Computer **A**ided **D**esign

MATLAB Introduction for Civil Engineers

1/11/2020

Wael Darwich

Damascus University

Content

- Introduction
- Memory and variables
- Operators, scripts and functions
- Matrices calculator
- Logic operators and decisions
- Loops
- Plots

Example

Solve by end of this lecture?

- حساب كلفة فاتورة الكهرباء
- اختيار التعرفة الأفضل حسب الاستهلاك
  - سعر ثابت 100 ليرة للكيلو الواط الساعي
  - تعرفة متغيرة حسب الاستهلاك

التعـرفـــة:

من ١ إلى ٦٠٠ بسعر ٢ ل.س، من ٦٠١ إلى ١٠٠٠ بسعر ٦ ل.س، من ١٠٠١ إلى ١٥٠٠ بسعر ٢٠ ل.س، من ١٥٠١ إلى ٢٥٠٠ بسعر ٩٠ ل.س، من ٢٥٠١ فما فوق بسعر ١٥٠ ل.س لكل ك.وس

عزيزي المشترك:

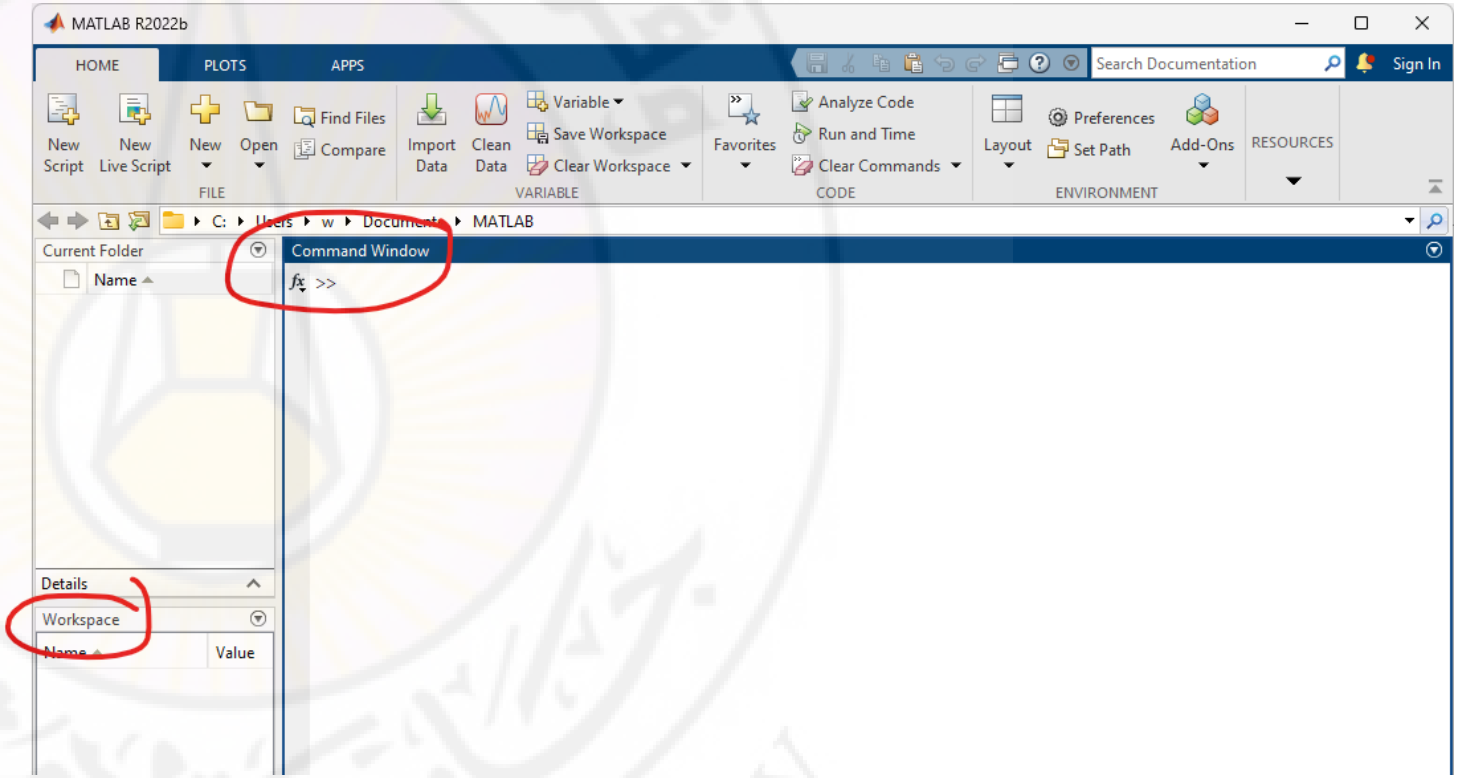
عزيزي المشترك يرجى عدم احضار اي مبلغ فوق المليون ليرة سورية وتسديد قيمة الفاتورة عن طريق(نافذة الدفع الالكتروني--او عن طرق شيك من اي مصرف مصدق اصولا او عن طريق المصرف التجاري في حساب الشركة العامة لكهرباء محافظة دمشق رقم الحساب/ ١٠٥١٠١٤٣٩٠٠١ / ومراجعتنا بعد التسديد للحصول على الفاتورة مباشرة
نرجو المبادرة الى تسديد الفواتير المترتبة عليكم تجنبا من سحب العداد في حال التاخر لمدة اربعة اشهر تطبيقا لنظام الاستثمار الجديد .

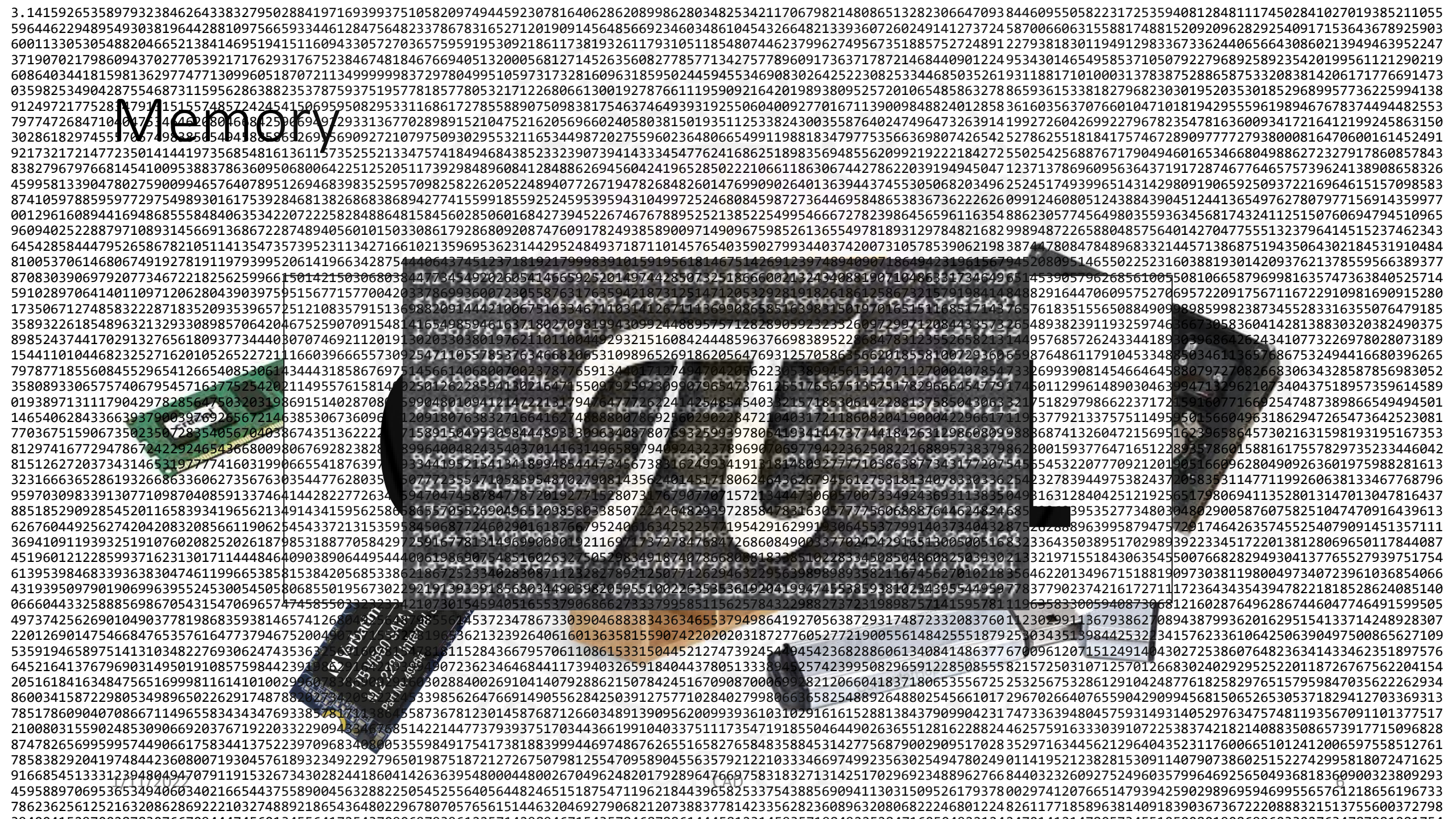
Programming Languages

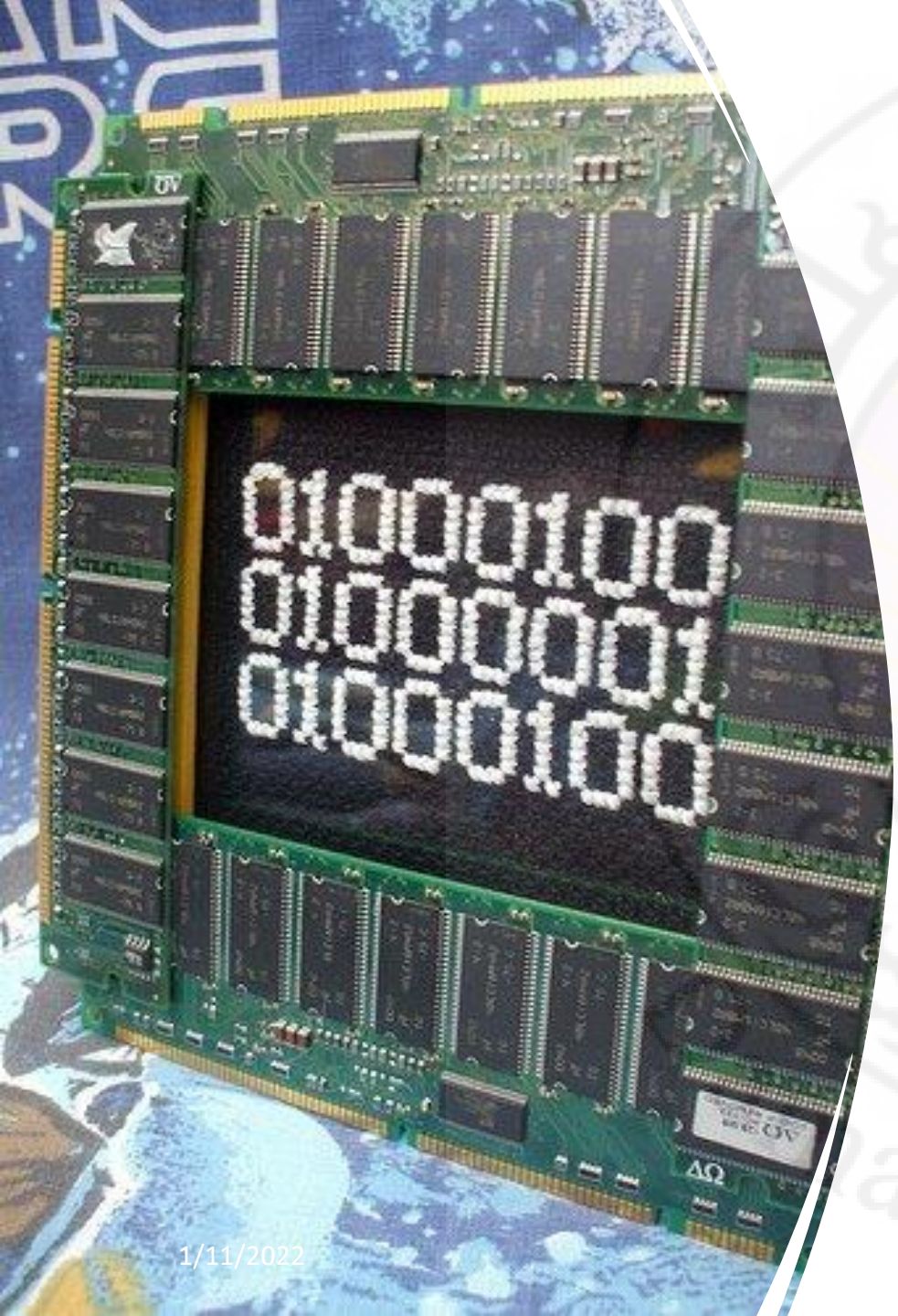
- `Fortran` Formula Translation, 1957
- `BASIC` Beginner's All-purpose Symbolic Instruction Code, 1964
- `C/C++` 1972/1985
- `MS Visual Basic` 1991
- `Python` 1991
- `Java` 1995

MATLAB

- Matrix Laboratory, 1970s
- Developed by MathWorks
- Matrix manipulations
- Algorithms
- Graph plotting
- Script (could compile)
- Numerical and symbolic engine




Memory



Bit vs Byte

- Bit or binary digit 0/1
- 2 Bit 0/1 0/1
- 3 Bit 0/1 0/1 0/1

2^3 values = 8

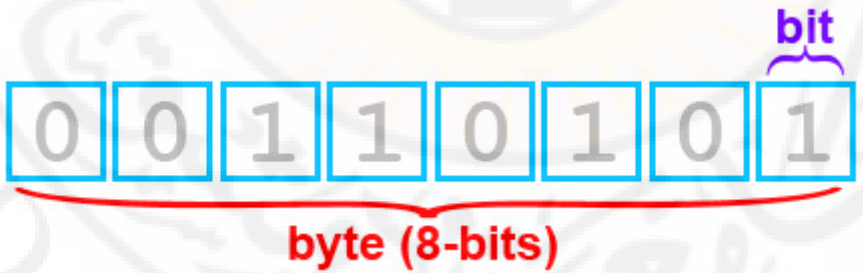
0:	**0**	**0**	**0**
1:	**0**	**0**	**1**
2:	**0**	**1**	**0**
3:	**0**	**1**	**1**
4:	**1**	**0**	**0**
5:	**1**	**0**	**1**
6:	**1**	**1**	**0**
7:	**1**	**1**	**1**

Memory

- Input ‘Hello’ to memory
- Hello = H + e + l + l + o

ASCII

- The **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$
- 255 character

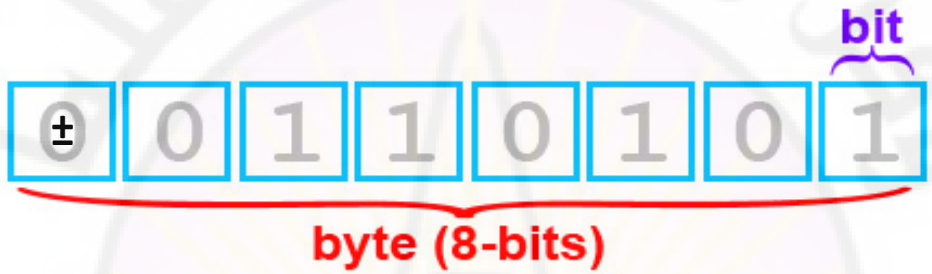


DEC	Symbol	DEC	Symbol	DEC	Symbol	DEC	Symbol	DEC	Symbol	DEC	Symbol	DEC	Symbol	DEC	Symbol
0	NUL	32		64	@	96	`	128	€	160		192	À	224	à
1	SOH	33	!	65	A	97	a	129		161	¡	193	Á	225	á
2	STX	34	"	66	B	98	b	130	‚	162	¢	194	Â	226	â
3	ETX	35	#	67	C	99	c	131	ƒ	163	£	195	Ã	227	ã
4	EOT	36	$	68	D	100	d	132	„	164	¤	196	Ä	228	ä
5	ENQ	37	%	69	E	101	e	133	…	165	¥	197	Å	229	å
6	ACK	38	&	70	F	102	f	134	†	166	¦	198	Æ	230	æ
7	BEL	39	'	71	G	103	g	135	‡	167	§	199	Ç	231	ç
8	BS	40	(	72	H	104	h	136	ˆ	168	¨	200	È	232	è
9	HT	41	)	73	I	105	i	137	‰	169	©	201	É	233	é
10	LF	42	*	74	J	106	j	138	Š	170	ª	202	Ê	234	ê
11	VT	43	+	75	K	107	k	139	‹	171	«	203	Ë	235	ë
12	FF	44	,	76	L	108	l	140	Œ	172	¬	204	Ì	236	ì
13	CR	45	-	77	M	109	m	141		173	-	205	Í	237	í
14	SO	46	.	78	N	110	n	142	Ž	174	®	206	Î	238	î
15	SI	47	/	79	O	111	o	143		175	¯	207	Ï	239	ï
16	DLE	48	0	80	P	112	p	144		176	°	208	Ð	240	ð
17	DC1	49	1	81	Q	113	q	145	‘	177	±	209	Ñ	241	ñ
18	DC2	50	2	82	R	114	r	146	’	178	²	210	Ò	242	ò
19	DC3	51	3	83	S	115	s	147	“	179	³	211	Ó	243	ó
20	DC4	52	4	84	T	116	t	148	”	180	´	212	Ô	244	ô
21	NAK	53	5	85	U	117	u	149	•	181	µ	213	Õ	245	õ
22	SYN	54	6	86	V	118	v	150	–	182	¶	214	Ö	246	ö
23	ETB	55	7	87	W	119	w	151	—	183	·	215	×	247	÷
24	CAN	56	8	88	X	120	x	152	˜	184	¸	216	Ø	248	ø
25	EM	57	9	89	Y	121	y	153	™	185	¹	217	Ù	249	ù
26	SUB	58	:	90	Z	122	z	154	š	186	º	218	Ú	250	ú
27	ESC	59	;	91	[	123	{	155	›	187	»	219	Û	251	û
28	FS	60	<	92	\	124	|	156	œ	188	¼	220	Ü	252	ü
29	GS	61	=	93	]	125	}	157		189	½	221	Ý	253	ý
30	RS	62	>	94	^	126	~	158	ž	190	¾	222	Þ	254	þ
31	US	63	?	95	_	127		159	Ÿ	191	¿	223	ß	255	ÿ

Memory

- Input ‘Hello’ to memory
- Hello = H + e + l + l + o
- 01001000 01100101 01101100 01101100 01101111

Numbers in Memory



- 1 Byte variable
  - Unsigned Integer 0 ~ 255 (2^8=256)

    1111 1111 Bin = 255 Dec
  - Signed integer -128 ~ 127 (2^7=128)

    111 1111 Bin = 127 Dec

Numbers in Memory



- 2 Byte integer (short)
  - Unsigned Integer 0 ~ 65535 (2^{16}=65536)
  - Signed Integer –32768 ~ 32767 (2^{15}=32768)

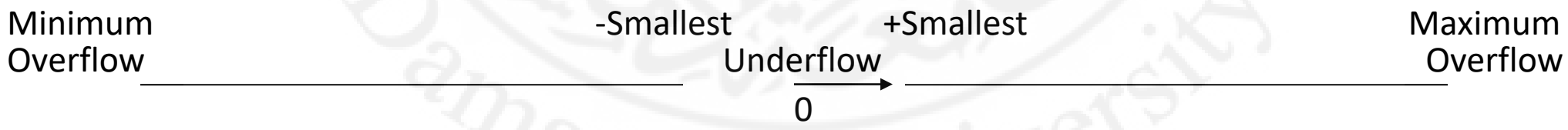
Numbers in Memory

8 Byte double (default in MATLAB)

- Accuracy 16 digits:

```
A = 1.123456789012345
B = 0.0000000000000001   (1E-16)
C = A + B
D = C - A
D = ?
```

- Maximum/Minimum ±1.8 E+308
- Smallest ±2.2 E−308

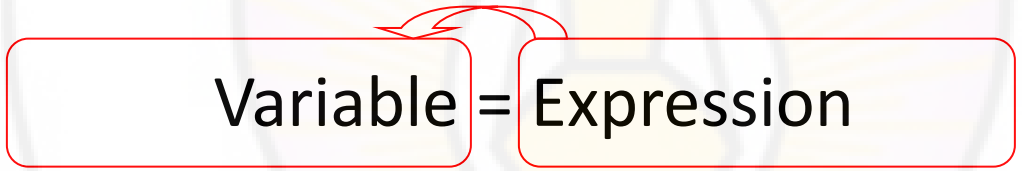


Variables

- Reserved locations in memory
- Names:
  - Including a-z, A-z, 0-9, _ (underscore)
  - Starting a-z, A-z
  - No space `section_radius`
  - Case sensitive
  - CamelCase: `SectionRadius`

Operators

- Assignments operator =
- `MyVariable = 123`



- `MyVariable = MyVariable + 1`
- `MyVariable + 1 = 2`
- Assignments vs. Equal

Arithmetic Operators

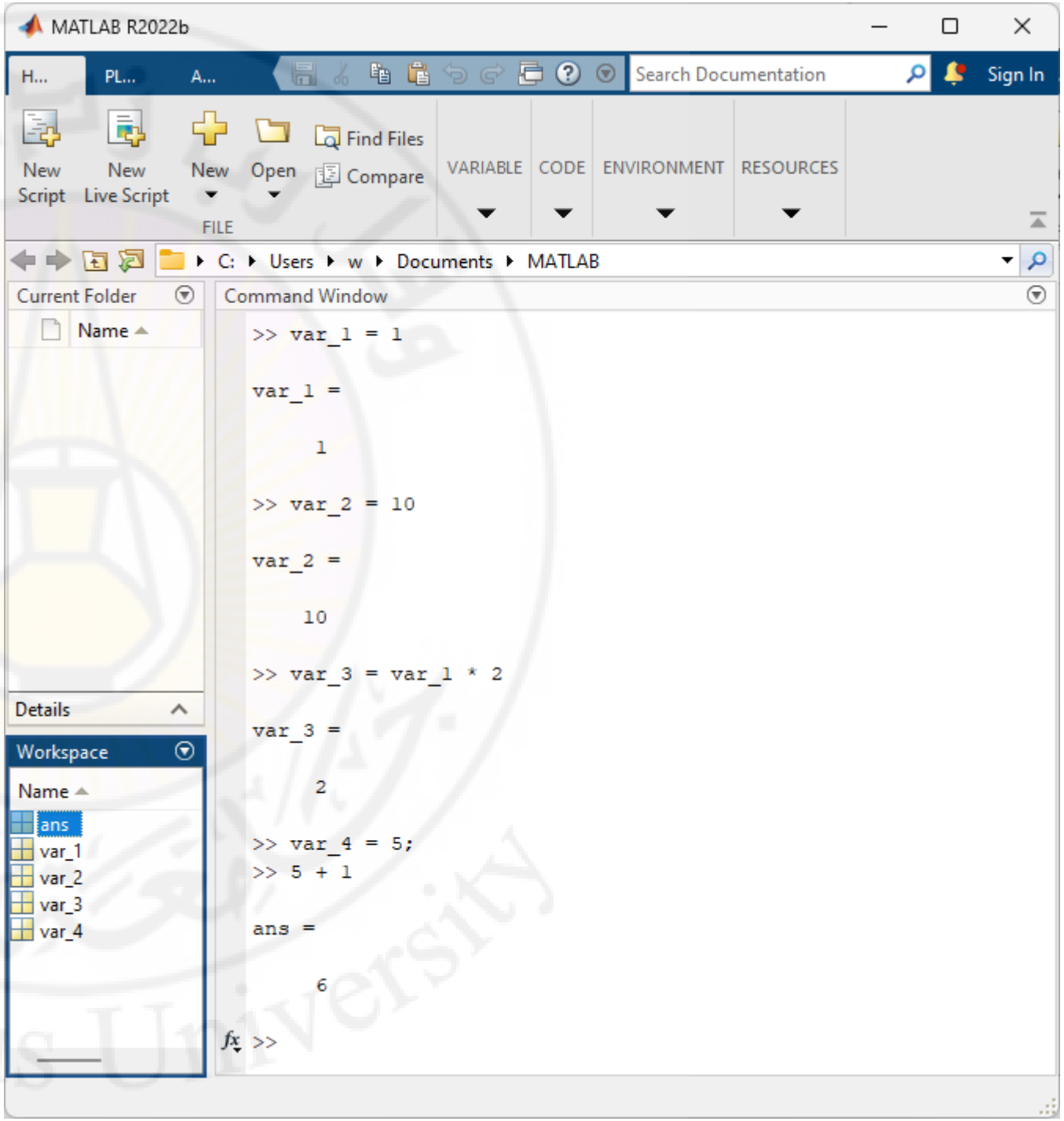
- + − × / ^
- Right division (forward slash, slash /)
- Left division (backward slash, backslash \)
  - a . x = b
  - x = a \ b
  - b / a = b . a^{-1}
  - a \ b = a^{-1}. b

Advanced calculator

- Simple calculations
- `sqrt, exp, log, log10`
- String
- Complex
- Comma
- Semicolon
- Save mat

Special variables

- `ans`
- `pi`
- `inf`
- `NaN`
- `i (or j)`
- `realmin = 2.2251e-308`
- `Realmax = 1.7977e+308`
- `intmax = 2147483647`



Built-in functions

• `abs(x)`	Absolute value
• `sqrt(x)`	Square root
• `round(x)`	Rounds towards nearest decimal or integer
• `fix(x)`	Truncates toward 0
• `floor(x)`	Rounds towards minus infinity
• `ceil(x)`	Rounds towards plus infinity
• `sign(x)`	Signum function
• `rem(x,y)`	Remainder after division
• `exp(x)`	Exponential
• `log(x)`	Natural logarithm
• `log10(x)`	Common (base 10) logarithm
• `log2(x)`	Base 2 logarithm

Trigonometry

- `sin, cos, tan, asin…` input angles in radians
- 1 degree = pi/180 radians
- `angle_radians = angle_degrees * (pi/180)`
- `angle_degrees = angle_radians * (180/pi)`
- `rad2deg, deg2rad`
- `sind, cosd`

Numerical precision

- `sin (0)`
- `sin (pi)`
- `sin (2 * pi)`

MATLAB

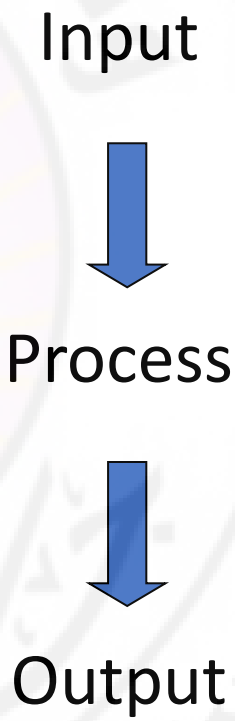
- `who` List current variables
- `whos` List current variables, long form
- `clear` Clear variables and functions from memory
- `clc` Clear command window
- `help` Help for functions in Command Window
- `doc` Reference page in Help browser
- `lookfor` Search for keyword

Scripts

- M file
- Comments
- Display
- Input

Functions

- Set of steps
- Input argument(s) (optional)
- [Output argument(s)] (optional)



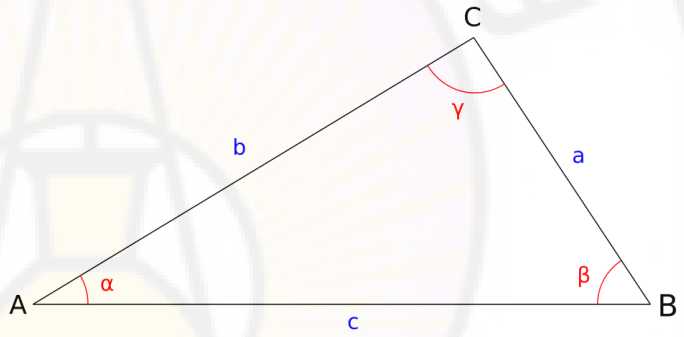
Function syntax

```
function [output1 output2] = func(input1, input2)
% Comment
code
```

- Function name, same as .m file name.
- Current directory
- Comments for help

Function example

$$c^2 = a^2 + b^2 - 2 \times a \times b \times \cos(\gamma)$$



- Define inputs
- Calculate output

Matrices

- Every variable is a matrix
- Vectors
- linspace
- Transpose

Computer **A**ided **D**esign

MATLAB Introduction for Civil Engineers 2

8/11/2020

Wael Darwich

Last Week

- Introduction to MATLAB
- Memory
- Assignment and Arithmetic Operators
- Built-in Functions and Variables
- Scripts and Functions
- Matrices

