INTRODUCTION TO SIGNAL PROCESSING USING THE SPC TOOLBOX

Monique P. Fargues and Dennis W. Brown

Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943-5121
e-mail: fargues@ece.nps.navy.mil

Abstract -- We discuss an interactive software developed to complement the teaching of digital signal processing (DSP) concepts in various courses taught by the Electrical and Computer Engineering Department at the Naval Postgraduate School. The SPC software was designed to facilitate the implementation of signal processing concepts learned in the classroom and to illustrate their advantages and drawbacks. The MATLAB® Version 4 environment is used to take advantage of the graphical user interface capabilities. SPC is a window-based, user-friendly tool that allows the user to develop digital filters, analyze signals, and easily design basic signal modeling tools.

I. INTRODUCTION

Digital Signal Processing (DSP) is a continuously changing filed whose emphasis has increased tremendously in academic institutions in the last ten years. While DSP is quite important in electrical engineering education, the mathematical concepts used can be rather difficult when presented in a classical classroom environment. These concepts become more meaningful when they are actually applied to specific problems. Thus, the last few years have seen numerous innovative proposals to emphasize applications along with theory and to provide a better "hands-on" approach to the teaching of DSP concepts [1,2].

Students studying at the Naval Postgraduate School (NPS) have between twelve months to two and a half years to complete a Masters' Degree depending on their undergraduate degrees. The average student pursuing a Master degree at NPS is very different from the average graduate student studying at a civilian U.S. university; he/she is usually older than the "typical" graduate student, has a significant amount of work experience, but has been out of school for as much as eight years. In addition, his or her undergraduate major may be in a discipline other than electrical engineering. As a result of the very diversified backgrounds, NPS students favor hands-on approaches to problems, which allows them to better visualize what theoretical concepts mean and how they can be applied in practical applications. Therefore, courses taught in the Digital Signal Processing track in the Electrical and Computer Engineering Department at the Naval Postgraduate School make heavy use of projects and computer-based assignments in an effort to better relate theory and applications.

The initial motivation behind the development was based on the following three observations; 1) similar software is often required to be written by students in various courses (i.e, there is a significant software overlap); 2) software written under these circumstances is usually case specific (i.e., even though the DSP technique implemented may be quite generic, the resulting software has only been thoroughly tested for a very specific application, and may or may not be "valid" for a different application); 3) "home-coded" software written in early courses is sometimes not properly commented, which significantly reduces the chances of the code being used in another course. Thus, in summary, students potentially "waste" a significant amount of time writing and rewriting similar software for different courses which build on each other. For example, parametric modeling techniques covered in the *Statistical Signal Processing* course are used in *Speech Signal Processing* and *Spectral Estimation* courses taught at NPS. SPC was designed to decrease the need to code some of the basic DSP and communications techniques, and as a result to decrease the time needed for programming. This decrease should allow students to focus more on understanding the methods to be analyzed and to tackle more challenging projects.

SPC is a MATLAB-based set of individual tools which are linked together so that different operations can be cascaded. MATLAB version 4.2 and its Signal Processing Toolbox are required to run the software to its full capabilities. SPC functions can be divided into five main groups: time-domain signal analysis, spectral estimation, filtering, signal modeling, communication signal generation, and graphical tools. In addition, SPC makes extensive use of the graphical user interfaces (GUI)

The views expressed in this document are those of the authors and do not reflect the official policy or position of the U.S. Government.

offered in MATLAB version 4. SPC uses pull-down menus, push buttons, edit boxes, pop-up menus, check boxes, scroll bars, and drag-and-drop cursor selection. This software allows the user to develop digital filters and to visualize their effects, to analyze speech signals, and to design basic parametric AR and ARMA techniques. In addition, numerous graphical tools are provided which allow the user to zoom in on some section of a given signal, do cut-and-copy operations, etc.... Section 2 presents an overview of the capabilities of the software. Next, section 3 presents a brief overview of the Runtime paradigm, a new programming paradigm that was developed during the development of SPC to allow the user to run multiple copies of the same tool. A few of the capabilities of SPC are illustrated in Section 4. Finally, conclusions are presented in Section 5.

