Teaching Transfer Functions with MATLAB and Real-Time DSP

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

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

Abstract

This paper describes a synergistic combination of hardware and software which makes teaching and demonstrating the concepts of system transfer functions much easier and more effective. The program interfaces MATLAB directly to an inexpensive Texas Instruments TMS320C6711 DSP Starter Kit without requiring specialized programming by the professor or student. It eliminates the need to purchase expensive software or hardware for teaching these concepts.

1 Introduction

Modern software tools such as MATLAB greatly facilitate a professor's ability to demonstrate a variety of concepts, including linear systems and the underlying system transfer function (TF). These concepts can be further reinforced, and greater interest generated by the students, if they can be easily demonstrated in real-time using modern digital signal processing (DSP) hardware. Affordable hardware is now available to schools, for example, in the form of DSP Starter Kits (DSKs) by Texas Instruments. This paper will examine how MATLAB, already accepted as a powerful educational tool in engineering, can be closely integrated with a DSK for teaching purposes while avoiding the tedium of manually programming the DSP device.

2 Signal Processing Background

The process whereby an engineer determines the transfer function of an unknown "black box" based upon the inputs and outputs is known as *system identification* (SI). The need to

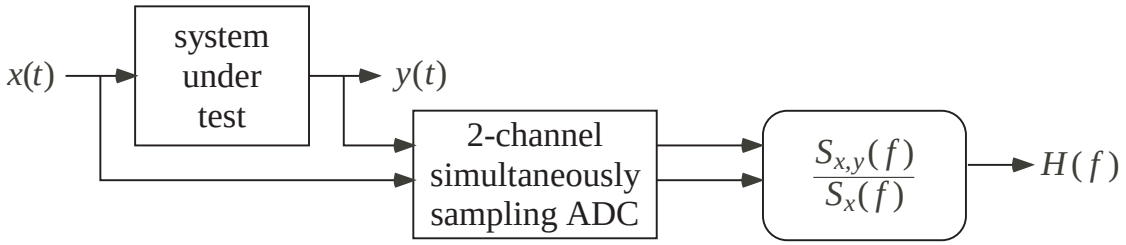

Figure 1: Calculating the system transfer function $H(f)$ in real time.

perform SI may appear in a variety of contexts, including signal processing, control systems, and communications. Indeed, students' imaginations can often be "fired up" by relating that during the Cold War, it was common (and legal) to subject "covertly obtained" Soviet electronics to a thorough SI analysis. A similar (and quite legal) process goes on today both in the military and commercial sectors on equipment from potential adversaries or business competitors.

If we approach the SI task in the time domain, we are faced with the prospect of deconvolution. For many situations, this leads to lengthy calculations and unbounded conditions. In the frequency domain, we can make use of efficient Fast Fourier Transform (FFT) algorithms and compute the power spectral density (PSD) for the output of some linear time invariant (LTI) system using the well-known[1,2] relation

$$S_y(f) = |H(f)|^2 S_x(f) \quad \text{thus} \quad |H(f)|^2 = \frac{S_y(f)}{S_x(f)} \tag{1}$$

where $S_x(f)$ and $S_y(f)$ are the PSD for the input $x(t)$ and the output $y(t)$, respectively, and $H(f)$ is the transfer function of the system.* Note that while $Y(f)/X(f) = H(f)$, we must use the PSD of the input and output to include the more general case where the input may not be deterministic but can be represented by a wide-sense stationary (WSS) random process. See the Appendix if a review of the PSD would be helpful.

Equation 1 only provides the magnitude squared of the unknown system transfer function. A method to find $H(f)$ when $x(t)$ is a random process can be shown to be

$$S_{x,y}(f) = H(f) S_x(f) \quad \text{thus} \quad H(f) = \frac{S_{x,y}(f)}{S_x(f)} \tag{2}$$

where $S_{x,y}(f)$ is the cross-PSD of $x(t)$ and $y(t)$.[3–5] The application of Equation 2 is graphically depicted in Figure 1. See the Appendix if a review of the cross-PSD would be helpful.