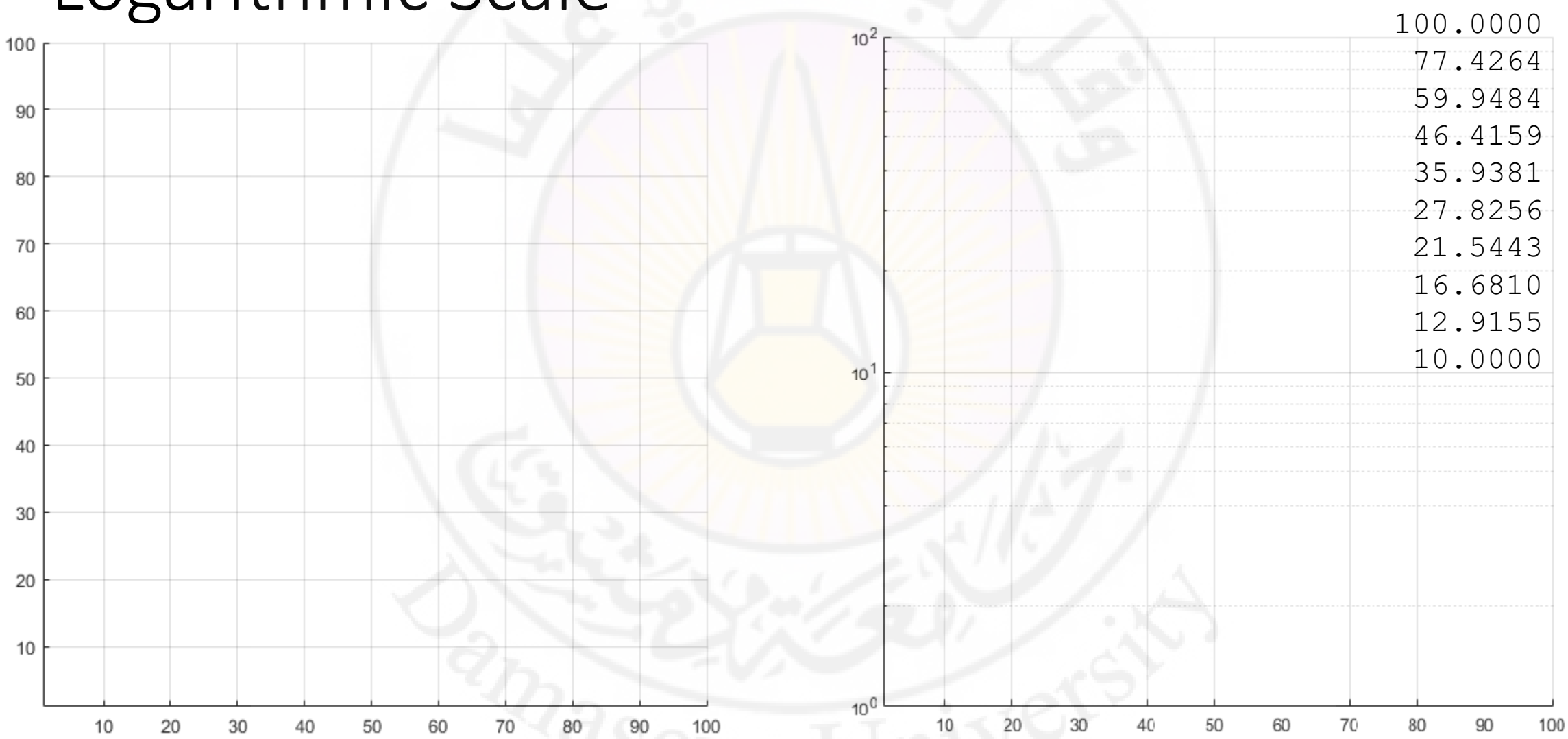
Content

- Matrices calculator
- Logic Operators and Decisions
- Example
- Loops
- Figures

Matrices

- Every variable is a matrix
- Vectors
- Transpose
- Index
- Colon: `start : step_size : end`
- `linspace` Linearly spaced vector
- `linspace(X1, X2, N)` Generates N points between X1 and X2
- `logspace` Logarithmically spaced vector

Logarithmic Scale



Matrices

- `disp`
- `num2str`
- `format`
- `sqrt, factorial, sum, prod, max, min, mean`
- `length, size`
- `zeros, ones, eye`

Matrices

```
A=[1 2 3; 4 5 6; 7 8 9]
```

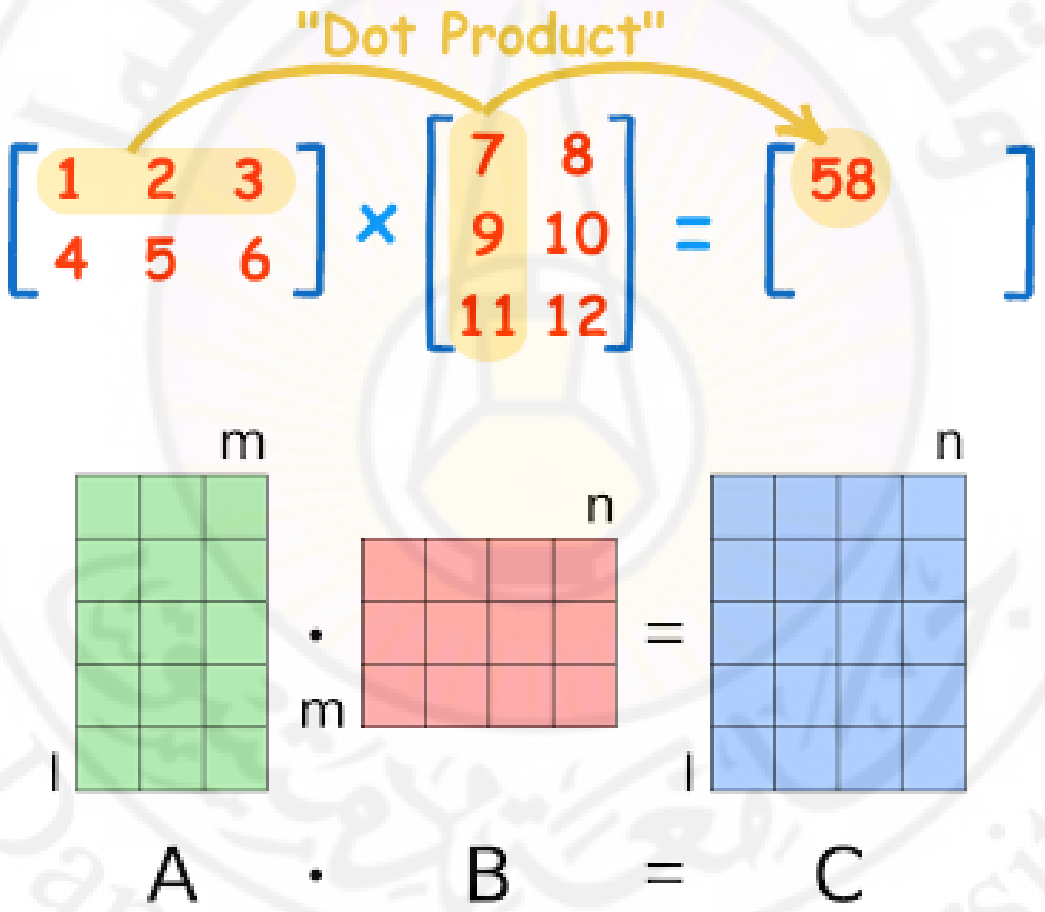
```
A =
     1     2     3
     4     5     6
     7     8     9
```

```
B = A(1,1)
C = A(2,1:3) = A(2,:)
D = A(3,[2 3]) = A(3,[2:3]) = A(3,2:3)
A(1, 1) = 4
```

Matrices

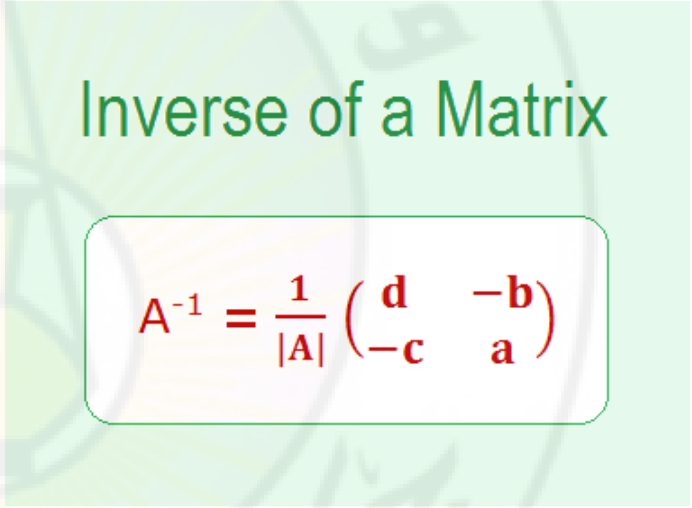
- `sum` Sum of elements (by columns)
- `mean` Average or mean value (by columns)
- `diag` Diagonal of a matrix
- `triu` Extract upper triangular part
- `tril` Extract lower triangular part
- `size` Size of array

Matrices



Matrices

- A = $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
- Determinant = |A| = ad - bc
- Inverse
- `A * inv(A)`
- `A * A, A^2`
- `A .* A, A .^2`



Linear Equations

$$
\begin{aligned}
3.x + 2.y - z &= 10 \\
-x + 3.y + 2.z &= 5 \\
x - y - z &= -1
\end{aligned}
$$

$$
\begin{bmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} =
\begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}
$$

$$
A \,.X = B
$$

$$
A^{-1}.A \,.X = A^{-1}.B
$$

$$
X = A^{-1}.B = A \setminus B
$$

Symbolic Math Toolbox

$$\begin{aligned} 3.x + 2.y - z &= 10 \\ -x + 3.y + 2.z &= 5 \\ x - y - z &= -1 \end{aligned}$$

- `syms` Short-cut for constructing symbolic variables (not numeric)

```
syms x y z
eq1 = 3*x +2*y - z == 10;
eq2 = -x +3*y +2*z == 5;
eq3 = x - y - z    == -1;
solve([eq1, eq2, eq3], [x, y, z])
```

MATLAB

- Matrix calculator
- Symbolic toolbox
- Scripting

Relational Operators

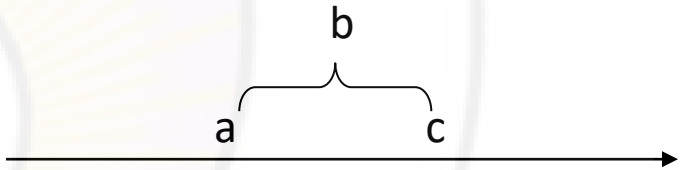
• Equal	==	(not equal)
• Unequal	~=	(VB <>)
• Smaller	<	
• Greater or equal	>=	
• And	&	
• Or	|	
• Not	~	

Relational Operators

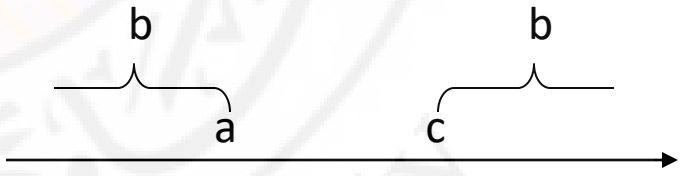
- `false  =  0`
- `true  =  1`
- Any none zero value is true

Relational Operators

- Not: ~
  - ~False = True ; ~0 = 1
  - ~True = False ; ~1 = 0; ~2 = 0
- And: b > a & b < c
  - True & True = True
  - False & False = False
  - True & False = False & True = False



- Or: b < a | b > c
  - True | True = True
  - False | False = False
  - True | False = False | True = True



Precedence of Operators

1.	Brackets	( )
2.	Power	^ .^
3.	Unary Plus/Minus, Not	+ − ~
4.	Multiple and divide	* / \ .* ./ .\
5.	Addition Subtraction	+ −
6.	Colon	:
7.	Compare	< > ~=
8.	And	&
9.	Or	|

Example

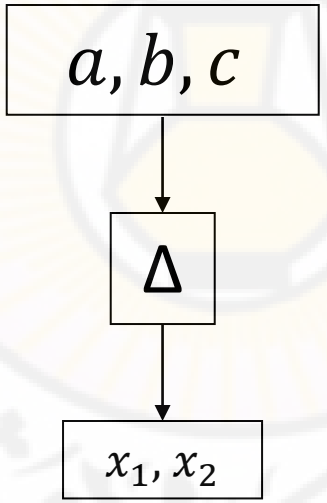
- Quadratic (second order) equations:

  $A \cdot x^2 + B \cdot x + C = 0$

- Inputs:
  - A, B, C
- Outputs:
  - X_1, X_2

Quadratic Equations

$$a\,.x^2 + b\,.x + c = 0$$



Decisions

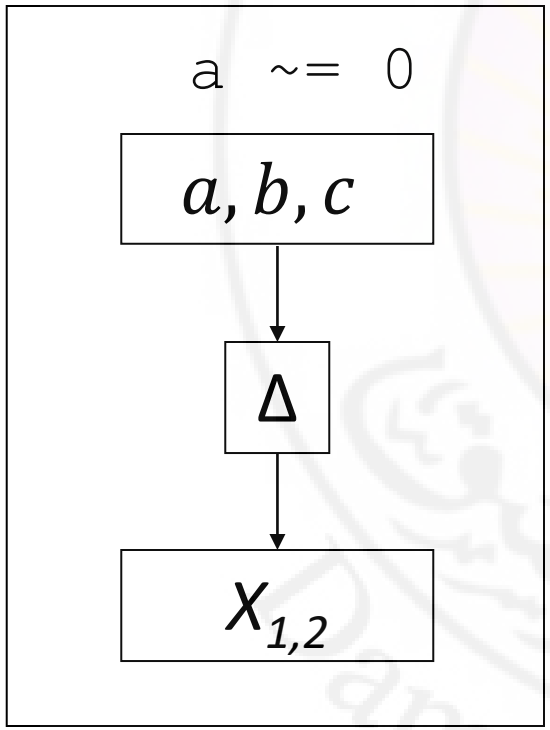
$$a \,.\, x^2 + b \,.\, x + c = 0$$

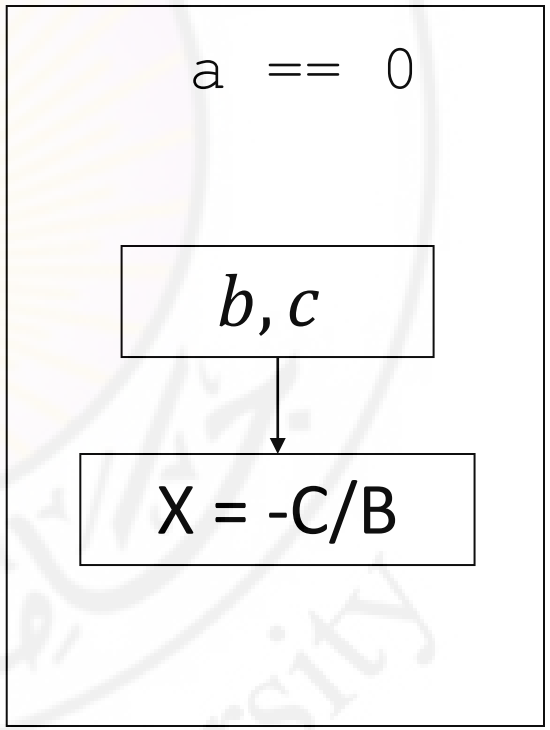
$$x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2 \,.\, a}$$

$$a = 0$$

Decisions

$$a\,.x^2 + b\,.x + c = 0$$





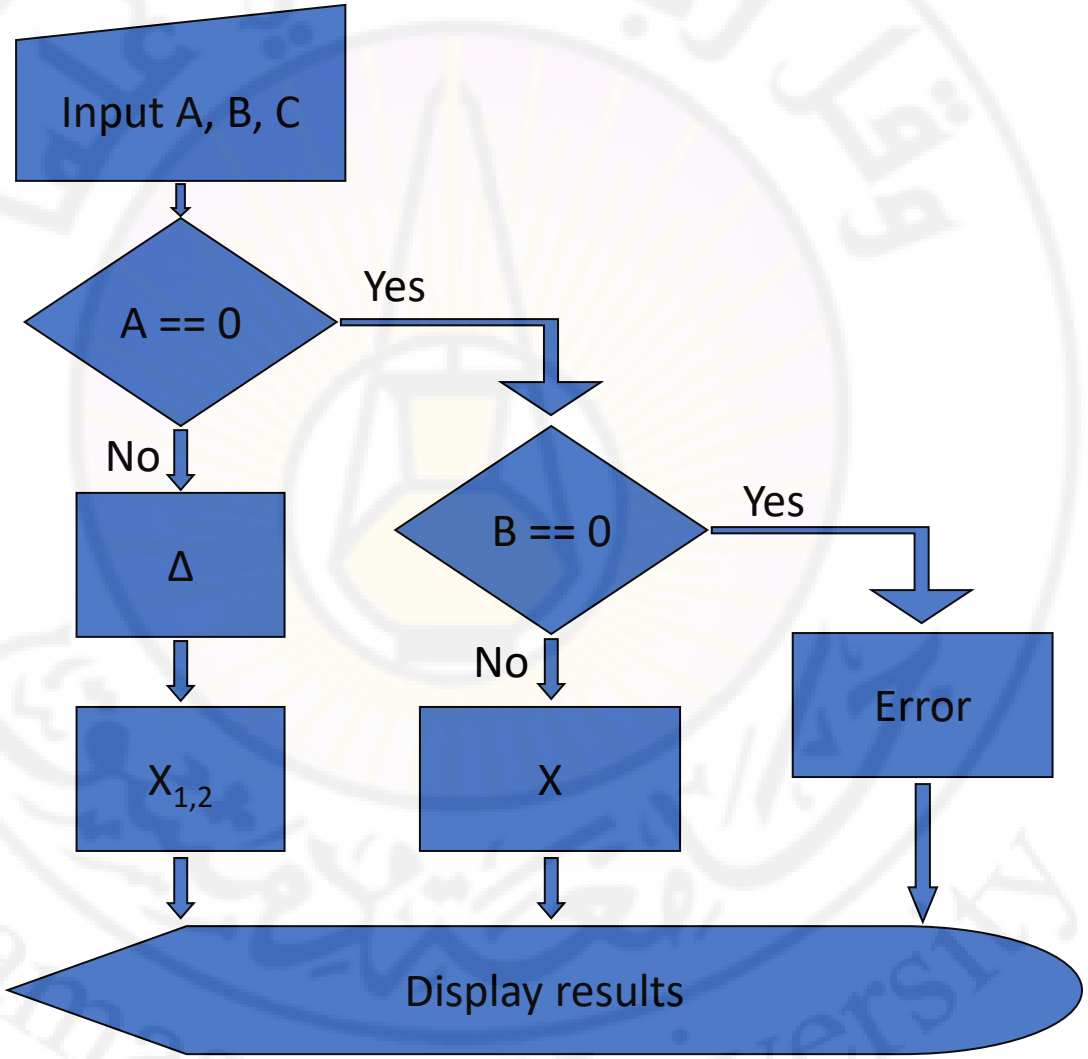
Error Check

$$a\,.x^2 + b\,.x + c = 0$$

```
a == 0
b == 0
```

No solution!

Flowchart



Decisions Syntax

```
if condition
   true
end
```

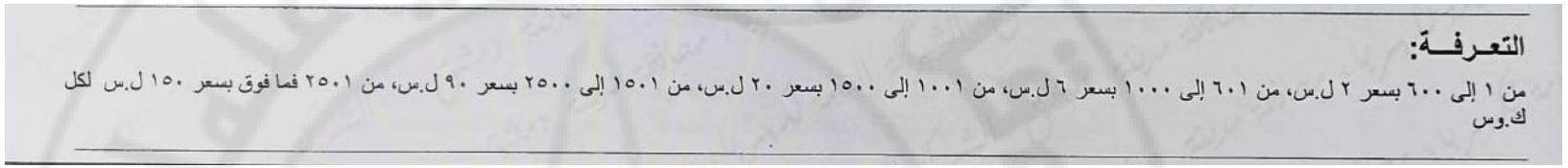
Decisions Syntax

```
if condition
   true
else
   false
end
```

Decisions Syntax

```
if condition1
   true1
elseif conditio2
   true2
else
   all false
end
```

Example


التعـرفـــة:

من ١ إلى ٦٠٠ بسعر ٢ ل.س، من ٦٠١ إلى ١٠٠٠ بسعر ٦ ل.س، من ١٠٠١ إلى ١٥٠٠ بسعر ٢٠ ل.س، من ١٥٠١ إلى ٢٥٠٠ بسعر ٩٠ ل.س، من ٢٥٠١ فما فوق بسعر ١٥٠ ل.س لكل ك.وس

```
function tarrif = ebill_ex(kwh)

if kwh < 1
  tarrif = 0;
elseif kwh <= 600
 tarrif = kwh * 2;
elseif kwh <= 1000
 tarrif = (kwh - 600) * 6 + 600 * 2;
elseif kwh <= 1500
 tarrif = (kwh - 1000) * 20 + 600 * 2 + 400 * 6;
elseif kwh <= 2500
 tarrif = (kwh - 1500) * 90 + 600 * 2 + 400 * 6 + 500 * 20;
else
 tarrif = (kwh - 2500) * 150 + 600 * 2 + 400 * 6 + 500 * 20 + 1000 * 90;
end
```

Second Order Equation Solver

```
function [x1, x2] = root_ex(a, b, c)
% root_ex(a, b, c) computes the roots of the
% second order equation a x^2 + b x + c = 0
```

Second Order Equation

```
if (a~=0)
    d2 = sqrt(b^2-4*a*c);
    x1 = (-b+d2)/(2*a);
    x2 = (-b-d2)/(2*a);
    disp('There are two roots for this equation.')
end
```

Second Order Equation

```
if (a~=0)
    d2 = delta2(a, b, c);
    x1 = (-b+d2)/(2*a);
    x2 = (-b-d2)/(2*a);
    disp('There are two roots for this equation.')
end

function d2 = delta2(a, b, c)
% Sub function to calculate delta square root
d2 = sqrt(b^2-4*a*c);
```

Second Order Equation

```
if (a==0)

end
```

Second Order Equation

```
if (a==0)
    if  (b==0)



    else



    end
end
```

Second Order Equation

```
if (a==0)
    if  (b==0)
        x1 = NaN;
        x2 = NaN;
        disp('This equation is degenerate!')
    else
        x1 = -c/b;
        x2 = x1;
        disp('There is a single root for this equation.')
    end
end
```

```
function [x1, x2] = root_ex(a, b, c)
% root_ex(a, b, c) computes the roots of the
% second order equation a x^2 + b x + c = 0
if (a==0)
    if  (b==0)
        x1 = NaN;
        x2 = NaN;
        disp('This equation is degenerate!')
    else
        x1 = -c/b;
        x2 = x1;
        disp('There is a single root for this equation.')
    end
else
    d2 = delta2(a, b, c);
    x1 = (-b+d2)/(2*a);
    x2 = (-b-d2)/(2*a);
    disp('There are two roots for this equation.')
end
function d2 = delta2(a, b, c)
% Sub function to calculate delta square root
d2 = sqrt(b^2-4*a*c);
```

Roots Solver

- `[x1, x2] = root_ex(1, 0, -4)`
- All inputs 0?
- MATLAB built in solver
- `roots([1 0 -4])` Find polynomial roots
- `roots(1:101)`
- Symbolic Math Toolbox
- `syms x`
- `solve(x^2 -4 == 0, x)`

Bottle Problem

- Half full / half empty?
- Calculate volume



Loops

- Repeat for a number of iterations
- Use condition (`while`) or counter (`for`)

```
counter = 1;
while counter <= 3
    disp(counter)
    counter = counter + 1;
end
```

```
for counter = 1:3
    disp(counter)
end
```

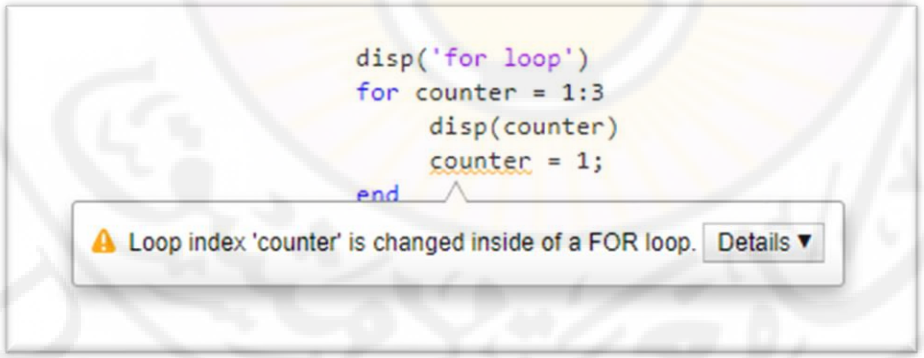
```
1
2
3
```

Endless Loops!

```
counter = 1;
while counter <= 3
    disp(counter)
end
```

```
for counter = 1:3
    disp(counter)
    counter = 1;
end
```

```
1
1
1
1
...
```




```
1
2
3
```

Endless Loops!

```
counter = 1;
while true
    disp(counter)

    counter = counter + 1;
    if counter > 3
        break
    end
end
```

Nested Loops

```
for x = 1:3
    for y = 1:3
        disp([x y])
    end
end
```

Outer

Inner

x: 3 iterations

y: 3 iterations

3×3 = 9 iterations

```
y = 1     1  1
y = 2     1  2        } x = 1
y = 3     1  3
y = 1     2  1
y = 2     2  2        } x = 2
y = 3     2  3
y = 1     3  1
y = 2     3  2        } x = 3
y = 3     3  3
```

Nested Loops

```
for x = 1:3
    for y = 4:5
        disp([x y])
    end
end
```

Outer

Inner

x: 3 iterations

y: 2 iterations

3×2 = 6 iterations

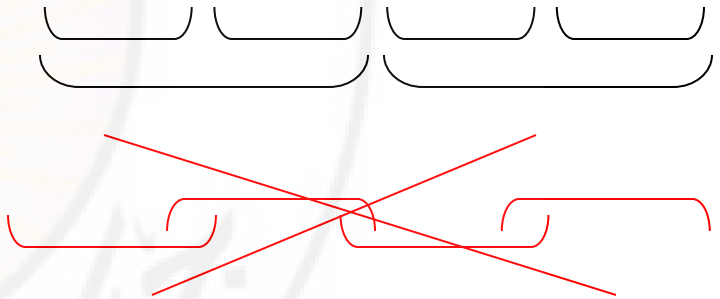
```
1  4
1  5
2  4
2  5
3  4
3  5
```

When working with nested loops, the outer loop changes only after the inner loop is completely finished

Ends

- If / End : Same number
- For, While / End : Same number
- (1+2×(3+4)) = (1+2×(3+4))

≠ (1+2×(3+4))



A bracket will always close the last opened one

Same for if/for/while & end

Nested Loops

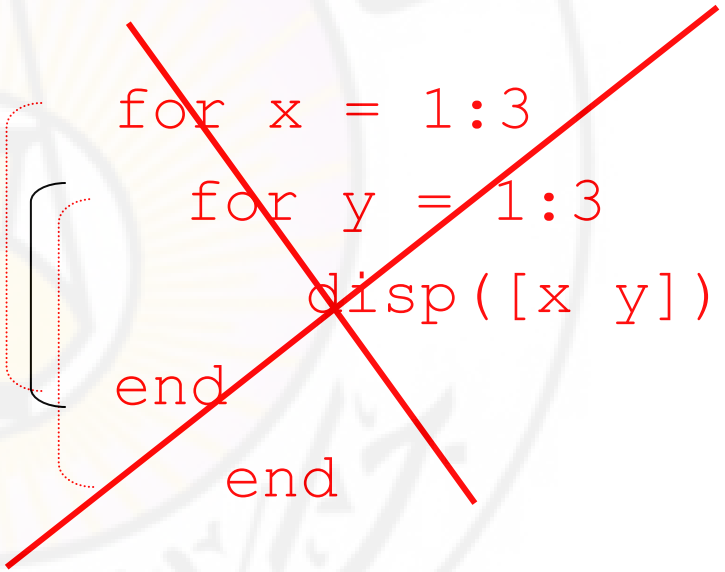
Nested loops

```
for x = 1:3
    for y = 1:3
        disp([x y])
    end
end
```

Allowed

Unnested loops

```
for x = 1:3
    for y = 1:3
        disp([x y])
end
    end
```



Allowed BUT not as it looks

Loops

- `for`
- `while`

For Syntax

```
for counter = start:step:end
    code
end
```

- Using counter
- Predefined number of steps

While Syntax

```
code1
while condition
    code2
end
```

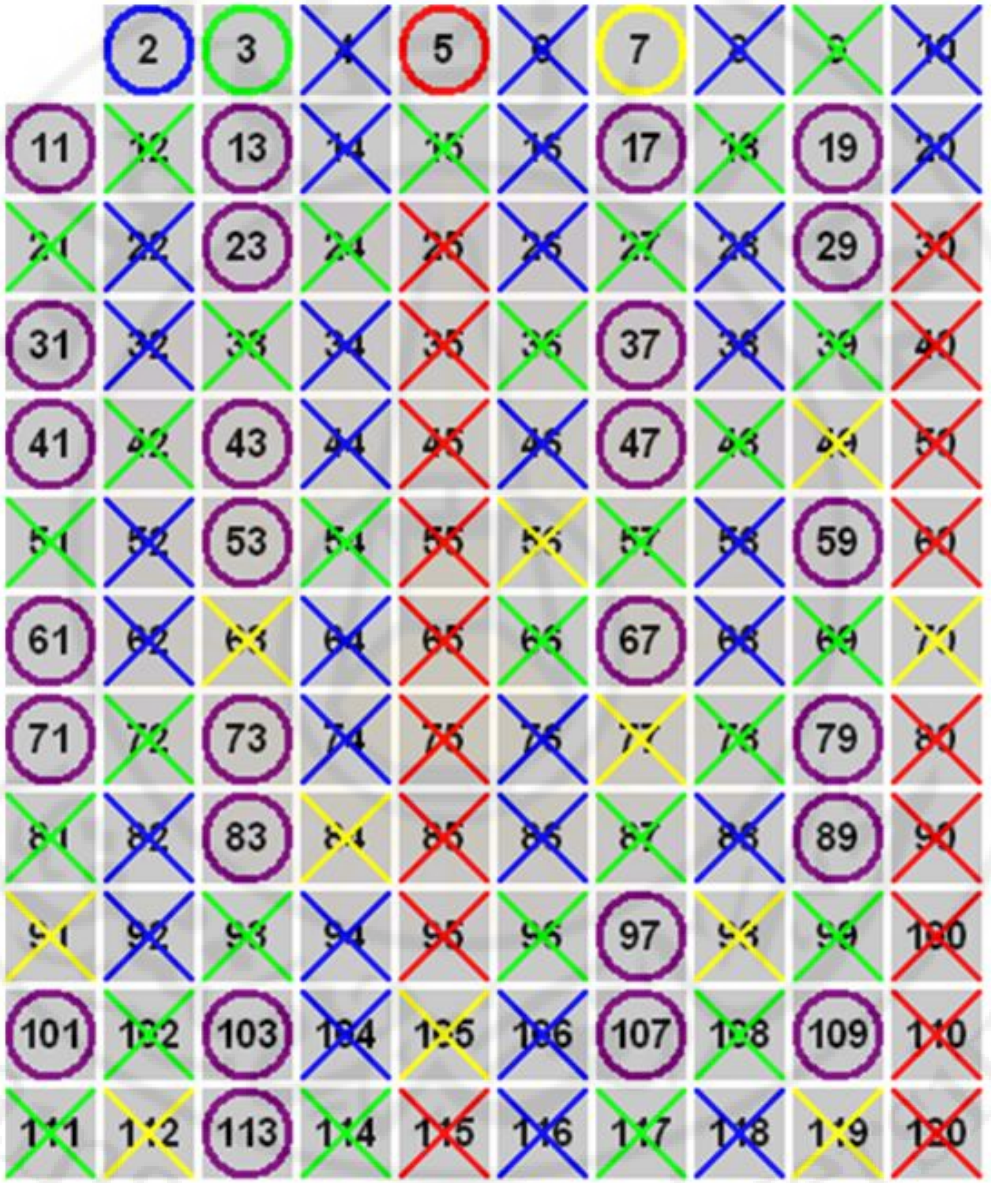
- No counter
- Undefined number of steps, could be 0
- `code1` : Needs to satisfy the condition to get in
- `code2` : Needs to break the condition to get out

