1.6. J-DSP LABORATORY ON DIGITAL FILTER BASICS

In this section, we introduce our first Java-DSP laboratory exercise. The purpose of this exercise is to familiarize the students with J-DSP and demonstrate how this web-based simulation environment can be used to obtain hands-on experiences with discrete-time signals and digital filters. It is important for the students to read this section carefully and go through the initial steps so as to become familiar with the software. A condensed manual in the appendix of the book provides additional information and specifics on the J-DSP functions

1.6.1. GENERAL INFORMATION ON J-DSP

This book provides a series of computer laboratory exercises that provide hands-on learning experiences on several DSP topics. The laboratories are based on an object-oriented Java™ tool called Java Digital Signal Processing (J-DSP). J-DSP has been developed at Arizona State University (ASU) and is written as a platform-independent Java applet that can be accessed at http://jdsp.asu.edu with the use of a Java-enabled web browser. J-DSP has a rich suite of signal processing functions that include signal generators, digital filters, pole-zero root computation, various types of filter design algorithms an FFT, a plot function, a sound player, multi-rate converters, statistical signal analysis, spectral estimation, and many others. J-DSP provides a user-friendly environment that embodies Java's graphical capabilities. Its highly intuitive graphical user interface (GUI) is easy to learn. All functions 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 software that performs a specific signal processing or signal plot or play function. Block parameters can be edited through dialog windows and simulation results can be plotted. The figure below shows the J-DSP editor environment. By connecting blocks, signal flow is established and an algorithm can be simulated. Signals at any point of a simulation can be visualized through the appropriate blocks. Blocks can be manipulated (i.e. edit, move, delete and connect) using the mouse. System execution is dynamic, i.e., any change at any point of a system will

automatically take effect in all subsequent blocks. Windows can be left open to enable the user to view signals at different points.

