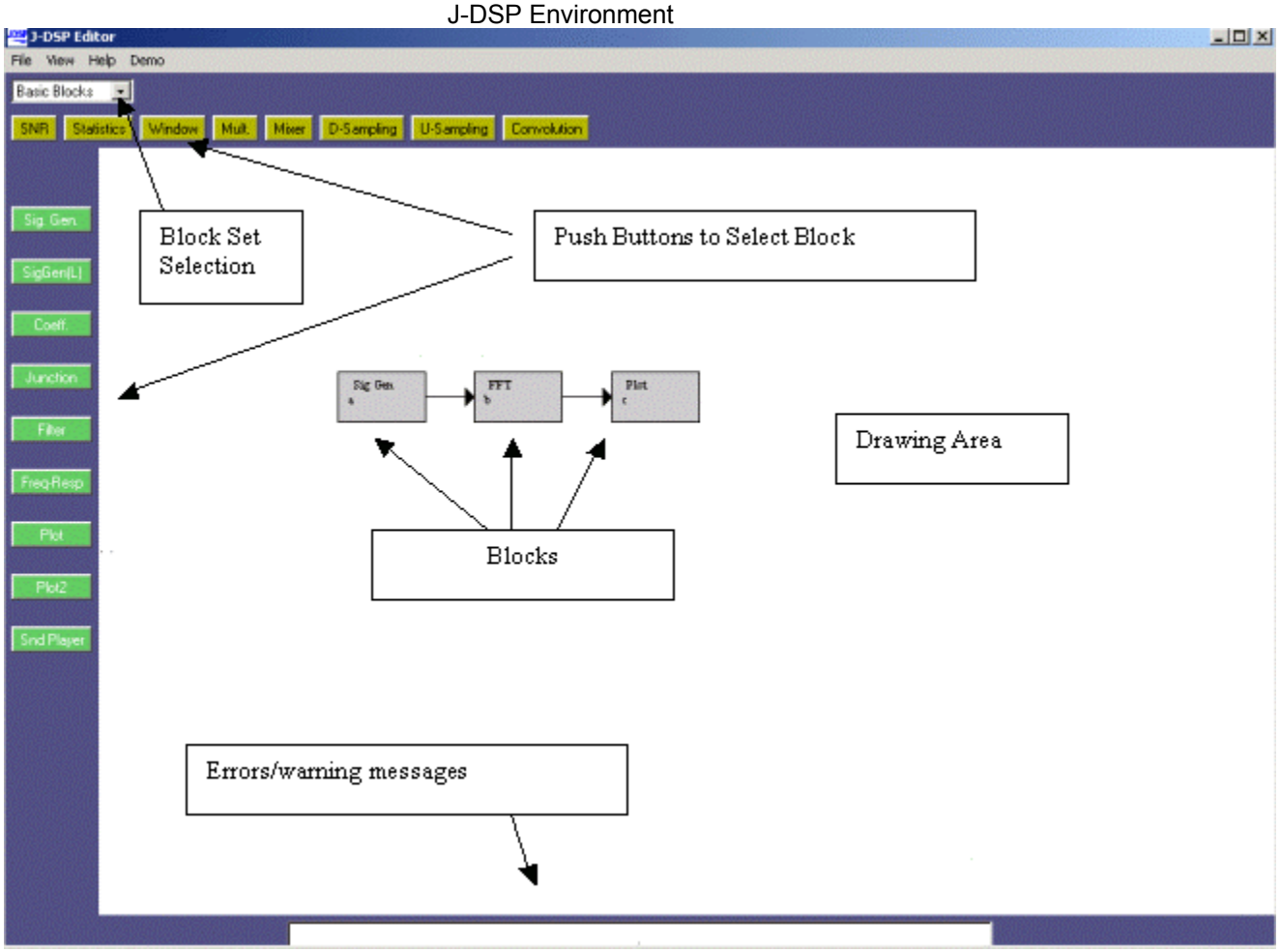
General Information on J-DSP

In this document, we provide a series of computer laboratory exercises for an internet-based signal-processing laboratory that facilitates hands-on learning experiences. The laboratory is based on an object-oriented Java™ tool called Java Digital Signal Processing editor (J-DSP). J-DSP has been developed at Arizona State University (ASU) and is written as a platform-independent Java applet that resides either on a server or on a local hard-drive. It is accessible through the use of a web browser. J-DSP has a rich suite of signal processing functions that facilitate interactive on-line simulations of modern statistical signal and spectral analysis algorithms, filter design tools, QMF banks, and state-of-the-art vocoders.

