Texas Instruments DSPS Fest
Houston, TX, August 4–6, 1999

Making DSP Fun for Students Using Matlab and the C31 DSK

Cameron H. G. Wright
Department of Electrical Engineering, U.S. Air Force Academy, USAFA, CO 80840

Thad B. Welch and Michael G. Morrow
Department of Electrical Engineering, U.S. Naval Academy, Annapolis, MD 21402

Abstract *– This paper describes some innovative programs using a graphical user interface (GUI) for* **Matlab** *and the C31 DSK which makes DSP concepts interesting and entertaining, yet educational. The programs eliminate the need to purchase any expensive specialized software or hardware, relying on the commonly available* **Matlab** *program and the inexpensive Texas Instruments C31 DSK for this purpose.*

The programs described in this paper are a follow-on to the one presented at DSPS Fest '98. The new programs are far more capable and even easier to use than their predecessor: one author (TBW) even managed to get 4th graders to design real-time digital filters then run them on a C31 DSK with one of these programs, and college students love the graphical user interface and the ease of interfacing with the DSP hardware. The programs have been used with great success at both the U.S. Air Force Academy and the U.S. Naval Academy.

Keywords *– DSK, TMS320C31,* **Matlab***, education, DSP*

I. 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]. 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 [10]. Affordable hardware is now available to schools: Texas Instruments, for example, markets DSP Starter Kits (DSKs) for $99 [11]. While fixed-point processors are more prevalent in industry [12] (albeit floating point is gaining in use), 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.

A. Teaching with MATLAB

MATLAB is an excellent learning tool for DSP education, enabling an easier transition for the student from theory to practice. This greatly facilitates a student's ability to apply signal processing concepts to real-world DSP hardware such as the widely-used Texas Instruments TMS320C series of fixed-point and floating-point DSP microprocessors. In particular, the sptool program supplied with the latest release of the student edition[1] of MATLAB and also available in the latest Signal Processing Toolbox (version 4.2, written for MATLAB 5.3 Professional) provides an excellent interactive graphical user interface (GUI) for designing both FIR and IIR digital filters [13]. The sptool program also allows interactive viewing and analysis of signals and their spectra, but this paper concentrates on the filter design capabilities.

A discussion of how to effectively use sptool in DSP education can be found in previous papers [14–16]. Various filter specifications can be easily selected by the student, with an immediate customizable 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. 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. The Filter Designer tool also includes a method to design filters by interactively placing poles and zeros on the z-plane. The student can also process any stored signal with the desired filter and view the resulting output signal and its associated spectrum by clicking the "Apply" button from the filter column of the main sptool window. 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 GUI that 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 [14].

Author e-mail addresses: CHGW: `c.h.g.wright@ieee.org`, TBW: `t.b.welch@ieee.org`, MGM: `morrow@nadn.navy.mil`

[1] The Mathworks is now introducing the more powerful "Student Version" of MATLAB which will eventually replace the "Student Edition" of MATLAB.

B. 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 C31 DSK described in this paper costs only $99 and contains on a single inch circuit board the following items.[2]

- TMS320C31 DSP microprocessor (capable of up to 50 MFLOPS) with 50 MHz clock oscillator. The C31 contains 2 K words (a word is 32 bits) of on-chip RAM, and can also be used with external memory on an add-on card.
- TLC32040 analog interface chip (AIC), which combines a selectable cutoff frequency antialiasing filter (which can also be bypassed), a selectable sampling frequency (up to 20 kHz but can be used at higher frequencies) 14-bit analog-to-digital converter (ADC), a digital-to-analog converter (DAC), a reconstruction filter, and a small output power amplifier which can drive loads $\geq$ 300 W. The analog input and output are intended for audio line-level ($\pm$3 V peak) connections.
- Regulated power supply that accepts either 7–12 Vdc or 6–9 Vac input.
- Host logic for the PC parallel port communication (IEEE 1284).
- Various connectors: RCA jacks for the analog in and out, DB25 parallel port, a 2.1 mm power jack, and four 32-pin headers which can connect all C31 signals to custom add-on cards.

The C31 DSK comes with an assembler, debugger, and assorted documentation. An optimizing C compiler and a wide variety of other development tools are available at extra cost.

The C31 DSK is inexpensive, easy to set up, and can greatly enhance a DSP class. 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 C31 DSK from TI described above) for teaching purposes. While the fixed-point effects are important concepts for students to grasp, many schools would appreciate a way to teach and demonstrate these topics 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. If the student desires to load and run a digital filter design on the DSK without the added effects of fixed-point processors, it is also easily accomplished.

[2] A C33 DSK with much larger memory on board will be available soon.

