TEACHING DSP CONCEPTS USING MATLAB AND THE TMS320C31 DSK

Cameron H. G. Wright
Department of Electrical Engineering
U.S. Air Force Academy, CO
c.h.g.wright@ieee.org

Thad B. Welch
Department of Electrical Engineering
U.S. Naval Academy, MD
t.b.welch@ieee.org

Walter J. Gomes III
Naval Undersea Warfare Center
Newport, RI
gomes_w@code80.npt.nuwc.navy.mil

ABSTRACT

A graphically-oriented MATLAB program written by the authors facilitates teaching real-world digital signal processing concepts such as quantization of digital filter coefficients that occur in fixed-point processors, for example the widely used TMS320C5x. While many universities have or plan to buy the inexpensive floating-point TMS320C31 DSKs for pedagogical reasons, this MATLAB program simulates certain fixed-point effects on these floating-point devices and eliminates the need to purchase expensive specialized software programs or extra hardware. The program described in this paper provides an interactive graphical user interface which communicates directly with the DSK, and demonstrates in real-time how coefficient quantization adversely affects filter performance, without the need for tedious programming of the TMS320C31.

1. INTRODUCTION

Modern software tools such as MATLAB greatly facilitate a 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]. These concepts can be further reinforced, and greater interest generated by the students, if they can be easily implemented in real-time on modern DSP hardware. Affordable hardware is now available to schools: Texas Instruments, for example, markets DSP Starter Kits (DSKs) for $99 [10]. While fixed-point processors are more prevalent in industry [11], floating-point processors are becoming more popular for schools due to pedagogical reasons. We will examine how MATLAB, already accepted as a powerful learning tool for DSP, can be closely integrated with a DSK for teaching purposes while avoiding the tedium of manually programming the DSP processor.

This work was supported by the National Science Foundation and the Air Force Office of Scientific Research.

1.1. Teaching with MATLAB

MATLAB is an excellent learning tool for DSP education, enabling an easier transition for the student from theory to practice. In particular, 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.x, written for MATLAB 5.x Professional) provides an excellent interactive graphical user interface (GUI) for designing both FIR and IIR digital filters [12].

Various filter specifications can be easily selected by the student, with a customizable display of the resulting magnitude, phase, impulse response, step response, poles and zeros on the z-plane, and/or group delay. The student can modify the design parameters and interactively see the results. The student can also process any stored signal with the desired filter and view the resulting output signal and its associated spectrum. 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.

1.2. Teaching with DSKs

Another powerful tool to energize and excite students is the ability to implement a particular signal processing technique in real-time on a DSP microprocessor such as one of the Texas Instruments (TI) TMS320C series. When a student speaks into a microphone and hears their "personally designed" digital filter algorithm working in real-time, they are often "hooked" on DSP from then on. The recent availability of affordable DSP Starter Kits (DSKs) has made this feasible for most schools. The kits typically come with an assembler and debugger—sometimes even a C compiler.

There are obstacles to using DSKs, however. The learning curve for programming modern DSP microprocessors is a significant hurdle for most students. They must contend with specialized topics such as parallel instruction execution, block-repeat, bit-reversed addressing, and the often unfamiliar Harvard architecture—and must usually program at the assembly language level. This scares away many students. While fixed-point processors are more prevalent in industry due to their cost and speed advantages, they

add further problems: coefficient quantization, scaling, and other fixed-point ALU and register effects. From a pedagogical point of view, fixed-point processors (such as the widely-used TI TMS320C5x series) tend to be harder to teach in introductory courses compared to floating-point processors such as the TMS320C3x and TMS320C4x. For this reason, many schools are opting to buy floating-point DSP hardware (such as the $99 C31 DSK from TI) for teaching purposes. While the fixed-point effects are important concepts for students to grasp, many schools would like a way to teach this without having to buy additional hardware. The program described below integrates MATLAB closely with the C31 DSK, eliminates the need to create individual assembly language or C programs to manipulate the hardware, and allows the primary fixed-point effects to be simulated in real-time on the floating-point DSK.

2. COMBINING MATLAB AND THE DSK

What is needed is a GUI-based program for MATLAB that can communicate seamlessly with the DSK. While the capabilities provided by `sptool` are impressive and greatly facilitate students' comprehension of various DSP topics, there is no straightforward way to use it directly with a DSK. Also lacking in `sptool` is the ability to simulate for teaching purposes certain fixed-point effects (such as filter coefficient quantization) prior to using the filter design in fixed-point hardware, since MATLAB performs double precision calculations. Specialized software programs exist which address this design issue, but they are typically expensive, require the student to learn another interface, and typically are not written for educational purposes.