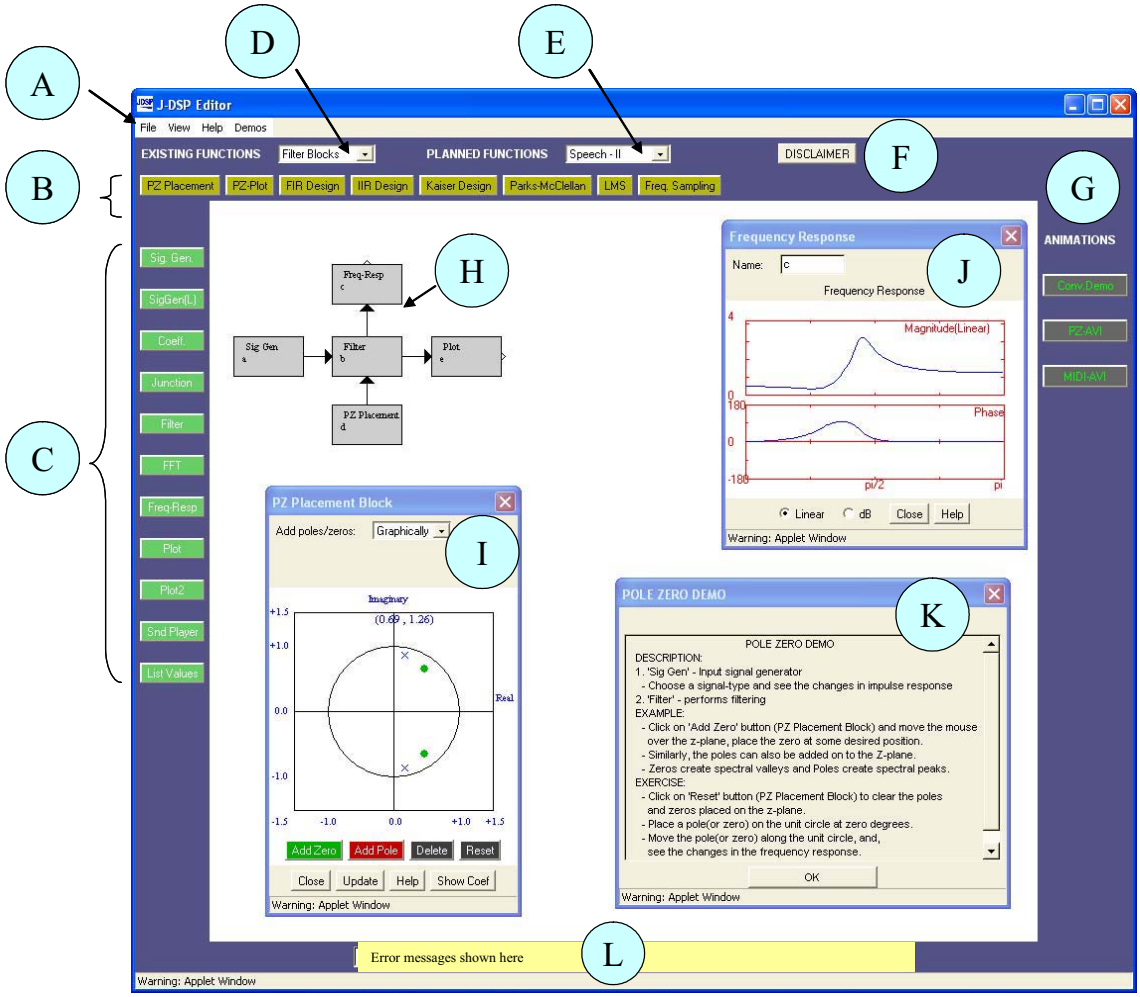


Figure 1.11. J-DSP simulation environment. In the figure, ‘A’: Menu items; ‘B’: Filter blocks (this section changes according to the selection of ‘D’ or ‘E’); ‘C’: Permanent blocks; ‘D’and ‘E’: List menu to select the group of functions; ‘F’: Disclaimer; ‘G’: Interactive visual demonstrations; ‘H’: Simulation flowgram; ‘I’: Dialog window (corresponding to the PZ Placement block in the block diagram ‘H’); ‘J’: Plot window to view the results; ‘K’: Help window provided for all the blocks; ‘L’: A field that shows error messages (if any).

Please note that the following notation have been used through out this document:

- Block names: bold and italic e.g. ***Plot***
- Drop down menu item name: big font and bold e.g. **Basic Blocks**
- Button: third brackets e.g. [update]
- Option to be chosen by user in a dialog box of a block: inverted comma e.g. “Gain”

1.6.2. GETTING STARTED – GENERATING AND ANALYZING SIGNALS

The easiest way to explain some of the functions of J-DSP is to work through a simple example. To start J-DSP, type the URL, http://jdsp.asu.edu/, and press skip animation to enter the J-DSP web page. With the mouse navigate to the upper left frame where it says "Start Center" and press "Start." Read the quickly information in the panel and press "proceed." The Java applet will launch when you press again the "Start" button. Read the disclaimer and if you agree press "I accept." The J-DSP environment appears and is ready for programming a DSP simulation. Please note that 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.

To program a simulation that involves a signal, a filter, and a plotter proceed with the following steps. Press the ***Sig Gen*** button on the left, move the mouse to the center, and click the left mouse button. Note that the signal generator block is established. Similarly, create a ***Filter*** and a ***Plot*** block. Note that each block has signal input(s) designated by the small 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 are used to interface and run functions such as filter design, frequency response, pole-zero plot, 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*** block parameter input as shown below. Now, connect the ***Filter*** block to the ***Plot*** block so that your editor window looks like the block

diagram shown below (Figure 1.12). Note that you can view the dialog box of each block by double clicking on the block (see Figure 1.13).

