

1 Aim

1. To review some fundamental concepts about signals and systems.
2. To revise and/or acquire and develop skills in using MATLAB simulations.
3. Digital low pass filtering.
4. Practice ZT.
5. Investigate operation of sampling/A2D.

2 MATLAB and Signals as Vectors

As we have discussed in lectures, signals can be regarded as vectors living in a vector space. This means we can use the powerful tools from linear algebra to help us process signals. The MATLAB software package is built on a base of linear algebra algorithms; indeed, MATLAB is short for MATrix LABoratory.

We now do some work with MATLAB to create and display signals, and “verify” the vector nature of signals.

MATLAB EXERCISE 2.1

1. Create a folder in your directory for CLAB1.
2. Start MATLAB and set the working directory to your CLAB1 folder.
3. Create a sinusoidal signal of frequency 7 rad/sec of duration 10 sec with a sampling time $T_s = 0.015$ sec by typing into your MATLAB command window the following:

```
>> t = 0:0.015:10;  
>> x = sin(7*t);  
>> plot(t,x);
```

The first command creates a vector t of sample times. You can find out how long it is by typing

```
>> length(t)
```

The second command creates a vector x of samples of the signal, while the third command creates the graphic object you can see - a plot of the signal in the time domain.

4. Now create a second sinusoid, of the same duration and sampling time, but of frequency 200 rad/sec:

```
>> n = sin(200*t);
>> plot(t,n);
```

5. Create a signal y as the linear combination

$$y = x + 0.5n$$

and plot it (MATLAB uses the operator $*$ for multiplication). As you can see, this is again a signal.

Your MATLAB workspace now has four objects, t, x, n, y . Keep these, as they will be used below.

3 Using MATLAB's DFT

The MATLAB command `fft` is used to calculate the DFT of a signal. Using MATLAB's help system, find out the syntax and operation of the `fft` command.

From theory, we know that the spectrum of

$$x(t) = \sin(\omega_0 t) = \frac{1}{2j}(e^{j\omega_0 t} - e^{-j\omega_0 t})$$

is

$$X(\omega) = \pi j[\delta(\omega + \omega_0) - \delta(\omega - \omega_0)]$$

QUESTION 3.1 1. Verify this expression for the spectrum.

2. Sketch the spectrum.

Let's look at what the DFT algorithm does.

MATLAB EXERCISE 3.2

1. Type in the following commands to compute the DFT of the signal x :

```
>> fftx = fft(x);
>> mx = abs(fftx); px = unwrap(angle(fftx));
>> f = (0:length(x)-1)/length(x);
>> plot(f,mx);
```

Note that we have extracted the magnitude and phase of the spectrum (though we have only plotted the magnitude here). We also have used a normalised frequency scale on the horizontal axis (1 corresponds to the sampling frequency).

2. Hmm, why are there two peaks, at the locations you can see?

3. Plot the phase. Refer to the help system for the meaning of the `unwrap` command.

4. Similarly, plot the DFT spectrum of the signals n and y .

5. Summarise the main features you observe about the spectra.

The DFT is not a perfect tool for calculating the spectrum of a sampled continuous time signal. The DFT uses a finite length sequence, or aperture, or window, through which to view the signal. Let's look at this a bit.

MATLAB EXERCISE 3.3

1. To reveal more about the spectrum, let's switch to a decibel scale, as follows:

```
>> mxdb=20*log(mx);  
>> plot(f,mxdb);
```

2. What do you notice? How well does this approximate the ideal spectrum?
3. Let's see what happens if we increase the size of the window:

```
>> t1 = 0:0.015:100;  
>> x1 = sin(7*t1);  
>> fftx1 = fft(x1);  
>> mx1 = abs(fftx1); px1 = unwrap(angle(fftx1));  
>> f1 = (0:length(x1)-1)/length(x1);  
>> mx1db=20*log(mx1);  
>> figure;  
>> plot(f1,mx1);
```

4. Which case results in improved resolution?
-

4 ZT Calculations

QUESTION 4.1 *Using the definition of ZT, calculate the ZT of the following signals:*

1. *Unit impulse*

$$x[n] = \delta[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0 \end{cases}$$

2. *Unit step function*

$$x[n] = u[n] = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n \geq 0 \end{cases}$$

3. *Delated unit impulse*

$$x[n] = \delta[n - n_0] = \begin{cases} 1 & \text{if } n = n_0 \\ 0 & \text{if } n \neq n_0 \end{cases}$$

4. *Exponential (positive time)*

$$x[n] = \alpha^n u[n] = \begin{cases} \alpha^n & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases}$$

where α is a complex number.

