MATLAB INTERFACE WITH JAVA SOFTWARE

Andreas Spanias, Constantinos Panayiotou, Thrassos Thrasyvoulou, and Venkatraman Atti
MIDL, Department of Electrical Engineering
Arizona State University, Tempe, AZ 85287

Abstract

The J-DSP editor is an object oriented environment that enables distance learning students to perform on-line laboratories. The editor has a rich collection of signal processing functions and is currently being used in a senior-level DSP course at ASU. In this paper, we present new enhancements to the infrastructure of J-DSP that provide embedded MATLAB™ scripting capabilities. The synergy of the J-DSP object-oriented environment with MATLAB programming enables students and instructors to exchange data and perform DSP simulations on both platforms. The advantage here is that complex algorithmic programming can be done visually on the internet using J-DSP and then executed in MATLAB. Conversely MATLAB programs can be mapped to flowchart-like diagrams and run in J-DSP. Although *Simulink* does the latter as well, the J-DSP tool runs on any platform requiring only a Java-enabled browser. Moreover the Java software integrates seamlessly with web content and animations supporting internet courses. The MATLAB scripts are generated with a new interpreter that has been developed for J-DSP. This interpreter encodes all simulation parameters in a script that contains the equivalent MATLAB code. When the generated script is loaded through the MATLAB editor (M-editor) the user can reproduce the J-DSP simulation in the MATLAB environment. It is also noted that there is a provision in J-DSP for generating HTML embeddable scripts that allows the user to embed simulations in web content. These synergies of MATLAB-JDSP-HTML can be very useful not only for students but also instructors. We have used embedded HTML scripts in our web course called MATLAB for DSP applications.

"Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition

1. Introduction

The MIDL has developed and evaluated an exemplary laboratory tool, the Java-DSP (J-DSP), for use in undergraduate and graduate courses such as digital signal processing (DSP), communication systems, controls, multidimensional DSP, and time-frequency representations[3]. The J-DSP is an object-oriented simulation editor written as a Java applet and provides hands on experiences for distance learning and on-campus students. The environment provided by the J-DSP editor allows students to establish and execute educational lab simulations from any computer that has Java-enabled browser. Furthermore, it comes with various features such as the ability to export a particular flowgram into different formats.

An existing feature of J-DSP was the provision of HTML embedded *scripting* capabilities[1]. This feature was developed and integrated into J-DSP to enable users to perform interactive simulations through HTML code. The *J-DSP scripts* allow instructors to construct and integrate animated visualization modules using high-level J-DSP scripts.

The newly proposed and developed feature of J-DSP, described in this paper, is the provision for MATLAB embedded scripting capabilities; it allows users to "translate" the J-DSP simulation flowgram into MATLAB code. Moreover, it enables users to repeat, verify and more importantly expand J-DSP simulations in the MATLAB environment. MATLAB provides a vast selection of additional functions that currently do not exist in the object-oriented environment of J-DSP. Furthermore, the synergy of the Java applet environment with MATLAB programming enables students and instructors to exchange data and perform DSP simulations on both platforms. In other words, this new feature acts as an *interface* between the MATLAB and the J-DSP object-oriented environment. This paper is dedicated to describe this new capability of J-DSP and provides a few examples how users can export MATLAB scripts from J-DSP flowgrams.

This paper consists of four sections that further discuss J-DSP's MATLAB script capabilities. Section 2 describes how to create MATLAB scripts through J-DSP simulations and provides a four-step procedure for script generation. Section 3 illustrates a script generation example in J-DSP. Section 5 elaborates on the current state of the MATLAB script development in J-DSP and

section 6 provides a number of suggestions regarding the expansion of J-DSP's MATLAB script capabilities.

2. Generating MATLAB scripts from J-DSP simulations

The following section provides an insight how to "compile" J-DSP flowgrams into MATLAB code. J-DSP is essentially fitted with an interpreter designed to encode all simulation parameters in a MATLAB™ m-file, which when loaded through the M-editor can run simulation in the MATLAB environment. The purpose of this new J-DSP Editor capability is to allow users to verify and/or expand their own J-DSP simulation examples in MATLAB. In addition, they can use the graphics of MATLAB and its real-time extensions.