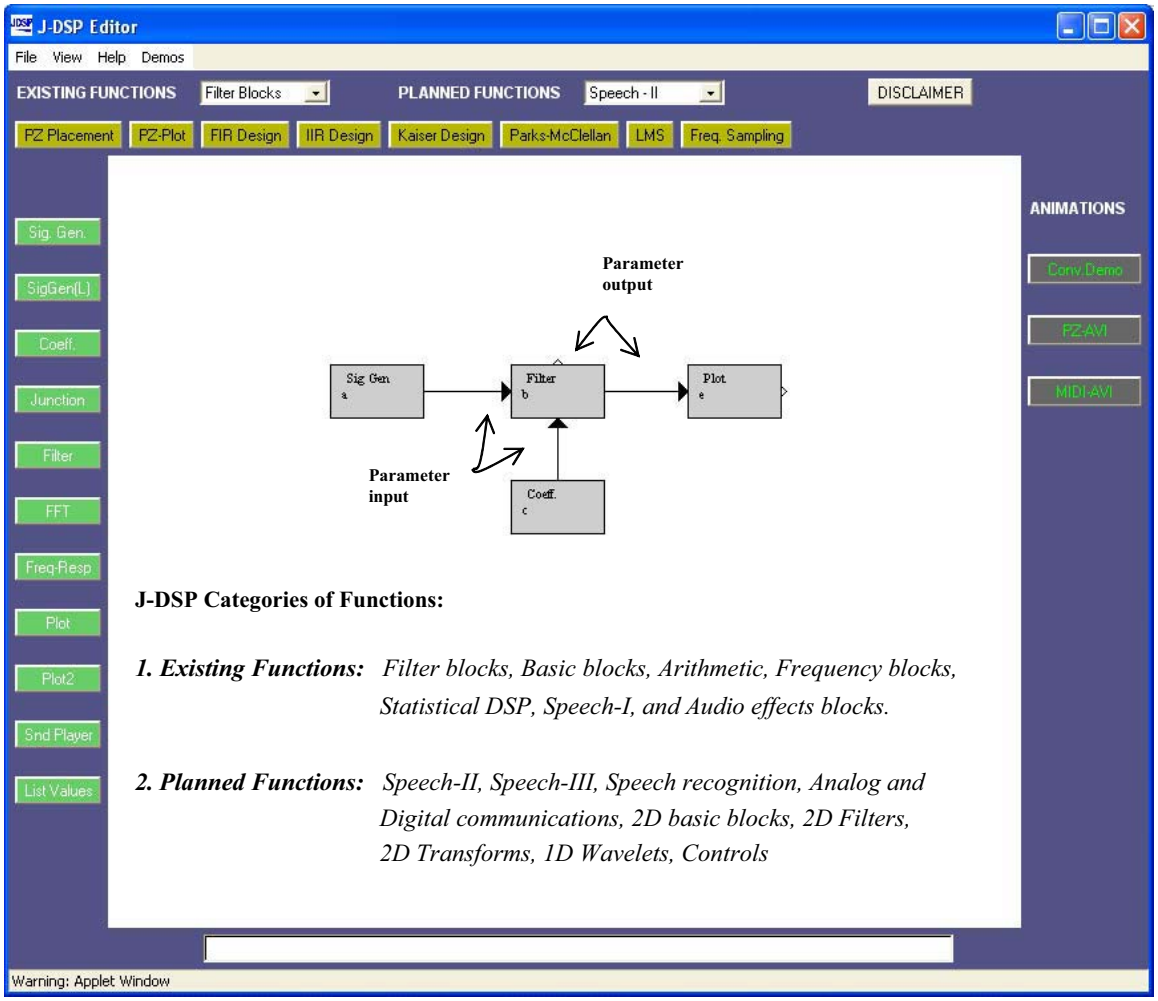


Figure 1.12. Lab-1 working with J-DSP

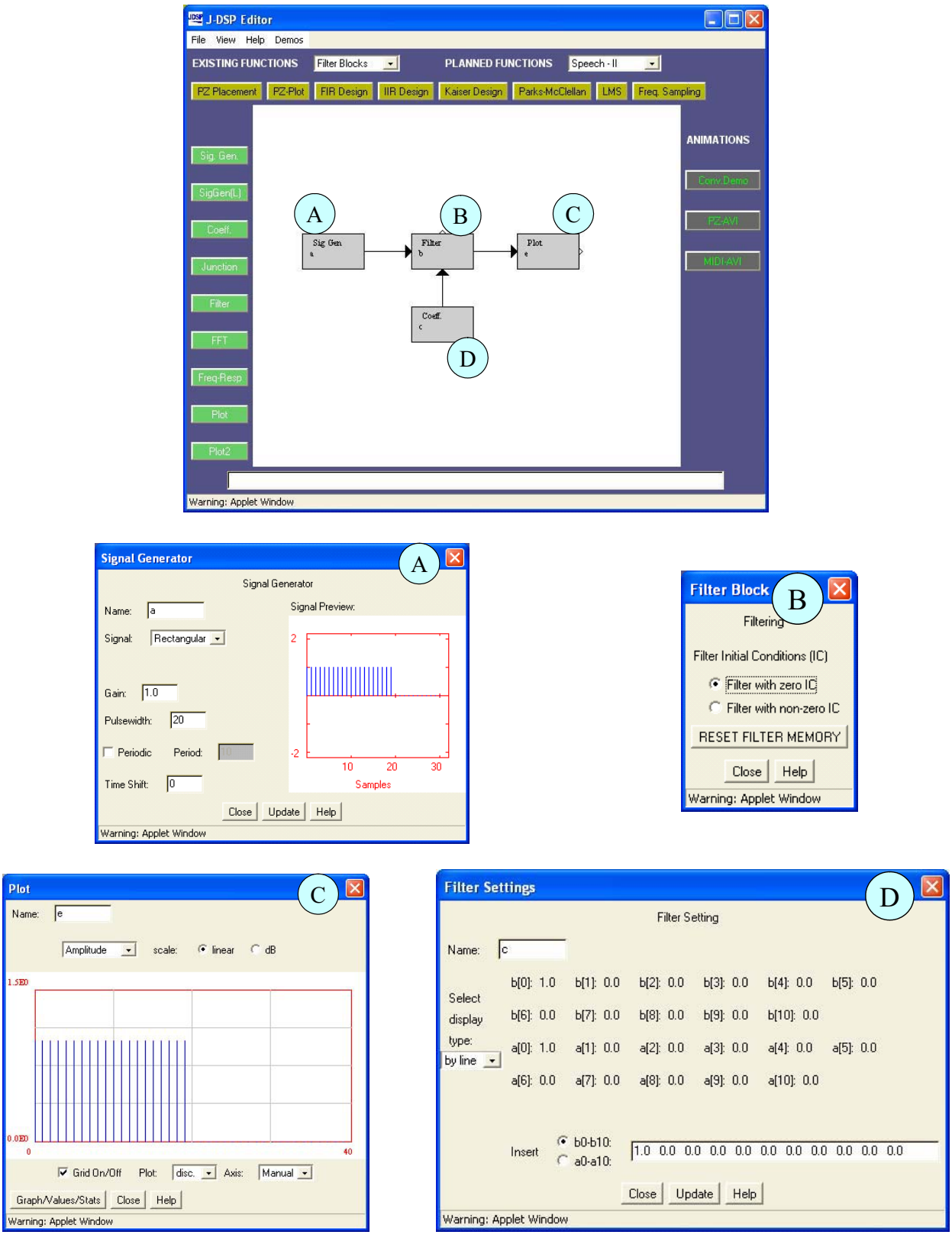


Figure 1.13. Dialog windows (**'A'** through **'D'**) corresponding to the blocks in the flowgram

1.6.3. CHOOSING SIGNALS

Let us now form a signal using the signal generator. Double click inside the ***Sig Gen*** box and a dialog window will appear. 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.

