University of Batna 2
Faculty of Mathematics and computer science
Department of computer science

Numerical Methods Practical Works

Oussama HARKATI

2nd year of a bachelor's degree in computer science

2023-2024

Practical Work "02"

V. Loops.

V.1 While

A while loop executes a block of instructions as long as a logical expression is true.

Syntax

```
while <expression>
    < Instructions >
end
```

Example

In the script editor window, type:

Commands

a = 0;
while (a < 3)
  fprintf ('The value of a: %d\n', a);
  a = a + 1;
end

Results
The value of a: 0 The value of a: 1 The value of a: 2

V.2 for

A for loop is used to repeat a block of instructions a given number of times.

Syntax

```
for index = value
   <Instructions>
end
```

Value has one of the following forms:

Form	Description
initVal :endVal	Increments the index variable from initVal to endVal by 1.
initVal : step : endVal	Index increments by step value in each iteration, or decrements when step value is negative.
Values from a vector	The index variable takes the values of a given vector.

Example

In the script editor window, type:

Commands
for a = 10 : 15   fprintf ('The value of a: %d\n', a); end
Results
The value of a: 10 The value of a: 11 The value of a: 12 The value of a: 13 The value of a: 14 The value of a: 15

Example

In the script editor window, type:

Commands
for a = 10 : 2 : 15   fprintf ('The value of a: %d\n', a); end
Results
The value of a: 10 The value of a: 12 The value of a: 14

Example

In the script editor window, type:

Commands
for a = [10 15 2 0]   disp (a) end
Results
10 15 2 0

V.3 Loops control

V.3.1 The break instruction

The break instruction ends the execution of a for or while loop then transfers the execution to the instruction immediately following the loop.

Example

In the script editor window, type:

Commands

for i=1:10
  if mod(i,2) == 1
  disp(i)
  else
  break
  disp(i+j) % j is the imaginary unit $\sqrt{-1}$.
  end
end

Results
1

V.3.1 The instruction continue

The continue instruction is used to pass control to the next iteration of for or while i.e., continue skips the rest of the code in its body and immediately tests the condition before repeating.

Example

In the script editor window, type:

Commands

for i=1:10
  if mod(i,2) == 1
  disp(i)
  else
  continues
  disp(i+j) % j is the imaginary unit $\sqrt{-1}$.
  end
end

Results
1 3 5 7 9

VI. Vectors.

In the command window, type the expressions

Commands	Results
>> A = [1 2 3 5 10]	A = 1 2 3 5 10
>> B = [10, 12, -5]	B = 10 12 -5
>> C = [5; 0; 1]	C = 5 0 1
>> D = [10: -2 : 1]	D = 10 8 6 4 2
>> E = [11; 12; 30; 41; 15]; >> E (3)	years = 30
>> F = [1 2 3]; >> F(:)	years = 1 2 3
>> G = [1 2 3 4 5 6 7 8]; >> sub_G = G(3:6)	sub_G = 3 4 5 6

VI.1 Vector creation

MATLAB lets you create two types of vectors: row vectors and column vectors.

VI.1.1 Line vector

To create line vectors, simply open a bracket, write the elements of the vector using a space or comma to separate them, then close the bracket as follows: $[x_1,\ x_2,\ x_3, \dots, x_n\ ]$ or $[x_1\ \ x_2\ \ x_3 \dots\ x_n\ ]$.

VI.1.1 Column vector

Column vectors are created in the same way as row vectors but using a semicolon to separate the elements as follows:
$[x_1;\ \ x_2;\ x_3; \dots; x_n\ ]$.

VI.2 Elements access

You can refer to one or more elements of a vector in different ways, where the $i^{i'eme}$ component of a vector v is called $v(i)$.

1. To access a single element, simply write vector_name(element_index).

Commands	Results
>> vect = [10 3 107 9 113 -10];	
>> vect (5)	ans = 113

2. To access a finite set of vector elements, simply write vector_name (firstElementIndex : lastElementIndex).

Example

Commands	Results
>> vect = [10: 50];	
>> vect (20:25)	ans = 29 30 31 32 33 34

3. To access all the elements of a vector, simply write vector_name (:)

Example

Commands	Results
>> vect = [1: 0.2: 2];	
>> vect (:)	ans = 1 1.2 1.4 1.6 1.8 2

VI.3 Elements modification

Vector elements can be modified by assigning new values to them.

