J-DSP-CONTROL: A CONTROL SYSTEMS SIMULATION ENVIRONMENT [+]

T. Thrasyvoulou, K. Tsakalis and A. Spanias
MIDL
Department of Electrical Engineering
Arizona State University, Tempe, AZ 85287-7206
thrassos@asu.edu, tsakalis@asu.edu, spanias@asu.edu

ABSTRACT

J-DSP-C is a java-based object-oriented programming environment that was developed at Arizona State University for use in control systems education. This environment enables the simulation of dynamical systems on-line from any computer system equipped with an Internet browser. The J-DSP-C features are primarily aimed to provide an on-line laboratory experience to distance learning students. Combined with a report submission and scoring facility, J-DSP-C integrates interactive examples into web content for education and demonstration purposes.

KEYWORDS

Simulation, Education, Control Systems.

1. INTRODUCTION

This paper presents a prototype laboratory software tool, called Java-Digital Signal Processing - Control (J-DSP-C). J-DSP-C can be used to simulate control systems on-line, taking advantage of recent developments in Internet technology. The main motivation behind its development stems from the recent increase of the percentage of distance-learning students at Arizona State University (ASU) and other major metropolitan universities. Distance learning is a trend in today's education, and J-DSP-C fills a gap by providing a simple yet powerful tool for on-line control systems simulations. The software is freely distributed and is designed to enhance distance-learning and continuing education with an on-line laboratory experience. Its precursor is J-DSP that was developed in the ASU Multidisciplinary Initiative on Distance Learning (MIDL) laboratory. J-DSP was successfully tested in ASU's undergraduate digital signal processing class. Based on this experience, further enhancements of this tool are currently under development in MIDL, for use in controls, communications, and image processing education. The control systems extension J-DSP-C will soon be tested for use in undergraduate control classes.

Similar to its precursor, J-DSP-C is an object-oriented simulation environment that enables students to establish and execute control systems simulations from any computer equipped with an Internet browser. All functions in J-DSP-C appear as graphical blocks that are accessed through pull-down menus and are grouped according to their functionality. The J-DSP-C editor allows the user to graphically setup and simulate systems with arbitrary interconnection topology. An example of the J-DSP-C graphical user interface (GUI) is illustrated in Figure 1. The following sections provide an overview of the J-DSP prototype, the J-DSP-C enhancements and limitations, its current functionality, and examples of use.

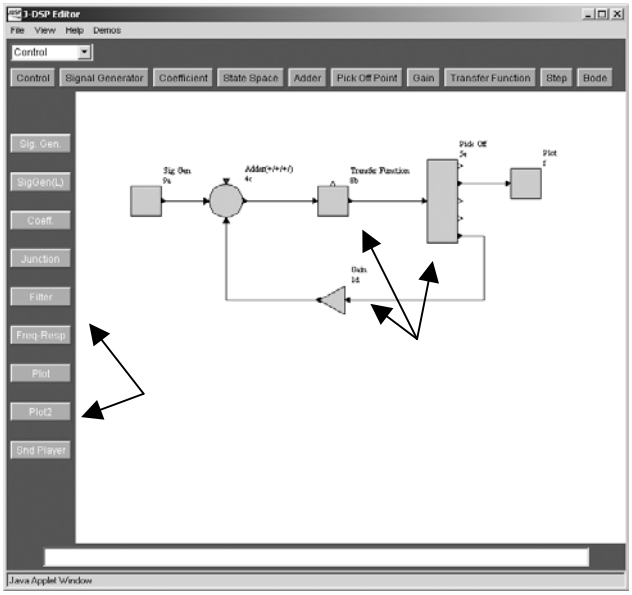

Figure 1: J-DSP-C user interface

Details on J-DSP and J-DSP-C are given in [1] and in the on-line documentation at http://jdsp.asu.edu [3]. Funding for the development of J-DSP and J-DSP-C was provided by the NSF CCLI program and includes

[+] Supported in part by NSF CCLI grant DUE 0089075. J-DSP concept by A. Spanias. For more information on J-DSP and its dissemination please contact spanias@asu.edu.

software development and dissemination along with a series of on-line laboratory exercises.