2. SPC SOFTWARE OVERVIEW

The main capabilities of SPC are summarized below, and further details may be found in Brown [3]. A draft version of the software and user's guide may be obtained by anonymous ftp from ftp. nps.navy.mil. Get the file /ece/pub/spctools /spctools.tar. SPC can also be found on the anonymous ftp site at The Mathworks at ftp.mathworks.com. Get the files *pub/contrib/signal* /spctools.zip and spctools.sh.

The main features of SPC are described below. Note that the tools are linked so that operations from different tools may be cascaded easily.

a. SIGEDIT: provides tools which do the following operations on a given data set: cut, paste, copy, amplify sections, play (using the MATLAB soundtool command). All modifications can be done via mouse-driven commands applied directly to the plot of the data set.

b. SIGFILT: provides tools to filter a given data set. This tool uses the signal processing MATLAB toolbox to design various IIR and FIR filters. A window displaying the signal power information is opened, and the user can graphically select the cutoff frequency regions by dragging vertical cursors present in the plot. Once applied, the filtered signal power information gets displayed on top of the filter magnitude response. This capability allows the students to better visualize what the filtering effects are in the frequency domain.

c. VOICEDIT: provides tools which compute the zero-crossing rate (ZCR) and the short time energy (STE) of a given data set. The user can select window length, type, and overlap. Smoothing and median filters are available to smooth out the resulting ZCR and STE. A window displaying two different plots opens, the first one contains the ZCR and STE information, while the second one contains the original time signal. ZCR and STE curves can be used to isolate voiced and unvoiced sections of the time-domain signal. Vertical cursors can then be used to isolate and store separately the desired section of the signal.

d. SIGMODEL: provides tools which compute AR, MA and ARMA models obtained from a given data set. The available AR methods are the correlation, the covariance, the modified covariance, and the Burg method. Available MA methods are Prony's, Shank's, and Durbin's method. The user obtains ARMA modeling by selecting both AR and MA options. The tool can be used for synthesis operations, as the user has the option to select a white noise or periodic impulse train to send through the estimated model.

e. SPECT2D: provides tools which compute classical spectral estimation techniques such as the periodogram, the Daniell and the Welsh estimator. In addition, two subspace methods are available; minimum variance and MUSIC estimators.

f. SPECT3D: provides tools which can be used to generate a spectrogram. The user has full control of the time window length, type, and overlap. In addition, SPECT3D uses MATLAB graphical capabilities to plot mesh plots using various shading options. Note that, the three-dimensional mesh plot orientation may be changed easily using scrollbars added to the sides of the SPECT3D figure window.

g. GFILTERD: provides a graphical filter design tool. This tool was designed to allow students to visualize the effects of poles and zeros on a desired filter transfer function. A window displaying three plots is opened. The first graph is a polar plot of the complex plane. The second and the third plots show the magnitude response and phase of the filter frequency response obtained after placing poles and zeroes in the complex plane. Note that placement, addition, and deletion of poles and zeros can be entered on the complex plane graphically using the mouse. This versatility allows the student to visualize very easily what the effects of poles and zeros are on a filter transfer function.

h. ZOOMTOOL: provides graphical tools which allow the user to zoom in on certain sections of a plot. In addition, mouse driven horizontal and vertical cursors are available to the users to identify specific values on the plot. This tool is used by most of the other tools to provide zooming capabilities.

i. V3DTOOL: provides graphical tools which allow the user to reorient three dimensional plots easily. It can also be used to easily change color map, brightness and shading used to render the three dimensional plot.

j. Additional utilities: m-files supporting 8 bit .au, .wave, and .voc files have been added to import and export audio files in and out of the MATLAB workspace. In addition, several modulation/demodulation schemes for communication signals are available.