Example

Commands	Results
>> v = [1 2 3 4 5];	
>> v (5) = v (3)—v (2);	
>> disp (v)	v = 1 2 3 4 1

MATLAB also has a table editor that lets you modify the dimensions and entries of a vector or matrix. To use this editor, double-click on the variable to be edited in the Workspace.

VI.4 Vector operations.

VI.4.1 Addition and subtraction.

You can add or subtract two vectors. Both vectors must be of the same type and have the same number of elements.

Example

In the script editor window, type:

Commands
A = [7, 11, 15, 23, 9]; B = [2, 5, 13, 16, 20]; C = A + B D = A – B
Results
C = 9 16 28 39 29 D = 5 6 2 7 -11

VI.4.2 Multiplication by a scalar.

Multiplying a vector by a number produces a new vector of the same type, with each element of the original vector multiplied by the chosen number.

Example

In the command window, type:

Commands	Results
>> vect = [12 34 10 8];	
>> newVect = 5 * vect	newVect = 60 170 50 40

VI.4.3 Transposed vector.

The transposition operation changes a column vector into a row vector and vice versa. The transposition operation is represented by (').

Example

In the command window, type:

Commands	Results
>> vect = [12 34 10 8]	vect = 12 34 10 8
>> transVect = vect'	transVect = 12 34 10 8

VI.4.4 The scalar product (Inner product).

In MATLAB, you can calculate the scalar product of two vectors using the dot command.

Example

In the command window, type:

Commands	Results
>> vec1 = [5 6 7 9];	
>> vect2 = [1 2 3 4];	
>> vect3 = dot (vect1, vect2)	vect3 = 74

Both vectors must have the same number of elements. In mathematics, the inner product of $A = [x_1\ x_2\ \dots\ x_n]$ and $A = [y_1\ y_2\ \dots\ y_n]$ is:

$$A.B = \sum_{i=1}^{n} x_i \times y_i$$

VI.4.5 Vectors' concatenation.

MATLAB lets you add vectors together to create new vectors.

1) The concatenation of two line vectors

In the command window, type:

Commands	Results
>> vec1 = [1 0 8 2];	
>> vect2 = [0 0 3 5];	
>> vect3 = [vect1 vect2]	vect3 = 1 0 8 2 0 0 3 5
>> vect4 = [vect1; vect2]	vect4 = 1 0 8 2 0 0 3 5

To perform the second concatenation, both vectors must have the same number of elements.

2) The concatenation of two column vectors

In the command window, type:

Commands	Results
>> vec1 = [0; 1; 2];	
>> vect2 = [3; 5];	
>> vect3 = [vect1; vect2]	vect3 = 0 1 2 3 5
>> vect4 = [10; 20];	
>> vect5 = [vect2 vect4]	vect4 = 3 10 5 20

To perform the second concatenation, both vectors must have the same number of elements.

VI.4.6 Other Operations.

Operation	Description
.*	Multiply two vectors component by component.
./	Divide the components of two vectors in pairs.
. ^	Raise the components of one vector to the power of the components of the second vector.
$sum(u)$	Sum of the components of a vector.
$mean(u)$	Average of the components of a vector.
$length(u)$	Gives the length of a vector.
$min(u)$	Gives the smallest component of a vector.
$max(u)$	Gives the largest component of a vector.

VII. Matrices.

VII.1 Creating a matrix.

In MATLAB, you can create a matrix by entering elements in each row and using semicolons to mark the end of each row.

Example

In the command window, type:

Commands	Results
>> M1 = [1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8]	M1 = 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7 4 5 6 7 8
>> M2 = [1, 2, 3, 4, 5; 2, 3, 4, 5, 6; 3, 4, 5, 6, 7; 4, 5, 6, 7, 8]	M2 = 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7 4 5 6 7 8

VII.2 Elements access.

