block generates a sliding window. The output buffer size \( N \) indicates the length of the window, while \( N - M \) (where \( M \) is the buffer overlap), specifies the number of samples
the window slides at each time step.

The Mean block also has a running mean option that performs a cumulative mean calculation on the input sequence.

Try:
running_mean

Example: Median Block

The Median block in the Signal Processing Blockset performs basic median operation on the input sequence. You can use this block with the Buffer Block as a median filter.

A median filter is a nonlinear filtering technique that uses a sliding window on a sequence. The filter replaces the center value in the window with the median value of all the points within the window.

The advantage of median filters over linear filters such as averaging filters comes from the robustness of the median with respect to outliers. Outlying data has much less effect on the center of a data set when the center is measured by the median rather than the mean. As a result, median filters are useful for removing spike noise involving extreme outliers while otherwise preserving the shape of the signal.

Try:
edit medianfilter
medianfilter

Discrete Fourier Transform (DFT)

The DFT of a real-valued \(N\)-point signal \(y\) is a complex-valued \(N\)-point conjugate-symmetric signal \(Y\), defined by the formula

\[
Y(n) = \sum_{k=0}^{N-1} y(k)e^{-j\frac{2\pi}{N}kn}
\]

where \(e^{-j\frac{2\pi}{N}}\) are the complex \(N^{th}\) roots of unity.

The points of the DFT correspond to the \(N\) discrete frequencies from 0 to \(2\pi\), or the DC frequency to the Sampling Frequency respectively.

Note
The DFT defines the discrete frequencies of a sampled periodic signal. The Discrete-Time
Fourier Transform (DTFT), on the other hand, defines all frequencies of a sampled aperiodic signal.

Try:
fftgui

Drag the mouse to move some of the points in the “real” window and see how the other windows change.

**Fast Fourier Transform (FFT)**

The FFT block in the Signal Processing Blockset uses an algorithm to calculate efficiently the DFT of an input signal. The algorithm requires that the input is frame-based with a power-of-two column length.

The FFT operation requires approximately $2n^2$ floating-point operations, where $n$ is the FFT length. Modern-day FFT algorithms have reduced the computational complexity from $n^2$ to $n \log_2 n$. By restricting $n$ to be a power of 2, an FFT of length $n$ requires fewer than $3n \log_2 n$ floating-point operations using the new algorithms. For $n = 2^{20}$ or 1,048,576, the calculation is faster by a factor of almost 35,000 times.

- In this example, the Buffer block sets the FFT window size by creating each windowed signal as a frame-based input.
- The FFT block returns a complex-valued sample-based signal containing the frequency domain data.
- The Vector Scope blocks are adjusted to display this frequency input.

The Matlab `fft` function is based on FFTW, “The Fastest Fourier Transform in the West,” developed by MIT graduate students Matteo Frigo and Steven G. Johnson. For information about FFTW, including a C subroutine library, go to:
http://www.fftw.org

**Example: DFT**

In this example, we use the FFT block to analyze signals in the frequency domain:

1. Open `medianfilter` and `running_mean` from previous examples.
2. Take the FFT of the signals before and after the filtering operations and use the Vector Scope block to observe the signals in the frequency domain.
3. Set the input domain of the vector scope to Frequency Sample time of the original time series to the proper rate.

4. Insert a Complex to Magnitude block before the Vector Scope block.

5. Use the IFFT block to view the original signal.

Try:

```plaintext
fft_running_mean
fft_median
```

Use the following blocks to perform the frequency analysis:

*Buffer, FFT, Complex to Magnitude-Angle, Freq, IFFT, Time*

FFT demos:

```plaintext
sigdemo1
phone
```

### Aliasing

Sampling an analog signal in the time domain has the effect of producing spectral copies at multiples of the sampling frequency. When the sampling frequency is less than the signal bandwidth, aliasing will occur.

The example `fft_aliasing` illustrates a 5Hz sine wave sampled at 15Hz and the same 5Hz signal sampled at 7.5Hz. In the first case, the original 5Hz signal is reconstructed without aliasing while in the second case, the 5Hz signal is aliased to 2.5Hz.

Try:

```plaintext
fftaliasing
```

### Additional Transforms

One of the drawbacks of the Fourier transform is that time information is lost when obtaining the frequency content. This is trivial for stationary signals since the signal’s statistics do not change with shifts in time, but for time-varying or “bursty” signals, the time information may be important to the user.

The Short-Time FFT block is one solution to this problem. By applying the FFT to a sliding window in the time domain, time is preserved by taking snapshots of the FFT spectrum throughout the simulation. The disadvantage of this method is that it is *uniform* on all time
intervals and is not robust to the local idiosyncrasies in the frequency content. As a result, some nonstationary characteristics of a signal can go undetected.

Wavelet analysis uses a more adaptable method, and is capable of revealing trends, breakdown points, and discontinuities in higher derivatives that Fourier analysis might be unable to capture. A wavelet is a waveform of effectively limited duration that has an average value of zero.

Wavelet analysis uses filters to separate the signal into its low and high frequency content. These signal components are manipulated with the sliding wavelet to produce the wavelet coefficients of similarity.

The Discrete Wavelet Transform (DWT) block decomposes the signal into a combination of the approximation and detail coefficients, which describe the relation of low and high frequency respectively with the wavelet.

Try:

dspxfrm3

(The material in this lab handout was put together by Paul Beliveau and derives principally from the MathWorks training document “Simulink for Signal Processing”, 2006.)

©2009GM