Prime Numbers Example

To find all prime numbers less than or equal to integer n (Eratosthenes' method)

1. Create a list of consecutive integers from two to n: (2, 3, ..., n).
2. Initially, let p equal 2, the first prime number.
3. Strike from the list all multiples of p less than or equal to n. (2p, 3p, 4p, etc.)
4. Find the first number remaining on the list after p (this number is the next prime); replace p with this number.
5. Repeat steps 3 and 4 until p^2 is greater than n.
6. All the remaining numbers in the list are prime.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120



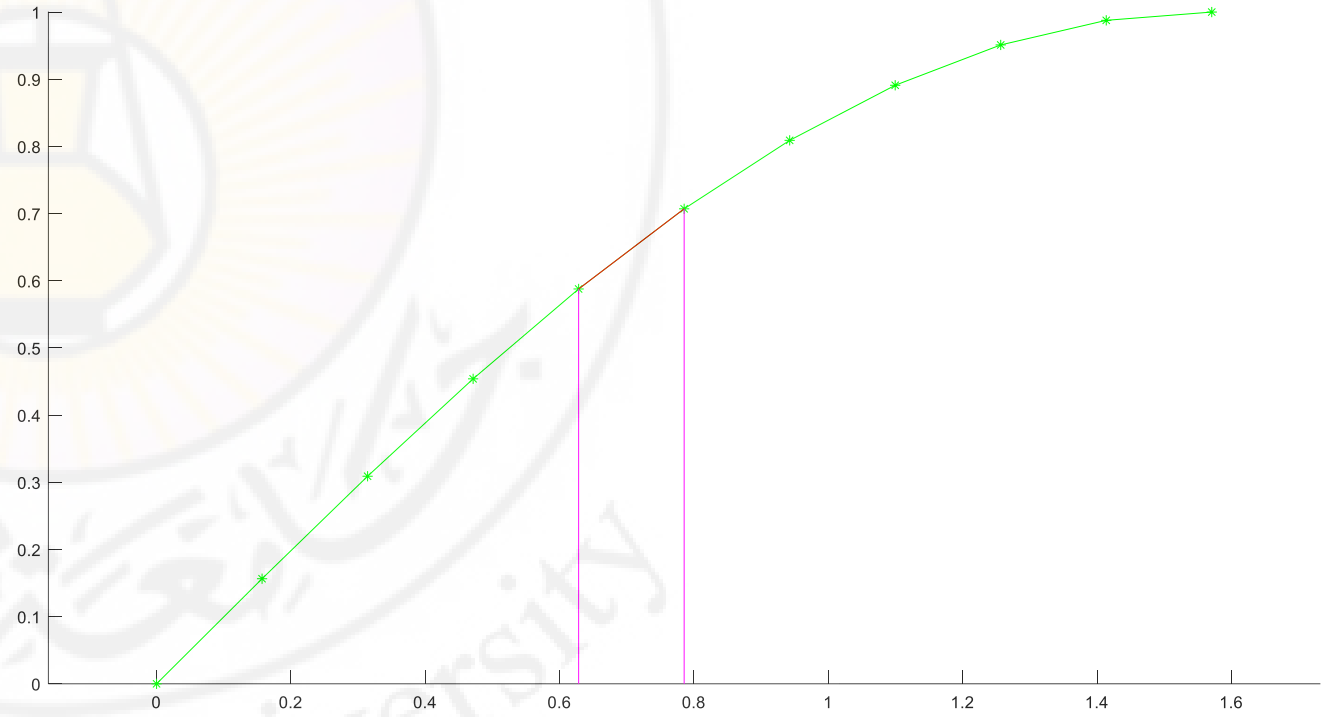
Prime numbers

2	3	5	7
11	13	17	19
23	29	31	37
41	43	47	53
59	61	67	71
73	79	83	89
97	101	103	107
109	113		

Numerical Integration

- Calculate area under function `y=sin(x)` in the range `[0 π/2]`

```
x1 = 0; x2 = pi/2;
range = x2 - x1;
steps = 10;
dx = range/steps;
x = x1 : dx : x2;
y = sin(x);
A = 0;
for i = 1:steps
    dA = dx * 0.5*(y(i)+y(i+1));
    A = A + dA;
end
disp(A)
```



Symbolic Math Toolbox

```
syms x
int(sin(x))                          -cos(x)

int(sin(x), 0, pi/2)                 1

int(sin(exp(x)), 0, pi/2) sinint(exp(pi/2)) - sinint(1)
eval(…)                              0.641721200446473
```

Other Examples

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$$= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

```
function y = sin_ex(x)
y = 0;
for n = 0 : 10
    y = y + (-1)^n * x^(2*n+1) / factorial(2*n+1);
end
```

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

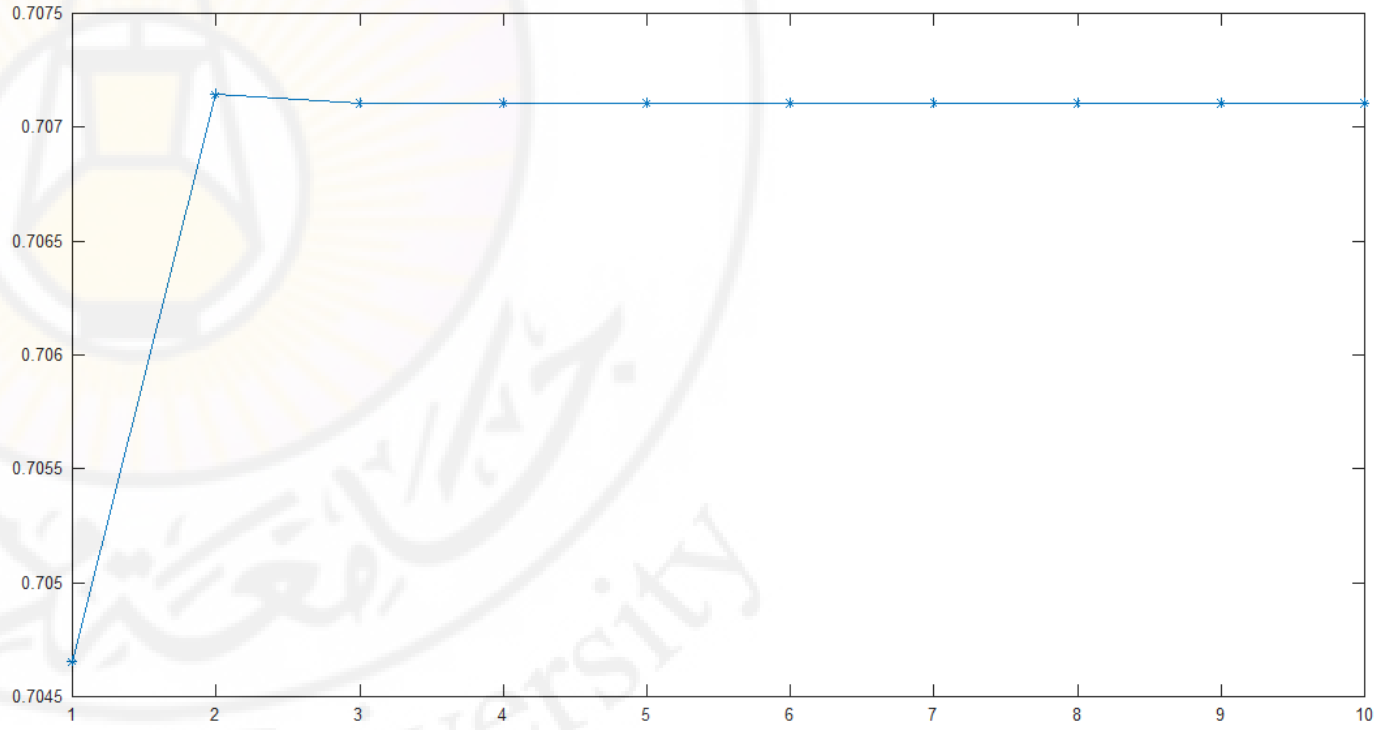
$$= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

```
function y = cos_ex(x)
y = 0;
for n = 0 : 10
    y = y + (-1)^n * x^(2*n) / factorial(2*n);
end
```

Number of Steps

- Engineers can determine the required number of steps
  - Accuracy
  - Efficiency

Max n	sin(45°)
1	0.704652651209168
2	0.707143045779360
3	0.707106469575178
4	0.707106782936867
5	0.707106781179619
6	0.707106781186568
7	0.707106781186547
8	0.707106781186547
9	0.707106781186547
10	0.707106781186547



Figures

- Next week…

Computer Aided Design

MATLAB Introduction for Civil Engineers 3

15/11/2020

Wael Darwich

Content

- Graphics
  - 2D plot
  - Functions
  - Examples, Beams and Trusses
  - 3D plot

Last Week

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$$= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

```
function y = sin_ex(x)
y = 0;
for n = 0 : 10
    y = y + (-1)^n * x^(2*n+1) / factorial(2*n+1);
end
```

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

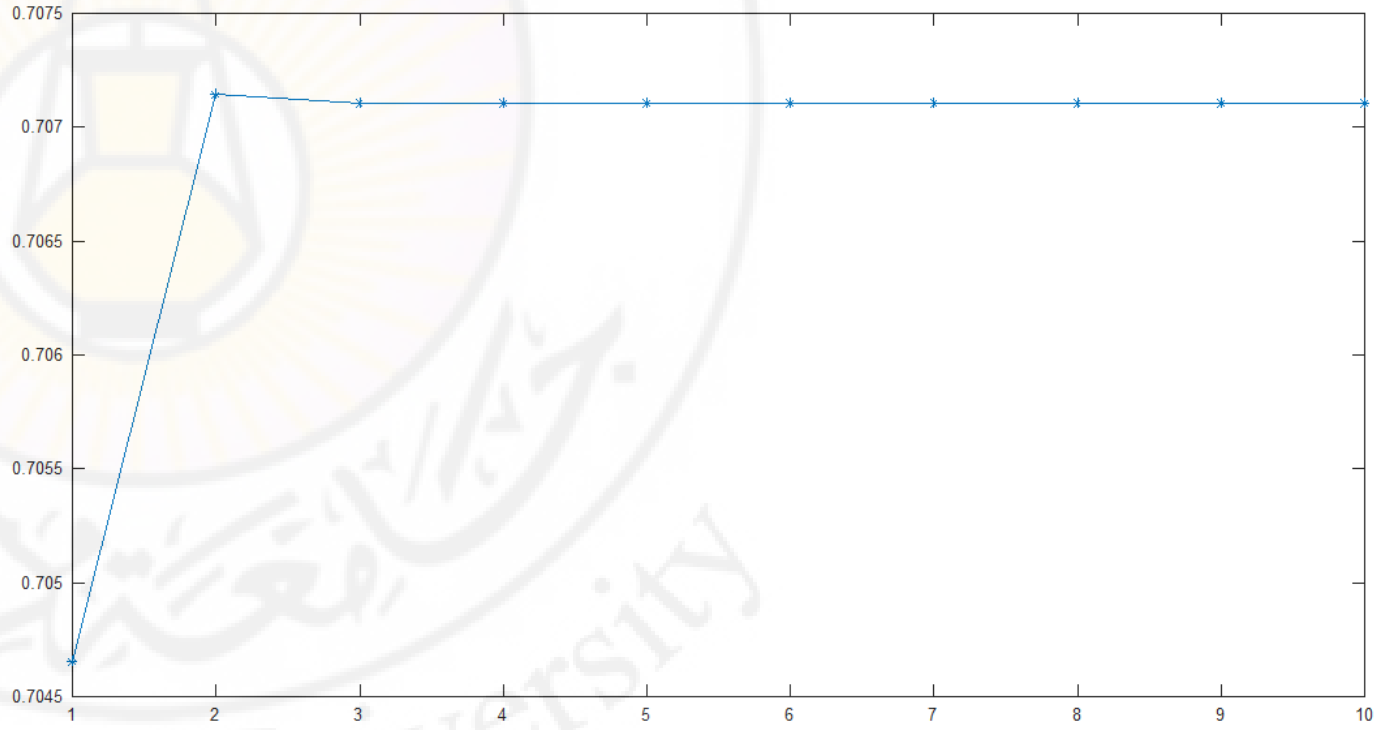
$$= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

```
function y = cos_ex(x)
y = 0;
for n = 0 : 10
    y = y + (-1)^n * x^(2*n) / factorial(2*n);
end
```

Number of Steps

- Engineers can determine the required number of steps
  - Accuracy
  - Efficiency

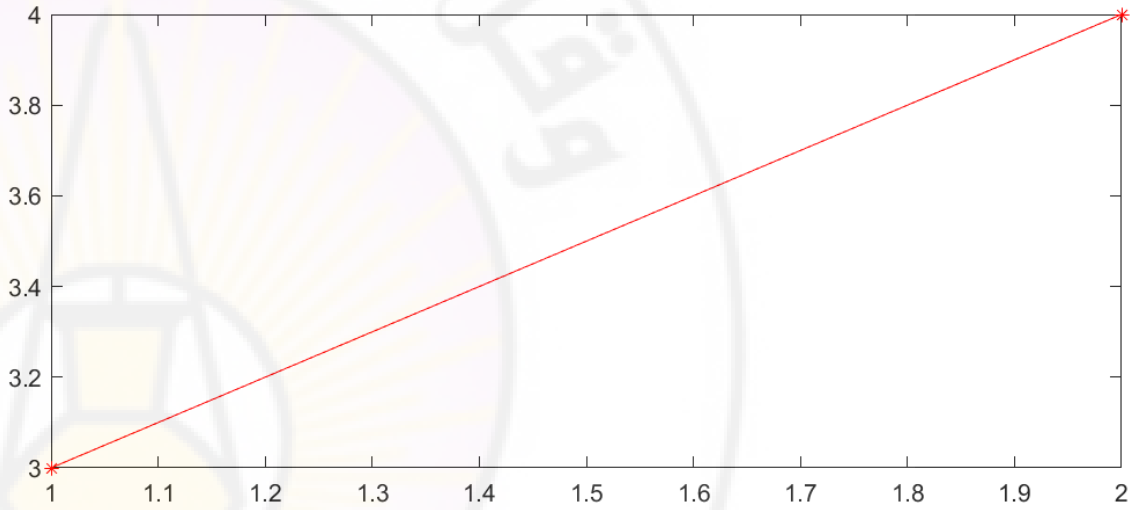
Max n	sin(45°)
1	0.704652651209168
2	0.707143045779360
3	0.707106469575178
4	0.707106782936867
5	0.707106781179619
6	0.707106781186568
7	0.707106781186547
8	0.707106781186547
9	0.707106781186547
10	0.707106781186547



Plot

```
x = [1 2];
y = [3 4];


plot(x, y)
plot(x, y, '-*r')
```



Plot

- `plot(X,Y)` plots vector Y versus vector X.
- `plot(X,Y,S)` where S is a character string:

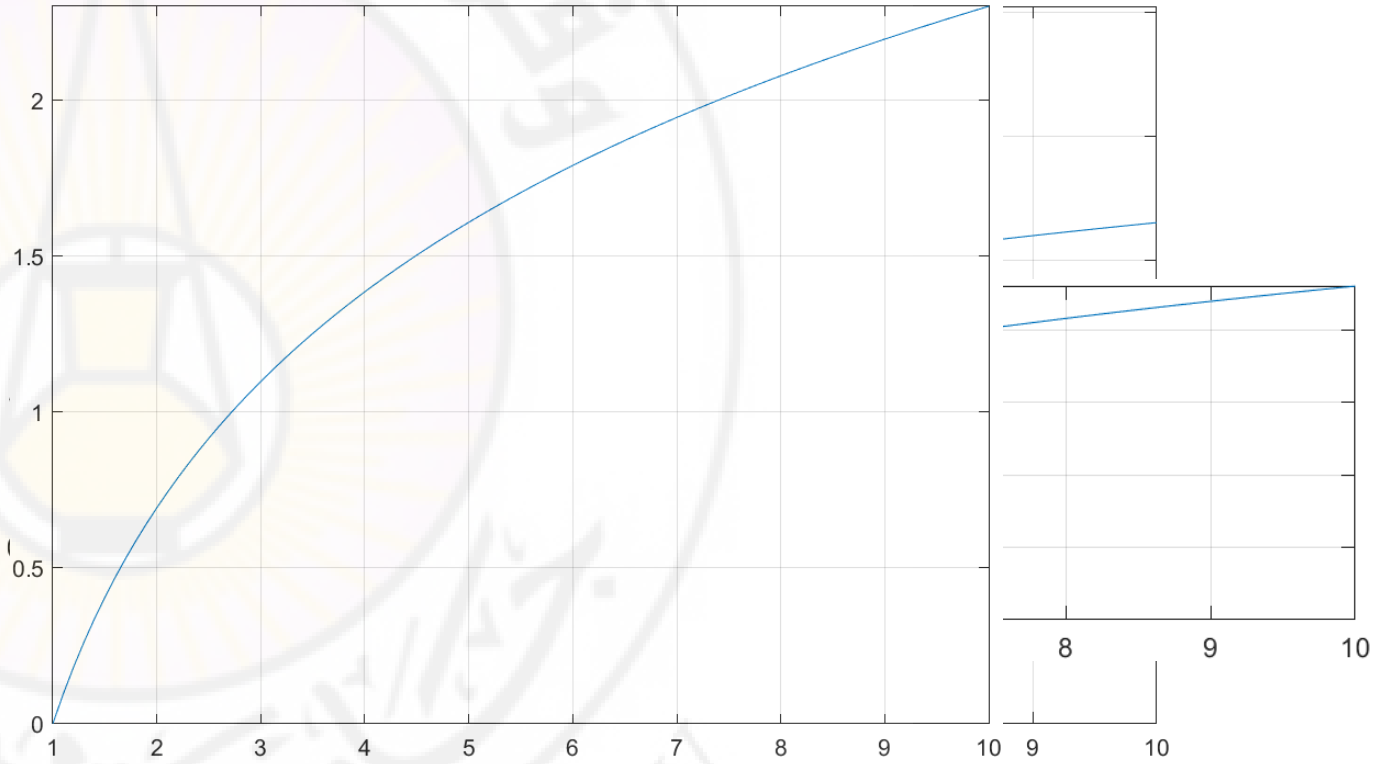
`b`	blue	`.`	point	`-`	solid
`g`	green	`o`	circle	`:`	dotted
`r`	red	`x`	x-mark	`-.`	dashdot
`c`	cyan	`+`	plus	`--`	dashed
`m`	magenta	`*`	star	(none)	no line
`y`	yellow	`s`	square		
`k`	black	`d`	diamond		
`w`	white	`v`	triangle (down)		
		`^`	triangle (up)		
		`<`	triangle (left)		
		`>`	triangle (right)		
		`p`	pentagram		
		`h`	hexagram		

Plot

```
x=1:10;
y=log(x);
plot(x,y, '*:')

x=1:0.1:10;
y=log(x);
plot(x,y, '-')

grid
axis equal
axis([1 10 0 2.5])  % x1 x2 y1 y2
axis([1 Inf 0 Inf])
daspect([3 1 1])    % Data aspect ratio
```

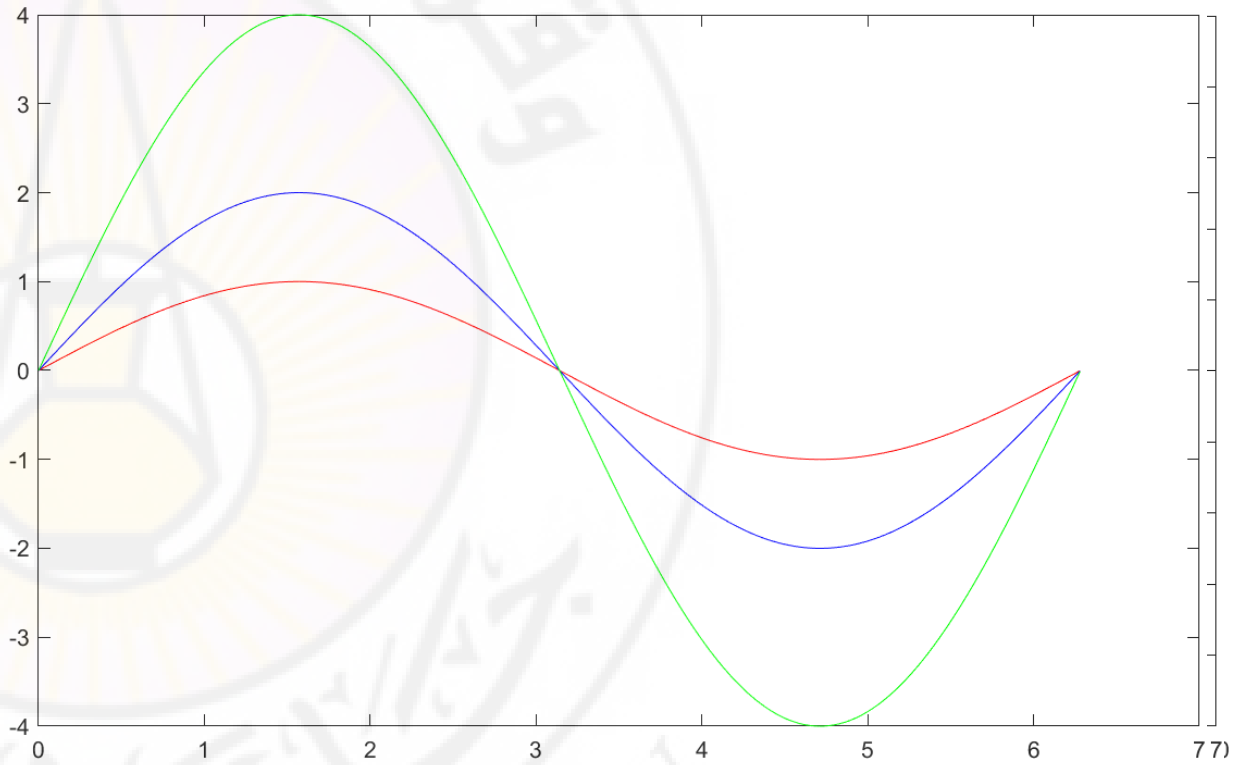


Plot

```
x = 0:0.1:2*pi;
y = sin(x);
plot(x,y)

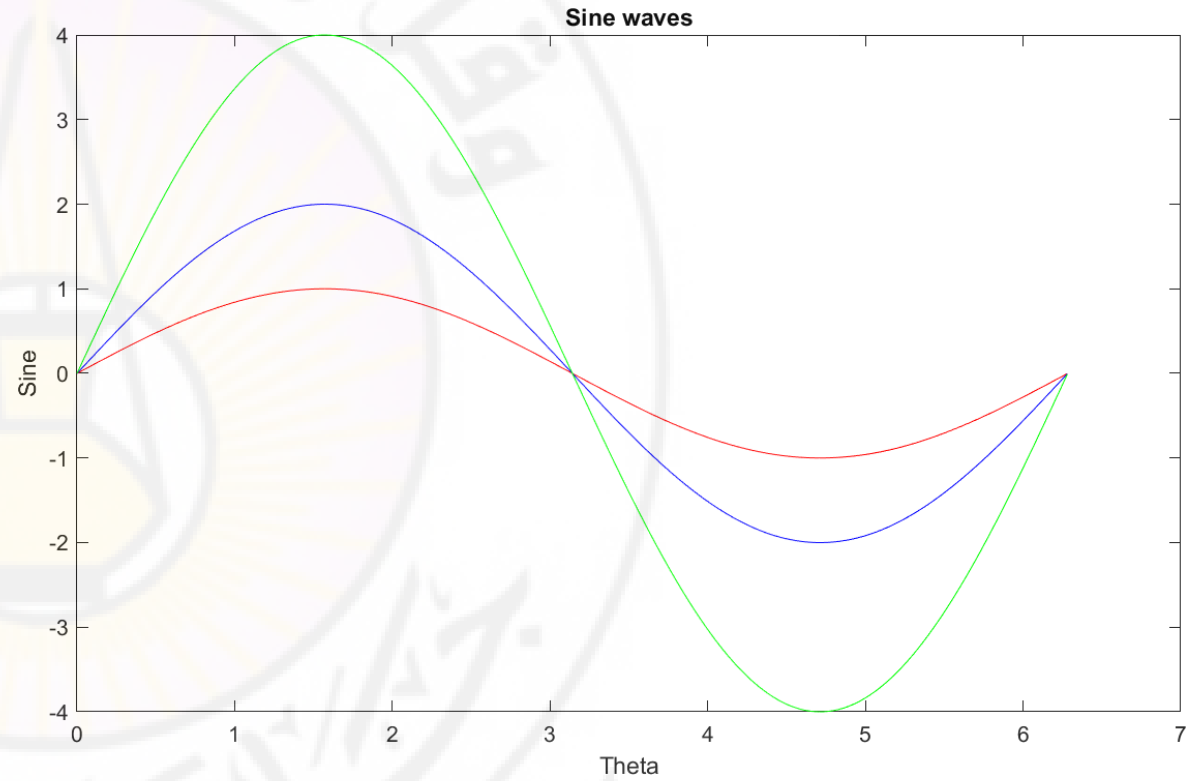
plot(y)
% plots Y columns vs. index

x = 0:.01:2*pi;
y1 = sin(x);
y2 = sin(2*x);
y3 = sin(4*x);
plot(x, y1, 'r', x, y2 , 'b', x, y3, 'g')
plot(x, y1, 'r', x, 2*y1 , 'b', x, 4*y1, 'g')
```



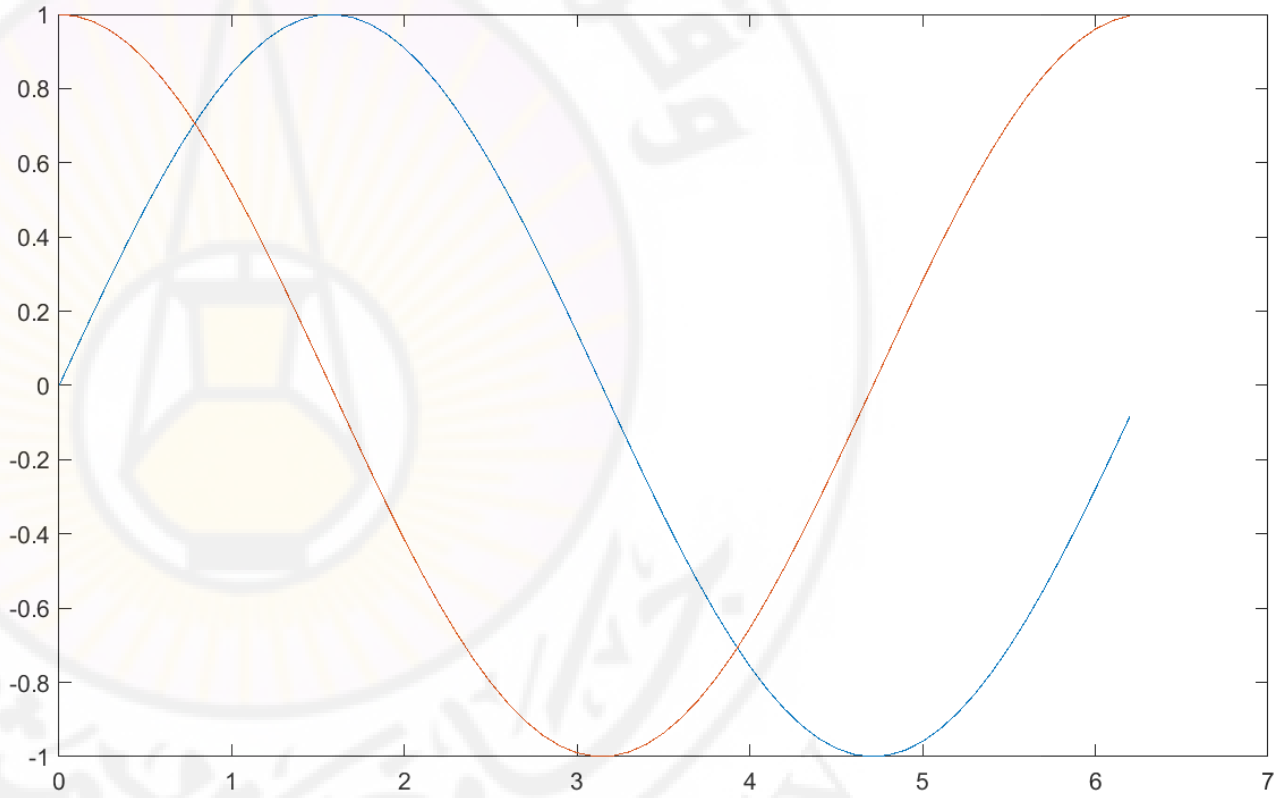
Plot

- `title('Sine waves')`
- `xlabel('Theta')`
- `ylabel('Sine')`
- `clf`
- `hold on`
- `hold off`