3. SPC PROGRAMMING PARADIGM

The development of the SPC toolbox included the development of a new MATLAB programming paradigm. This paradigm, referred to thereafter as the "Runtime" paradigm, was necessary to achieve the high degree of integration of separate SPC GUI tools and the ability to run multiple copies of the same tool.

Programming a GUI interface in MATLAB requires the ability to reference "handles" of GUI controls. A "handle" is best described as a pointer, and is very similar to the pointers found in C++. Referencing handles is necessary for finding and/or setting the current state of a control. Two program paradigms for keeping track of handles are provided by the MathWorks (TMW) [4]. The first TMW paradigm makes use of declared global variables for handle storage. The second one involves creating a vector of control handles and storing the resulting vector in the UserData property of the figure window the GUI interface is created in. This vector can be retrieved, when necessary, giving the programmer access to the desired handle. These paradigms are referred to as the "Global Variable" paradigm and "UserData" programming paradigm respectively. Both of these paradigms were deemed inadequate in developing the SPC toolbox for the following reasons:

1. Use of the Global Variable paradigm prevents running multiple copies of a tool since starting a second tool will destroy the stored handles to the first tool.

2. Although proper use of the UserData paradigm allows for multiple copies of a tool to be executed, remembering the index in the handle vector is cumbersome for the programmer. It should be noted however, that both paradigms do have a speed advantage over the Runtime paradigm about to be presented [3], and both are used in the SPC Toolbox in certain situations requiring this advantage.

Understanding the Runtime paradigm first requires an understanding of MATLAB Handle Graphic objects. All graphical objects in MATLAB, from figure windows to push button controls, are objects having parent-child (tree) structured relationships. Note that all figure windows have the same parent, the non-displayable "root" object, whose handle is always zero. In addition to object handles, understanding object properties is required for the Runtime paradigm. Object properties define the size and appearance of controls and the response of controls and other graphic objects to various stimuli (key strokes, mouse actions). Object properties are set when created or by using the set command. For example, the partial command

```
handle = uicontrol('Style', 'pushbutton','String','OK',....)
```

creates a push button control labeled "OK". Not all properties have to be set explicitly by the user. In the above example, the property 'Type' was automatically set to the value 'uicontrol' by the UICONTROL command. The GET command can be used to retrieve property values which can then be tested. For example, the commands

```
label = get(handle,'String')
if strcmp(label,'OK'),
... action ...
end
```

retrieve the value stored in the String property and perform "action" if the String property is equal to OK.

Two properties, Parent and Children, are key to the Runtime paradigm. As one would expect, the Parent property contains the handle of the parent object and the Children property contains a vector of handles to all the children of an object. By retrieving handles to the parent and children objects, the object tree can be traversed either up, down, or across. At this point, the Runtime paradigm can be defined. The basic idea behind this new paradigm is that any object, having a set of known object properties, can be found by traversing the object tree formed by the Parent and Children properties until the object has been found." That is, object handles can be found at Runtime, hereby, eliminating the need to store them.

In the previous example a push button control labeled 'OK' was used. When the user clicks on this push button, the figure window the push button is located in, becomes the "current" figure. Under the Runtime paradigm, the handle to the 'OK' push button is found by first getting the handle to the current figure window (the GCF command), retrieving the figure window children handles, and then searching through the children until the 'OK' push button object is found. Experienced MATLAB

GUI programmers may notice that the 'OK' push button object above can also be found by the "get current object" (GCO) command. However, assume now that an edit box in the same window is to be read after the user selects the 'OK' push button. At this point, since the edit box is not the current object, the GCO command is of no use. If the edit box had been uniquely identified by having previously set its UserData property to 'Input', its handle could have been found at Runtime by retrieving the figure window children and searching for the edit box with the UserData property set to 'Input'.

In the SPC Toolbox, the "find" functions support the Runtime programming paradigm. For instance, the command "h = findpush(gcf,'OK')" would retrieve the handle of the push button control in the previous example. Beginning with MATLAB version 4.2, The MathWorks included a built-in command to directly support the Runtime programming paradigm. The find object command (FINDOBJ) would be used as "h = findobj(gcf, 'Type', 'uicontrol', 'Style', 'pushbutton', 'String', 'OK')" in place of FINDPUSH.

