An Introduction to Matlab and Numerical Computations

September 24, 2020

MATLAB Windows

Command Window

MATLAB Windows

Command Window

- The Command Window is MATLAB 's main window and opens when MATLAB is started.

MATLAB Windows

Command Window

- The Command Window is MATLAB 's main window and opens when MATLAB is started.
- Commands are typed next to the prompt ($>>$) and are executed when the Enter key is pressed.

MATLAB Windows

Command Window

- The Command Window is MATLAB 's main window and opens when MATLAB is started.
- Commands are typed next to the prompt ($>>$) and are executed when the Enter key is pressed.
- Once a command is typed and the Enter key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.

MATLAB Windows

Command Window

- The Command Window is MATLAB 's main window and opens when MATLAB is started.
- Commands are typed next to the prompt (>>) and are executed when the Enter key is pressed.
- Once a command is typed and the Enter key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Output generated by the command is displayed in the Command Window, unless a semicolon (;) is typed at the end.

MATLAB Windows

Command Window

- When the symbol % (percent symbol) is typed in the beginning of a line, the line is designated as a comment and is not executed.

MATLAB Windows

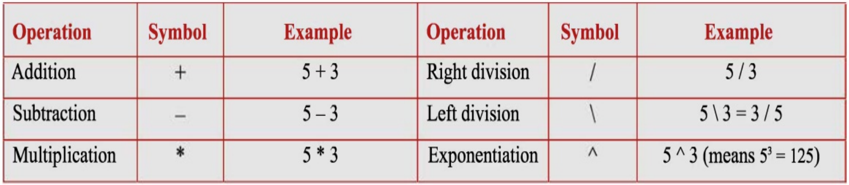
Command Window

- When the symbol % (percent symbol) is typed in the beginning of a line, the line is designated as a comment and is not executed.
- The *clc* command (type *clc* and press Enter) clears the Command Window. The command does not change anything that was done before.

Command Window

- When the symbol % (percent symbol) is typed in the beginning of a line, the line is designated as a comment and is not executed.
- The *clc* command (type *clc* and press Enter) clears the Command Window. The command does not change anything that was done before.
- The up-arrow key (↑) can also be used to recall commands that were typed before.

Elementary Arithmetic Operations with Scalars

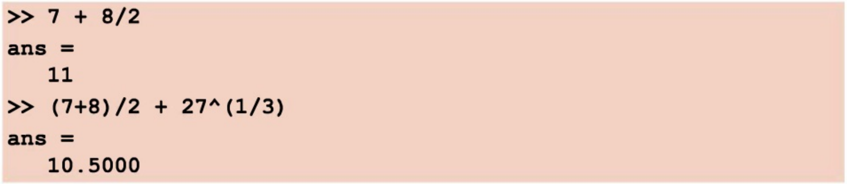


Operation	Symbol	Example	Operation	Symbol	Example
Addition	+	5 + 3	Right division	/	5 / 3
Subtraction	–	5 – 3	Left division	\	5 \ 3 = 3 / 5
Multiplication	*	5 * 3	Exponentiation	^	5 ^ 3 (means $5^3 = 125$)

Examples

Examples

First Example



```
>> 7 + 8/2
ans =
   11
>> (7+8)/2 + 27^(1/3)
ans =
   10.5000
```

Examples

First Example

```
>> 7 + 8/2
ans =
   11
>> (7+8)/2 + 27^(1/3)
ans =
   10.5000
```

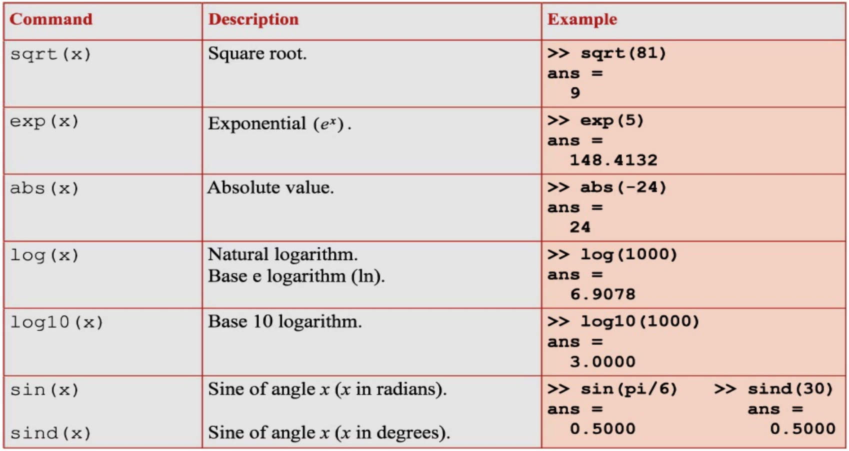
Second Example

```
>> a = 12
a =
   12
>> B = 4;
>> C = (a - B) + 40 - a/B*10
C =
   18
```

Since a semicolon is typed at the end of the command, the value of B is not displayed.

Bluit-in Elementary Math Functions

Bluit-in Elementary Math Functions



Command	Description	Example

`sqrt(x)`	Square root.	`>> sqrt(81)`
		`ans =`
		`9`

`exp(x)`	Exponential (e^x).	`>> exp(5)`
		`ans =`
		`148.4132`

`abs(x)`	Absolute value.	`>> abs(-24)`
		`ans =`
		`24`

`log(x)`	Natural logarithm.
	Base e logarithm (ln).
		`>> log(1000)`
		`ans =`
		`6.9078`

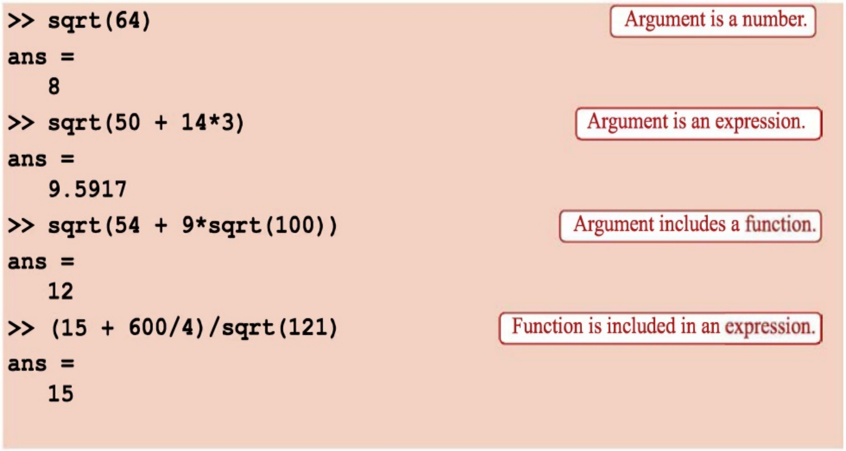
`log10(x)`	Base 10 logarithm.	`>> log10(1000)`
		`ans =`
		`3.0000`

`sin(x)`
 	
`sind(x)`
	Sine of angle x (x in radians).
	