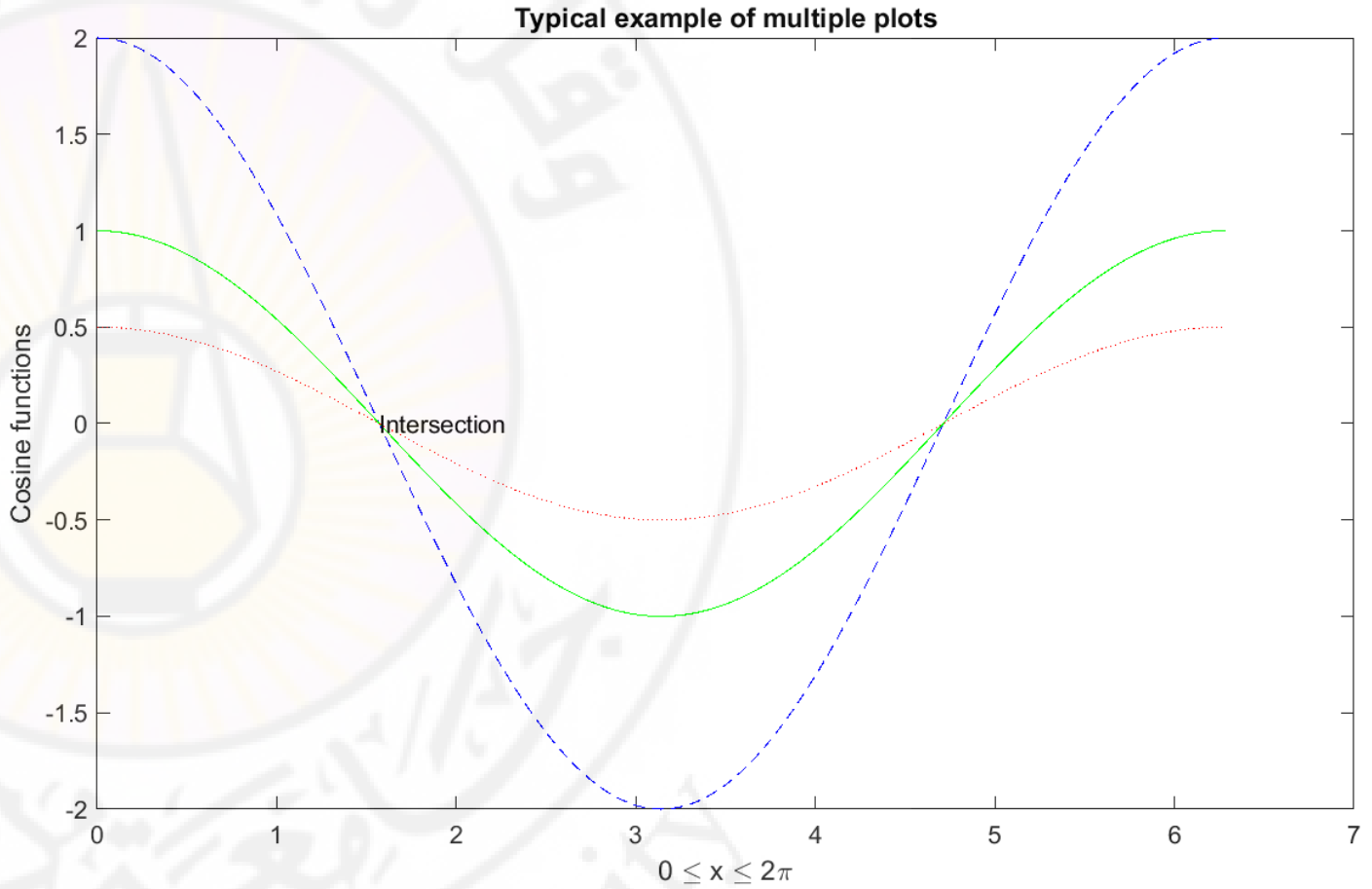
Plot

```
x=0:0.1:2*pi;
plot(x, sin(x))
hold on
plot(x, cos(x))
hold off
```



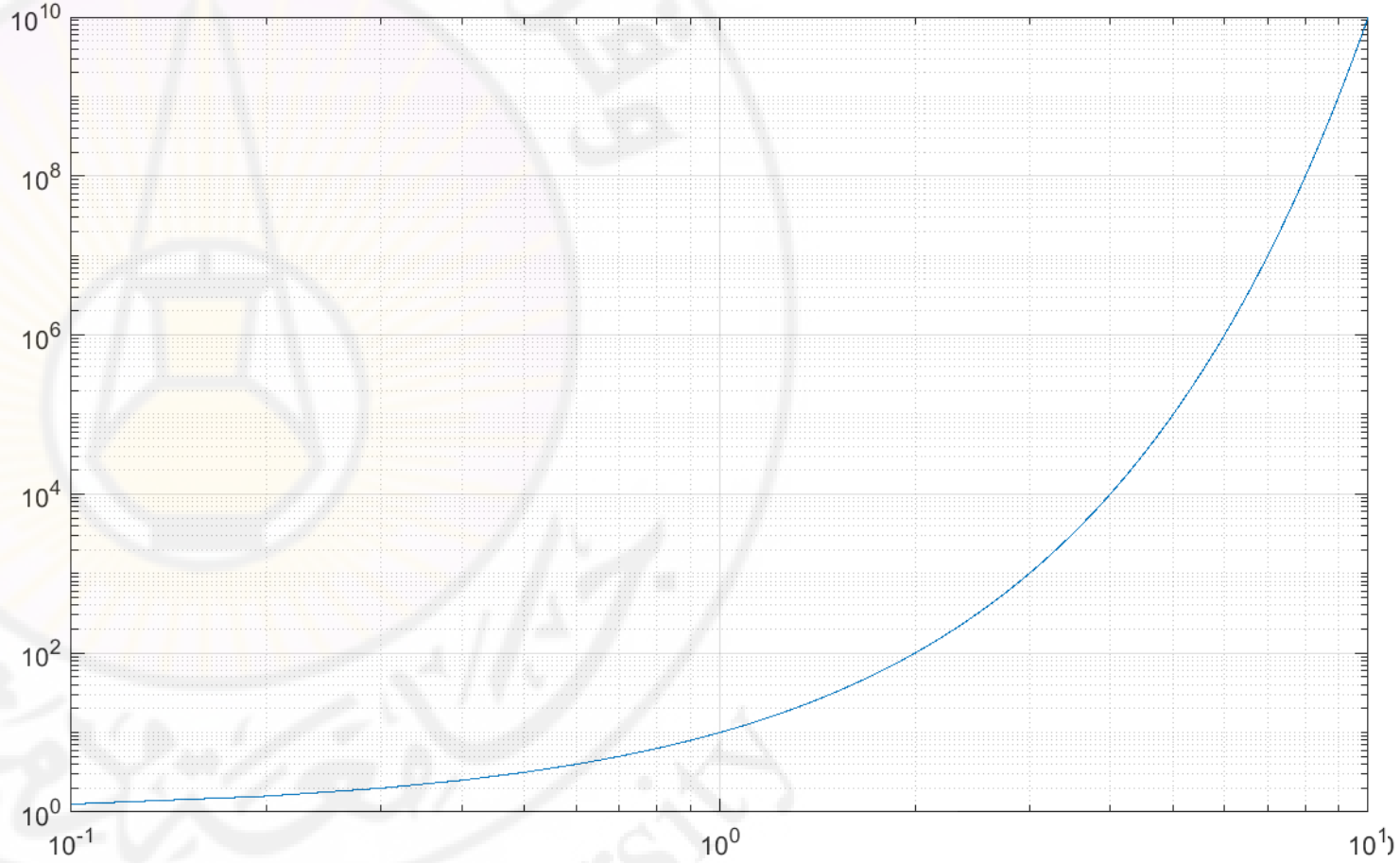
Plot

```
x = 0:pi/100:2*pi;
y1 = 2*cos(x);
y2 = cos(x);
y3 = 0.5*cos(x);
plot(x,y1,'--b',x,y2,'-g',x,y3,':r')
% Use LaTeX
xlabel('0 \leq x \leq 2\pi')
ylabel('Cosine functions')
title('Typical example')
text(pi/2, 0, 'Intersection')
```



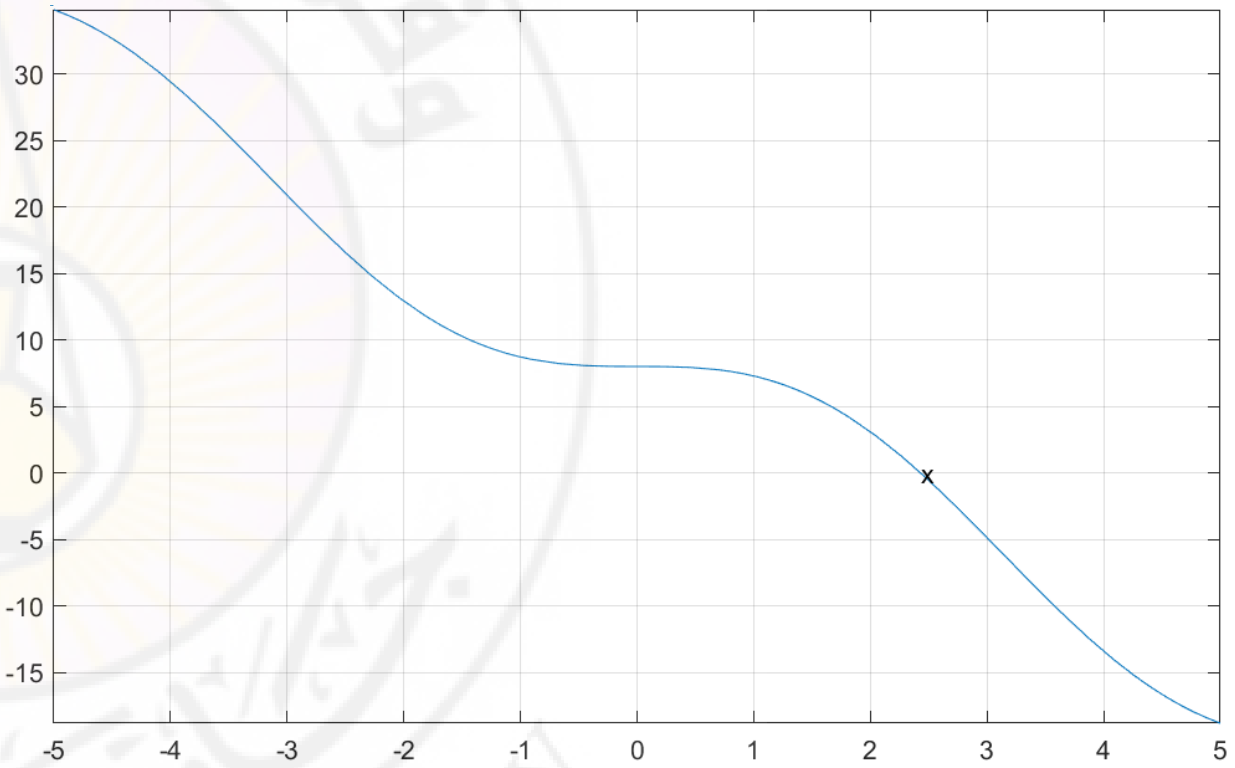
Logarithmic Scale

```
x=0:0.1:10;
y=10.^x;
plot(x,y),          grid on
semilogx(x,y),      grid on
semilogy(x,y),      grid on
loglog(x,y),        grid on
```



Fplot

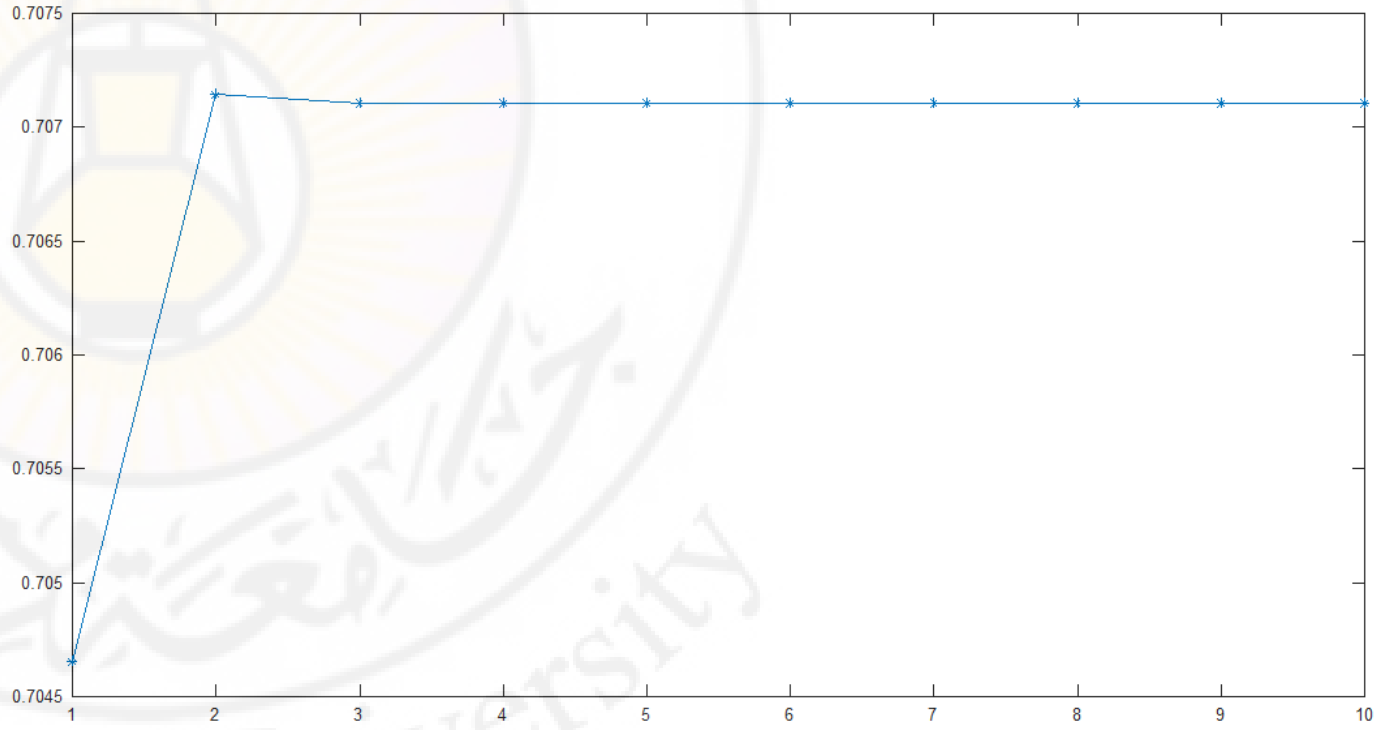
```
fplot(@sin)
fplot(@sin, [0 pi])
fplot(@(x) x.^2 + 5.*x + 6,[-6.5 1.5])
fplot(@(x) 8 - 4.5 * (x-sin(x)))
fzero(@(x) 8 - 4.5 * (x-sin(x)), 0)
ans =
    2.4305
```



Example 1

- Engineers can determine the required number of steps
  - Accuracy
  - Efficiency

Max n	sin(45°)
1	0.704652651209168
2	0.707143045779360
3	0.707106469575178
4	0.707106782936867
5	0.707106781179619
6	0.707106781186568
7	0.707106781186547
8	0.707106781186547
9	0.707106781186547
10	0.707106781186547

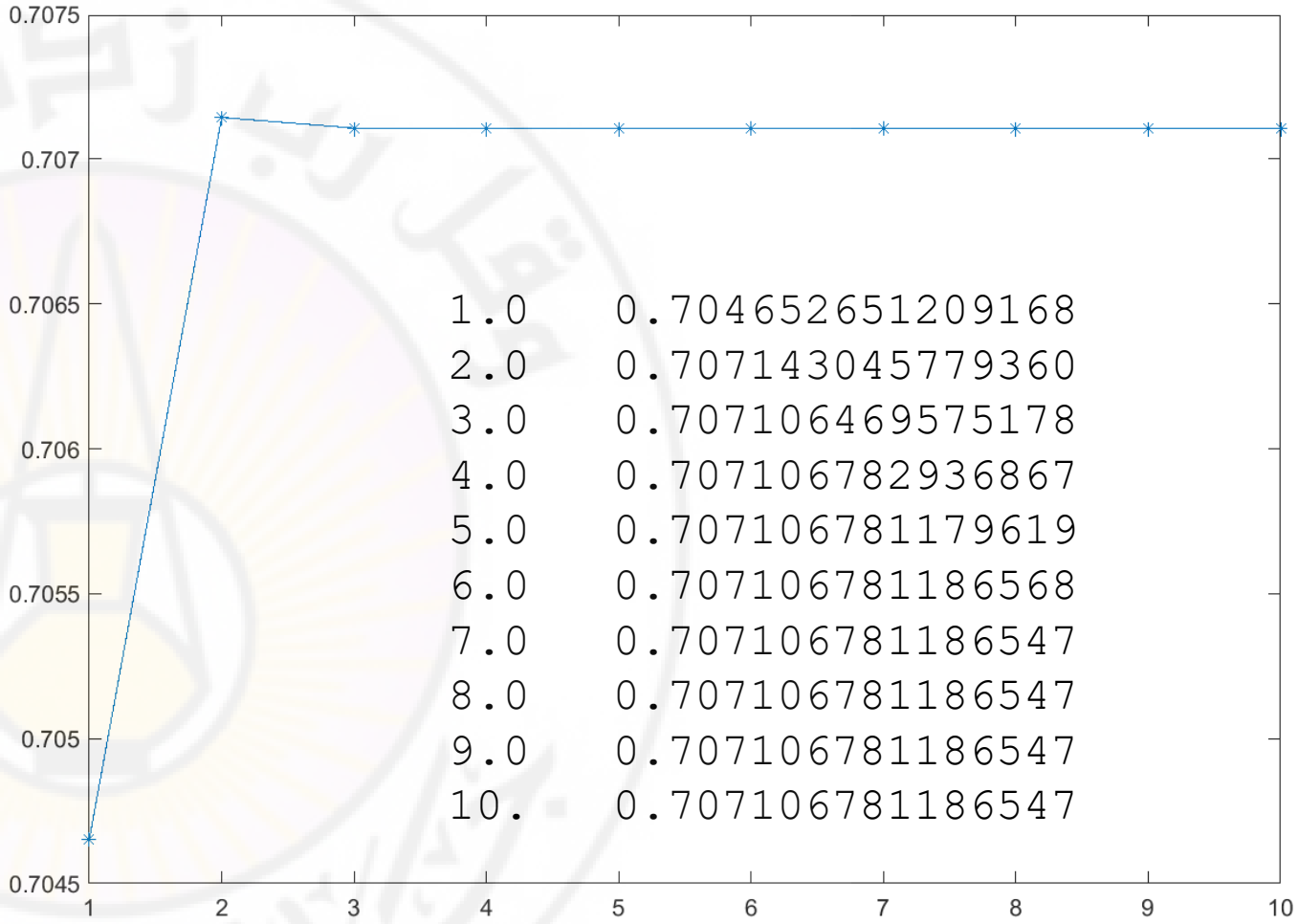


Number of Steps

```
theta = deg2rad(45);
x = 1:10;
y = zeros(size(x));

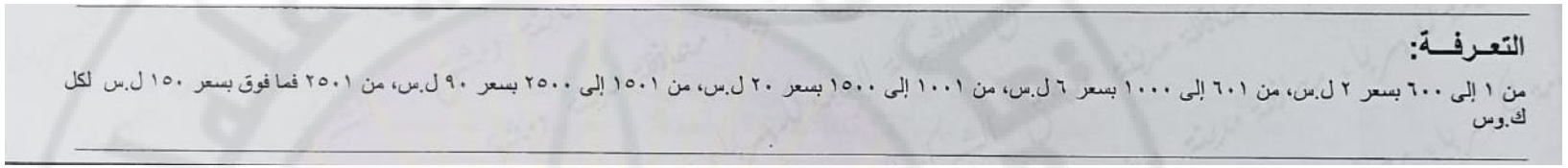
for n = x
    y(n) = sin_ex(theta, n);
end
disp([x; y]');
plot(x, y, '*-')

function result = sin_ex(theta, maxn)
    result = 0;
    for n = 0 : maxn
        result = result + (-1)^n * theta ^(2*n+1) / factorial(2*n+1);
    end
end
```




```
1.0    0.704652651209168
2.0    0.707143045779360
3.0    0.707106469575178
4.0    0.707106782936867
5.0    0.707106781179619
6.0    0.707106781186568
7.0    0.707106781186547
8.0    0.707106781186547
9.0    0.707106781186547
10.    0.707106781186547
```

Example 2



التعـرفـــة:

من ١ إلى ٦٠٠ بسعر ٢ ل.س، من ٦٠١ إلى ١٠٠٠ بسعر ٦ ل.س، من ١٠٠١ إلى ١٥٠٠ بسعر ٢٠ ل.س، من ١٥٠١ إلى ٢٥٠٠ بسعر ٩٠ ل.س، من ٢٥٠١ فما فوق بسعر ١٥٠ ل.س لكل ك.وس

```
function tariff = ebill_ex(kwh)

if kwh < 1
  tariff = 0;
elseif kwh <= 600
 tariff = kwh * 2;
elseif kwh <= 1000
 tariff = (kwh - 600) * 6 + 600 * 2;
elseif kwh <= 1500
 tariff = (kwh - 1000) * 20 + 600 * 2 + 400 * 6;
elseif kwh <= 2500
 tariff = (kwh - 1500) * 90 + 600 * 2 + 400 * 6 + 500 * 20;
else
 tariff = (kwh - 2500) * 150 + 600 * 2 + 400 * 6 + 500 * 20 + 1000 * 90;
end
```

Compare to fixed rate of 25 per kwh!

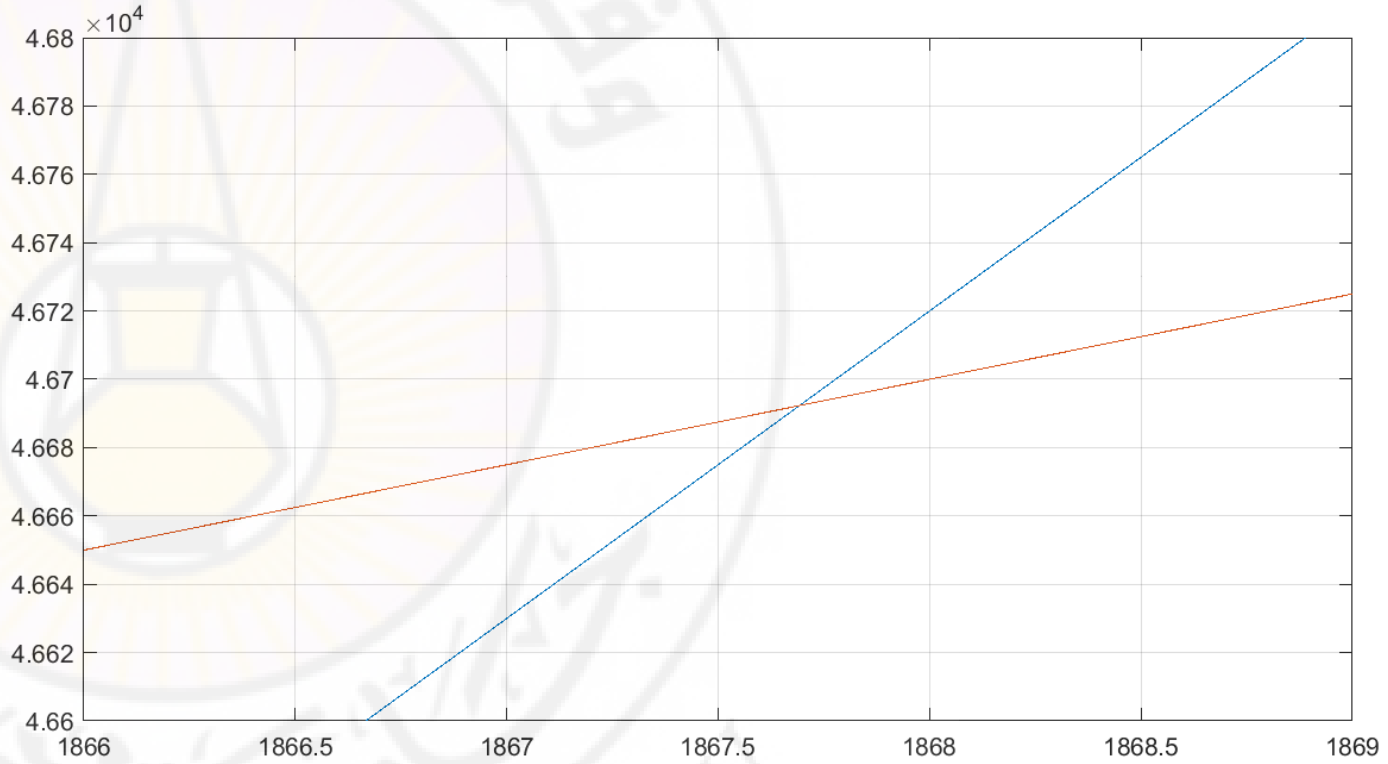
Tariff

```
x = 1:2600;
y = zeros(size(x));

for kws = x
    y(kws) = ebill_ex(kws);
end

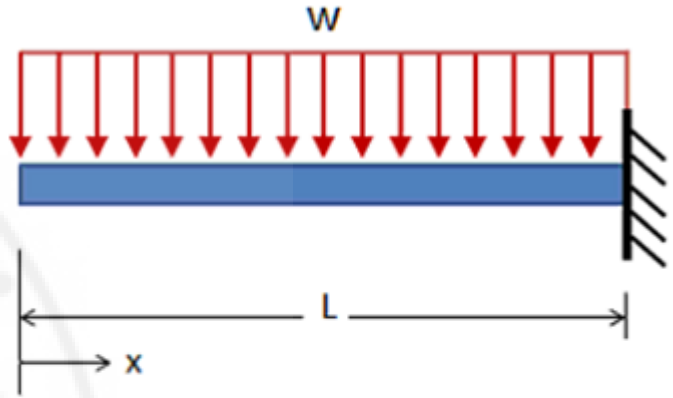
plot(x, y)

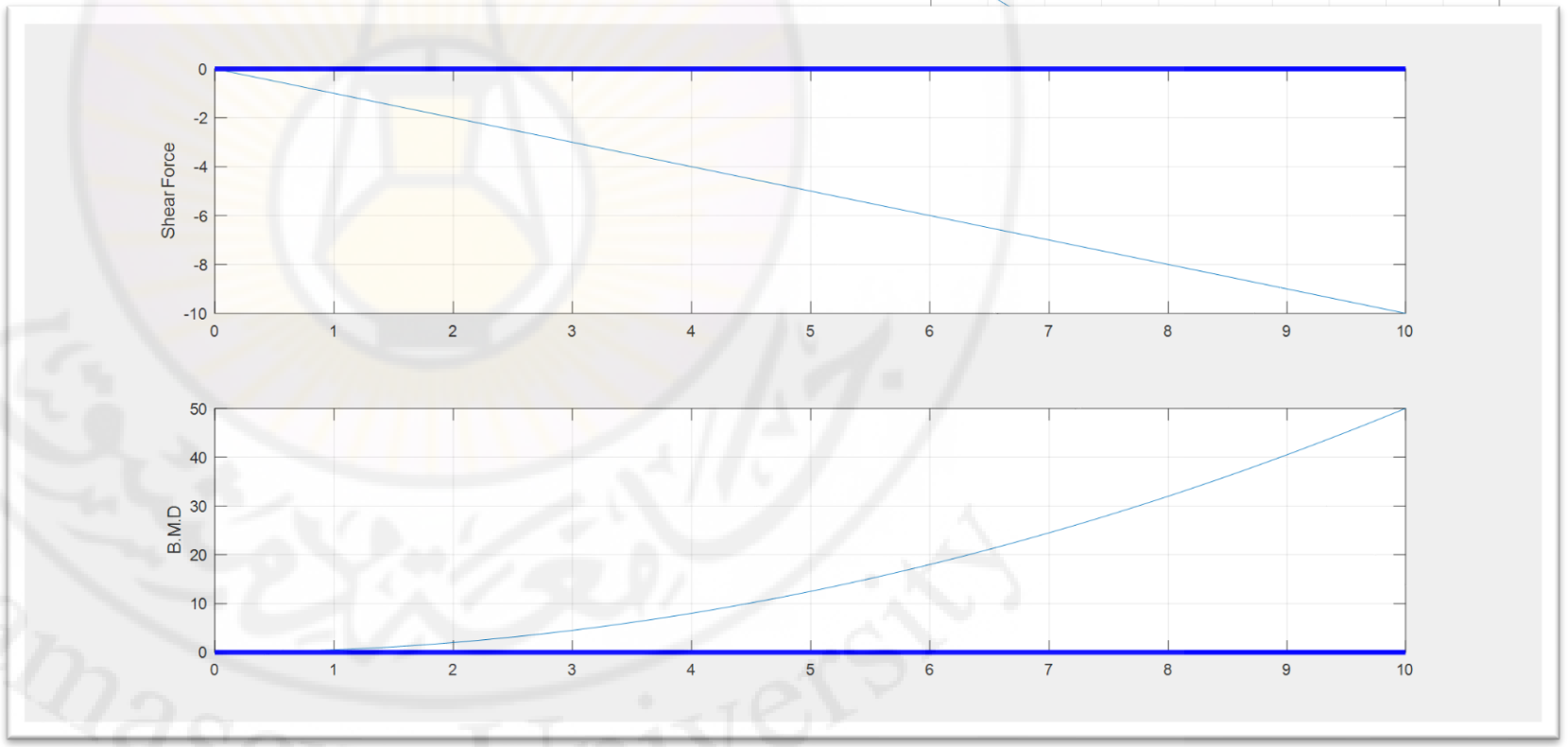
y2 = 25 * x;
hold on
plot(x, y2)
axis([1866 1869 46600 46800])
```



