# Teaching Practical Hands-On DSP with MATLAB and the C31 DSK

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

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

## Abstract

A graphically oriented MATLAB program, written by the authors, facilitates teaching real-world DSP concepts such as quantization of digital filter coefficients that occur in fixed-point processors. While many universities own or plan to buy, the inexpensive floating-point TMS320C31 DSKs for pedagogical reasons, this MATLAB program fully supports that decision by allowing filter coefficient quantization effects to be demonstrated using either fixed-point or floating point filtering algorithms on a single DSP device. This program eliminates the need to purchase expensive specialized software programs or additional DSP 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 both coefficient quantization and filter implementation affect filter performance. All of this is accomplished without the need for tedious programming of the DSK. The latest version of the software allows the floating-point TMS320C31 to directly implement the fixed-point filtering algorithm. This enhancement represents a considerable cost and timesavings for both the student and professor since only one type of digital signal processor is required.

## 1. 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 available which take advantage of this ability,[4–9] and a growing trend is for DSP concepts to be introduced earlier in the curriculum.[10] 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.[11] While fixed-point processors are more prevalent in industry,[12] floating-point processors are becoming more popular for schools for 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.

### 1.1 Teaching with MATLAB

MATLAB is an excellent learning tool for DSP education, enabling an easier transition from theory to practice for the student. 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 is supplied with the latest release of the latest Signal Processing Toolbox (version 4.3). The Student Version of MATLAB (release 11) does not come with the Signal Processing Toolbox; however, students may purchase this and other toolboxes at a very reasonable price. This program 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 using sptool in DSP education can be found in a previous paper by Wright and Welch.[14] Various filter specifications are 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. By clicking the "Apply" button from the filter column of the main sptool window, 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.

Note that sptool is actually an easy to use GUI that executes m-file programs for filter design that existed in previous versions of the Signal Processing Toolbox of MATLAB. The only really new aspect is the interactive GUI. Students tend to use the sptool GUI much more frequently than they ever used the collection of individual m-files from previous versions.[14]

### 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) TMS320 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. 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 $3.5 \times 5$ inch circuit board the following items,

- TMS320C31 DSP microprocessor (capable of up to 50 MFLOPS) with a 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 anti-aliasing 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 ≥ 300 Ω. The analog input and output are intended for audio line-level (± 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. An interesting addition to the TI provided tools, winDSK, was developed by Morrow[15], and is available at,

URL: http://wseweb.ew.usna.edu/ee/LINKS/EE_Links.htm

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 or C programming language level. These issues frighten away many students. While fixed-point processors are more prevalent in industry due to their cost, speed, and power advantages, they add additional problems: coefficient quantization, scaling, and other fixed-point ALU and register effects. From a pedagogical perspective, 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. Direct implementation of the fixed-point algorithm on the floating-point processor is also available. If the student desires to load and run a digital filter design on the DSK without the added effects of fixed-point processors, that is also easily accomplished.

## 2. Combining MATLAB with the C31 DSK

The authors identified an urgent 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 upper-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 perspective, the fact remains that fixed-point DSP hardware (such as the Texas Instruments TMS320C5x series) is far more prevalent due to it's cost, speed, and power 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, or are not written with educational purposes in mind.

### 2.1 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 demonstrate filter coefficient quantization effects using floating-point algorithms or implement both the filter coefficient quantization effects and fixed-point register/ALU effects using fixed-point algorithms. 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 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 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 quantization, and implemented either as a Direct Form Type II transpose or as second-order cascaded sections), number of bits (8 to 32 for floating point implementation or 8 to 16 for fixed-point implementation), plotting preference (magnitude vs. frequency, phase vs. frequency, or poles and zeros on the complex *z*-plane), and Q format (Q12 to Q15) for fixed point implementation. 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 anti-aliasing filter is in the signal path. The input and output gain windows allow for pre-scaling and post-scaling during fixed-point implementation. Any value may be entered, however, the value will be rounded to the nearest power of 2. The GUI regions labeled "Eqn 6.99", "Eqn 6.102", and "Eqn 6.104" refer to the scale factors discussed in Oppenhiem and Schafer's text[16]. These scale factors are calculated and displayed as soon as the "Apply" button is clicked with the mouse.

