Graphical Design Of Frequency Sampling Filters For Use In A Signals And Systems Laboratory

Andreas Spanias¹, Constantinos Panayiotou² and Venkatraman Atti³

Abstract - In this paper, we present educational filter design tools used in the signals and systems course. A graphical-user-interface (GUI) for filter design using the frequency-sampling method has been developed and embedded as a module in the Java-DSP (J-DSP) editor. Three realizations are presented: the non-recursive, non-recursive using least-squares, and recursive. The paper describes several laboratory exercises that involve interactive filter design using the ‘Freq. Samp’ block. The performance of the frequency sampling method is compared against the performance of the Kaiser and Parks-McClellan algorithms. The exercises have been used at Arizona State University in the signals and systems course. Pre- and post-assessment quizzes have been assigned and results have been compiled to evaluate the learning attributed specifically to the J-DSP software and exercises.

Index Terms - Filter design, Frequency-sampling, FIR and IIR filters, and QMF bank simulations.

INTRODUCTION

At Arizona State University (ASU), as part of an effort to introduce undergraduates in digital signal processing (DSP) to application-oriented content, we are developing a series of on-line modules that include Java software, animated demonstrations, and computer laboratory exercises. In this paper, we describe the web-based educational software and the associated on-line labs designed to expose students in a signals and systems course to filter design techniques. A great deal of interest in DSP within the student community motivated us to develop various filter-design modules and on-line demos. These modules and demos have been incorporated in the ASU’s Java-DSP (J-DSP) software tool. The J-DSP editor is an on-line object-oriented simulation tool that enables students to simulate DSP algorithms over the Internet. Some of the filter-design modules in J-DSP include: the finite impulse response (FIR) filter design based on – the Parks-McClellan (Mini-Max) algorithm, frequency-sampling method (both recursive and least-squares realization), windowing method, and the Kaiser-design method; infinite impulse response (IIR) filter design based on IIR analog filter approximations.

This paper elaborates on the frequency-sampling (‘Freq. Samp’) module and describes laboratory exercises that involve filter design using the ‘Freq. Samp’ block. In this lab, students compare the performance of frequency sampling against other design methods such as the Kaiser and Parks McClellan algorithms. The ‘Freq. Samp’ block enables students to draw interactively the desired frequency response (for low-pass, high-pass, band-pass filters, or filters with arbitrary frequency response). Furthermore, this method allows the participant to analyze recursive and least-squares realizations of digital filters in terms of pole-zero locations, impulse response, sensitivity to quantization, etc. The students can also analyze the resulting frequency response for each realization using the relevant J-DSP functions.

FIGURE 1
AN EXAMPLE FILTER DESIGN SIMULATION IN J-DSP

This paper is organized as follows. First, we present the various filter design modules implemented in J-DSP. Next, a detailed description on the filter design using the frequency sampling method and the GUI associated with the ‘Freq. Samp’ block in J-DSP are given. This is followed by a detailed analysis of the three filter design realizations (i.e., non-recursive, least-squares, and recursive FIR filter design) using frequency sampling method. Next, we present a quadrature mirror filter (QMF) bank simulation in J-DSP. Finally, assessment results obtained from an undergraduate course at ASU are summarized.

¹ Andreas Spanias, Electrical Engineering Department, MIDL lab, Arizona State University, Tempe, AZ., spanias@asu.edu
² Constantinos Panayiotou, Electrical Engineering Department, MIDL lab, Arizona State University, Tempe, AZ., costasp@asu.edu
³ Venkatraman Atti, Electrical Engineering Department, MIDL lab, Arizona State University, Tempe, AZ., atti@asu.edu

© 2004 IEEE

October 20 – 23, 2004, Savannah, GA
34th ASEE/IEEE Frontiers in Education Conference
T1D-7
FILTER DESIGN MODULES IN J-DSP

J-DSP’s role, significance, and contribution to the undergraduate DSP education have been presented, in some detail, earlier in Error! Reference source not found.-Error! Reference source not found. J-DSP is an educational tool that was developed to enable on-line simulations and web-based computer laboratories for DSP-related courses in Electrical Engineering. The J-DSP version-1 (CD-ROM ISBN 0-9724984-0-0) is approximately 42,000 lines of Java code and is accompanied by a series of J-DSP laboratory exercises and a manual posted on the supporting web site. The J-DSP editor, Figure 1, includes a suite of built-in signal processing functions ranging from simple signal manipulators to complex filter design functions as well as speech processing algorithms. We note that the NSF-funded J-DSP was built from the ground up to deliver web-based laboratory experiences to undergraduate students and therefore is different from platform-specific commercial simulation packages such as MATLAB/SIMULINK.