	Sine of angle x (x in degrees).
		`>> sin(pi/6)`
		`ans =`
		`0.5000`
		
		`>> sind(30)`
		`ans =`
		`0.5000`

Examples

Examples

Third Example



```
>> sqrt(64)
ans =
   8
>> sqrt(50 + 14*3)
ans =
   9.5917
>> sqrt(54 + 9*sqrt(100))
ans =
   12
>> (15 + 600/4)/sqrt(121)
ans =
   15
```

Argument is a number.

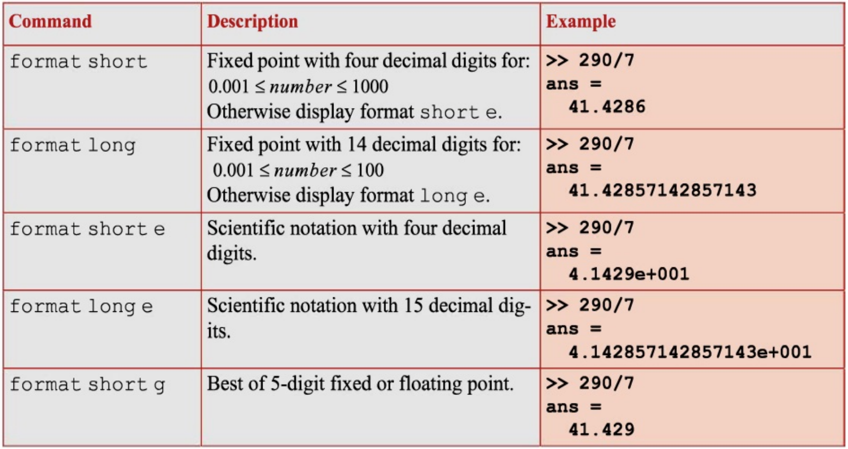
Argument is an expression.

Argument includes a function.

Function is included in an expression.

Display Formats

Display Formats



Command	Description	Example
`format short`	Fixed point with four decimal digits for: $0.001 \le number \le 1000$ Otherwise display format `short e`.	`>> 290/7` `ans =` `41.4286`
`format long`	Fixed point with 14 decimal digits for: $0.001 \le number \le 100$ Otherwise display format `long e`.	`>> 290/7` `ans =` `41.42857142857143`
`format short e`	Scientific notation with four decimal digits.	`>> 290/7` `ans =` `4.1429e+001`
`format long e`	Scientific notation with 15 decimal digits.	`>> 290/7` `ans =` `4.142857142857143e+001`
`format short g`	Best of 5-digit fixed or floating point.	`>> 290/7` `ans =` `41.429`

MATLAB Windows

Figure Window

MATLAB Windows

Figure Window

- The Figure Window opens automatically when graphics commands are executed and contains graphs created by these commands.

MATLAB Windows

Figure Window

- The Figure Window opens automatically when graphics commands are executed and contains graphs created by these commands.

Editor Window

MATLAB Windows

Figure Window

- The Figure Window opens automatically when graphics commands are executed and contains graphs created by these commands.

Editor Window

- The Editor Window is used for writing and editing programs. This window is opened from the File menu in the Command Window where it is used for creating script files.

MATLAB Windows

Figure Window

- The Figure Window opens automatically when graphics commands are executed and contains graphs created by these commands.

Editor Window

- The Editor Window is used for writing and editing programs. This window is opened from the File menu in the Command Window where it is used for creating script files.

Help Window

MATLAB Windows

Figure Window

- The Figure Window opens automatically when graphics commands are executed and contains graphs created by these commands.

Editor Window

- The Editor Window is used for writing and editing programs. This window is opened from the File menu in the Command Window where it is used for creating script files.

Help Window

- The Help Window contains help information. This window can be opened from the Help menu in the toolbar of any MATLAB window. The Help Window is interactive and can be used to obtain information on any feature of MATLAB.

Arrays

Arrays

Creating a Vector

Arrays

Creating a Vector

```
variable_name = [number number ... number]
```

Arrays

Creating a Vector

```
variable_name = [number number ... number]
```

```
variable_name = m:q:n
```

Arrays

Creating a Vector

```
variable_name = [number number ... number]
```

```
variable_name = m:q:n
```

```
variable_name = linspace(xi,xf,n)
```

Arrays

Creating a Vector

```
variable_name = [number number ... number]
```

```
variable_name = m:q:n
```

```
variable_name = linspace(xi,xf,n)
```

Examples

Arrays

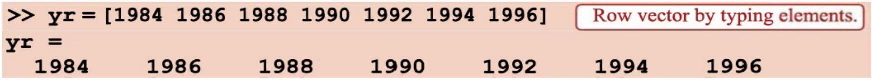
Creating a Vector

```
variable_name = [number number ... number]
```

```
variable_name = m:q:n
```

```
variable_name = linspace(xi,xf,n)
```

Examples



```
>> yr = [1984 1986 1988 1990 1992 1994 1996]
yr =
  1984      1986      1988      1990      1992      1994      1996
```

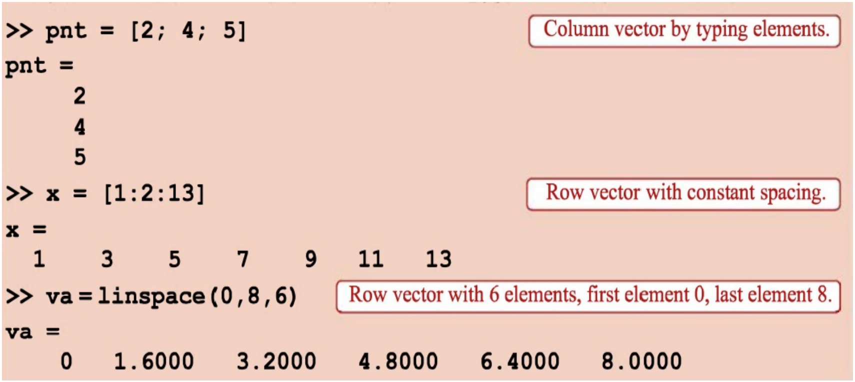
Row vector by typing elements.

Arrays

Examples

Arrays

Examples



```
>> pnt = [2; 4; 5]
pnt =
     2
     4
     5
>> x = [1:2:13]
x =
  1     3     5     7     9    11    13
>> va=linspace(0,8,6)
va =
     0    1.6000    3.2000    4.8000    6.4000    8.0000
```

Column vector by typing elements.

Row vector with constant spacing.

Row vector with 6 elements, first element 0, last element 8.

