DEVELOPMENT OF SOFTWARE INFRASTRUCTURE FOR COLLABORATIVE SIMULATIONS IN JAVA-DSP

Thesis Defense
Master of Science, Electrical Engineering

Ravi Kanth Reddy Chilumula
Arizona State University, Tempe, AZ 85287-5706, USA

Committee:
Dr. Andreas Spanias, Chair
Dr. Antonia Papandreou-Suppappola, Member
Dr. Lina Karam, Member

Outline

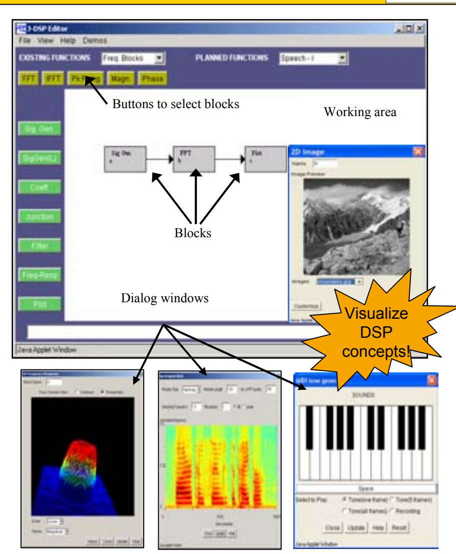
- Motivation
- J-DSP – A Multi-disciplinary Simulation Environment
- Development of Software Infrastructure for Collaborative Simulations
  - Server module
  - Working procedure
  - J-DSP Chat Functionality
- Collaborative Signal Processing Simulation exercises
- Preliminary evaluation – Script transfer delay estimation
- Assessment & Preliminary Statistical Results
- Signal Processing Functions Implementation
  - Averaging Filter, Median Filter, Decimation and Correlogram+
- Conclusions & Future Work

Motivation

- Collaboration results in better understanding of concepts [1] by discussions with peers and instructors (via a chat window in J-DSP). Accelerate learning by exposing students to advanced level concepts in DSP [2].
- Most signal processing applications involve analysis-synthesis, transmitter-receiver and encoder-decoder kind of simulations which can be used to form collaborative simulations for 2 or more users.
- To our knowledge no other signal processing simulation tool that has an in-built support for running collaborative simulations exists.
- The existing J-DSP tool already possesses many capabilities and supports simulation of a wide variety of signal processing functions. Also, J-DSP is already in use for performing on-line laboratories and is being used in at least four other universities [3] [4] [5].
- Provide hands-on experiences to undergraduate/ graduate DSP students and distance learners.

The J-DSP tool

- J-DSP is an on-line, object-oriented graphical DSP editor written as a Java applet.
- Useful for distance learning education as it enables distance learners to simulate DSP systems.
- Users can view the results at any point of the simulation, graphically or numerically.
- Provides a simple and user-friendly interface.
- Platform independent and is universally and freely accessible.



J-DSP Overview

Basic functionality in J-DSP

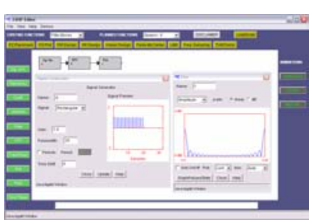
- Fundamental DSP Functions (FFT, IFFT, Windowing etc.)
- Basic Arithmetic Functionality
- Multi-rate DSP
- Pole-Zero z-domain diagrams
- Frequency Response
- Visualization Blocks
- Digital Filtering
- FIR/IIR Filter Design
- Spectral Estimation
- 3D Animations

Other functionalities:

- Analog and Digital Communications
- Control Systems
- Image and 2D Signal Processing
- Speech Analysis and Synthesis
- Time/Frequency Representations
- MIDI Functionality for High Schools
- Digital Audio Effects
- J-DSP Scripts for Interactive Web lectures
- Automated MATLAB Scripts

J-DSP Scripts

J-DSP simulation can be represented in the form of special text called J-DSP script [6].



Export Script

Import Script

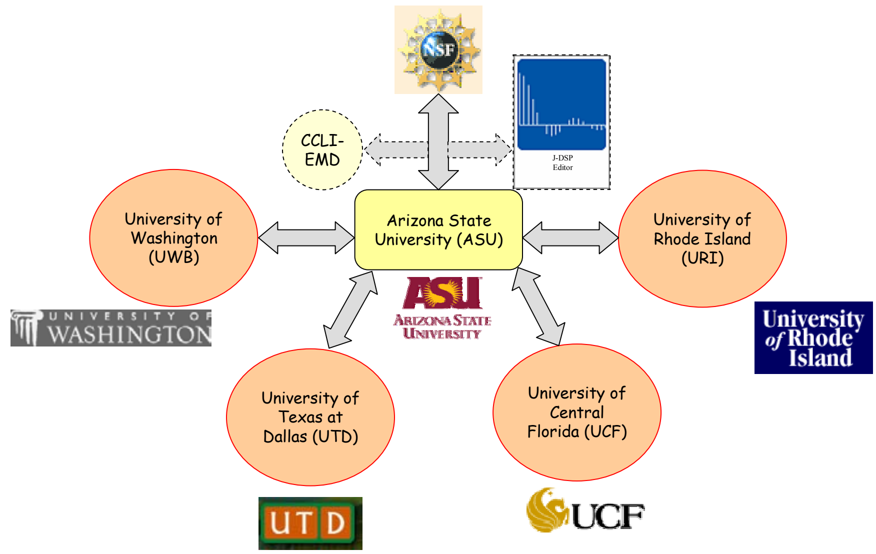
```
applet CODE="JDsp.class" width="400" height="250" param
name="numCommand" value="10" !-- START PARTS -- param
name="0" value="B0-siggen(0,0)" param name="1" value="B1-
fft(1,0)" param name="2" value="B2-plot(2,0)" !-- END PARTS
--!-- START CONNECTIONS -- param name="3" value="C-0-
4-1-0" param name="4" value="C-1-4-2-0" !-- END
CONNECTIONS -- !-- START OPEN DIALOGS -- param
name="5" value="O-0" param name="6" value="O-2" !-- END
OPEN DIALOGS -- !-- START PART PARAMATERS. * DO
NOT MODIFY! * -- param name="7"
value="P0~20,10,0,~1.0,0.9,0.0,0.2,~a,Rectangular,No,null,~~"
param name="8" value="P1~256,~~b,~~" param name="9"
value="P2~~~c,cont.,Amplitude,linear,~false,~"  !-- END PART
PARAMATERS -- /applet
```

Editor frame **J-DSP Script**

This scripting capability is used in the new software infrastructure.