2. J-DSP OVERVIEW

The J-DSP editor is the basis on which J-DSP-C was developed. J-DSP was originally developed as a platform to simulate the typical systems and operations encountered in digital signal processing. Its underlying philosophy was dictated by its suitability for operation over the Internet. Using object-oriented programming principles, J-DSP proved to be a successful tool to address the typical computational and visualization needs of DSP courses. The existing functionality includes basic filter design, fast Fourier transforms (FFT), upsampling, downsampling, signal generation and plotting. More advanced functions include autocorrelation, various types of periodograms and correlograms and AR time-series. (See [1,3] for details.)

To address the needs of other DSP-related problems, the J-DSP editor is currently being expanded to offer new specialized functionality. Several new functions are being developed within this framework to support experiments on speech analysis-synthesis, time frequency representations, image processing, and communications systems [2].

On the other hand, enabling the support of experiments in feedback systems required more substantial modifications. A key characteristic of J-DSP was its sequential processing of information. Once a block is introduced in the editor, its output is immediately computed based on the input it receives. Although this is an attractive feature, especially for educational examples, it cannot accommodate general interconnection topologies, such as feedback systems. Addressing this issue, J-DSP-C represents a significant enhancement of the basic J-DSP engine, while it preserves the same fundamental object-oriented structure and most of the already developed infrastructure.

3. THE J-DSP-C ENVIRONMENT

The handling of feedback and, in general, arbitrary interconnections of blocks required some major changes in the J-DSP editor infrastructure. Our objective during this development was to preserve the original structure as much as possible together with its underlying concept. That is, the J-DSP-C development adheres to the same simplicity, compactness, and object-oriented philosophy of its precursor. This resulted in some compromises in terms of generality and computational efficiency. However, in its primary mission as an educational tool, the J-DSP-C limitations are not too severe and can usually be circumvented by a careful planning of the simulation experiments.

The modifications of the basic editor can be divided into two categories. One is the enhancement of the GUI interface, together with the associated changes in the object classes. The other is the modification of the execution procedure to enable the simulation of feedback loops.

In the original J-DSP, blocks were rectangular and had two ports, the left one designated as an input port and the right one as output. Two additional ports top/bottom were available for exchange of parameters. This configuration was adequate to represent the usual DSP operations but too inflexible for feedback systems. In order to achieve consistency with the standard notation and appearance of feedback control systems, the J-DSP-C GUI and the editor can now handle blocks with multiple input/output ports that are not limited to rectangular shapes. For example, a summation node can now appear as a circle with several input ports, while a gain block is shown as a triangle. The new environment also allows blocks to be rotated and flipped and the connections to be edited or modified. This capability is quite essential in maintaining a "clean" visual appearance of more complicated system interconnections. The differences between the GUI capabilities of the J-DSP and J-DSP-C editors are illustrated in Figures 2 and 3.

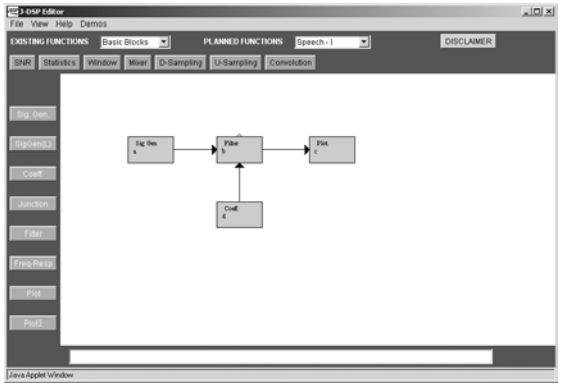

Figure 2: J-DSP user interface

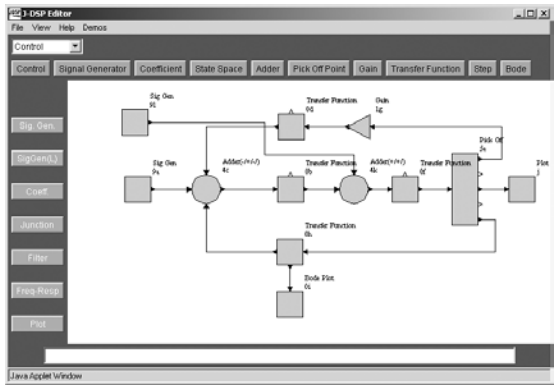

Figure 3: J-DSP-C user interface

A more extensive modification of the original J-DSP engine was necessary to enable the simulation of feedback systems. The class of control blocks was expanded to include state-space descriptions of dynamical systems. All blocks now have a state attribute (trivial for memoryless blocks) to enable the recursive computation of the system response. For simplicity, discrete-time approximations are used internally to compute the response of continuous-time systems. This conversion is transparent to the user, but care should be exercised in the selection of the sampling time so that the discretization is reasonably accurate. (A warning is issued when the sampling time seems too large.) In this setting, the computation of the response of the system interconnection is computed recursively in time and iteratively with respect to the various blocks.