The procedure for the script generation is relatively simple. First, a user creates the desired flowgram using the familiar drag-and-drop procedure of the editor as described in [1,2,3]. Second, by selecting "*File*" then "*Export as Script*" and then "*MATLAB™ Code*", the user obtains the script with all the J-DSP simulation parameters, ready to use in the M-editor. To provide details on script generation with the aid of graphical illustrations the following paragraphs describe a four step process.

2.1 M-Script generation in four steps.

The following paragraphs provide step-by-step instructions on how to generate a simple MATLAB script from a J-DSP simulation.

Step1: The user must start the J-DSP Editor and create the desired flowgram in the simulation area. Then, the participant must set the desired parameters in all blocks for a complete simulation.

Step 2: Next, from the main menu, the user selects "*File*" and then "*Export as Script*" as shown in Figure 1. The "*Script Export*" dialog window appears and the user must choose now the format in which to export the current simulation set-up. This will be referred to as the J-DSP script window. From the J-DSP script window the user must select the "*MATLAB(TM) code*" from the "Copy and paste this code:" drop down menu to obtain the MATLAB code describing the simulation, as shown in figure 1. In the J-DSP script window the user is also capable of choosing to export the simulation in HTML format for online demonstrations. For HTML scripts generation in J-DSP refer to [1].

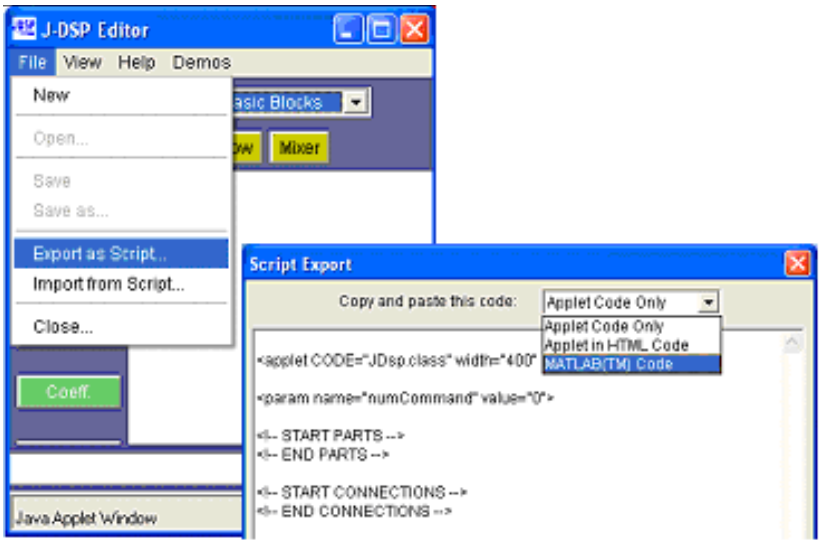

Figure 1: From the J-DSP script window we select *MATLAB(TM) Code*

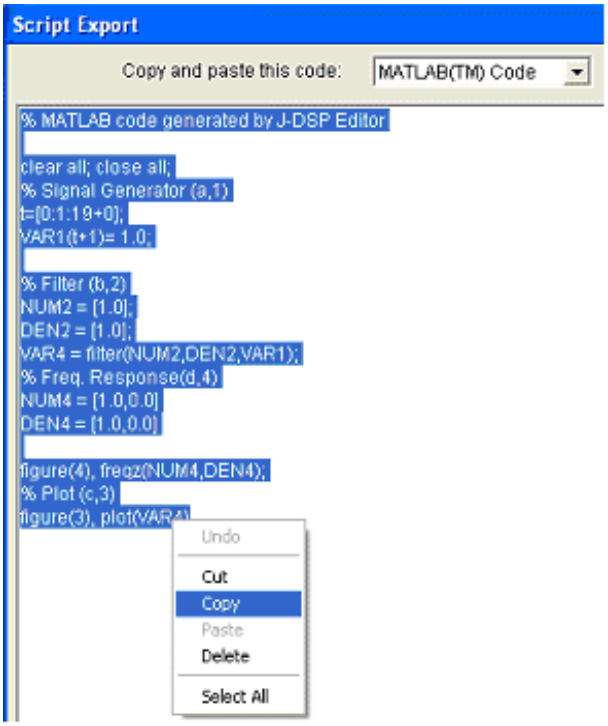

Figure 2: Selecting the generated MATLAB script

Step 3: The user must select now the code from the J-DSP script window either by pressing *Ctrl+C* to copy the code into the clipboard or by right clicking. The latter procedure is illustrated in Figure 2. Non-Windows™ users should be able to follow a similar procedure. If the participant is unable to use the *Ctrl+C* option to copy the generated script, installation of the

latest Java virtual machine of Sun found at http://www.java.com might be necessary. Instructions are also provided in the troubleshooting section of the J-DSP web site located at http://jdsp.asu.edu.