II. COMBINING MATLAB WITH THE C31 DSK

The authors identified a pressing need for a GUI-based program which would run under Matlab, be able to directly utilize the benefits of sptool mentioned above, and also communicate seamlessly with the C31 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, suitable for presentation to our senior-level EE majors, such as filter coefficient quantization. Matlab performs double precision calculations in sptool, thus a filter design could perform far differently than expected if implemented on a fixed-point processor [14]. While floating-point DSP hardware (such as Texas 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 still more prevalent due to it's cost and speed advantages. It therefore behooves the professor to expose the students to the important differences between floating-point and fixed-point hardware. 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. A Fixed-Point Simulation Using Matlab

In response to this need, the authors wrote a Matlab program that 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, and eliminating the need for buying fixed-point hardware just for this purpose. Based upon feedback received at IEEE ICASSP

'99, the authors recently addded the additional capability to simulate fixed-point register and ALU effects. The program allows the student to interactively compare the theoretical filter performance with the real-world performance that would be encountered using any fixed-point DSP microprocessor, yet still make full use of sptool. The actual performance of the student's filter design can be observed in real-time with the click of a mouse button, which loads and runs the filter on the C31 DSK. The program eliminates the need for the student to learn another software interface, eliminates the need for the students to manually program the DSK, and is perfectly suited to educational use. While it runs outside of sptool, the program easily exchanges information in both directions by using the same data structure format defined by sptool.

B. A Typical Example

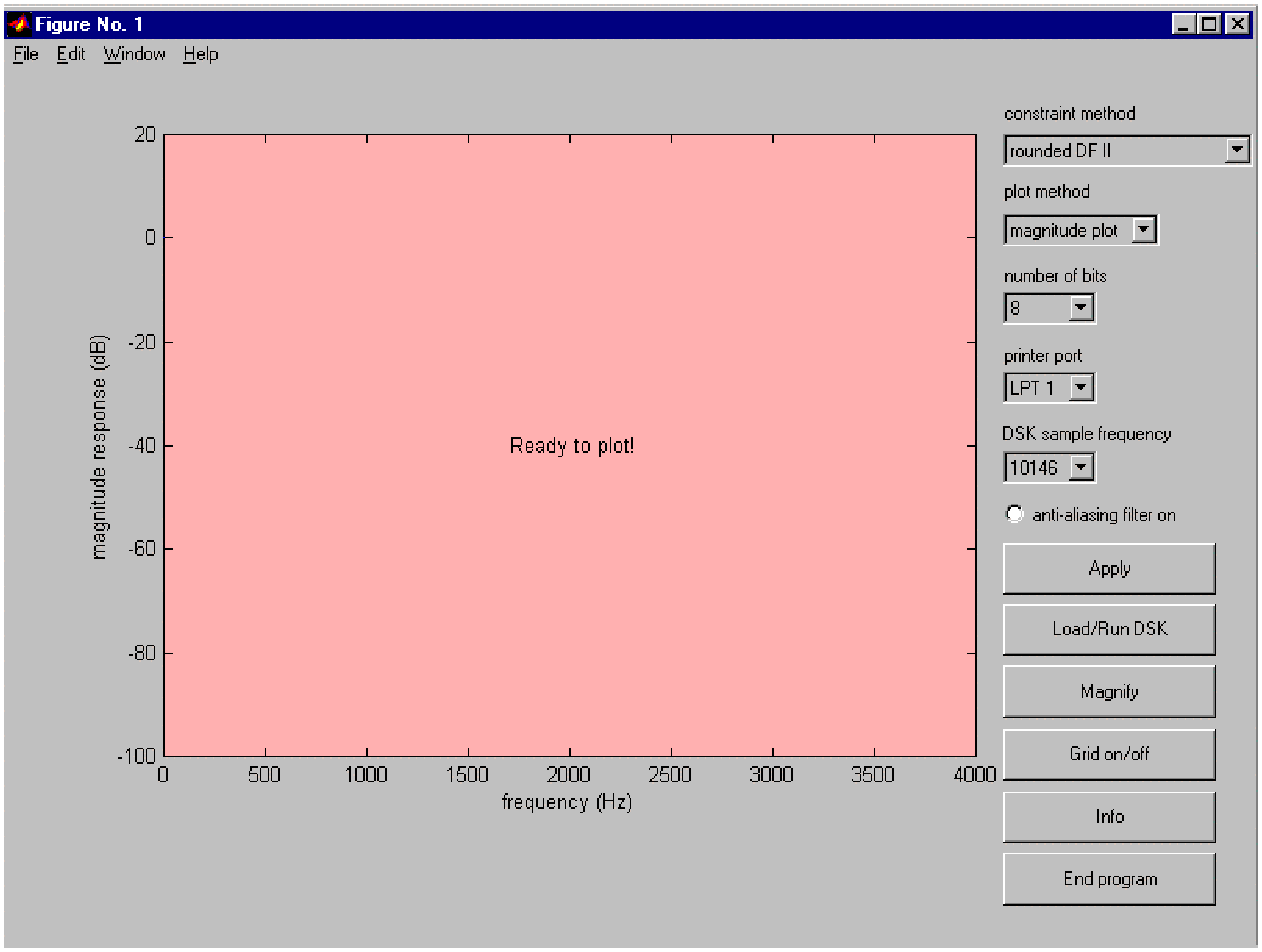
In order to examine the effects of digital filter coefficient quantization or other fixed-point effects, the student merely designs a filter to the desired specifications using sptool in the normal manner. The student then exports the filter from sptool to the MATLAB workspace and runs our program by typing qfilt at the MATLAB command prompt. This brings up the custom GUI shown in Figure 1 which allows the user to select with the mouse the simulation constraint method (rounding or truncating coefficient quantization, floating- or fixed-point ALU and register behavior, and implementation as either a Direct Form Type II transpose or as second-order cascaded sections), number of bits (8 to 32) for the fixed-point effects, and plotting preference (magnitude vs. frequency, phase vs. frequency, or poles and zeros on the complex z-plane). The GUI also allows control over the DSK, and the user can select the port to which the DSK is connected (LPT1–LPT3), the sampling frequency of the AIC (fifty choices from 4509 Hz to 20292 Hz), and control whether or not the antialiasing filter is in the signal path. Note that the previous version of this program used a command line interface and had no ability to communicate with a DSK; we have found the GUI version to be far more appealing to our students and the ability to run their filters in real time on a DSK has been *incredibly* motivational. When the "Apply" button is clicked with the mouse, the program automatically generates and displays any of the three selected plots which each compare the floating-point vs. fixed-point filter implementations on the same plot.

