جامعة بغداد / كلية العلوم
قسم علوم الحاسوب
مختبر التحليل العددي/ المرحلة الثانية
2021-2020

م.م نهلة عبد الرحمن حسين

لغة البرمجـــــــــة MATLAB:)The MATLAB programming language(

1-مقدمة عن لغة الماتلاب
2 .الثوابت والمتغٌٌرات.
3 -المصفوفات والعملٌات على المصفوفات.
4 -المصفوفات متعددة ألابعاد.
5 -مصفوفات الخالٌا.
6 -السلاسل والرموزٌ.
7 -جمل اإلدخال واإلخراج.
8 -الجمل الشرطٌـــة.
9 -جمل الدوران والتكرار.
10 -الدوال والبرامج الفرعٌة.
11 -الرسوم البٌٌانٌة.

المحاضرة الاولى

مقدمة عن لغة الماتلاب

MATLAB is a programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models.

MATLAB combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly. It includes the Live Editor for creating scripts that combine code, output, and formatted text in an executable notebook.

Variables

Variables are defined using the assignment operator,

MATLAB is a weakly typed programming language because types are implicitly converted.[

It is an inferred typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change.

Values can come from constants, from computation involving values of other variables, or from the output of a function.

For example:

```
>> x = 17 x = 17 >> x = 'hat' x = hat >> x = [3*4,
/2] x = 12.0000 1.5708 >> y = 3*
(x) y = -1.6097 3.0000
```

المحاضرة الثانية
Examples

Ex. 1 Write your first Matlab program

```
a = 3;
b = 5;
c = a+b
```

(1) Output: The semicolon at the end of a statement acts to suppress output (to keep the program running in a "quiet mode").

(2) The third statement, c = a+b, is not followed by a semicolon so the content of the variable c is "dumped" as the output.

Ex. 2 The meaning of "a = b

In Matlab and in any programming language, the statement "a = b" does not mean "a equals b".

Instead, it prompts the action of replacing the content of a by the content of b. a = 3; b = a; b Output: 3

Remark: Think of the two "variables" a and b as two buckets labeled "a" and "b". The first statement puts the number 3 into bucket a. The second statement puts the content of bucket a into bucket b, such that we now have "3" in bucket b.

(The content of bucket a remains unchanged after this action.) The third statement dumps the content of bucket b so the final output is "3".

Ex. 3 Basic math operations a = 3; b = 9;
c = 2*a+b^2-a*b+b/a-10

Output: 53 Remark: The right hand side of the third statement includes all 4 of the basic arithmetic operators, + (addition), - (subtraction), * (multiplication), and / (division), in their usual meaning. It also includes the symbol, ^, which means "to the power of", so "b^2" means (the content of b) to the power of 2, i.e., 92 = 81.

The right hand side of that statement is first evaluated: r.h.s. = 2 x 3 + 92 - 3 x 9 + 9/3 - 10 = 53. The content of r.h.s., now 53, is then assigned to the variable c in the left hand side. Since this statement is not followed by a semicolon, the content of c is dumped as the final output.

Ex. Formatted output

a = 3; b = a*a; c = a*a*a; d = sqrt(a);
fprintf('%4u square equals %4u \r', a, b) fprintf('%4u cube equals %4u \r', a, c) fprintf('The square root of %2u is %6.4f \r', a, d) Output: 3 square equals 9 3 cube equals 27 The square root of 3 is 1.7321

Remarks: The command "fprintf" is for formatted output, using the format specified in the first string '...' in the parentheses. The "%4u" (4-digit integer) and "%6.4f" (real number that preserves 4 digits to the right of the floating point) are the format for the variable(s) for output. The "sqrt" in the 4th statement is the intrinsic function for square root

المحاضرة الثالثة

Matlab Commands

Matlab Commands:

Conditions الجمل الشرطية

if, else if, else

While, end

if, else if, else

Execute statements if condition is true Otherwise, the expression is false.

Syntax

if *expression statements* else if *expression statements* else *statements* end

Example

Find the value of y

Y= x-3   x>=0

Y=x^2   x<0

while

While loop to repeat when condition is true

Syntax

while *expression statements* **end**

Description

while *expression*, *statements*, end evaluates an expression, and repeats the execution of a group of statements in a loop while the expression is true.

An expression is true when its result is nonempty and contains only nonzero elements

(logical or real numeric). Otherwise, the expression is false.

