Filter Signals Using MATLAB

Apply MATLAB Based Filters In The DSO's Processing Path

There is a common need to filter signals prior to analysis. Whether the need is to equalize frequency response or eliminate noise prior to further processing it is very useful to be able to apply a user selected filter to the data. LeCroy's WaveMaster™ series oscilloscopes allow users to embed any of MATLAB's library of filter types right into the oscilloscope's processing path. In figure 1 we show an example of a 2-pole Butterworth filter which has been applied to the acquired waveform using the MATLAB math function.

Figure 1 The response of a MATLAB based 2 pole Butterworth filter (lower trace) to a swept sine input (upper trace).

The MATLAB math function allows the user to call the MATLAB program and execute a MATLAB script file right in the scopes processing path. The output from MATLAB is returned to the next processing stage and operations continue within the scope. Figure 1 shows the basic setup of the MATLAB math function. The function accepts one or two input signals and returns a single output. Selecting the MATLAB tab of the math dialog box allows the user to load an existing .m file or create a new one in the built-in editor, as shown in figure 2.

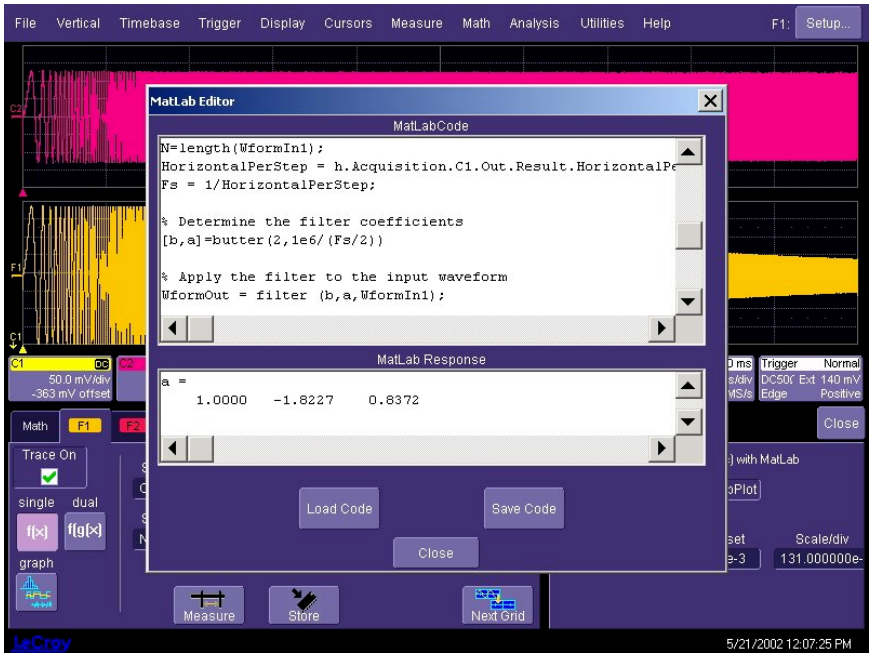

Figure 2 A view of the editing window in the WaveMaster MATLAB math function showing part of the MATLAB .m file being executed

The .m file used in this example is shown in figure 3. The filter type used is a Butterworth lowpass filter. MATLAB offers a choice of some 7 filter types. This filter is a relatively slow cutoff second

order filter. The command to create the filter coefficients is:

[b,a]=butter(2,1e6/(Fs/2))

Where b represents the numerator coefficients of the digital filter and a represents the denominator coefficients of the digital filter. The arguments for the Butterworth filter are order (2 in this case) and the cutoff frequency (this must be normalized to Nyquist which is why we have divided by 1/2 the sampling frequency, Fs).

The filter is implemented using the filter command:

WformOut = filter (b,a,WformIn1);

This applies the filter coefficients to the selected data, in this case the input waveform (WformIn1)

The following command queries the scope via Microsoft automation to obtain the sampling frequency.

% Interfacing the scope via automation

h=actxserver('Lecroy.wavemasterapplication');

% Get Sample Frequency and Data length
N=length(WformIn1);
HorizontalPerStep = h.Acquisition.C1.Out.Result.HorizontalPerStep;
Fs = 1/HorizontalPerStep;

In this example we implemented a simple low pass filter using MATLAB. You can extend this to use any of the available MATLAB functions or scripts.

```
%  This MATLAB script will implement a second order Butterworth low pass filter with a cutoff frequency of 1 MHz
%  and apply it to the input waveform,WformIn1
%
%  Because the data coming in does not have information
%  about the time between points, we get this information by
%  connecting to the WaveMaster application and get the time
%  between the points for Channel 1. Therefore, if you are not
%  sending data with the same time between points as channel
%1, the frequency scale in the plot will be wrong.
%
% Interfacing the scope via  automation

h=actxserver('Lecroy.wavemasterapplication');

% Get Sample Frequency and Data length
N=length(WformIn1);
HorizontalPerStep =
h.Acquisition.C1.Out.Result.HorizontalPerStep;
Fs = 1/HorizontalPerStep;

% Determine the filter coefficients
[b,a]=butter(2,1e6/(Fs/2))

% Apply the filter to the input waveform
WformOut = filter (b,a,WformIn1);
```

Figure 3 The MATLAB .m file that implements a 1 MHz , 2 pole , Butterworth low pass filter applied to the input trace from channel 2.

WaveMaster also supports embedded math operations based on VBScripts, Mathcad, or Excel.