J-DSP provides a user-friendly environment through Java's graphical capabilities. Its highly intuitive graphical user interface (GUI) is easy to understand and use. All functions in J-DSP appear as graphical blocks that are divided into groups according to their functionality. Selecting and establishing individual blocks can be done by a drag-and-drop-process. Each block is linked to a signal processing function. The figure below shows the J-DSP editor environment. By connecting blocks together, a variety of DSP systems can be simulated. Signals at any point of a simulation can be examined through the appropriate blocks. Blocks can be edited through dialog windows, allowing the user to change the corresponding function's parameters to desired values and/or to view results. Blocks can easily be manipulated (i.e. edit, move, delete and connect) within the specified drawing area, using the mouse. System execution is dynamic, which means that any change at any point of a system will automatically take effect in all related blocks. Any number of block windows can be left open to enable viewing results at more than one point in the editor.

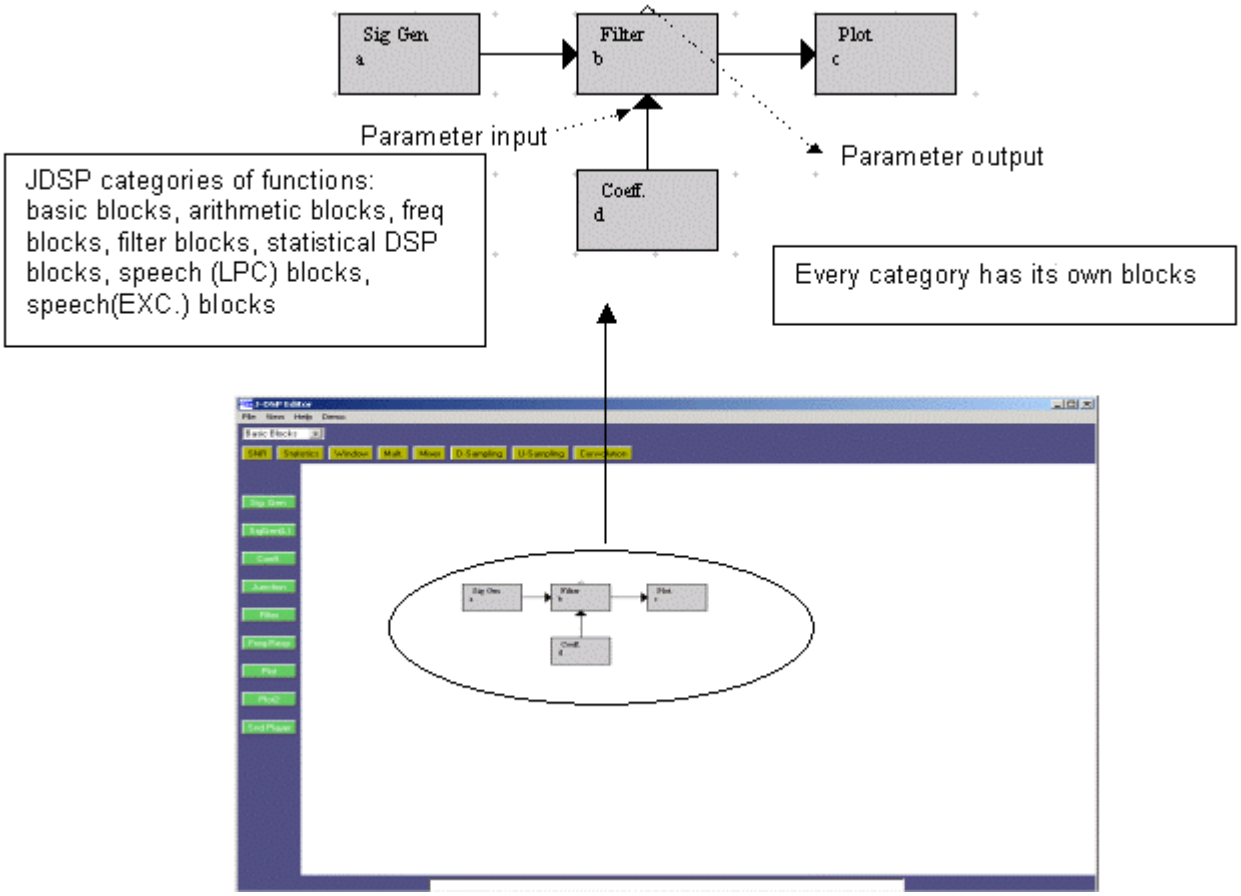
J-DSP Environment