Example

Use a while loop to calculate factorial(10).

```
n = 10;
 f = n;
while n > 1
 n = n-1;
f = f*n;
end
disp(['n! = ' num2str(f)])
```

```
n! = 3628800
```

المحاضرة الرابعة

Numerical Analysis Methods

Bisection Method

•**Bisection Method** is a **numerical method** in **Mathematics** to find a **root** of a given ***function within an interval.***

Root of a function *f(x)* = a **value *a*** such that:

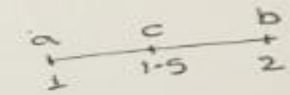
•***f(a) = 0***

Fixed point :- هي ايجاد اكثر من صوره للمعادله الاصليه

Bisection

$f(x) = x^2 - 3 \quad ; \quad [a, b]$

$c = \frac{a+b}{2} = 1.5 \quad [1, 2]$



$f_a = 1^2 - 3 = -2$

$f_c = 1.5^2 - 3 = -0.75$

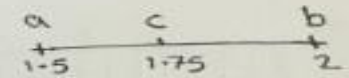
$f_a f_c = (-2)(-0.75) = +$

$a = c = 1.5$

$b = 2$

$b - a = 2 - 1.5 = 0.5$

$\left.\begin{array}{l} a = 1.5 \\ b = 2 \end{array}\right\} c = \frac{1.5+2}{2} = 1.75$



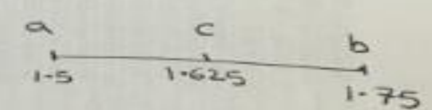
$f_a = 1.5^2 - 3 = -0.75$

$f_c = 1.75^2 - 3 = 0.0625$

$f_a f_c = -$

$\left.\begin{array}{l} b = c = 1.75 \\ a = 1.5 \end{array}\right\} c = \frac{1.5+1.75}{2} = 1.625$

$b - a = 1.75 - 1.5 = 0.25$



$f_a = -0.75$

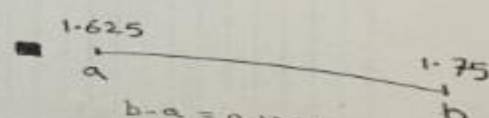
$f_b = 0.0625$

$f_c = -0.359$

$f_a f_c = +$

$a = c = 1.625$

$b = 1.75$



$b - a = 0.125$

a	b	c	fa	fc	Update	new b-a
1	2	1.5	-2	-0.75	a=c	0.5
1.5	2	1.75	-0.75	0.0625	b=c	0.25
1.5	1.75	1.625	-0.75	-0.359	a=c	0.125

c = a+b/2

if f(a) * f(c) < 0 then b=c

else a=c

Bisection Method

Find the root of non-linear equation using Bisection Method in MATLAB

Find the root of **f(*x*) = x^2 - 3**. with the interval **[1, 2]**.

a	*b*	f(*a*)	f(*b*)	*c* = (a + b) /2	f(*c*)	Update	new b − a
1.0	2.0	-2.0	1.0	1.5	-0.75	a = c	0.5
1.5	2.0	-0.75	1.0	1.75	0.062	b = c	0.25
1.5	1.75	-0.75	0.0625	1.625	-0.359	a = c	0.125
1.625	1.75	-0.3594	0.0625	1.6875	-0.1523	a = c	0.0625
1.6875	1.75	-0.1523	0.0625	1.7188	-0.0457	a = c	0.0313
1.7188	1.75	-0.0457	0.0625	1.7344	0.0081	b = c	0.0156
1.71988/td>	1.7344	-0.0457	0.0081	1.7266	-0.0189	a = c	0.0078

For Command

تقوم حلقات باعادة تنفيذٌ مجموعة من الأوامر لعدد معينٌ من المرات خطوة معٌينة, وتعطى

الصيغٌة العامة لحلقة for كما يلٌي :

```
for i = x1: x3: x2
(commands)
end;
```

X1 : هي القيمة الابتدائية (Initial value)

X2 : القيمة النهائية (Final value)

X3 : مقدار الزيادة في كل مرة (step size)

Example:

```
for n = 1: 10
x (n) = sin (n * pi / 10);
end;
```

نتيجة البرنامج

```
>> x
x =
Columns 1 through 7
0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090
Columns 8 through 10
0.5878 0.3090 0.0000
```

```
for i = 0: 2: 10
disp (i);
end;
```

الإخراج

0

2

4

6

8

10

```
for i = 10: -2: 1
disp (i);
end;
```

الإخراج

```
10
8
6
4
2
```

برنامج لطبع جدول الضرب

```
for i =1: 10
for j = 1: 10
mult (i, j) = i * j;
end;
end;
```