5. Sine function

$$x[n] = \sin(\omega_0 n)$$

6. Cosine function

$$x[n] = \cos(\omega_0 n)$$

7. Rectangular window of length N

$$x[n] = w[n] = \begin{cases} 1 & \text{if } 0 \leq n \leq N - 1 \\ 0 & \text{if } n < 0 \text{ or } n \geq N \end{cases}$$

5 Noise Reduction 1

The example $y = x + 0.5n$ can be thought of as “signal” x , plus “noise” $0.5n$. A common problem in signal processing is the removal or reduction of noise. How do we do it? I guess it depends on the nature of the signal and the noise.

The spectrum of the signal y suggests a simple fix in this case: pass it through a low pass filter, with a suitably chosen cut-off frequency.

There are many ways to design digital filters. Here we look at a simple first order digital low pass filter, defined by the *transfer function*:

$$H(z) = \left(\frac{1-a}{2} \right) \frac{z+1}{z-a}$$

where $0 \leq a < 1$. The 3dB (cut-off) frequency is approximately

$$\omega_c = 1 - a \text{ rad/sample}$$

Here, frequency is normalised: $0 \leq \omega < 2\pi$ (sampling frequency being 2π rad/sample).

The signal x has frequency 7 rad/sec, while the noise has frequency 200 rad/sec; these normalise to $7T_s = 0.105$ rad/sample and $200T_s = 3$ rad/sample.

Choose $\omega_c = 0.2$ rad/sample, to give $a = 0.8$, and the transfer function becomes

$$H(z) = \left(\frac{1-0.8}{2} \right) \frac{z+1}{z-0.8} = \frac{z+1}{10z-8}$$

The filter can be specified in MATLAB using the `filter` command. The `tf` command is also useful for defining a transfer function object. Look up the syntax and operation of the `filter` and `tf` using MATLAB’s help system.

MATLAB EXERCISE 5.1

1. Filter the signal using the command:

```
>> b=[1 1]; a=[10 -8];  
>> yf=filter(b,a,y);
```

2. Obtain plots of the filtered signal yf in both the time and frequency domains. How effective was the noise reduction?

3. It is also useful to obtain the Bode diagram of the filter, as follows:

```
>> lpf=tf([1 1],[10 -8],1);
>> bode(lpf)
```

4. Use the Bode diagram to determine the actual 3dB point.
-

You may wish to try out the MATLAB tools `sptool` and `fdatool` for signal analysis and design.

6 Noise Reduction 2

Let's look at another noise reduction problem. The previous example was straightforward because the signal and noise were disjoint in the frequency domain. What can be done if the signal and noise spectra are not disjoint?

Consider a signal x and let now the noise n be a standard white Gaussian noise. The power spectral density of n is constant across all frequencies (hence the name "white"), and therefore not disjoint with the signal $x(t) = \sin(7t)$. White noise is commonly used to model noise, such as sensor noise, in DSPC applications.

MATLAB EXERCISE 6.1

`sigpwgn1.m`

1. Download the file `sigpwgn1.m` into your working MATLAB folder.
2. Run the program by typing in your MATLAB command window:

```
>> sigpwgn1
```

3. Look closely at the signal in the time and frequency domains, and describe what you see. Compare with the previous example.
 4. Consider how the signal might be filtered to remove the noise.
-

Later in the course we will use the *Kalman filter* to reduce the effect of additive white Gaussian noise.

As you can see from Matlab Exercise 6.1, it is very convenient to use files to store MATLAB commands. The file `sigpwgn1.m` is an example of a *script file* or *m-file*. MATLAB m-files end in a `.m`

You can view and edit script files using the MATLAB m-file editor. To open the m-file `sigpwgn1.m`, in the main MATLAB window,

File – > Open...

and select `sigpwgn1.m` in the file browser. The file will open in the m-file editor. Examine the file and notice how the MATLAB commands are used.

You might like to write some m-file scripts for the earlier exercises.

7 Sampling and Analog to Digital Conversion

QUESTION 7.1 Find out and describe in one page how sampling and conversion to digital format (A2D) can be achieved.