Session 2548

MATLAB Exercises to Explain Discrete Fourier Transforms

Kathleen A.K. Ossman, Ph.D.
University of Cincinnati

Introduction

Digital Signal Processing is used in many applications including cellular phones, CDs, DVDs, speech recognition, pattern recognition, and control systems. Traditionally, digital signal processing (DSP) has been taught as an advanced undergraduate or graduate course in an engineering curriculum. Over the last decade, in response to the increase in industrial applications, DSP has been introduced earlier in engineering schools. However, courses in digital signal processing are noticeably absent in engineering technology programs. A recent look at ABET accredited electrical/electronics engineering technology programs [1] showed that only 6 of the 66 programs accessed offered DSP as a required course in the curriculum. Another nine programs offered DSP as a technical elective and the remaining 51 did not offer DSP to their students at all. *As DSP becomes more pervasive in industrial applications, it is imperative that engineering technology graduates have some exposure to digital signal processing theory and practice.* The main difficulty in teaching DSP to technology students is the level of mathematics. Students opening a textbook on digital signal processing [2] – [4] are faced with pages and pages of equations. In developing an introductory course in DSP for our electrical and computer engineering technology students, I have found MATLAB and SIMULINK [5] to be invaluable illustrative tools for teaching the mathematically intense concepts of DSP. This paper gives a brief overview of the required DSP course taught in the ECET department at the University of Cincinnati then details the specific MATLAB exercises used to illustrate one of the more difficult concepts in the course: discrete Fourier transforms (DFT).

Topics of Digital Signal Processing I

This introductory course in digital signal processing covers sampling and reconstructing of analog signals, convolution and correlation, finite impulse response (FIR) and infinite impulse response (IIR) digital filters, the discrete Fourier transform (DFT), and efficient computation of the DFT using fast Fourier transforms (FFT). The course goals and schedule are as follows:

GOALS

- To develop a clear understanding of the practical issues involved in sampling an analog signal and reconstructing an analog signal from a digital signal.
- To explore the advantages of digital filters over analog filters, learn to design digital filters, and implement the designs using Texas Instruments' 6x DSK boards.

- To understand what a Discrete Fourier Transform (DFT) is, how to compute a DFT efficiently using a fast Fourier Transform (FFT), and the advantages of using FFTs for computationally intensive digital processing applications.
- To explore how convolution and correlation are used in filtering and pattern recognition applications.

SCHEDULE

Week 1	Introduction and Sampling
Week 2	Number Systems: Fixed Point and Floating Point Digital Filter Considerations
Week 3	Finite Impulse Response (FIR) Filter Design
Week 4	Infinite Impulse Response (IIR) Filter Design
Week 5	Programming Filters – Circular Buffering
Week 6	Multi-rate DSP
Week 7	Discrete Fourier Transforms (DFT)
Week 8	DFT and Fast Fourier Transforms (FFT)
Week 9	Convolution
Week 10	Correlation

In the accompanying lab, MATLAB and SIMULINK along with one of Texas Instruments' DSP Starter Kits are used to investigate the topics discussed in the lecture. In previous years, Texas Instruments' TMS320C31 floating point DSP Starter Kit was used in the lab. The kit was quite adequate for our introductory course and was relatively cheap (approximately $100.00), but was recently discontinued. This year we are using Texas Instruments' TMS320C6x DSP Starter Kit which is more expensive ($395.00) but includes the Code Composer Studio software which allows students to program in C, change variables in real-time, and plot data in both the time and frequency domain. Students are required to purchase the book DSP Applications Using C and the TMS320C6x DSK [6] for the course. In addition, the textbook Digital Signal Processing: A Practical Guide for Engineers and Scientists [7] is strongly recommended as a reference text. Next year, we plan to add a second course in DSP to our curriculum as a technical elective that will focus on real-time DSP applications and architecture of DSP chips.