To demonstrate the process a student would use, a digital filter was previously designed using sptool with the following parameters: bandpass elliptic IIR, sample frequency $Fs = 8117$ 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 1439 Hz. When the filter coefficients have been quantized by qfilt to 16 bits (as would be the case with the Texas Instruments TMS320C5x) and implemented as a Direct Form Type II transpose, the result is shown in Figure 2 through Figure 4.

The student can imediately see that with quantization effects, the filter performance is altered radically. There are significant changes to the originally calculated magnitude (Figure 2) and phase (Figure 3) response of the filter, which were predicated on the assumption of floating-point processing. 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 4 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 and examining the impulse response. As expected, the filter "blows up" and would be unstable. Yet the quantized filter magnitude response in Figure 2, 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 4 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, the 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 qfilt 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 non-trivial poles), then the calculated magnitude and phase response will be valid and the plot background would be white.

The student might then explore if the same filter would behave any differently if it was implemented as a cascaded second order section (referred to in some DSP texts as "biquads"). The student simply selects this with the mouse and clicks the "Apply" button once again for the various plots. As can be readily seen in Figure 5 and Figure 6, the filter is now stable (note the plot titles and background color) and comes so close to matching the floating point performance that the difference is virtually indistinguishable. Without qfilt, the student would likely assume that the filter design from sptool would meet the desired specifications no matter how it was implemented. By using our program, however, the student gains a better understand-



Ready to plot!

constraint method: rounded DF II
plot method: magnitude plot
number of bits: 8
printer port: LPT 1
DSK sample frequency: 10146
anti-aliasing filter on

Fig. 1. Initial screen of the graphical user interface (GUI) for qfilt.

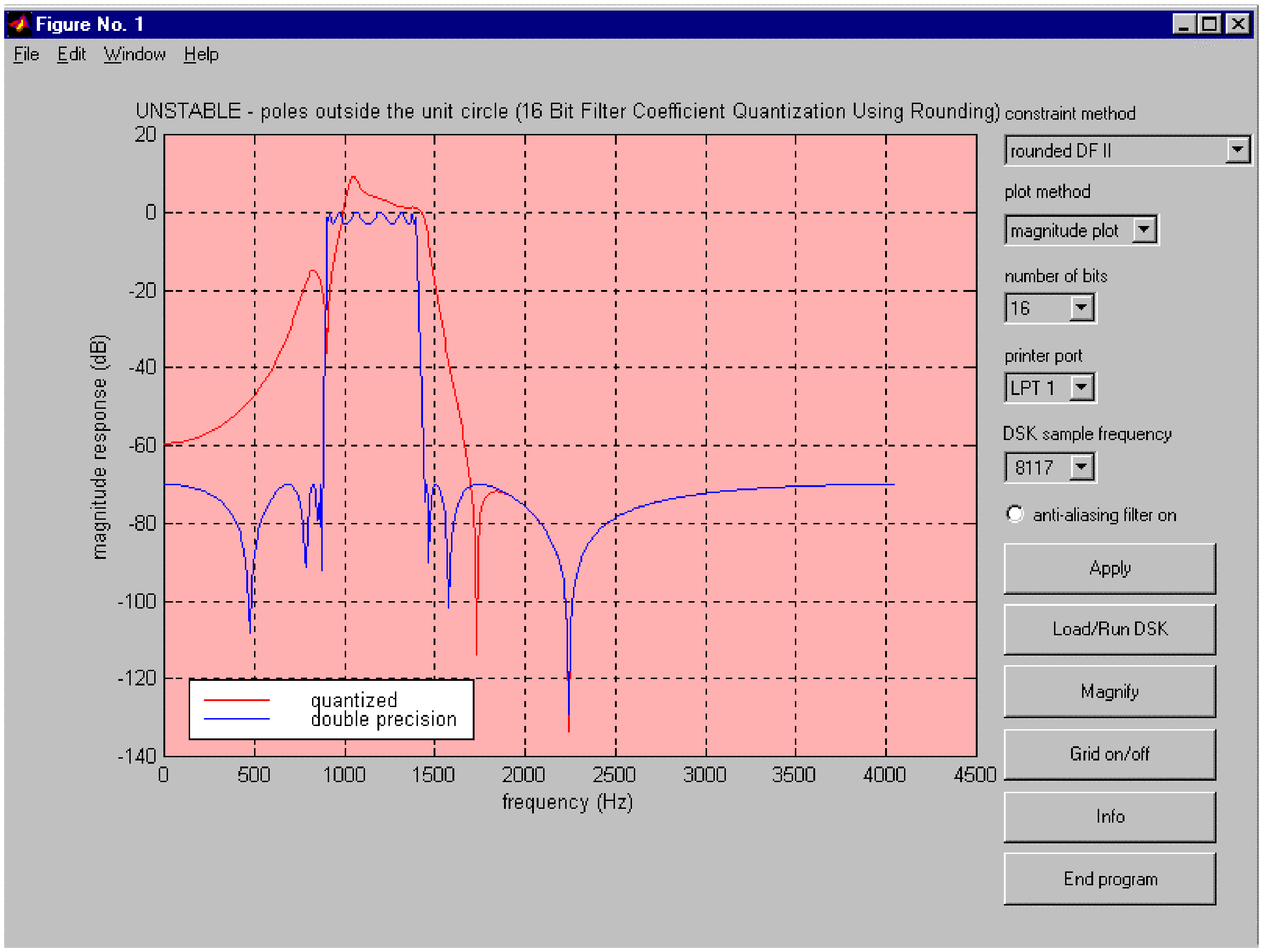
ing of the design ramifications of a fixed-point digital filter realization, including the significant differences of the direct form versus second-order section implementations.