Ira A. FULTON
School of Engg.

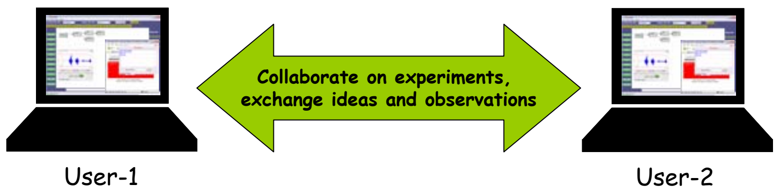
Collaborative Research



Ira A. FULTON
School of Engg.

Collaborative Simulations Software Infrastructure

J-DSP Blackboard concept

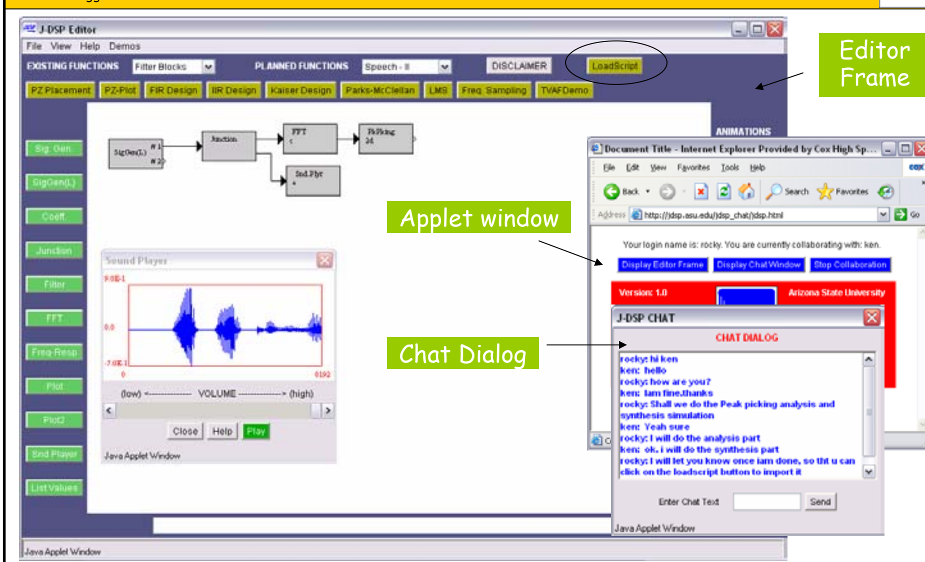


Applications:

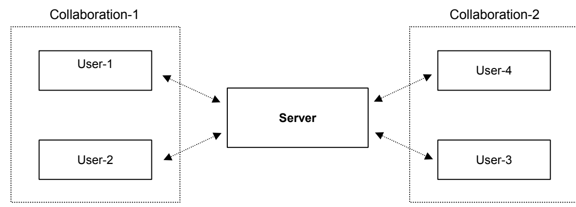
- Research
- Distance learning and on-campus education

J-DSP Blackboard Concept by Prof. Andreas Spanias, ASU.

Snapshot of a real-time J-DSP collaborative simulation environment



Overview of J-DSP Collaboration Infrastructure

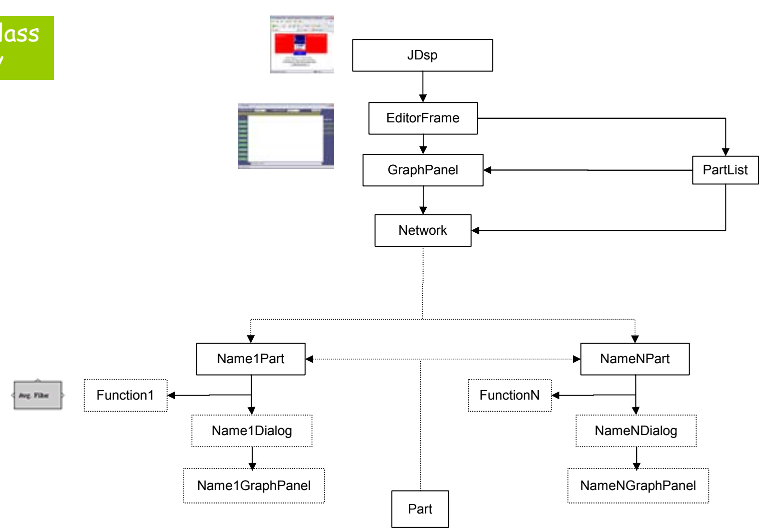


Why do we need a Server module?

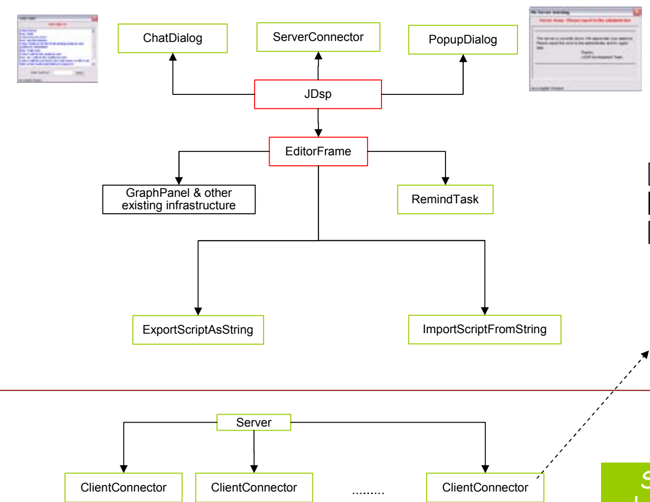
- Transfer of J-DSP scripts [6] for running collaborative experiments required some form of communication between different users.
- J-DSP is basically a Java applet which is executed by first downloading it in to the clients computer by the web browser.
- For security reasons these Java applets are allowed to communicate through sockets only with the server from which they were downloaded and not with any another computer.
- Also, since all the communications are via server, the administrator can have a control over them. Several actions involved with the software can be moderated.

