

Introduction to MATLAB

Dr. Philip Loizou

MATLAB is a high-level programming language that has been used extensively to solve complex engineering problems. The language itself bears some similarities with ANSI C and FORTRAN.

MATLAB works with three types of windows on your computer screen. These are the *Command window*, the *Figure window* and the *Editor window*. The *Figure window* only pops up whenever you plot something. The *Editor window* is used for writing and editing MATLAB programs (called M-files) and can be invoked in Windows from the pull-down menu after selecting **File | New | M-file**. In UNIX, the *Editor window* pops up when you type in the command window: `edit filename` ('filename' is the name of the file you want to create).

The command window is the main window where you communicate with the MATLAB interpreter. The MATLAB interpreter displays a command `>>` indicating that it is ready to accept commands from you.

For example, enter the following 3x3 matrix:

```
>>A=[1,2 3; 4,5,6; 7 8 10]
```

After you press the <Enter> key, MATLAB responds:

```
A=
```

```
    1    2    3
    4    5    6
    7    8   10
```

You can manipulate this matrix using standard linear algebra techniques. For instance, to invert this matrix, enter

```
>> B=inv(A)
```

Similarly, you can define arrays or vectors. To create an array having as elements the integers 1 through 6, just enter:

```
>> x=[1,2,3,4,5,6]
```

Alternatively, you can use the `:` notation, i.e.,:

```
>> x=1:6
```

The `:` notation above means create a vector starting from 1 TO 6, in steps of 1. If you want to create a vector from 1 to 6 in steps of say 2, then use:

```
>> x=1:2:6
```

```
Ans =
```

```
    1    3    5
```

You can use the command window as a calculator, or you can use it to call other MATLAB programs (M-files).

Say you want to evaluate the expression: $a^3 + \sqrt{bD} - 4c$, where $a=1.2$, $b=2.3$, $c=4.5$ and $D=4$. Then in the command window, type:

```
>> a = 1.2;
>> b=2.3;
>> c=4.5;
>> D=4;
>> a^3+sqrt(b*D)-4*c
ans =
    -13.2388
```

Arithmetic Operations

There are four different arithmetic operators:

- + addition
- subtraction
- * multiplication
- / division (for matrices it also means inversion)

In addition, there are:

- .* multiplication of two vectors, element by element
- ./ division of two vectors, element-wise

The arithmetic operators + and – can be used to add or subtract matrices, scalars or vectors.

Example:

```
>> X=[1,3,4]
>> Y=[4,5,6]
>> X+Y
ans=
    5 8 10
```

For the vectors X and Y the operator + adds the elements of the vectors, one by one, assuming that the two vectors have the same dimension. In the above example, both vectors had the dimension 1x3, i.e., one row with three columns. An error will occur if you try to add a 1x3 vector to a 3x1 vector. The same applies for matrices.

To compute the **dot product** of two vectors, you can use the multiplication operator *
For the above example, it is:

```
>> X*Y'
ans =
    47
```

Note the single quote ' after Y. The single quote denotes the transpose of a matrix or a vector.

To compute a point-to-point multiplication of two vectors, you can use the .* operator:

```
>> X .* Y
ans =
    4    15    24
```

Array indexing

In MATLAB, all arrays (vectors) are indexed starting with 1, i.e., y(1) is the first element of the array y. Note that the arrays are indexed using parenthesis (.) and not square brackets [.] as in C/C++.

You can allocate memory for one-dimensional arrays (vectors) using the **zeros** command. The following command allocates a 100-dimensional array:

```
>> Y=zeros(100,1);
>> Y(30)
ans =
    0
```

Similarly, you can allocate memory for two-dimensional arrays (matrices). The command
>> Y=zeros(4,5)
defines a 4 by 5 matrix.

Relational operators

SYMBOL	MEANING
<=	Less than equal
<	Less than
>=	Greater than equal
>	Greater than
==	Equal
~=	Not equal

Logical operators

SYMBOL	MEANING
&	AND
	OR
~	NOT

Special characters and MATLAB functions

Symbol	Meaning
pi	p (3.14....)
sqrt	indicates square root, e.g., sqrt(4)=2
^	indicates power (e.g., 3^2=9)
abs	Absolute value . e.g., abs(-3)=3
NaN	Not-a-number obtained when computing mathematically undefined operations, such as 0/0
Inf	Represents $+\infty$
;	Indicates the end of a row in a matrix. It is also used to suppress printing
%	Denotes a comment. Anything to the right of % is ignored by the MATLAB interpreter and is considered as comments.
'	Denotes transpose of a vector or matrix. It's also used to defined strings, e.g., str1='DSP';
X=zeros(1,N)	Creates a vector x with N zeros, i.e., X=[0, 0 ,0 ,...,0]
X=ones(1,N)	Creates a vector x with N ones, i.e., X=[1, 1, 1, ... 1]
:	<p>Used for generating vectors and also in for-loops.</p> <p>Example: >> X=1:4 ans = 1 2 3 4</p> <p>The : notation created a vector from 1 to 4, in steps of 1. To create a vector from 1 to 5 in steps of 2:</p> <p>Example 2: >> X=1:2:5 ans = 1 3 5</p>

Control flow

MATLAB has the following flow control constructs:

- **if** statements
- **switch** statements
- **for** loops
- **while** loops
- **break** statements

The **if**, **for**, **switch** and **while** statements need to terminate with an **end** statement.