If $x(t)$ and $y(t)$ are stored in memory or on disk, then off-line calculation of Equation 2 may be performed entirely using software such as MATLAB (see the `TFE` command in the Signal Processing Toolbox). While this technique may show the correct $H(f)$, we have

*In common with many texts, we use the term "transfer function" to mean $H(f)$. More precisely, the transfer function is $H(s)$ or $H(z)$ and $H(f)$ is the frequency response.

found that our students are not particularly impressed with what they perceive to be a "canned demo." An alternative is to have a real-time system available as a teaching tool. For some reason (perhaps related to characteristics of the "Nintendo Generation"),[6] real-time demonstrations and student exercises have a far more profound learning effect on our students. As indicated in Figure 1, real-time SI requires simultaneously sampling the input and output of the unknown linear system.

Just how fast the system must be to perform in "real-time" has ramifications for how much of the operation needs to be performed in hardware versus software. This will be discussed further below. Regardless of how the actual TF is determined, we wish to graphically display it on the PC by taking advantage of the richness and ease of use provided by the MATLAB graphical user interface (GUI). We also wish to make the numerical coefficients of the TF available to the student via the MATLAB workspace, allowing additional processing or demonstrations. In addition to teaching our students about the TF, various concepts related to the FFT, PSD, cross-PSD, autocorrelation, crosscorrelation, and the Wiener-Khinchin theorem are conveniently reiterated by this method of performing SI. By combining this with a real-time demonstration, immediate reinforcement of the how the real world matches up with seemingly esoteric theory provides a powerful tool for the professor's "teaching kit."

3 Making Transfer Functions More "Real"

In the past, proceeding beyond a MATLAB-only simulation to a real-time hardware implementation has been impeded by a very abrupt transition, in terms of both cost and the learning curve of unfamiliar systems and software. By developing a software and hardware bridge between MATLAB and real-time DSP hardware, we have made it possible to smoothly and incrementally transition from simulation to a full hardware implementation, all while retaining the impressive capabilities of the MATLAB display engine. Using this approach, students are able to apply their knowledge of DSP to develop their own real-time transfer function evaluator if time permits.

3.1 Hardware System Requirements

Our students are already very familiar with MATLAB, but we also want them to learn more about hardware-based DSP; this topic seems to provide a perfect opportunity. Recall that a real time SI implementation implied a need to simultaneously sample $x(t)$ and $y(t)$. As a first, very inexpensive step toward the real-time goal, it is fortuitous that nearly all PC sound cards (and similar circuitry which may be included on the main PC system board) simultaneously sample the left and right audio channels. Using either the Data Acquisition Toolbox for MATLAB or a custom m-file, we can bring data from the sound card into the MATLAB workspace for processing. A typical sound card permits up to 16-bit, 44.1k samples-per-second (sps) on each channel, yielding a perfectly adequate bandwidth for audio systems.

However, the speed of the overall SI process using the sound card will be limited by several factors: 1) the speed by which one can bring frames of data from the sound card into the MATLAB workspace, 2) the time to calculate the PSD and cross-PSD using MATLAB, and 3) the time to update and display the on-screen plot of the TF.[7] To increase the speed and use the opportunity to teach students more about DSP hardware, we can perform the first two actions outside of the PC on an inexpensive DSP board.

When selecting the primary DSP hardware board, our main criteria were low cost, sufficient processing power, and a versatile software development environment. After investigating the available products, we chose to center our DSP educational platform around MATLAB and the Texas Instruments (TI) C6711 DSK, which makes use of the VLIW architecture TMS320C6711 microprocessor (clocked at 150 MHz) and basic support circuitry. We have had good results with teaching other DSP concepts using the C6x DSK, and felt that transfer function evaluation would also benefit from this approach.[8]

The C6711 DSK has the following advantages:

- it comes with an excellent software development environment (Code Composer Studio),
- it can perform roughly one billion instructions per second,
- it has plenty of memory (16 MB External SDRAM and 128 KB External Flash),
- it provides a flexible Expansion Daughter Card Interface, and
- it is relatively inexpensive ($195 academic price).