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. This download and run process takes less than a quarter of a second, which gives a feeling of immediacy to the student. No programming is necessary, making this especially attractive for introducing students to DSP hardware. The "Load/Run DSK" button activates a 32-bit dynamic link library (DLL) written with Microsoft Visual C++ 5.0 and the MATLAB MEX file process to run under Windows 9x or Windows NT; different programs execute depending upon whether the user has selected Direct Form Type II transpose or cascaded second order sections. To stay within the on-chip memory limits of the C31 DSK, the maximum order supported is a 254 order IIR Direct Form Type II transpose and a 256 order IIR cascaded second order sections. By specifying a high bit number (such as 32), quantization effects are miniscule and the C31 DSK can be used as a normal floating-point unit or, as described above, the same floating-point DSK can be use to simulate fixed-point unit.

A companion program called polezero with a similarly designed GUI for teaching DSP using MATLAB and the C31 DSK allows students to perform interactive adjustment and "what if?" analysis of pole-zero plots [16]. The polezero program allows the student to design basic digital filters by interactively placing poles and zeros on the complex z-plane, observing the results, and immediately running the resulting filter on the C31 DSK. The initial screen presented to the user for polezero is shown in Figure 7.

III. CONCLUSIONS

The programs qfilt and polezero written by the authors provide the educator with easy to use, inexpensive, and interactive methods to teach various concepts of digital filter design so important in DSP classes. The programs are completely compatible with sptool provided with version 5 of the Student Edition of MATLAB and also with version



constraint method: rounded DF II

plot method: magnitude plot

number of bits: 16

printer port: LPT 1

DSK sample frequency: 8117

anti-aliasing filter on

Fig. 2. Magnitude plot of 8th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose.

4.2 of the Signal Processing Toolbox. They both easily communicate with the C31 DSK that is used by many universities, they eliminate the need for tedious programming of the DSK, and both are freely available from the following Web site.

`wseweb.ew.usna.edu/ee/links/ee_links.htm`

Should the URL be changed, then navigate from the Naval Academy home page and select Academics, Academic Divisions and Departments, Electrical Engineering, Links.

Note that while these programs do not require programming by the student, our experiences have shown that once most students "play" with these programs a bit, become comfortable with the DSK, and start to see what the device can do, they *want* to learn how to program the DSK. This is how learning DSP can be fun for the student.

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.