Step 4: Finally, the user must run the MATLAB program (version 6.1 R.12 or higher) and open a new M-editor window (or else called M-file). Then, the participant using either the mouse (right click and then Paste) or pressing *Ctrl+V* needs to paste the J-DSP's exported script in the already opened M-editor window and save the M-file under a useful name, illustrated in Figure 3. To run the simulation the user must either click on the "*Run*" button or press "F5".

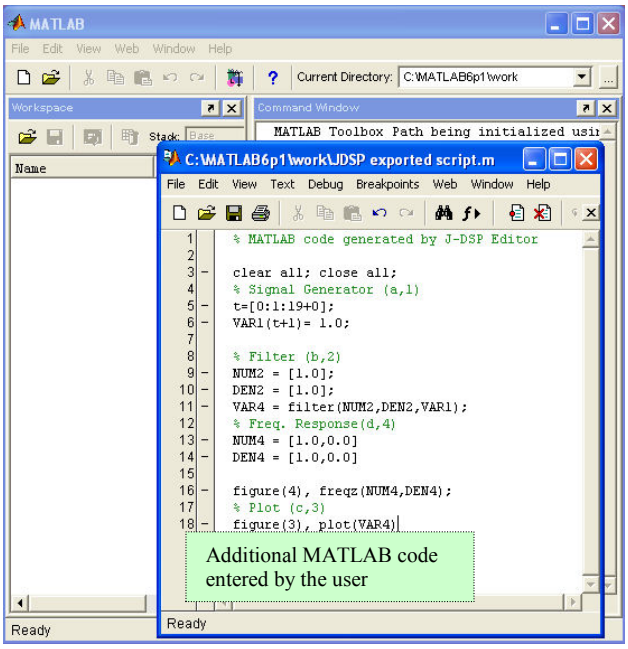

Figure 3: Paste the J-DSP generated script in an M-file

3. Example of a J-DSP simulation exported as a MATLAB script

Figure 4 illustrates a simulation flowgram in the J-DSP environment and the corresponding generated MATLAB script. The simulation involves the addition of two sinusoids generated by two signal generators ***Sig.Gen*** SIN-1 and ***Sig.Gen*** SIN-2 and the result is illustrated in the frequency domain using the ***Plot*** block. Figure 5 depicts the result of the same simulation in MATLAB followed by copying and pasting the generated script in the M-editor window saved as *JDSPexportedscript.m*.

