

## Vectors, Scripts, and Plots in Matlab

- This section introduces fundamental features and capabilities of MATLAB as related to numerical methods.
- These range from vector and matrix operations to plotting functions and sets of data.
- We will discuss several MATLAB built-in functions (commands) and their applications, as well as preparing user-defined functions to perform specific tasks.

### Built-In Functions

in the command window type

```
>> sin(pi/2)
```

```
>> exp(2)
```

```
>> sqrt(5)
```

Note: In Matlab anything that comes in a line after a % is a comment.

## Vectors

- In Matlab, the basic objects are matrices, i.e. arrays of numbers. Vectors can be thought of as special matrices.
- A row vector is recorded as a  $1 \times n$  matrix and a column vector is recorded as a  $m \times 1$  matrix.

To enter a row vector in Matlab, type the following in the command window:

```
>> v = [1 2 3 4]
```

Note: Commas may be used instead of spaces between elements.

For column vectors, the elements must be separated by semicolons.

```
>> w = [1 ; 2; 3; 4]
```

Note: Arrays of numbers with equal spacing can be created more effectively. For example, a row vector whose first element is 2, its last element is 17, with a spacing of 3 is created as

```
>> v = [2:3:17] or >> v = 2:3:17
```

To create a column vector with the same properties:

```
>> w = [2:3:17]'
```

Any component of a vector can be easily retrieved.

```
>> v(4)
```

## Option 2: linspace

Another way to create vectors with equally spaced elements is by using the `linspace` command.

6 equally-spaced points between 1 and 10

```
>> x = linspace(1,10,6)
```

Note: The default value for the number of points is 100.

## Matrices

A matrix can be created by using brackets enclosing all of its elements, rows separated by a semicolon.

```
>> A = [ 1 2 3 ; 4 5 6]
```

An entry can be accessed by using the row and column number of the location of that entry.

```
>> A (2, 1)
```

An entire row or column of a matrix is accessed by using a colon.

2nd row of A

```
>> A (2, :)
```

3rd column of A

```
>> A (:, 3)
```

To replace an entire column of matrix A by a given vector v, we proceed as follows.

```
>> v = [1; 0];  
>> A_new = A; % Pre-allocate  
               % the new matrix  
>> A_new(:, 2) = v % Replace the  
                   % 2nd column with v
```

Now try:

```
>> B_new = [A v]
```

Next, try the commands:

```
>> zeros(5)
>> zeros(5,2)
>> ones(3)
>> ones(3,2)
```

Now, explore the commands:  
length and size

## Element-by-Element Operations

Suppose we want to raise each element of a vector to power of 2.

```
>> x = linspace(0,10,6)
>> x.^2
```

### Element-by-Element Operations

MATLAB Symbol	Description
<code>.*</code>	Multiplication
<code>./</code>	(right) Division
<code>.^</code>	Exponentiation

## Plots

Plotting a vector of values versus another vector of values is done by using the `plot` command.

```
>> x = linspace(1,10,20);  
>> y = x.^2;  
>> plot(x,y,'*')
```

Check: `>> help plot` and `help hold`

## Script Files

- A script file comprises a list of commands as if they were typed at the command line.
- Script files can be created in the MATLAB Editor, and saved as an M file.
- When the script file is executed (run), MATLAB executes the commands.
- Next, create a mfile with

```
x = linspace(0,1)
x2 = x .^2;
x3 = x .^3;
x4 = x .^4;
plot (x,x,'k ',x,x2 ,'b ',...
      x,x3,'g', x,x4 ,'r')
```



To increase happiness, write a well-commented script program that graphs the functions  $\sin x$ ,  $\sin 2x$ ,  $\sin 3x$ , and  $\sin 4x$  on the interval  $0, 2\pi$  on **one plot**. ( $\pi$  is pi in Matlab.) Use a sufficiently small step size to make all the graphs smooth.

## Matlab functions

- The function can be a simple single mathematical expression or a complicated and involved series of calculations.

