1996

Incorporating MATLAB's Signal Processing Toolbox into a DSP course at an undergraduate E.E. program

Sol Neeman Ph.D.
Johnson & Wales University - Providence, sneeman@jwu.edu

Incorporating MATLAB's Signal Processing Toolbox into a DSP course at an undergraduate E.E. program

Sol Neeman
School of Technology
Providence, RI 02903

Abstract

This paper provides some suggestions for incorporating MATLAB's Signal Processing Toolbox into a DSP course. Often, in a DSP course, students have difficulties understanding abstract and non-intuitive concepts and seeing their relevance to the practical part of their curriculum. The tools offered by MATLAB's Signal Processing Toolbox, can help to make these concepts more tangible and provide a perspective for students.

Some basic tools relevant to an undergraduate DSP course will be introduced, including examples of tool use and graphic results. The tools presented will be applied in the areas of synthesis and analysis of signals, FFT computation, impulse response and convolution, frequency response, the Z-transform, filter design and filter applications.

Introduction

One of the engineering courses that makes extensive use of mathematical models in the E.E. program is the course in DSP. The mathematical tools often are crucial to providing insight into topics which are otherwise not available. This presents difficulties for some students since the math models require one step up in the level of abstraction needed. Also, the "time constant" needed to "absorb" and understand the models at a level that enables students to use them with confidence, varies sharply.

The computational and graphic tools offered by MATLAB Signal Processing Toolbox, can help reduce this "time constant" and provide a hands-on tool so students can practice their use at their own pace. These tools can be integrated into the curriculum to cover the main topics in a DSP course at an undergraduate level:

1. Discrete signals, DFT and aliasing.
2. Response of LTI systems in the time domain.
3. LTI systems, frequency response and geometric evaluation of a transfer function.
4. The different forms of representing LTI systems.
5. Digital filters.

Analysis and synthesis of discrete signals, DFT, the sampling theorem and the phenomena of aliasing

The DSP Toolbox offers a set of commands for synthesizing various basic discrete signals: sine, square, triangle, unit step, impulse, exponential and sync functions. These signals can be synthesized using a common time index and saved for future use. More complex signals can be synthesized by combining the basic signals.

FFT computations can be illustrated and practiced on signals previously synthesized. In general, the length of the sequence should be chosen to be a power of 2 so that radix 2 FFT can be used without the need for zero padding. Also care should be taken with the indices since MATLAB's indices begin with 1 rather than 0. It is convenient to use lower case letters to represent sequences in the time domain and upper case letters to represent the corresponding frequency components.

First the relation between Bin # and the actual frequency should be explained in the FFT computation; this relation is given by:

$$f=(\text{bin \#}-1)*fs/N \qquad \text{For bin \#= 1 to } N/2+1$$

where fs is the sampling frequency, N is the number of points in the sequence(N chosen to be even) and f the actual frequency.
Thus bin # 1 through bin # (N/2 +1) correspond to the frequency range dc to fs/2.

As an example, consider the following real sequence and its fft computed by MATLAB:

```
x=[4 3 7 -9 1 0 0 0]        % define a sequence of length 8
X=fft(x)                    % compute and display its fft
```

which results in:

```
X=
        6.0
       11.48   -2.75i
       -2.0             -12.00i
       -5.48   +11.24i
       18.00
       -5.48   -11.24i
```

```
-2.0              +12.00i
11.48    +2.75i
```

For *fs*=1000Hz, the first value, 6.0, corresponds to dc and the Th value, 18.00, corresponds to 500Hz, the Nyquist frequency. The last three values correspond to negative frequencies and for real sequences they are complex conjugates of the first half.

When graphing FFT values versus frequency in Hz, a frequency vector should be defined and the negative portion of the computed FFT values should be truncated:

```
Hz=(fs/2)*(0:N/2)/(N/2)  % freq. vector dc to fs/2
                         % of N/2 +1 points
X(N/2 +2: N)=[];         % truncate negative portion
```

Then both the magnitude and the phase of the frequency spectrum can be graphed vs the vector *Hz*.