The original version of this program[14] used a command line interface and had no ability to communicate with a DSK. Another version of this program[17] added the ability to communicate with the DSK with filter coefficient quantization effects demonstrated using floating-point algorithms. 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 very motivational. When the "Apply" button is clicked with the mouse, the program automatically generates and displays any of the three selected plots with each comparing the floating-point vs. fixed-point filter implementations.

To demonstrate the process a student would use, a digital filter was designed using sptool with the following parameters: lowpass elliptic IIR, sample frequency $F_s$ = 10,000 Hz, passband 500 Hz with 1 dB ripple maximum, stopband 750 Hz, and stop band attenuation of ≥80 dB. The resulting design produced by sptool is of 7th order. 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 immediately 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 is 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 $z = e^{jw}$ (mathematically equivalent to evaluating the discrete-time Fourier transform (DTFT) of the filter). The student should know 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 only recently started to check for this condition, we added a routine in qfilt that 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 that case the plot background would be white.

The student might then explore whether the same filter would behave any differently if it were 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 differences are virtually indistinguishable. Without qfilt, the student would assume that the filter design from sptool meets the desired specifications. By using our 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.

The unquantized filter coefficients and the quantized filter coefficients are available in the MATLAB workspace in the data structure named filt1 and filt2 respectively. If we use the standard notation that **B** represents the numerator coefficients and **A** represents the denominator coefficients of the filter's transfer function **H(z)** we obtain,

original **B** = 1.0e-003 * [0.1585 -0.5992 0.9131 -0.4655 -0.4655 0.9131 -0.5992 0.1585]

sos **B** = 1.0e-003 * [0.1526 -0.5770 0.8792 -0.4482 -0.4482 0.8792 -0.5770 0.1526]

rounded **B** = 1.0e-003 * [0.1526 -0.6104 0.9155 -0.4578 -0.4578 0.9155 -0.6104 0.1526]

original **A** = [1.0000 -6.5302 18.4564 -29.2562 28.0842 -16.3231 5.3183 -0.7493]

sos **A** = [1.0000 -6.5302 18.4562 -29.2559 28.0838 -16.3229 5.3182 -0.7493]

rounded **A** = [1.0000 -6.5302 18.4564 -29.2562 28.0842 -16.3232 5.3183 -0.7493].

It is now clearly seen that even though the rounded coefficients are closer in value to the original coefficients this does not result in the most accurate pole/zero placement. Additionally, with the **A** and **B** coefficient vectors available in the workspace, other MATLAB commands may be used to explore these or other effects in more detail. The application of an input scale factor will also be visible in the filter coefficients.

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 "Load/Run DSK" button activates a 32-bit dynamic link library (DLL) written with Microsoft Visual C++ 6.0 and the MATLAB MEX file process to run under Windows 9x; different programs are loaded into the DSK depending upon whether the user has selected Direct Form Type II transpose or cascaded second order sections, or floating-point or fixed-point implementation.

The filter coefficients are then downloaded to the DSK before running the DSK program. The entire process is accomplished in about 300 milliseconds, and gives the student immediate real-world implementation of their filter design. Floating-point filters are implemented on the C31 DSK using the native floating-point hardware capabilities of the processor. To stay within the memory and processing limitations of the DSK, the maximum filter supported is a 254 order IIR Direct Form Type II transpose and a 256 order IIR cascaded second order section. At high bit

numbers (such as the full 32 bit capability of the C31 DSK), quantization effects become insignificant.

Fixed point filters are implemented using the C31's 32 bit integer ALU, with coefficients and data limited by software to 16 bits in Q12 through Q15 format. Intermediate multiplication results are stored as 32 bit integers in Q24 through Q30 format, respectively, with all final results stored as 16 bit integers. The fixed-point filter implementation is constrained by the DSK's memory and processing limitations to a 254 order IIR Direct Form Type II transpose and a 200 order IIR cascaded second order section.