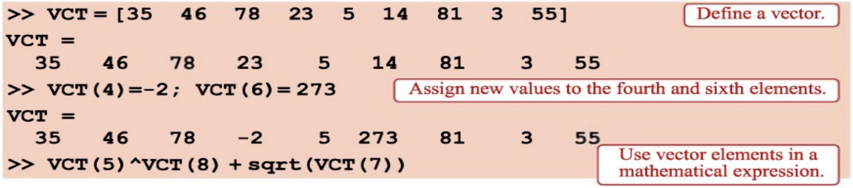
Arrays

Arrays

Array addressing

Arrays

Array addressing



```
>> VCT = [35  46  78  23  5  14  81  3  55]
VCT =
   35    46    78    23     5    14    81     3    55
>> VCT(4)=-2; VCT(6)=273
VCT =
   35    46    78    -2     5   273    81     3    55
>> VCT(5)^VCT(8) + sqrt(VCT(7))
```

Define a vector.

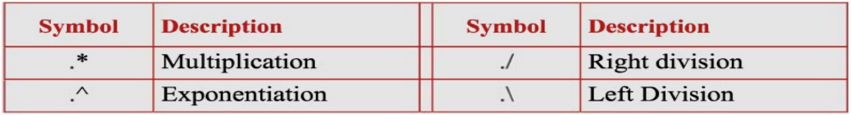
Assign new values to the fourth and sixth elements.

Use vector elements in a mathematical expression.

Arrays

Arrays

Element-by-element operations



Symbol	Description	Symbol	Description
.*	Multiplication	./	Right division
.^	Exponentiation	.\	Left Division

Arrays

Element-by-element operations

Symbol	Description	Symbol	Description
.*	Multiplication	./	Right division
.^	Exponentiation	.\	Left Division

```
>> z = [1:2:15]
z =
   1     3     5     7     9   11   13   15
>> y = (z.^3 + 5*z)./(4*z.^2 - 10)
y =
 -1.0000  1.6154  1.6667  2.0323  2.4650  2.9241  3.3964  3.8764
```

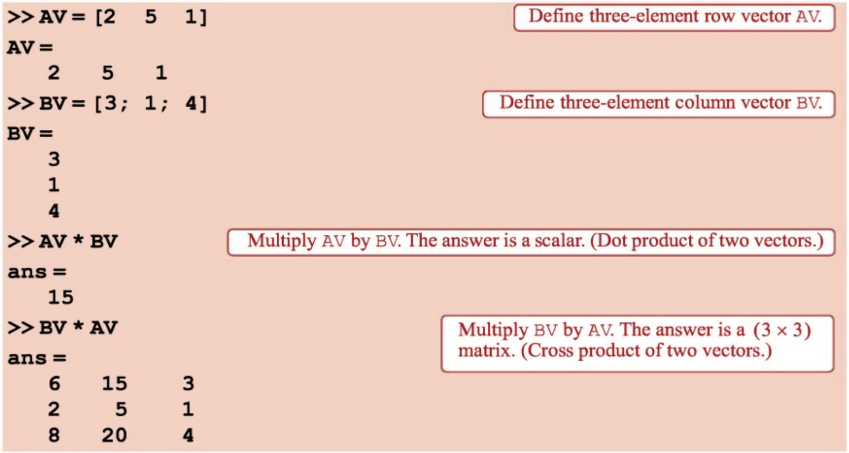
Define a vector z with eight elements.

Vector z is used in element-by-element calculation of the elements of vector y.

Mathematical Operations with Arrays

Mathematical Operations with Arrays

Multiplication of arrays



```
>> AV = [2   5   1]
AV =
     2     5     1
>> BV = [3; 1; 4]
BV =
     3
     1
     4
>> AV * BV
ans =
     15
>> BV * AV
ans =
     6    15     3
     2     5     1
     8    20     4
```

Define three-element row vector AV.

Define three-element column vector BV.

Multiply AV by BV. The answer is a scalar. (Dot product of two vectors.)

Multiply BV by AV. The answer is a (3×3) matrix. (Cross product of two vectors.)

Arrays

Arrays

Creating a two-dimensional array (matrix)

Arrays

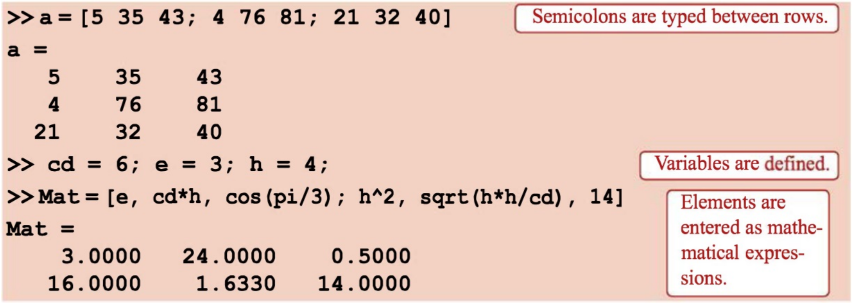
Creating a two-dimensional array (matrix)

```
variable_name=[1st row elements; 2nd row ele-
               ments;....; last row elements]
```

Arrays

Creating a two-dimensional array (matrix)

```
variable_name=[1st row elements; 2nd row ele-
              ments;....; last row elements]
```



```
>>a=[5 35 43; 4 76 81; 21 32 40]
a =
    5     35     43
    4     76     81
   21     32     40
>> cd = 6; e = 3; h = 4;
>>Mat=[e, cd*h, cos(pi/3); h^2, sqrt(h*h/cd), 14]
Mat =
     3.0000    24.0000     0.5000
    16.0000     1.6330    14.0000
```

Semicolons are typed between rows.

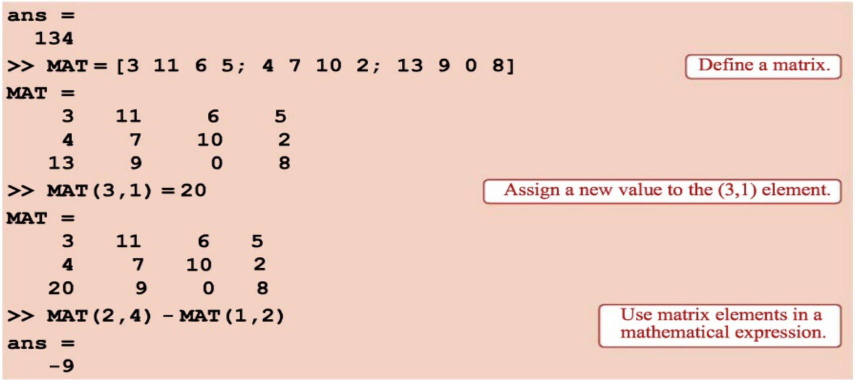
Variables are defined.

Elements are entered as mathematical expressions.