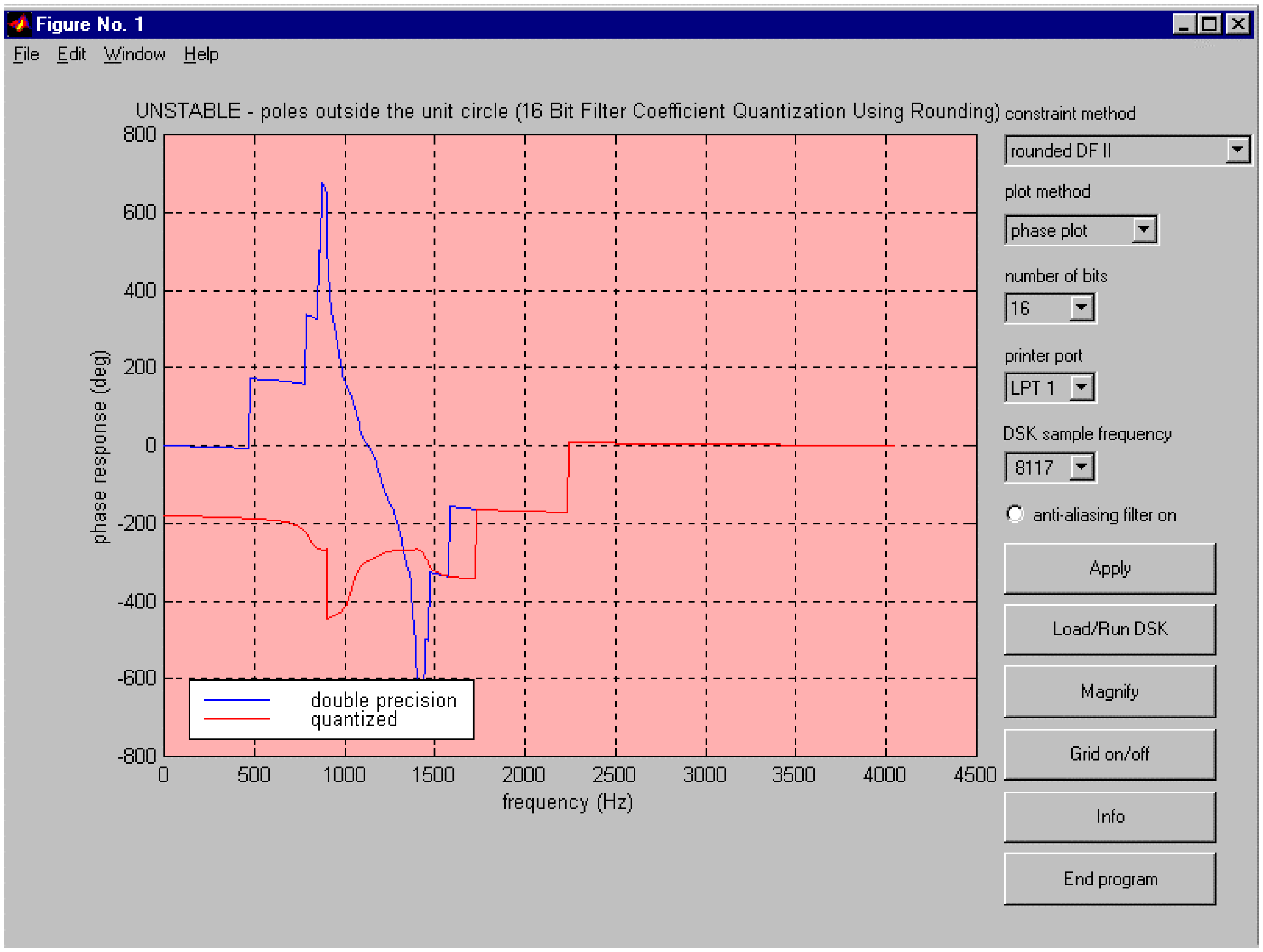
[7] A. Ambardar and C. Borghesani, *Mastering DSP Concepts Using* MATLAB. Prentice-Hall, 1998.

[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] R. Chassaing, *Digital Signal Processing: Laboratory Experiments Using C and the TMS320C31 DSK.* John Wiley & Sons, 1999.

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



UNSTABLE - poles outside the unit circle (16 Bit Filter Coefficient Quantization Using Rounding)

constraint method: rounded DF II

plot method: phase plot

number of bits: 16

printer port: LPT 1

DSK sample frequency: 8117

Fig. 3. Phase plot of 8th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose.