The recursive portion of the solution is made possible by using state-space descriptions. At each time instant, the solution can be advanced by one time step and only the state vector needs to be stored in memory. The iterative portion of the solution is required to ensure that the correct inputs are computed for all blocks. That is, at each time instant, the input/output computations are iterated until they converge, before updating the states. This simplified approach is compatible with the object-oriented definition of the various blocks. However, a subtle point and a key limitation is that convergence is not necessarily guaranteed for any system interconnection. It can be shown that:

- When a feedback loop has no algebraic part (at least one of the systems has no direct throughput) then the iteration convergence in finite steps, proportional to the number of blocks in the loop.
- When a feedback loop has an algebraic part then the iteration converges exponentially as $(1-\rho)^k$, where ρ is the loop direct throughput, provided that $\rho < 1$. Otherwise, the iteration diverges.

In practice, this limitation is not very restrictive but it should be obeyed when defining the feedback interconnections. (Again, a warning is issued when the local iteration fails to converge in a prescribed number of steps.)

Finally, to avoid repeated unnecessary computations while editing the system interconnections, a simulation button has been introduced to disable J-DSP's automatic block execution. Instead the simulation computations are performed on-demand by pressing this button.

4. J-DSP-C FUNCTIONALITY

A fundamental set of functions has been developed in order to accommodate the need for control systems simulations. The currently available blocks are briefly described below, grouped in terms of their fundamental properties.

4.1. Signal generators

Signal generators are an essential part of every simulation. The J-DSP editor has been fitted with two signal generators, providing a variety of signals. The first signal generator supplies a simple step signal that is encountered most frequently in control systems simulations. It is simple to use and within easy reach. The second signal generator has been designed to offer a more elaborate selection of signals. Among others, this block provides discrete impulses, sinusoids, sinc functions, random signals with either uniform, Gaussian, or Rayleigh distributions and exponential signals. Where applicable, signals can be chosen to repeat periodically.

4.2. Memoryless Systems and Arithmetic operations

These include summation nodes, gains and various other blocks performing arithmetic operations. For example, the user can multiply two signals, compute their exponential or their natural logarithm.

4.3. Dynamical Systems

Currently this class contains linear dynamical systems that can be specified in terms of their transfer function or their state-space description. The transfer function block is used to enter a rational transfer function describing a system. More precisely, this block simulates a system given a transfer function in the form

$$H(s) = \frac{\sum_{i=0}^{n} b_i s^i}{\sum_{i=0}^{n} a_i s^i}$$

The numerator and denominator coefficients b_i and a_i are entered in the block's dialog box, shown in Figure 4.

An alternative and more general way to specify a linear system is with its state-space description. This block implements the equation

$$\dot{\underline{x}} = A\underline{x} + Bu$$
$$y = C\underline{x} + Du$$

where, $A \in R^{nxn}, B \in R^{nxm}, C \in R^{pxn}, D \in R^{pxm}$, and $x \in R^n, u \in R^m, y \in R^p$. $\underline{x}$ is the state vector, u is the system input and y is the system's output response. Currently, the maximum number of states is limited to 10. The dialog window for this block is shown in Figure 5, along with a window for entering a matrix. Notice that the user can select to enter matrices in canonical or other

common forms, where many of the entries have fixed values that are automatically initialized.