نتيجة البرنامج

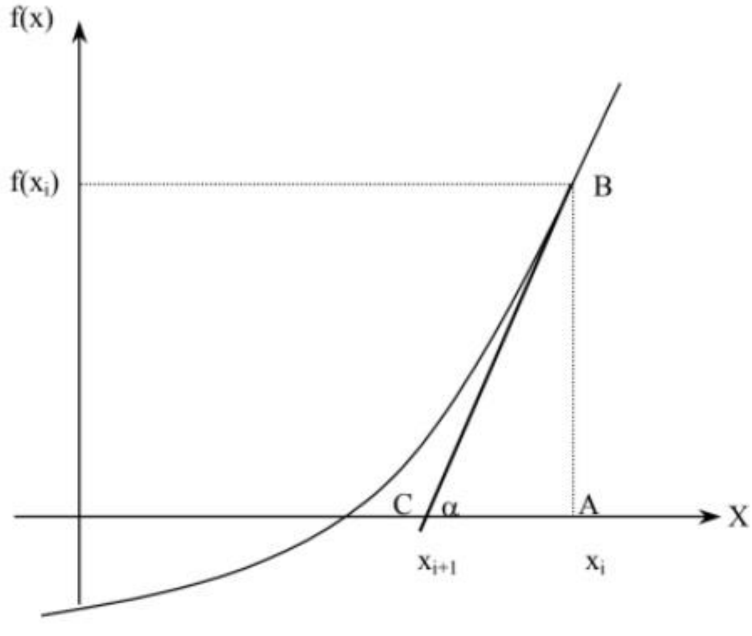
```
 1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 3 6 40
. . . . . . . . . . . .
. . . . . . . . . . . .
10 20 30 40 50 60 70 80 90 100
```

المحاضرة الخامسة

Newton Raphson Method

Newton-Raphson method

- The **Newton-Raphson method** (also known as **Newton's method**) is a way to quickly find a good approximation for the root of a real-valued function.
- **f (x) is continues and it has a derivative**
- It uses the idea that a continuous and differentiable function can be approximated by a straight line tangent to it.



$$\tan(\alpha) = \frac{AB}{AC}$$

$$f'(x_i) = \frac{f(x_i)}{x_i - x_{i+1}}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$\tan \alpha = \frac{AB}{AC}$$

$$f'(Xi) = \frac{f(xi)}{Xi - X_{i+1}}$$

$$f(xi) = f'(Xi)\,(Xi - X_{i+1})$$

$$\frac{f(x)}{f'(xi)} = (Xi - X_{i+1})$$

$$\boldsymbol{X_{i+1} = Xi - \frac{f(xi)}{f'(Xi)}}$$

Algorithm

Newton Raphson method Steps (Rule)	
Step-1:	Find points a and b such that $a<b$ and $f(a) \cdot f(b) < 0$.
Step-2:	Take the interval $[a,b]$ and find next value $x0=(a+b) \ /2$
Step-3:	Find $f(x0)$ and $f\text{‘}\ (x0)$ $x1=x0-f(x0)$ / $f'(x0)$
Step-4:	If $f(x1)=0$ then $x1$ is an exact root, else $x0=x1$
Step-5:	Repeat steps 2 to 4 until $f(xi)=0$ or $\lvert f(xi) \rvert \le \text{Accuracy}$

$f(x) = x^4 - x - 10$; $x_0 = 2$

$f'(x) = 4x^3 - 1$

$f(2) = 2^4 - 2 - 10 = 4 \neq 0$

$f'(2) = 4(2)^3 - 1 = 31$

$x_1 = x_0 - \frac{f(x)}{f'(x)} = 2 - \frac{4}{31} = 1.87096$

$\boxed{x_1 = 1.87096}$

$f(x_1) = 0.3583 \neq 0$

$f'(x_1) = 4(1.87)^3 - 1.87 = 24.2868$

$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1.87 - \frac{0.3583}{24.2868}$

$\boxed{x_2 \approx 1.855}$ 24 $\boxed{|x_2 - x_1| = 0.015}$

$f(x_2) = -0.01434 \neq 0$

$f'(x_2) = 5.383$

$x_3 = 1.85524 + \frac{0.01434}{5.383} = 1.857$

$\boxed{x_3 = 1.857}$

$\boxed{|x_3 - x_2| = 0.002}$

Example:-

F (x) = x^2 - 5 on the interval [1,3], X1=2

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - 5}{2x_n}.$$

$$
\begin{aligned}
x_1 &= 2 \\
x_2 &= 2.25 \\
x_3 &= 2.23611111111111111111111111111111 \\
x_4 &= 2.23606797791580400276052449965493 \\
x_5 &= 2.23606797749978969644787282832711 \\
x_6 &= 2.23606797749978969640917366873127
\end{aligned}
$$

It is quite remarkable that the results stabilize for more than ten decimal places after only 5 iterations

Example

Let's approximate the root of the following function with Newton Raphson Method

$$f(x) = x - \cos(x)$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n - \cos(x_n)}{1 + \sin(x_n)}.$$

$$\begin{aligned}
x_1 &= 1. \\
x_2 &= 0.750363867840243893034942306682177 \\
x_3 &= 0.739112890911361670360585290904890 \\
x_4 &= 0.739085133385283969760125120856804 \\
x_5 &= 0.739085133215160641661702625685026 \\
x_6 &= 0.739085133215160641655312087673873 \\
x_7 &= 0.739085133215160641655312087673873 \\
x_8 &= 0.739085133215160641655312087673873
\end{aligned}$$

Example

Let us to solve **$f(x) = \cos x - 2x$** to 5 decimal places.

Make sure your calculator is in radian mode.