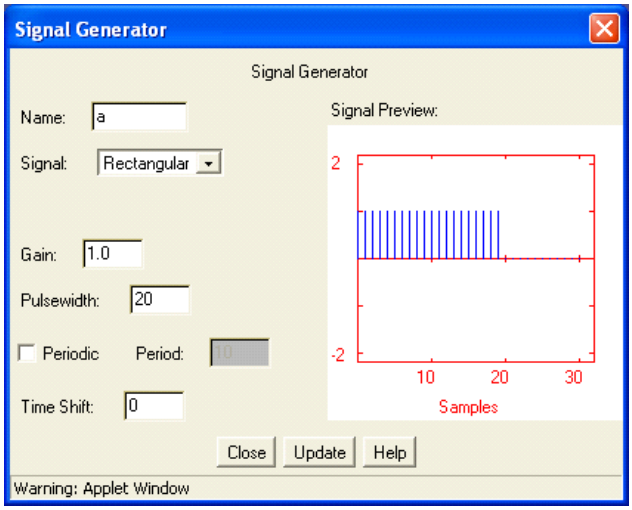


Figure 1.14. The signal generator ('***Sig. Gen.***') dialog window in J-DSP

On the right side of the signal generator window, you can 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 drop-down menu and selecting a signal. If you select a User-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. Note that 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. 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 preview the signal. Count the number of samples within a period. How many do you have within a period? (ans: 20 samples).

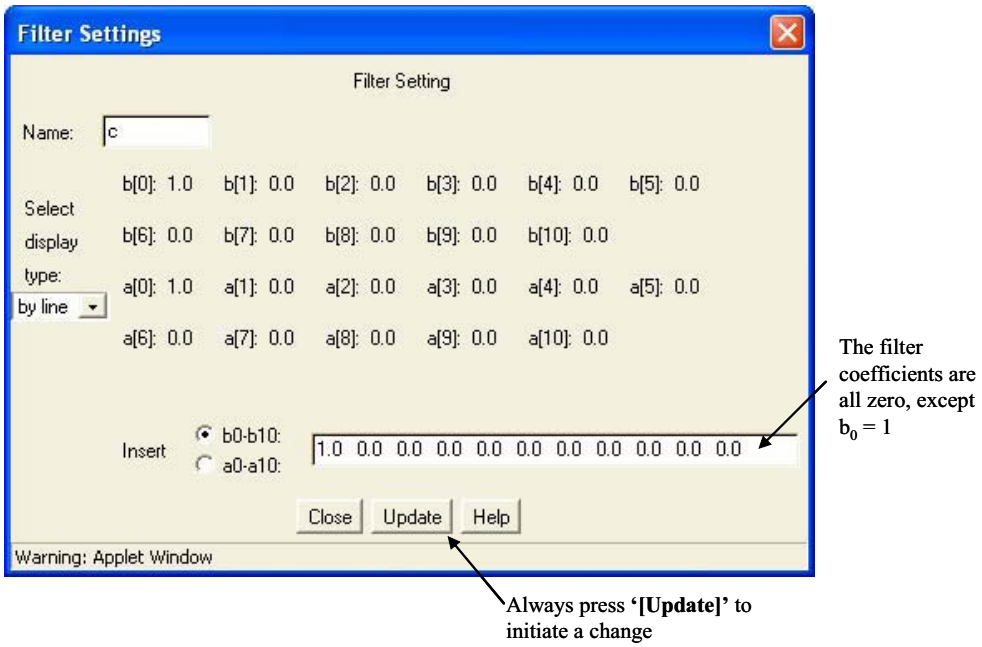
Step 1.2: Create a sinusoid with "frequency" π, "amplitude" 3.75, "pulse width" 40 (remember to press update for changes to take place). What happens? (ans: no signal because we have aliasing, i.e, the signal has been sampled precisely at the zero crossings).

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 and the signal makes no sense – sinusoidal frequencies must be chosen smaller than π ro avoid aliasing).