Arrays

Arrays

Array addressing



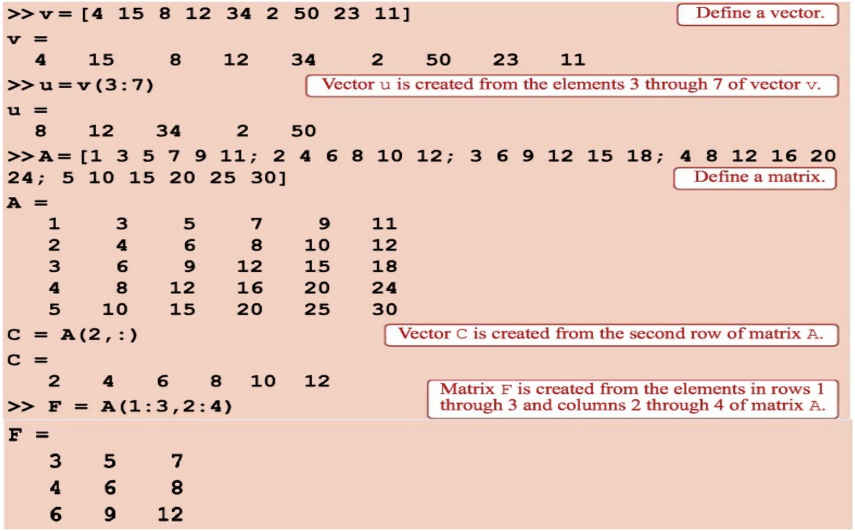
```
ans =
  134
>> MAT=[3 11 6 5; 4 7 10 2; 13 9 0 8]
MAT =
     3    11     6     5
     4     7    10     2
    13     9     0     8
>> MAT(3,1)=20
MAT =
     3    11     6    5
     4     7    10    2
    20     9     0    8
>> MAT(2,4)-MAT(1,2)
ans =
    -9
```

Define a matrix.

Assign a new value to the (3,1) element.

Use matrix elements in a mathematical expression.

Using a colon : in Addressing Arrays



```
>> v = [4 15 8 12 34 2 50 23 11]
v =
   4    15     8    12    34     2    50    23    11
>> u = v(3:7)
u =
   8    12    34     2    50
>> A = [1 3 5 7 9 11; 2 4 6 8 10 12; 3 6 9 12 15 18; 4 8 12 16 20
24; 5 10 15 20 25 30]
A =
     1     3     5     7     9    11
     2     4     6     8    10    12
     3     6     9    12    15    18
     4     8    12    16    20    24
     5    10    15    20    25    30
C = A(2,:)
C =
     2     4     6     8    10    12
>> F = A(1:3,2:4)
F =
     3     5     7
     4     6     8
     6     9    12
```

Define a vector.

Vector u is created from the elements 3 through 7 of vector v.

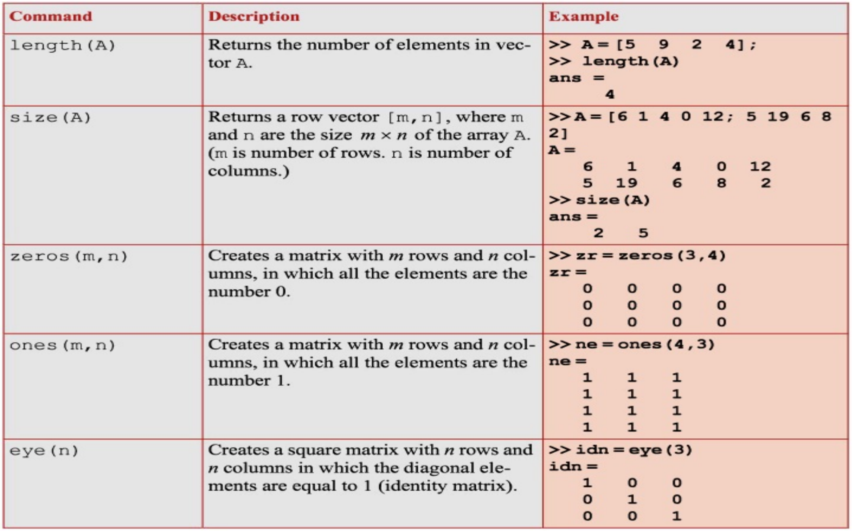
Define a matrix.

Vector C is created from the second row of matrix A.

Matrix F is created from the elements in rows 1 through 3 and columns 2 through 4 of matrix A.

Built-In Functions for Handling Arrays

Built-In Functions for Handling Arrays



Command	Description	Example

`length(A)`	Returns the number of elements in vector A.	`>> A=[5 9 2 4];`
		`>> length(A)`
		`ans =`
		`4`

`size(A)`	Returns a row vector [m,n], where m and n are the size $m \times n$ of the array A. (m is number of rows. n is number of columns.)	`>>A=[6 1 4 0 12; 5 19 6 8 2]`
		`A=`
		`6 1 4 0 12`
		`5 19 6 8 2`
		`>>size(A)`
		`ans=`
		`2 5`

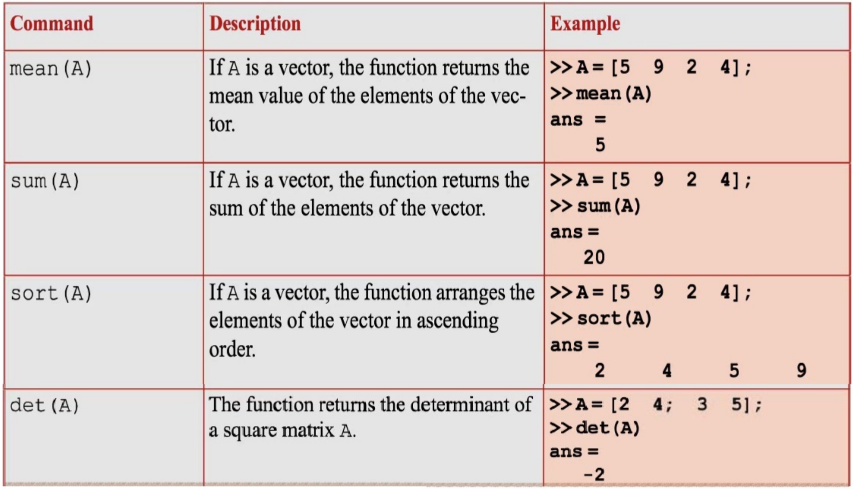
`zeros(m,n)`	Creates a matrix with *m* rows and *n* columns, in which all the elements are the number 0.	`>>zr=zeros(3,4)`
		`zr=`
		`0 0 0 0`
		`0 0 0 0`
		`0 0 0 0`

`ones(m,n)`	Creates a matrix with *m* rows and *n* columns, in which all the elements are the number 1.	`>>ne=ones(4,3)`
		`ne=`
		`1 1 1`
		`1 1 1`
		`1 1 1`
		`1 1 1`

`eye(n)`	Creates a square matrix with *n* rows and *n* columns in which the diagonal elements are equal to 1 (identity matrix).	`>>idn=eye(3)`
		`idn=`
		`1 0 0`
		`0 1 0`
		`0 0 1`