Discrete Fourier Transforms

One of the most difficult topics to teach in an introductory DSP course is Discrete Fourier Transforms. This section is a basic outline of how this topic is covered in lecture. The main points to understand about a Discrete Fourier Transform (DFT) are:

1. The Discrete Fourier Transform of a time signal, if done "correctly," is simply a sampled version of the spectrum of the signal over some finite frequency range.
2. The Discrete Fourier Transform is very useful in computationally intensive filtering, convolution, or correlation applications where it is often more efficient to do calculations in the frequency domain instead of the time domain.
3. The Discrete Fourier Transform can be efficiently computed using a technique called a Fast Fourier Transform.

We begin the mathematical development of the DFT with the Fourier Transform:

$$\boxed{\text{FT:}\quad \begin{aligned} x(t) &= \int X(f)e^{j2\pi ft}df \\ X(f) &= \int x(t)^{-j2\pi ft}dt \end{aligned}}$$

The signal x(t) is some general analog time signal and X(f) is the spectrum of the time signal. Both are complete representations of the signal, x. Obviously, neither one of these representations are useful for DSP since both are analog. At this point in the course, students are comfortable with the idea of sampling the time signal, x(t), possibly processing those sampled values in some manner, then reconstructing an analog signal from the sampled values. It is logical to suggest that the same thing could be done using sampled values of the spectrum rather than sampled values of the time signal. The obvious question is how to obtain sampled values of the spectrum of a signal. Sampling the analog time signal leads to the Discrete-Time Fourier Transform (DTFT):

$$\boxed{\text{DTFT:}\quad \begin{aligned} x(n) &= \int X_p(f)e^{j2\pi nf}df \\ X_p(f) &= \sum_{n=-\infty}^{\infty} x(n)e^{-j2\pi nf} \end{aligned}}$$

The time signal, x(t), must be sampled at a sufficient rate to avoid aliasing. Xp(f) is periodic but still continuous over frequency which is unsuitable for DSP. Sampling Xp(f) over one period or an integer number of periods leads to the Discrete Fourier Transform (DFT):

$$\boxed{\text{DFT:}\quad \begin{aligned} x(n) &= (1/N)\sum_{k=0}^{N-1} X_T(k)e^{j2\pi nk/N} \\ X_T(k) &= \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} \end{aligned}}$$

The signal x(n) is created by taking N samples of the original signal x(t) over some finite duration of time, D. The signal $X_T(k)$ is the DFT of the analog time signal, x(t). The previous equation shows it can be computed by taking the N samples of x(t), multiplying each sample by a weighted exponential function, then summing the results. The DFT will have N values spaced apart in the frequency domain by 1/D. The following important result is typically proved in an engineering course but simply stated in our course:

If we sample x(t) sufficiently fast for a long enough duration of time then

$$X_T(k) \approx (1/ts)X(kf_o)$$

where D = the duration of time over which x(t) is sampled
N = the number of samples of x(t)
ts = D/N = the sampling interval
Sf = 1/ts = the sampling rate or frequency
f_o = 1/D = the resolution or spectral spacing of $X_T(k)$

In other words, the DFT, which can be computed from a finite number of samples of x(t), is approximately equal to the sampled spectrum of x(t) scaled by a constant, 1/ts. The accuracy of the DFT depends on two factors: the sampling rate chosen for x(t) and the duration of time over which x(t) is sampled. The important factors to stress are

1. If x(t) is not sampled sufficiently fast, aliasing will occur causing the DFT to be a poor match to the original signal spectrum.

2. If x(t) is not sampled for a long enough duration of time, the resolution of the DFT will be poor.

The effect of sampling rate and the duration of time over which x(t) is sampled are explored using several MATLAB exercises.