Existing J-DSP Infrastructure

Client side class hierarchy



New J-DSP Infrastructure



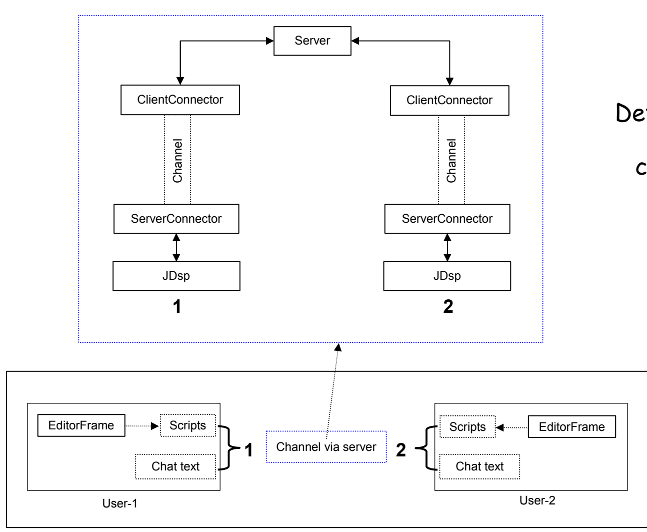
Client side class hierarchy

New classes
Existing classes
Existing classes, modified

One for each client

Server side class hierarchy

New J-DSP Infrastructure II



Detailed mechanism behind the J-DSP chat and collaborative simulation features

Ira A. FULTON School of Engg.

New J-DSP Infrastructure III

The Export Script mechanism of J-DSP transforms the simulation environment in to the J-DSP Script representation which is sent as a Java string via server to the collaborator's J-DSP environment.

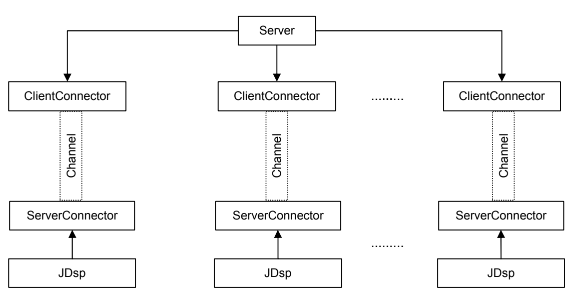
This process is automated to operate once in every 2 seconds.

Also each user is provided with a button labeled *LoadScript* which would call a function that imports the collaborator's simulation environment from the latest J-DSP script received.

Button for importing the collaborator's simulation environment. Activated only when collaborating with someone.

Ira A. FULTON School of Engg.

New J-DSP Infrastructure IV



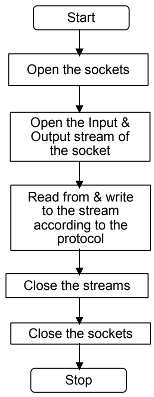
All users are connected to the server.

Interaction between the server side and client side, takes place through the *ServerConnector* and *ClientConnector* classes.

Ira A. FULTON School of Engg.

Virtual Channel via Sockets

J-DSP Editor



Typical operations in a socket based communication

Ira A. FULTON School of Engg.

Software details

J-DSP Editor

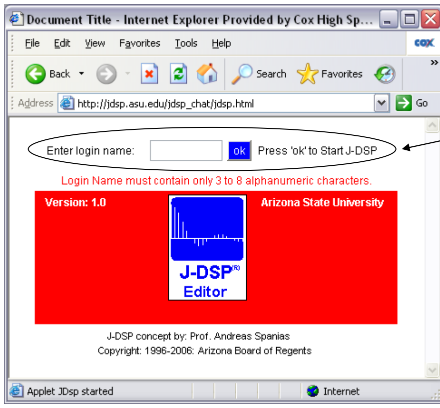
Class Name	No. of lines of code
JDsp	620 *
EditorFrame	120 *
Server	81
ServerConnector	163
ClientConnector	133
PopupDialog	91
ImportScriptFromString	155
ExportScriptAsString	564
ChatDialog	175
Total	**2102**

2100 lines of new Java code in total.

Does not include comments and extra spaces.

*only new code

Working with J-DSP Collaborative Simulations



J-DSP Initial Login Screen
Enter name, press 'OK' button

http://jdsp.asu.edu/jdsp_chat/jdsp.html

Java Run-time Environment (JRE) must be installed to run this software.

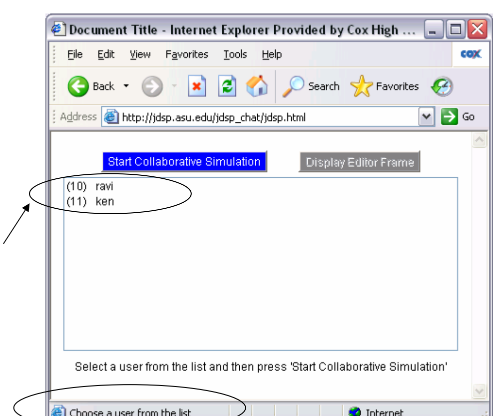
Ira A. FULTON
School of Engg.

Working with J-DSP Collaborative Simulations II

J-DSP
Editor

List of users available for collaborative simulations

Messages are displayed for the user

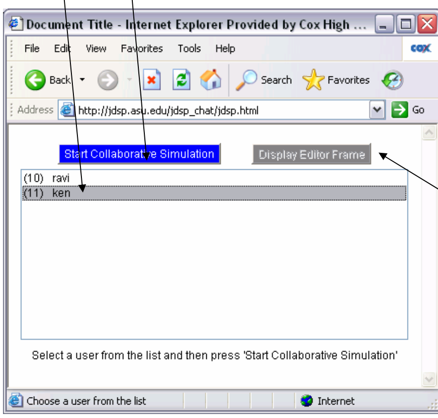


Ira A. FULTON School of Engg.

Working with J-DSP Collaborative Simulations III

Step 1: Select a user

Step 2: Click the 'Start Collaborative Simulation' Button



Button useful to reopen the editor frame window.

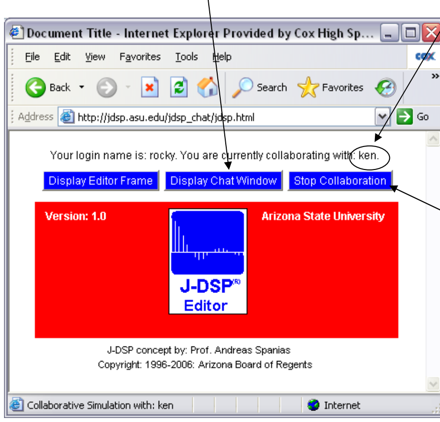
Gets activated as soon as the editor frame is closed and vice versa.