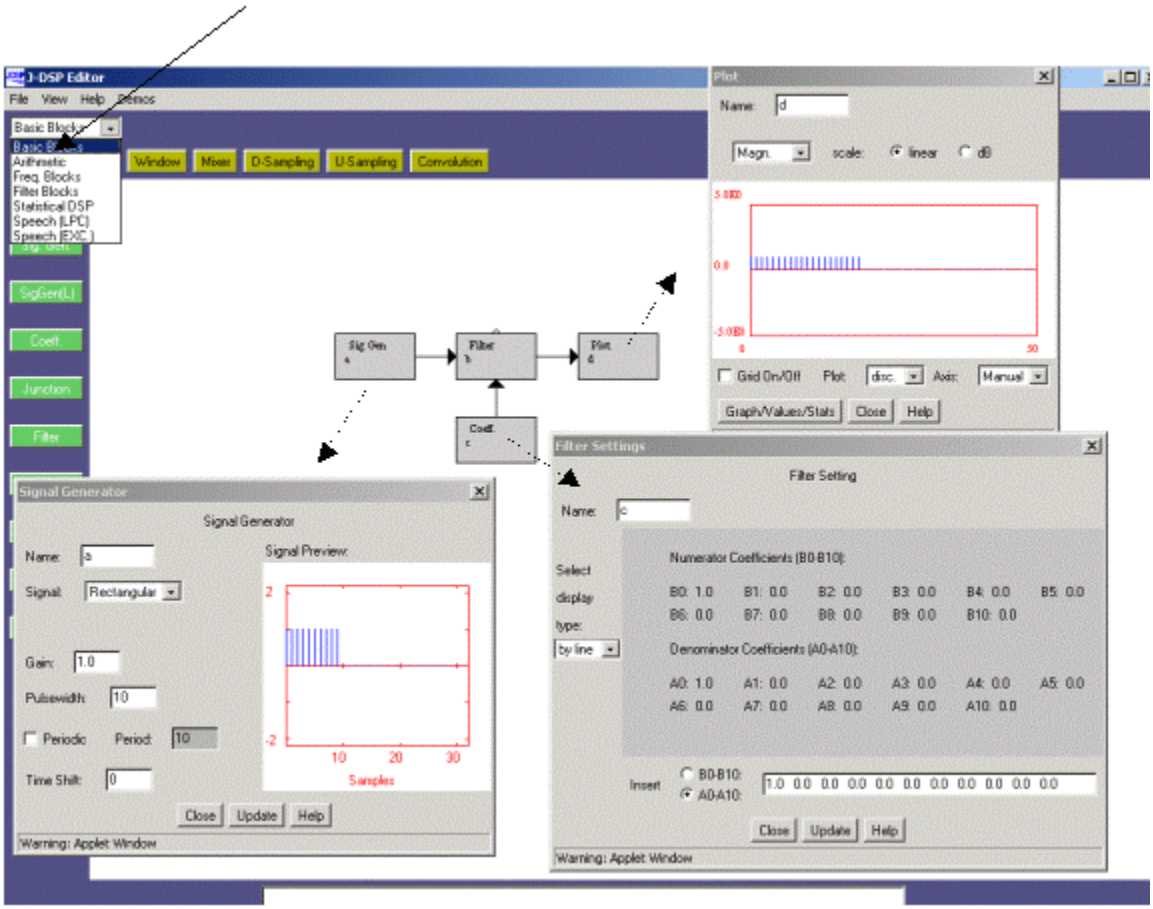
Lab 1: Working with J-DSP

The easiest way to explain some of the functions of J-DSP is to work through a simple example. To start J-DSP, go to the link, http://www.eas.asu.edu/~midle/jdsp/, and click on the "Start J-DSP" link, and press "start" in the subsequent page. It may take 30 seconds to download the program and a few more seconds to establish the first block but once the first block is established, the program should run quickly. Adjust the size of the J-DSP editor window so that you are still able to read this text or make a printout of this page. Press the ***Sig Gen*** button on the left part of the window. Move the mouse to the center of the window and click the left mouse button. You have created a signal generator block. There are two signal generators, ***Sig Gen*** for processing a single frame of the signal and ***Sig Gen (L)*** for frame-by-frame processing which is typically used in speech applications. Similarly, create a ***Filter*** and a ***Plot*** block and note that blocks cannot be placed on top of one another. There are two plot blocks, i.e., ***Plot*** (single plot) and ***Plot2*** (two plots). For now, use ***Plot***.

