Chapter 2

2.1 Mathematical functions

MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.

Typing `help elfun` and `help specfun` calls up full lists of *elementary* and *special* functions respectively.

There is a long list of mathematical functions that are *built* into MATLAB. These functions are called *built-ins*. Many standard mathematical functions, such as $\sin(x)$, $\cos(x)$, $\tan(x)$, e^x, $ln(x)$, are evaluated by the functions `sin`, `cos`, `tan`, `exp`, and `log` respectively in MATLAB.

Table 2.1 lists some commonly used functions, where variables `x` and `y` can be numbers, vectors, or matrices.

Table 2.1: Elementary functions

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

In addition to the elementary functions, MATLAB includes a number of predefined

constant values. A list of the most common values is given in Table 2.2.

Table 2.2: Predefined constant values

`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

2.1.1 Examples

We illustrate here some typical examples which related to the elementary functions previously defined.

As a first example, the value of the expression $y = e^{-a}\sin(x) + 10\sqrt{y}$, for $a = 5$, $x = 2$, and $y = 8$ is computed by

```
>> a = 5; x = 2; y = 8;
>> y = exp(-a)*sin(x)+10*sqrt(y)
y  =
   28.2904
```

The subsequent examples are

```
>> log(142)
ans   =
      4.9558

>> log10(142)
ans   =
      2.1523
```

Note the difference between the natural logarithm `log(x)` and the decimal logarithm (base 10) `log10(x)`.

To calculate $\sin(\pi/4)$ and e^{10}, we enter the following commands in MATLAB,

```
>> sin(pi/4)
ans   =
      0.7071

>> exp(10)
ans   =
      2.2026e+004
```

NOTES:

- Only use built-in functions on the right hand side of an expression. Reassigning the value to a built-in function can create problems.
- There are some exceptions. For example, `i` and `j` are pre-assigned to $\sqrt{-1}$. However, one or both of `i` or `j` are often used as loop indices.
- To avoid any possible confusion, it is suggested to use instead `ii` or `jj` as loop indices.

2.2 Basic plotting

2.2.1 overview

MATLAB has an excellent set of graphic tools. Plotting a given data set or the results of computation is possible with very few commands. You are highly encouraged to plot mathematical functions and results of analysis as often as possible. Trying to understand mathematical equations with graphics is an enjoyable and very efficient way of learning mathematics. Being able to plot mathematical functions and data freely is the most important step, and this section is written to assist you to do just that.

2.2.2 Creating simple plots

The basic MATLAB graphing procedure, for example in 2D, is to take a vector of x-coordinates, $\mathbf{x} = (x_1, \ldots, x_N)$, and a vector of y-coordinates, $\mathbf{y} = (y_1, \ldots, y_N)$, locate the points (x_i, y_i), with $i = 1, 2, \ldots, n$ and then join them by straight lines. You need to prepare x and y in an identical array form; namely, x and y are both row arrays or column arrays of the *same* length.

The MATLAB command to plot a graph is `plot(x,y)`. The vectors $\mathbf{x} = (1, 2, 3, 4, 5, 6)$ and $\mathbf{y} = (3, -1, 2, 4, 5, 1)$ produce the picture shown in Figure 2.1.

```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot(x,y)
```

NOTE: The `plot` functions has different forms depending on the input arguments. If `y` is a vector `plot(y)`produces a piecewise linear graph of the elements of `y` versus the index of the elements of `y`. If we specify two vectors, as mentioned above, `plot(x,y)` produces a graph of `y` versus `x`.

For example, to plot the function $\sin(x)$ on the interval $[0, 2\pi]$, we first create a vector of x values ranging from 0 to 2π, then compute the *sine* of these values, and finally plot the result:

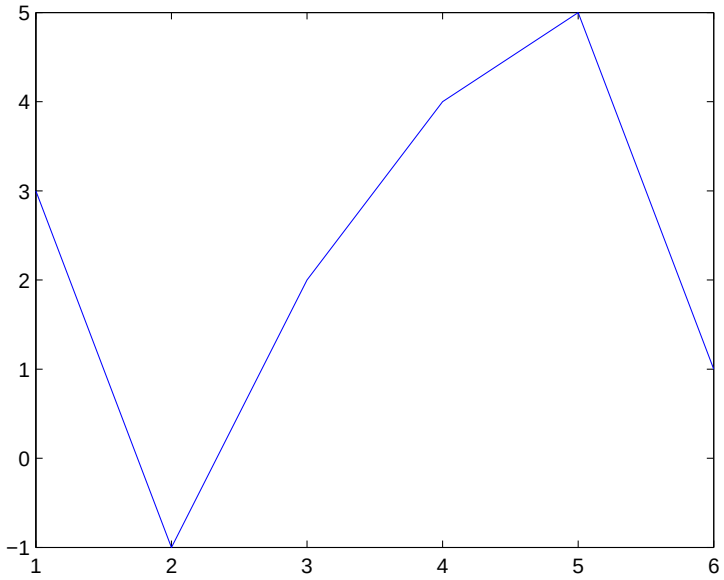

Figure 2.1: Plot for the vectors x and y

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```

NOTES:

- `0:pi/100:2*pi` yields a vector that
  - starts at 0,
  - takes steps (or increments) of $\pi/100$,
  - stops when 2π is reached.
- If you omit the increment, MATLAB automatically increments by 1.

2.2.3 Adding titles, axis labels, and annotations

MATLAB enables you to add axis labels and titles. For example, using the graph from the previous example, add an x- and y-axis labels.

Now label the axes and add a title. The character `\pi` creates the symbol π. An example of 2D plot is shown in Figure 2.2.