MATLAB Exercises to Explain the DFT

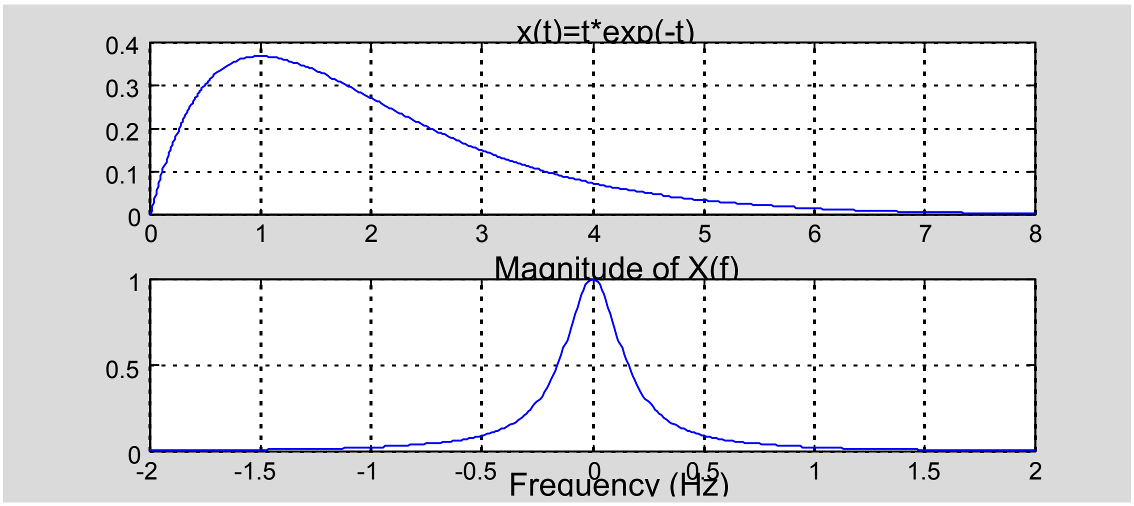
Example 1: Effect of Sampling Rate, Sf, on the DFT

In this example, the effect of sampling rate on the DFT is explored. The time signal is chosen to be a decaying exponential. The signal and its spectrum are given by:

$$x(t) = te^{-t} \quad \Leftrightarrow \quad X(f) = 1/(1+j2\pi f)^2$$

The signal and its spectrum are first plotted using MATLAB.

MATLAB Code:
```
t=0:0.01:8;
x=t.*exp(-t);
subplot(2,1,1);plot(t,x);title('x(t)=t*exp(-t)');grid
f=-2:0.01:2;
xf=1./(1+2*pi*j*f).^2;
subplot(2,1,2);plot(f,abs(xf));title('Magnitude of
X(f)');grid;xlabel('Frequency (Hz)');
```



In order to compute a DFT, the time signal x(t) must be sampled over some duration of time. According to the Nyquist Theorem, the sampling frequency should be at least twice the bandwidth of the signal. As seen from the plot of the spectrum, the spectral components above 1 Hz are very small. In order to see the effects of sampling rate, two different sampling frequencies are chosen: 2 Hz and 16 Hz. The duration of time over which the signal is sampled is chosen to be 8 seconds. The table below shows some of the key parameters for this example.

