ON-LINE LABORATORIES FOR SPEECH AND IMAGE PROCESSING AND FOR COMMUNICATION SYSTEMS USING J-DSP

A. Spanias, V. Atti, Y. Ko, T. Thrasyvoulou, M.Yasin, M. Zaman,
T. Duman, L. Karam, A. Papandreou, K. Tsakalis

Department of Electrical Engineering, MIDL - TRC
Arizona State University, Tempe, AZ 85287-7206, USA

ABSTRACT

J-DSP[+] is a java-based object-oriented programming environment that was developed at Arizona State University for use in the undergraduate DSP class [1]. In this paper, we describe innovative software extensions on J-DSP to accommodate on-line laboratories for speech processing, image processing, and communications systems. Significant modifications in the object-oriented GUI of J-DSP that enable simulation of feedback systems are also presented. The speech processing functions enable on-line simulations of speech coding algorithms and include PCM and ADPCM quantization as well as more elaborate algorithms such as the LPC and the CELP. Image processing functionalities include development of 2-D signal processing capabilities including 2-D-FFT, 2-D-filter design, and 2-D graphics and picture processing. Communications functionality covers several aspects of analog and digital modulation and demodulation. On-line laboratory exercises have been developed in the aforementioned areas and posted on a web site (http://jdsp.asu.edu). This site also includes on-line evaluation forms for the exercises. Statistical and qualitative evaluations that assess the learning experiences of the students that use J-DSP are presented.

1. INTRODUCTION

At Arizona State University (ASU) DSP-related courses are well attended by distance learning students. In order to provide on-line laboratory experiences to distance learners the ASU Multidisciplinary Initiative on Distance Learning (MIDL) laboratory developed and tested successfully an exemplary laboratory prototype tool [1], called Java-DSP (J-DSP), for use in the undergraduate DSP class. This simulation environment enables students to establish and execute DSP simulations from any computer equipped with a browser. The MIDL is currently developing and evaluating significant extensions of this J-DSP prototype in other areas of undergraduate education. In this paper, we present innovative software extensions on J-DSP to accommodate on-line laboratories for speech processing, image processing, and communications systems. Significant modifications in the object-oriented GUI of J-DSP that enable simulation of feedback systems are also discussed. The extensions presented have been funded by the NSF CCLI program and involve developing and disseminating the new J-DSP functions along with a series of on-line laboratory exercises.

[+] Funded in part by a Grant from NSF CCLI. For more information on J-DSP and its dissemination contact spanias@asu.edu.
J-DSP concept by A. Spanias

2. EXISTING DSP FUNCTIONS IN J-DSP

The J-DSP editor is an object-oriented simulation environment. All functions in J-DSP appear as graphical blocks that are divided into groups according to their functionality. Existing functionalities include: filter design, FFT, plotting, autocorrelation, periodograms, correlograms, upsampling, downsampling, AR time-series, signal generation, etc. Details on these blocks are given in [1] and at http://jdsp.asu.edu. Simulations are established by linking blocks and establishing a flowgram.

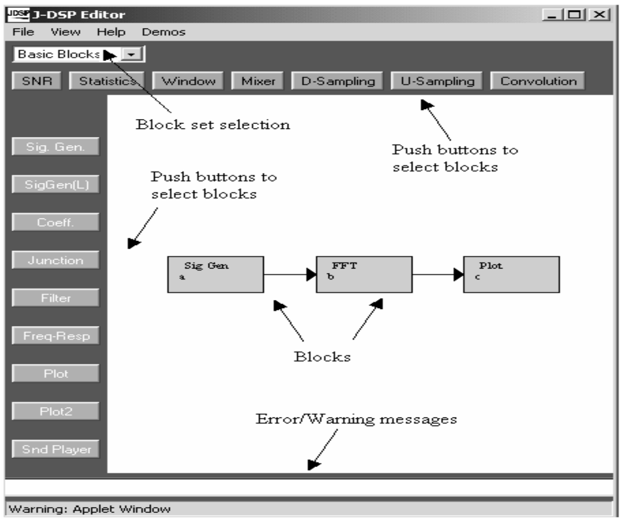


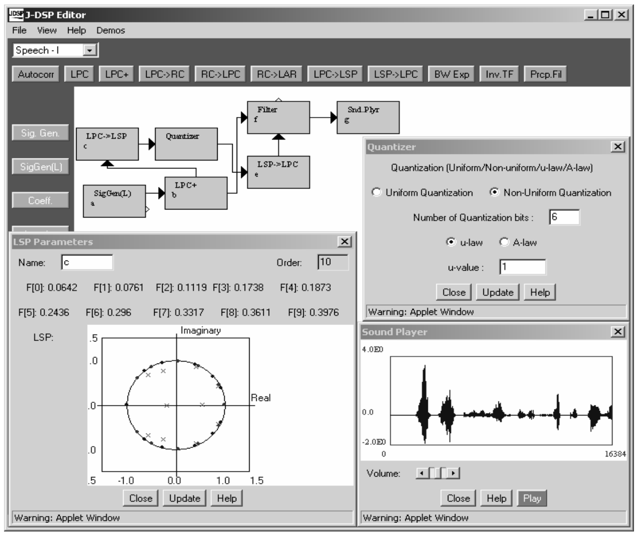
Figure 1. J-DSP Environment
(J-DSP can be accessed from http://jdsp.asu.edu)