The C6711 DSK's principle disadvantage is that it only has a telephone quality (maximum sampling frequency of $f_s \approx 8$ kHz), single channel codec (the TI TLC320AD535). However, we feel the DSK's advantages out-weigh this disadvantage. The single channel 0–4 kHz bandwidth of the unmodified DSK is adequate for many simple DSP demonstrations and exercises. However, for teaching transfer function evaluation (and other topics) that require more capability, we designed a small circuit board which takes advantage of the two 80-pin connectors that make up the Expansion Daughter Card Interface on the DSK. This board includes a 12-bit four-channel, simultaneous sampling, parallel-bus analog-to-digital converter (ADC).[9] The converter is capable of operating at up to 400k samples per second (sps) on a single channel, or 400 ksps divided between a maximum of 4 channels. For transfer function evaluation, we will only need two of the four available channels, thus allowing an input bandwidth of up to 0–100 kHz. This bandwidth and 12 bits of resolution is more than adequate for our purposes here.

There is now a commercially available device very similar to our original circuit board design which also greatly enhances the input capabilities of the C6711 DSK. Texas Instruments makes the THS1206EVM (see `http://focus.ti.com/docs/prod/folders/print/ths1206.html` for more information) module, which attaches directly to the Expansion

Daughter Card Interface of the DSK. This commercial product includes a 12-bit four-channel,[†] simultaneous sampling, analog-to-digital converter (ADC) that needs only 216 mW max power. The converter is capable of operating at up to 6 Msps divided between the maximum of 4 channels. The cost at the time of this writing was approximately $100 each in small quantities.

3.2 Floating-Point or Fixed-Point?

One of the primary choices in practical DSP hardware today is the question of floting-point versus fixed-point implementations. While the floating-point ability of the TI C6711 DSK offers a pedagogical advantage (topics such as scaling and overflow may be postponed until later), the C6711 processor can also run fixed-point code if the professor so desires. This "two for the price of one" ability represents another strong advantage of the C6711 DSK in our view.

4 Desired Educational Process

We've briefly described our DSP-based hardware system that can rapidly gather samples from the input and output of an unknown system. If our students were proficient at DSP programming, they could then use this to implement the desired SI analysis algorithm. However, our students begin the course knowing how to use MATLAB, not how to program DSP chips in assembly or C (despite the help of TI's Code Composer Studio). What we needed was a tool that allowed for algorithm development in MATLAB. Once the student was comfortable with what they had learned, it would facilitate the migration of the algorithm—in part or whole—onto the DSP hardware. The desired progression would be as follows.

1. Study the traditional DSP theory,
2. use MATLAB with simulated data,
3. use MATLAB with real-world data,
4. implement the process (in part or whole) in real-time on the TI DSK hardware, and
5. repeat to improve the design or to develop new features.

The third step of this process presents a practical problem. While MATLAB now has a very capable Data Acquisition (DAQ) Toolbox that allows for direct data acquisition and data insertion into the MATLAB workspace which works with a variety of sound cards and DAQ hardware boards, it does *not* support programmable DSP systems such as a DSK. Even if

[†]Four single-ended inputs or two differential inputs may be used

the DAQ Toolbox could somehow be used with a DSK, this method would be too slow to allow the transition to step four: a true real-time implementation. Since we wish to minimize multiple software environments in the interest of time (for students and faculty), a single development environment solution is highly desirable. For this reason, we developed a direct DSK-to-MATLAB interface.