Detailed descriptions on block manipulation and J-DSP infrastructure have appeared in Error! Reference source not found.-Error! Reference source not found.. In the next few sub-sections, we present an overview of the various filter design modules in J-DSP. J-DSP supports FIR filter design based on: the Fourier series, the Kaiser window, the Min-Max algorithm, and frequency-sampling method. IIR filter design experiments based on IIR analog filter approximations (Butterworth, Chebyshev, and elliptic filter realizations) can also be performed in J-DSP.

The Window Design Method

The window FIR filter design method is implemented in J-DSP by expanding the frequency response of an ideal filter in a Fourier series and then truncating and smoothing the response using a window. The user needs to enter the following information: Window type: Hamming, Hanning, Blackman, Bartlett, rectangular, or Kaiser; Filter order (maximum is 64); Type: low-pass, high-pass, pass-band, or stop-band. Cut-off frequencies ($f_c$), take values from 0 to 1, where $f_c = 1$ corresponds to half the sampling frequency.

The Kaiser Design Method

The Kaiser design process involves calculating the Fourier series of the ideal filter and then multiplying it with a Kaiser window that best fits the filter specifications. Filter specifications are: Filter type: low-pass, high-pass, stop-band or pass-band; $W_{p1}$, $W_{s1}$ – pass-band and stop-band edge cut-off frequencies, respectively; $W_{p2}$, $W_{s2}$ – second pass-band and stop-band edge cut-off frequencies, respectively (for pass-band filters); $PB$, $SB$ – pass-band and stop-band tolerances in dB.

The Min. Max. Method

FIR filter design experiments based on the Parks-McClellan algorithm (Figure 2) form an important laboratory exercise that exposes students to the concepts of optimal filter design and linear-phase filters.

![Figure 2: The Min. Max. Filter Design Module in J-DSP](image)

Filter Design by Analog Filter Approximations

IIR filter design in J-DSP is based on the bilinear transformation. Butterworth, Chebyshev type I and II, and elliptic filters can be designed. First a continuous-time, low-pass prototype meeting the desired filter specifications is generated. This prototype is then converted to the specified filter type (pass-band, high-pass, or low-pass) through a frequency transformation. The bilinear transformation is then applied, to transform the continuous-time filter response to discrete-time. From the discrete-time filter transfer function, the coefficients are calculated that describe and best fit the specified filter parameters.

FILTER DESIGN USING FREQUENCY-SAMPLING METHOD IN J-DSP

The idea of frequency sampling filters is that a desired frequency response can be approximated by sampling the continuous frequency response at N uniformly spaced points and then obtaining an interpolated frequency response that passes through the frequency samples, as shown in Figure 3. The resulting filter will have a frequency response that is exactly the same as the original response at the sampling instants. Moreover, the smoother the frequency response being approximated the smaller the error of interpolation between the sample points. However, between the frequency samples, the response may be significantly different (Figure 3(c)). To obtain a good approximation to the desired frequency response we must take a sufficient number of frequency samples. Figure 4 shows a typical specification for a low-pass filter with three transition band frequency samples.
The Frequency-sampling (Figure 5) functionality in J-DSP enables digital filter design with three possible realizations, namely, non-recursive (non-R), recursive (R) and non-recursive using the least-squares method (non-R/LS). The J-DSP’s ‘Freq. Samp.’ block can be used to design linear phase finite impulse response (FIR) filters based on the frequency-sampling method described above. The desired frequency response is drawn using the dialog window shown in Figure 5. The user specifies the number of lines and the number of samples used in the frequency-sampling method and draws the desired frequency response on a grid. Consecutive placement of points on the grid creates auxiliary lines automatically to assist the user to visualize the resulting frequency response. The frequency sampling method allows us to design non-recursive FIR filters such as low-pass, high-pass, band-pass filters and filters with arbitrary frequency response. Furthermore, this method allows recursive and least-squares realizations of FIR filters, leading to computationally efficient filters. Next, we address these realizations in detail.