Ira A. FULTON School of Engg.

Working with J-DSP Collaborative Simulations IV

Button useful to reopen the chat window. Similar to "Display Editor Frame" button.

Collaborator's name is displayed

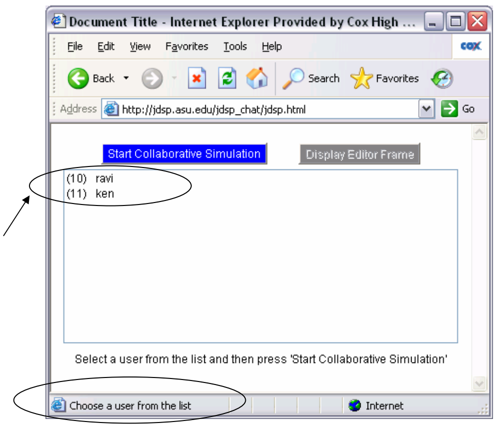


Clicking on this button stops the current collaboration. A user list containing users who are not already involved in a collaboration is displayed again.

Working with J-DSP Collaborative Simulations II

List of users available for collaborative simulations

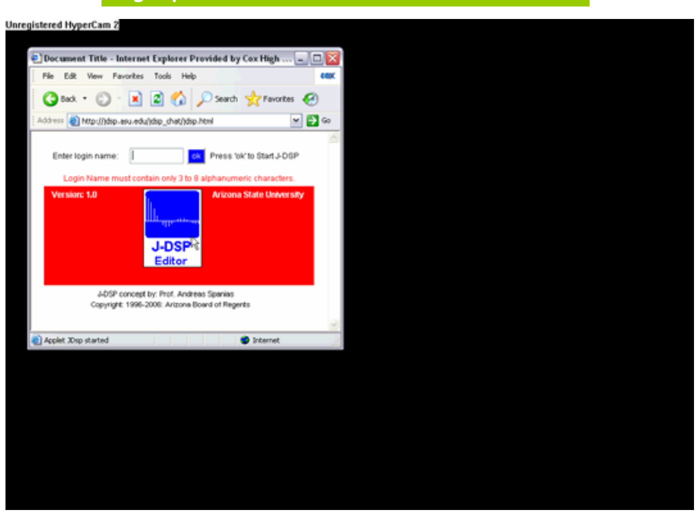
Messages are displayed for the user



Ira A. FULTON School of Engg.

Demonstration Video

Login procedure and other user interface

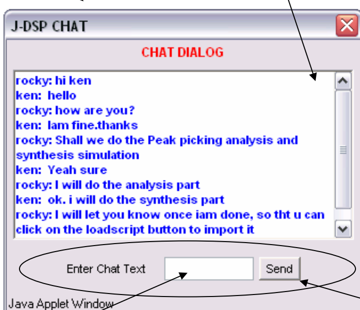


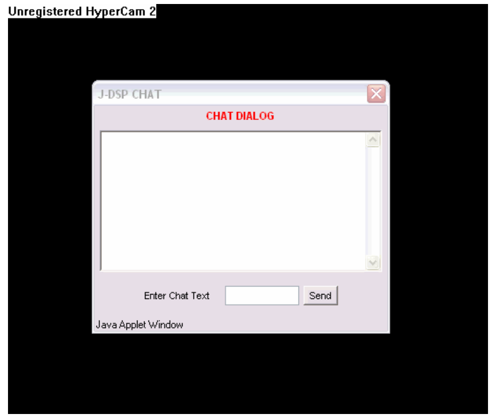
Ira A. FULTON
School of Engg.

J-DSP Chat Functionality

Chat Window

Chat text is displayed here.





Enter Text and press 'ENTER' button on the keyboard or Click on the 'Send' button

Ira A. FULTON
School of Engg.

Pop-up dialog window

In case the server module is down, a pop-up dialog is displayed

J-DSP Collaborative Simulation Exercises

J-DSP Collaborative Simulation Exercises

Real-time collaborative experiments:

1. Pole-zero cancellations and filter system stability
2. Peak picking analysis and synthesis of a speech signal
3. Auto-Regressive (AR) System Identification using Linear Prediction

Ira A. FULTON
School of Engg.

J-DSP Collaborative Simulation Exercises II

1. In the pole-zero cancellation simulation, user-1 setups a stable filter system with a pair of poles using the pole-zero placement facility and also obtains the impulse response of the system. User-2, adds another two pairs of poles just outside the unit circle making the system unstable. User-1 adds two pairs of zeros to cancel these poles and stabilizes the system.

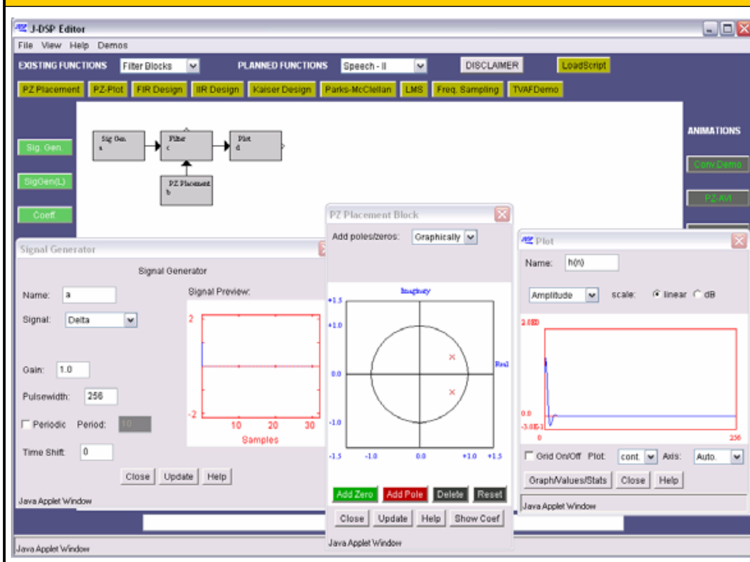
2. In the peak-picking analysis and synthesis system, user-1 generates the FFT transformed version of a male speech signal and then picks a few components from it for reconstruction. User-2 performs the synthesis of speech signal by using inverse FFT on the components selected by user-1. Both users perform subjective and objective analysis, comparison of the speech signals obtained by peak picking and picking first few components.