The recursion formula (1) becomes

$x_{n+1} = x_n - (\cos x_n - 2x_n) / (-\sin x_n - 2)$

With an initial guess of $x_0 = 0.5$,

we obtain: $x_0 = 0.5$

$x_1 = 0.45063$ $x_2 = 0.45018$ $x_3 = 0.45018 \ldots$

with no further changes in the digits, to five decimal places.

Therefore, to this degree of accuracy, the root is $x = 0.45018$.

Possible problems with the method

The Newton-Raphson method works most of the time if your initial guess is good enough.

Occasionally it fails but sometimes you can make it work by changing the initial guess.

Let's try to solve f(x) = 0 where **$f(x) = x - \tan x$.**

The recursion formula (1) becomes

$x_{n+1} = x_n - (x_n - \tan x_n) / (1 - \sec^2 x_n)$ Let's try an **initial guess of $x_0 = 4$.**

With this initial guess we find that $x_1 = 6.12016$,

$x_2 = 238.40428$,

$x_3 = 1957.26490$, etc.

Clearly these numbers are not converging. We need a new initial guess.

Let's **try $x_0 = 4.6$.**

Then we find $x_1 = 4.54573$, $x_2 = 4.50615$, $x_3 = 4.49417$, $x_4 = 4.49341$, $x_5 = 4.49341$, etc.

A couple of further iterations will confirm that the digits are no longer changing to 5 decimal places.

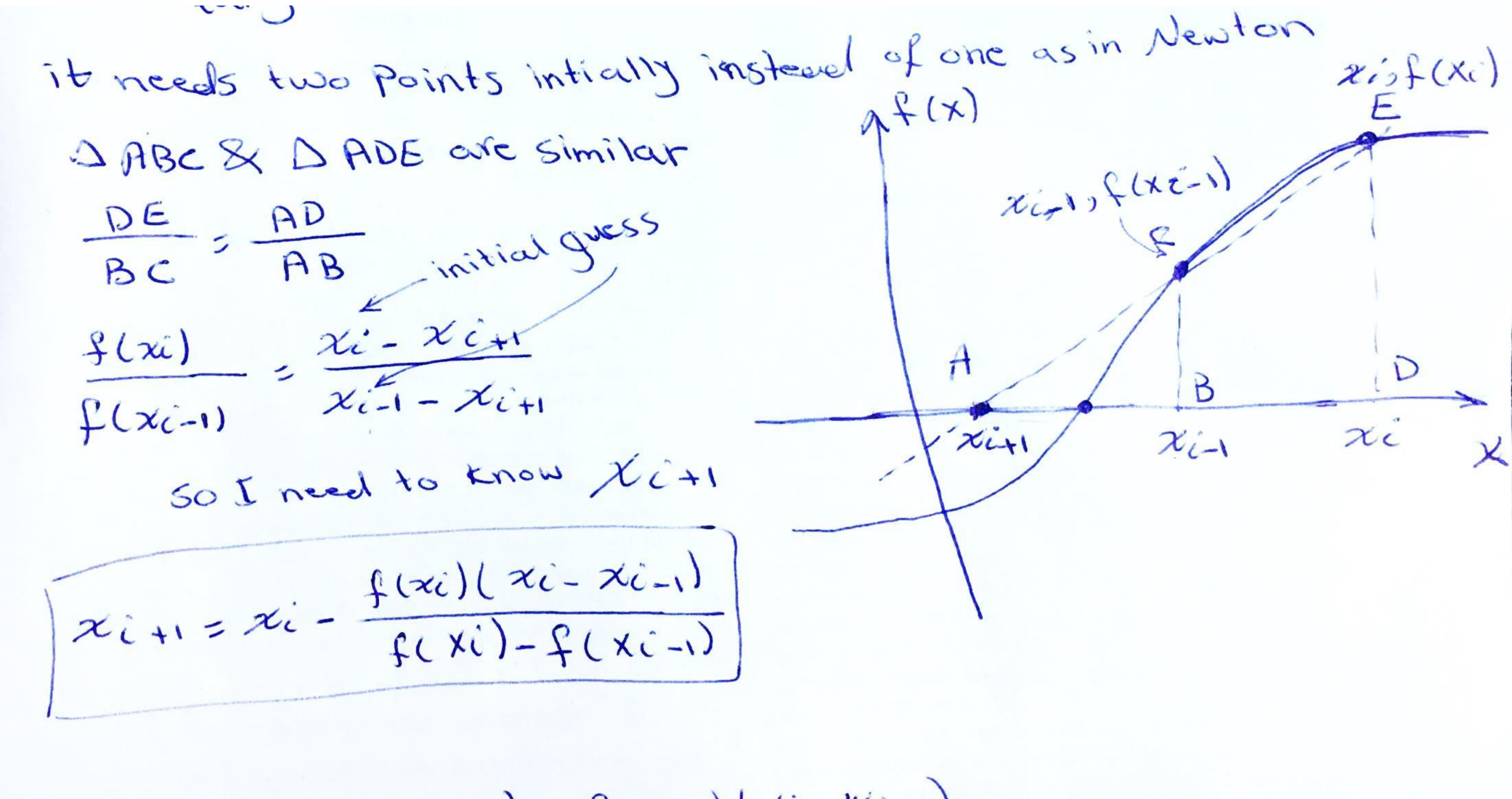
As a result, we conclude that a root of x = tan x is x = 4.49341 to 5 decimal places.