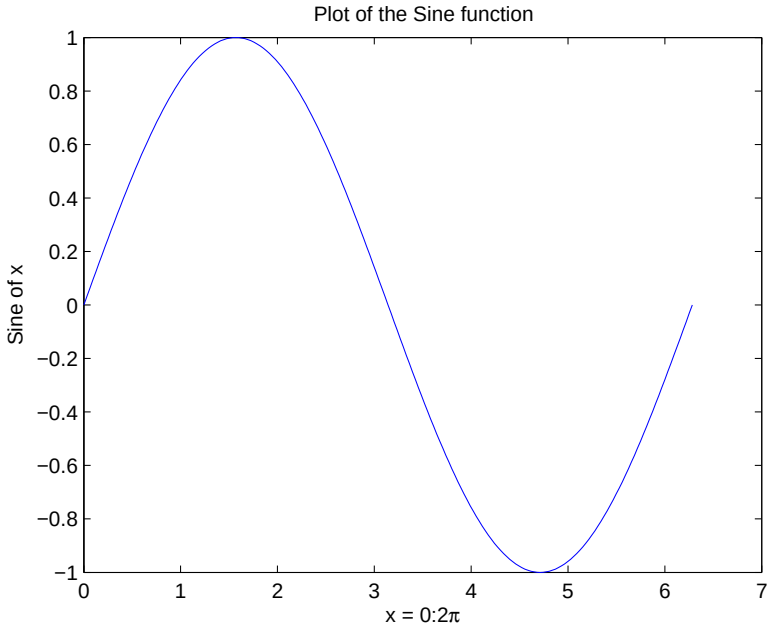

Figure 2.2: Plot of the Sine function

```
>> xlabel('x = 0:2\pi')
>> ylabel('Sine of x')
>> title('Plot of the Sine function')
```

The color of a single curve is, by default, blue, but other colors are possible. The desired color is indicated by a third argument. For example, red is selected by `plot(x,y,'r')`. Note the single quotes, ' ', around `r`.

2.2.4 Multiple data sets in one plot

Multiple (x, y) *pairs* arguments create *multiple* graphs with a single call to *plot*. For example, these statements plot three related functions of x: $y_1 = 2\cos(x)$, $y_2 = \cos(x)$, and $y_3 = 0.5 * \cos(x)$, in the interval $0 \leq x \leq 2\pi$.

```
>> x = 0:pi/100:2*pi;
>> y1 = 2*cos(x);
>> y2 = cos(x);
>> y3 = 0.5*cos(x);
>> plot(x,y1,'--',x,y2,'-',x,y3,':')
>> xlabel('0 \leq x \leq 2\pi')
>> ylabel('Cosine functions')
>> legend('2*cos(x)','cos(x)','0.5*cos(x)')
```

```
>> title('Typical example of multiple plots')
>> axis([0 2*pi -3 3])
```

The result of multiple data sets in one graph plot is shown in Figure 2.3.

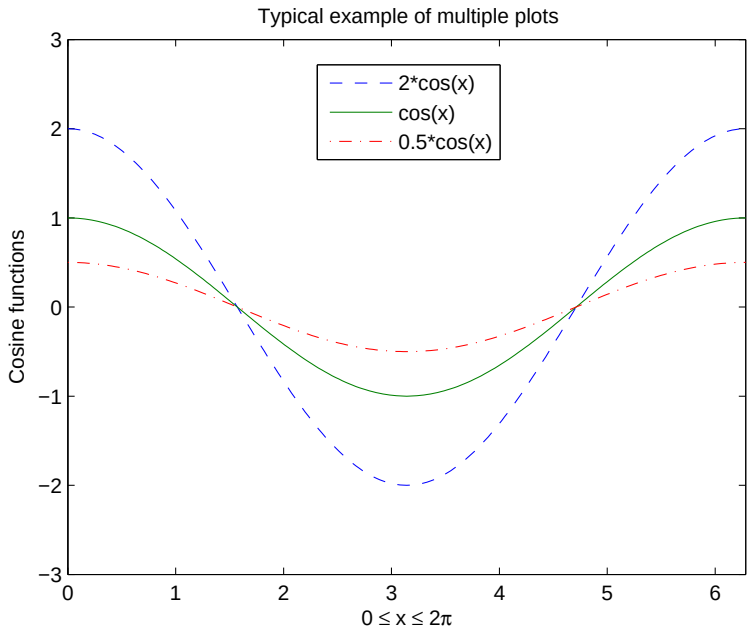

Figure 2.3: Typical example of multiple plots

By default, MATLAB uses *line style* and *color* to distinguish the data sets plotted in the graph. However, you can change the appearance of these graphic components or add annotations to the graph to help explain your data for presentation.

2.2.5 Specifying line styles and colors

It is possible to specify *line styles*, *colors*, and *markers* (e.g., circles, plus signs, ...) using the `plot` command:

```
plot(x,y,'style_color_marker')
```

where `style_color_marker` is a *triplet* of values from Table 2.3.

To find additional information, type `help plot` or `doc plot`.

Table 2.3: Attributes for `plot`

Symbol	Color	Symbol	Line Style	Symbol	Marker
`k`	Black	$-$	Solid	$+$	Plus sign
`r`	Red	$--$	Dashed	o	Circle
`b`	Blue	:	Dotted	$*$	Asterisk
`g`	Green	$-.$	Dash-dot	.	Point
`c`	Cyan	none	No line	$\times$	Cross
`m`	Magenta			s	Square
`y`	Yellow			d	Diamond

2.3 Exercises

Note: Due to the teaching class during this Fall Quarter 2005, the *problems* are *temporarily* removed from this section.