Illustrating the phenomenon of aliasing

Often students have difficulties seeing the importance and implications of the sampling theorem. The square wave can provide an interesting illustration of the phenomenon of aliasing. Since it is not bandlimited, any discrete representation of a square wave will result in aliasing[3]. Students can compare the predicted aliased components to those viewed in the spectrum of a square wave to get an insight into this phenomenon (see fig. 1.)

Synthesized noisy signals can demonstrate to students how powerful the Fourier transform is in detecting a signal buried in noise. As an example, generate a noisy 50Hz sine wave:

```
t=0:0.001 : 1.023;       % Define a time index
                         %      of 1024 points w/
fs=1000Hz
xn=sine(2*pi*50*t) + 5*rand(1,length(t))
                  % add noise
```

the fft of the noisy signal can now be computed and presented graphically. The 50Hz component is clearly distinguishable in the frequency spectrum of the signal (see fig. 2)

2.Convolution and the response of LTI systems in the time domain

The output of an LTI system to an input is given in the time domain by the convolution of the input and its impulse response. The commands "conv" and "deconv", convolute and deconvolute two signals; the latter can be used to compute the impulse response of a system from a set of input/output.

First, students can compute the convolution of short sequences without the help of the computer and verify the results using MATLAB. Next, synthesized signals can be used to represent inputs and the impulse response of a system and the output can be computed using convolution. Similarly, synthesized signals can be used to represent input/output sequences and the impulse response can be computed by MATLAB.

3.The transfer function of an LTI system, its frequency response, poles and zeros and geometric evaluation of a transfer function

The frequency response of an LTI system can be computed and graphed using the coefficients of the polynomials of the transfer function. In MATLAB, the frequency response can be evaluated at n point (preferably n chosen to be power of 2) equally spaced at the upper half of the unit circle, 0 to PI(which correspond to the frequency range 0 to fs/2).

If vectors b and a contain the coefficients of the polynomials in the numerator and the denominator, the following commands would compute and display both the magnitude and the phase of the frequency response evaluated at 128 points and displayed versus the actual frequency:

```
[h,w]=frequency(b,a,128);
Hz=w*fs/2*pi;
subplot(2,1,1), plot(Hz,abs(h))
subplot(2,1,2),plot(Hz,angle(h)
```

Geometric evaluation of a transfer function

The geometric location of the poles and zeros of a transfer function can provide qualitative and quantitative information about the characteristics of the frequency response of a filter[4].

The commands "roots" and "zplane" can be used to graphically represent the poles and zeros on the unit circle in the Z plane.

Students can compare their prediction of the frequency response of a filter (using the geometry of the poles/zeroes) to that generated by MATLAB. This can help develop an insight into the relation between the two (see fig. 3 for the frequency response and the geometry of the poles and zeros in a butterworth HPF)

4.Various forms of LTI system representations

MATLAB provides various mathematical models to represent discrete LTI systems: polynomial transfer function form, factored form, partial fraction expansion form and state space form. MATLAB also provides the commands needed to convert one form to another.

The polynomial transfer function form is the z transform of the difference equation which describes the system in the time domain:

$H(Z)=Y(Z)/X(Z)=$
$=b_0 + b_1 z^{(-1)} + .. + b_M z^{(-M)} / a_0 + a_1 z^{(-1)} + .. + a_N z^{(-N)}$

In MATLAB, the transfer function is represented by two row vectors which contain the coefficients of the two polynomials.

Thus the transfer function:

$$H(Z)=3 + 2z^{(-1)} + 4z^{(-2)} / 1 + 2z^{(-1)} + 3z^{(-2)} + 4z^{(-3)}$$

is represented in MATLAB by:

```
b=[3 2 4];
a=[1 2 3 4];
```

This representation is convenient for deriving the frequency response of the system as well as the poles and zeros and their geometric location.