3. In the system identification example, user 1 creates a 6th order AR system and User 2 identifies the system using linear prediction. User-1 also informs user-2 that it is a 6th order system using the chat dialog.

Note: Collaborators exchange their simulation environment several times and also discuss various observations during the whole process using the chat window.

Ira A. FULTON
School of Engg.

J-DSP Collaborative Simulation exercise
Pole-zero cancellation



Snapshot of User-1's J-DSP Environment.

User-1 establishes a filter system with conjugate pair of poles.

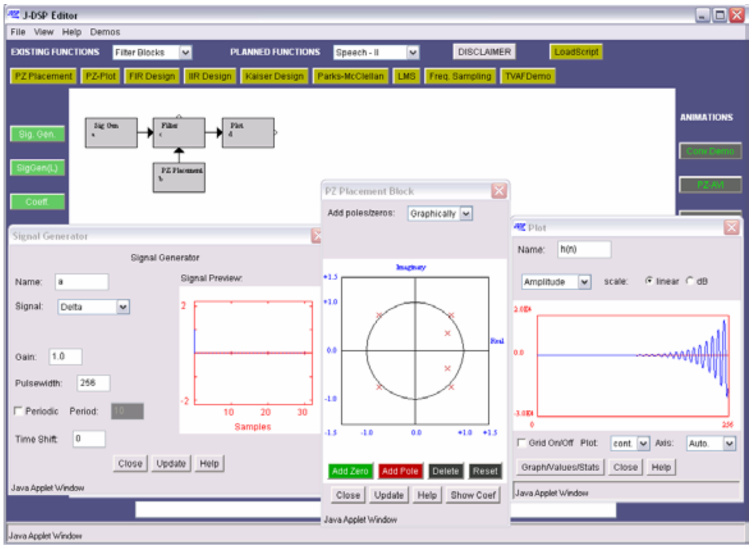
P_1 (0.66,0.36)

This is a stable system.

ASU Arizona State University
Ira A. Fulton School of Engg.

J-DSP Collaborative Simulation exercise Pole-zero cancellation II

J-DSP Editor



User-2 adds two more pairs of conjugate poles to make it an unstable system.

P_2 (0.735,0.735)

P_3 (-0.735,0.735)

ASU Arizona State University
Ira A. Fulton School of Engg.

J-DSP Collaborative Simulation exercise Pole-zero cancellation III

J-DSP Editor

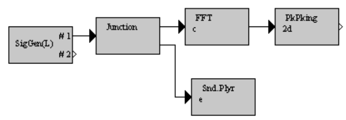
User-1 adds two pairs of conjugate zeros to make it a stable system.

Z_1 (0.735,0.735)

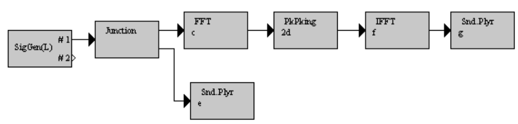
Z_2 (-0.735,0.735)

Pole-zero cancellations are used to stabilize an unstable filter system

J-DSP Collaborative Simulation exercise
Peak picking analysis & synthesis of a speech signal

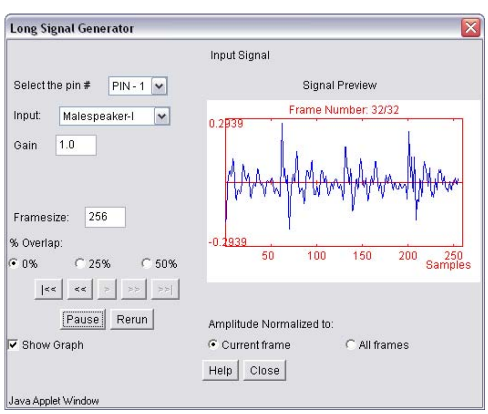


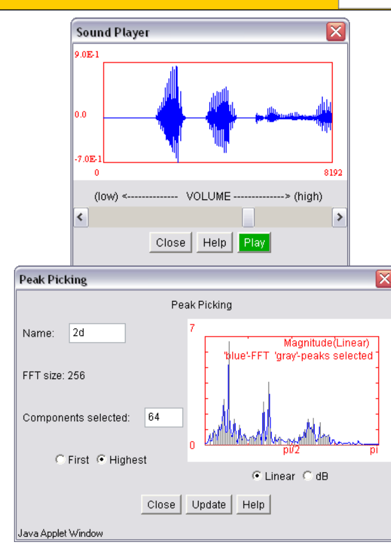
J-DSP simulation diagram for Peak picking analysis, User-1



J-DSP simulation diagram for Peak picking analysis & synthesis, User-2

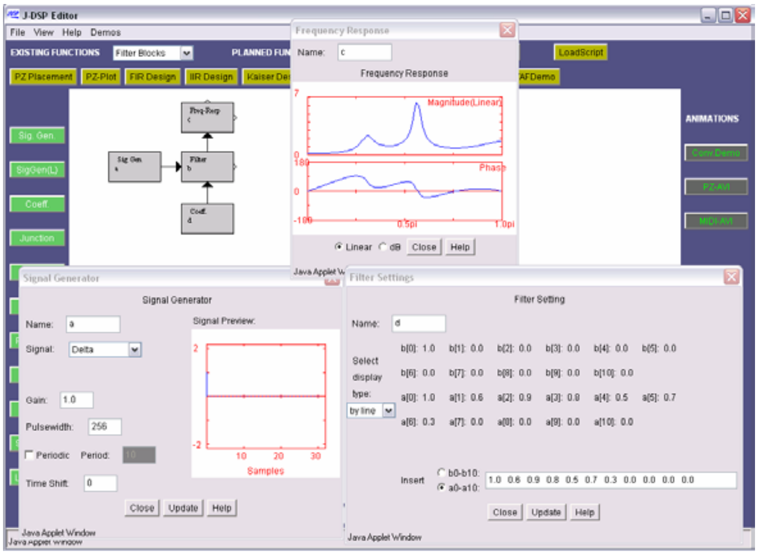
J-DSP Collaborative Simulation exercise
Peak picking analysis & synthesis of a speech signal II





Ira A. FULTON School of Engg.

