Session 2

Discrete time signal

2.1 Generating Discrete Time Signals

2.1.1 The "linspace" Function

All directly generable signals that rely on mathematical functions such as trigonometric functions, exponential functions, logarithmic functions, hyperbolic functions … etc need what we call a time base signal. To understand what the time base signal is and why it is required, keep in mind that you cannot compute the value of a sine without knowing the value of its input argument.

If we are talking about sinusoidal signals, for instance, the signal is computed as follows:

$$x(t) = A\sin(2\pi ft + \theta)$$

A is the amplitude, f is the frequency, and θ is the phase shift in radians. In order to evaluate the value of the signal at any time instant, you must substitute with the value of t along with the values of the parameters. When we want to generate a signal, we have to compute the signal at a set of values of t corresponding to the time interval of interest. Thus, t should be a vector not just a single value.

This vector is called the time base signal. It contains the values of time at which the sine should be computed. The values of time in the time vector start and end at the limits of the interval in which we are interested. The difference between these two values is called the duration of the signal.

The total number of points in the time base signal (N_t) satisfies the relation:

$$N_t = \text{duration} \times \text{sample rate}$$

Thus, if we want to generate a sine wave in the time interval from 2 to 10 seconds with a sample rate of 100 samples per second, the time base signal must consist of 800 points whose values increment uniformly from 2 to 10.

The linspace function is used in generating the time base signal. The linspace function has the

syntax shown below:

$$t = linspace(a, b, N)$$

The first and second input arguments (a and b) are the initial and final values of the generated signal. The third argument (N) is the number of points of the signal. Thus, we must have N=(a-b)fs.

As an example, the following command could be used to generate a time base signal for a sine wave in the time interval from 3 to 5 seconds with a sample frequency of 100 samples per second.

```
>>t=linspace(3,5,(5-3)*100);
```

2.1.2 Direct Generation of Elementary Signals

DC Signal

A DC signal may be easily generated using the "zeros" or "ones" functions. The syntax of both functions is the same. This syntax is shown below.

$$x1 = zeros(1, N)$$
$$x2 = ones(1, N)$$

The "zeros" function generates a 1×N vector of zeros. The "ones" function generates a 1×N vector of ones. Using basic operations, we could generates DC signals of any other value, as the following examples demonstrate.

Examples

The first example below shows an instruction used to generate a DC signal of value -5 and sample rate 100 Hz in the time interval from -3 to 3 seconds using the "zeros" function. The second example shows an instruction used to generate a DC signal of value 17.3 and sample rate 20 Hz in the time interval from 0 to 4 seconds using the "ones" function. Note that generating a DC signal does not need a time base signal, because the signal is not a function in time.

```
>>x=zeros(1,600)-5;
>>x=17.3*ones(1,80);
```

Ramp Signal

A ramp signal is a signal whose value increases or decreases uniformly with time. A ramp signal has the following general form:

$$y(t) = at + b$$

The parameter *a* is called the slope, and the parameter *b* is called the intercept of the signal.

Example

In the following example, we are generating a ramp signal of sample rate 1000 Hz, slope -2 volts/second and intercept -3 volts in the time range from -2 seconds to 4 seconds.

```
>>t=linspace(-2,4,6*1000);
>>x=-2*t-3;
```

Polynomial Signals

Ramp signals are a special case of polynomial signals. In the following example, we will see how a polynomial signal of higher degree may be generated.

Example

The polynomial signal we want to generate a polynomial signal that follows the equation:

$$x(t) = 3t^3 - 11t^2 + 7$$

We want to generate this signal in the time range from 0 to 1 second with a sample rate of 1000 Hz. The following instructions generate this signal.

```
>>t=linspace(0,1,1000);
>>x=3*t.^3-11*t.^2+7;
```

Polynomial signals require a time base signal to be generated, because they are computed as functions in time.

Trigonometric, Hyperbolic, Exponential, and Logarithmic Signals

Now, we will discuss how to generate trigonometric, hyperbolic, exponential, and logarithmic signals. These groups of signals are generated similarly, so we will give one example from each group. However, we will list the functions that are used to generate the signals in each group.

Trigonometric functions

sin, cos, tan, asin, acos, atan, sec, csc, cot, asec, acsc, acot.

Hyperbolic functions

sinh, cosh, tanh, asinh, acosh, atanh, sech, csch, coth, asech, acsch, acoth.