Example 3

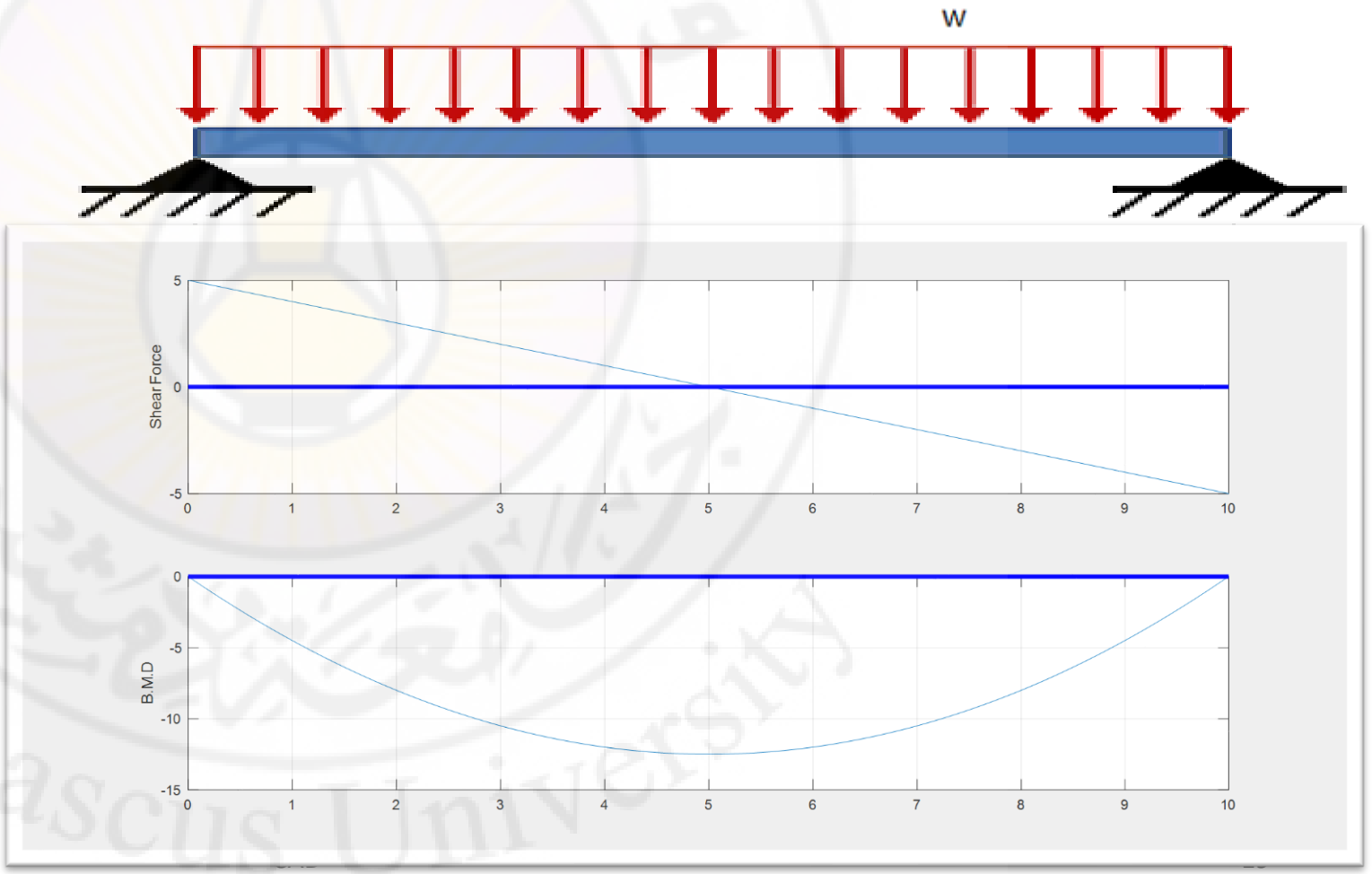
```
L = input('Length L='); % e.g. 10
w = input('w=');        % e.g. 1
x = 0:0.1:L;
v = -w*x;
m = w*x.*x/2;
subplot(2,1,1);
plot(x, v), grid
ylabel('Shear Force');
subplot(2,1,2);
plot(x, m), grid
ylabel('B.M.D');
```





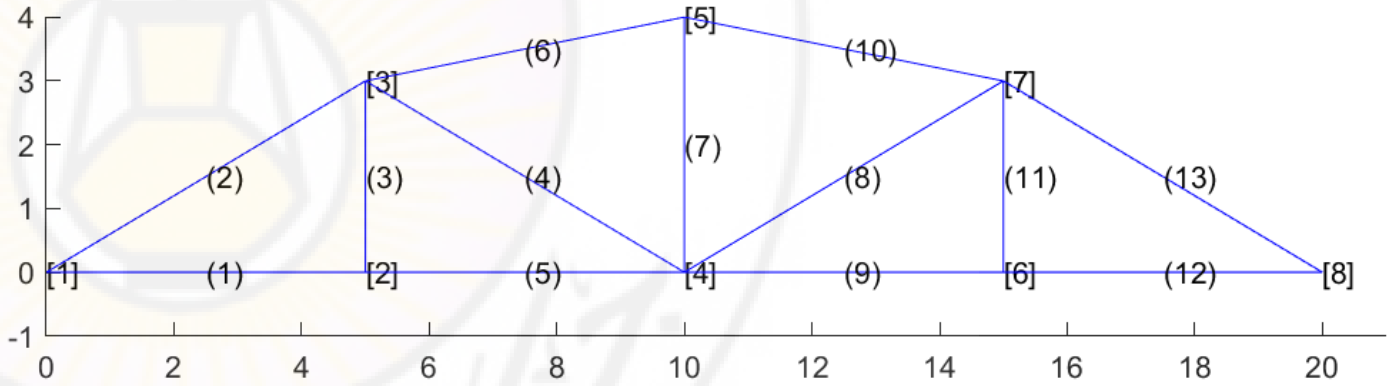
Example 4

```
L = input('Length L='); % e.g. 10
w = input('w=');        % e.g. 1
x = 0:0.1:L;
R = w*L/2; % Support reaction
v = R-w*x;
m = -R.*x+w*x.*x/2;
subplot(2,1,1);
plot(x, v), grid
ylabel('Shear Force');
subplot(2,1,2);
plot(x, m), grid
ylabel('B.M.D');
```



Truss

```
x0 = 0; y0 = 0; dx = 5; y1 = 3; y2 = 4;
% node x y
nodes = [
x0 + 0 * dx     y0;
x0 + 1 * dx     y0;
x0 + 1 * dx     y1;
x0 + 2 * dx     y0;
x0 + 2 * dx     y2;
x0 + 3 * dx     y0;
x0 + 3 * dx     y1;
x0 + 4 * dx     y0;
];
```



Truss

```
clf

axis equal

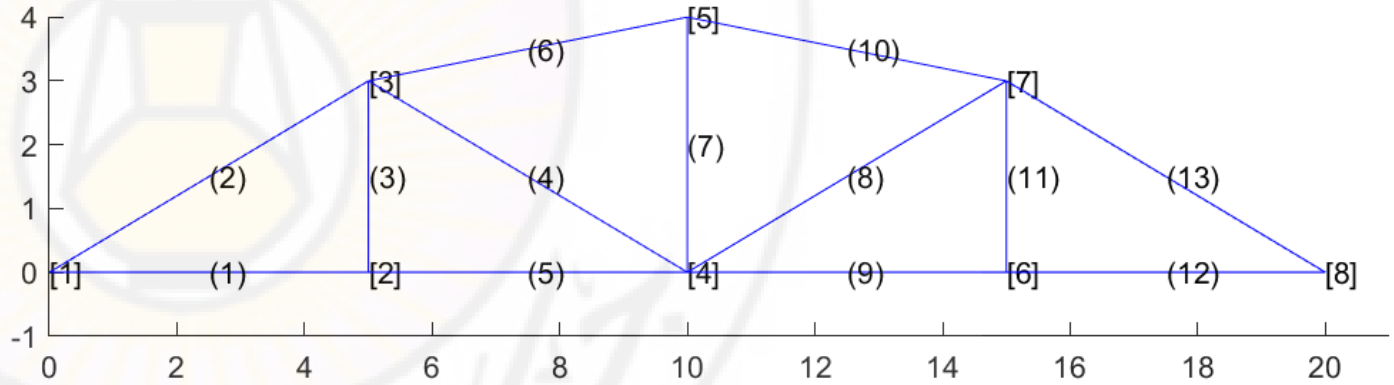
hold on



for n = 1:length(nodes)

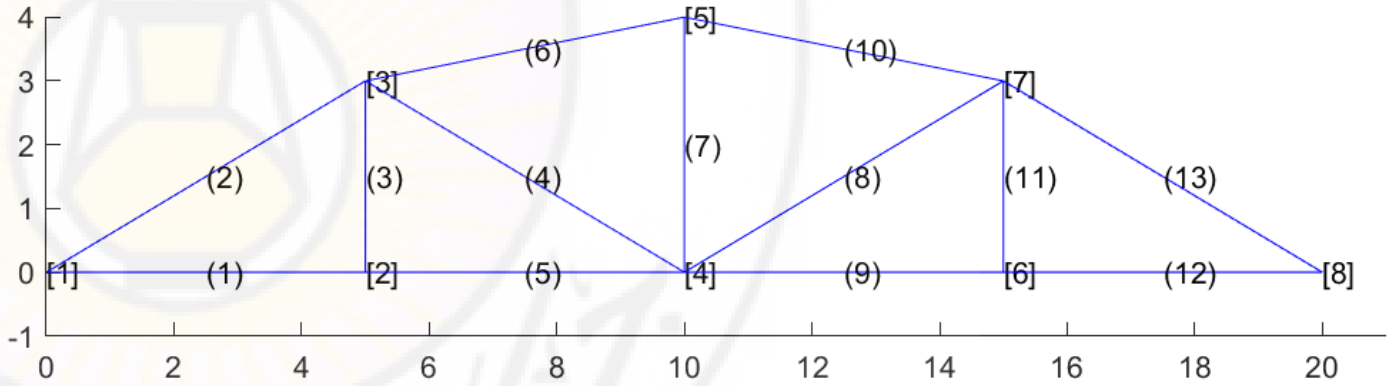
    % x = nodes(n, 1)

    text(nodes(n, 1), nodes(n, 2), "["+num2str(n)+"]")
end
```



Truss

```
elements = [
1 2;
1 3;
2 3;
3 4;
2 4;
3 5;
4 5;
4 7;
4 6;
5 7;
6 7;
6 8;
7 8
];
```



Truss

```
for el = 1:length(elements)
    n1 = elements(el, 1);
    n2 = elements(el, 2);
    plot([nodes(n1, 1) nodes(n2, 1)], [nodes(n1, 2) nodes(n2, 2)], 'b')
    x = (nodes(n1, 1) + nodes(n2, 1))/2;
    y = (nodes(n1, 2) + nodes(n2, 2))/2;
    text(x, y, "("+num2str(el)+")")
end
```

Truss

```
displacement = [
0 0;
0.0012 -0.0079;
0.0040 -0.0073;
0.0023 -0.0090;
0.0024 -0.0073;
0.0036 -0.0085;
0.0007 -0.0077;
0.0048 0; ];
```

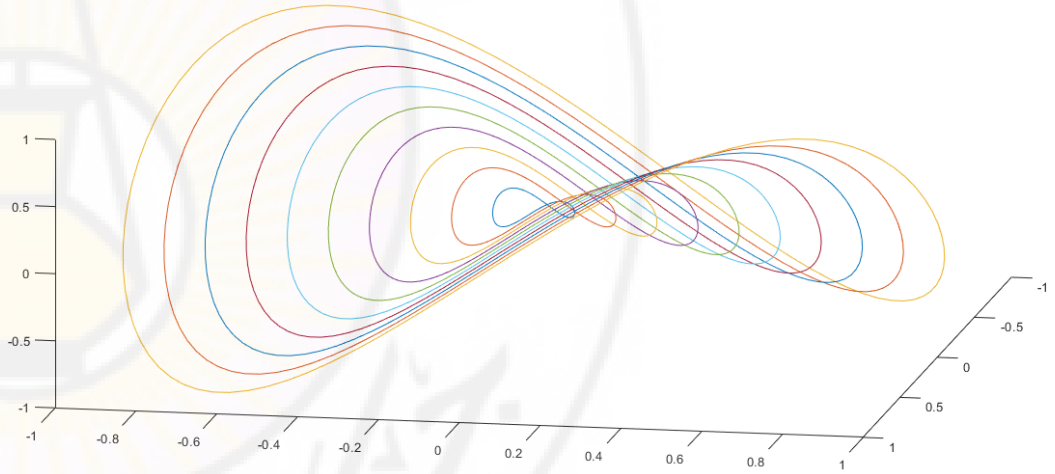
Deformed Truss

```
max_scale = 100;
delay = 0.01;
for scale = 1:max_scale
    clf
    axis equal
    grid, hold on
    coord = nodes + scale .* displacement;
    for el = 1:length(elements)
        n1 = elements(el, 1);
        n2 = elements(el, 2);
        plot([coord(n1, 1) coord(n2, 1)], [coord(n1, 2) coord(n2, 2)], 'b')
    end
    axis([0 21 -2 y2])
    drawnow
    pause(delay)
end
```



3D Graphics

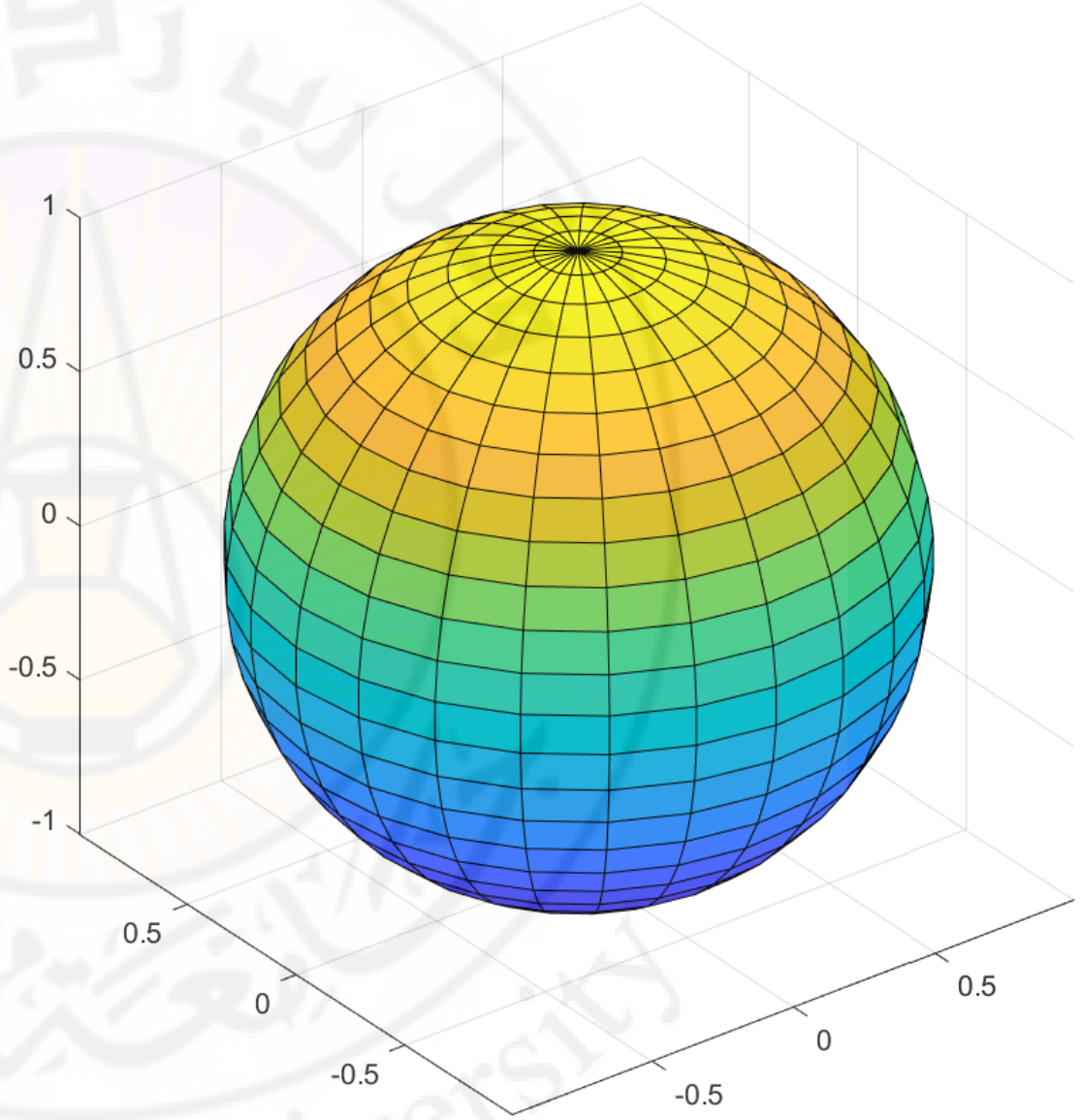




3D Graphics

```
sphere(25)
```

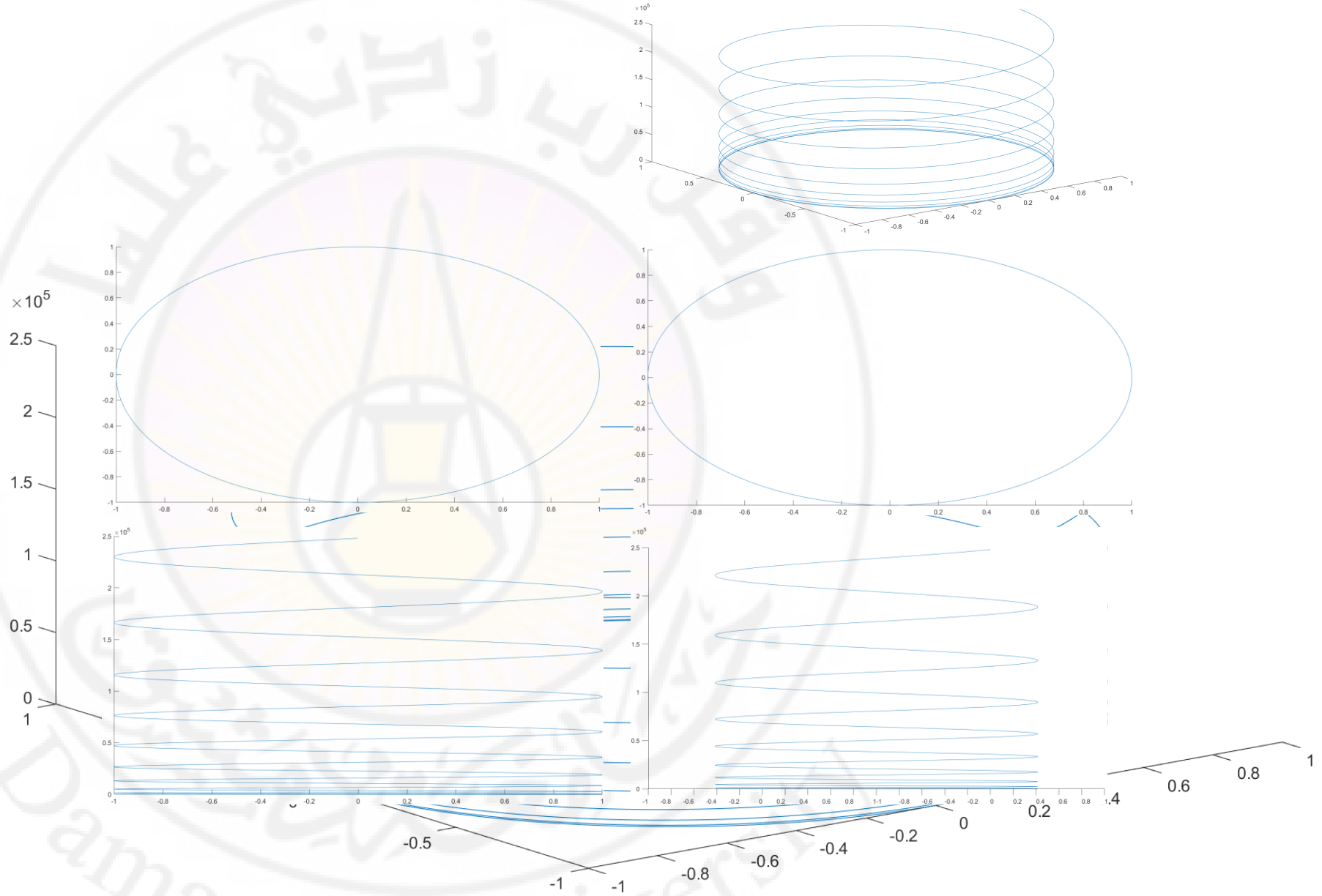
```
axis equal
```



Plot3

```
t=.01:.01:20*pi;
x=cos(t);
y=sin(t);
z=t.^3;

plot3(x,y,z)
view(2)
view(3)
view(45, 0)
view(90, 0)
view(0, 90)
```

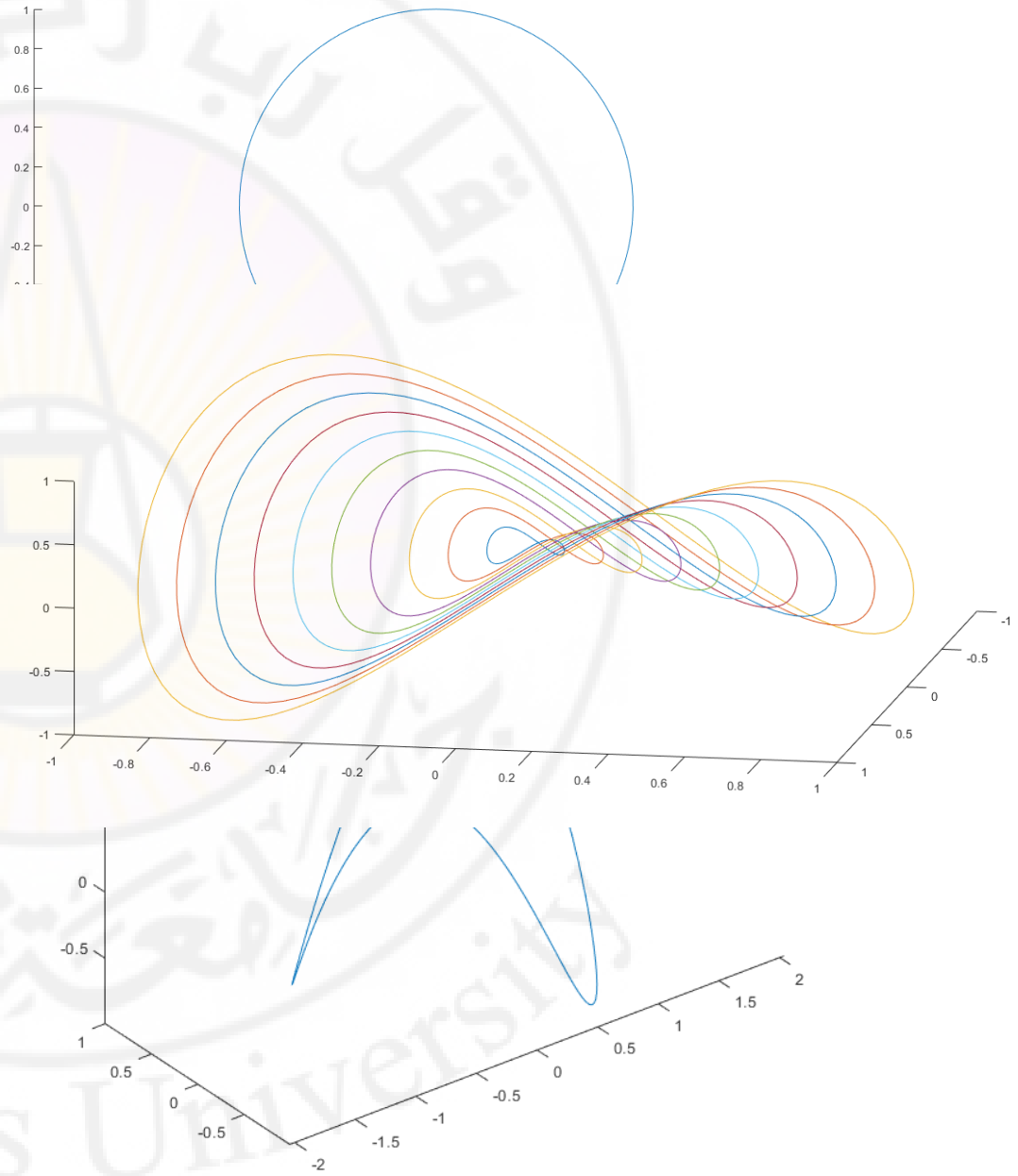


Plot3

```
theta = linspace(0,2*pi,100);

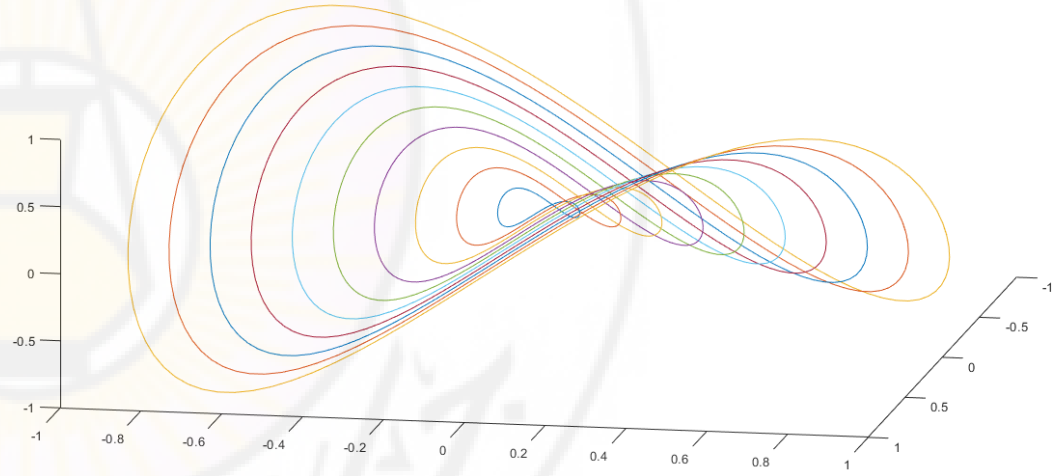
    x = 1*sin(theta);
    y = 1*cos(theta);
    z = 1*sin(2*theta);
    plot3(x,y,z);

view(2)
axis equal
view(3)
```



Plot3

```
clf
hold on
theta = linspace(0,2*pi,100);
for r = 0:0.1:1
    x = r*sin(theta);
    y = r*cos(theta);
    z = r*sin(2*theta);
    plot3(x,y,z);
end
view(2)
axis equal
view(3)
daspect([1 1 3])
```



3D Graphics

```
[x, y]=meshgrid(-2:0.5:2);
```

```
x =

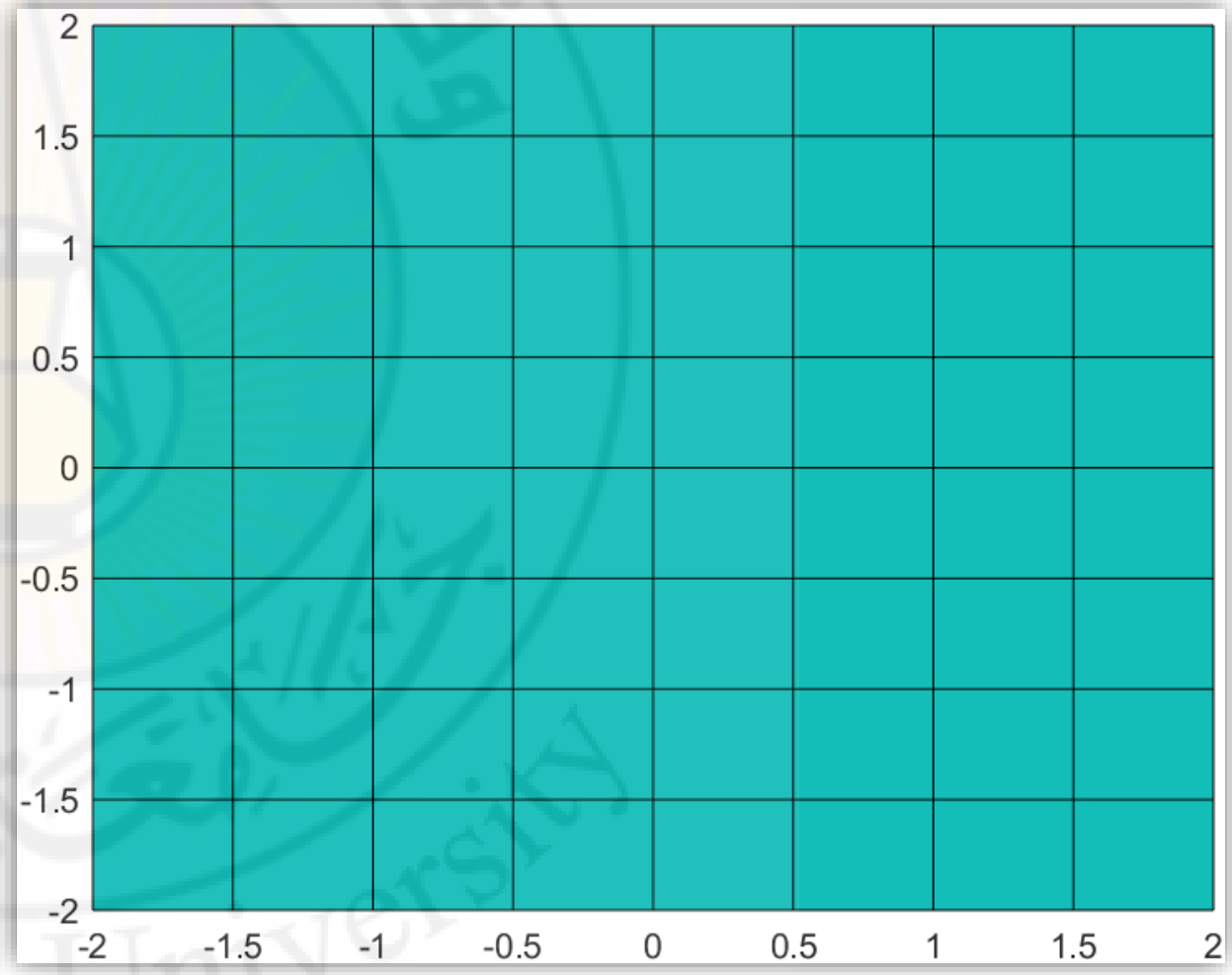
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
```