3. NEW J-DSP FUNCTIONS

Several new functions have been developed to support experiments exposing undergraduates to additional DSP-related topics such as speech analysis-synthesis, image processing, and communications systems.

3.1. Speech Processing

The speech processing blocks supported by J-DSP include: frame-by-frame processing of speech, filter parameter transformations, line spectrum pairs (LSP), bandwidth expansion, perceptual weighting, pulse code modulation (PCM), differential pulse code modulation (DPCM), adaptive differential pulse code modulation (ADPCM), quantization, vector quantization (VQ), open-loop and closed-loop pitch estimation methods, voicing decision, LPC-10e implementation, etc. A sound player block is also provided to listen to the processed speech record.



Quantizer

Quantization (Uniform/Non-uniform/u-law/A-law)

Uniform Quantization [✓ Non-Uniform Quantization]

Number of Quantization bits : 6

[✓ u-law] A-law

u-value : 1

Warning: Applet Window

LSP Parameters

Name: c Order: 10

F[0]: 0.0642 F[1]: 0.0761 F[2]: 0.1119 F[3]: 0.1738 F[4]: 0.1873

F[5]: 0.2436 F[6]: 0.296 F[7]: 0.3317 F[8]: 0.3611 F[9]: 0.3976

LSP:

Warning: Applet Window

Figure 2. LPC vocoder

A typical LPC vocoder flowgram with filter parameter transformations is shown in Figure 2. The pole-zero representation of the LP coefficients and LSP can be viewed. The reconstructed speech, obtained by filtering the residual error with the quantized LPC, is also shown. The quantizer block performs uniform and non-uniform quantization (μ-law and A-law). An example simulation demonstrating the concepts of DPCM and ADPCM [2] is shown in Figure 3.

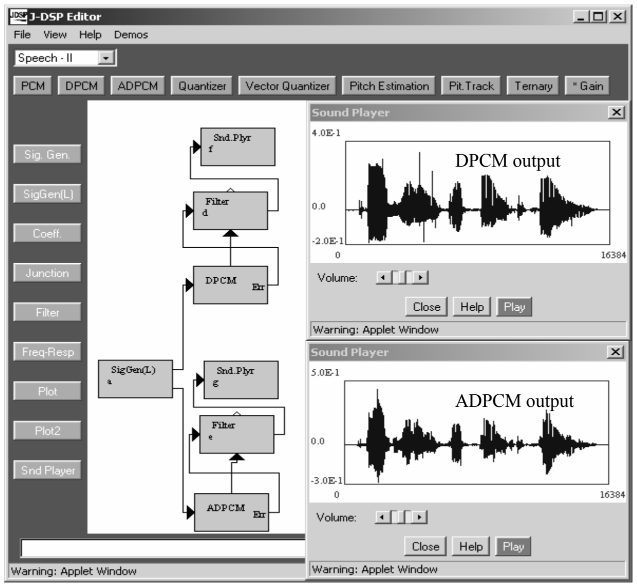
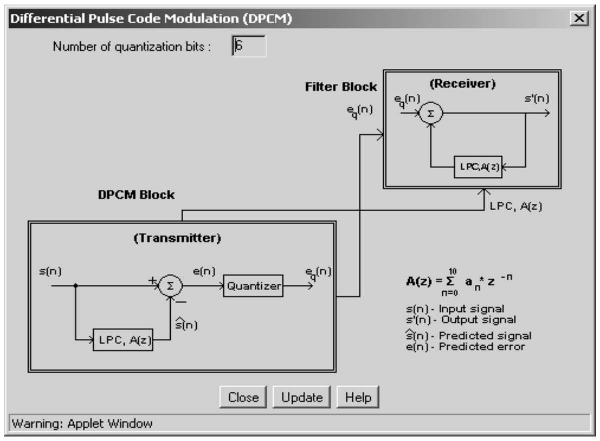