المحاضرة السادسة

Secant Method

Secant method

In numerical analysis, the **secant method** is a root-finding **algorithm** that uses a succession of roots of **secant lines** to better approximate a root of a function f. Using a secant instead of a tangent line.



Secant Method Algorithm

1- Take the interval $[X_1, X_2]$

2- find next value $X_3 = X_1 - f(X_1) \cdot (X_2 - X_1) / f(x_2) - f(x_1)$

3- If $f(X_3) = 0$ then X_3 is an exact root,

else $X_1 = X_2$ and $X_2 = X_3$

4- Repeat steps 2 & 3 until $f(X_3) = 0$ or $|X_3 - X_2| \leq$ Accuracy

Secant Method

$f(x) = x^3 - x - 1$; $x_0 = 1$ and $x_1 = 2$

$f(x_0) = 1^3 - 1 - 1 = -1$

$f(x_1) = f(2) = 2^3 - 2 - 1 = 5$

$$x_2 = x_0 - f(x_0) \cdot \frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

$$x_2 = 1 - (-1) \cdot \frac{(2-1)}{(5) - (-1)} = 1 + 1 \cdot \frac{1}{7}$$

$\boxed{x_2 = 1.16667}$

$f(x_2) = f(1.16667) = 1.16667^3 - 1.16667 - 1$

$f(x_2) = -0.5787 \neq 0 \Rightarrow x_2$ is not the root

2nd iteration $x_1 = 2$ and $x_2 = 1.16667$

$f(x_1) = f(2) = 2^3 - 2 - 1 = 5$

$f(x_2) = f(1.16667) = -0.5787$

$$x_3 = x_1 - f(x_1) \cdot \frac{x_2 - x_1}{f(x_2) - f(x_1)} = 2 - 5 * \frac{1.6667 - 2}{-0.5787 - 5}$$

$\boxed{x_3 = 1.25311}$

$f(x_3) = f(1.25311) = 1.25311^3 - 1.25311 - 1 = -0.28536 \neq 0$

3rd iteration

$x_2 = 1.16667$ and $x_3 = 1.25311$

$f(x_2) = f(1.16667) = -0.5787$

$f(x_3) = f(1.25311) = -0.28536$

$$x_4 = x_2 - f(x_2) \cdot \frac{x_3 - x_2}{f(x_3) - f(x_2)} = 1.16667 - (-05787) \frac{1.25311 - 1.1666}{-0.28536 - (-0.57}$$

$\boxed{x_4 = 1.33721}$

$f(x_4) = f(1.33721) = 0.05388 \neq 0$

$\vdots$

$\boxed{x_6 = 1.32471}$; $f(x_6) = f(1.32471) = -0.00004$

n	$x0$	$f(x0)$	$x1$	$x2$	$f(x2)$	Update
1	1	-1	2	1.16667	-0.5787	$x0=x1$ $x1=x2$
2	2	5	1.16667	1.25311	-0.28536	$x0=x1$ $x1=x2$
3	1.16667	-0.5787	1.25311	1.33721	0.05388	$x0=x1$ $x1=x2$
4	1.25311	-0.28536	1.33721	1.32385	-0.0037	$x0=x1$ $x1=x2$
5	1.33721	0.05388	1.32385	1.32471	-0.00004	$x0=x1$ $x1=x2$

المحاضرة السابعة
False Position Method

False Position Method

The **false-position method** is a modification on **the bisection method**:
if it is known that the root lies on [a, b], then it is reasonable that we can approximate the function on the interval by interpolating the points (a, f(a)) and (b, f(b)).
This method attempts to solve an equation of the form f(x)=0.
The algorithm requires a function f(x) and two points a and b for which f(x) is positive for one of the values and negative for the other.
We can write this condition as f(a) * f(b) < 0.

1) The false Position method can be derived from geometry:

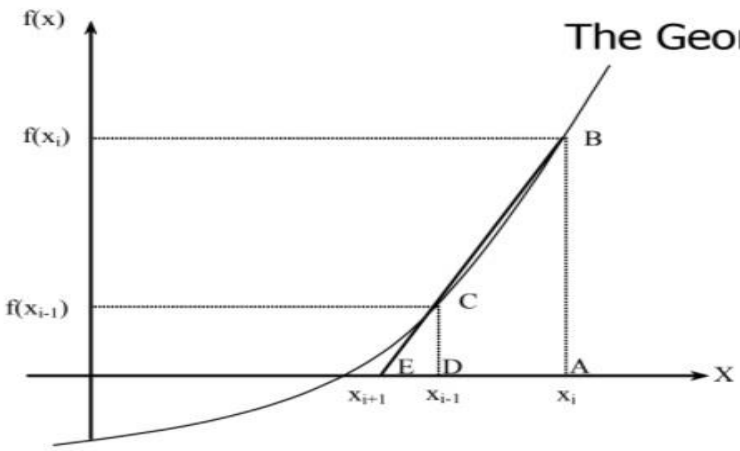


Figure 1 Geometrical representation of the false position method.

