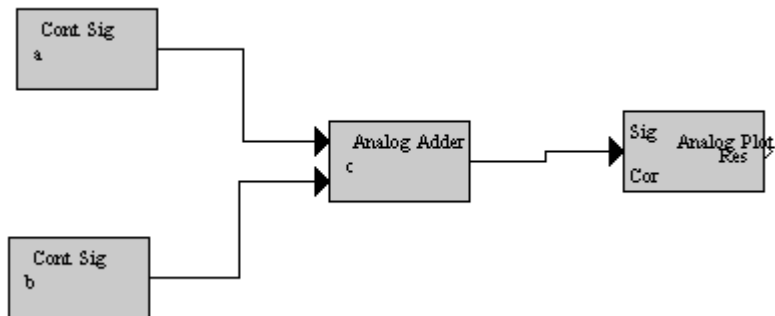
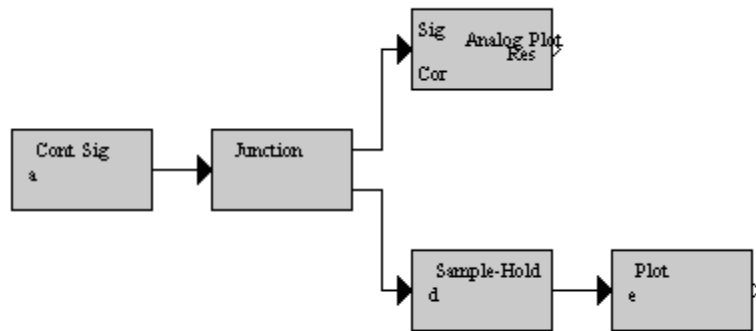


Updates to the J-DSP Environment

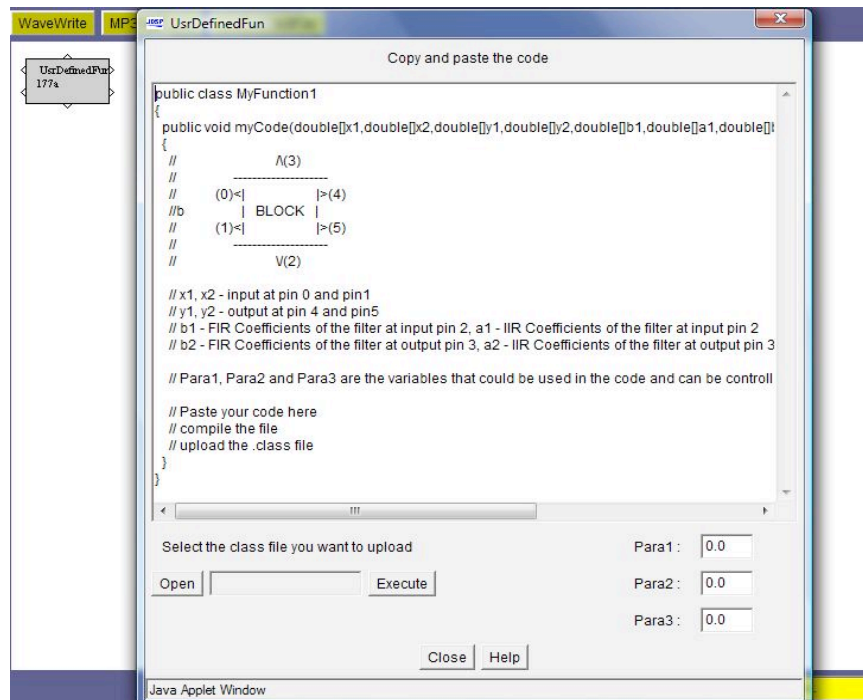
Functions

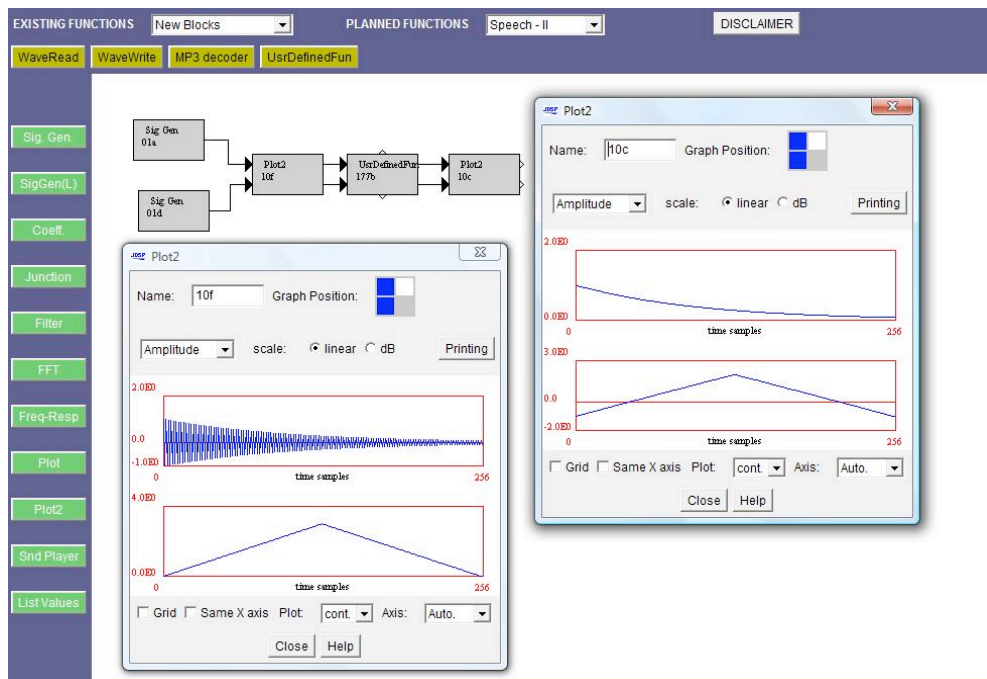
- **Analog Blocks:**
 - **Analog Signal Generator:** Generates analog sine/triangle/square/exponential signals with the desired amplitude, frequency and phase.
 - **Analog Plot:** Plots the analog signal generated. Note that this plot function will not work with any discrete signals in J-DSP. Similarly, the normal Plot function cannot be used with analog signals.
 - **Analog Adder:** Adds two different analog signals and generates an output analog signal.
 - **Sample-Hold:** Performs sample-and-hold operation on an analog signal using the desired sampling frequency and the output is a discrete signal. Note that the normal Plot function should be used for visualizing the discrete signal.