3D Graphics

```
[x, y]=meshgrid(-2:0.5:2);

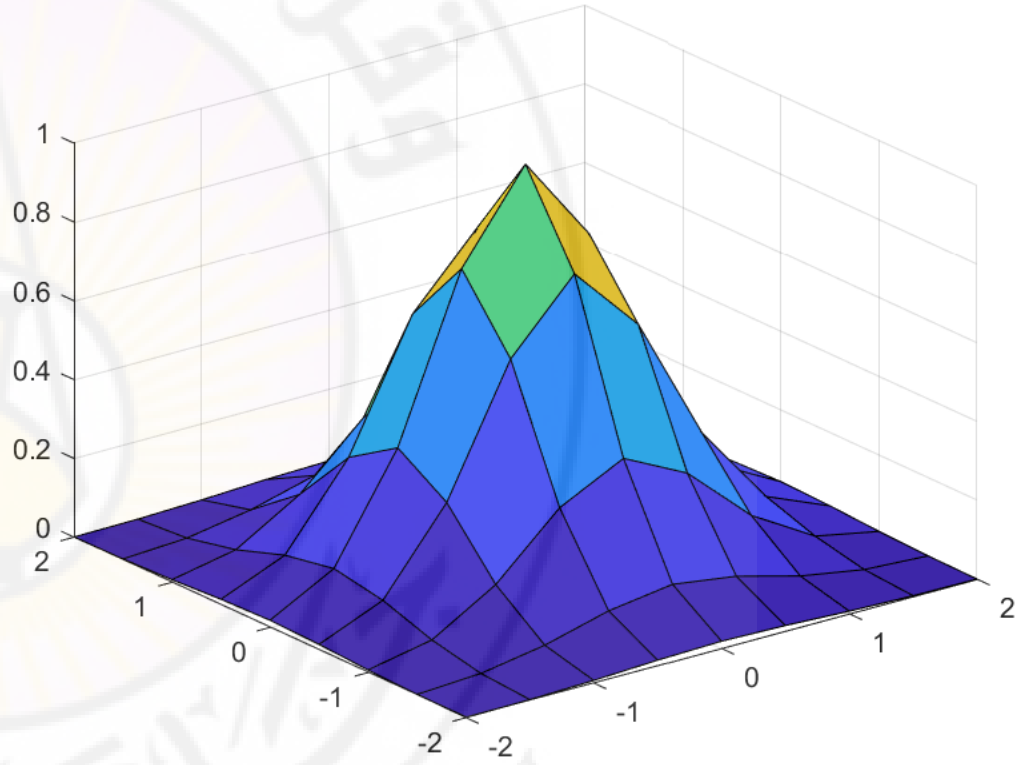
y =

   -2.0000   -2.0000   -2.0000   -2.0000
   -1.5000   -1.5000   -1.5000   -1.5000
   -1.0000   -1.0000   -1.0000   -1.0000
   -0.5000   -0.5000   -0.5000   -0.5000
         0         0         0         0
    0.5000    0.5000    0.5000    0.5000
    1.0000    1.0000    1.0000    1.0000
    1.5000    1.5000    1.5000    1.5000
    2.0000    2.0000    2.0000    2.0000
```



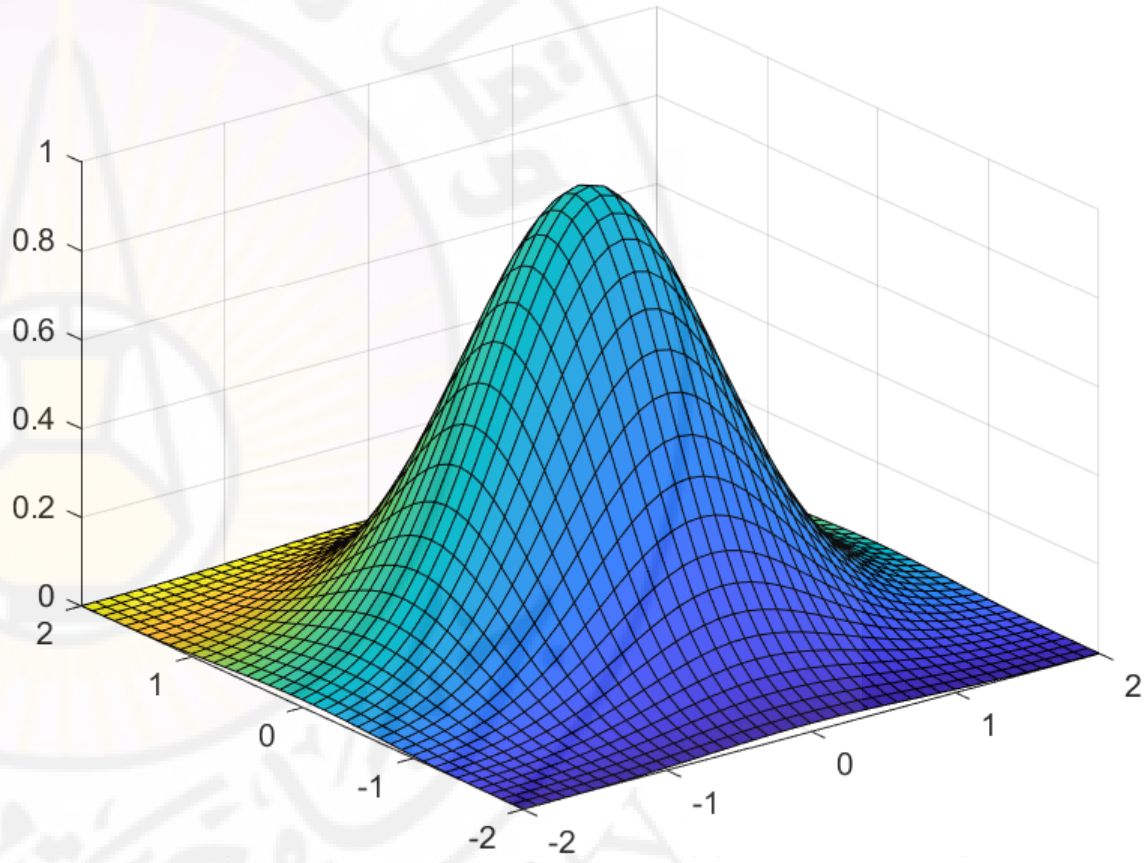
3D Graphics

```
[x, y]=meshgrid(-2:0.5:2);
z=exp(-x.^2 - y.^2);
surf(x, y, z)
```



3D Graphics

```
[x, y]=meshgrid(-2:0.1:2);
z=exp(-x.^2 - y.^2);
surf(x, y, z)
contour(x, y, z)
meshc(x, y, z)
view(0, -45)
surf(x, y, z, x)
surf(x, y, z, y)
```



3D Truss

- Nodes `[x y z]`
- `plot3`

Computer **A**ided **D**esign

MATLAB Introduction for Civil Engineers 4

22/11/2020

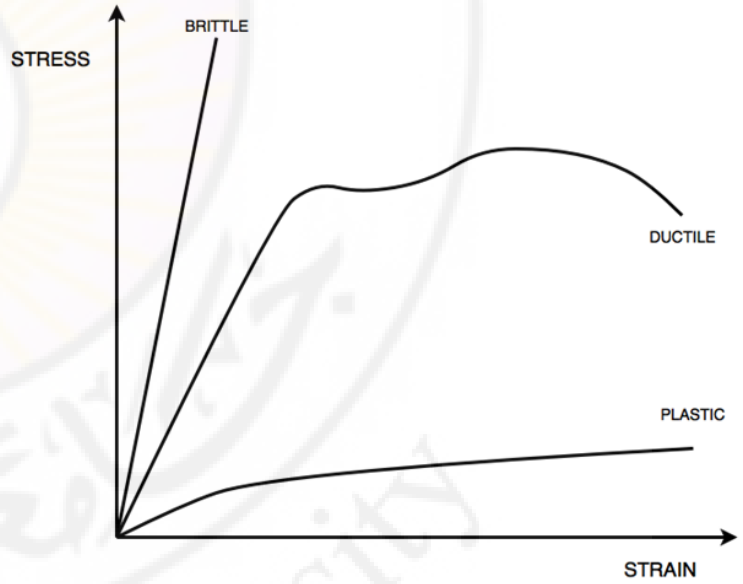
Wael Darwich

Content

- Statistics
  - Graphics
- Regression and Curve Fitting
  - Linear
  - Higher order
  - Example

Statistics

- Data (collected from measurements)
  - e.g. section dimensions, concrete compressive strength, soil density…
- Mathematical model
  - e.g. Young's modulus $= \frac{\sigma}{\varepsilon} = \frac{F/A}{dL/L}$



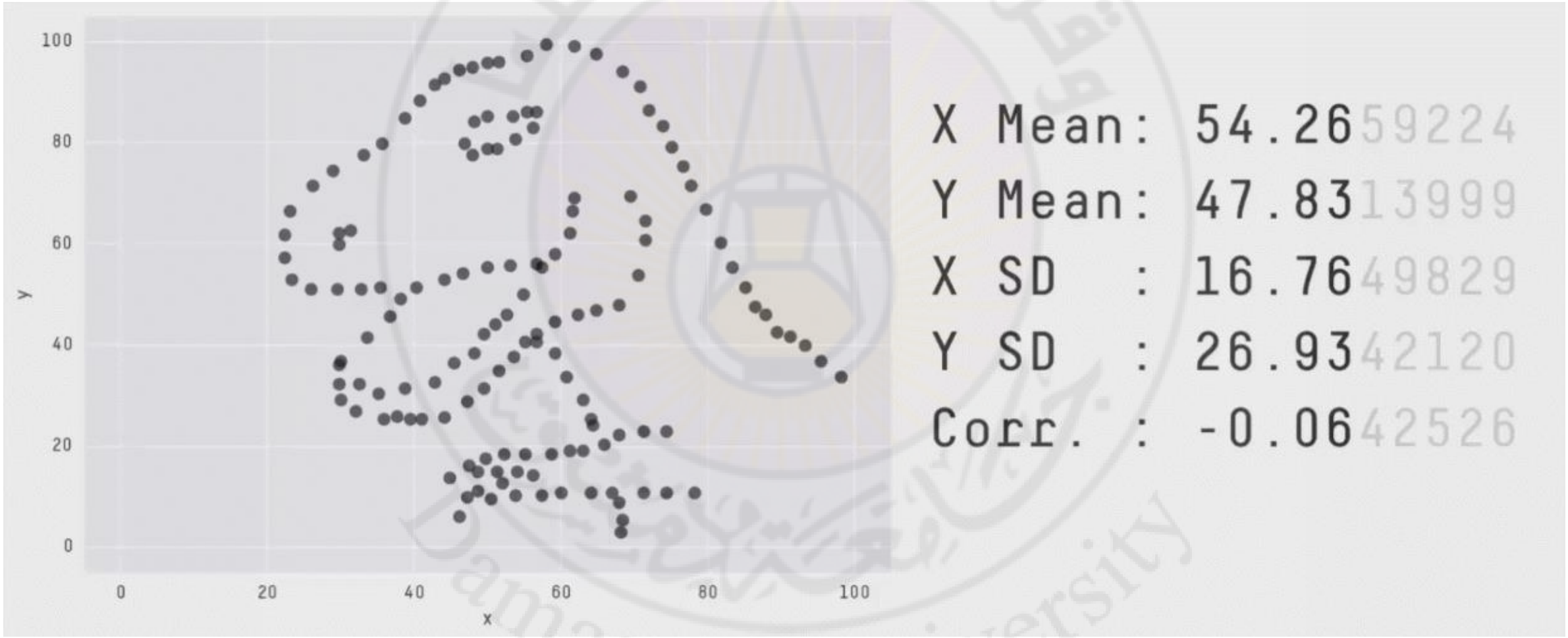
Data Analysis

- S275 yield strength specimens $f_y = 275\ MPa$
- `[328.0 307.1 305.8 292.1 305.7 325.7 343.3 326.8 307.5 314.3]`
- `[`**`292.1`** `305.7 305.8 307.1 307.5 314.3 325.7 326.8 328.0` **`343.3`**`]`
- Range: max – min value `= 51.1821` `range`
- Mean: the average `= 315.6246` `mean`
- Median: middle number `= 310.9009` `median`
- Mode: most often `= 292.105` `mode`
- Variance: dispersion of data `= 225.9345` `var`
- Standard deviation: $= \sqrt{\text{Variance}}$ `= 15.0311` `std`

Data Analysis

- $x = [x_1, x_2 \dots x_N]$
- $\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$
- Range = Max - Min
- $s_N = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N}}$
- Never trust summary statistics alone;
- Always visualize your data

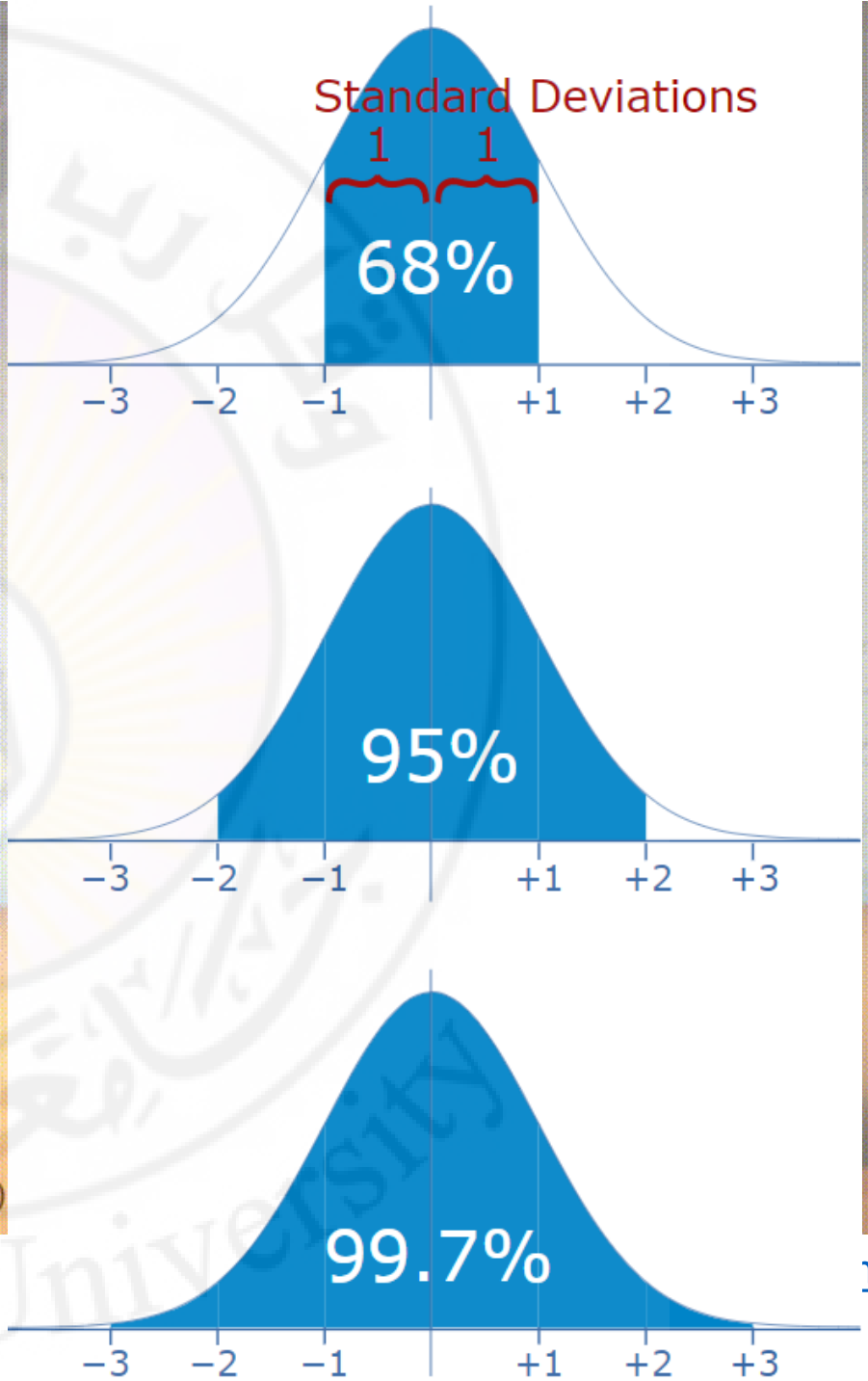
Data Visualization



https://www.autodesk.com/research/publications/same-stats-different-graphs

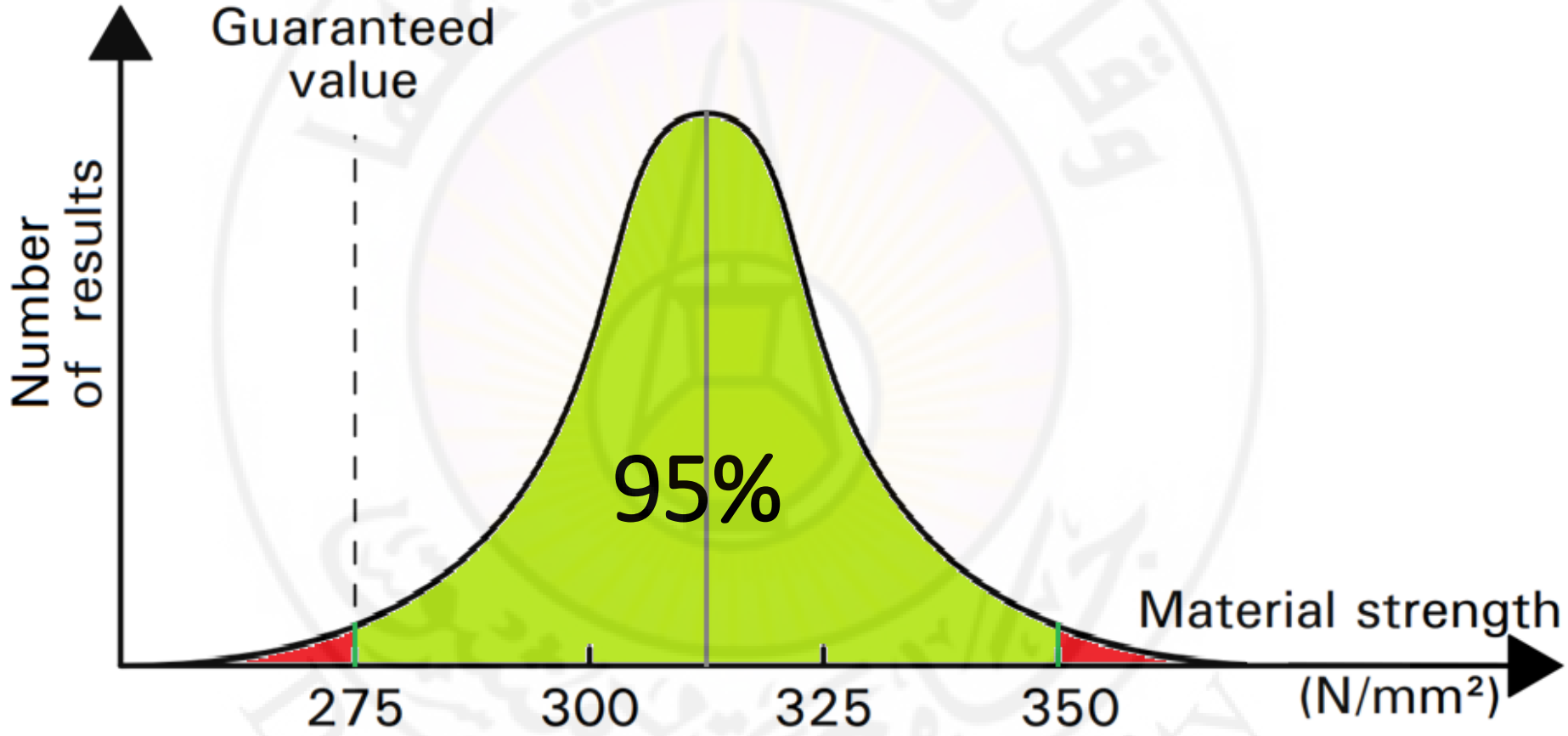
Normal Distribution

- Bell curve
- Symmetry about the centre
- mean = median = mode



ıx.html

S275



In this case the number of results falling below the guaranteed minimum is very small

Graphics

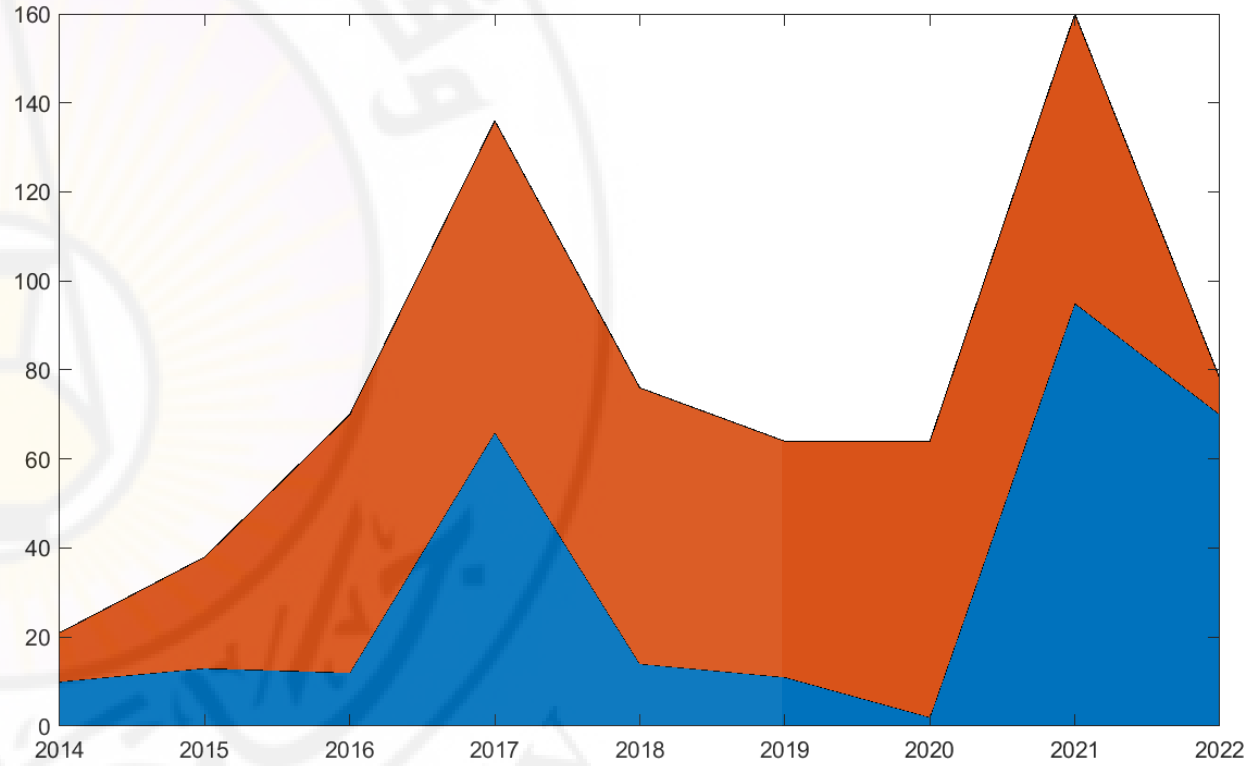
- Bar
- Area
- Scatter
- Histogram

Bar & Area

```
year = 2014:2022;
count = [10 25 58 70 62 53 62 65 8];
bar(year, count)


counts = [10 11; 13 25; 12 58;
          66 70; 14 62; 11 53;
          2 62; 95 65; 70 8;];
bar(year, counts)

area(year, count)
area(year, counts)
```



Random Data

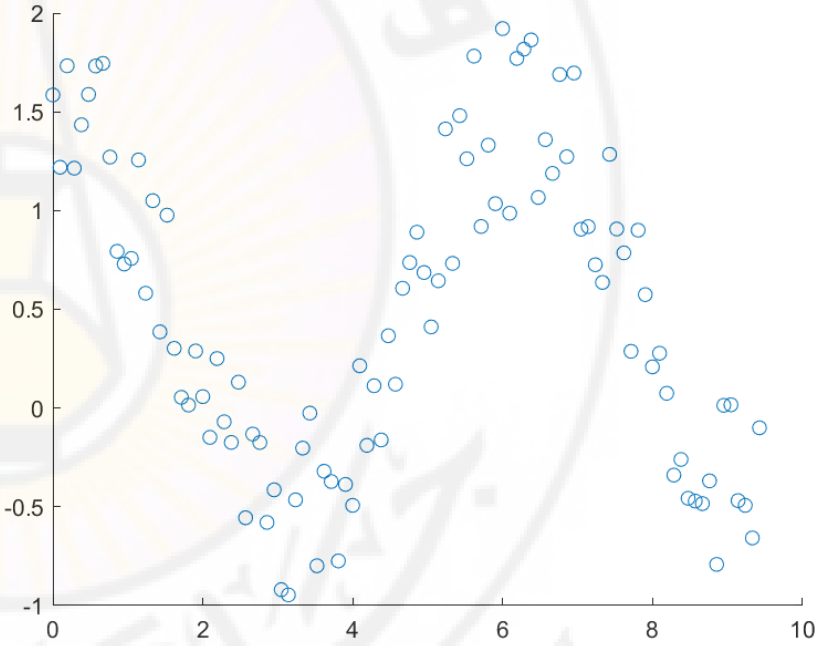
```
rand(1)          0.3692   between 0 and 1
rand(3)                  0.1112    0.2417    0.1320
                         0.7803    0.4039    0.9421
                         0.3897    0.0965    0.9561

                 0.5752    0.3532
rand(3, 2)       0.0598    0.8212
                 0.2348    0.0154

rand(1, 10)
     0.0430    0.1690    0.6491    0.7317    0.6477    0.4509    0.5470    0.2963    0.7447    0.1890
```

Scatter

```
x = linspace(0, 3*pi, 100);
y = cos(x) + rand(1, 100);
scatter(x, y)
```