The factored form of the transfer function is given by:

$$H(Z)=k\,(z-z_0)(z-z_1)...(z-z_M) / (p-p_0)..(p-p_N)$$

where $z_0, z_1, .., p_0, p_1, ...$ are the zeros and poles of the transfer function. This form is convenient for implementing the system in the cascade form and directly provides the values of the zeros/poles which are useful for stability evaluation.

The conversion from the polynomial form to the factored form can be done using the command "root". For the above example, the following will provide the zeros and poles in two column vectors and the constant k as a scalar, respectively:

```
q=roots(b)
q=
        -0.3333 + 1.1055i
        -0.3333 - 1.1055i

p=roots(a)
p=
        -1.6506
        -0.1747 + 1.5469i
        -0.1747 - 1.5469i

k=b(1)/a(1)
k=
        3
```

The reverse derivation, that of the polynomials' coefficients from the zeros and poles of the transfer function, can be done using the command "poly":

```
k*poly(q)
        3 2 4

poly(p)
        1 2 3 4
```

Similarly, conversion between polynomial form and partial fraction form can be done using the "residue" command, conversion from transfer function form to state space form can be done using the command "tf2ss" and conversion from factored form to state space form via the command "zp2ss".

5.Digital Filters

MATLAB provides both a set of built in classical filters as well as the tools to design both FIR and IIR filters. Once the relation between the frequency response and the geometry of the poles and zeros of a transfer function are understood, students can construct various filters (low pass, high pass, etc. with various roll off) then use synthesized data to verify the action of the filter.

As an example, consider the following comb filter:

$$H(z)=1-z^{(-k)} / 1-(1/k)\, z^{(-k)} \qquad k=\text{positive integer}$$

This filter will reject the frequency fs/k and all its harmonics(as well as the dc component)[4].

Suppose we need to remove the dc component, the 50Hz component and its harmonics from a signal sampled at a frequency of 1000Hz. Since fs=1000Hz, we need to choose k=20, so that fs/k=50Hz.

Then we can synthesize a signal, say a sine wave of 125Hz, to which we add the following undesired components: dc, 50Hz, 100Hz and 150Hz(since the sampling frequency is 1000Hz, we should make sure all the components we add have frequency components less than fs/2=500Hz). The signal then is filtered using the above mentioned comb filter, using both the commands "filter" and "filtfilt", the latter of which minimizes the startup transient effect.

```
b=[1 zeros(1,19)-1];      % num.coef. of comb filter
a=[1 zeros(1,19)-1/20];   % den.coef.. of comb filter
[H,w]=freqz(b,a,128)      % compute freq. response
t=0:.001:1.023; %define a time index
x=sin (2*pi*125*t) +      % synthesize signal
  1+sin(2*pi*50*t) +      % undesired components
 .5*sin(2*pi*100*t) +
```

```
.25*sin(2*pi*150*t);
y=filter(b,a,x);              %  filter the data
z=filtfilt(b,a,x)
```

Fig. 4 shows the frequency response of the comb filter, and the filtered signal using "filtfilt".

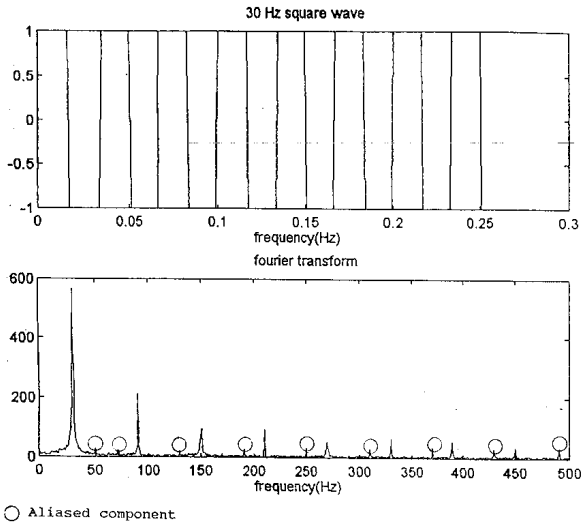

Figure1. A 30Hz synthesized square wave and its frequency spectrum.