Note that each block has signal input(s) designated by the little triangle(s) on the left and signal output(s) to the right. Some blocks carry parameter inputs and outputs at the bottom and top of the block respectively. For example, the ***Filter*** block has a coefficient input on the bottom and a coefficient output on the top. Parameter inputs facilitate functions like filter design, frequency response, LPC etc.

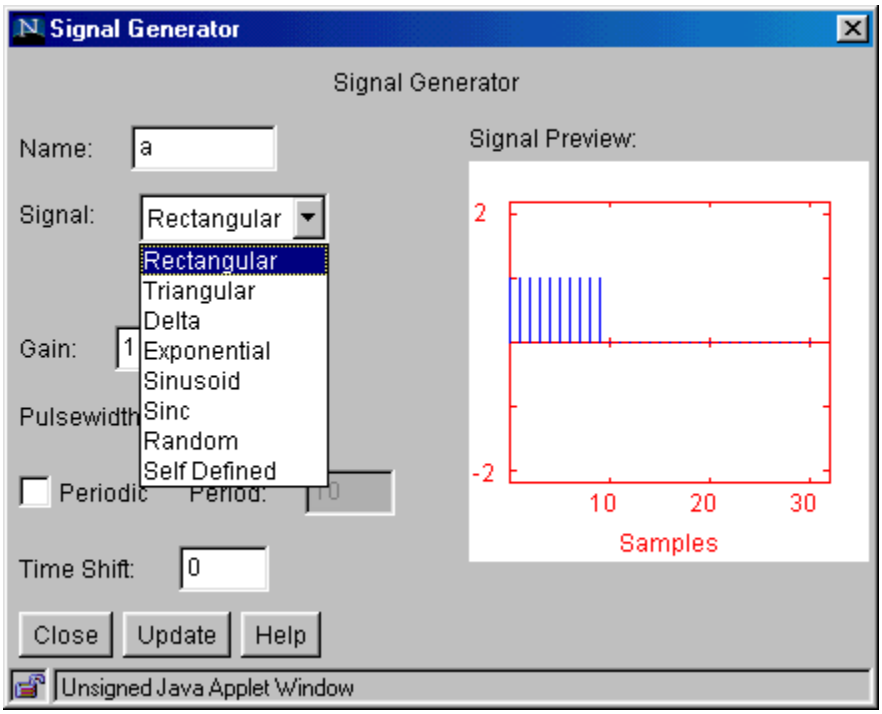


To select a block, click once to highlight it. You can then move it by placing the mouse arrow over it, holding down the left mouse button and dragging the box to a new location. To delete a block, simply select it and press the "delete" key on your keyboard. To link blocks, click once inside the small triangle on the right side of the signal generator box and while holding the mouse button down, drag the mouse arrow to the triangle on the left side of the filter box. Release the mouse button to create a connection between the two boxes. Always make the connections in the direction of the signal flow. The ***Coeff.*** block is used to specify filter coefficients. The block is connected to the filter's lower parameter input triangle. Now, connect the ***Filter*** block to the ***Plot*** block so that your editor window looks like the block diagram. Note that you can view the dialog box of each block by double clicking on the block, as shown in the figure below.



Choosing Signals

Let us now form a signal using the signal generator. Double click inside the ***Sig Gen*** box and a dialog window appears. If you do not see a dialog window, you are using an older Internet browser and must download the newest version of Netscape or Internet Explorer and start over. Use Internet Explorer 5.5 or later, or Navigator version 4.6 or later, with its Java plug in.



On the right side of the signal generator window, you see a preview of the signal. You may change the "name" of the signal, the "gain", the "pulse width", the "period" and the "time shift" by typing the desired value into the appropriate box. The signal type can be changed by clicking on the pop down menu and selecting a signal. If you select a "self-defined signal", an "edit signal" button will appear allowing you to edit the signal

With all signals except audio, J-DSP assumes a normalized sampling frequency of 1Hz. Hence the sampling frequency in terms of radians is 2π. All frequencies are entered as a function of π, e.g., 0.1π, 0.356π, etc. Any sinusoidal frequency at or above π will result in aliasing.

Step 1.1: Create a sinusoid with "frequency" 0.1π, "amplitude" 3.75, "pulse width" 40. When all of the parameters have been entered, press the [update] button to update the signal preview and remember that whenever changes are made to this box, the [update] button must be pressed in order for the changes to take effect. On the right, you can see a preview of the input signal. Count the number of samples within a period. How many do you have? (ans: 20 samples).