- **User Defined Block**

- In order to allow users to add their own customized code as a J-DSP block without making J-DSP an open source, the user-defined block has been developed. This block is similar to other J-DSP blocks, except that it has an interface to facilitate users to add their Java class files during the runtime. Fig. 1 shows the dialog box of this block. As shown, this block provides the users with the input and output pins to which the input and output signal arrays can be associated. The block takes in the input signal array, implements the user-defined function and provides the output. Users can also include upto three parameter variables namely, *Para1*, *Para2* and *Para3* in their code and the values of these parameters could be controlled dynamically from the dialog box.
- The text area in the dialog box provides the prototype of the Java class MyFunction1, in which users can insert their code. The Open button on the dialog box helps users in locating the compiled Java file or class file on their system. The Execute button loads the file in the J-DSP simulation dynamically. As soon as the file is loaded, it is executed and the output at the output pins change accordingly. Fig. 2 shows an example functionality of the user-defined block. The Java code for the block is given in Fig. 3.





```

public class MyFunction1
{
    public void myCode(double[] x1, double[] x2, double[] y1, double[] y2
    {
        /*
            ^ (3)
            (0) < [ BLOCK ] > (4)
            (1) < [ BLOCK ] > (5)
            v (2)
        */
        // x1 - input at pin 0, exponential signal
        // check the top portion of the plot on the Left side

        // y1 - output at pin 4, absolute value of input signal
        // check the top portion of the plot on Right side

        for(int i=0; i<256; i++)
        {
            y1[i] = Math.abs(x1[i]);
        }

        // x2 - input at pin 1, triangular signal
        // check the bottom portion of the plot on the Left side

        // y2 - output at pin 5, input-Paral (Paral = 1)
        // check the bottom portion of the plot on Right side

        for(int i=0; i<256; i++)
        {
            y2[i] = x2[i] - paral;
        }
    }
}

```

General Features

- **Scroll Bar:**

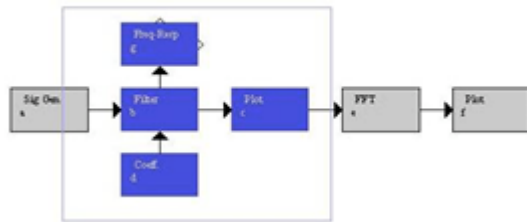
- Individual scrolling functionality for Drawing Area (Simulation Area), Permanent Blocks and Existing Functions
- Helps in changing the size of the window without cropping any content.

- **Import Script Resolution:**

- Full resolution while importing scripts. Blocks don't get displaced anymore, even by a pixel.

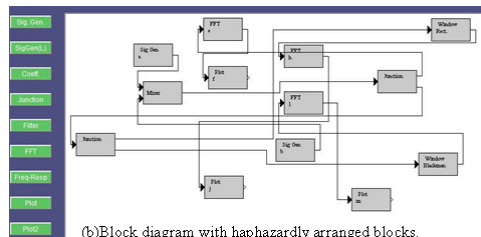
- **Multiple Block Select:** Allows user to select more than one block and connection at a time. This allows user to

- Move
- Delete, multiple blocks simultaneously.

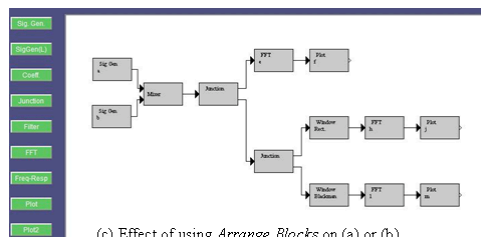


- **Arrange and Clean:**

- Automatically arranges the simulation block diagram, in a neat way and cleans up the orphan blocks (**Under the Edit Menu**).



(b)Block diagram with haphazardly arranged blocks.



(c) Effect of using *Arrange Blocks* on (a) or (b).