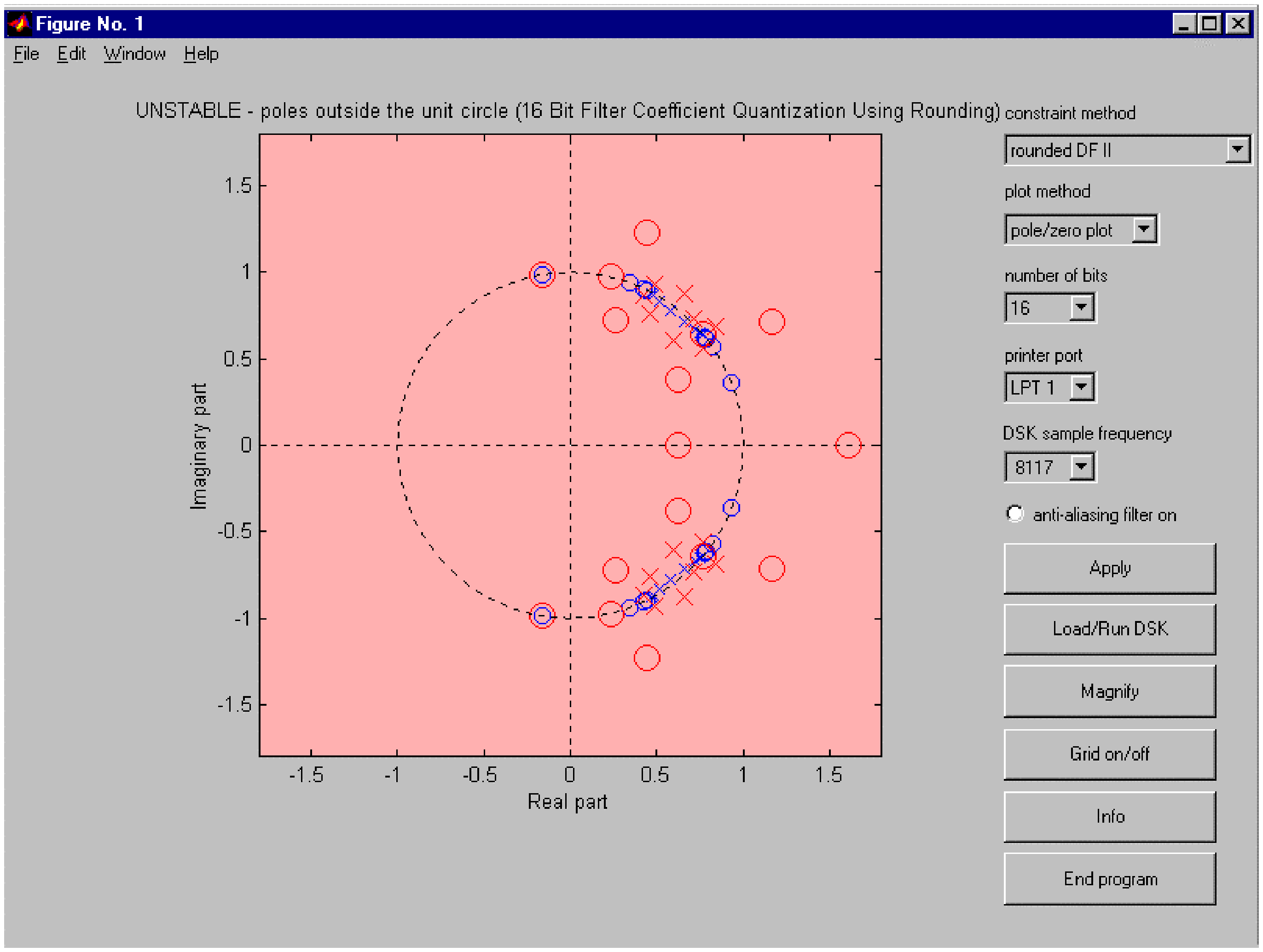
[12] C. Inacio and D. Ombres, "The DSP decision: Fixed point or floating?," *IEEE Spectrum*, pp. 72–74, Sept. 1996.

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

[14] C. H. G. Wright and T. B. Welch, "Teaching real-world DSP using MATLAB," *ASEE Computers in Education Journal*, vol. IX, pp. 1–5, Jan–Mar 1999.

[15] C. H. G. Wright and T. B. Welch, "Teaching DSP concepts using MATLAB and the TMS320C31 DSK," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 1999. Paper 1778.

[16] T. B. Welch, C. H. G. Wright, and M. G. Morrow, "Poles, zeros, and MATLAB, oh my!," in *Proceedings of the 1998 ASEE Annual Conference*, (Charlotte, NC), June 1999. Paper 1320-02.



UNSTABLE - poles outside the unit circle (16 Bit Filter Coefficient Quantization Using Rounding)