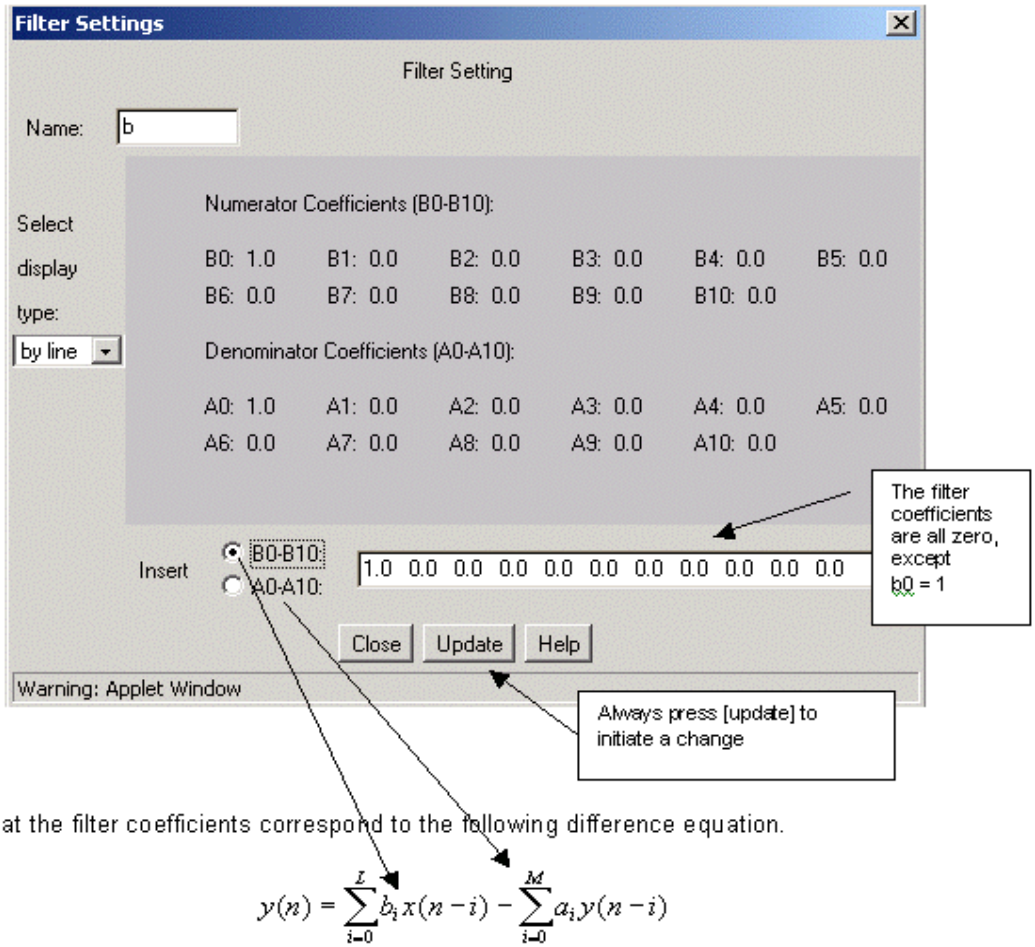
Step 1.2: Create a sinusoid with "frequency" π, "amplitude" 3.75, "pulse width" 40 (remember to press update for changes to take place). What happens? (ans: we have aliasing, i.e, no signal).

Step 1.3: Create a sinusoid with "frequency" 1.3π, "amplitude" 3.75, "pulse width" 40. What happens? Count the number of samples in a period. (ans: we have aliasing again , signal makes no sense).

Step 2: Next, we want to take a look at the ***Filter*** output in the time and frequency domain. Set the values in ***Sig Gen*** as per step 1.1. Double-click the ***Plot*** block and a new dialog window will

appear. You should again see the input signal because the filter is just letting the signal pass through unaffected, as no coefficients have been set. If you press the [Graphs/Values/Stats] button, a table with the values of the signal pops up. In the first column you see the indices of the samples and the second column shows you the values. Close the value dialog box.

Step 2.1 Let us now see the filter in action. Keep the ***Plot*** window open to observe any changes. Double click the ***Coeff.*** block. You should see the following:



Note that the filter coefficients correspond to the following difference equation.