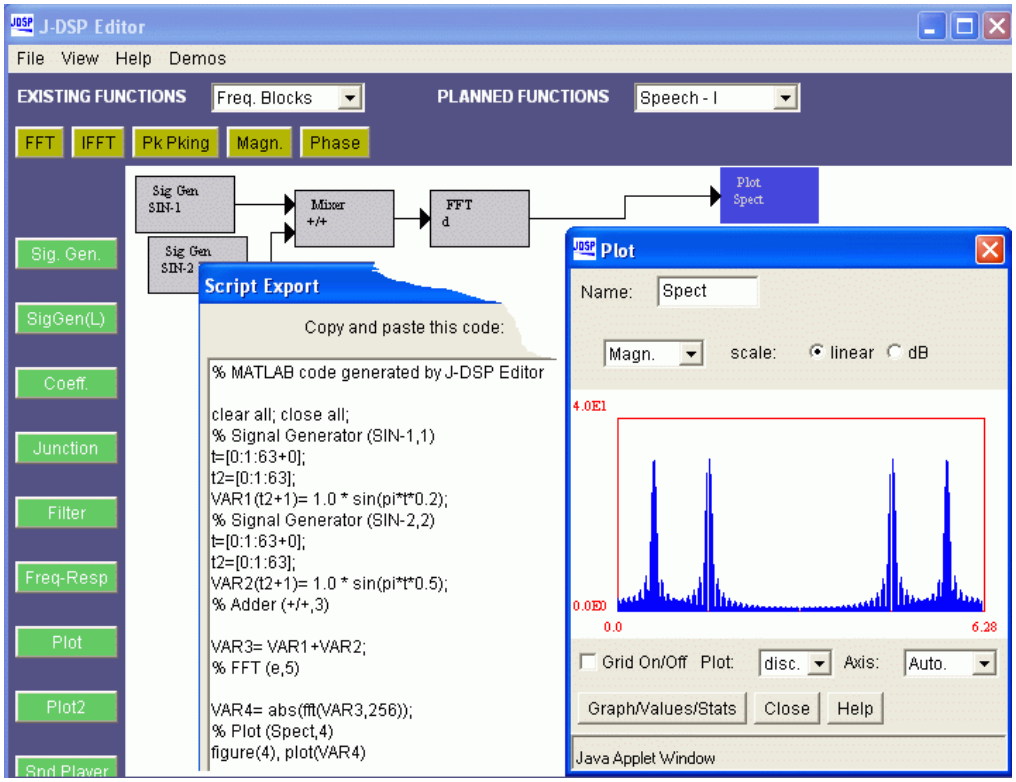

Figure 4: J-DSP simulation and the generated MATLAB script

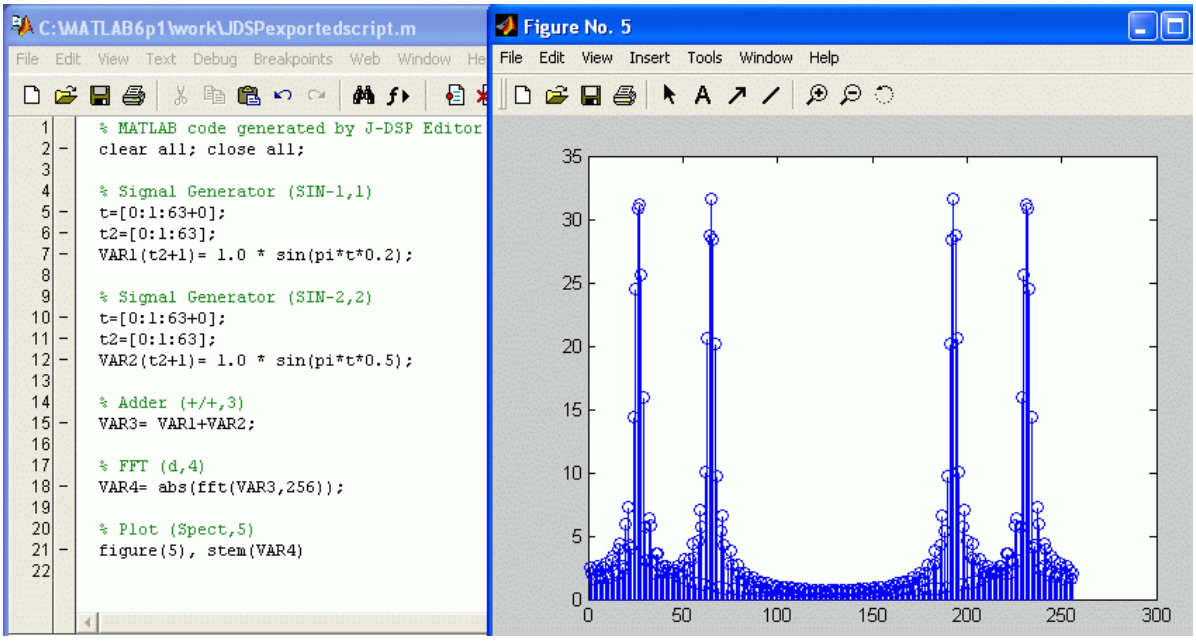

Figure 5: MATLAB simulation created by the J-DSP exported script

The next section discusses the development of J-DSP blocks with MATLAB script support and provides some of the MATLAB commands that were used in those blocks.

4. J-DSP functions with MATLAB script support

This section refers to specific J-DSP functions that have embedded the MATLAB script capability and point out their corresponding MATLAB command used for script generation.