Built-In Functions for Handling Arrays

Built-In Functions for Handling Arrays



Command	Description	Example

`mean(A)`	If `A` is a vector, the function returns the mean value of the elements of the vector.	`>> A = [5 9 2 4];`
		`>> mean(A)`
		`ans =`
		`5`

`sum(A)`	If `A` is a vector, the function returns the sum of the elements of the vector.	`>> A = [5 9 2 4];`
		`>> sum(A)`
		`ans =`
		`20`

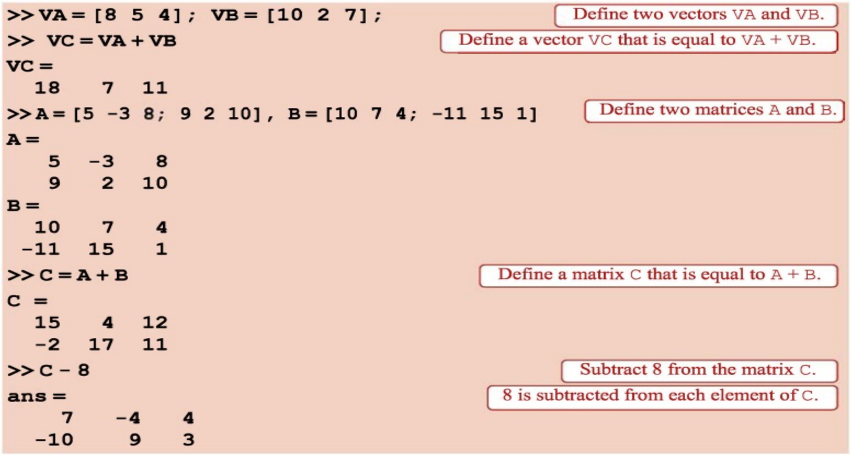
`sort(A)`	If `A` is a vector, the function arranges the elements of the vector in ascending order.	`>> A = [5 9 2 4];`
		`>> sort(A)`
		`ans =`
		`2 4 5 9`

`det(A)`	The function returns the determinant of a square matrix `A`.	`>> A = [2 4; 3 5];`
		`>> det(A)`
		`ans =`
		`-2`

Mathematical Operations with Arrays

Mathematical Operations with Arrays

Addition and subtraction of arrays



```
>> VA = [8  5  4];  VB = [10  2  7];
>>  VC = VA + VB
VC =
  18    7   11
>> A = [5  -3  8;  9  2  10],  B = [10  7  4;  -11  15  1]
A =
    5   -3    8
    9    2   10
B =
   10    7    4
  -11   15    1
>> C = A + B
C  =
  15    4   12
  -2   17   11
>> C - 8
ans =
     7    -4    4
   -10     9    3
```

Define two vectors VA and VB.

Define a vector VC that is equal to VA + VB.

Define two matrices A and B.

Define a matrix C that is equal to A + B.

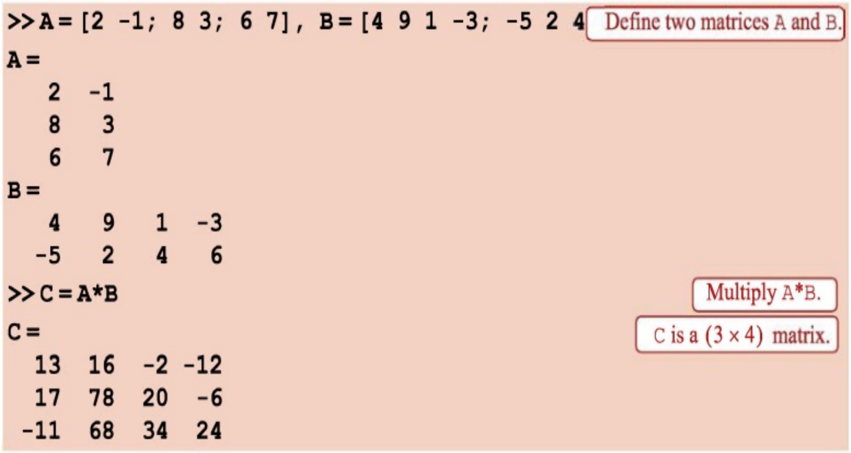
Subtract 8 from the matrix C.

8 is subtracted from each element of C.

Mathematical Operations with Arrays

Mathematical Operations with Arrays

Multiplication of arrays

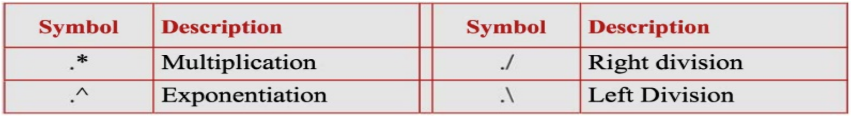


```
>>A=[2 -1; 8 3; 6 7], B=[4 9 1 -3; -5 2 4
A=
    2   -1
    8    3
    6    7
B=
    4    9    1   -3
   -5    2    4    6
>>C=A*B
C=
   13   16   -2  -12
   17   78   20   -6
  -11   68   34   24
```

Mathematical Operations with Arrays

Mathematical Operations with Arrays

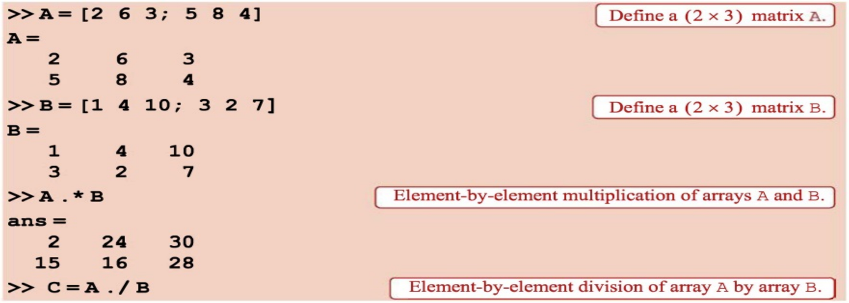
Recall Element-by-element operations



Symbol	Description	Symbol	Description
.*	Multiplication	./	Right division
.^	Exponentiation	.\	Left Division

Mathematical Operations with Arrays

Mathematical Operations with Arrays



```
>>A= [2  6  3;  5  8  4]
A=
     2      6      3
     5      8      4
>>B= [1  4  10;  3  2  7]
B=
     1      4     10
     3      2      7
>>A .* B
ans=
     2     24     30
    15     16     28
>> C=A ./ B
```

Define a (2×3) matrix A.