$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=0}^{M} a_i y(n-i)$$

Step 2.2 Keep the values in ***Sig Gen*** as per step 1.1. Change the filter coefficient to b_0=4 and press [update]. Double click on the ***Plot*** block. You should see that the amplitude of the sinusoid has changed (ans: peak amplitude 4x3.75= 15).

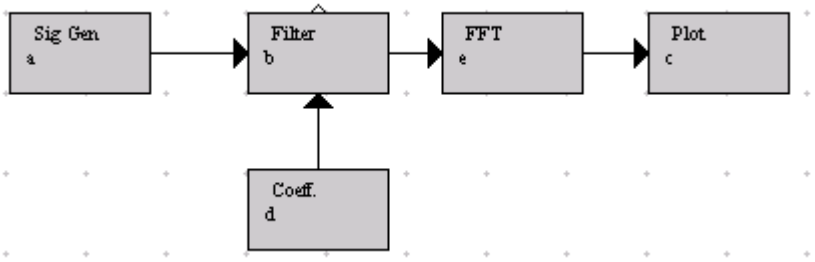
Step 2.3: Implement a pure delay by setting b_5=1 and rest of the coefficients (including b0) to zero and press [update]. What happens to the sinusoid?

Step 2.4 Implement a simple LPF, set b_0 = 0.2 and a_1 = -0.8 and press [update]. Generate a sinusoid with "gain" 1, "frequency" 0.1π, "pulse width" 256. What do you observe?
What kind of signal do you get at the output? Why? What is the peak-to-peak value? Do we have a change? Is there a phase shift? What filter function determines the time shift?

Step 2.5: Select the ***Freq-Resp*** block from the panel of general blocks on the left of the window and place it to the north of the ***Filter*** block. Connect the parameter output to the ***Freq-Resp*** block. Double click the ***Freq-Resp*** block. You should see the magnitude and phase response of

the filter. Change the coefficient to $a_1 = 0.8$ instead of $a_1 = -0.8$. What do you see in the frequency response and output? (ans: HPF, decrease in amplitude)

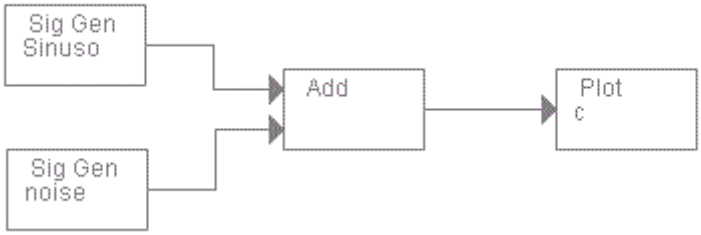
Step 3: To view the signal in the frequency domain, insert an ***FFT*** box between the ***Filter*** and the ***Plot*** box as shown below. The ***FFT*** box can be found under the **Freq. Blocks** menu.



Step 3.1 Set the ***Filter*** parameters and input as per step 2.4. Double click on the ***FFT*** block and change the "FFT size" to 256 points and then press [Close]. Now, you can see the magnitude and the phase of the signal in the frequency domain. The magnitude has a sharp peak approximately at 0.31, the frequency of our sinusoidal signal (0.1x3.1459).

Step 3.2 Change the sinusoidal frequencies as per steps 1.2 and 1.3 but with "pulse width" 256. What do you observe?

Step 3.3 Delete the filter. Set the sinusoidal "frequency" in ***Sig Gen*** as per step 1.1 but with "pulse width" 256. Now create a second ***Sig Gen*** block and a ***Mixer*** block, name it 'Add'. Your editor window should then look like the following:



Change the name of the second input box to ``'Noise' and the name of the output box to 'SigNoi'. The names are restricted to six characters. Following that, we edit the new input box called 'noise'. Open the dialog window and change the "signal type" to "random". Choose a "variance" of 4 and extend the "pulse width" to 256 samples, in order to have noise over the full length of the signal. Now take a look at the output signal. In the time domain it is very hard to see that a sinusoid is present. However, if you view the signal in the frequency domain with an FFT of size of 256, then you still find a peak at approximately 0.31.

Step 3.4: Change the "gain" of the sinusoid up or down and observe the spectra (FFT plot). Try different values to make the sinusoid to dominate or to be masked by noise.
Remember the "Hunt for the Red October"(a movie about a stealth sub-marine defecting from the Soviet navy)? Sonar operators were viewing FFT spectra listening to sonar as they were searching for submarine propeller signatures (sinusoids) in ocean noise (random signal). Stealth submarines would like their propeller signatures to be buried in (or masked by) ocean noise.