A companion program for teaching DSP using MATLAB and the `C31 DSK, which allows students to perform interactive adjustment and "what if?" analysis of a pole-zero plot, is described in an accompanying paper[18] by Welch, Wright, and Morrow.

## 3. Conclusions

The qfilt program written by the authors provides the educator with an easy to use, inexpensive, and interactive method for teaching the concepts of filter coefficient quantization. The program is completely compatible with the MATLAB sptool program. 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 our Web site,

URL: http://wseweb.ew.usna.edu/ee/LINKS/EE_Links.htm

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

[9] Childers, D. G., *Speech Processing and Synthesis Toolboxes*, John Wiley & Sons, Inc., 2000.

[10] Yoder, M. A., McClellan, J. H., and Schafer, R. W., "Experiences in Teaching DSP First in the ECE *Proceedings of the 1997 ASEE Annual Conference*, paper 1220-06, June 1997.

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

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

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

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

[15] Morrow, G. M., "winDSK: A Windows-Based C31 DSK Demonstration and Debugging Program," *Proceedings of the 1999 Texas Instrument DSPS Conference*, August 1999.

[16] Oppenheim, A. V. and Schafer, R. W., with Buck, J. R., *Discrete-Time Signal Processing*, Prentice-Hall, 1999.

[17] Wright, C. H. G., Welch, T. B., and Morrow, M. G., "Teaching Real-World DSP Using MATLAB and the TMS320C31 DSK" in *Proceedings of the 1999 ASEE Annual Conference*, (Charlotte, NC), June 1999. Paper 1320-03.

[18] Welch, T. B., Wright, C. H. G., and Morrow, M. G., "Poles, Zeros, and MATLAB, Oh My!" in *Proceedings of the 1999 ASEE Annual Conference*, (Charlotte, NC), June 1999. Paper 1320-04.

Commander THAD B. WELCH, PhD, PE, 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 system design and analysis, RF channel measurements, and real-time signal processing. He is a member of ASEE and Eta Kappa Nu and a senior member of the IEEE. Email: t.b.welch@ieee.org

Lieutenant Commander MICHAEL G. MORROW, PE, is a Master Instructor in the Department of Electrical Engineering at the U.S. Naval Academy. His research interests include real-time digital systems, power system automation, and software engineering. He is a member of ASEE and IEEE. Email: morrow@nadn.navy.mil

Lieutenant Colonel CAMERON H. G. WRIGHT, PhD, PE, is an Associate 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