4. APPLYING SPC TO DIGITAL SIGNAL PROCESSING

In this section, we illustrate how SPC can be used to present various DSP concepts. First, we will show how the GFILTERD tool may be used in filter design to teach students what the effects of changing pole and zero locations are on the overall filter transfer function. Next, we will illustrate how SPC can be used to graphically filter digitized signals using the SIGFILT tool. Finally we will briefly introduce the SIGMODEL tool which can be used to model signals using AR/MA/ARMA models.

a. Using GFILTERD: This tool provides an interactive, graphical environment to design digital filters through individual (complex-conjugate pair) placement of poles and zeros on a pole zero plot of the filter transfer function, as illustrated in Figure 1. Poles and zeros may be placed, moved, and deleted using the mouse or using keyboard commands.

b. Using SIGFILT: This tool can be used to apply filters to a digitized signal. Filter type and order can be selected using the pull-down menu labelled FILTER available at the top of this window, as illustrated in Figure 2. The desired order may be entered using the box option at the bottom of the window. Finally cutoff frequency values may be selected either by entering their values in the appropriate boxes, or by dragging the vertical cursors available in the frequency plot. After applying the filter, the graph contains the filter magnitude response, the resulting filtered signal frequency response, and the original signal frequency response. The resulting filtered signal can then be saved for further processing.

c. Using SIGMODEL: This tool consists of a set of pull down menus and a group of controls along with a graph of the signal under study. Using this tool, the user can designate a portion of the signal to be used for estimating the model using graphical cursors, and designate the length of the modeled signal. The ability to set-up these parameters are quite useful in speech signal processing applications when developing models for a single pitch period based on using data from several pitch periods.

5. CONCLUSIONS

SPC is a user-friendly window-based interactive MATLAB-based software which was originally designed to assist students in the application of DSP concepts learned in the classroom. The graphical capabilities present in SPC facilitate the analysis of data and decrease the programming required for students. We illustrated a few of the software capabilities in filter design and ARMA modeling applications.

REFERENCES

[1] Special Session on Signal Processing Education, *IEEE ICASSP-94 Proceedings*, Vol. 6, April 1994.

[2] Special Session on Signal Processing Education, *IEEE ICASSP-93 Proceedings*, Vol. 1, April 1993.

[3] D. W. Brown, *Computer Tools for Digital Signal Processing*, MSEE Thesis, Naval Postgraduate School, to be completed.

[4] *Building a Graphical User Interface*, The MathWorks Inc., 1993.