Figure 3. DPCM and ADPCM



Number of quantization bits : 6

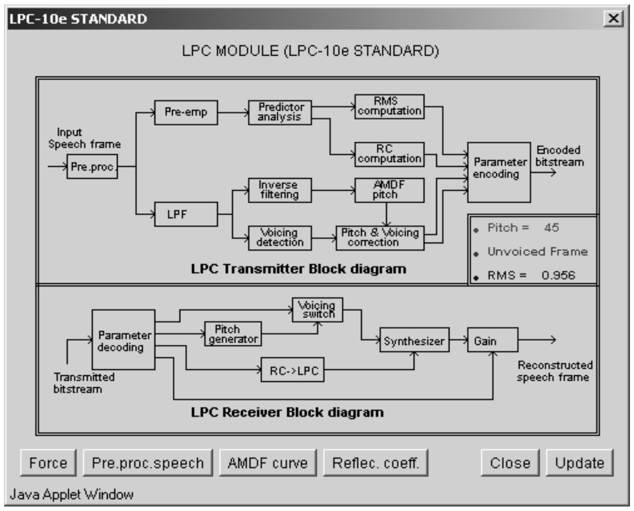
$A(z) = \sum_{n=0}^{10} a_n^* z^{-n}$

$s(n)$ - Input signal
$s'(n)$ - Output signal
$\hat{s}(n)$ - Predicted signal
$e(n)$ - Predicted error

Figure 4. DPCM dialog window

The block diagram shown in Figure 4 represents a simulation of a DPCM transmitter-receiver. The pitch estimation block is implemented to compute the pitch based on AMDF, open loop and closed loop methods.

Figure 5 shows the simulations of the LPC-10e standard implemented in J-DSP. The LPC-10e federal standard has been implemented as a separate module that processes 20ms speech frames (160 samples). A special feature is provided to view the updated parameter values (pitch, voicing decision, root mean squared (RMS) energy and reflection coefficients) at the end of each frame. Using the 'Force' option, provided in the LPC-10e block, one could force all the voicing frames to be voiced or unvoiced and/or force the pitch value to be a constant and evaluate the effect of each parameter on the output speech. Exercises have been developed to highlight the LPC concepts through ADPCM and LPC10e.



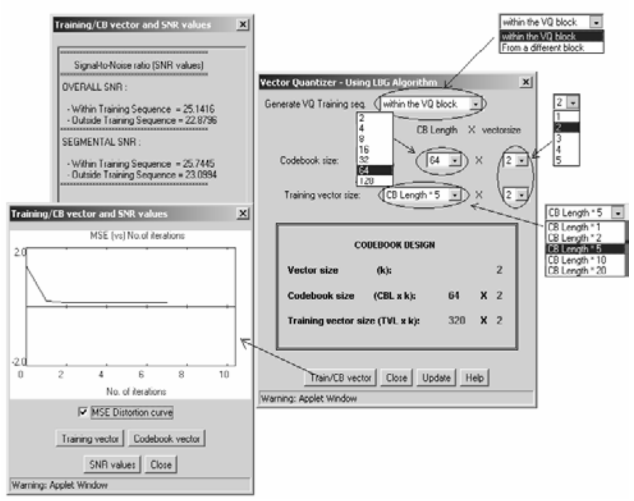
Pitch = 45

Unvoiced Frame

RMS = 0.956

Figure 5. LPC module

The vector quantization block uses the Linde, Buzo and Gray (LBG) algorithm to design codebooks. Figure 6 shows the dialog window of the vector quantizer block, and the various options provided. These include viewing the designed codebook vector, MSE distortion curve, etc. Using the vector quantizer block, one can design codebooks of various sizes and test their performance based on the overall and segmental SNR values.



Signal-to-Noise ratio (SNR values)
OVERALL SNR :
- Within Training Sequence = 25.1416
- Outside Training Sequence = 22.8796
SEGMENTAL SNR :
- Within Training Sequence = 25.7445
- Outside Training Sequence = 23.0994

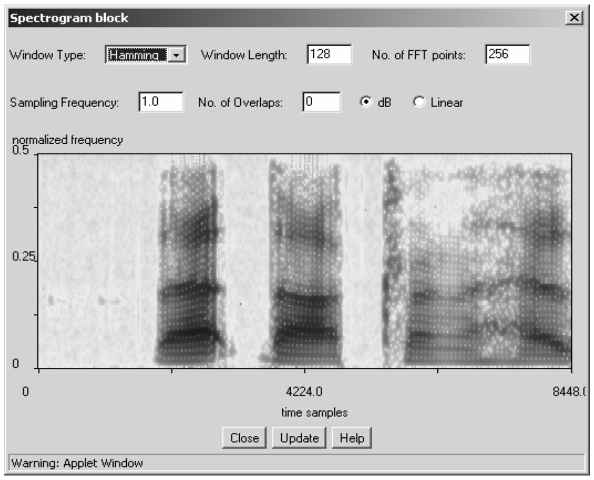
Generate VQ Training seq. within the VQ block
Codebook size: 64 X 2 (CB Length X vectorsize)
Training vector size: CB Length * 5 X 2