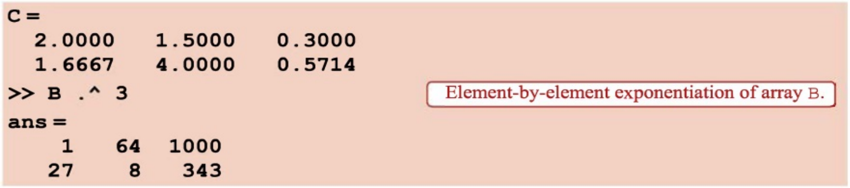
Define a (2×3) matrix B.

Element-by-element multiplication of arrays A and B.

Element-by-element division of array A by array B.

Mathematical Operations with Arrays

Element-by-element operations



```
C =
  2.0000    1.5000    0.3000
  1.6667    4.0000    0.5714
>> B .^ 3
ans =
    1    64   1000
   27     8    343
```

Element-by-element exponentiation of array B.

Plotting

Plotting in 2-D

```
plot(x,y)
```

Plotting

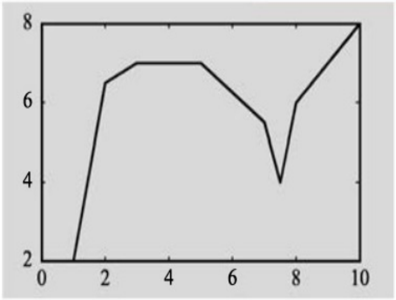
Plotting in 2-D

```
plot(x,y)
```

```
>>x=[1  2  3  5  7  7.5  8  10];
>>y=[2  6.5  7  7  5.5  4  6  8];
>>plot(x,y)
```

Plotting

Plotting in 2-D



Plotting

Plotting in 2-D

```
plot(x,y,'line specifiers')
```

Plotting

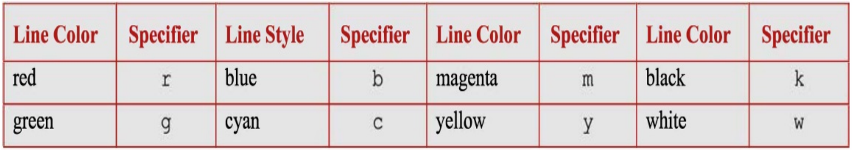
Plotting in 2-D

```
plot(x,y,'line specifiers')
```

Line Style	Specifier	Line Style	Specifier
solid (default)	-	dotted	:
dashed	--	dash-dot	-.

Plotting

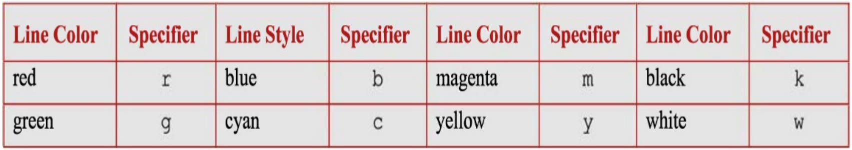
Plotting in 2-D



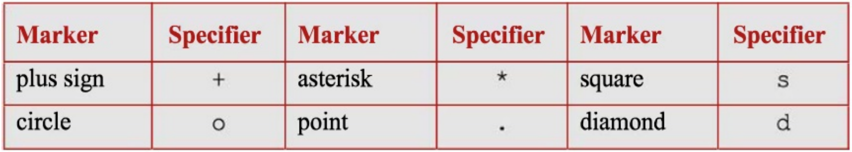
Line Color	Specifier	Line Style	Specifier	Line Color	Specifier	Line Color	Specifier
red	r	blue	b	magenta	m	black	k
green	g	cyan	c	yellow	y	white	w

Plotting

Plotting in 2-D



Line Color	Specifier	Line Style	Specifier	Line Color	Specifier	Line Color	Specifier
red	r	blue	b	magenta	m	black	k
green	g	cyan	c	yellow	y	white	w



Marker	Specifier	Marker	Specifier	Marker	Specifier
plus sign	+	asterisk	*	square	s
circle	o	point	.	diamond	d

Plotting

Plotting in 2-D

```
xlabel('text as string')
ylabel('text as string')
title('text as string')
```

Plotting

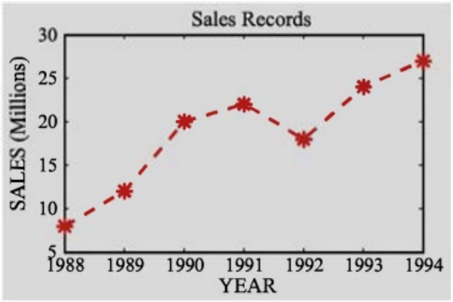
Plotting in 2-D

```
xlabel('text as string')
ylabel('text as string')
title('text as string')
```

```
>>yr=[1988:1:1994];
>>sle=[8  12  20  22  18  24  27];
>>plot(yr,sle,'--r*','linewidth',2,'markersize',12)
>>xlabel('YEAR')
>>ylabel('SALES (Millions)')
>>title('Sales Records')
```

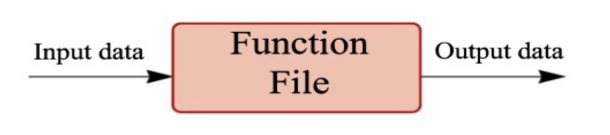
Plotting

Plotting in 2-D

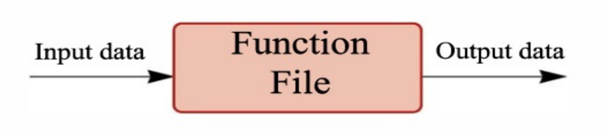


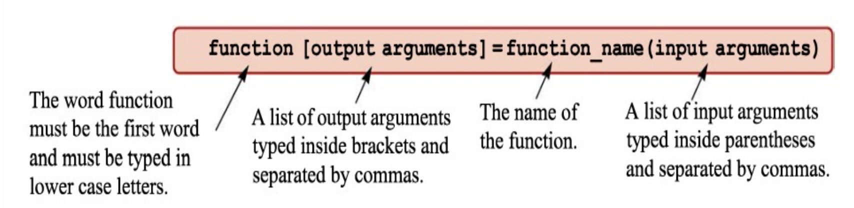
PROGRAMMING IN MATLAB

Defining Functions



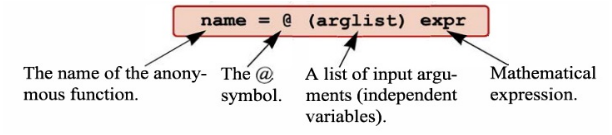
Defining Functions




function [output arguments] = function_name (input arguments)
The word function must be the first word and must be typed in lower case letters.
A list of output arguments typed inside brackets and separated by commas.
The name of the function.
A list of input arguments typed inside parentheses and separated by commas.

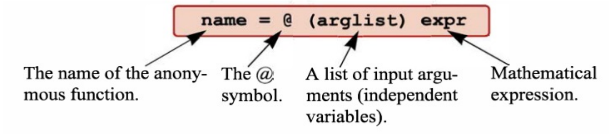
PROGRAMMING IN MATLAB

Defining Anonymous Function


name = @ (arglist) expr
The name of the anony-mous function.
The @ symbol.
A list of input argu-ments (independent variables).
Mathematical expression.