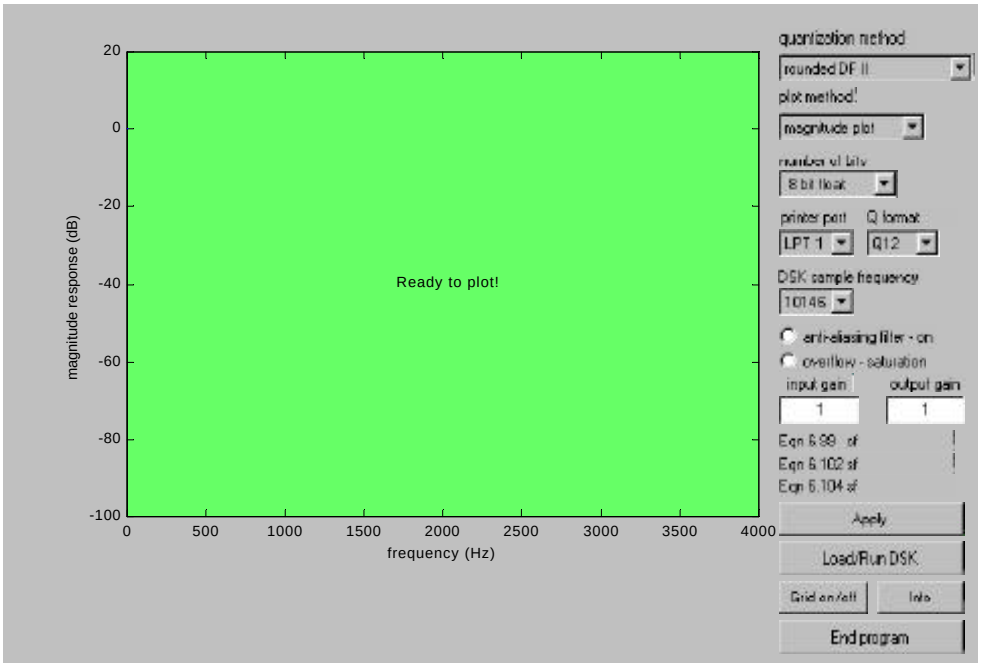


**Figure 1. Initial screen of qfilt GUI.**

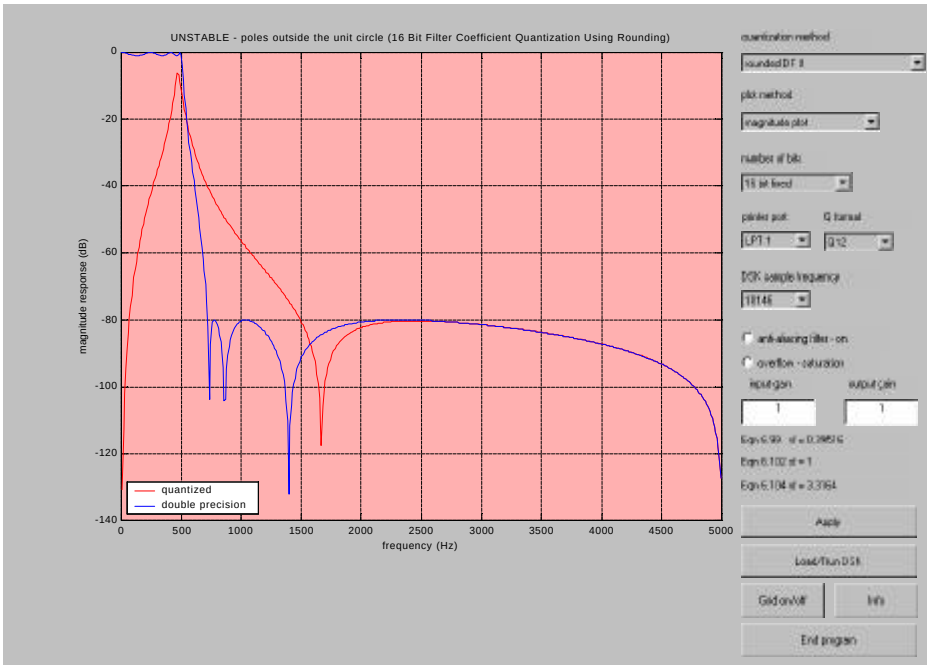


**Figure 2. Magnitude plot of 7th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose using fixed-point algorithms.**