CODEBOOK DESIGN
Vector size (k): 2
Codebook size (CBL x k): 64 X 2
Training vector size (TVL x k): 320 X 2

Figure 6. Vector quantization based on the LBG algorithm

3.2. Time-Frequency Representations

In order to compute the time-varying spectra of speech and other signals a 3-D spectrogram function was developed. Figure 7 shows the spectrogram plot of a speech signal. The spectrogram block may be used to analyze the properties of speech segments and other signals. Exercises with the spectrogram are also provided.

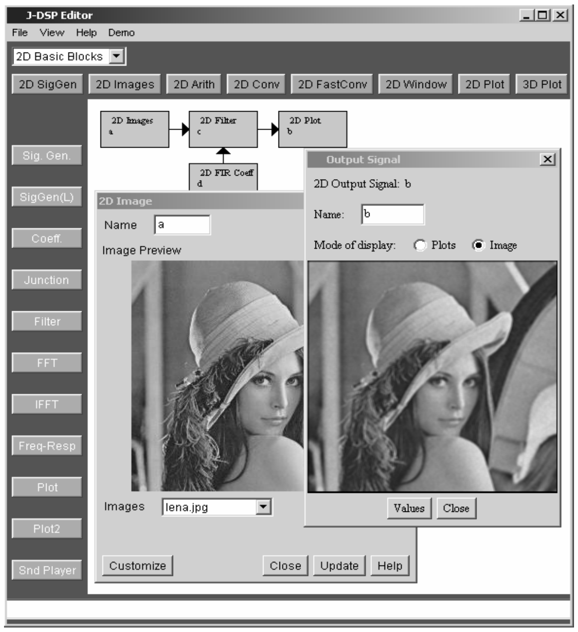


Window Type: Hamming Window Length: 128 No. of FFT points: 256
Sampling Frequency: 1.0 No. of Overlaps: 0 [✓ dB] Linear

Figure 7. Spectrogram plot in J-DSP

3.3. Image Processing

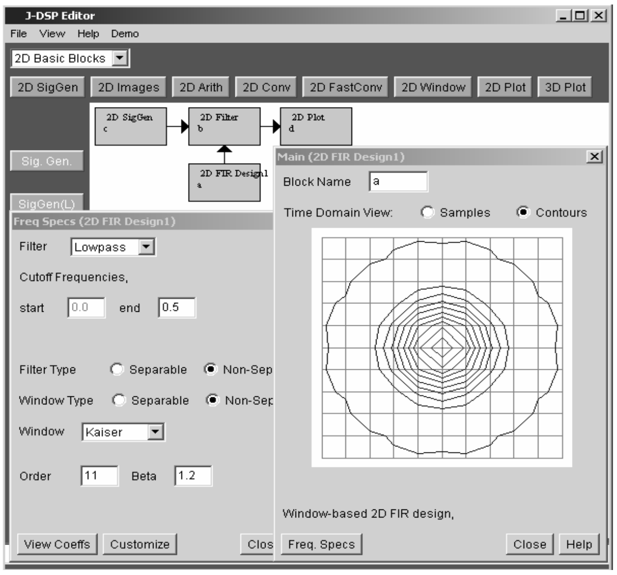
The new blocks implemented in J-DSP to facilitate the 2-D signal processing techniques are: 2-D signal generator, filtering, convolution, FIR filter design, FFT, frequency response, transforms and select functions for image restoration and enhancement. Students can perform simulations of window-based 2-D FIR design. Low-pass, high-pass, band-pass and band-stop filters can be designed using separable or non-separable design techniques. 2-D FIR filters can be implemented using time-domain convolution and FFT-based fast convolution. 2-D transforms include the discrete Fourier transform (DFT), the discrete cosine transform (DCT), and the discrete wavelet transform (DWT). Row-Column (RC) decomposition using a 1-D FFT has been implemented in the 2-D FFT block. A simple 2-D filtering system in J-DSP is shown in Figure 8.