5 MATLAB to DSK Interface Software

We have written an interface between MATLAB and the DSK which is encapsulated into a generalized interface command set that supports multiple input and output channels, variable sample rates, various triggering configurations, and variable frame sizes. Files are available to support our custom ADC board, the TMS1206EVM, or the standard codec which comes with the DSK. The interface was developed using MATLAB's "mex" facility and Microsoft Visual C++, and is centered around an object that encapsulates the hardware interface between the host PC and the DSK. The TI application programming interface (API) furnished with the DSK allows operation under the Win32 target common to Windows 9x/NT and later versions. Our interface software requires that the DSK tools be installed on the computer, and that the two files `C6X_DAQ.DLL` and `DAQ_SIMUL.OUT` be placed in a MATLAB-accessible directory. At the most basic level, this interface allows a novice user to operate the DSK as a data acquisition board with a simple command sequence, with no requirement of knowing how to use Code Composer or how to program in C. Initially, all signal processing can be done in the MATLAB environment using "live" data acquired from the DSK. As the students progress, they can move processing functions from MATLAB down to the DSK by altering the DSK code (that was used to create the `DAQ_SIMUL.OUT` file), and still continue to use MATLAB as a graphical display engine.

To support advanced applications and special situations, the interface can be extended by user-defined read and write commands. These require the user to create commands at the DSK level, but do not require alterations to the MATLAB interface software (in the `C6X_DAQ.DLL` file).

6 Classroom Use

After learning the SI theory and implementing the appropriate algorithm in MATLAB (not just using the `TFE` command!), students can benefit greatly from seeing their work in action. The transfer function can be calculated either off-line or in real-time, but as noted above real-time operation seems to have a far greater (and more enduring) learning effect on students. An example of a plot created in MATLAB to show the transfer function can be seen in Figure 2, where the input $x(t)$ was Gaussian "white" noise. The actual transfer function of the same system is shown in Figure 3. It can be instructive to run the SI program on just

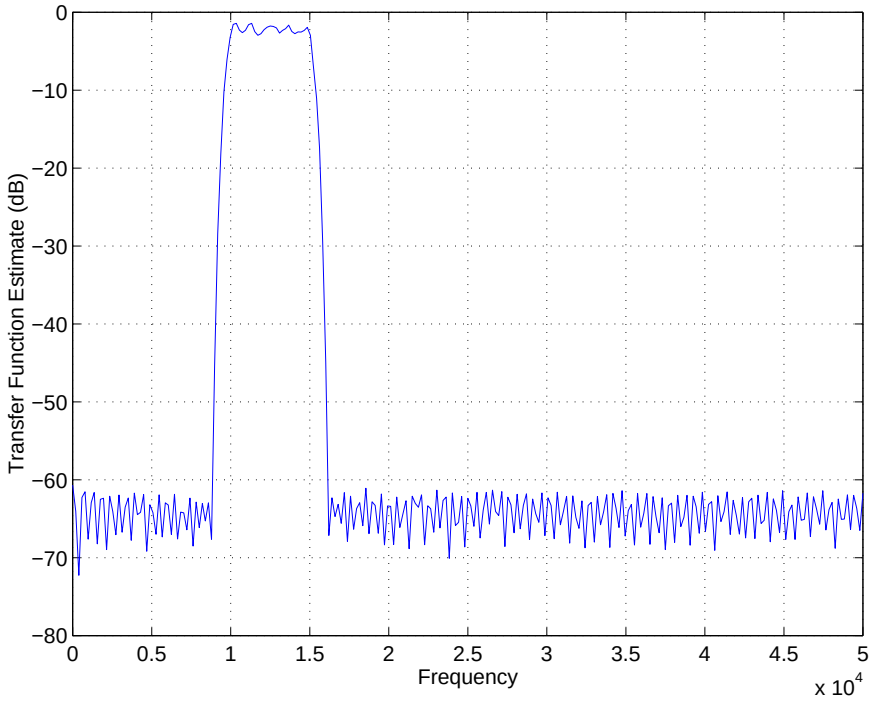

Figure 2: An example of the real-time transfer function evaluation display for a BPF with passband from 10–15 kHz. The input was simply Gaussian noise.