Exponential functions
exp
Logarithmic functions
log, log10

Examples

In the following examples, we will generate the following six signals. All signals have an interval of interest from 1 to 5 seconds, and a sample rate of 150 Hz.

$$y_1(t) = 3\cos(4\pi t + \pi/4)$$
$$y_2(t) = 4\sinh(3.5t + 2)$$
$$y_3(t) = 4e^{-0.5t}$$
$$y_4(t) = 2^{-0.6t+0.3}$$
$$y_5(t) = \ln(3/t)$$
$$y_6(t) = \log(0.6t)$$

```
>>t=linspace(1,5,600);
>>y1=3*cos(4*pi*t+pi/4);
>>y2=4*sinh(3.5*t+2);
>>y3=4*exp(-0.5*t);
>>y4=2.^(-0.6*t+0.3);
>>y5=log(3./t);
>>y6=log10(0.6*t);
```

2.1.3 Indirect Generation

Indirect generation usually involves generating a signal by analyzing the signal into elementary segments that may be generated directly, and then conjoining the individual segments using concatenation. The following example shows the generation of the signal shown in figure 2.3.

The sample rate of the signal is 1000 Hz. The first segment is a DC signal of value 1 from -3 to -1 seconds. The second segment is half a cycle of a cosine wave of frequency 1/4 Hz (because half a cycle occupies 2 seconds), amplitude 4, phase shift 0, and DC offset 1. The third segment is an instant of the first segment. The following instructions may be used to generate this signal.

```
>>y1=ones(1,2000);
>>t=linspace(-1,1,2000);
>>y2=4*cos(2*pi*t/4);
>>y=[y1 y2 y1];
```

2.2 The Plot Command

The plot command syntax is described below.

>> plot (vector1 , vector2 , line specification string)

The plot command operates as follows. It plotting by marking the position of the points on the graph. Every point is defined by an x coordinate in vector1 and a y coordinate in vector2. The x coordinate of the first point is vector1(1) and its y coordinate is vector2(1). The x coordinate of the nth point is vector1(n) and its y coordinate is vector2(n). Then, Matlab connects every point with the point before it and the point after it using straight lines. If the points are close enough to one another, the breaking of straight lines at their joints will be invisible and the plot will appear smoothly curved. You can repeat the example using linspace with 10 points instead of 100 to see the difference.

The appearance of the curve can be controlled through the line specification string. The line specification string defines three things:

The line color, The line type, The point marker shape.

For example, a line specification string ‘r--p' means that the line color is red, the line is dashed and that the marker shape is pentagonal.

2.2.1 Plotting More than One Curve On the Same Graph

If you try to plot a cosine curve after the previous sine plot, you will realize that the cosine curve will replace the sine curve. The sine curve will be lost. What about if we want to plot many curves on the same graph?

The “hold on instruction tells Matlab to hold all current plots on the graph and superimpose any new curve onto the existing ones. Typing :

```
>> hold on
```

lets Matlab hold the existing curves. Any plot command issued after “hold on will not cause Matlab to delete old plots.

To quit holding current plots, type:

```
>> hold off
```

Note:

To plot a curve on another figure, you can use the function figure.

```
>>figure
```

2.2.2 Using “subplot

The function “subplot allows you to create multiple axis boxes on the same figure. The following shows a figure created using “subplot

2.3 Operation On Sequences:

Here we briefly describe basic sequence operations and their MATLAB equivalents.

1. **Signal addition**: This is a sample-by-sample addition given by

$$\{x_1(n)\} + \{x_2(n)\} = \{x_1(n) + x_2(n)\}$$

It is implemented in MATLAB by the arithmetic operator "+". How-ever, the length ofx1(n) andx2(n)must be the same. If sequences are of unequal lengths, or if the sample positions are different for equal-length sequences, then we cannot directly use the operator +

```
% implements y(n) = x1(n)+x2(n)
% -----------------------------
%   y = sum sequence over n, which includes n1 and n2
%  x1 = first sequence over n1
%  x2 = second sequence over n2 (n2 can be different from n1)
%
n = min(min(n1),min(n2)):max(max(n1),max(n2));  % duration of y(n)
y1 = zeros(1,length(n)); y2 = y1;               % initialization
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;     % x1 with duration of y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;     % x2 with duration of y
y = y1+y2;                                      % sequence addition
```

2. **Signal multiplication**: This is a sample-by-sample (or "dot") multiplication) given by

$$\{x_1(n)\} \cdot \{x_2(n)\} = \{x_1(n)x_2(n)\}$$

It is implemented in MATLAB by the array operator .*. Once again, the similar restrictions apply for the .* operator as for the + operator.

```
n = min(min(n1),min(n2)):max(max(n1),max(n2));  % duration of y(n)
y1 = zeros(1,length(n)); y2 = y1;               %
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;     % x1 with duration of y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;     % x2 with duration of y
y = y1 .* y2;                                   % sequence multiplication
```

3. **Scaling**: In this operation each sample is multiplied by a scalar α.

$$\alpha\{x(n)\} = \{\alpha x(n)\}$$

An arithmetic operator (*) is used to implement the scaling operation in MATLAB.