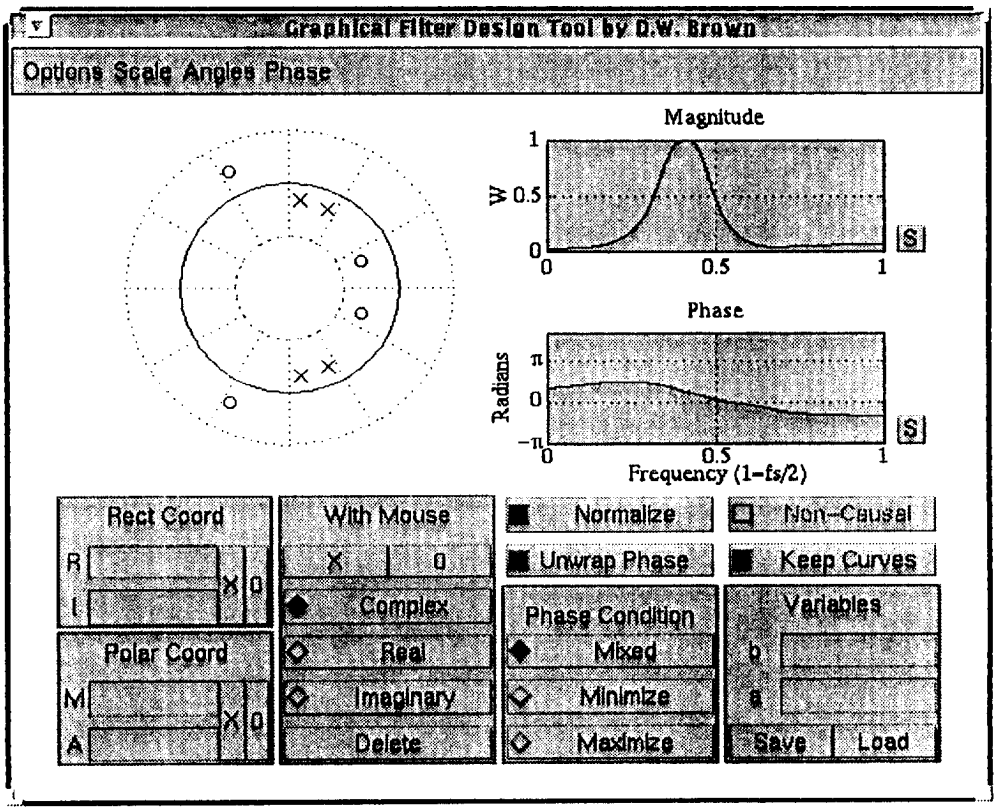

Fig. 1. GFILTERD tool window.

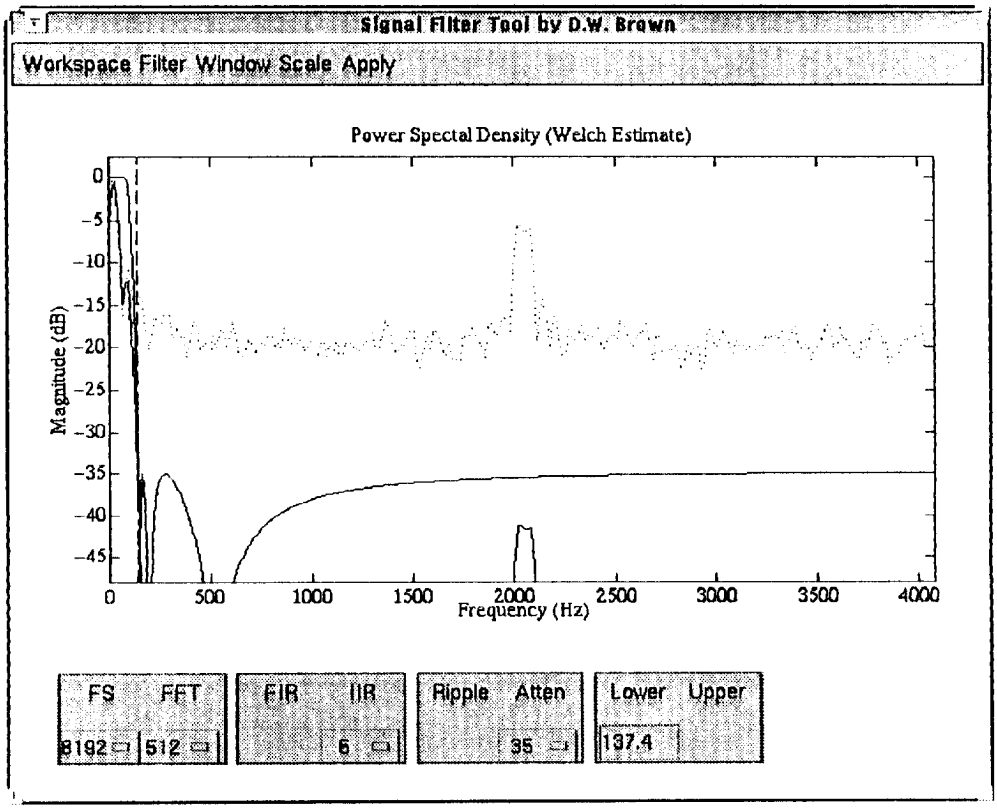

Fig. 2. SIGFILT tool window.