1. To access an element in the $i^{'eme}$ row and $j^{i'eme}$ column, we write matrix_name (row_index, column_index)

Example

Commands	Results
>> M = [1 5 3;4 9 2;1 0 9];	
>> M (2, 3)	ans = 2

2. To access all elements of a column matrix_name (: , column_index)

Example

Commands	Results
>> M = [1 5 3;4 9 2;1 0 9];	
>> M (:, 3)	ans = 3 2 9

3. To access all the elements of a line matrix_name (line_index, :)

Example

Commands	Results
>> M = [1 5 3;4 9 2;1 0 9];	
>> M (1, :)	ans = 1 5 3

4. You can access all the elements of several lines by matrix_name (indexFirstLine : lastLineIndex, :)

Example

Commands	Results
>> M = [1 5 3;4 9 2;1 0 9];	
>> M (2: 3, :)	ans = 4 9 2 1 0 9

5. You can access all the elements of several columns by matrix_name (: firstcolumnindex : lastColumnIndex)

Example

Commands	Results
>> M = [1 5 3;4 9 2;1 0 9];	
>> M (:, 2: -1:1)	ans = 5 1 9 4 0 1

6. You can access a sub-matrix matrix_name (firstRowIndex : lastRowIndex, firstColumnIndex : lastColumnIndex)

Example

Commands	Results
>> M = [1 5 3;4 9 2;1 0 9];	
>> M (2: 3, 2: -1:1)	ans = 9 4 0 1

VII.3 Elements Modification.

You can modify the elements of a matrix by selecting the elements to be modified and assigning them new values

Example

Commands	Results
>> M = [1 2 3 4; 5 6 7 8];	
>> M (1: 2, 2: 3) = [1 2; 5 3]	ans = 1 1 2 4 5 5 3 8

MATLAB also has a table editor that lets you modify the dimensions and entries of a vector or matrix. To use this editor, double-click on the variable to be edited in the Workspace.

VII.4 Deleting rows or columns.

You can delete a row or column from a matrix by assigning an empty set of brackets [] to that row or column. [] denotes an empty matrix.

Example

In the command window, type

Commands	Results
>> M = [1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];	
>> M (4 , :) = []	M = 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7
>> L = M;	
>> L (: , 2) = []	L = 1 3 4 5 2 4 5 6 3 5 6 7

VII.5 Operation on matrices.

VII.5.1 Addition and subtraction.

You can add or subtract two matrices. Both matrices must have the same number of rows and columns.

Example

In the command window, type

Commands	Results
>> A = [1 2 3; 4 5 6; 7 8 9];	
>> B = [9 8 7; 6 5 4; 3 2 1];	
>> C = A + B	C = 10 10 10 10 10 10 10 10 10
>> D = A - B	D = -8 -6 -4 -2 0 2 4 6 8

VII.5.2 Scalar operations.

When you add, subtract, multiply or divide a matrix by a number, this is called a scalar operation.

Scalar operations produce a new matrix with the same number of rows and columns, with each element of the original matrix added, subtracted, multiplied by or divided by the number.

Example

In the command window, type

Commands	Results
>> M = [10 12 24; 14 8 6; 28 8 10];	
>> s = 2;	
>> A = M + s	A = 12 14 26 16 10 8 30 10 12
>> S = M - s	S = 8 10 22 12 6 4 26 6 8
>> P = M * s	P = 20 24 48 28 16 12 56 16 20
>> D = M / s	D = 5 6 12 7 4 3 14 4 5

VII.5.3 Matrix division.

You can divide two matrices using left (\) or right (/) division operators. Both matrices must have the same number of rows and columns.

Example

In the script editor, type

Commands
M1 = [1 0 2; 0 1 9; 3 0 1] M2 = [0 0 3; 4 5 0; 1 8 9] RD = M1 / M2 LD = M1 \ M2
Results
RD = 1.2222 0.2963 -0.1852

```
        2.5556 -0.0370  0.1481
        2.0000  0.8889 -0.5556
LD =    0.4000  3.2000  3.0000
        5.8000 19.4000 -0.0000
       -0.2000 -1.6000  0.0000
```

Note

1. A/B is the solution to the equation $xB = A$.
2. $A\backslash B$ is the solution to the equation $Ax = B$.

VII.5.4 Transposed matrix.

The transposition operation switches rows and columns in a matrix. It is represented by a single quotient (').

Example

In the command window, type

Commands	Results
>> A = [10 12 23; 14 8 6; 27 8 9]	A = 10 12 23 14 8 6 27 8 9
>> B = A'	B = 10 14 27 12 8 8 23 6 9

VII.5.5 Concatenation of matrices.