4. **Shifting**: In this operation, each sample of $x(n)$ is shifted by an amount k to obtain a shifted sequence $y(n)$.

$$y(n) = \{x(n-k)\}$$

If we let $m = n-k$, then $n = m+k$ and the above operation is given by

$$y(m+k) = \{x(m)\}$$

Hence this operation has no effect on the vector `x`, but the vector `n` is changed by adding k to each element. This is shown in the function

```
% implements y(n) = x(n-k)
% -------------------------
```

```
n = m+k; y = x;
```

EXAMPLE: Generate and plot each of the following sequences over the indicated interval.

a. $x(n) = 2\delta(n+2) - \delta(n-4), \quad -5 \leq n \leq 5.$

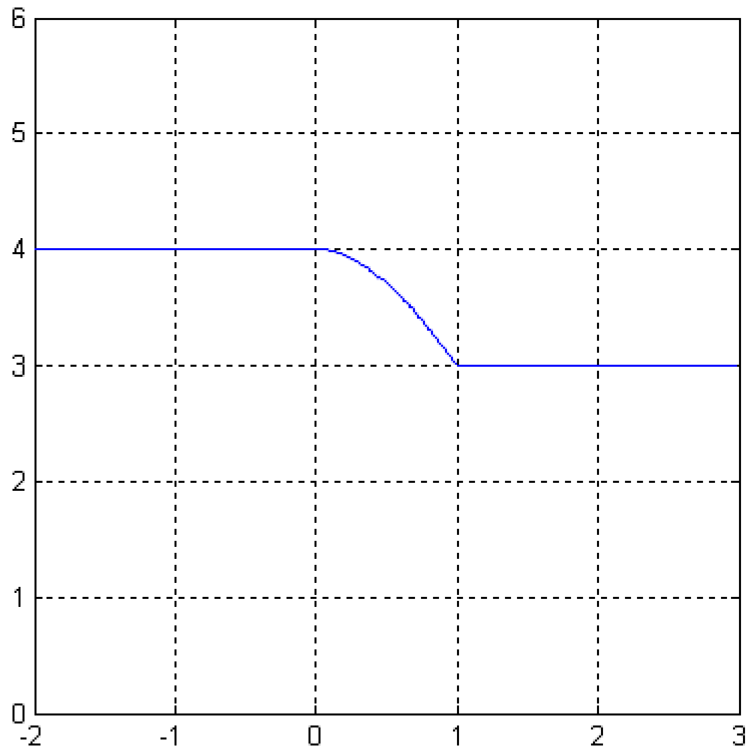
```
>> n = [-5:5];
>> x = 2*impseq(-2,-5,5) - impseq(4,-5,5);
>> stem(n,x); title('Sequence in Problem 2.1a')
>> xlabel('n'); ylabel('x(n)');
```

b. $x(n) = n\,[u(n) - u(n-10)] + 10e^{-0.3(n-10)}\,[u(n-10) - u(n-20)],\ 0 \leq n \leq 20.$

```
>> n = [0:20]; x1 = n.*(stepseq(0,0,20)-stepseq(10,0,20));
>> x2 = 10*exp(-0.3*(n-10)).*(stepseq(10,0,20)-stepseq(20,0,20));
>> x = x1+x2;
>> subplot(2,2,3); stem(n,x); title('Sequence in Problem 2.1b')
>> xlabel('n'); ylabel('x(n)');
```

Exercises 2:

1. Write down the Matlab instructions that will generate the signal depicted below. This signal consists of a DC segment from -2 to 0 seconds, a quarter cycle of a sinusoidal wave from 0 to 1 seconds, and another DC segment from 1 to 3 seconds. The sample rate is 100 Hz.



2.

Two discrete time sinusoidal signals are $T = 2s$

$$x[nT] = \cos\left(\frac{2n}{3}\right).$$
$$y[nT] = \cos\left(\frac{8\pi n}{38}\right).$$

i. Plot $x[n]$ and $y[n]$ for $|t| < 40$ seconds. Plot the corresponding continuous-time sinusoid from which the samples are taken

ii. Are the sequences periodic (Hint: use stem)? if so what is the period and how many cycles of the continuous time signal are in one period?.