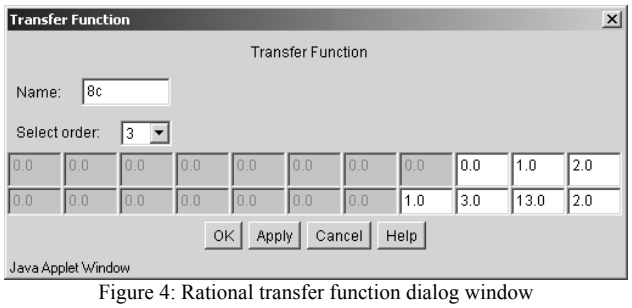

Figure 4: Rational transfer function dialog window

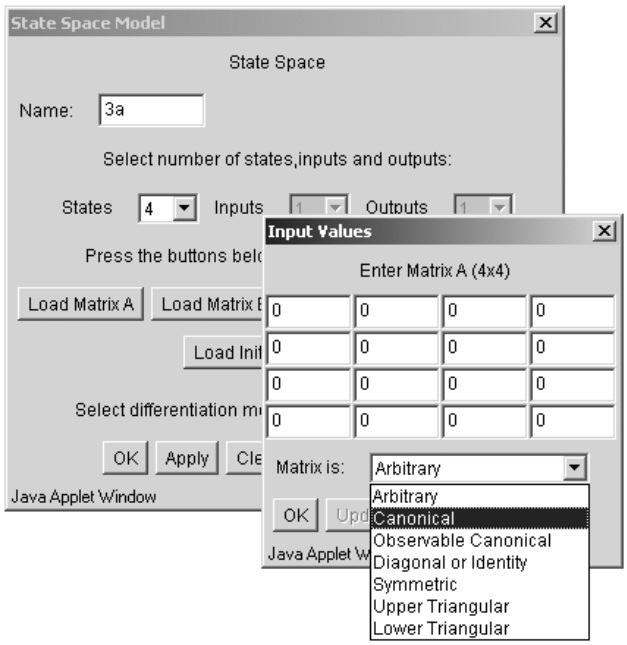

Figure 5: State Space system's dialog window

4.4. Plotting and Visualization Blocks

The results of a simulation can be examined using the graphical output capabilities of the Plot block. This block simply plots its input in a linear or logarithmic scale. It has zooming capabilities and can provide statistical properties of the displayed signal. In the same family, the Bode and Nyquist plot blocks can be used to visualize system properties and aid the design of control systems. The Bode plot displays the magnitude and phase of the system transfer function (see Figure 6), while the Nyquist block plots real versus imaginary parts.

4.5. Blocks Under Development

Future enhancements of the J-DSP-C editor will include additional components to facilitate the modeling and simulation of nonlinear systems (e.g., inverted pendulum and other mechanical systems). Additional blocks will perform matrix manipulations and least-squares approximation, enabling the implementation of adaptive systems. Other enhancements include the support of systems with multiple inputs and outputs, and the ability to group several blocks into a composite one. Notice that the ability to change the block shape and number of input/output ports has been introduced with this goal in mind.

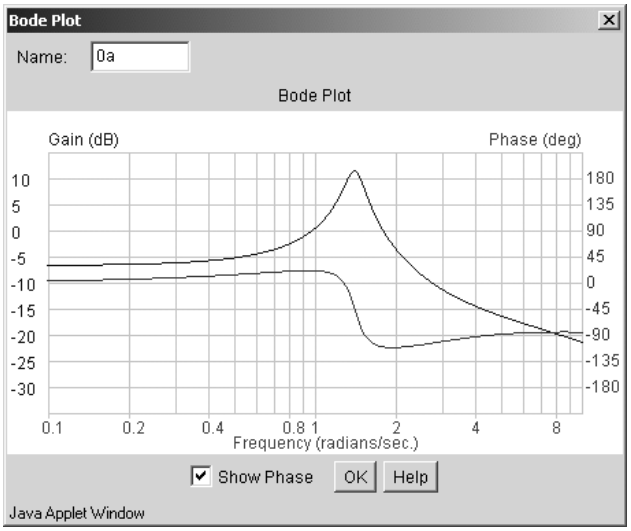

Figure 6: Bode plot

5. EXAMPLES

In this section, two illustrative examples of J-DSP-C simulations are presented. The first example simulates a simple unity feedback system with

$$H(s) = \frac{s^2 + 4s + 5}{s^3 + 6s^2 + 3s + 10}$$

The block diagram and the plot of the simulation result for a step reference input are shown in Figure 7.