Histogram

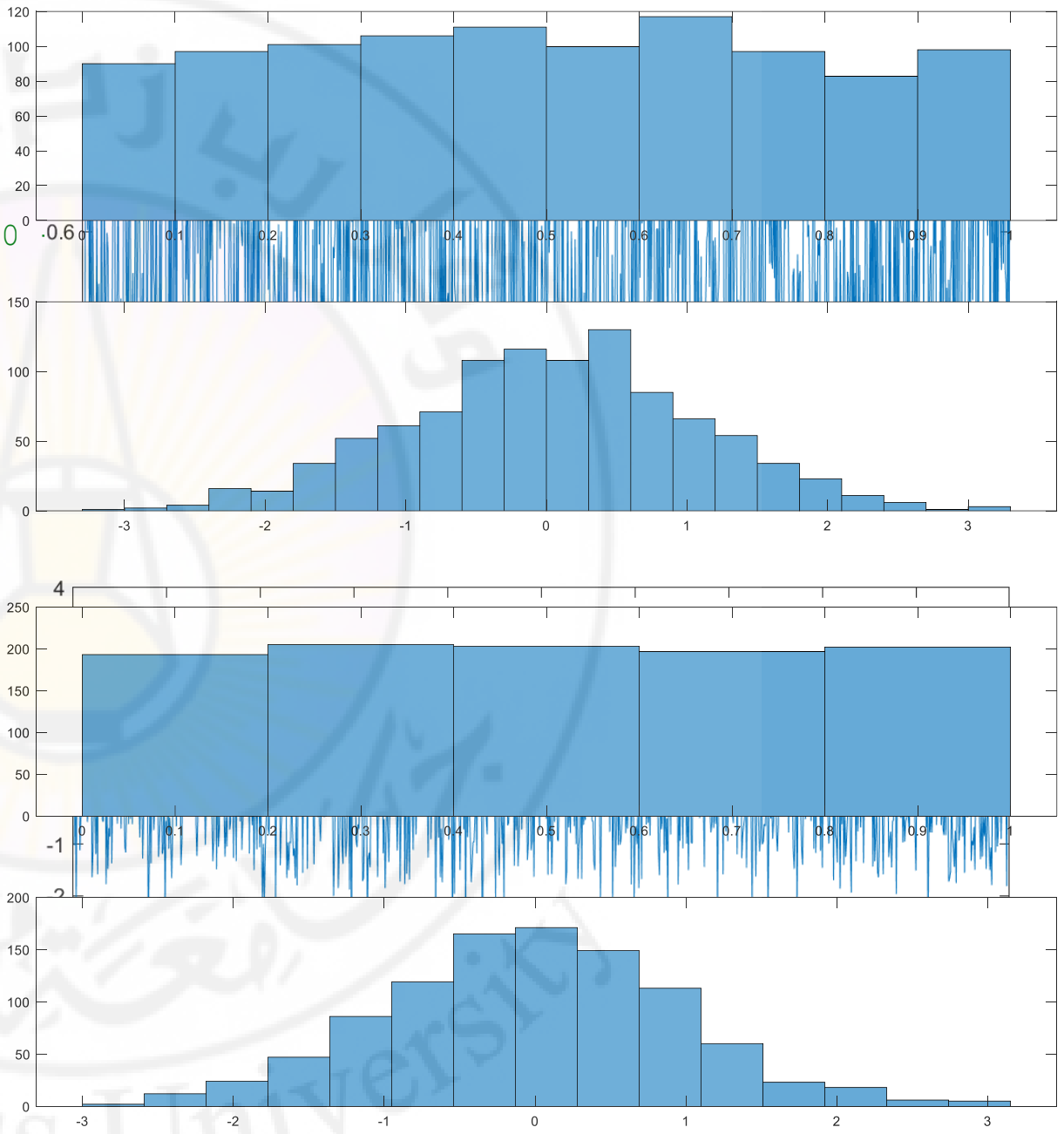
```
% rand: Uniformly distributed pseudorandom numbers 0
min(rand(1, 1e6))
max(rand(1, 1e6))

% randn: Normally distributed pseudorandom numbers
max(randn(1, 1e6))
min(randn(1, 1e6))

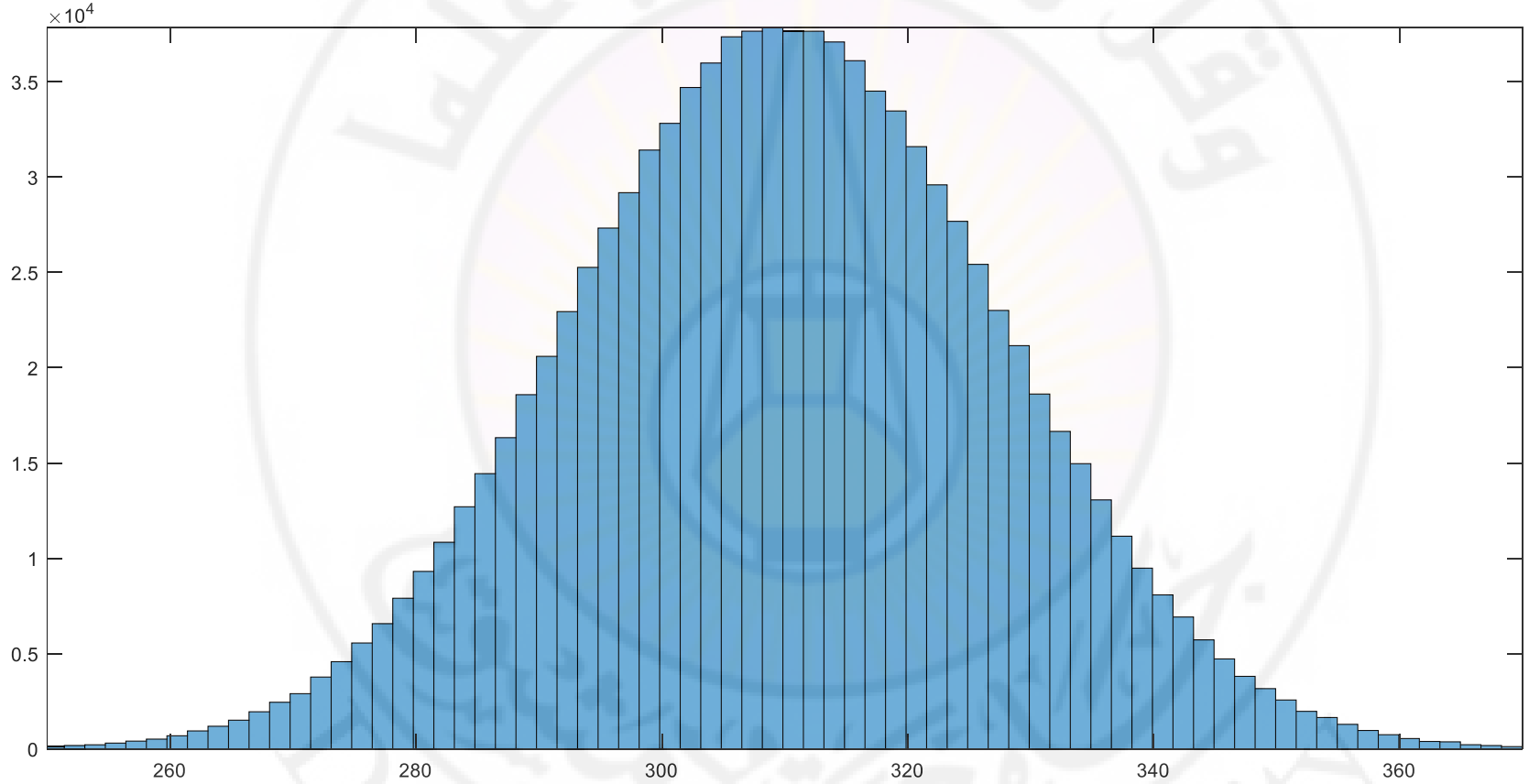
plot(rand(1000,1))
plot(randn(1000,1))

histogram(rand(1000,1))
histogram(randn(1000,1))

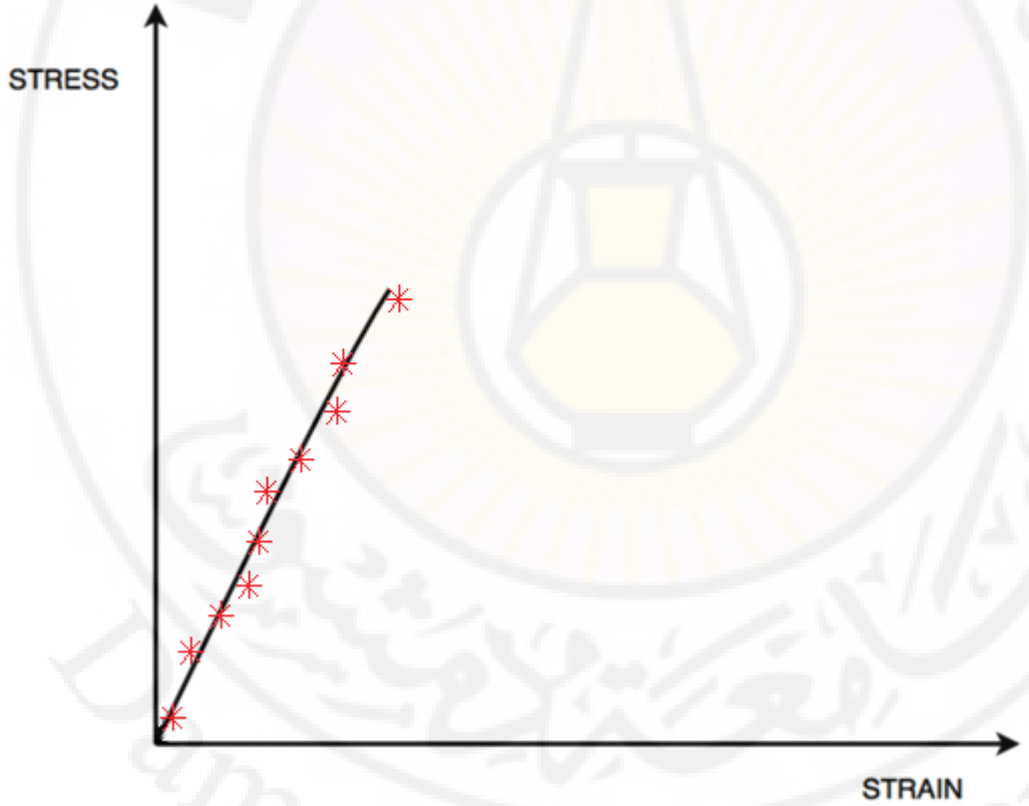
histogram(rand(1000,1), 5) % change bins number
histogram(randn(1000,1), 15)
```



S275

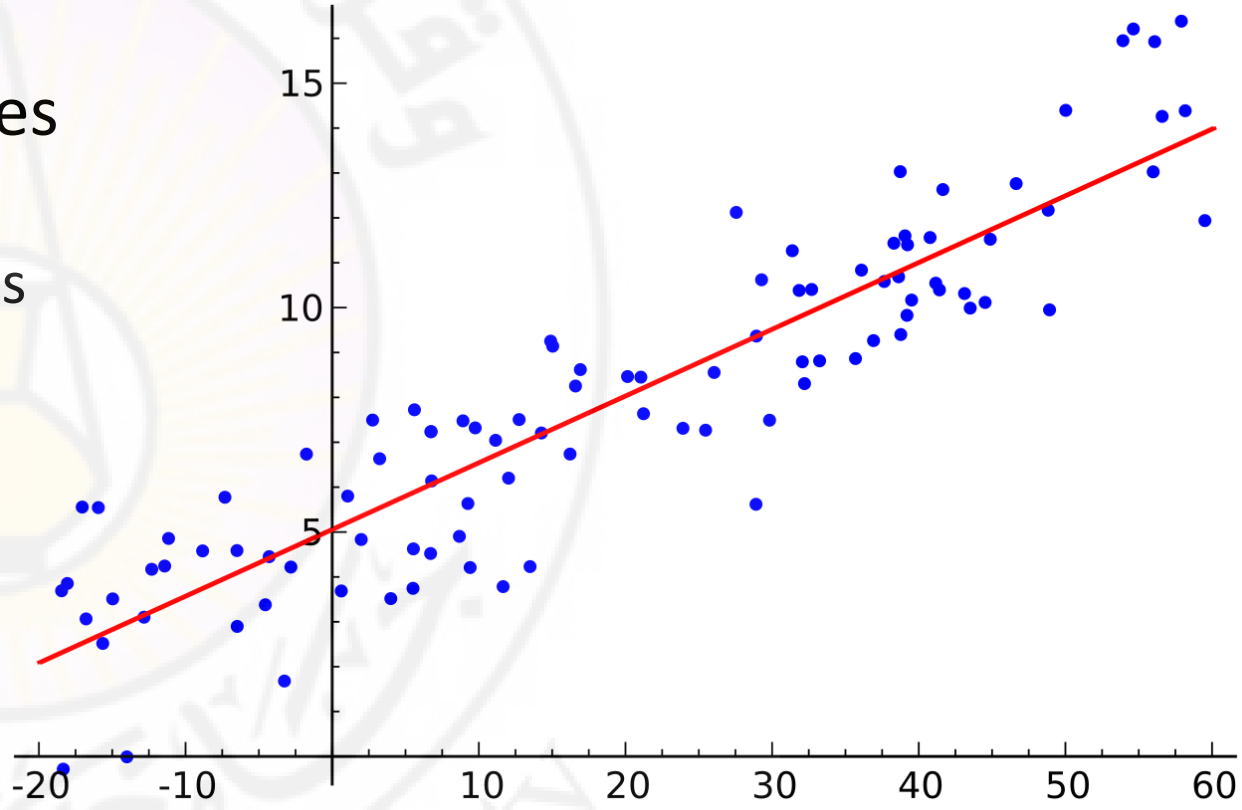


Curve Fitting



Regression Analysis

- Find relationships between variables
  - Linear regression
  - minimizes sum of squared differences between true data and that line
- Exponential, power, sine...
- Single vs multiple regression



Regression Analysis

- n points

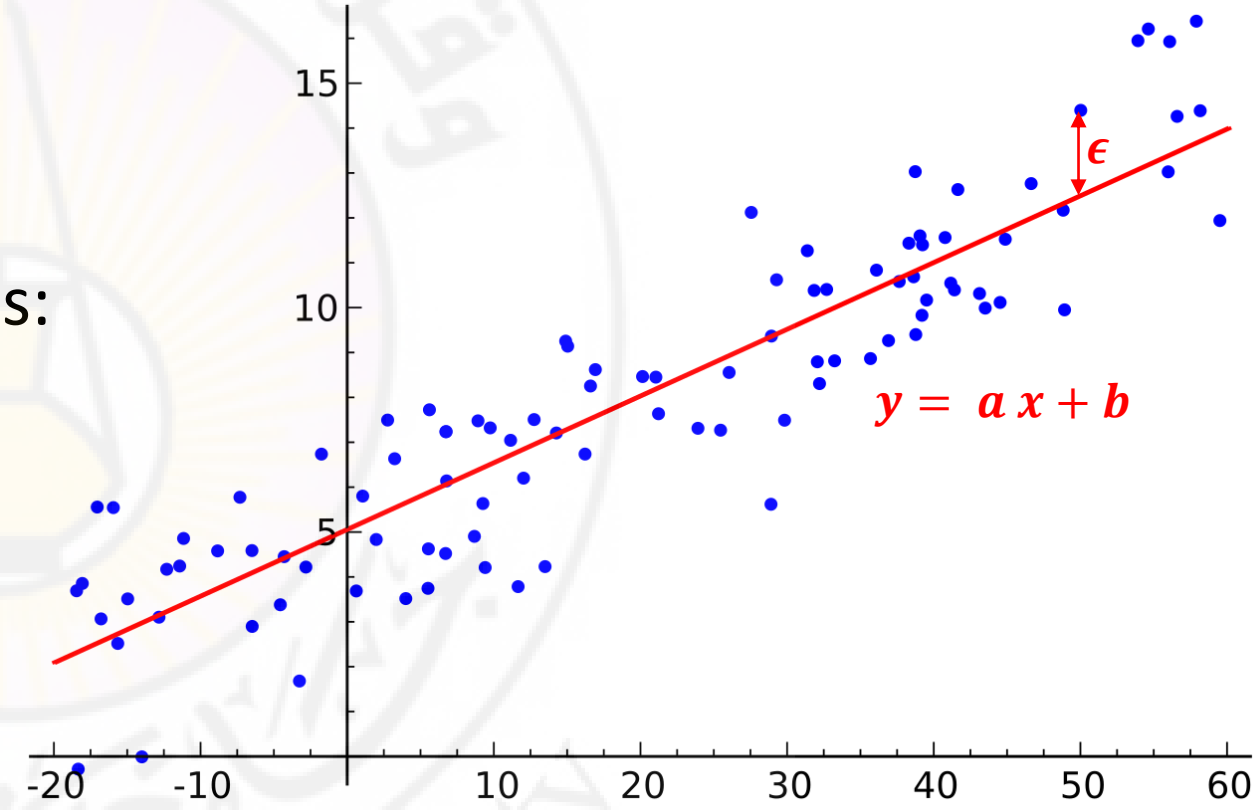
  $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$

- for linear regression each point has:

$$y = a\,x + b$$
$$z = a\,x + by + c$$

Due to error:

$$y = \beta_0 + \beta_1\,x + \varepsilon$$



Regression Analysis

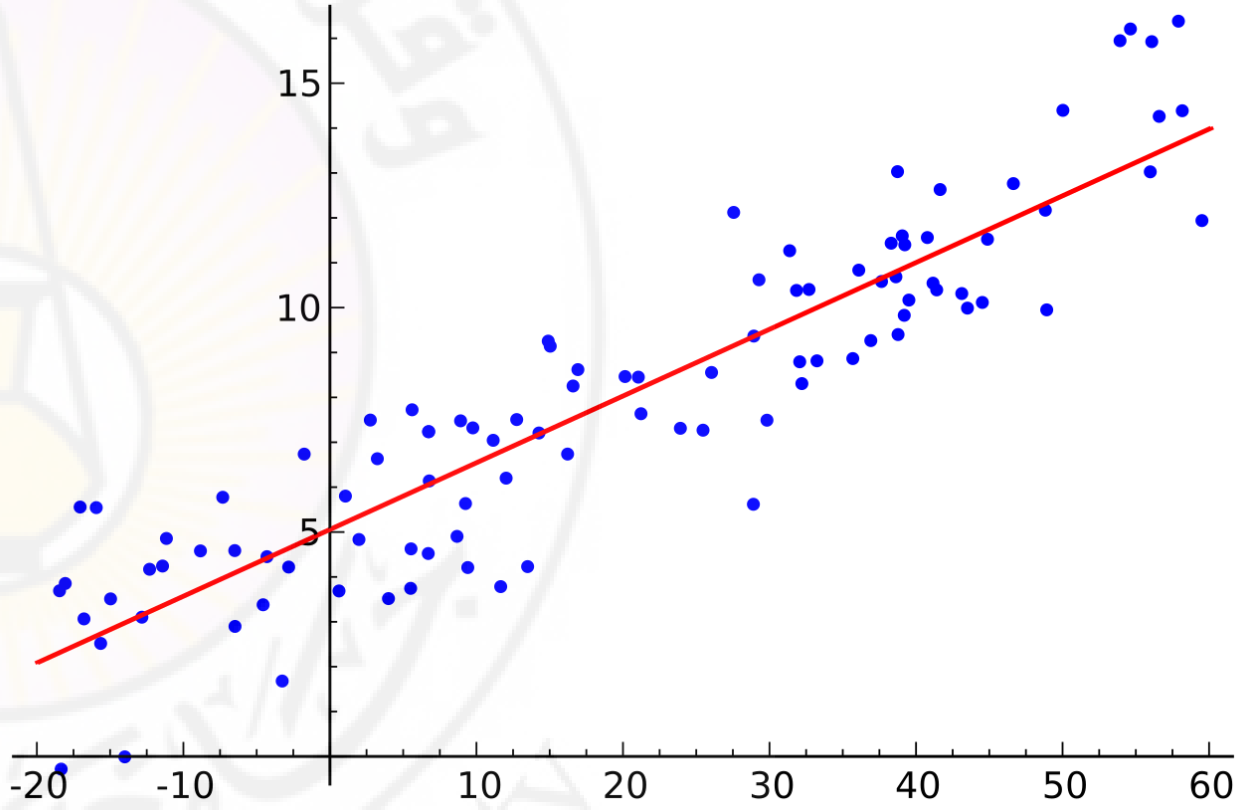
For n points:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \quad \mathrm{x} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

$$y = x\beta + \varepsilon$$

Find β to minimize n errors $\varepsilon_1, \varepsilon_2 \ldots \varepsilon_n$

$$\varepsilon_i = y_i - x_i\beta$$



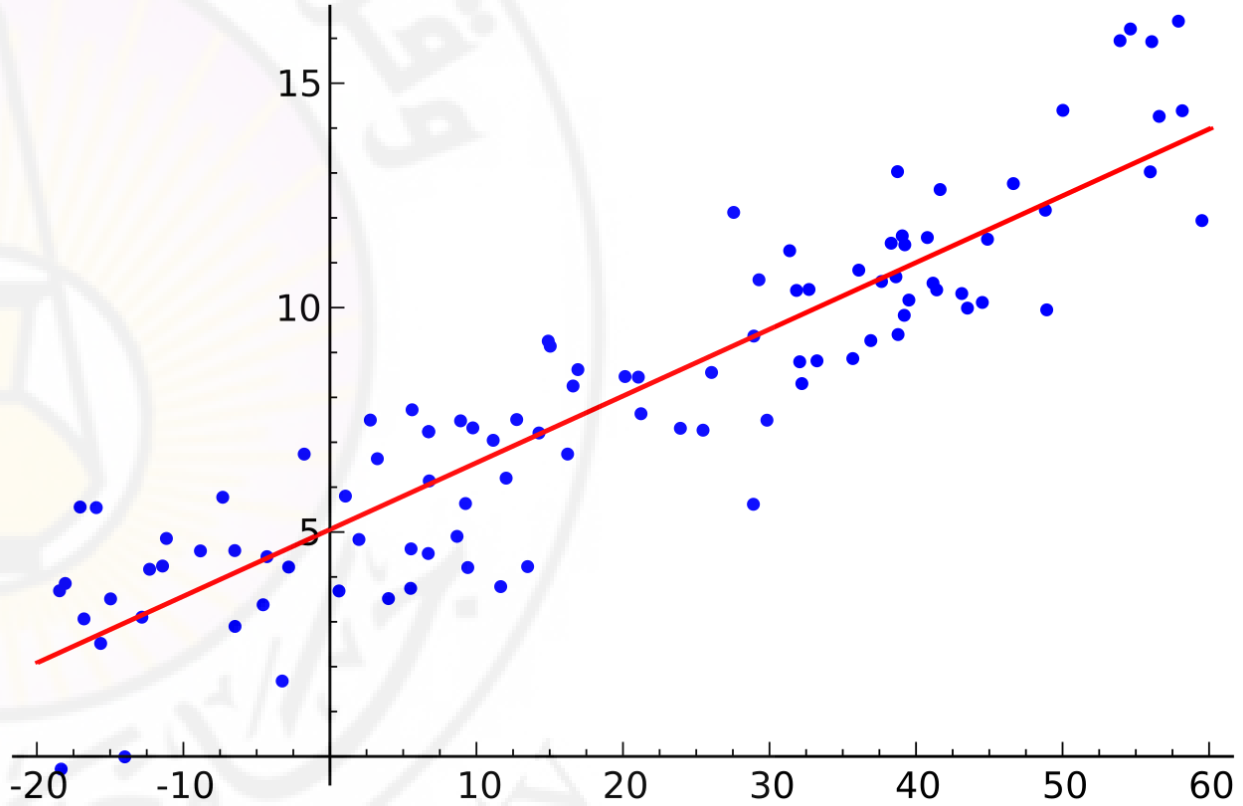
Least Squares Method

$$\varepsilon_i = y_i - x_i\beta$$

- Minimize all errors
- Minimize both +tive and –tive errors
- Penalize larger errors more
- Minimize sum of squares

$$error = \sum_{i=1}^{n} \varepsilon_i^2$$

$$= \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$



Least Squares Method

Minimize error:

$$error = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 = \sum_{i=1}^{n} y_i^2 + (\beta_0^2 + \beta_1^2 {x_i}^2 + 2\beta_0\beta_1 x_i) - 2(y_i\beta_0 + y_i x_i \beta_1)$$

$$= \sum_{i=1}^{n} y_i^2 + \beta_0^2 + \beta_1^2 {x_i}^2 + 2\beta_0\beta_1 x_i - 2y_i\beta_0 - 2y_i x_i \beta_1$$

Least Squares Method

Minimize error:

$$error = \sum_{i=1}^{n} y_i^2 + \beta_0^2 + \beta_1^2 {x_i}^2 + 2\beta_0\beta_1 x_i - 2y_i\beta_0 - 2y_i x_i\beta_1$$

$$\frac{\partial_{error}}{\partial_{\beta_0}} = \sum_{i=1}^{n} 2\beta_0 + 2\beta_1 x_i - 2y_i = 2\sum_{i=1}^{n} \beta_0 + \beta_1 x_i - y_i$$

$$\frac{\partial_{error}}{\partial_{\beta_1}} = \sum_{i=1}^{n} 2\beta_1 {x_i}^2 + 2\beta_0 x_i - 2y_i x_i = 2\sum_{i=1}^{n} x_i(\beta_1 x_i + \beta_0 - y_i)$$

Least Squares Method

$$\frac{\partial_{error}}{\partial_{\beta_0}} = 2\sum_{i=1}^{n} \beta_0 + \beta_1 x_i - y_i = 0 \qquad \sum_{i=1}^{n} \beta_0 + \sum_{i=1}^{n} \beta_1 x_i = \sum_{i=1}^{n} y_i \qquad \beta_1 \sum_{i=1}^{n} x_i + n\beta_0 = \sum_{i=1}^{n} y_i$$

$$\frac{\partial_{error}}{\partial_{\beta_1}} = 2\sum_{i=1}^{n} x_i(\beta_1 x_i + \beta_0 - y_i) = 0 \qquad \beta_1 \sum_{i=1}^{n} {x_i}^2 + \beta_0 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i$$

Least Squares Method

$$n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$\beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i y_i$$

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

Higher Order

- Higher order (j):

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ \ldots + \beta_j x^j + \ \varepsilon$$

$$y = \sum_{k=0}^{j} \beta_k \, x^k \ + \ \varepsilon$$

$$error = \ \sum_{i=1}^{n} \varepsilon_i^2$$

$$= \ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

$$= \ \sum_{i=1}^{n} (y_i - \sum \beta_k x_i^k)^2$$

Higher Order

$$\begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i{}^2 \end{bmatrix}^{-1} \begin{bmatrix}\sum y_i \\ \sum x_i y_i\end{bmatrix}$$

$$\begin{bmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \vdots\\ \beta_j\end{bmatrix} = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \dots & \sum x_i^j \\ \sum x_i & \sum x_i^2 & \sum x_i & \dots & \sum x_i^{j+1} \\ \sum x_i^2 & \sum x_i^2 & \sum x_i^2 & \dots & \sum x_i^{j+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_i^j & \sum x_i^{j+1} & \sum x_i^{j+2} & \dots & \sum x_i^{j+j} \end{bmatrix}^{-1} \begin{bmatrix}\sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \\ \vdots \\ \sum x_i^j\, y_i\end{bmatrix}$$

Example

```
a = 1.2; b = 2;
points = 10;
x = linspace(0, 100, points); y = a * x + b + rand (1, points)*20;

plot(x, y, 'r:*')
scatter(x, y, 'b')

n = length(x);
sigma_x = sum(x); sigma_x2 = sum(x.^2);
sigma_y = sum(y); sigma_xy = sum(x.*y);

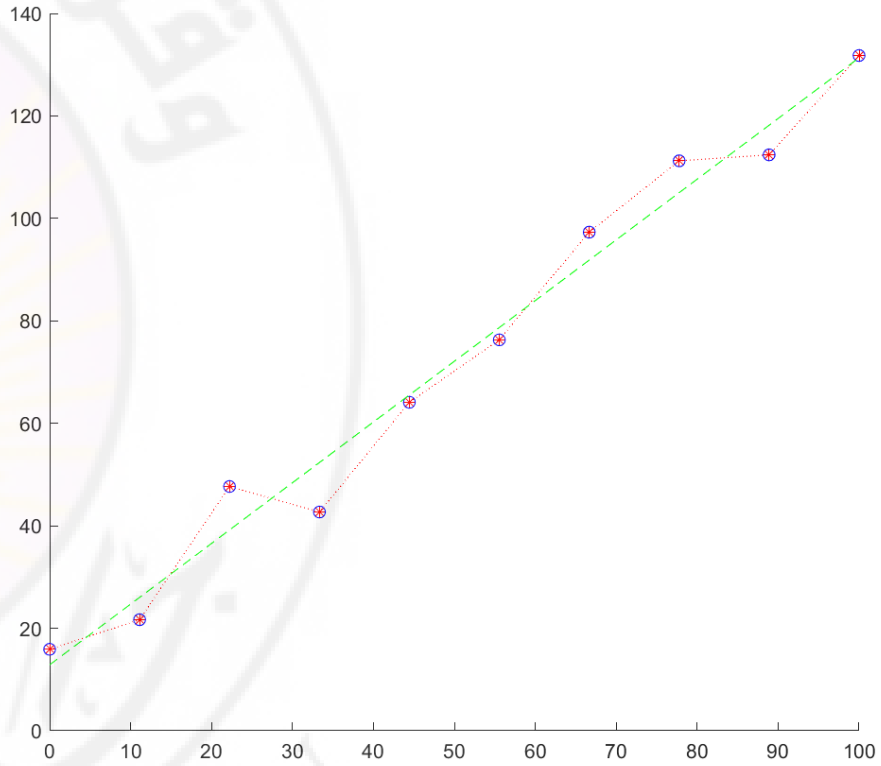
Beta = [n sigma_x; sigma_x sigma_x2] \ [sigma_y; sigma_xy];

p = polyfit(x, y, 1);

y2 = p(1) * x + p(2);
y2 = polyval(p, x);

plot(x, y2, 'g--')
```

$$\begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{bmatrix}^{-1} \begin{bmatrix}\sum y_i \\ \sum x_i y_i\end{bmatrix}$$



Computer **A**ided **D**esign

MATLAB Introduction for Civil Engineers 5

29/11/2022

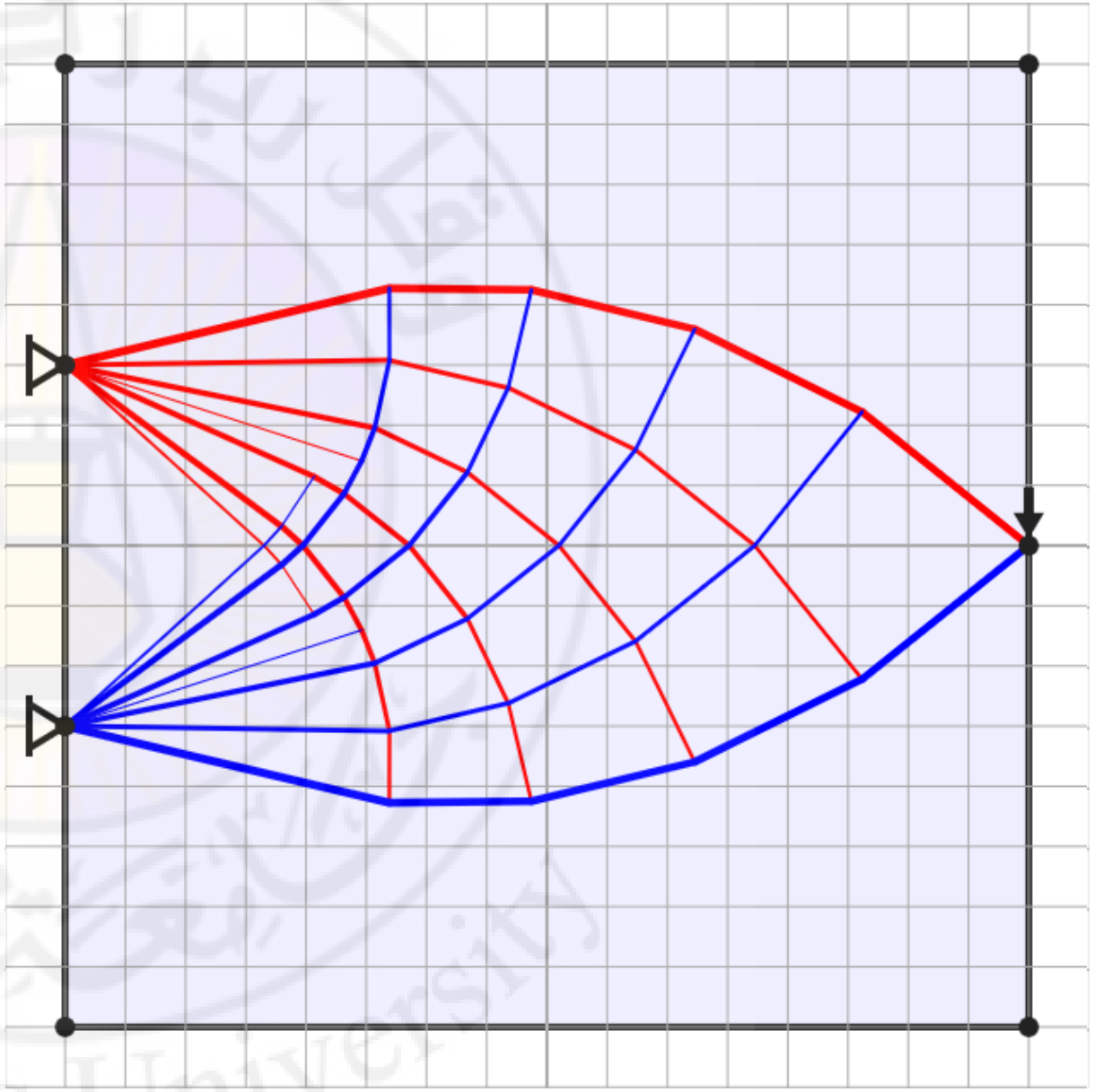
Wael Darwich

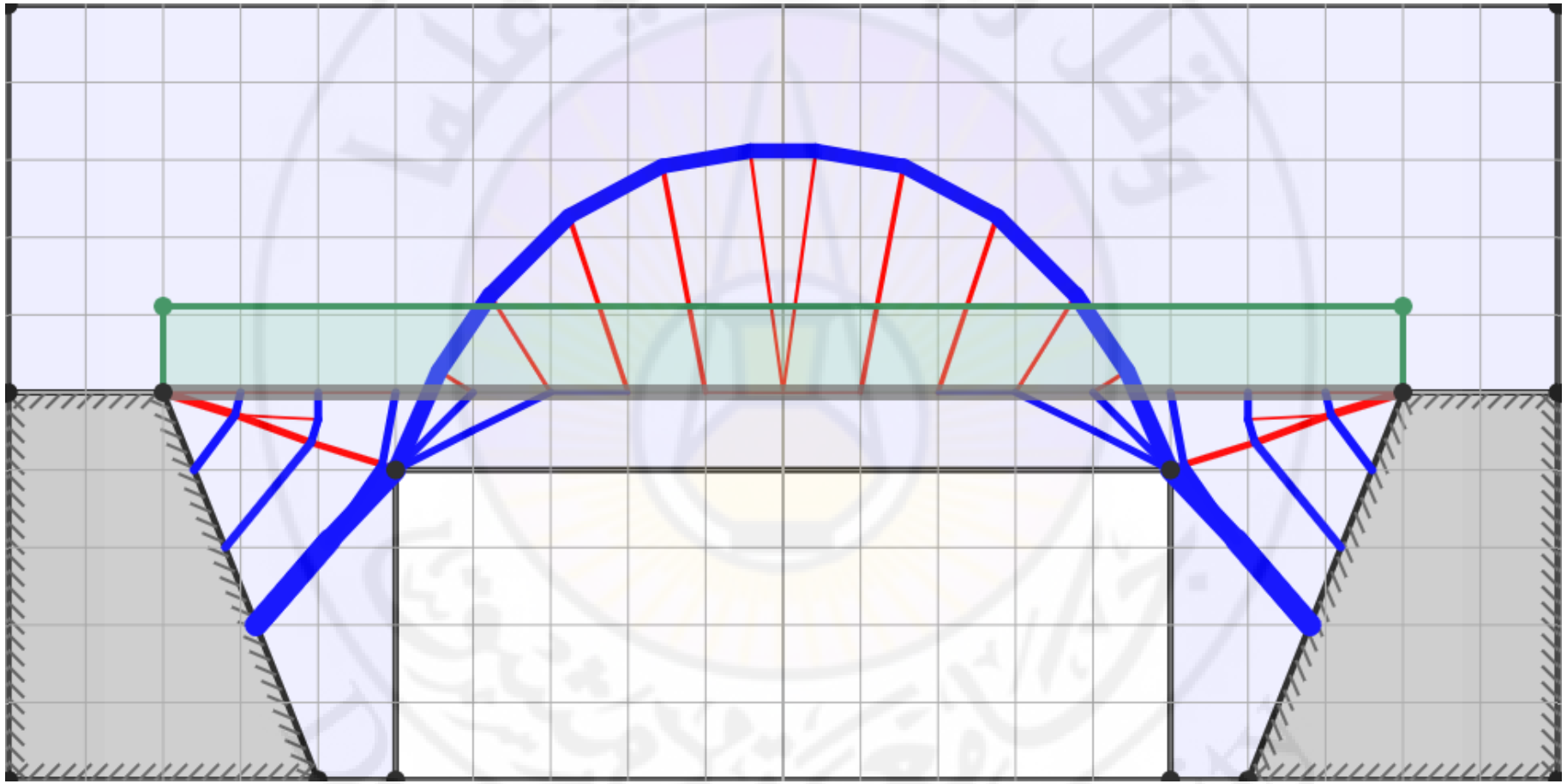
Content

- Optimization
  - What is it
  - Maximize revenue example
- Projectile Motion
  - Analytical solution
  - Numerical solution
- Other Examples
- Layout Optimization
  - Examples

Optimization

- Optimum design
- Minimize material
- Truss example
- Minimize cost / maximize gain
- Minimize build time





Mathematical Optimization

- Mathematical programming
- Objective function
  - E.g. most efficient design
  - Maximization, minimization
- Variables
  - Continuous, e.g. geometry
  - Discrete, e.g. section or material selection (integer, binary)
- Constraints

Maximize Revenue

- A company is studying the pricing of a product.
- The expected number of people $n(p)$ that would buy the product is linearly decreasing with the product price p.
  - If the price is p_0, then nobody would buy it.
  - If the product was free, then a number n_0 of people would want it.
- What should be the product price to maximize total revenue?