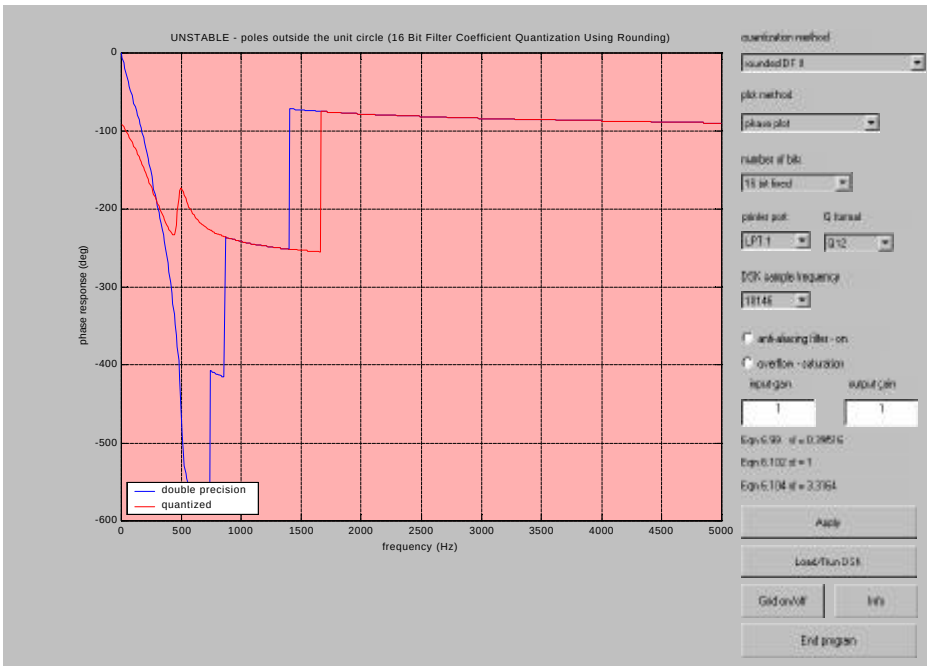


**Figure 3. Phase plot of 7th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose using fixed-point algorithms.**

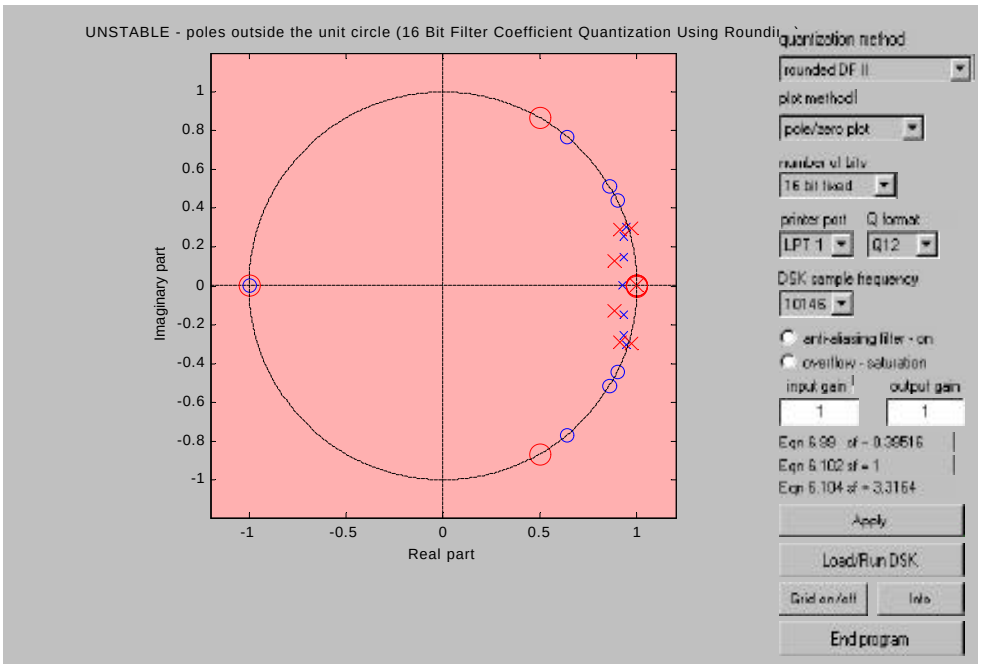


**Figure 4. Pole-zero plot of 7th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose using fixed-point algorithms.**