Non-recursive FIR filters

An FIR filter can be uniquely specified by giving either the impulse response coefficients \( \{h(n)\} \) or, equivalently, the DFT coefficients \( \{H(k)\} \). These sequences are related by the DFT relations

\[
H(k) = \sum_{n=0}^{N} h(n) e^{-j2\pi nk/N} \quad (1)
\]

\[
h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) e^{j2\pi nk/N} \quad (2)
\]

The impulse response, \( h(n) \), of a linear phase FIR filter with positive symmetry, for \( N \) even, can be expressed as follows,

\[
h(n) = \frac{1}{N} \sum_{k=0}^{N} 2[H(k)\cos(2\pi n - 2\pi k/N) + H(0)], n = 0,1,\ldots,N
\]

where \( a = (N-1)/2 \). \( H(k) \) are the samples of the frequency response taken at intervals of \( kF_S/N \) (\( F_S \) in our case is normalized at \( 2\pi \)). For \( N \) odd, the upper limit in the summation becomes \( (N-1)/2 \). A lowpass non-recursive filter implemented using the ‘FreqSamp.’ block is illustrated in Figure 6.

Figure 7 depicts a comparison of the non-recursive lowpass filter realizations with and without samples in the transition band for \( N \) equal to 16. We can see the trade-off between the sidelobe level and the transition width.
FIGURE 7
COMPARISON OF TWO NON-R LOWPASS FILTER REALIZATIONS WITH AND WITHOUT TRANSITION SAMPLES

Least Squares Implementation

The least-squares (LS) technique uses the frequency sampling method using fewer, \( L \) time-domain impulse response points, to approximate the \( N \)-point frequency response. In our case the filter design is accomplished using only half the sampled points (\( L = N/2 \)). We will elaborate on the least squares formulation and show how it connects to our example. Consider the set of linear equations described as

\[
Ax = b
\]

where, \( A \) is an \( N \times L \) matrix, \( b \) is an \( N \times 1 \) vector and \( x \) is an \( L \times 1 \) vector which is the unknown. For the case where \( L < N \), we have more equations than unknowns (over-determined case) and a unique solution is obtained by least-squares minimization, i.e.,

\[
x = (A^H A)^{-1} A^H b
\]

In this case, the unknown vector is the impulse response, \( h(n) \); matrix \( A \) corresponds to the FFT matrix (\( F \)) and the vector \( b \) is associated to the amplitude frequency samples embedded in the vector, \( H \). The approximation error in vector form can be expressed as,

\[
e = H - Fh
\]

or,

\[
e = \begin{bmatrix} 
H(0) \\
H(1) \\
\vdots \\
H(L-1) \\
H(L) \\
\vdots \\
H(N-1) 
\end{bmatrix} - \begin{bmatrix} 
\hat{h}(0) \\
\hat{h}(1) \\
\vdots \\
\hat{h}(L-1) \\
0 \\
\vdots \\
0 
\end{bmatrix}
\]

where

\[
F = \frac{1}{N}
\]

and \( H^k = e^{j2\pi k/N} \)

where \( F_L \) is an \( NL \times DFT \) matrix and \( h_{LS} \) is the size reduced impulse response, \( Lx1 \). The approximation error now can be written as

\[
e = H - F_L h_{LS}
\]

and the solution to the least-squares problem according to equation (3) is given by,

\[
h_{LS} = (F_L^H F_L)^{-1} F_L^H H = \frac{1}{N} F_L^H H
\]

From (1), (2), and (4), the least-squares impulse response can be written as,

\[
h_{LS}(n) = \frac{1}{N} \left[ \sum_{k=0}^{N-1} 2|H(k)| \cos \left[ 2\pi k(n-a)/N \right] + H(0) \right], n = 0, 1, ..., L
\]

where, \( a \) is now equal to \((L-1)/2\). A non-recursive lowpass filter using the least-squares frequency sampling method is illustrated in Figure 8.