The Geometric Similar Triangles ABE and DCE

$$\frac{AB}{AE} = \frac{DC}{DE}$$

can be written as

$$\frac{f(x_i)}{x_i - x_{i+1}} = \frac{f(x_{i-1})}{x_{i-1} - x_{i+1}}$$

On rearranging, the false position method is given as

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

False Position Method Algorithm

False Position method Steps (Rule)	
Step-1:	Find points $x0$ and $x1$ such that $x0<x1$ and $f(x0) \cdot f(x1)<0$.
Step-2:	Take the interval $[x0,x1]$ and find next value $x2=x0-f(x0) \cdot \{(x1-x0) /(f(x1)-f(x0))$
Step-3:	If $f(x2)=0$ then $x2$ is an exact root, else if $f(x0) \cdot f(x2)<0$ then $x1=x2$, else if $f(x1) \cdot f(x2)<0$ then $x0=x2$.
Step-4:	Repeat steps 2 & 3 until $f(xi)=0$ or $\lvert f(xi)\rvert \leq \text{Accuracy}$

Find a root of an equation $f(x)=2X^3 - 2X - 5$ where X0=1 , X1=2 using False Position method

<u>1*st* iteration</u> :

Here f(x0) =$f(1)= 2*(1)^3-2*(1) -5= -5<0$

and f(x1)= $f(2) = 2*(2)^3 -2*(2) -5= 7>0$

∴ Now, Root lies between ***X* 0=1 and *X* 1=2**

X2= X0 – f(x0) . $\frac{X1-X0}{f(x1)-f(x0)}$

X2 = 1- (-5) . $\frac{2-1}{7-(-5)}$

<u>X2=1.41667</u>

$f(X2)=f(1.41667)=2\cdot(1.41667)^3-2\cdot1.41667-5= - 2.14699<0$

<u>2*nd* iteration</u> :

Here $f(1.41667)=-2.14699<0$ and $f(2)=7>0$

∴ Now, Root lies between *X* 0=1.41667 and *X* 1=2

X 3= **X0 – f(x0) .** $\frac{X1-X0}{f(x1)-f(x0)}$

*X*3=1.41667-(-2.14699) . $\frac{2-1.41667}{7-(-2.14699)}$

*X*3=1.55359

$f(X3)=f(1.55359)=2\cdot(1.553593)^3-2(\cdot1.55359) - 5=- 0.60759<0$

3*rd* iteration :

Here $f(1.55359)=-0.60759<0$ and $f(2)=7>0$

∴ Now, Root lies between **$X_0=1.55359$ and $X_1=2$**

$$X_4 = X_0 - f(x_0) \cdot \frac{X_1 - X_0}{f(x_1) - f(x_0)}$$

$$X_4=1.55359-(-0.60759) \cdot \frac{2-1.55359}{7-(-0.60759)}$$

$X_4=1.58924$

$f(x_4)=f(1.58924)=2\cdot(1.58924)^3-2\cdot(1.58924)-5=-0.15063<0$

4*th* iteration :

Here $f(1.58924)=-0.15063<0$ and $f(2)=7>0$

∴ Now, Root lies between **$X_0=1.58924$ and $X_1=2$**

$$X_5 = X_0 - f(x_0) \cdot \frac{X_1 - X_0}{f(x_1) - f(x_0)}$$

$$X_5=1.58924-(-0.15063) \cdot \frac{2-1.58924}{7-(-0.15063)}$$

$X_5=1.59789$

$f(X_5)=f(1.59789)=2\cdot(1.59789)^3-2\cdot(1.59789)-5=-0.0361<0$

- - - -

- - - -

$x_8=1.60056$

$f(x_8)=f(1.60056)=2\cdot1.60056^3-2\cdot1.60056-5=-0.00048<0$

n	$x0$	$f(x0)$	$x1$	$f(x1)$	$x2$	$f(x2)$	Update
1	**1**	-5	**2**	7	1.41667	-2.14699	$x0=x2$
2	**1.41667**	-2.14699	**2**	7	1.55359	-0.60759	$x0=x2$
3	**1.55359**	-0.60759	**2**	7	1.58924	-0.15063	$x0=x2$
4	**1.58924**	-0.15063	**2**	7	1.59789	-0.0361	$x0=x2$
5	**1.59789**	-0.0361	**2**	7	1.59996	-0.00858	$x0=x2$
6	**1.59996**	-0.00858	**2**	7	1.60045	-0.00203	$x0=x2$
7	**1.60045**	-0.00203	**2**	7	1.60056	-0.00048	$x0=x2$

Fixed Point Method

Fixed Point Iteration Method : A **point**, say, s is called a **fixed point** if it satisfies the equation x = g(x).

with some initial guess x_0 is called the **fixed point** iterative scheme.