Name: a
Images: lena.jpg
2D Output Signal: b
Name: b
Mode of display: Plots [✓ Image]

Figure 8. A simple 2-D filtering system in J-DSP

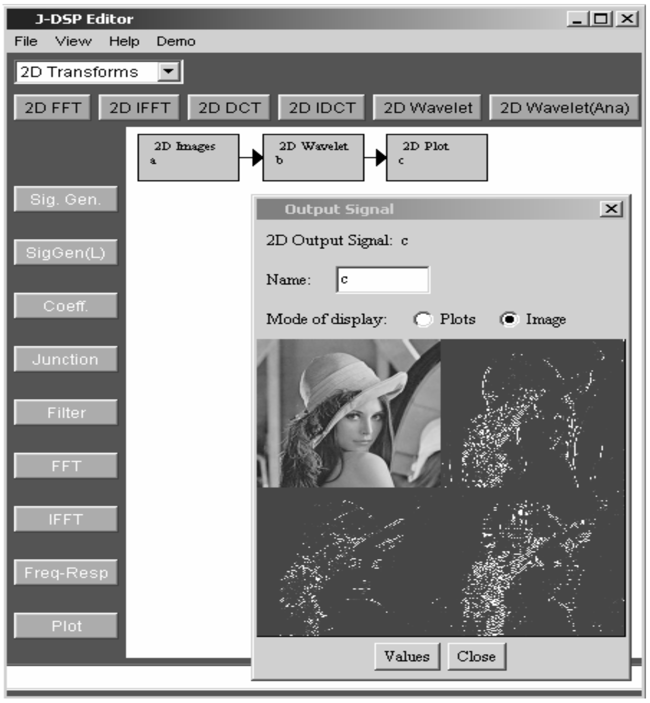
Figure 9 shows a window based 2-D FIR design, using a non-separable 2-D Kaiser window. Based on the filter specifications of the 2D-FIR design block, an 11th order low pass filter is designed. The impulse response of the designed filter can be viewed as samples or as contours as shown in Figure 9.



Block Name: a
Time Domain View: Samples [✓ Contours]
Filter: Lowpass
Cutoff Frequencies,
start 0.0 end 0.5
Filter Type: Separable [✓ Non-Sep]
Window Type: Separable [✓ Non-Sep]
Window: Kaiser
Order: 11 Beta: 1.2
Window-based 2D FIR design,

Figure 9. Window-based 2-D FIR design

The 2-D DFT and the 2-D DCT blocks are implemented as separable transforms, based on row-column decomposition using 1-D FFT and 1-D DCT algorithms respectively. The 2-D DWT is implemented using the Antonini 9/7 and 7/9 low-pass and high-pass filters. Figure 10 shows an example of a level-1 2D discrete wavelet transform.



2D Output Signal: c

Name: c

Mode of display: Plots [✓ Image]

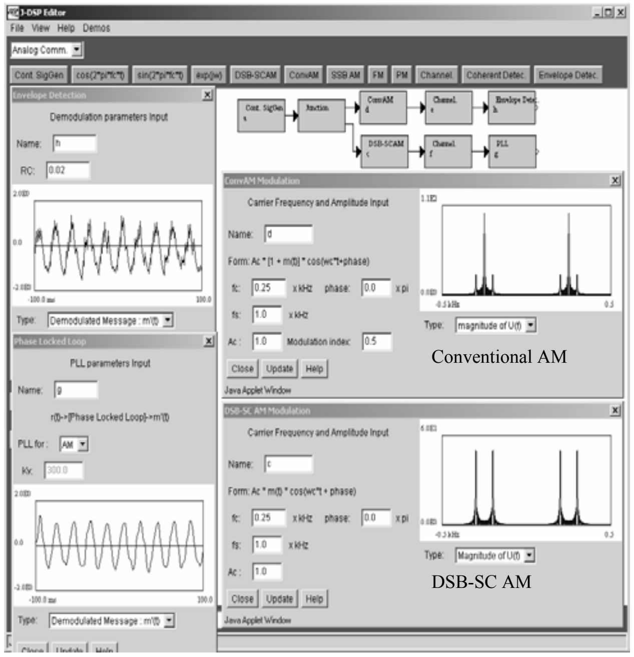
Figure 10. 2-D discrete wavelet transform (2-D DWT)

3.4. Communication Functions

The J-DSP communication functions support simulations of analog and digital communication systems. Analog modulation blocks such as amplitude modulation (DSB-SC AM, SSB AM, and conventional AM) and angle modulation (FM/PM) have been developed. Digital modulation schemes include: binary pulse amplitude modulation (PAM), M-ary PAM, phase shift keying (PSK), quadrature PSK and M-ary PSK. Receiver blocks supported are the matched filter demodulator and maximum likelihood detector. A Monte Carlo simulation block has been developed to compute bit error rate probabilities.

Figure 11 shows the spectra of the DSB-SC and conventional amplitude modulated signals. In case of DSB-SC AM scheme, the absence of the large carrier components around the carrier frequency, is clearly evident. The channel block is developed to simulate the additive white Gaussian noise (AWGN) channel with user-defined noise power spectral density. The envelope detector block, developed based on the RC circuit, is used to demodulate the conventional AM signal. The phase-locked loop (PLL) block is used to demodulate the DSB-SC AM signal.