J-DSP Collaborative Simulation exercise AR System Identification using LP



Snapshot of J-DSP User-1's Environment.

User-1 establishes an AR System that will be identified by User-2

User-1 informs User-2 through the chat dialog that the AR system is of 6th order.

Ira A. FULTON School of Engg.

J-DSP Collaborative Simulation exercise AR System Identification using LP II

User-2 adds LPC+ block and the Frequency Response block.

The LPC+ block identifies the AR system coefficients.

Demonstration Video

AR System Identification using Linear Prediction

Preliminary Evaluation

Preliminary Evaluation of Software

Delay Estimation

- A collaborative simulation in J-DSP involves an implicit transfer of information from one user to the other.
- Time delays for the script transfer between two computers are calculated for different types of network connections, with users from different places.
- The delay calculations are made indirectly using the *ping utility* available in the Operating System.
- Average Round Trip Times (ARTT) are obtained using the ping utility for three different packet sizes namely 1024 bytes, 5120 bytes and 10240 bytes.
- ARTT is halved to represent time taken to travel in one direction.

$$Bit\ rate = \frac{Packet\ size}{ARTT/2} \qquad \text{------ (1)}$$

- These three bit rates are averaged for each type of connection. Then the time delay (T_d) for the script transfer from a computer to server or vice versa is calculated as shown below.

$$T_d = \frac{Script\ size}{Average\ Bit\ Rate} \qquad \text{------ (2)}$$

- Equation-2 is repeated for a 3-block simulation, 9-block simulation and 21-block simulation.
- As we have the time delays for script transfer from computer-1 to server and server to computer-2 (for different connections and simulations), they can be added respectively to get the over all delay for a particular type of connection and a particular demo.
- All timings are measured in milli seconds.

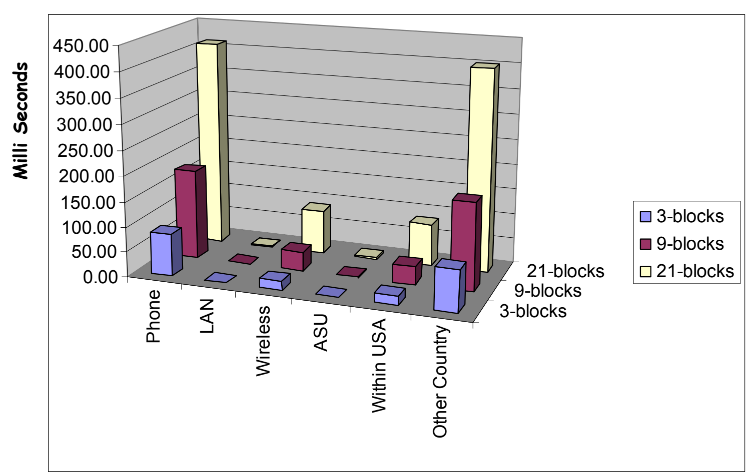
Delay values in Experiment-1

Connection type	Traffic index / Resp. time	Sum of delays between Computer-1 to Server and Server to Computer-2 values for different script sizes (in ms)		
		3-blocks (816 bytes)	9-blocks (1720 bytes)	21-blocks (3980 bytes)
Telephone	94/59	84.60	178.33	412.64
LAN (J-DSP lab)	93/60	0.21	0.46	1.06
Wireless	93/60	18.16	38.26	88.53
ASU	93/60	0.92	1.94	4.48
Within USA	94/55	17.37	36.61	84.71
Other Country (India)	67/326	82.40	173.68	401.89

We calculated the delay between two computers for various internet connection types, different places and simulations. Traffic Index and Response time are based on the website resource *http://internettrafficreport.com*. The traffic index is a score from *0 to 100* where *0* is slow and *100* is fast.

Delay values in Experiment-1

Delay plotted for different connections and no. of blocks in J-DSP



Delay values in Experiment-2

The following collaborative sites have been used for testing the developed software infrastructure:

- University of Washington, Bothell
- University of Leceister, UK
- Osmania University, Hyderabad, India

Connection type	Traffic index / Resp. time	Sum of delays between Computer-1 to Server and Server to Computer-2 values for different script sizes (in ms)		
		816 bytes	1200 bytes	1090 bytes
UWB, Telephone	94/59	84.60	124.41	113.01
London, DSL	93/60	18.15185	26.69	24.24
India, DSL	67/326	82.39	121.17	110.06

Preliminary Evaluation Results

The abilities of the new software infrastructure have been demonstrated to be working.

Collaborative simulations using J-DSP are simple and effective.

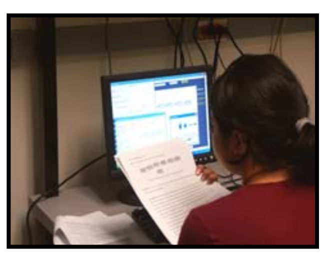
An inbuilt Chat Dialog makes them even better.

The real time delay calculations indicate the performance of the software and show that it is feasible to use the software in real-time.

Wireless Routers with firewall posed a problem as they blocked Java Sockets and hence did not allow collaborative simulations to be run smoothly.

In the collaborative simulation test with UWB site, the user at UWB was using a telephone modem and still the simulations were successful and fast enough.

Preliminary Assessment Statistics

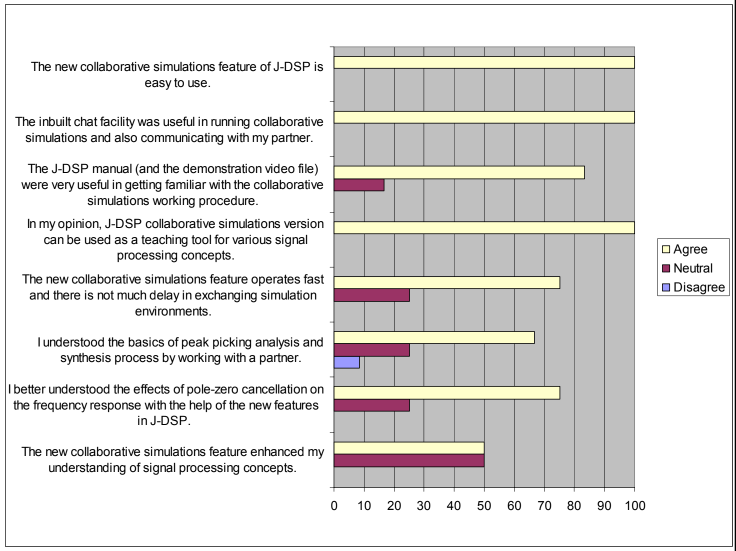


Ira A. FULTON School of Engg.