Recursive Implementation

The DFT samples, \( H(k) \), corresponding to an FIR sequence can be regarded as samples of the filter’s z transform, evaluated at \( N \) points uniformly spaced around the unit circle i.e.

\[
H(k) = H(z) \bigg|_{z=e^{j(2\pi/N)k}}
\]

This is given by,

\[
H(z)=H(z)H(z)
\]

where,

\[
H(z)=N^{-1}-z^{-N}
\]

Note that \( H(z) \) has \( N \) zeros uniformly distributed around the unit circle, and on the other hand, \( H(z) \) has \( N \) single poles. In practice, however, the design suffers from the inaccuracy of poles and zeros locations. The latter causes the \( H(z) \) to be unstable and
consequently the filter to perform as an IIR. Stability problems can be avoided by sampling $H(z)$ at a radius $r$ slightly less than unity.

$$H_r(z) = \frac{1-r^2}{N} \sum_{k=0}^{N-1} \frac{H(k)}{1-r \cos(\frac{2\pi k}{N}) z^{-1}}$$

The above equation can be expanded as,

$$H_r(z) = \frac{1-r^2}{N} \sum_{k=0}^{N-1} \frac{H(k)}{1-2r \cos(\frac{2\pi k}{N}) z^{-1} + r^2 z^{-2}} H(0)$$

In the case of the recursive frequency sampling (Figure 9) method, the poles are canceled by the zeros and the overall transfer function has effectively no poles. The latter statement indicates the nature of such a filter which is an FIR instead of an IIR filter.

**QUADRATURE MIRROR FILTER BANK SIMULATION IN J-DSP**

Work is being done on the development of J-DSP modules to perform QMF and filterbank simulations. In addition, graphical-modules that expose students to the analysis/synthesis filterbank structures and subband filtering concepts have been developed. Figure 10 shows the two-channel QMF simulation in J-DSP. Some audio coding applications require dividing the processed signals into subbands. This gives the advantage that bands can be processed separately taking into consideration perceptual properties associated with each band. Dividing signals into subbands introduces aliasing noise to the signal due to the overlapping bands. Aliasing cancellation is achieved using QMF banks. QMF banks consist of antialiasing filters, a down-sampling stage, an up-sampling stage, and interpolation filters (Figure 10).

**ASSESSMENT RESULTS**

The assessment questions are directly related to the technical aspects of the J-DSP on-line laboratories. The concept-specific forms focus on each exercise by posing questions that determine whether the student has learned a specific DSP concept. For instance, 87% of the students agreed that the filter design exercise helped them understand which window is suitable for sharp transitions, 88% of the students understood better the signal symmetries in the FFT spectra because of J-DSP visualization, and 91% of the students reported that the z-transform exercise helped them understand the relation between the pole-zero locations and the frequency response plots.

More results are given in Table 1. Students also recommended some changes in J-DSP on-line tools used in the lab simulations. One common suggestion is to accommodate a facility to save a workspace in J-DSP editor for the future use. This problem is now solved as the new version of J-DSP is capable of importing/exporting the work space details as a script file. After an initial assessment by the J-DSP support team, all software bugs reported by the students were fixed. We are especially appreciative of the ability to get immediate feedback from the students on the operation of the software.
TABLE I
STATISTICS BASED ON THE CONCEPT-SPECIFIC ASSESSMENT. TOTAL NUMBER OF STUDENTS THAT CONTRIBUTED THE DATA = 87 [FROM THE SPRING AND FALL SEMESTERS OF THE YEAR 2003]

<table>
<thead>
<tr>
<th>Evaluation Questions</th>
<th>Strongly Agree (%)</th>
<th>Agree (%)</th>
<th>Neutral (%)</th>
<th>Disagree (%)</th>
<th>Strongly Disagree (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. My understanding of the concepts of FIR and IIR filter design is enhanced by the J-DSP labs</td>
<td>42</td>
<td>47</td>
<td>8</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2. I have learned how to generate a sinusoid with a digital filter</td>
<td>29</td>
<td>55</td>
<td>11</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3. The relationship between the impulse response and the transfer function is clear</td>
<td>95</td>
<td>N/A</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. After performing the J-DSP lab it is clear that the FFT spectral resolution is limited by the FFT size, the window type, and the window size</td>
<td>99</td>
<td>N/A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. J-DSP labs enhanced my learning of the basic DSP concepts</td>
<td>92</td>
<td>N/A</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CONCLUSIONS

This paper described web-based computer laboratory experiments and related assessment results for digital filter design modules that have recently been integrated into the ASU’s J-DSP tool. Filter design experiments based on windowing, frequency-sampling, the Kaiser-design, Min-Max design, and IIR analog filter approximations have been presented.

ACKNOWLEDGMENT

The Java-DSP project was funded in part by the NSF CCLI program grant DUE 0089075.

REFERENCES