Figure 12 shows an example that compares the performance of amplitude and angle modulation based on the output signal-to-noise ratio (SNR) values.



Demodulation parameters Input

Name: h

RC: 0.02

Type: Demodulated Message : m'(t)

PLL parameters Input

Name: g

r(t)->[Phase Locked Loop]->m'(t)

PLL for: AM

Kv: 300.0

Type: Demodulated Message : m'(t)

Carrier Frequency and Amplitude Input

Name: d

Form: Ac * [1 + m(t)] * cos(wc*t+phase)

fc: 0.25 x kHz phase: 0.0 x pi

fs: 1.0 x kHz

Ac: 1.0 Modulation index: 0.5

Type: magnitude of U(f)

Carrier Frequency and Amplitude Input

Name: c

Form: Ac * m(t) * cos(wc*t + phase)

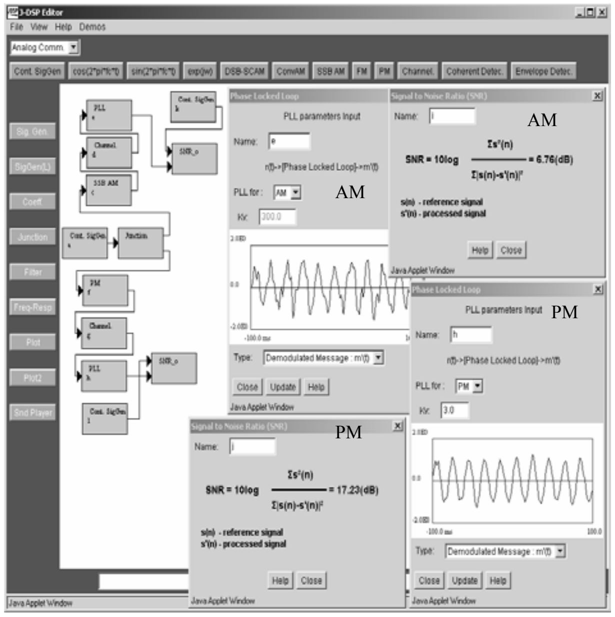
fc: 0.25 x kHz phase: 0.0 x pi

fs: 1.0 x kHz

Ac: 1.0

Type: Magnitude of U(f)

Figure 11. Conventional AM and DSB-SC modulation schemes



PLL parameters Input

Name: e

r(t)->[Phase Locked Loop]->m'(t)

PLL for: AM

Kv: 300.0

Type: Demodulated Message : m'(t)

Name: i

$$\text{SNR} = 10\log\frac{\Sigma s^2(n)}{\Sigma|s(n)-s'(n)|^2} = 6.76(\text{dB})$$

s(n) - reference signal
s'(n) - processed signal

PLL parameters Input

Name: h

r(t)->[Phase Locked Loop]->m'(t)

PLL for: PM

Kv: 3.0

Type: Demodulated Message : m'(t)

Name: j

$$\text{SNR} = 10\log\frac{\Sigma s^2(n)}{\Sigma|s(n)-s'(n)|^2} = 17.23(\text{dB})$$

s(n) - reference signal
s'(n) - processed signal

Figure 12. Performance of SSB-AM and PM schemes

A typical digital communication system with Monte Carlo simulation is shown in Figure 13. A sequence generator block was developed to generate the bit sequence. A Monte Carlo block is used to analyze the performance curves (probability of error versus the SNR plot) for various modulation schemes. A 'bit error rate (BER) and symbol error rate (SER)' block is implemented to view the probability of error versus the SNR per bit on a semi-log scale.

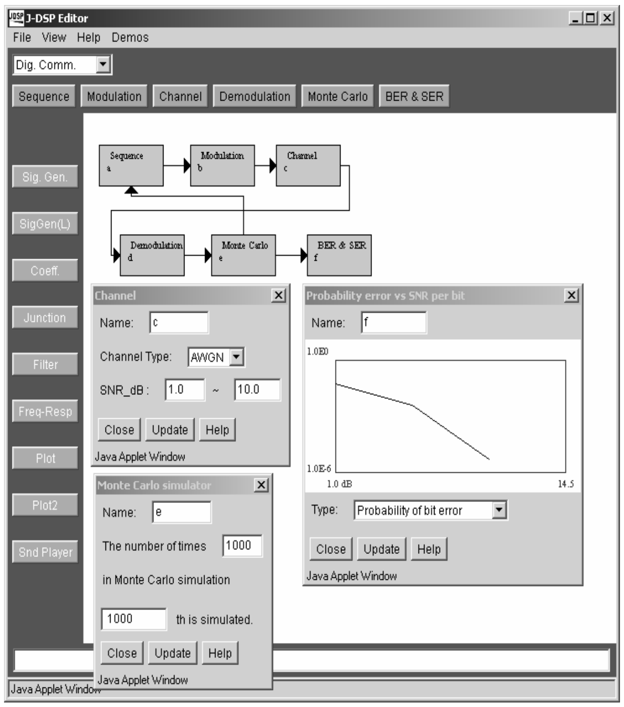


Figure 13. Computation of probability of error based on the Monte Carlo simulation