Preliminary Assessment Statistics

J-DSP Editor

General & concept specific evaluation

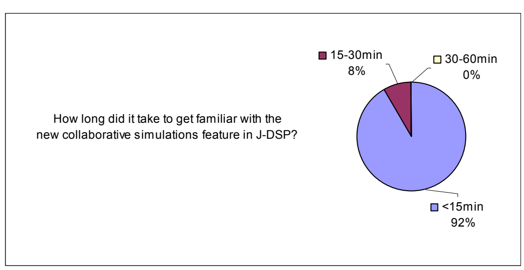


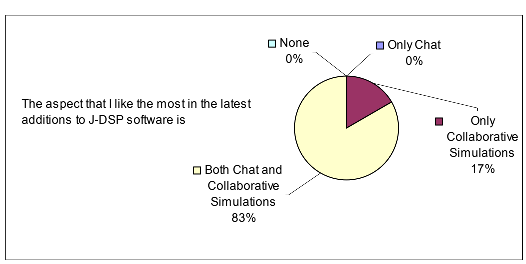
Ira A. FULTON School of Engg.

Preliminary Assessment Statistics II

J-DSP Editor

General evaluation



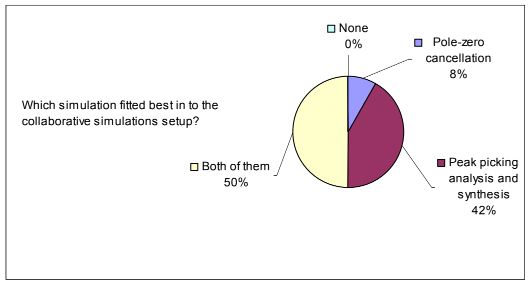


Ira A. FULTON School of Engg.

Preliminary Assessment Statistics III

J-DSP Editor

Concept Specific



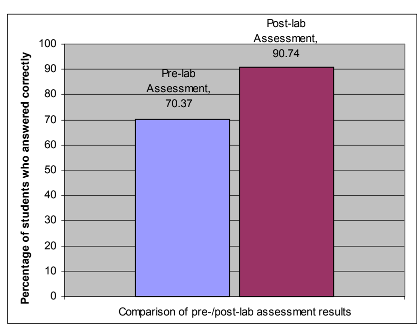
Ira A. FULTON School of Engg.

Preliminary Assessment Statistics IV

J-DSP Editor

Pre/post Assessment

- This assessment instrument focuses on evaluating whether learning of certain topics is attributed specifically to the usage of new software.
- A 20% average improvement can be noted after performing the two collaborative simulations.



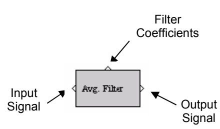
Averaging Filter

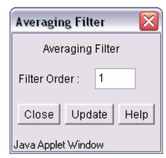
An averaging filter of order *L* (i.e., *length L+1*) is a simple low pass FIR filter that averages over a window of *L+1* samples. It is also called a linear mean filter.

$$y(n) = \frac{1}{L+1}\sum_{i=0}^{L} x(n-i).$$

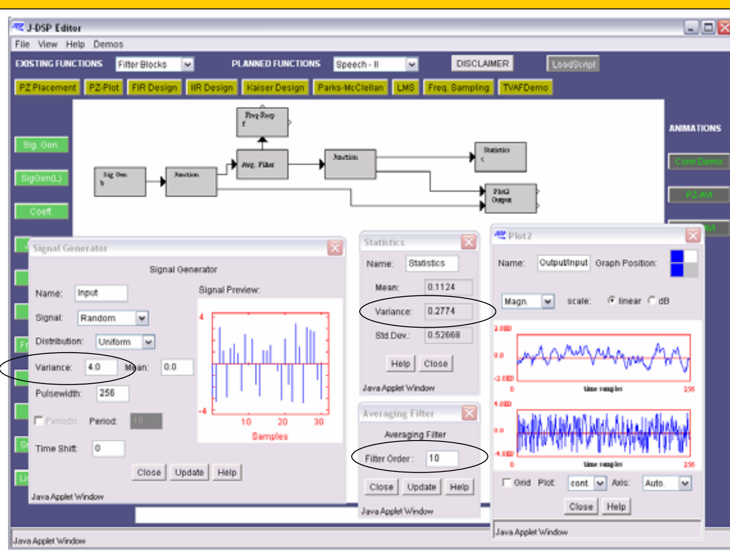
$$b[i] = \frac{1}{L+1}, i = 0,1,\ldots L.$$

x(n) - input signal
y(n) - output signal

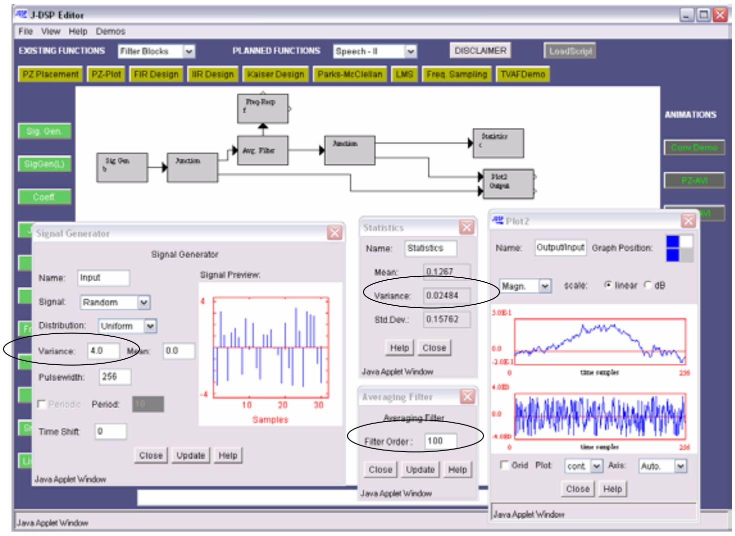




Averaging Filter II



Averaging Filter III

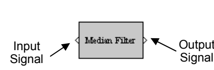


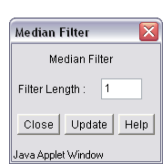
Median Filter

A median filter of length *L* is based on a moving window (of length *L*) applied to the input sequence. It calculates the median for each set of the samples by ranking the data within the window. This is taken as the output sample corresponding to the input sample on which the window had been centered. Typically, *L* is chosen to be odd, so that the window is centered on one particular input sample. However, for an even *L*, it is placed such that it has more samples to the left of it.