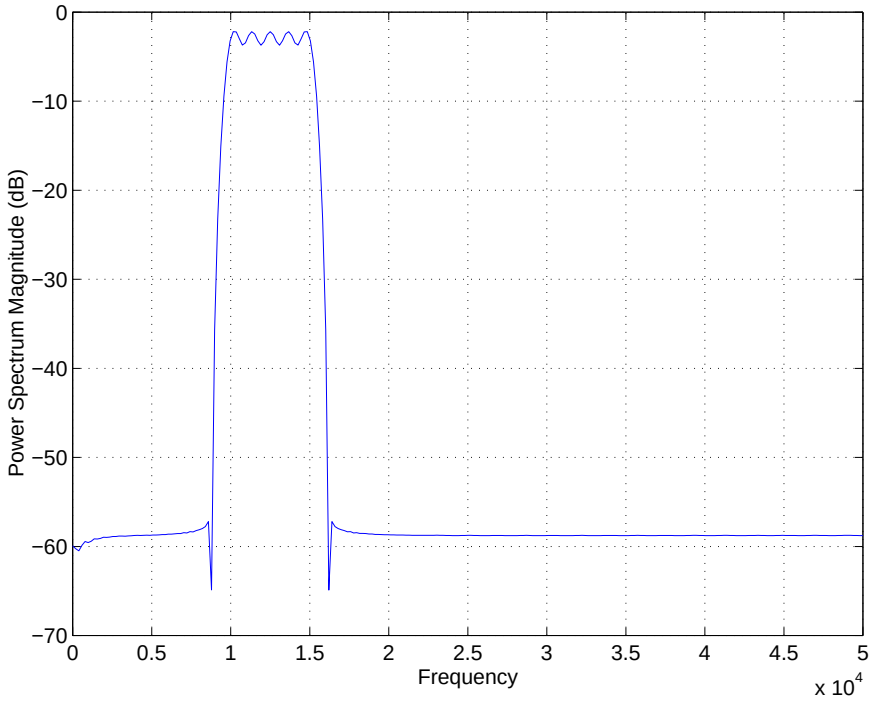

Figure 3: The actual transfer function for the same BPF from Figure 2.

the input and output of the DSK alone, with no other system connected. This shows the students the characteristics of the measuring system itself, which is a valuable lesson that applies to any instumentation situation. For an additional view of how our MATLAB to DSK interface can be used, see the paper by York et al.[10] also presented at this conference.

7 Conclusion

The demonstration platform described in this paper overcomes major obstacles to using DSPs to demonstrate concepts in real time. It integrates MATLAB closely with the powerful C6711 DSK, and eliminates the need to create individual assembly language or C programs to manipulate the hardware, while still allowing the professor the freedom to introduce the students to programming DSP processors in those languages if desired. This hardware/software platform is an excellent teaching aid and eliminates the need for the student to learn another software or hardware interface.

The authors freely distribute the software portion of this system for educational, non-profit use, and invite user comments and suggestions for improvement. This package includes files to support data acquisition using the custom ADC circuit board described above, the THS1206EVM board, and the DSK's native codec. The software may be downloaded from URL `http://www.usna.edu/EE/links/ee_links.htm` and interested parties are invited to contact the authors via e-mail.

A Appendix: Helpful mathematical relationships

Recall the Wiener-Khinchin theorem, which states the power spectral density (PSD) of some wide-sense stationary (WSS) random process $x(t)$ is the Fourier transform $\mathcal{F}\{\cdot\}$ of the autocorrelation function:

$$S_x(f) = \mathcal{F}\{R_{xx}(\tau)\} = \int_{-\infty}^{\infty} R_{xx}(\tau) e^{-j2\pi f\tau}\, d\tau$$

and the autocorrelation R_{xx} is the joint moment of $x(t_1) = x_1$ and $x(t_2) = x_2$, given by

$$R_{xx}(t_1, t_2) = E[x(t_1)x(t_2)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 p_{x_1x_2}(x_1, x_2)\, dx_1\, dx_2$$

where $p_{x_1x_2}(x_1, x_2)$ is the second-order joint pdf of x(t). Given that x(t) is WSS, $p_{x_1x_2}(x_1, x_2)$ is dependent only on the time difference between t_1 and t_2 such that

$$R_{xx}(t_1, t_2) = R_{xx}(t_2 - t_1)$$

and letting $\tau = t_2 - t_1$ we have

$$R_{xx}(\tau) = E[x(t)x(t + \tau)]$$

Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition

for $x(t)$ being WSS. If $x(t)$ is also ergodic, we can express the autocorrelation as a time average such as

$$R_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)x(t+\tau)\, dt$$

Note that a commonly used method for determining the PSD for some signal using DSP techniques is to obtain the magnitude squared of an appropriately windowed sliding FFT, a technique often referred to as Welch's periodogram method.[11,12] Other methods of higher-order spectral analysis may also be used if needed.[13]

