Teaching Real-World DSP Using MATLAB

Cameron H. G. Wright

Department of Electrical Engineering
U.S. Air Force Academy, CO

Thad B. Welch

Department of Electrical Engineering
U.S. Naval Academy, MD

ABSTRACT

An enhancement to MATLAB written by the authors makes teaching various real-world DSP concepts easier and more effective. These enhancements eliminate the need to purchase expensive specialized software programs for this purpose. The particular DSP concept described in this paper is an interactive demonstration showing how quantization of digital filter coefficients, which is inevitable with widely-used fixed-point DSP hardware, adversely affects filter performance.

INTRODUCTION

Modern software tools such as MATLAB greatly facilitate the professor's ability to demonstrate the concepts of digital signal processing (DSP) in class, and to assign realistic projects to reinforce these concepts.¹⁻³ An increasing number of DSP textbooks are becoming available which take advantage of this ability,⁴⁻⁸ and a growing trend is for DSP concepts to be introduced earlier in the curriculum.⁹ MATLAB is an excellent learning tool when used for DSP education in this way. It enables an easier transition for the student to apply concepts to real-world DSP hardware such as the widely-used Texas Instruments TMS320C5X series of fixed-point DSP microprocessors. For example, the `sptool` program supplied with the latest release of the student edition of MATLAB (version 5) and also available in the latest Signal Processing Toolbox (version 4, written for MATLAB 5 Professional) provides an excellent interactive graphical interface for designing both FIR and IIR digital filters.¹⁰ The `sptool` program also allows interactive viewing and analysis of signals and their spectra, but this paper concentrates on the filter design capabilities.

For the student to design a digital filter, `sptool` is executed from the MATLAB command window, and the "New Design" button is clicked under the column for filters. This executes the Filter Designer tool, as shown in Figure 1. Various filter specifications can be easily selected by the student, with an immediate display of the resulting magnitude response. For a more complete analysis of the filter design, the student can click the "View" button from the filter column of the main `sptool` window, executing the Filter Viewer tool, which displays magnitude, phase, impulse response, step response, poles and zeros on the z-plane, group delay, etc., all at the click of a button. See Figure 2. The student can switch back to the Filter Designer tool with a click of the mouse to modify the design parameters and interactively see the results. By clicking the "Apply" button from the filter column of the main `sptool` window, the student can process any selected signal with the selected filter and view the resulting output signal and its associated spectrum.

Figure 1. Graphical user interface for digital filter design provided by MATLAB's sptool program.

The sptool program encourages the student to pursue “what if?” explorations to satisfy their intellectual curiosity and gain a more complete understanding of the underlying DSP concepts.

Note that sptool is simply an easy to use graphical user interface (GUI) which executes m-file programs for filter design that existed in previous versions of both the Signal Processing Toolbox and the Student Edition of MATLAB. The only really new aspect is the interactive GUI. Students tend to use the sptool GUI much more than they ever used the collection of individual m-files from previous versions. Several of our students who already owned the previous Student Edition version chose to purchase version 5 primarily for the availability of sptool.

MORE FUNCTIONS NEEDED

While the capabilities provided by sptool are impressive and greatly facilitate students' comprehension of various DSP topics, it lacks some functions that would allow teaching certain real-world design ramifications. One of the more common topics of this type, suitable for presentation to our senior-level EE majors, is the effect of filter coefficient quantization that occurs in fixed-point DSP microprocessors. While floating-point DSP hardware (such as Texas

Figure 2. Graphical user interface for digital filter analysis as provided by MATLAB's sptool program.

Instruments TMS320C3X series) is much easier to present from a pedagogical standpoint, the fact remains that fixed-point DSP hardware (such as the Texas Instruments TMS320C5X series) is far more prevalent due to its cost and speed advantages. It therefore behooves the professor to expose the students to the important differences between floating-point and fixed-point hardware.

Unfortunately, the ability to demonstrate these ramifications of using fixed-point DSP microprocessors is not currently supported in the latest version of sptool for MATLAB. The calculations performed by MATLAB are double precision, so the filter performance shown to the student using sptool is based on double precision filter coefficients, which would apply to implementation on floating-point DSP hardware. Yet the students must learn that when transitioning from the MATLAB environment to fixed-point hardware such as the TMS320C5X series, there can be serious effects on the transfer function of the digital filter. The locations of the poles and zeros on the complex z-plane will change, sometimes significantly, and the performance of the real-world filter as implemented on the TMS320C5X may be quite different from the performance predicted by sptool using MATLAB. This makes it more difficult for the student to learn how to design a realistic and robust digital filter.