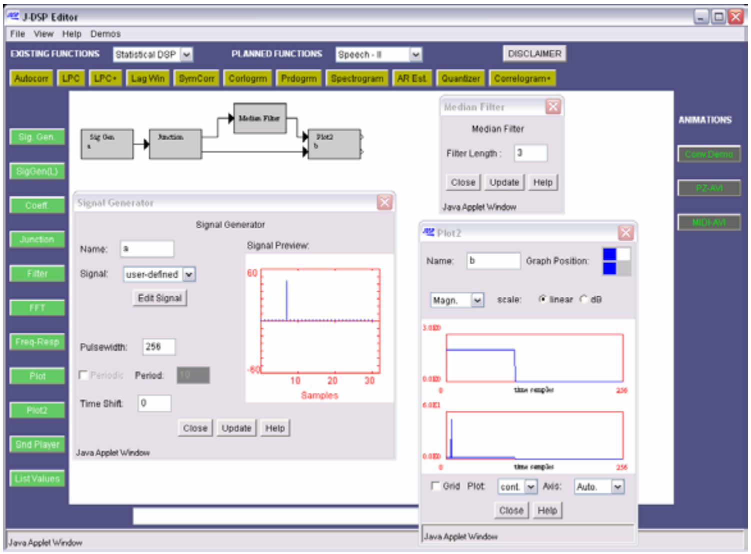
The non-linear function involved with a median filter, $L=2i+1$.

$$y(n) = median[x(n-i), x(n-i+1), ...x(n), ...x(n+i-1), x(n+i)].$$





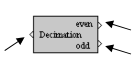
Median Filter II



A transient is smoothed down by the median filter.

Decimation block

Input Signal



Even samples

Odd samples

This block separates an input signal into even and odd signal samples.

Correlogram+

Correlogram estimates the *power spectral density* (PSD) of the input signal by calculating FFT of biased autocorrelation estimates of the data set. The output depends on the type of lag window selected, input signal and the no. of lags. Window types supported in this block are Rectangular, Bartlett, Hamming, and Gaussian. *M (=2L+1)* biased autocorrelation samples are chosen by truncating with the window *w(m)*, for calculating the PSD estimate.

$$For,\ input\ signal\ x(n),\ n = 0,1,\ldots N-1.$$

$$\hat{r}_{xx}(m) = \frac{1}{N}\sum_{n=0}^{N-|m|-1} x(n+|m|)x(n),\ \ -(N-1) \le m \le (N-1).$$

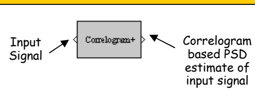
$$\hat{r}_{ww}(m) = \hat{r}_{xx}(m)w(m).$$

$$\hat{R}_{xx}(e^{j\Omega}) = \frac{1}{N}\sum_{m=-L}^{L} \hat{r}_{ww}(m)e^{-j\Omega m}.$$

where $\hat{R}_{xx}(e^{j\Omega})$ is the PSD estimate of the input signal

$\hat{r}_{xx}(m)$ is the biased autocorrelation estimate

Correlogram+ II



Correlogram estimates the PSD of a sinusoidal signal of frequency 0.5pi. The peak in the PSD plot can be observed to be at 0.5 pi.

Ira A. Fulton School of Engg.

J-DSP Editor

Conclusions & Future Work

Ira A. Fulton School of Engg.

J-DSP Editor

Conclusions & Remarks

- Developed collaborative simulations software infrastructure including a server module.
- Developed an in-built chat facility for collaborators.
- Developed few signal processing functions in J-DSP.
- Preliminary evaluation in the form of script transfer delay calculations under different network conditions involving students from different universities.
- Developed collaborative signal processing exercises for undergraduate DSP coursework.
- Preliminary assessment of the new software infrastructure and the laboratory exercises.

Ira A. FULTON School of Engg.

Future Work

J-DSP Editor

- Support multiple users in each collaboration.
- Chat facility can be upgraded to have features similar to those available in commercial chat software.
- Develop more collaborative simulation examples.
- Make it seamless (remove Loadscript button).
- Can be modified to support an instructor and multiple students environment. It can be automated too.
- Login procedure could be password enabled.
- An option to approve/disapprove a collaboration request.
- Interface it with other J-DSP versions. For eg., J-DSP Sensor motes version, DSP boards version.

Ira A. FULTON School of Engg.

References

J-DSP Editor

1. G. C. Orsak and D. M. Etter, "Collaborative signal processing education using the Internet and MATLAB," *IEEE Signal Processing Mag.*, vol. 12, pp. 23-32, Nov. 1995.
2. A. Spanias and V. Atti, "Interactive on-line undergraduate laboratories using J-DSP," in *IEEE Trans. on Education Special Issue on Web-based Instruction*, vol. 48, no. 4, pp. 735-749, Nov. 2005.
3. A. Spanias, R. Chilumula *et al*, 'Multi-University Development and Dissemination of Online Laboratories in Probability Theory, Signals and Systems, and Multimedia Computing' in *IEEE Proceedings of Frontiers in Education (FIE-2005)*, Oct. 19-22, Indianapolis.
4. A. Spanias and R. Chilumula, 'A Collaborative Project on Java-DSP Involving Five Universities' in *ASEE 2006*, June 18-21, Chicago.
5. A. Spanias, R. Chilumula and C. Huang, 'Collaborative Signals and Systems Laboratories at ASU, UWB, UCF, UTD, and URI' in *IEEE Proceedings of Frontiers in Education (FIE-2006)*, Oct. 28-31, San Diego.
6. A. Spanias, and F. Bizuneh, 'Development of new functions and scripting capabilities in Java-DSP for easy creation and seamless integration of animated DSP simulations in web courses,' in *Proc. of 2001 IEEE Int. Conf on Acoustics, Speech, and Signal Processing*, May 2001, Salt Lake city, Utah.