1.6.4. ESTABLISHING SIMULATIONS

Next, we want establish a digital filter simulation. Establish the ***Coeff*** block and connect it to the bottom parameter input of the ***Filter*** block. Double click on the ***Coeff*** block and note that it has a default coefficient of *bo=1* ; the rest of the coefficients are zero. Hence the default gain of the filter is unity. Close the ***Coeff*** block. Open again ***Sig Gen*** and set the signal values as per step 1.1. Double-click the ***Plot*** block and a new dialog window will appear. You should again see the same input signal as in step 1.1 because the filter is letting the signal pass through unaffected, since only coefficient *bo=1* and the rest are zero. Note that the ***Plot*** block displays the signal by default as a continuous curve [cont.]; to view the discrete samples select [disc.] from the drop down menu in the center of the bottom panel. If you press the [Graphs/Values/Stats] button on the left, a table with the values of the signal appears. In the first column you see the indices of the samples and the second column shows the values. Close the value dialog box and continue.

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 ('b_0-b_{10}' and 'a_0-a_{10}') correspond to the following difference equation,

$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i)$$

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]. Observe 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 the rest of the coefficients (including b0) to zero and press [update]. What happens to the sinusoid? (ans: sinusoid is delayed)

Step 2.4 Implement a simple low pass filter (LPF), set $b_0 = 0.2$ and $a_1 = -0.8$ (set rest to zero) and press [update]. Use ***Sig Gen*** and generate a sinusoid with "gain" 1, "frequency" 0.1π, "pulse width" 256. What do you observe? (answ: amplitude change and phase shift)

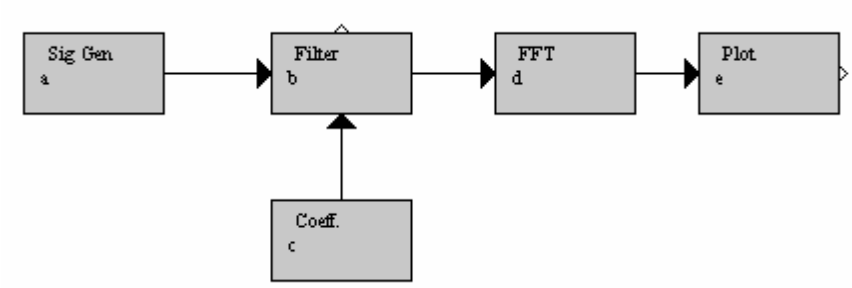
Step 2.5: Select the ***Freq-Resp*** block from the panel of general blocks on the left of the window and place it to the north of the ***Filter*** block. Connect the parameter output to the ***Freq-Resp*** block. Double click the ***Freq-Resp*** block. You should see the magnitude and phase response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

1.6.5. VISUALIZING RESULTS GRAPHICALLY

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 on the left panel

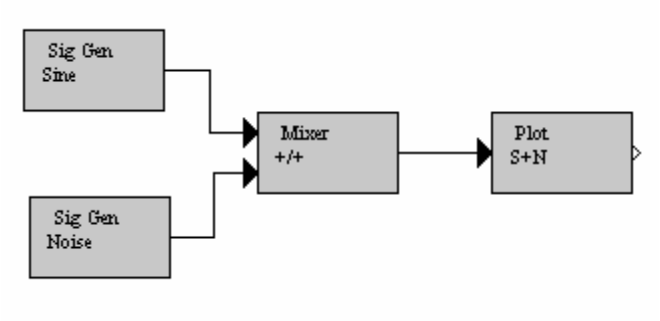
.



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, i.e., 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 and observe the changes in the plot.

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 ***Adder*** (mixer) block (look under Basic blocks). Your editor window should then look like the following:



Change the name of the first ***Sig Gen*** block to 'Sinusoi', the second ***Sig Gen*** block to 'noise' and the ***Plot*** block to 'SigNoi'. The names are restricted to six characters. Following that, we edit the ***Sig Gen*** block 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 mask or to be masked by noise.