Specialized software programs exist which address this design issue, but they are typically expensive, require the student to learn another interface, and/or are not written for educational purposes.

A DEMONSTRATION PROGRAM

To provide the ability to demonstrate filter coefficient quantization effects while still taking advantage of the many benefits of **sptool**, the authors wrote a MATLAB program which implements this feature as a MATLAB m-file. The program allows the student to interactively compare the theoretical filter performance with the real-world performance using any fixed-point DSP microprocessor, yet still make full use of **sptool**. The program, called **q_filt.m**, is available from the authors for free, eliminates the need for the student to learn another software interface, and is perfectly suited to educational use.

Programmers at The MathWorks wrote **sptool** such that it displays only one signal, filter, or spectrum at a time. What is needed for our purpose is to directly compare, on the same plot, key features of digital filter attributes for both double precision and fixed-point implementations. We also wanted to enable the student to continue to make maximum use of **sptool**. While there is a “plug-in” capability to **sptool** for adding additional features, there is no documentation provided that outlines the software requirements for plug-ins. The authors hope to collaborate in the future with The MathWorks in developing a fully integrated quantization plug-in for **sptool**. In the meantime, we developed **q_filt** as a program that, while it runs outside of **sptool**, easily exchanges information in both directions by using the same data structure format defined by **sptool**.

In order to examine the effects of digital filter coefficient quantization, the student merely designs a filter to the desired specifications using **sptool** in the normal manner. The student then exports the filter to the MATLAB workspace from the main **sptool** window and runs **q_filt** using the syntax:

```
filt2 = q_filt(filt1, num_bits);
```

where **filt2** will be the quantized version of **filt1** for fixed-point DSP hardware using **num_bits** number of bits. The program automatically generates and displays three figures which each compare the two filter implementations on the same plot. The filter characteristics shown in the figures are the transfer function poles and zeros on the complex z-plane, the magnitude response, and the phase response. An example of these three figures for the digital filter previously shown in Figures 1 and 2, where the filter coefficients have been quantized to 16 bits (as is the case for the Texas Instruments TMS320C5X series), is shown in Figures 3, 4, and 5. The quantized version of the filter, **filt2**, can be directly imported into **sptool** from the MATLAB workspace for further analysis.

Figure 3. Comparison of transfer function poles (X) and zeros (O) for floating-point (smaller symbols) and fixed-point (larger symbols) digital filter implementations.

Note how in Figure 3 the poles and zeros are shifted due to the quantization process (we ignore stability concerns for the moment). This results in significant changes to the calculated magnitude and phase response of the filter. The original (IIR elliptic bandpass) filter **filt1** specifications shown by **sptool** using double precision coefficients (applicable to floating-point DSP hardware) included, for example, 70 dB of attenuation in the stop band with very narrow transition regions, and no more than 3 dB of ripple in the pass band (see Figure 1). When quantization to 16 bits is taken into account by the **q_filt** program, this filter performance is altered radically as shown in Figure 4. The phase response of the filter is also changed greatly due to the quantization process. While the phase response in the pass band of the floating-point filter implementation is only slightly nonlinear, it is highly nonlinear in the pass band of the fixed-point filter, as shown in Figure 5. Without **q_filt**, the student would likely assume that the filter design arrived at by using **sptool** would meet the desired specifications. By using this additional program, however, the student gains a better understanding of the design ramifications of fixed-point digital filter implementation. But this isn't the whole story, as we hinted at earlier in reference to filter stability.

Figure 4. Comparison of the magnitude response for floating-point (double precision) and fixed-point (quantized) digital filter implementations.

There is always a danger in relying too heavily on the results of computer simulations and blindly accepting the results. The filter `filt1` used for the example above clearly demonstrates this, as the calculated magnitude and phase response for `filt2` are misleading. It is evident in Figure 3 that `filt2`, as a result of the quantization process, now has some poles outside the unit circle on the complex z -plane. Assuming this is a causal filter design, this implies that the region of convergence for the z -transform does not contain the unit circle, which means the filter design is unstable. We verify this by importing `filt2` into `sptool` and using the Filter Viewer to examine the impulse response, as shown in Figure 6. As expected, the filter “blows up” and would not be usable at all. Yet the magnitude response for `filt2` shown in Figure 4, while no longer meeting the specifications for `filt1`, didn’t look as if it was going to “blow up.” How do we explain this discrepancy?