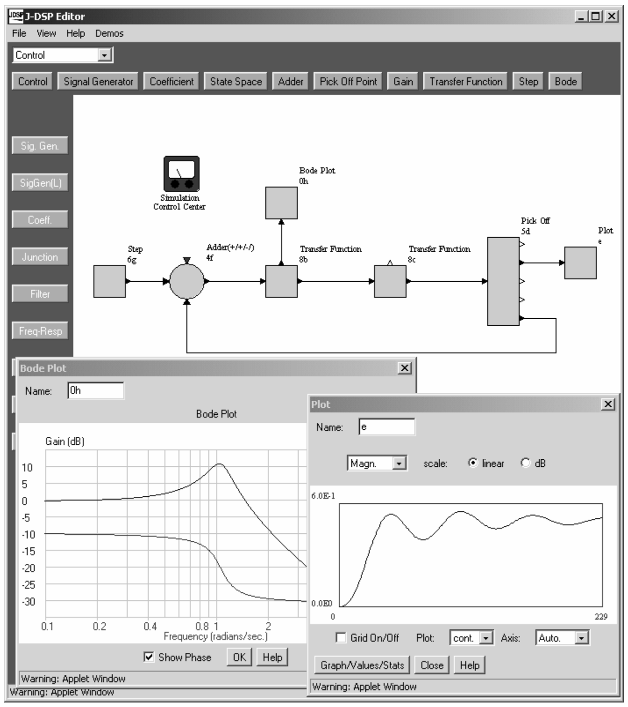


Figure 14. An example simulation of a control system.

3.5. Control Systems Functions

In addition to the functions in other areas, more functionality has been developed to facilitate control systems simulations. However, due to the nature of these simulations and the need for feedback, the new capabilities have been bundled with recent developments in the J-DSP infrastructure, explained in section 4. The controls systems simulation capabilities of J-DSP currently involve blocks for state space and transfer function representation of systems. More precisely, a state space block simulates a system given in a state space form represented by four matrices A, B, C, and D. The system implements the equation

$$\dot{\underline{x}} = A\underline{x} + Bu$$

$$y = C\underline{x} + Du$$

where $\underline{x}$ is the state vector, u is the system input and y is the system output response.

Other J-DSP control blocks include a simple adder, gain and a step signal generator. In addition, a Bode and Nyquist plot blocks are currently under development. Figure 14 shows a simple control simulation that includes two systems in cascade and a feedback loop, a bode plot of the first plant and the step response of the system.

4. J-DSP INFRASTRUCTURE EXTENSIONS

Earlier J-DSP versions have been designed with the ability to interpret parameters contained in a simple HTML (Hypertext Markup Language) file and in turn load the J-DSP Editor with a fully functional flowgram as described by these parameters. This J-DSP Editor capability has been designed to allow for interactive J-DSP content to be added in web pages.

The J-DSP Editor has the ability to automatically generate J-DSP scripts. A user simply needs to create the desired flowgram using the familiar drag and drop procedure of the editor. Then, by selecting *File* and then *Export as HTML*, the user receives the script ready to copy and paste into an HTML file. This script also includes all the block parameters, exactly as they were defined when the flowgram was saved, something earlier versions were not capable of.

In addition to the development of the new script-saving functionality, more work addresses the way blocks are designed and manipulated in the J-DSP editor. However, this work is currently being developed in a new, not yet distributed J-DSP version. Changes in the Java object-oriented program and the J-DSP GUI now facilitate state-space realizations of digital filters as well as control systems simulations. Perhaps the most important modification successfully implemented is the addition of feedback capability. Most of the J-DSP code supporting the GUI was re-written while at the same time blocks have been re-designed to offer additional features. J-DSP blocks can now be rotated and flipped thereby allowing the realization of feedback systems.