2.1. A Fixed-Point Simulation

In response to this need, the authors wrote a MATLAB program which takes up where `sptool` leaves off, adjusting the filter coefficients to simulate fixed-point hardware, allowing interactive analysis of the design effects, and seamlessly downloading the filter code to a C31 DSK when the user is ready. This allows the floating-point DSK to simulate a fixed-point device as desired (except for register and ALU effects, which are less of an issue at the introductory level), and eliminating the need for buying fixed-point hardware just for this purpose. 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 eliminates the need for the student to learn another software interface, and is perfectly suited to educational use. While it runs outside of `sptool`, it easily exchanges information in both directions by using the same data structure format defined by `sptool`.

2.2. A Typical Example

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 from `sptool` to the MATLAB workspace and runs our program by typing `q_filt` at the MATLAB command prompt. This brings up a custom GUI that allows the user to select with the mouse the coefficient quantization method (rounding or truncation, implemented either as a direct form Type II transpose or as second-order cascaded sections), number of bits (8 to 32), plotting preference (magnitude vs. frequency, phase vs. frequency, or poles and zeros on the complex z-plane). Note that the previous version [13] of this program used a command line interface; we feel the GUI version is more appealing to students. The program automatically generates and displays any of the three plots selected which each compare the floating-point vs. fixed-point filter implementations on the same plot. To stay within space limitations, only two figures are shown which depict the program output. A digital filter was previously designed using `sptool` with the following parameters: bandpass elliptic IIR, sample frequency $F_s = 8000$ Hz, passband 900–1400 Hz with 3 dB ripple maximum, transition regions of ≤ 50 Hz, and stop band attenuation of ≥ 70 dB. The resulting design produced by `sptool` is an 8th order filter with actual stopband edges at 872 Hz and 1438 Hz. When the filter coefficients have been quantized by `q_filt` to 16 bits (as would be the case with the Texas Instruments TMS320C5x), the result is shown in Figures 1 and 2.

With quantization effects, the filter performance is altered radically. There are significant changes to the originally calculated magnitude (Figure 1) and phase (not shown) response of the filter, which were predicated on the assumption of floating-point processing. Without `q_filt`, the student would likely assume that the filter design from `sptool` would meet the desired specifications. By using this additional program, however, the student gains a better understanding of the design ramifications of a fixed-point digital filter realization, including the significant differences of the direct form versus second-order section implementations. But this isn't the whole story!

There is always a danger in relying too heavily on the results of computer simulations and blindly accepting the results. The filter used for the example above clearly demonstrates this, as even the filter magnitude and phase response *after* quantization can be misleading. It is evident in Figure 2 that due to the quantization process, some poles have moved 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, meaning the filter design is unstable. We can verify this by importing the quantized filter back into `sptool`

Figure 1: Magnitude plot before (solid) and after (dashed) quantizing the filter coefficients.