Using the same WSS and ergodic assumptions for a random process y(t), the cross-correlation of x(t) and y(t) is

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)y(t+\tau)\, dt$$

and the cross-PSD $S_{x,y}(f)$ is given by

$$S_{x,y}(f) = \mathcal{F}\{R_{xy}(\tau)\}$$

Note that a commonly used method for determining the cross-PSD for two signals using DSP techniques is to compute an appropriately windowed sliding FFT for each signal and then multiplying the two FFT results together, a technique often referred to as Welch's periodogram method for the cross-PSD.[11,12] As with the PSD, other methods of higher-order spectral analysis may also be used if needed.[13]

References

[1] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering.* Addison-Wesley, 2nd ed., 1994.

[2] J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures.* John Wiley & Sons, 2nd ed., 1986.

[3] C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing.* Prentice Hall, 1992.

[4] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing.* Prentice Hall, 3rd ed., 1996.

[5] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach.* McGraw-Hill, 2nd ed., 2001.

[6] T. B. Welch, B. Jenkins, and C. H. G. Wright, "Computer interfaces for teaching the Nintendo generation," in *Proceedings of the 1999 ASEE Annual Conference*, (Charlotte, NC), June 1999. Paper 3532-02.

[7] T. B. Welch, E. Zivi, C. Field, and J. Rice, "Real-time data acquisition in a signals and systems course," in *Proceedings of the 2000 ASEE Annual Conference*, (St. Louis, MO), June 2000. Session 3532.

[8] M. G. Morrow, T. B. Welch, and C. H. G. Wright, "An inexpensive software tool for teaching real-time DSP," in *Proceedings of the 1st IEEE DSP in Education Workshop*, (Hunt, TX), IEEE Signal Processing Society, Oct. 2000.

[9] M. G. Morrow, T. B. Welch, C. H. G. Wright, and G. W. P. York, "Demonstration platform for real-time beamforming," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001. (accepted for publication).

[10] G. W. York, M. G. Morrow, T. B. Welch, and C. H. Wright, "Teaching real-time sonar with the C6711 DSK and MATLAB," in *Proceedings of the 2001 ASEE Annual Conference*, (Albuquerque, NM), June 2001. (accepted).

[11] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice Hall, 2nd ed., 1999.

[12] The MathWorks, Inc., Natick, MA, *MATLAB: The Language of Technical Computing*, 2000.

[13] S. M. Kay, *Modern Spectral Analysis: Theory and Application*. Prentice Hall, 1988.

CAMERON H. G. WRIGHT, Ph.D, P.E., is Associate Professor and Deputy Department Head of the Department of Electrical Engineering at the U.S. Air Force Academy, Colorado Springs, CO. His research interests include signal and image processing, biomedical instrumentation, communications systems, and laser/electro-optics applications. Lt. Colonel Wright is a member of ASEE, IEEE, SPIE, NSPE, Tau Beta Pi, and Eta Kappa Nu. E-mail: `c.h.g.wright@ieee.org`

THAD B. WELCH, Ph.D, P.E., is an Associate Professor in the Department of Electrical Engineering at the U.S. Naval Academy, Annapolis, MD (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. Commander Welch is a member of ASEE, IEEE, and Eta Kappa Nu. E-mail: `t.b.welch@ieee.org`

MICHAEL G. MORROW, P.E., is a Faculty Associate in the Department of Electrical Engineering at the University of Wisconsin, Madison, WI (from 1996-2000 he was an 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 IEEE. E-mail: `morrow@ieee.org`