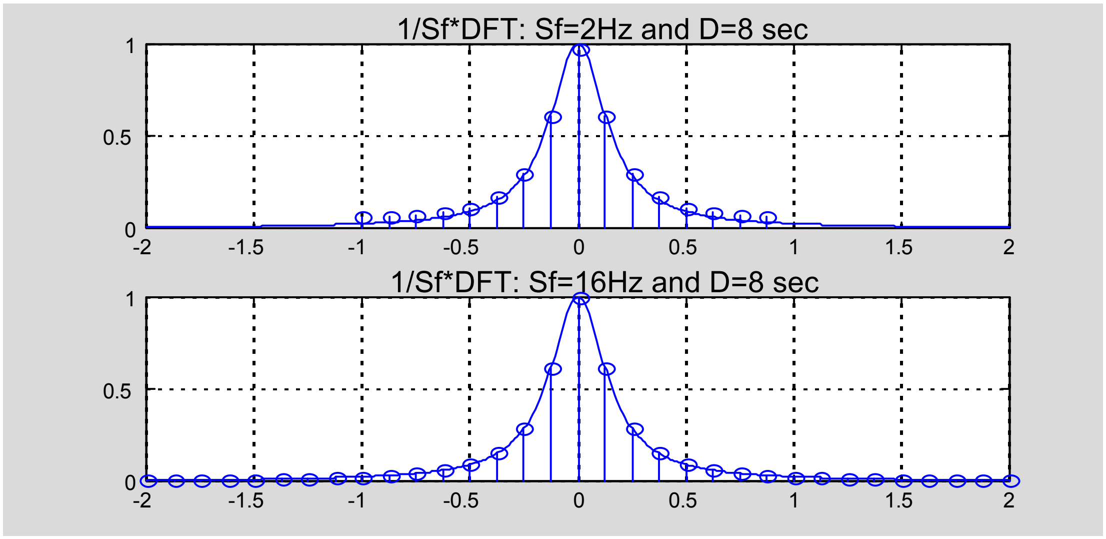
Sampling Frequency, Sf	Duration, D	Number of Samples, N	Resolution, fo	Frequency Range
2 Hz	8 sec.	16	1/8 Hz	–2 Hz to 2 Hz
16 Hz	8 sec.	128	1/8 Hz	–8 Hz to 8 Hz

An increase in sampling frequency or duration will increase the size of the DFT which in turn increases computational complexity. The following figures illustrate the effect of increasing the sampling rate on the accuracy of the DFT. At the lower sampling rate of 2 Hz, the accuracy of the DFT begins to degrade around 0.5 Hz due to aliasing. A sampling rate of 16 Hz results in a very accurate sampled spectrum. The DFT in the second case actually ranges from –8 to 8 Hz but was truncated to 2 Hz to allow a nice comparison with the lower sampling rate over the frequency range of interest.

MATLAB Code:

```
%  (a)  Sf= 2 Hz, D=8 sec, N=16 samples
Sf=2; D=8; N=16;
t=0:1/Sf: (N-1)*1/Sf;  x=t.*exp(-t);
ya=fft(x,N); ya=fftshift(ya);
fo=1/D;  %Spectral Resolution;  fa=-(N/2)*fo:fo:(N/2-1)*fo;
subplot(2,1,1);stem(fa,1/Sf*abs(ya));title('1/Sf*DFT: Sf=2Hz and D=8
sec');grid; hold on; plot(f,abs(xf)); hold off;
%   (b)  Sf=16 Hz; D= 8 sec; N=128 samples
Sf=16; D=8; N=128;
t=0:1/Sf: (N-1)*1/Sf;  x=t.*exp(-t);
yb=fft(x,N); yb=fftshift(yb);
fo=1/D;  %Spectral Resolution;  fb=-(N/2)*fo:fo:(N/2-1)*fo;
```