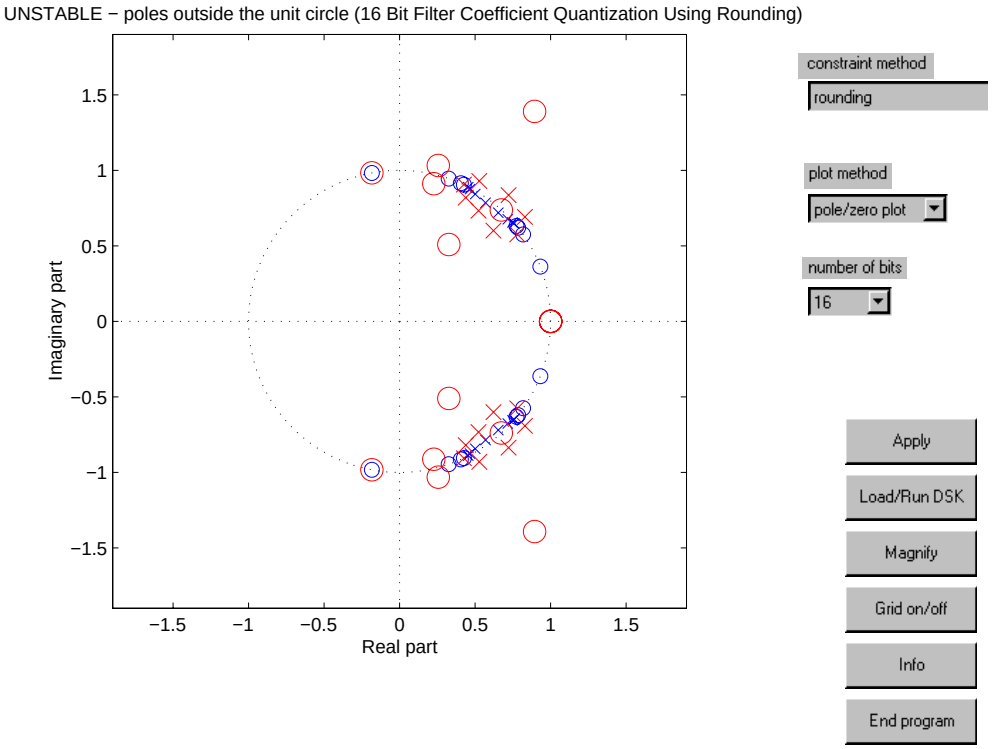

Figure 2: Pole-zero plot before (smaller symbols) and after (larger symbols) quantizing the filter coefficients. Symbols used in the plot: poles shown as "x" and zeros shown as "o."

and examining the impulse response. As expected, the filter "blows up" and would be unstable. Yet the quantized filter magnitude response in Figure 1, while no longer meeting the design specifications, doesn't look unstable. How do we explain this discrepancy? We routinely tell our students that no matter how fast the computer simulation may be, the students are smarter than the computer, and to *always* perform a "sanity check" on any results. In this case, Figure 2 would indicate a stability problem. MATLAB evaluates the magnitude and phase response of a discrete transfer function by substituting $z = e^{j\omega}$ (mathematically equivalent to evaluating the discrete-time Fourier transform (DTFT) of the filter). The student should know, however, that if the unit circle is not contained in the region of convergence of the z-transform, then the DTFT does not exist, and 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 detects it and warns the user by showing the plot with a red background and a special plot title. 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 will be valid; in this case the plot background is white.

When the filter design is satisfactory, the user can simply click the "Load/Run DSK" button on the GUI to download the software to the C31 DSK and run the filter algorithm for a real-time demonstration. No programming is necessary, making this especially attractive for introducing students to DSP hardware. The C31 DSK can be used as a floating-point unit or as a simulated fixed-point unit using `q_filt`.

3. CONCLUSIONS

The program `q_filt` written by the authors provides the educator with an easy to use, inexpensive, and interactive method to teach the concepts of filter coefficient quantization. The program is completely compatible with `sptool`, provided with version 5 of the Student Edition of MATLAB and also with version 4.x of the Signal Processing Toolbox. It easily communicates with the C31 DSK used by many universities, eliminates the need for tedious programming of the DSK, and is available free of charge from the authors via a Web site (the actual address was not finalized at time of publication).

4. REFERENCES

[1] R. F. Kubichek, "Using MATLAB in a speech and signal processing class," in *Proceedings of the 1994 ASEE Annual Conference*, pp. 1207–1210, June 1994.

[2] C. S. Burrus, "Teaching filter design using MATLAB," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 20–30, Apr. 1993.

[3] R. G. Jacquot, J. C. Hamann, J. W. Pierre, and R. F. Kubichek, "Teaching digital filter design using symbolic and numeric features of MATLAB," *ASEE Computers in Education*, vol. VII, pp. 8–11, January-March 1997.

[4] B. Porat, *A Course in Digital Signal Processing*. John Wiley & Sons, 1997.

[5] V. K. Ingle and J. G. Proakis, *Digital Signal Processing Using* MATLAB *V.4*. Bookware Companion Series, PWS Publishing, 1997.

[6] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, 1998.

[7] A. Ambardar and C. Borghesani, *Mastering DSP Concepts Using* MATLAB. Prentice-Hall, 1998.

[8] J. H. McClellan, C. S. Burrus, A. V. Oppenheim, T. W. Parks, R. W. Schafer, and S. W. Schuessler, *Computer-Based Exercises for Signal Processing Using* MATLAB 5. MATLAB Curriculum Series, Prentice-Hall, 1998.

[9] M. A. Yoder, J. H. McClellan, and R. W. Schafer, "Experiences in teaching DSP first in the ECE curriculum," in *Proceedings of the 1997 ASEE Annual Conference*, June 1997. Paper 1220-06.

[10] Texas Instruments, Inc., *TMS320C3x DSP Starter Kit User's Guide*, 1996.

[11] C. Inacio and D. Ombres, "The DSP decision: Fixed point or floating?," *IEEE Spectrum*, pp. 72–74, Sept. 1996.

[12] The MathWorks, Inc., Natick, MA, MATLAB*: The Language of Technical Computing*, 1996.

[13] C. H. G. Wright and T. B. Welch, "Teaching real-world DSP using MATLAB," in *Proceedings of the 1998 ASEE Annual Conference*, (Seattle, WA), June 1998. Paper 1220-03.