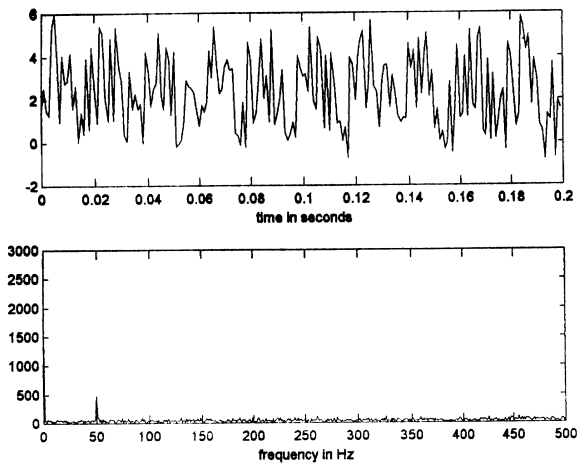

Figure2. A noisy 50Hz synthesized sine wave and its frequency spectrum.

Figure2. A noisy 50Hz synthesized sine wave and its frequency spectrum.

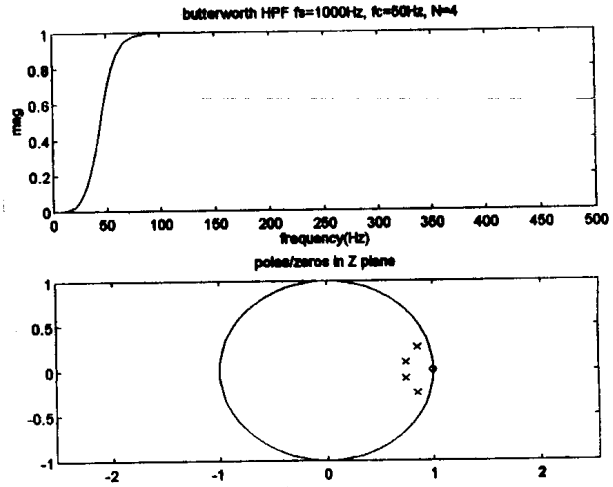

Figure3. The frequency response and the geometry of the poles and zeros in abutterworth HPF.

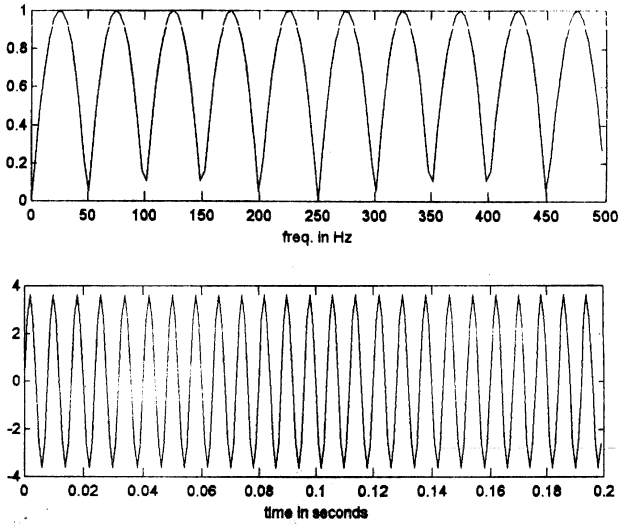

Figure4. A comb filter and the filtered signal using "filtfilt".

References

1. The MathWorks, Inc., "MATLAB Reference Guide", 1992.

2. The MathWorks, Inc., "Signal Processing Toolbox User's Guide", 1992.

3. Neeman, S. , "Using MATLAB to illustrate the phenomenon of aliasing", *ASEE 95 Annual Conf. Proc.*

4. Leland B. Jackson, " Digital Filters and Signal Processing", 2nd edition, *Kluwer Academic Publishers* (pp 47-50, 94-97).