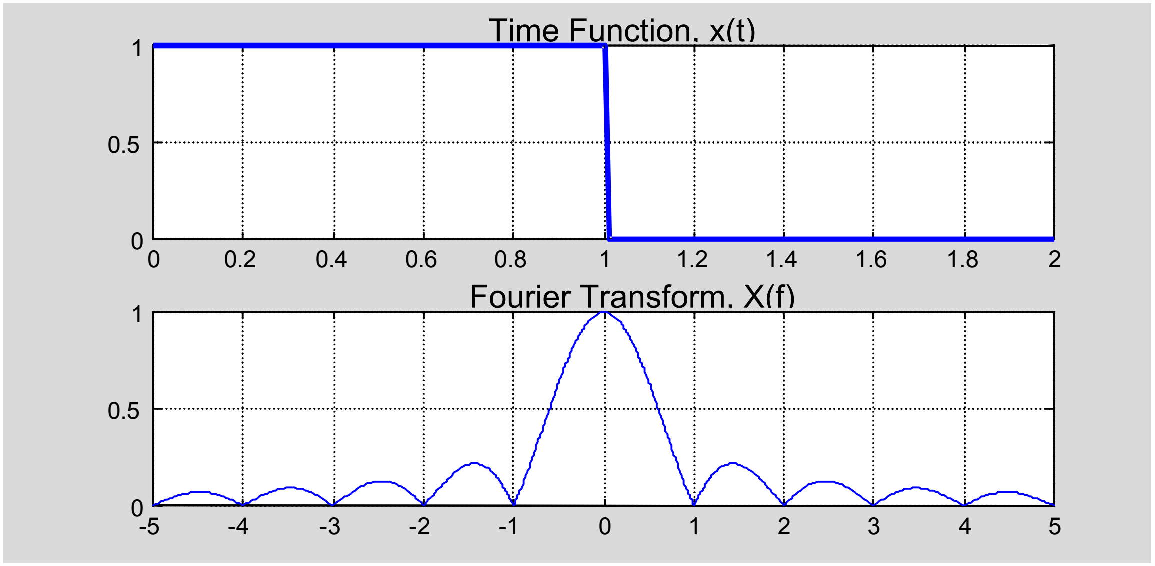
```
% Truncate the frequency range to 2 Hz.
% Find the entry in the frequency vector corresponding to -2 and +2 Hz:
f1= (-2-(-N/2*fo))/fo + 1;      % (Desired - Start)/Increment + 1
f2=(2 - (-N/2*fo))/fo + 1;
subplot(2,1,2); stem(fb(f1:f2),1/Sf*abs(yb(f1:f2)));title('1/Sf*DFT: Sf=16Hz
and D=8 sec');grid; hold on; plot(f,abs(xf)); hold off;
```



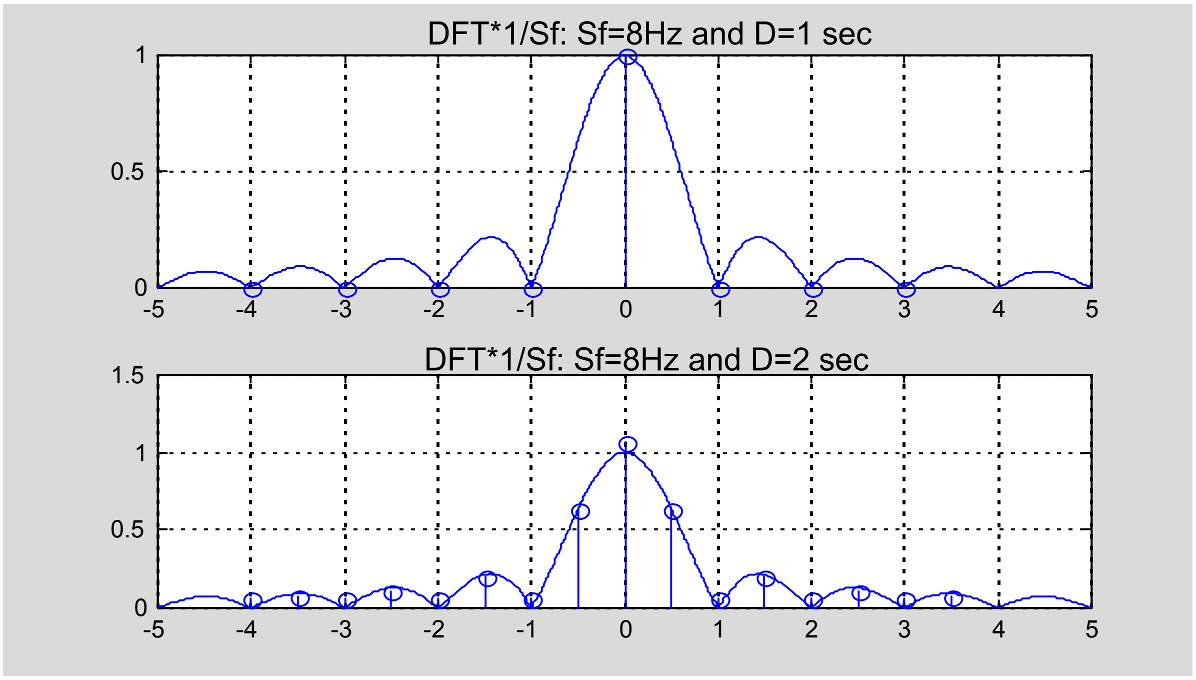
Example 2: Effect of Time Duration, D, on the DFT