We always tell our students that no matter how fast the computer simulation may be, they are always smarter than the software, and to always perform a “sanity check” on any results. In this case, Figure 3 tells our students that the region of convergence doesn’t include the unit circle. But MATLAB evaluates the magnitude and phase response of a discrete transfer function by substituting $z = e^{j\omega}$ in the transfer function. This is mathematically equivalent to evaluating the

discrete-time Fourier transform (DTFT) of the filter. An alert student knows, however, that if the unit circle is not contained in the region of convergence of the z-transform, then the DTFT does not exist, which means the magnitude and phase response as calculated by MATLAB is meaningless. Since MATLAB doesn't check for this condition, we added a routine in `q_filt` which will detect it and allow the user to abort the part of the program which calculates the magnitude and phase response. If no poles move outside the unit circle as a result of quantization, or we are dealing with FIR filters (which have no poles), then the calculated magnitude and phase response are valid.

Figure 5. Comparison of the phase response for floating-point (double precision) and fixed-point (quantized) digital filter implementations.

CONCLUSIONS

The program `q_filt` written by the authors provides the professor with an easy to use, inexpensive, and interactive method to teach the concepts of filter coefficient quantization to undergraduate EE majors. We have used it in the classroom at the Air Force Academy and received an enthusiastic response from our students. The program is completely compatible with the excellent program `sptool`, provided with version 5 of the Student Edition of MATLAB and also with version 4 of the Signal Processing Toolbox (to be used with version 5 of the Professional

Edition of MATLAB). It is available free of charge upon request from the authors. We highly recommend this interactive and graphical approach to teaching these DSP concepts to your students.

Figure 6. The impulse response of the quantized filter, shown after being imported back into sptool's Filter Viewer. The quantized version is obviously unstable.

REFERENCES

- [1] Kubichek, R. F., "Using MATLAB in a Speech and Signal Processing Class," *Proceedings of the 1994 ASEE Annual Conference*, pp. 1207–1210, June 1994.
- [2] Burrus, C. S., "Teaching Filter Design Using MATLAB," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 20–30, April 1993.
- [3] Jacquot, R. G., Hamann, J. C., Pierre, J. W., and Kubichek, R. F., "Teaching Digital Filter Design Using Symbolic and Numeric Features of MATLAB," *ASEE Computers in Education Journal*, vol. VII, no. 1, pp. 8–11, January-March 1997.
- [4] Porat, B., *A Course in Digital Signal Processing*, John Wiley & Sons, Inc., 1997.
- [5] Ingle, V. K., and Proakis, J. G., *Digital Signal Processing Using MATLAB V.4*, PWS Publishing, 1997.

- [6] Mitra, S. K., *Digital Signal Processing: A Computer-Based Approach*, McGraw-Hill, 1998.
- [7] Ambardar, A., and Borghesani, C., *Mastering DSP Concepts Using MATLAB*, Prentice-Hall, 1998.
- [8] Burrus, C. S., McClellan, J. H., Oppenheim, A. V., Parks, T. W., Schafer, R. W., and Schuessler, H. W., *Computer-Based Exercises for Signal Processing Using MATLAB*, Prentice-Hall, 1994.
- [9] Yoder, M. A., McClellan, J. H., and Schafer, R. W., "Experiences in Teaching DSP First in the ECE Curriculum," *Proceedings of the 1997 ASEE Annual Conference*, paper 1220-06, June 1997.
- [10] *MATLAB: The Language of Technical Computing*, The MathWorks, Inc., Natick, MA, 1996.

Major CAMERON H. G. WRIGHT, PhD, PE, is an Assistant Professor in the Department of Electrical Engineering at the U.S. Air Force Academy. His research interests include signal and image processing, biomedical instrumentation, communications systems, and laser/electro-optics applications. He is a member of ASEE, IEEE, SPIE, NSPE, Tau Beta Pi, and Eta Kappa Nu. Email: c.h.g.wright@ieee.org

Commander THAD B. WELCH, PhD, is an Assistant Professor in the Department of Electrical Engineering at the U.S. Naval Academy (from 1994–1997 he was an Assistant Professor in the Department of Electrical Engineering at the U.S. Air Force Academy). His research interests include multicarrier communication systems analysis and signal processing. He is a member of ASEE, IEEE, and Eta Kappa Nu. Email: t.b.welch@ieee.org