PROGRAMMING IN MATLAB

Defining Anonymous Function



```
name = @ (arglist) expr
```

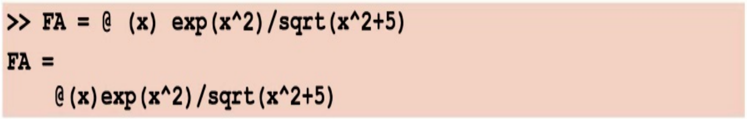
The name of the anonymous function.

The @ symbol.

A list of input arguments (independent variables).

Mathematical expression.

Example



```
>> FA = @ (x) exp(x^2)/sqrt(x^2+5)
FA =
    @(x)exp(x^2)/sqrt(x^2+5)
```

PROGRAMMING IN MATLAB

```
>> FA(2)
ans =
   18.1994
>> z = FA(3)
z =
  2.1656e+003
```

PROGRAMMING IN MATLAB

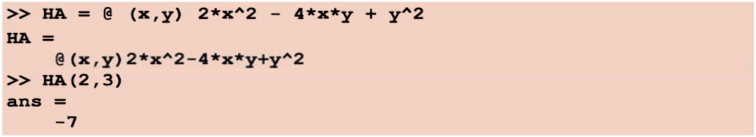
```
>> FA(2)
ans =
   18.1994
>> z = FA(3)
z =
  2.1656e+003
```

Example

```
>> FA = @ (x) exp(x.^2)./sqrt(x.^2+5)
FA =
    @(x)exp(x.^2)./sqrt(x.^2+5)
>> FA([1 0.5 2])
ans =
    1.1097     0.5604    18.1994
```

Using a vector as input argument.

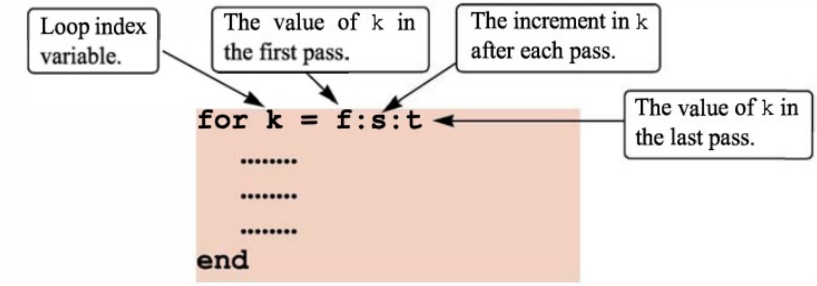
PROGRAMMING IN MATLAB



```
>> HA = @ (x,y) 2*x^2 - 4*x*y + y^2
HA =
    @(x,y)2*x^2-4*x*y+y^2
>> HA(2,3)
ans =
    -7
```

PROGRAMMING IN MATLAB

For Loop

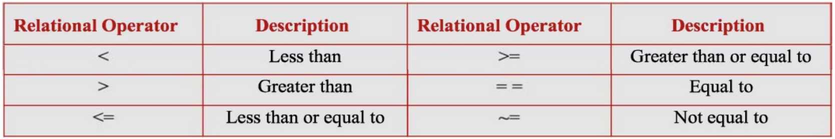


Loop index variable.
The value of k in the first pass.
The increment in k after each pass.
The value of k in the last pass.

```
for k = f:s:t
    ........
    ........
    ........
end
```

PROGRAMMING IN MATLAB

Relational and Logical Operators



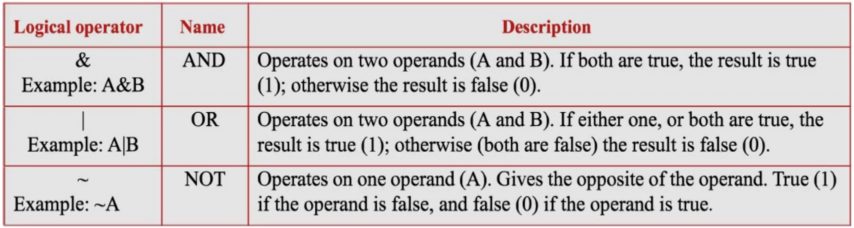
Relational Operator	Description	Relational Operator	Description
<	Less than	>=	Greater than or equal to
>	Greater than	= =	Equal to
<=	Less than or equal to	~=	Not equal to

Relational and Logical Operators

Relational Operator	Description	Relational Operator	Description
<	Less than	>=	Greater than or equal to
>	Greater than	= =	Equal to
<=	Less than or equal to	~=	Not equal to

```
>> 5 > 8
ans =
    0
>>  4 = = 6
ans  =
     0
```

Relational and Logical Operators



Logical operator	Name	Description
& Example: A&B	AND	Operates on two operands (A and B). If both are true, the result is true (1); otherwise the result is false (0).
| Example: A|B	OR	Operates on two operands (A and B). If either one, or both are true, the result is true (1); otherwise (both are false) the result is false (0).
~ Example: ~A	NOT	Operates on one operand (A). Gives the opposite of the operand. True (1) if the operand is false, and false (0) if the operand is true.

Relational and Logical Operators

Logical operator	Name	Description
& Example: A&B	AND	Operates on two operands (A and B). If both are true, the result is true (1); otherwise the result is false (0).
| Example: A|B	OR	Operates on two operands (A and B). If either one, or both are true, the result is true (1); otherwise (both are false) the result is false (0).
~ Example: ~A	NOT	Operates on one operand (A). Gives the opposite of the operand. True (1) if the operand is false, and false (0) if the operand is true.

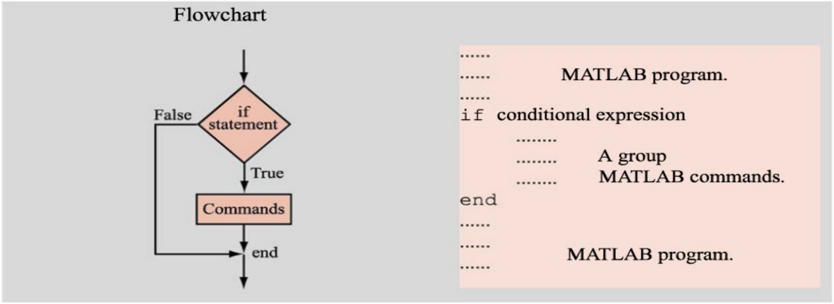
```
>> 3 & 7
ans =
     1
>> a = 5 | 0
a =
   1
>> ~25
ans =
     0
```

3 and 7 are both true (nonzero), so the outcome is 1.

1 is assigned to a since at least one number is true (nonzero).