In the second example, the feedback system contains two transfer functions, one in the forward path and the other in the feedback path. Here,

$$H_1(s) = \frac{s + 2}{s^3 + 3s^2 + 13s + 2}$$

$$H_2(s) = \frac{s^2 + 4s + 5}{s^3 + 6s^2 + 3s + 10}$$

For this case, the block diagram and the plot of the simulation result for a step reference input are shown in Figure 8.

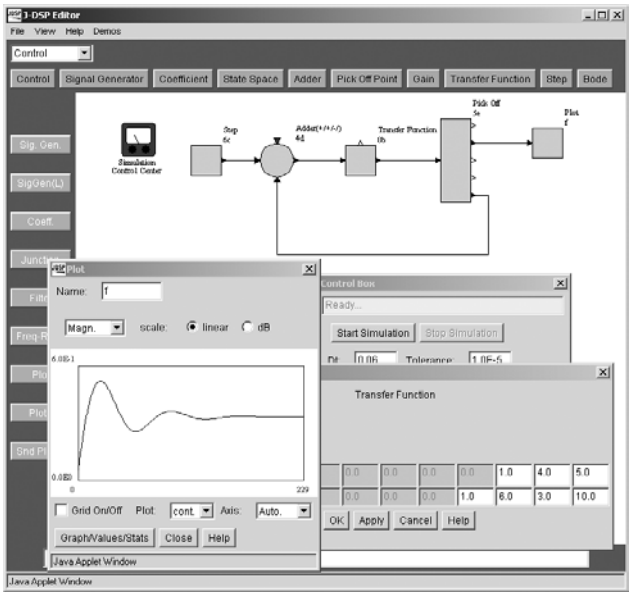

Figure 7: Block diagram and simulation results for example 1.

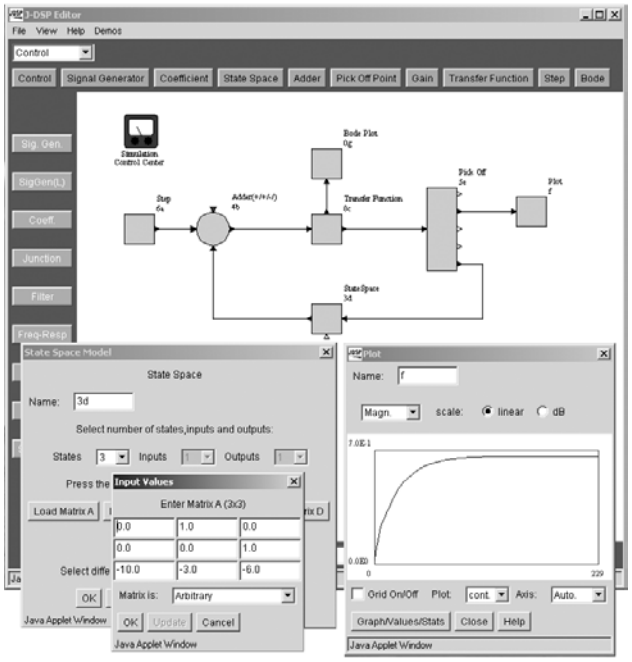

Figure 8: Block diagram and simulation results for example 2.

6. CONCLUSIONS

This paper presented the various enhancements and modifications of the J-DSP software tool for use in control systems education. This new version, J-DSP-C, is an object-oriented programming environment, that enables the simulation of feedback systems in a straightforward and easy to comprehend manner. The feedback system is defined through a GUI editor by connecting blocks together, maintaining a classical textbook appearance.

Although it lacks the power, efficiency, and generality of mainstream commercial software like MATLAB/Simulink™, J-DSP-C is easy to use and can provide a platform with sufficient flexibility to simulate the typical exercises found in control systems education. Its primary use is envisioned as a distance learning tool because of its ability to run over the Internet through a simple web browser. Furthermore, with its open architecture, J-DSP-C is continuously improved with respect to the collection of blocks available for use in simulations. Combined with a report submission and scoring facility, J-DSP-C integrates interactive examples into web content to improve the quality and effectiveness of control systems education.

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] A. Spanias et al, "On-line laboratories for speech and image processing and for communication systems using J-DSP", to appear at *2nd DSP-Education workshop*, Pine Mountain GA, Oct 13-16, 2002.

[3] J-DSP on-line help, http://jdsp.asu.edu