Y= x^3 **−7X+ 2**

x^3 **−7X+ 2 = 0**

In this method, we first rewrite the equation in the form x = g(x)

X = 1/ 7 * (x^3 **+ 2)**

and define the process **Xn+1 = 1/ 7 (**x^3 **n + 2)**

OR

X= $\sqrt[3]{7X-2}$

Xn+1= $\sqrt[3]{7Xn-2}$

Which one should be chosen?

Fixed point Iteration Method Algorithm

1- Start

2- Read values of x0 and e.

***Here x_0 is the initial approximation**

e is the absolute error or the desired degree of accuracy,

also the stopping criteria*

3- Calculate x1 = g(x_0)

4- If [x1 – x0] <= e, goto step 6.

Here [ ] refers to the modulus sign

5- Else, assign x0 = x1 and goto step 3.

6- Display x1 as the root.

7- Stop

For Convergance : $|g'(xo)| < 1$

$x = 1 + 0.5 \sin x$

Here $g(x) = 1 + 0.5 \sin x$

We can take $[a, b]$ with any $a \leq 0.5$ and $b \geq 1.5$.

Note that $g'(x_o) = 0.5 \cos x$, $|g'(x_o)| \leq 1/2$

Therefore, we can conclude that the fixed point iteration $x_{n+1} = 1 + .5 \sin x_n$ will converge

Example 8 -

$f(x) = x^3 + x - 1 \quad ; \; x_0 = 1$

write the equation in the form of fixed point $x = g(x) \quad ; \; x_{n+1} = g(x_n)$

$x^3 + x - 1 = 0$

$x = \underbrace{1 - x^3}_{g(x)}$

$x_{n+1} = 1 - x_n^3$

$x_0 = 1$

$x_1 = 1 - x_0^3$

$x_1 = g(x_0) =$

$x_1 = g(1) = 0$

$x_2 = g(0) = 1$

$x_3 = g(1) = 0$

$x_4 = g(0) = 1$

$x_5 = g(1) = 0$

$\vdots$

Does not Converges

$x^3 + x - 1 = 0$

$x(x^2 + 1) - 1 = 0$

$x = \underbrace{\frac{1}{x^2 + 1}}_{g(x)}$

$x_{n+1} = \frac{1}{x_n^2 + 1}$

$x_0 = 1$

$x_1 = g(x_0) = \frac{1}{1^2 + 1} = 0.5$

$x_2 = g(0.5) = 0.8$

$x_3 = g(0.8) = 0.61$

$x_4 = g(0.61) = 0.729$

$x_5 = g(0.729) = 0.653$

$x_6 = g(0.653) = 0.701$

$x_7 = 0.671$

$\vdots$

$x_{14} = \underline{0.6827}$

$x_{15} = \underline{0.6821}$

$x_{16} = \underline{0.6824}$

3 Decimal Places.

$x^3 + x - 1$

$x = 1 - x^3$

$g(x) = 1 - x^3$

$g'(x) = -3x^2$

$|g'(x_0)| = 3 > 1$ Does not Converges

$x = \frac{1}{x^2+1}$

$g(x) = \frac{1}{x^2+1}$

$g'(x) = \frac{-2x}{(x^2+1)^2}$

$|g'(x_0)| = g'(1) = \left| \frac{-2}{(1+1)^2} \right| = 0.5 < 1$

$|g'(1)| = \left| \frac{-2}{(1^2+1)^2} \right| = \left| \frac{-2}{4} \right| = 0.5 < 1$

Converges

1- write $x = g(x)$; $x_{n+1} = g(x_n)$

2- x_0

3- $g'(x)$

4- $|g'(x_0)| < 1$

It's too long iteration, about 15-20 iteration it needs

Finding roots with **Fixed Point iteration**

- Given $f(x)=0$ write x interms of
  $x = \cdots$
- Label left side as x_{n+1} and right side with x_n
- Pick x_1 and plug into quation
- Repeat until convergens.

Ex: find where $x^2 - x - 1 = 0$

$$x^2 - x - 1 = 0$$

$x^2 = x + 1$ or $x^2 - x = 1$

$x = 1 + \frac{1}{x}$ $\quad$ $x(x-1) = 1$

$x_{n+1} = 1 + \frac{1}{x_n}$ $\quad$ $x = \frac{1}{x-1}$

$x_{n+1} = \frac{1}{x_n - 1}$

$$\left\{\begin{array}{l} x^2 = x + 1 \\ x = \mp\sqrt{x+1} \\ x_{n+1} = \mp\sqrt{x_n + 1} \end{array}\right.$$

لا يؤخذ هذا الاحتمال

Because there are two answers for x_{n+1}.