constraint method: rounded DF II

plot method: pole/zero plot

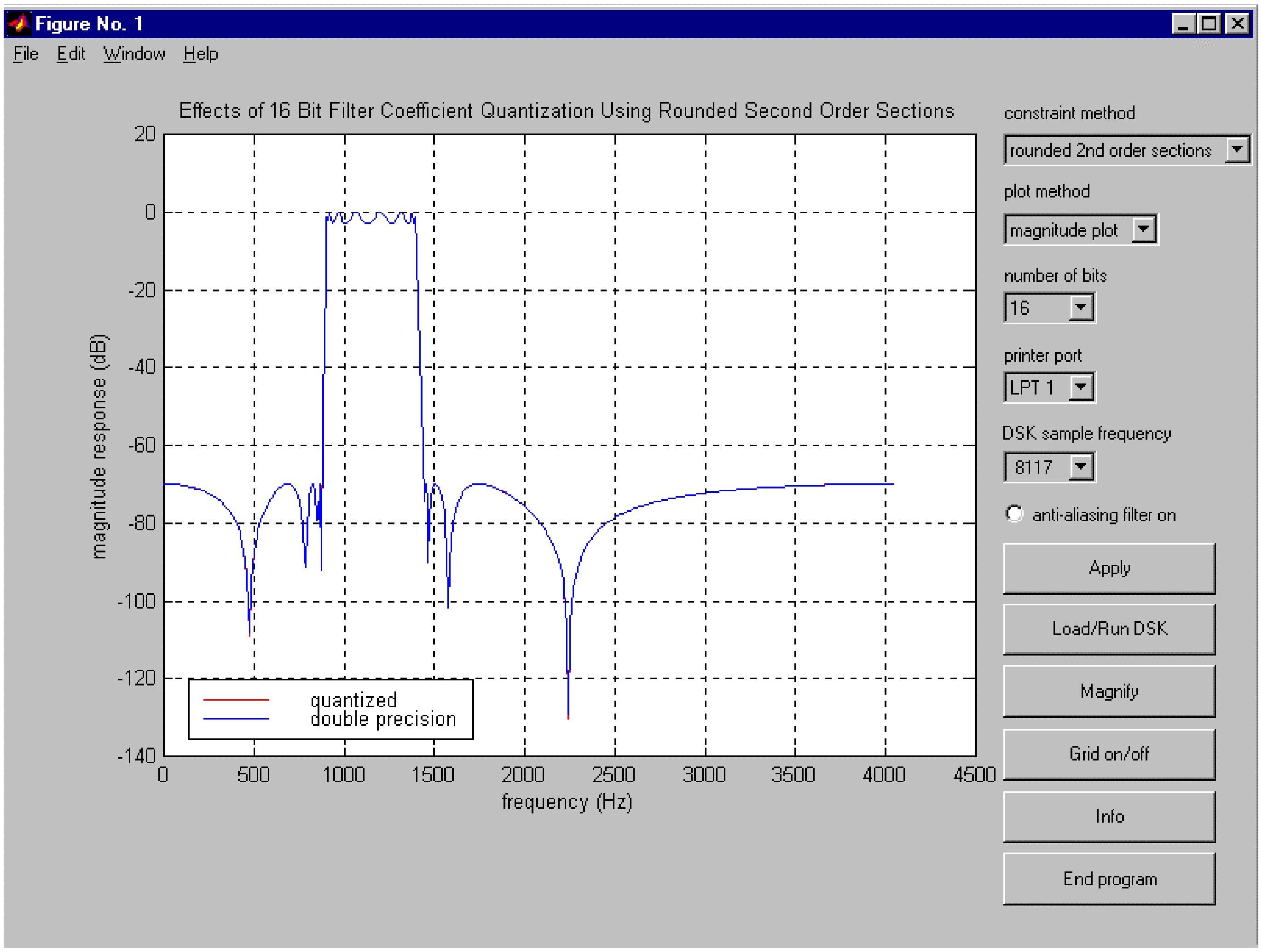
number of bits: 16

printer port: LPT 1

DSK sample frequency: 8117

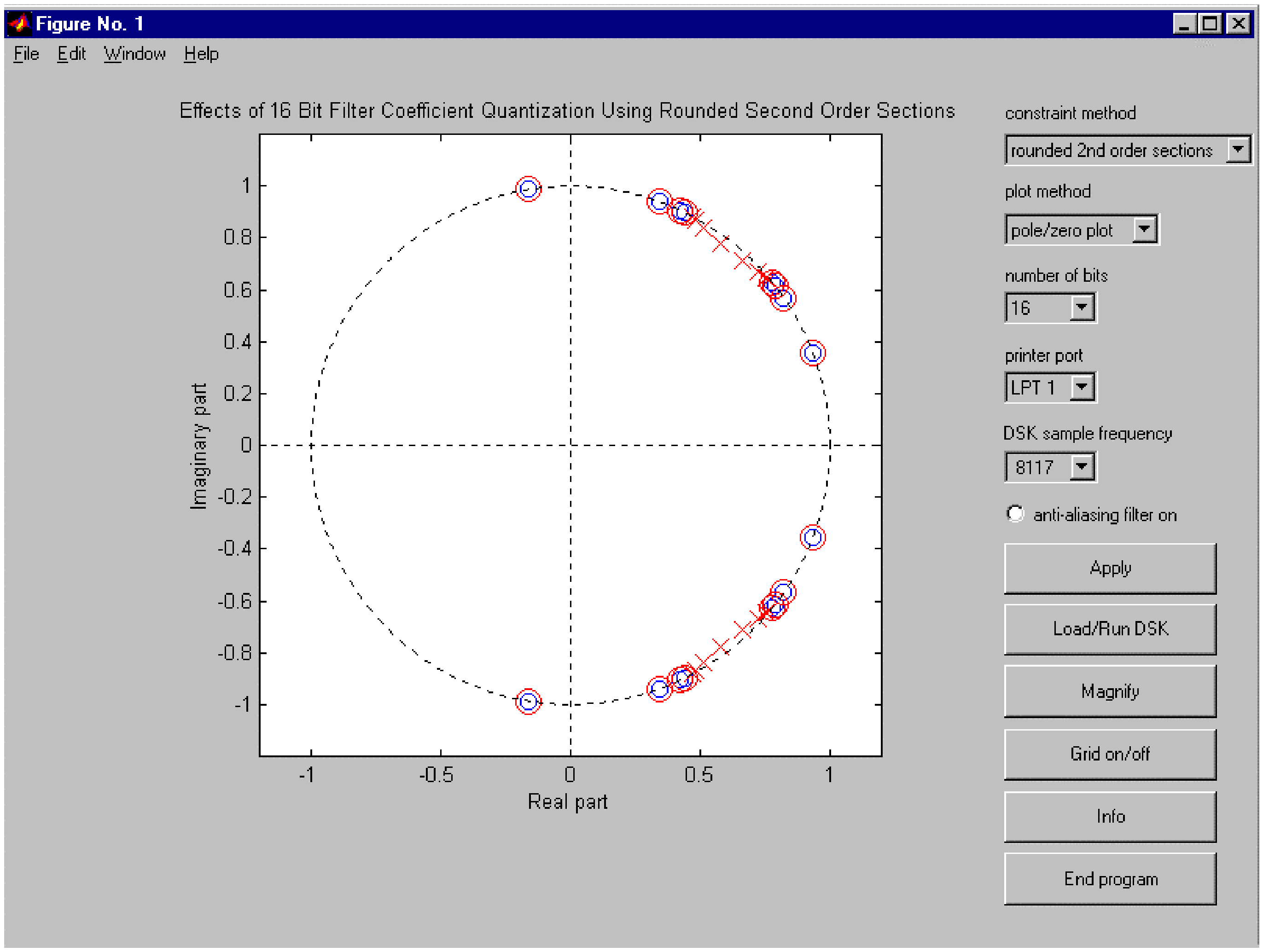
anti-aliasing filter on

Fig. 4. Pole-zero plot of 8th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose.