### Generic structure of a matlab function

```
function [output variables] = FunctionName(input variables)
% Comments

Expressions/statements

Calculation of all output variables

end
```

\* The first executable line in a function file must be the function definition line.

```
function [output arguments] = function_name(input arg.)
```

```

function y = myfunc (x)
% Computes the function
%            $2x^2 - 3x + 1$ 
% Input : x -- a number or vector;
% for a vector the computation
% is elementwise
% Output : y -- a number
% or vector of the
% same size as x
y = 2*x.^2 - 3*x + 1;
end

```

- Whenever you write code, it is a good practice to add comments that describe the code.
- Comments allow others to understand your code, and can refresh your memory when you return to it later.
- Add comments using the percent (%) symbol.

To increase happiness, let's write a function (name it FtoC) that converts temperature in degrees F to temperature in degrees C.  
(Hint:  $C = 5 * (F - 32) / 9$ ;) )

```
function degC = FtoC(degF)
% this function converts
% degF to degC
degC = 5 * (degF - 32) / 9;
end
```

Now try: Write a function to calculate the area and the circumference of a circle of a given radius.

Remark:

- If there are more than one, the input arguments are separated with commas.
- The function body contains the computer code that actually performs the computations.
- The code can use all MATLAB programming features.
- This includes calculations, assignments, any built-in or user-defined functions, flow control (conditional statements and loops), comments, blank lines.

**In general, the best way to use a function is to capture the result it returns and then use or print this result.**

## Anonymous function

- An anonymous function offers a way to create a function for simple expressions with or without creating an M file.
- Anonymous functions can only contain one expression and cannot return more than one output variable.
- They can either be created in the Command Window or as a script.

`My_function = @(arguments) (expression)`

Let's create a function to evaluate

$$\alpha = \sqrt{(1 + e^{(-bx/2)})}$$

```
>> alpha = @(b,x) (sqrt(1+exp(-b*x/2)))  
>> alpha(1,2)
```

HW: Read about the `inline` function.

## Program Flow Control

Program flow can be controlled with the following three commands:

`for`, `if`, and `while`.

### `for` Loop

A `for/end` loop repeats a statement, or a group of statements, a specific number of times. Its generic form is

```
for i = first:increment:last,  
    statements...  
end
```

1. The index `i` assumes its first value
2. all statements in the subsequent lines are executed with `i = first`, then the program goes back to the `for` command

3. `i` assumes the value

`i = first + increment` and the process continues until the very last run corresponding to `i = last`.

Ex. Write a script to generate a  $5 \times 5$  matrix  $A$  with diagonal entries all equal to 1, and super diagonal entries all equal to 2, while all other entries are zero.

```
A = eye(5);  
for i = 1: length(A)-1  
    A(i,i+1) = 2;  
end
```



## **if Command**

The most general form of the `if` command is  
Its generic form is

```
if condition 1
    set of expressions 1
else if condition 2
    set of expressions 2
else
    set of expressions 3
end
```

The simplest form of a conditional statement is  
the `if/end` structure. For example

```
x = 3;
my_func = @(x) (x^2+5*x-6);
if my_func(x) ~=0
    disp('x is not a root')
end
```

The `if/else/end` structure allows for choosing one group of expressions from two groups.

```
x = 1;  
my_func = @(x) (x^2+5*x-6);  
if my_func(x) ~= 0  
    disp('x is not a root')  
else  
    disp('x is a root :')  
end
```

## while Loop

A `while/end` loop repeats a statement, or a group of statements, until a specific condition is met.

Its generic form is:

```
while condition
    statements
end
```

Ex. Write a script to generate a  $5 \times 5$  matrix  $A$  with diagonal entries all equal to 1, and super-diagonal entries all equal to  $-3$ , while all other entries are zero, this time with the aid of the while loop.

```
A = eye(5); i = 1;
while i < 5
    A(i, i+1) = -3;
    i = i+1;
end
```

To increase the happiness, write a script file that employs any combination of the flow control commands to generate

$$\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -7 & 0 & 3 & 0 \\ 0 & -7 & 0 & 4 \end{bmatrix}$$

```
clc;
A = zeros(4);
i = 1;
while i < 5
    A(i,i) = i;
    if i < 3
        A(i,i+2) = -1;
        A(i+2,i) = -7;
    end
    i = i+1;
end
A
```

## Reference

1. Introduction to Numerical Methods and Matlab Programming for Engineers by Todd Young and Martin J. Mohlenkamp
2. An Introduction to Programming and Numerical Methods in MATLAB by S.R. Otto and J.P. Denier
3. Physical Modeling in MATLAB by Allen B. Downey