يبقى احتمالان فقط

$$x_{n+1} = 1 + \frac{1}{x_n}$$

Pick $x_1 = 2$

$$x_2 = 1 + \frac{1}{2} = 1.5$$

$$x_3 = 1 + \frac{1}{1.5} = 1.666$$

$$x_4 = 1 + \frac{1}{1.666} = 1.6$$

$$x_5 = 1 + \frac{1}{1.6} = 1.625$$

$$x_6 = 1 + \frac{1}{1.625} = 1.6125$$

Converging to 1.168

$$x_{n+1} = \frac{1}{x_n - 1}$$

Pick $x_1 = 1.6$

$$x_2 = \frac{1}{1.6 - 1} = 1.666$$

$$x_3 = \frac{1}{1.666 - 1} = 1.5$$

$$x_4 = \frac{1}{1.5 - 1} \quad 2$$

$$x_5 = \frac{1}{2 - 1} = 1$$

$$x_6 = \frac{1}{1 - 1} \text{ not define}$$

Not converging

When does it converge?

fixed point → $x_{n+1} = g(x)$

$f(\text{root}) = 0$

$$x_{n+1} - \text{root} = g(x_n) - g(\text{root})$$

Expand $g(x)$ using Taylor series

⋮

$$|\text{error}_{n+1}| \leq |g'(\phi)|\,|\text{error}_n|$$

If $|g'(\text{root})| < 1 \rightarrow$ Converges

المحاضرة التاسعة

System of Equation

A system of linear equations

$a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n=b_1$

$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n=b_2$

…

$a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n=b_m$

can be represented as the matrix equation ***A*x =b***, where *A* is the coefficient matrix, and *b* is the vector containing the right sides of equations.

If you do not have the system of linear equations in the form AX = B, use equationsToMatrix to convert the equations into this form. Consider the following system.

1. Solve System of Linear Equations Using linsolve

2x + y + z = 2

-x + y - z = 3

x + 2*y + 3*z = -10

```
 syms x y z
 eqn1 = 2*x + y + z == 2;
 eqn2 = -x + y - z == 3;
 eqn3 = x + 2*y + 3*z == -10;
[A,B] = equationsToMatrix([eqn1, eqn2, eqn3], [x, y, z])
 X = linsolve(A,B)
```

Example:

solve the following problem, symbolically, in the easiest way.

3x + y = 5

2x +3y = 7

```
linsolve([3,1;2,3],[5;7])
```

ans= 1.14285

1.57142

2. Solve System of Linear Equations Using solve

Use solve instead of linsolve if you have the equations in the form of expressions and not a matrix of coefficients.

Consider the same system of linear equations.

$$2x + y + z = 2$$

$$-x + y - z = 3$$

$$X + 2y + 3z = -10$$

Declare the system of equations.

```
syms x y z
eqn1 = 2*x + y + z == 2;
 eqn2 = -x + y - z == 3;
eqn3 = x + 2*y + 3*z == -10;
```

Solve the system of equations using solve. The inputs to solve are a vector of equations, and a vector of variables to solve the equations for.

```
sol = solve([eqn1, eqn2, eqn3], [x, y, z]);
xSol = sol.x
ySol = sol.y
 zSol = sol.z
```

Answer : xSol = 3; ySol = 1; zSol = -5

المحاضرة العاشرة

Array and Matrix

Assign the content of an array/matrix;

Basic operations Ex. 28 Assign the content of a (one-dimensional) array; Addition of two arrays

```
a = [2 12 25];
b = [3 7 4];
c = a+b
```

Output: c = 5 19 29

Ex. Assign the content of a matrix;

Addition of two matrices

```
a = [3 4; 1 6];
b = [5 2; 11 7];
c = a+b
```

Output: c = 8 6 12 13

This program performs the following acts:

a = [3 4 1 6]

b = [5 2 11 7]

c = a ⍰ b = [8 6 12 13]

Ex. Multiplication involving a scalar and an array (or a matrix)

a = [3 5; 1 4];

b = 2*a

Output:

b = 6 10 2 8

Ex. Element-by-element multiplication involving two 1-D arrays or two matrices of the same dimension

a = [2 3 5]; b = [2 4 9]; c = a.*b Output: c = 4 12 45

c = [a(1)*b(1) a(2)*b(2) a(3)*b(3)]

Ex. Element-by-element multiplication of two matrices

a = [2 3; 1 4];

b = [5 1; 7 2];

c = a.*b

Output: c = 10 3 7 8

Ex. Direct (not element-by-element) multiplication of two matrices

a = [2 3; 1 4];

b = [5 1; 7 2];

c = a*b

Output: c = 31 8 33 9

Ex. Elementary functions with a vectorial variable

a = [2 3 5];
b = sin(a)

Output: b = 0.9092 0.1411 -0.9589 Remark: The content of b is [sin(2) sin(3) sin(5)].

Ex. Another example of elementary functions with a vectorial variable

a = [2 3 5];
b = 2*a.^2+3*a+4
Output: b = 18 31 69