

Project #1

Graphical Representations of Continuous and Discrete Time Signals

In this project we will demonstrate how to display continuous and discrete time signals in Matlab. Let the signal that we will work with be a cosine for both cases. For the continuous time case, let us create a time axis first.

```
» t=0:0.005:1; MCL 1
```

We can now form our cosine signal as follows.

```
» xt=cos(2*pi*t); MCL 2
```

Can you tell the frequency of this cosine signal?

Finally, we can use the *plot* command to see our signal.

```
» plot(t,xt);grid MCL 3
```

At this point, Matlab pops up a figure window¹ with the plot of our cosine signal on it (Figure 1.1). We observe that Matlab takes care of axis scaling, ticking and tick labeling for us, automatically. We used the *grid* command² at MCL 3 to add grid lines to our plot. We can toggle the grid state (on and off) by issuing consecutive grid commands.

Try zooming into several parts of your cosine plot by issuing the *zoom* command at the prompt or by clicking the magnifier icon on the figure toolbar. What do you observe when you deeply zoom into the regions of the cosine where its slope is high? Have you seen that line segments approximate our smooth looking cosine? We note that by using the plot utility we just get the feeling of continuous time, as it connects consequent data points with line segments. When

¹ If there is one or more figure windows already open, Matlab uses the current figure window to make the plot. It is possible to open many figure windows and switch among them by using the *figure* command. Type “help figure” at Matlab prompt for more information.

² We will use the terms *command*, *utility*, and *function* interchangeably. Matlab commands or command lines are formed by using or calling one or more of Matlab functions or utilities.

the data points are finely spaced, depending on screen size and resolution, we may not even recognize those broken lines.

Actually, `xt` signal or vector that we created in Matlab contains just the samples of the cosine signal. Can you tell how many samples of the cosine signal did we get, in total and in per second sense?

Now change the time axis created in MCL 1 with a coarser one, as below.

```
» t=0:0.1:1;
```

Then, redo the remaining part by reissuing MCL 2 and MCL 3. What do you observe? Also, issue the following line, where we show a more complex use of the plot command, and try to understand/interpret the picture that you will get.

```
» plot(t,xt,t,xt,'or');grid
```

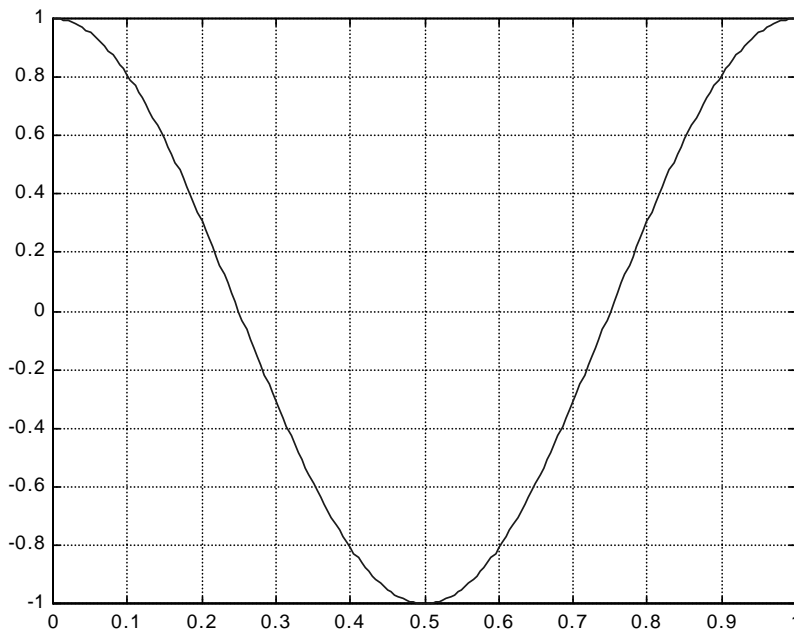


Figure 1.1 A “continuous time looking” cosine signal.

To demonstrate the visualization of discrete time signals in Matlab, we create a vector of indices first.

```
> n=0:20;
```

MCL 4

We can now form our discrete time cosine signal as follows.

```
> xn=cos(2*pi*n/20);
```

MCL 5

We now use the *stem* command to see our signal.

```
> stem(n,xn);grid
```

MCL 6

Below is how our signal looks (Figure 1.2).

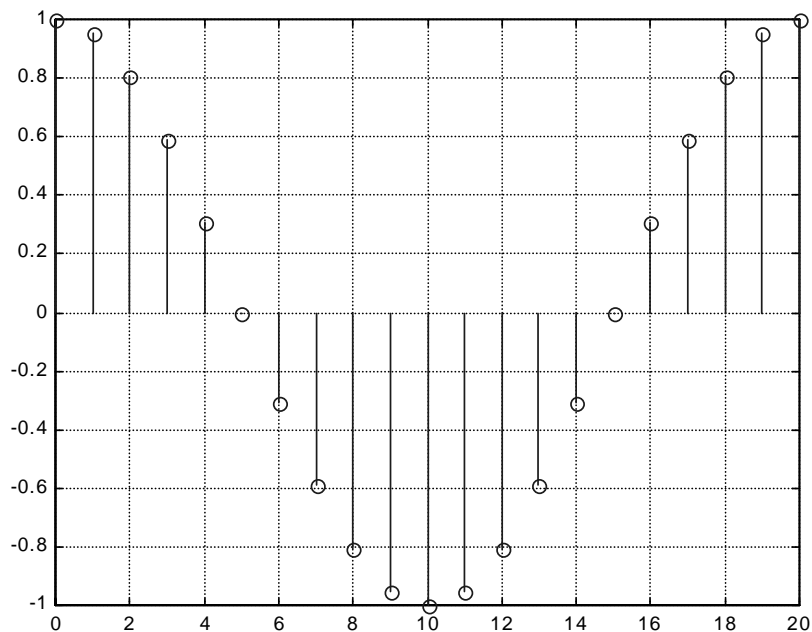


Figure 1.2 A discrete time cosine signal.

The stem command plotted our data points as lines originating or stemming from the horizontal axis (x-axis) at the index points. This type depiction is consistent with the fact that discrete time signals are defined only at the index points.

Now change the frequency of our discrete time cosine signal by modifying MCL 5 as follows.

```
» xn=cos(2*pi*n);
```

Then redo the stem plot by reissuing³ MCL 6. How do you explain this interesting plot/picture?

³ You can scroll through the Matlab commands that you have entered or issued before by using up and down arrow keys. To reuse or reissue a command that you used before, locate it by scrolling and hit enter/return. For more information see section 2.5 of the Matlab Tutorial.