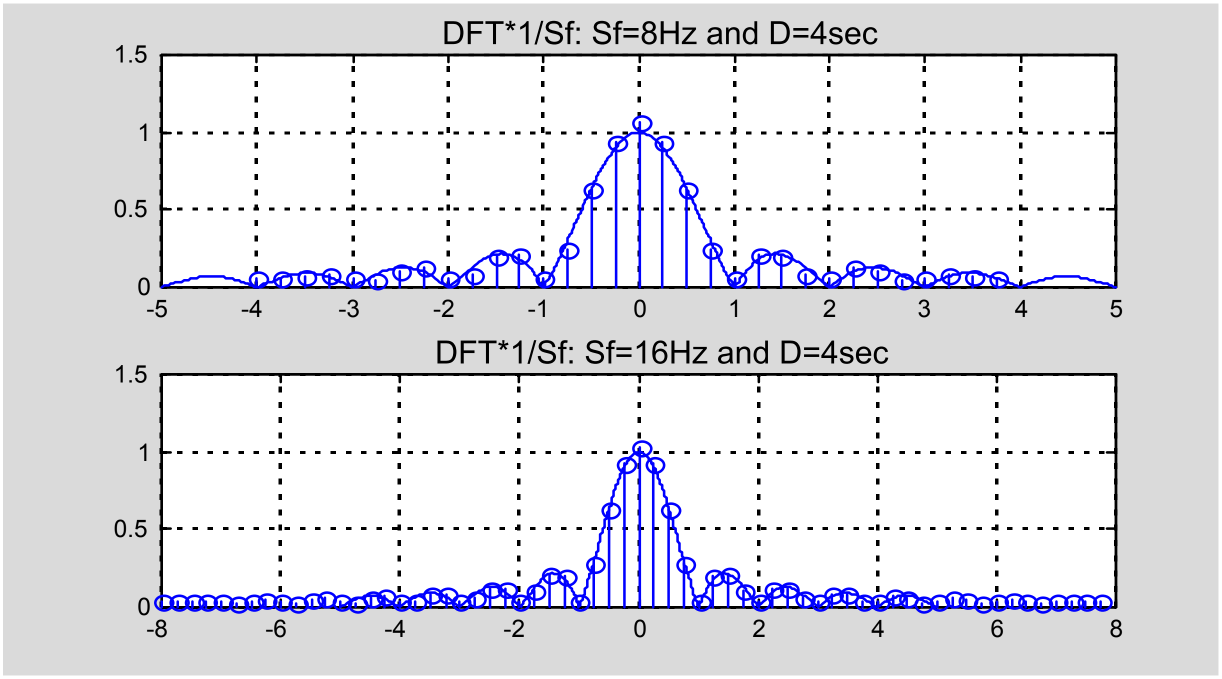
The time signal, x(t), is chosen to be a pulse of amplitude 1 with a duration of 1 second. The spectrum of this signal is a sinc function.

$$x(t) = \mathrm{rect}(t - 1/2) \quad \Leftrightarrow \quad X(f) = e^{-j\pi f}\mathrm{sinc}(f)$$