MATLAB lets you concatenate matrices in two ways:

1) Horizontal concatenation

In the script editor, type

Commands
M1 = [10 12 23; 14 8 6; 27 8 9]; M2 = [12 31 45; 8 0 -9; 45 2 11]; A = [M1, M2]
Results
A = 10 12 23 12 31 45 14 8 6 8 0 -9 27 8 9 45 2 11

2) Vertical concatenation

In the script editor, type

Commands
M1 = [10 12 23; 14 8 6; 27 8 9]; M2 = [12 31 45; 8 0 -9; 45 2 11]; B = [M1; M2]
Results
B = 10 12 23 14 8 6 27 8 9 12 31 45 8 0 -9 45 2 11

VII.5.6 Multiplication of matrices.

In the command window, type

Commands	Results
>> M = [1 2 3; 2 3 4; 1 2 5];	
>> N = [2 1 3 ; 5 0 -2; 2 3 -1];	
>> P = M * N	P = 18 10 -4 27 14 -4 22 16 -6

VII.5.7 Determinant.

The determinant of a matrix is calculated using MATLAB's det function.

Example

In the command window, type

Commands	Results
>> A = [1 2 3; 2 3 4; 1 2 5];	
>> det (A)	ans = -2

VII.5.8 Inverse of a matrix.

The inverse of a matrix is calculated using MATLAB's inv function.

Example

In the command window, type

Commands	Results
>> A = [1 2 3; 2 3 4; 1 2 5];	
>> inv(A)	ans = -3.5 2 0.5

	3 -1 -1 -0.5 0 0.5

VII.6 Special matrices

In the command window, type

Commands	Results
>> zeros (3)	ans = 0 0 0 0 0 0 0 0 0
>> zeros (3,2)	ans = 0 0 0 0 0 0
>> ones (4,3)	ans = 1 1 1 1 1 1 1 1 1 1 1 1
>> eye (4)	ans = 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
>> eye (3,4)	ans = 1 0 0 0 0 1 0 0 0 0 1 0

Remarks

Control	Description
zeros ()	The zeros () function creates a null matrix.
ones ()	The ones () function creates a matrix of 1's.
eye ()	The eye () function creates an identity matrix.

VIII. Functions.

VIII.1. Functions

A function is a group of instructions that together perform a task (similar to scripts but scripts don't have parameters or inputs.). In MATLAB, functions are defined in separate files. The name of the file and the function must be the same.

Syntax

```
function [out1, out2, ..., outn] = myfun (in1, in2, ..., inn)
        instructions
end
```

Here, the name of the function is myfun, in_1 to in_n are the inputs or the parameters and out_1 to out_n are the outputs or the results of the execution of the function.

Example

In the script editor window, type this code and save it as mymax

```
function max = mymax (n1, n2, n3, n4, n5)
   max = n1;
   if (n2 > max)
      max = n2;
   end
   if (n3 > max)
      max = n3;
   end
   if (n4 > max)
      max = n4;
   end
   if (n5 > max)
      max = n5;
   end
end
```

In the command window, type

Commands	Results
>> mymax (5, 7, 19, 0, 23)	ans = 23

Note

Function files are program files with .m extension.

VIII.2. Multiple Functions

Multiple functions can be included in the same function file. MATLAB will refer to

that collection of functions only through the file name. This structure is useful when functions in the file need to call one another.

Syntax

```
function [out1, ..., outn] = mainFunction (in1, ..., inn)
        instructions that call subFunction
end
%-------------------------------------------------
function [out1, ..., outn] = subFunction (in1, ..., inn)
        instructions
end
```

Example

In the script editor window, type this code and save it as myFunction.m.

```
function [p, q] = myFunction (x, y, z)
n = x * z;
m = x * y;
[v, w] = mySubFunction (n, m);
if (w~=0)
        p = v/w;
else
        p = [];
end
q = z - (v * w);
end
%--------------------------------------
function [a, b] = mySubFunction (c, d)
a = c + d ;
b = c – d ;
end
```

In this example, the script calls myFunction. myFunction will be visible to MATLAB while mySubFunction will not, it is accessible only within myFunction.
Multiple functions are particularly useful for writing graphical user interface (GUI) code.

VIII.3. Inline (Anonymous) Functions.

An inline function is a function that we can write in one line and use without saving it to a file.

Syntax

```
functionName = @(variables)  function syntax;
```

Example

In the command window, type:

Commands	Results
>> func = @(x) 2*x + exp (x);	
>> func (5)	ans = 158.4132

Example

In the command window, type:

Commands	Results
>> g = @(x, y) x+y;	
>> g (10, 44)	ans = 54