Examples:

IF:

```
if x>0
    str='positive';
elseif x<0
    str='negative';
elseif x==0
    str='zero';
else
    str='error';
end
```

WHILE:

```
x=-10;
while x<0
    x=x+1;
end
```

FOR loop:

```
X=0;
for i=1:10
    X=X+1;
end
```

The above code computes the sum of all numbers from 1 to 10.

BREAK:

The break statement lets you exit early from a **for** or **while** loop:

```
x=-10;
while x<0
    x=x+2;
    if x == -2
        break;
    end
end
```

Plotting

You can plot arrays using MATLAB's function: **plot**

The function **plot** (.) is used to generate line plots. The function **stem**(.) is used to generate “picket-fence” type of plots.

Example:

```
>> x=1:20;  
>> plot(x)  
>> stem(x)
```

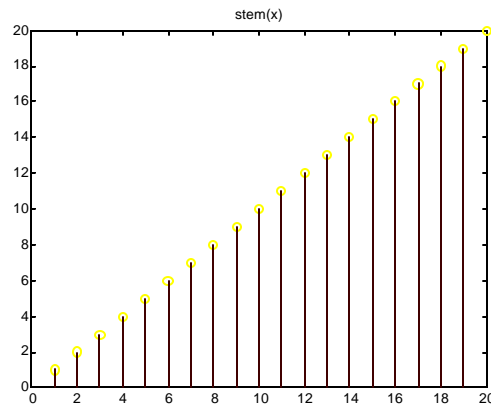
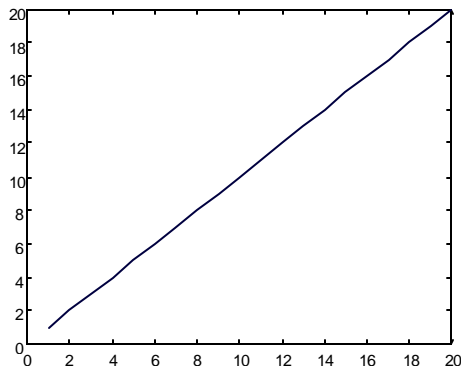


Figure 1. The function $y=x$ using the *plot* function (left) and the *stem* function (right).

More generally, **plot(X,Y)** plots vector Y versus vector X. Various line types, plot symbols and colors may be obtained with **plot(X,Y,S)** where S is a character string indicating the color of the line, and the type of line (e.g., dashed, solid, dotted, etc.). Examples for the string S include:

r	red	+	plus	--	dashed
g	green	*	star		
b	blue	s	square		

You can insert x-labels, y-labels and title to the plots, using the functions **xlabel**(.), **ylabel**(.) and **title**(.) respectively.

To find more information about the plot function, type in the MATLAB command window: help plot

Programming in MATLAB (M-files)

MATLAB programming is done using M-files, i.e., files that have the extension `.m`. These files are created using a text editor. To open the text editor, go to the **File** pull-down menu, choose **New**, then **M-file**. After you type the program in the `.m` file, save it, and then call it from the command window to execute it.

Say for instance that you want to write a program to compute the average (mean) of a vector `x`. The program should take as input the vector `x` and return the average.

Steps:

1. You need to create a new file, called “average.m”. If you are in Windows, open the text editor by going to the **File** pull-down menu, choose **New**, then **M-file**. If you are in UNIX, then type in the command window: `edit average.m`.
Type in the empty file:

```
function y=average(x)

L=length(x); % this is a built-in MATLAB function that returns the length (# of elements) of the vector x.
sum=0;
for i=1:L % this loop computes the sum of all elements in array x
    sum=sum+x(i);
end

y=sum/L; % the average
```

Remarks:

- `y` – is the output of the function “average”
- `x` – is the input array to the function “average”
- average – is the name of the function. It’s best if it has the same name as the filename.

The MATLAB files always need to have the extension `.m`

2. From the Editor pull-down menu, go to **File | Save**, and enter: `average.m` for the filename.
3. Go to the Command window to execute the program.

```
>> x=1:100; % x=[1, 2, 3, ..., 100]
>> y=average(x)
ans =
    50.5000
```

Bibliography

MATLAB 5.3, *Student Version Users Manual*, www.mathworks.com

D. Hanselman and B. Littlefield (1998). *Mastering MATLAB 5: A comprehensive tutorial and reference*, Prentice Hall.