In order to compute a DFT, the time signal x(t) must be sampled over some duration of time. It is necessary to choose a sampling rate, Sf, and a duration of time, D. Several choices are explored in this example. According to the sampling theorem, the sampling rate should be at least twice the bandwidth of the signal. This particular signal is not band-limited, but we will assume an effective bandwidth of 4 Hz which would require a sampling rate of at least 8 Hz. The sampling rate is initially chosen to be 8 Hz then increased to 16 Hz. Since the signal will not be pre-filtered in this example, some aliasing will occur. The duration of time, D, is initially chosen to be 1 sec then increased to 2 sec and 4 sec to illustrate the effect on resolution. The effects of varying sampling rate and duration on the accuracy of the DFT are shown in the following figures.





The figures clearly illustrate the effect of duration, D, on the resolution or spectral spacing of the DFT. Clearly, a duration of 1 second does not yield a good sampled version of the spectrum of x(t). As D is increased, the resolution improves. The discrepancies between the DFT and the original signal spectrum illustrate the effect of aliasing. A higher sampling rate reduces the aliasing effect and produces a closer match between the DFT and the signal spectrum. The MATLAB code is similar to the previous example and will not be included here.

The preceding examples are covered in lecture. Students are then given a set of signals in the lab for which they must experiment with sampling rate and duration to achieve a good match between the DFT and the original signal spectrum. Some nice choices for these signals are:

$$x(t) = e^{-\alpha t}u(t) \Leftrightarrow X(f) = 1/(\alpha + j2\pi f)$$

$$x(t) = \mathrm{tri}(t - ½) \Leftrightarrow X(f) = \mathrm{sinc}^2(f)e^{-j\pi f}$$

$$x(t) = \cos(2\pi\alpha t) \Leftrightarrow X(f) = 0.5[\delta(f + \alpha) + \delta(f - \alpha)]$$

Teaching Strategies and Student Comments

The major goal in this course is to explain the practical aspects of Digital Signal Processing to engineering technology students while avoiding the mathematical theory where possible. All concepts discussed in the course are accompanied by illustrative examples and lab exercises using MATLAB, SIMULINK, and the TMS320C6x DSK. Over the last couple of years, I have observed my students grasping the concepts of aliasing, filtering, finite wordlength effects, and spectral analysis as they complete exercises and experiments in the lab. The experiments students seem to enjoy the most are on FIR and IIR filtering. Students are required to pick filters, design the filters using the Filter Design and Analysis Tool in MATLAB, then implement and test the filters on the TMS320C6x DSK. These labs really highlight the versatility of a DSP chip. Student evaluations of both the lecture and the laboratory have been very positive.

Conclusion

The increasing use of DSP in engineering applications makes it necessary to include digital signal processing courses in the engineering technology curriculum. MATLAB and SIMULINK are valuable tools for illustrating mathematically intense DSP concepts. Some of the MATLAB exercises used in lecture and lab to explain the important issues of sampling rate and resolution for the discrete Fourier transform are discussed in this paper.

Bibliography

[1] http://www.abet.org/accredited_programs/TACWebsite.html

[2] Discrete-Time Signal Processing, Oppenheim, Schafer, and Buck, Prentice Hall, 1999.

[3] Analog and Digital Signal Processing, Ambardar, Brooks/Cole Publishing Co., 1999.

[4] Digital Signal Processing: A Computer Based Approach, Mitra, McGraw-Hill, 2001.

[5] MATLAB and SIMULINK, The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760.

[6] DSP Applications Using C and the TMS320C6x DSK, Chassaing, John Wiley and Sons, Inc. 2002.

[7] Digital Signal Processing: A Practical Guide for Engineers and Scientists, Smith, Elsevier Science, 2003.

Biography

Dr. Kathleen Ossman is an assistant professor in the Electrical and Computer Engineering Technology Department at the University of Cincinnati. She received a BSEE and MSEE from Georgia Tech in 1982 and a Ph.D. from the University of Florida in 1986. Her interests include feedback control systems and digital signal processing.