	J-DSP Block	Simulated function	Samples of Java code with MATLAB commands used for script generation	J-DSP Variables
1	***Sig.Gen.***	Delta	Mcode1= "t=["+timeShift+":1:"+L+"+"+timeShift+"];" Mcode2 = amplitude+"*[1, zeros(1,"+L+");";	timeShift, amplitude, L
2	***Autocorr.***	Biased based autocorrelation	Mcommand = "\nVAR"+OUT+"= **xcorr**(VAR"+IN+",round("+lag+"/2),'biased') ;"	lag, VARIN
3	***Plot***	Plots continuous graphs	Mcommand= "figure("+Blockname+"), **plot**(VAR"+IN+")";	VARIN
4	***Plot***	Plots discrete graphs	Mcommand= "figure("+Blockname+"), **stem**(VAR"+IN+")";	VARIN
5	***Freq.Resp.***	Plots the frequency response	Mcommand= "figure("+Blockname+"), **freqz**(NUM"+Blockname+",DEN"+Blockname+");";	NUM, DEN
6	***LPC***	Extracts the LPC coefficients	Mcommand= "VAR"+OUT+"= **lpc**(VAR"+IN+","+lpc_size+");";	VARIN
7	***Window***	Rectangular	Mcode1= "**window**(**@rectwin**,"+winlength+");";	winlength
8	***Window***	Blackman	Mcode1= "**window**(**@blackman**,"+winlength+");";	winlength
9	***Window***	Kaiser	Mcode1= "**window**(**@kaiser**,"+winlength+","+winbeta+");";	winlength, winbeta
10	***D-Sampling***	Down sampling	Mcommand ="VAR"+OUT+"= **downsample**(VAR"+IN+","+M+");";	VARIN, M
11	***Convolution***	Convolution	Mcommand= "\nVAR"+OUT+"= **conv**(VAR"+IN1+", VAR"+IN2+");";	VARIN1, VARIN2
12	***FFT***	FFT	Mcommand= "\nVAR"+OUT+"= abs(**fft**(VAR"+IN+"," +fft_size+"));";	VARIN, fft_size
13	***PZ-plot***	Plots poles & zeros of transf. functions	Mcommand="\n **zplane**([ZerosX"+Blockname+"+i*(ZerosY"+ Blockname +")]', [PolesX"+ Blockname +"+i*(PolesY"+ Blockname +")]');";	ZerosX, ZerosY, PolesX, PolesY

Table 1: List of MATLAB commands used in some of the Java modules

The most frequently used basic, arithmetic, frequency and filter blocks are "equipped" with MATLAB scripting support. Table 1 lists some of the J-DSP blocks with script support and their related Java code that generates a long string that builds-in the MATLAB script. The MATLAB commands used in the Java code are highlighted in **bold faced** in the third column; the fourth

column lists the J-DSP variables entered either by the user or assigned automatically by the editor during a simulation. Future work would be required to modify all the J-DSP blocks to acquire this new functionality because script generation is not available for a number of blocks that belong in the statistical, speech processing, communication systems, and controls blocksets.

5. Conclusions and future work

The synergy of the J-DSP object-oriented environment with MATLAB programming enables students and instructors to exchange data and perform DSP simulations on both platforms. The advantage here is that complex algorithmic programming can be done visually on the internet using J-DSP and then executed in MATLAB to exploit its speech, graphics, and real-time interface capabilities. Furthermore, this functionality enables students to perform simple simulations in the user friendly environment of the J-DSP editor and use the interface to verify, and more importantly to expand the simulations in MATLAB. The J-DSP-MATLAB scripting capabilities are being explored to teach students MATLAB programming for DSP applications.

A number of improvements can be made to this new J-DSP capability. First, the generated scripts can be modified so that when the simulation runs in MATLAB the plots start from "0" and not from "1". Specifically the generated scripts might be adjusted to follow MATLAB's convention that the index of vectors starts from "1" and not from "0" like in Java. Second, all J-DSP blocks may acquire scripting capabilities similar to those described in Section 4. Finally, an expansion of the exported script is suggested that will allow students to repeat the same J-DSP simulation in the Simulink environment.

References

[1] A. Spanias, and F. Bizuneth, "Development of new functions and scripting capabilities in Java-DSP for easy creation and seamless integration of animated DSP simulations in web courses," IEEE ICASSP '01, vol. 5, 7-11, pp. 2717-2720, May 2001.

[2] A. Spanias et. al., "Development and evaluation of a Web-based signal and speech processing laboratory for distance learning," IEEE ICASSP '00, vol. 6, 5-9, pp. 3534-3537, June 2000.

[3] A. Spanias, et al, "On-line Laboratories for Speech and Image Processing and for Communication Systems using J-DSP", Proc. of IEEE, 10th Digital Signal Processing Workshop, Callaway Gardens, Pine Mountain, Georgia, USA, Oct, 2002.