5. ON-LINE EXERCISES USING J-DSP

Exercises have been developed to emphasize the concepts of convolution, z-transform, filter design based on pole-zero placement, windowing, FIR and IIR filter design methods, FFT, power spectral density estimation based on correlogram and periodogram methods. These exercises have been used in our DSP class at ASU since 1999. Speech processing exercises include: PCM, ADPCM, LPC vocoders, bandwidth expansion and perceptual weighting filter. Students can experiment with LPC transformations involving direct form, reflection coefficients (RC), and Line Spectrum Pairs (LSP). J-DSP provides blocks that would facilitate the implementation of simple vocoders and students can experiment with pitch detection and its effect on speech synthesis. With regard to image processing several exercises have been designed including filter design, image filtering and enhancement, 2-D spectra, the DCT and its utility in JPEG, etc. Communication systems exercises include AM and FM modulators and demodulators, simulations with noise, digital modulation simulation and evaluation, and computation of bit error rates.

6. LEARNING ASSESSMENT

J-DSP user evaluation is obtained by means of on-line forms. The electronic forms have been developed for the evaluation of the J-DSP simulator and the on-line laboratory exercises. Qualitative as well as quantitative data is collected automatically and stored on the network. General assessment includes providing feedback on the DSP functions while specific forms focus on each exercise specifically by posing questions to determine whether the student has learned a concept. The evaluation forms can be accessed through the J-DSP web site http://jdsp.asu.edu/. The users fill out and submit such forms instantaneously. The feedback data too, can be accessed by links provided on-line.

Users provided valuable feedback by answering a comprehensive set of questions targeted to assess the usability and usefulness of the software. Overall, the response is very promising. 95% of the users appreciated the idea of an internet-based simulation tool such as J-DSP. From Figure 15 it is clear that it took most (70%) of the users less than half an hour to learn using the software. In fact, 85.5% of the users agreed that they would consider using J-DSP for small simulations.

Table-1: Statistics based on user evaluations

Evaluation questions	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1. Establishing and connecting blocks is easy.	53%	39%	7%	1%	0%
2. The graphical interface of J-DSP is intuitive and user-friendly.	31%	63%	5%	1%	0%
3. Setting up the required lab simulations was easy	40%	52%	8%	0%	0%

The students from the DSP and communication classes provided feedback directly related to the lab exercises. Students pointed out different parts of a lab exercise that helped them understand a certain concept. 87% of the students agreed that the FIR and IIR filter exercise helped them understand which window is suitable for sharp transitions in a filter. From the FFT exercise, 88% could clearly visualize signal symmetries on the FFT spectra. 91% users reported that the Z transform exercise helped them understand the relation between the positions of poles and zeros with the frequency response plots. 86% students agreed that the modulation exercise helped them understand the concepts of amplitude and angle modulation schemes.

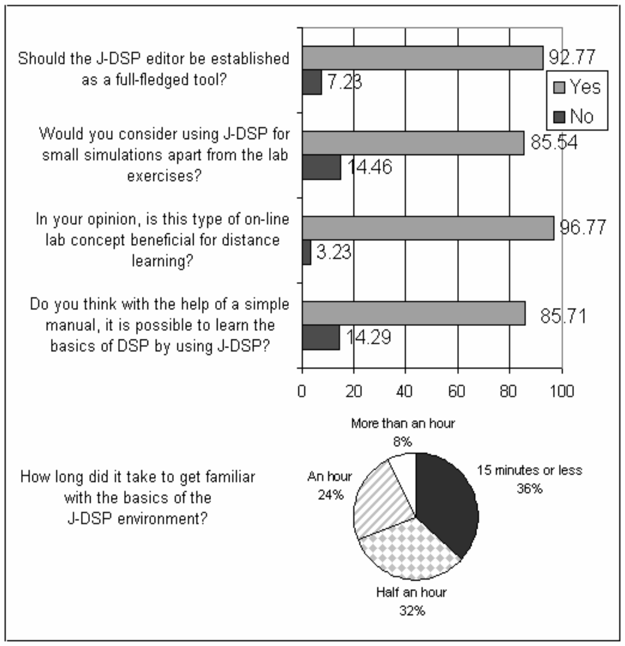


Figure 15. User feedback

7. REMARKS

This paper presented NSF funded extensions on J-DSP along with assessment results. Future J-DSP Editor versions will allow a user to create and save composite blocks, by grouping together a collection of primary blocks. New blocks have been designed to be easily adjustable with regard to the number of inputs and outputs. Each block can now have up to ten inputs or outputs on each side. Finally, in order to achieve better block and connection placement, changes have been made in order to allow a user to drag and modify a connection line as necessary.

8. REFERENCES

[1] A. Spanias et al, "Development and Evaluation of a Web-Based Signal and Speech Processing Laboratory for Distance Learning", *Proc. IEEE ICASSP-2000*, Istanbul, Vol. 6, pp. 3534-3537, June 2000.

[2] Andreas Spanias, "Speech Coding: A Tutorial Review", in the *Proc. of IEEE*, Vol. 82, No.10, pp. 1541-1582, Oct. 1994