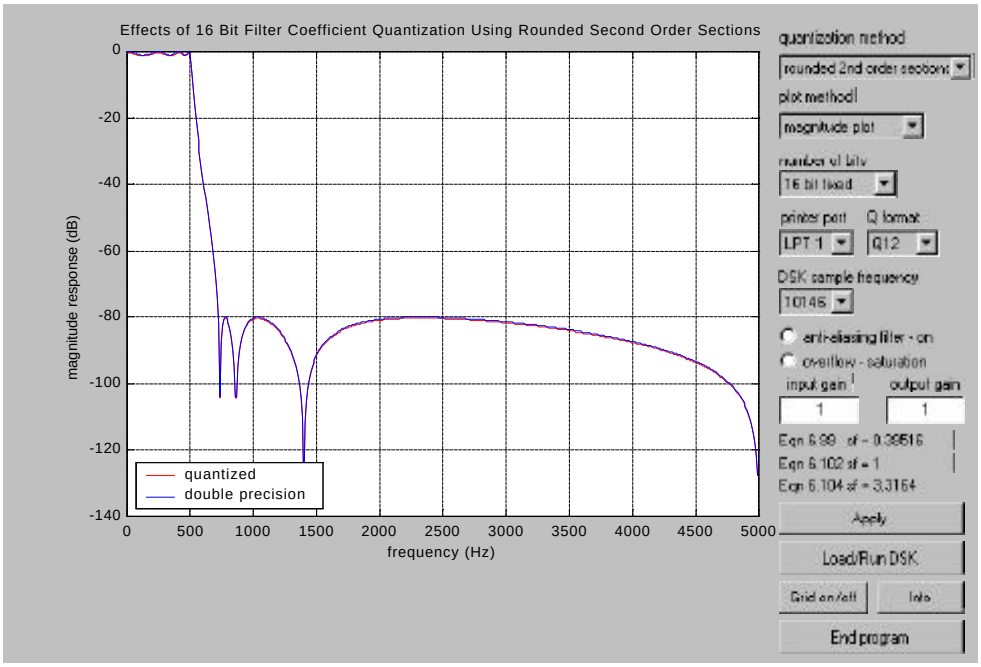


**Figure 5. Magnitude plot of 7th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose using fixed-point algorithms.**

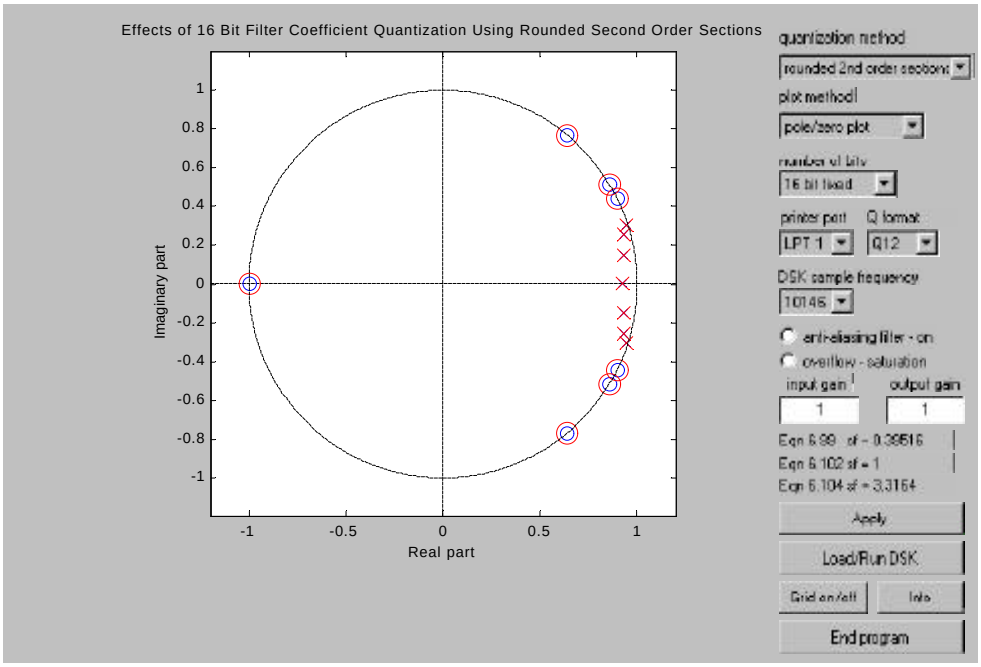


**Figure 6. Pole-zero plot of 7th order IIR Elliptic digital filter, quantized to 16 bits and implemented as a Direct Form II transpose using fixed-point algorithms.**