constraint method: rounded 2nd order sections
plot method: magnitude plot
number of bits: 16
printer port: LPT 1
DSK sample frequency: 8117
○ anti-aliasing filter on

Fig. 5. Magnitude plot of 8th order IIR Elliptic digital filter, quantized to 16 bits and implemented as cascaded second order sections.



constraint method
rounded 2nd order sections

plot method
pole/zero plot

number of bits
16

printer port
LPT 1

DSK sample frequency
8117

anti-aliasing filter on

Fig. 6. Pole-zero plot of 8th order IIR Elliptic digital filter, quantized to 16 bits and implemented as cascaded second order sections.

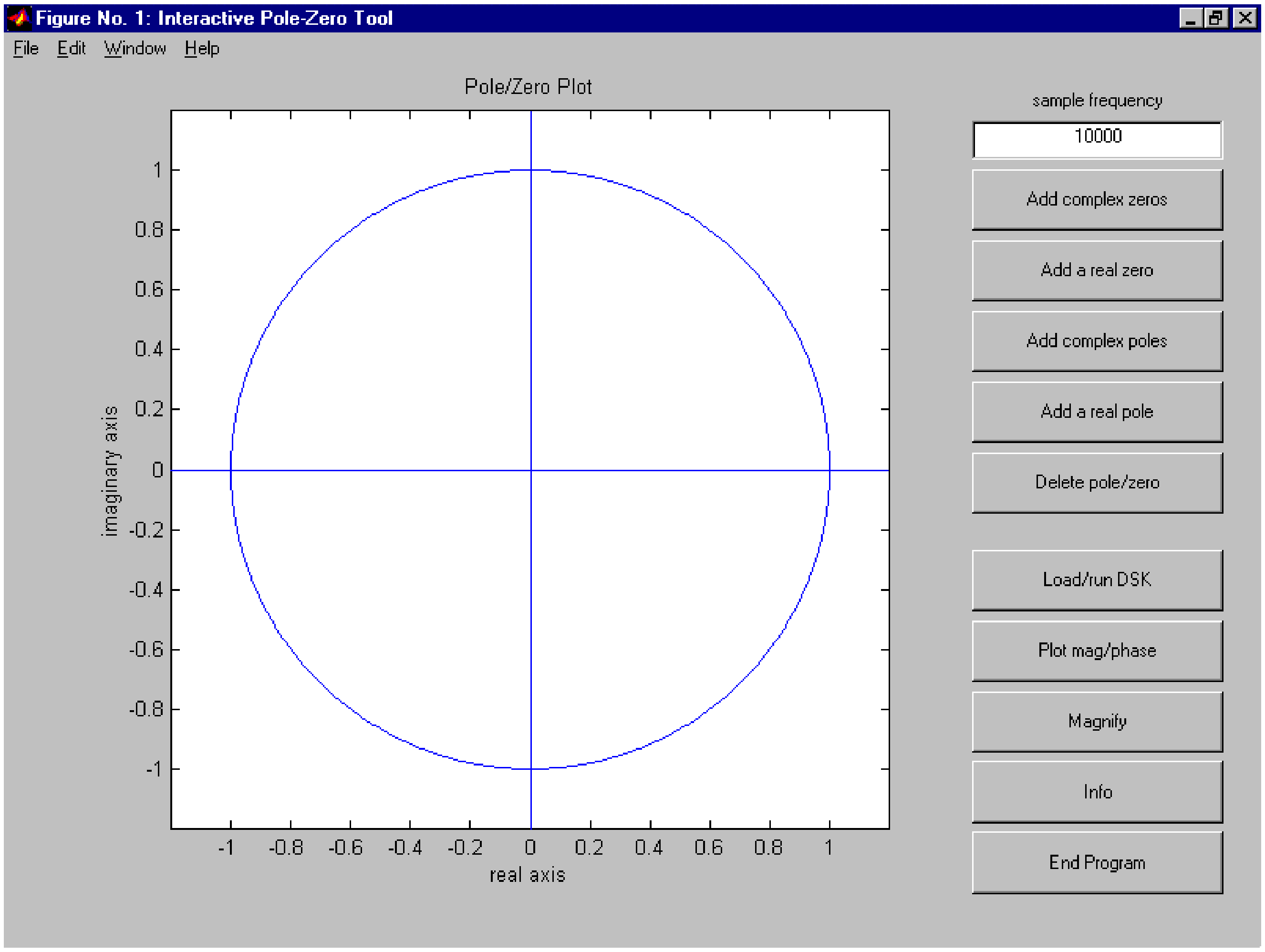


Fig. 7. Initial screen of the graphical user interface (GUI) for polezero.