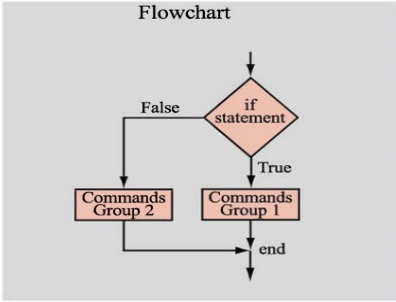
PROGRAMMING IN MATLAB

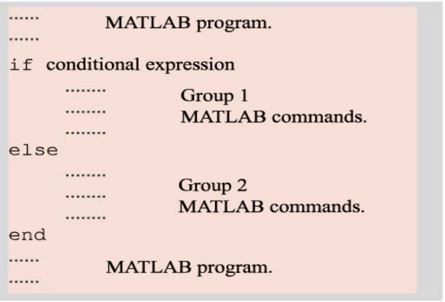
If-End Conditional Statement


Flowchart

```
......
......          MATLAB program.
......
if  conditional expression
        ........
        ........       A group
        ........       MATLAB commands.
end
......
......          MATLAB program.
......
```

If-Else-End Conditional Statement

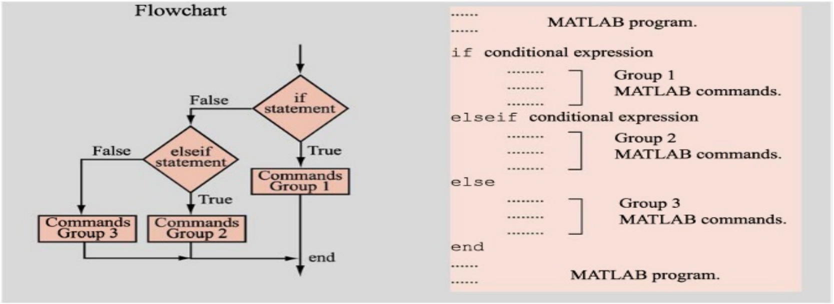




```
......          MATLAB program.
......
if  conditional expression
        .........        Group 1
        .........        MATLAB commands.
        .........
else
        .........        Group 2
        .........        MATLAB commands.
        .........
end
......          MATLAB program.
......
```

PROGRAMMING IN MATLAB

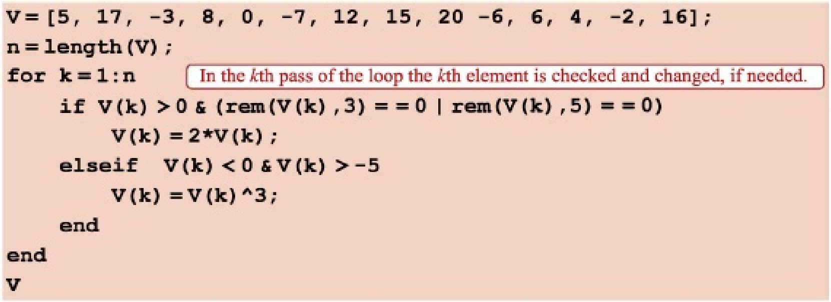
If-ElseIf-Else-End Conditional Statement



```
......
              MATLAB program.
......
if  conditional expression
          ........   ┐  Group 1
          ........   │  MATLAB commands.
          ........   ┘
elseif  conditional expression
          ........   ┐  Group 2
          ........   │  MATLAB commands.
          ........   ┘
else
          ........   ┐  Group 3
          ........   │  MATLAB commands.
          ........   ┘
end
......
              MATLAB program.
......
```

PROGRAMMING IN MATLAB

Exercise

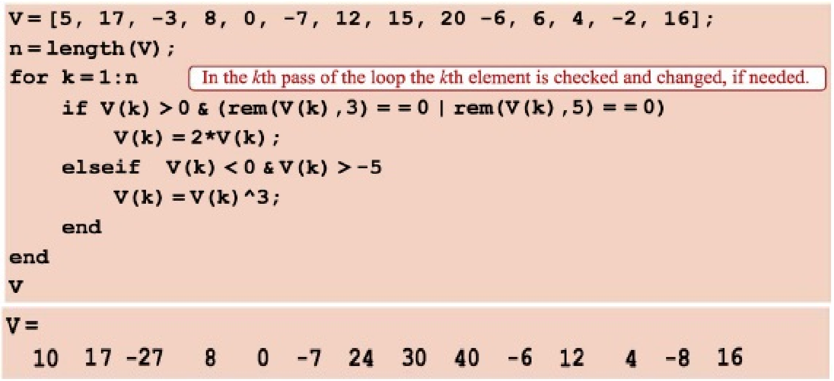


```
V=[5, 17, -3, 8, 0, -7, 12, 15, 20 -6, 6, 4, -2, 16];
n=length(V);
for k=1:n
    if V(k)>0 & (rem(V(k),3)==0 | rem(V(k),5)==0)
        V(k)=2*V(k);
    elseif  V(k)<0 & V(k)>-5
        V(k)=V(k)^3;
    end
end
V
```

In the kth pass of the loop the kth element is checked and changed, if needed.

PROGRAMMING IN MATLAB

Exercise



```
V=[5, 17, -3, 8, 0, -7, 12, 15, 20 -6, 6, 4, -2, 16];
n=length(V);
for k=1:n
    if V(k)>0 & (rem(V(k),3)==0 | rem(V(k),5)==0)
        V(k)=2*V(k);
    elseif  V(k)<0 & V(k)>-5
        V(k)=V(k)^3;
    end
end
V
```

In the kth pass of the loop the kth element is checked and changed, if needed.

```
V=
  10  17 -27   8   0  -7  24  30  40  -6  12   4  -8  16
```

Exercises

Ex-1

Write a MATLAB function for the following

$$y(x) = x\, e^{-0.7x} \sqrt(2x^2 + 1).$$

Assume that x is a vector and use the function to

- Calculate $y(3)$ and $y(8)$, and
- Make a plot of y versus x for $0 \leq x \leq 10$

Ex-2

Series expansion of exponential function is given by $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ...$ Write a code to calculate the RHS (series) sum upto first n terms

- Calculate error$=|e^x - \sum_0^n \frac{x^n}{n!}|$ for $n = 2, 4, 6, 8, 10$
- Stop summing if the error is < 0.0001.

Exercises

Ex-3

Write code using MATLAB to generate the Matrix
A=[2 -1 0 0 0;-1 2 -1 0 0;0 -1 2 -1 0;0 0 -1 2 -1;0 0 0 -1 2].

Ex-4

- Use Help Window/Google to understand While-End loop
- Write While-End loop to sum first 10 natural numbers which are multiples of 8.
- Write While-End loop to sum $'n'$ terms such that sum does not exceed 250.

References

Amos Gilat, Vish Subramaniam.
Numerical Methods for Engineers and Scientists An Introduction with Applications using MATLAB.
Third Edition, WILEY

Finally

Any Questions/Comments ??