Maximize Revenue

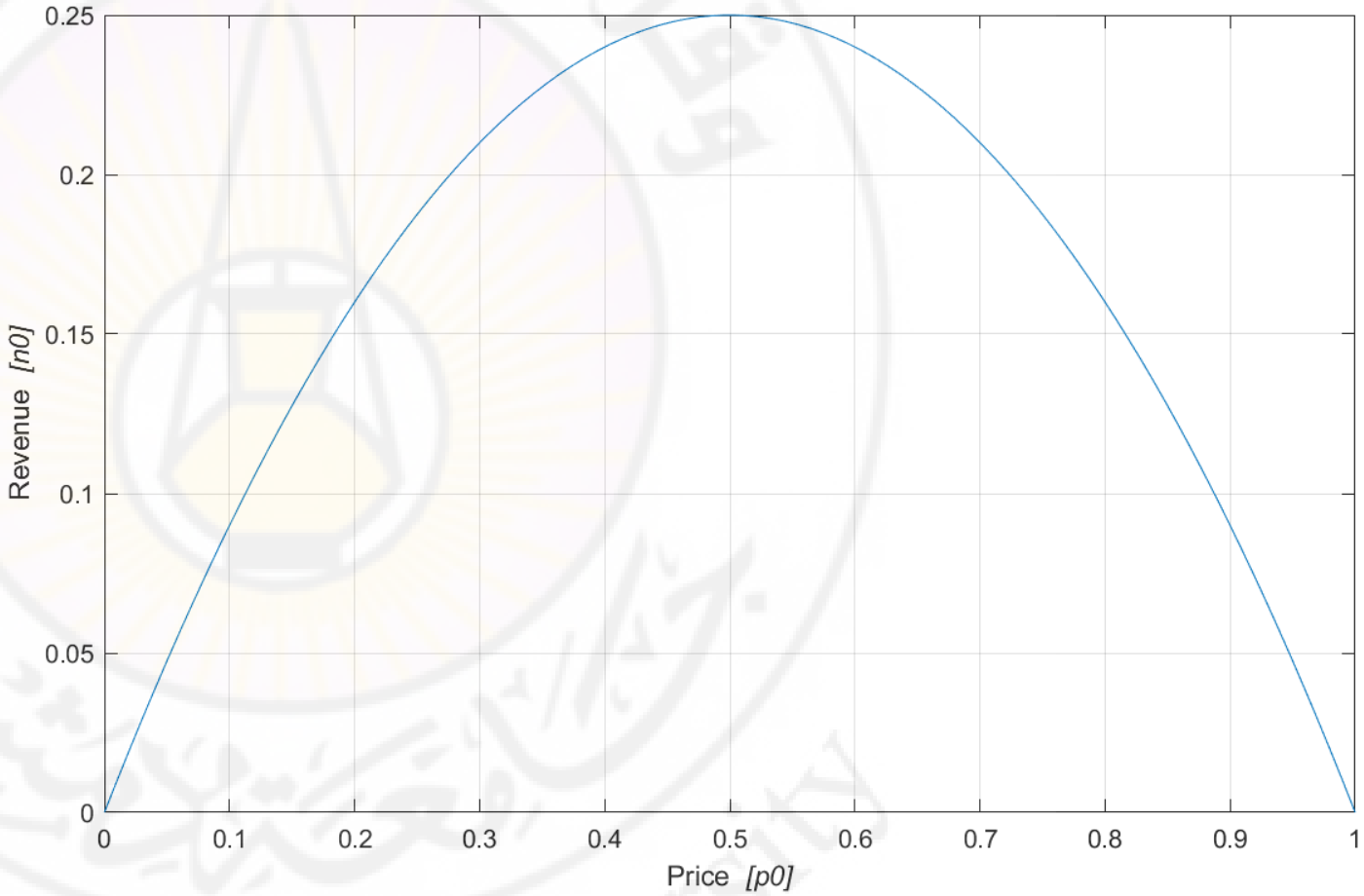
- Revenue = unit price × number of units
  - Unit price p (variable)
  - Number of units :
    - $n = 0$ at $p = p_0$
    - $n = n_0$ at $p = 0$
- p in the range $0 - p_0$
- n in the range $n_0 - 0$
- Revenue = $n \times p$

Numerical Solution

```
price = linspace(0, 1);
quantity = linspace(1, 0);

total = price .* quantity;

clf
plot(price, total)
grid on
xlabel('Price {\it [p0]}')
ylabel(Revenue {\it [n0]}')
```



Analytical Solution

- Revenue = unit price × number of units
  - Unit price p (variable)
  - Number of units :
    - $n = 0$ at $p = p_0$
    - $n = n_0$ at $p = 0$
  - $n = n_0 - p\,\frac{n_0}{p_0}$
- Revenue = $n \times p$

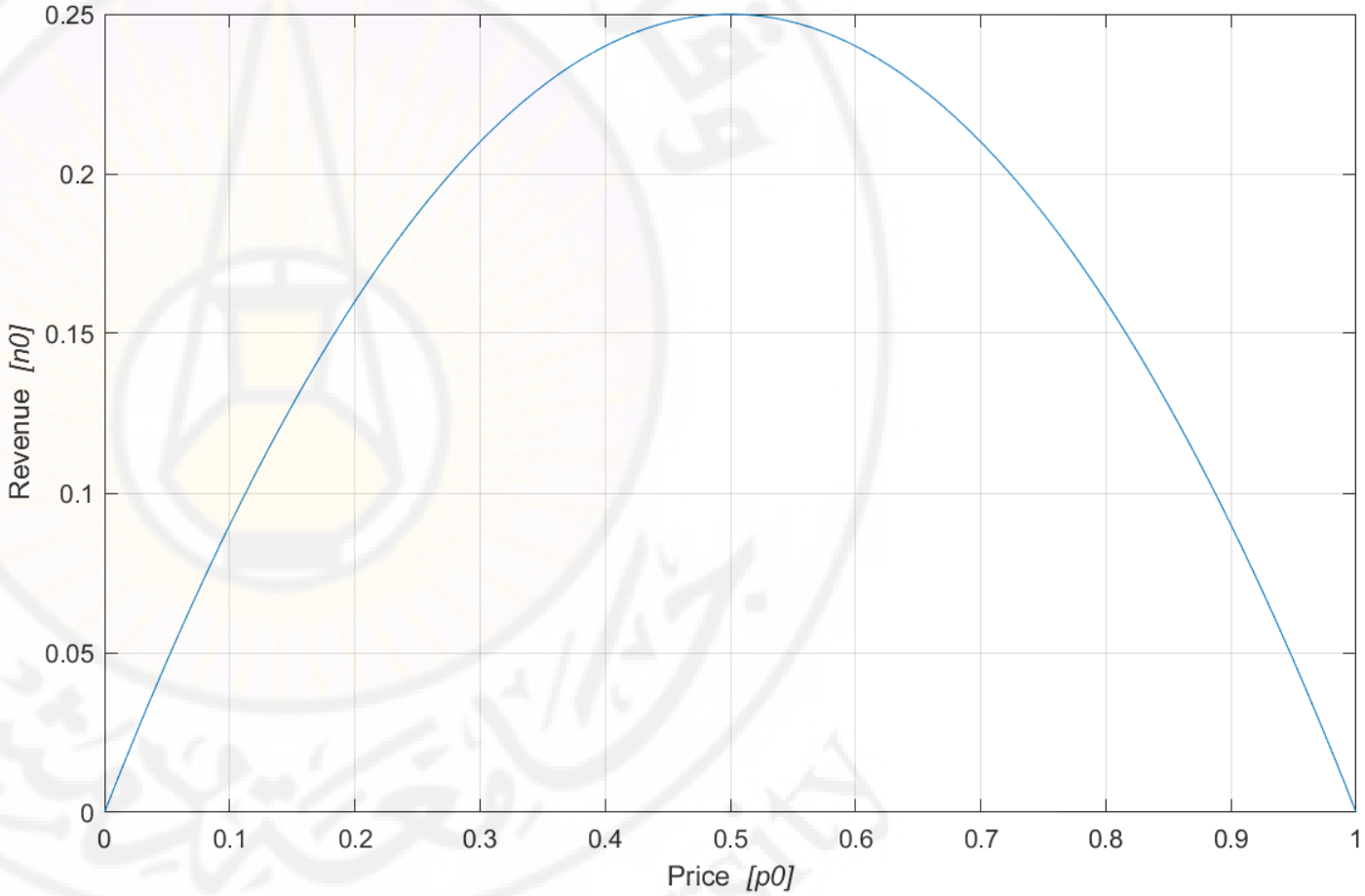
$$= p.n_0 - p^2\,\frac{n_0}{p_0}$$

Analytical Solution

- Revenue = $n \times p$

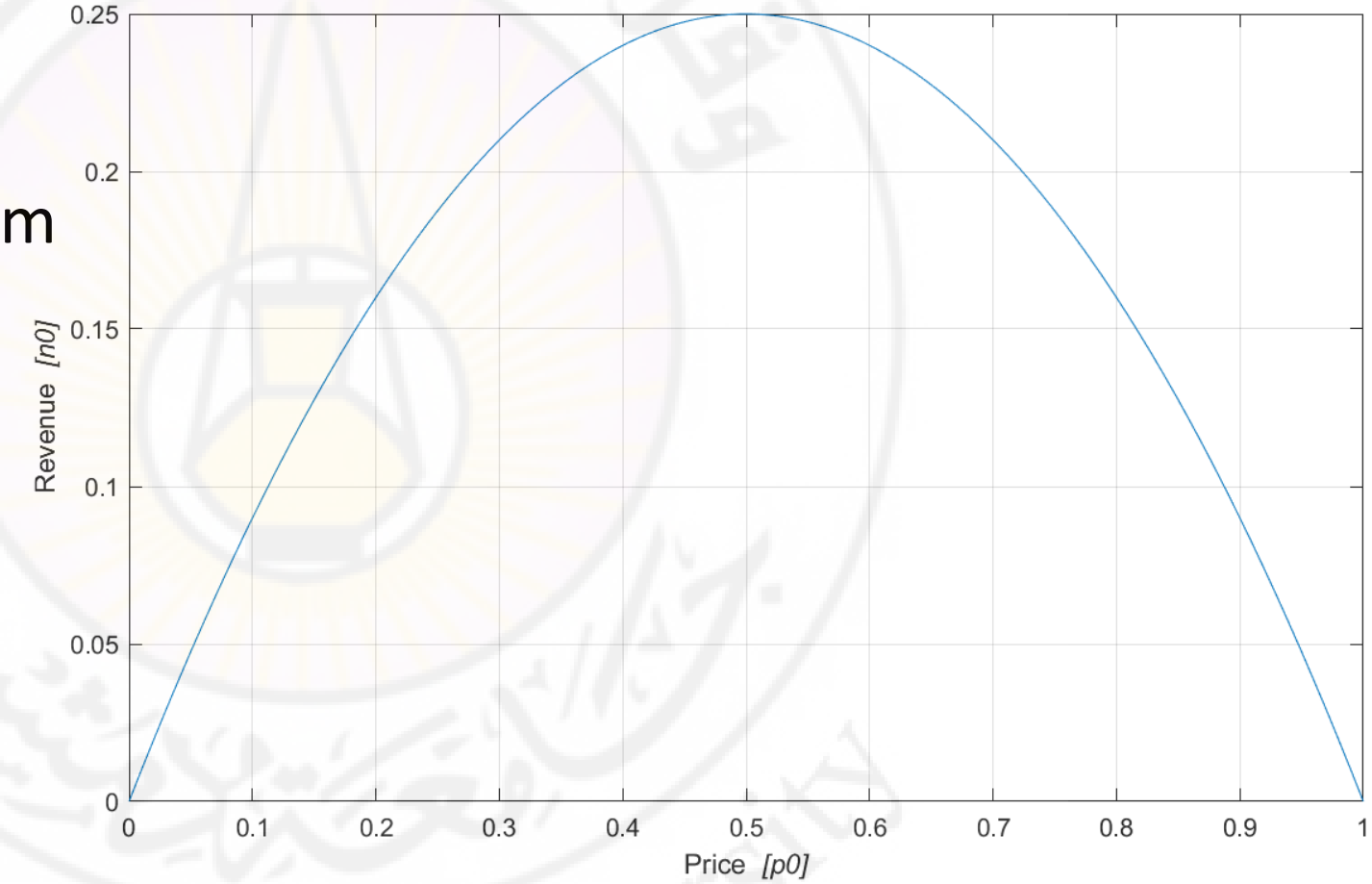
$$f(p) = p.n_0 - p^2 \frac{n_0}{p_0}$$

- $f' = n_0 - 2.\frac{n_0}{p_0}.p$
- $f' = 0$
- $p = n_0 . 0.5 . \frac{p_0}{n_0} = \frac{p_0}{2}$



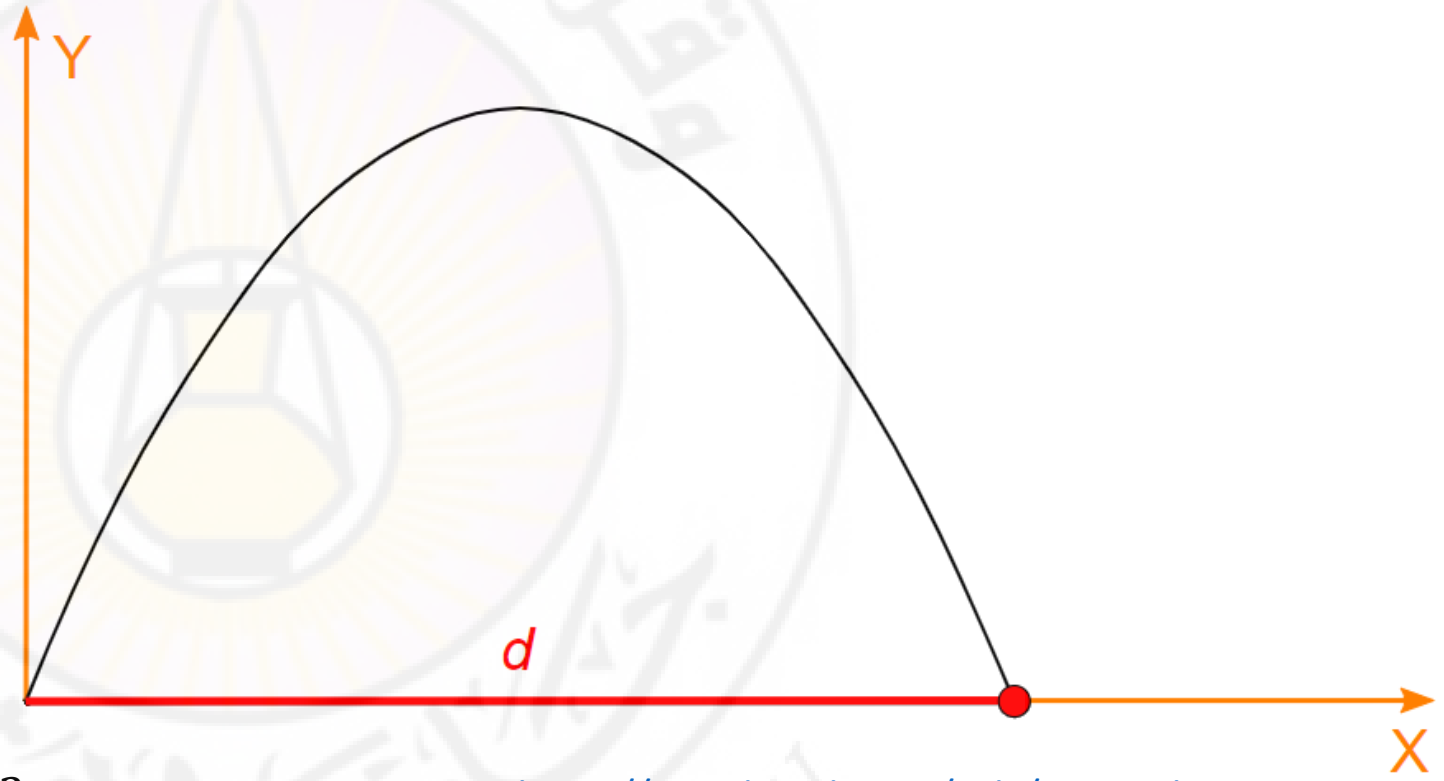
Analytical Solution

- $f(x)$
- $f'(x)$ maximum/minimum
- $f''(x)$:
  - < 0 maximum
  - > 0 minimum
  - = inflection



Projectile Motion

- Acceleration
  - $a_x = 0$
  - $a_y = -g$
- Velocity
  - $v_{0x} = v_0.\cos(\theta)$
  - $v_{0x} = v_0.\sin(\theta)$
- Displacement
  - $x = v_0.t.\cos(\theta)$
  - $y = v_0.t.\sin(\theta) - 0.5.g.t^2$
- Find maximum horizontal displacement



https://en.wikipedia.org/wiki/Projectile_motion

Projectile Motion

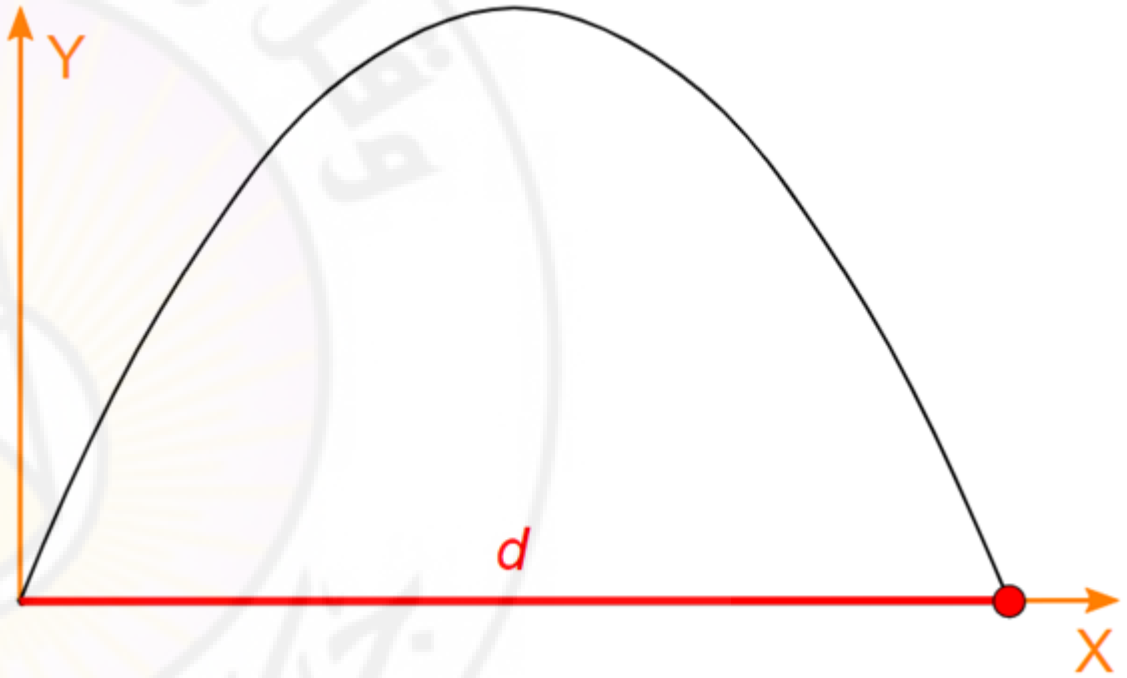
$$x = v_0.t.\cos(\theta)$$
$$y = v_0.t.\sin(\theta) - 0.5.g.t^2$$
$$y = 0$$
$$0.5.g.t^2 = v_0.t.\sin(\theta)$$

$$t = 2.v_0.\frac{\sin(\theta)}{g}$$

$$d = v_0.t.\cos(\theta)$$

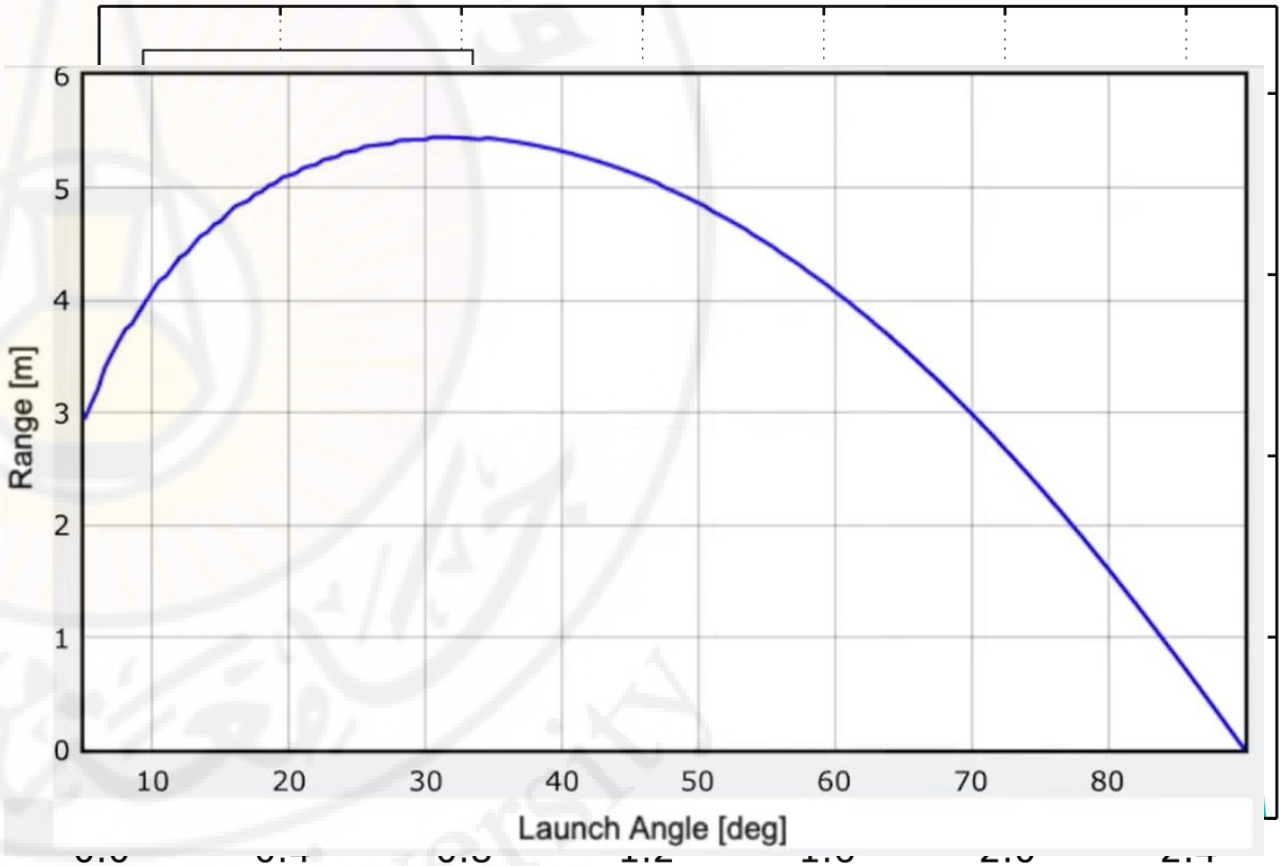
$$d = v_0.2.v_0.\frac{\sin(\theta)}{g}.\cos(\theta)$$

$$d = \frac{v_0^2}{g}\sin(2\theta)$$
$$\sin(2\theta) = 1 \Rightarrow \theta = 45°$$



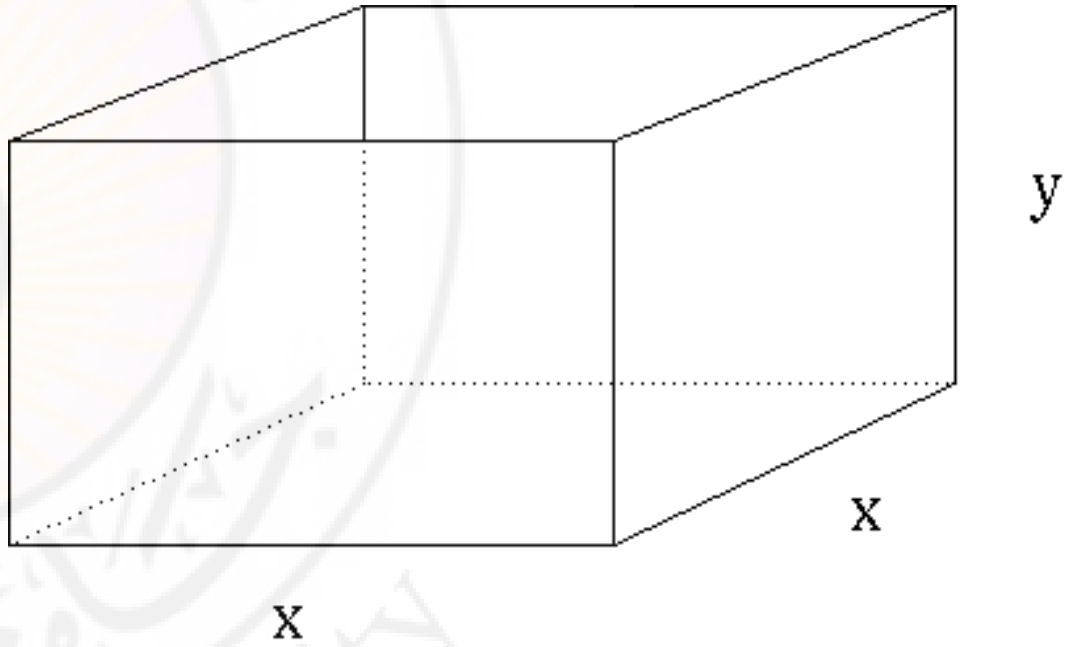
Projectile Motion

- Include air resistance
- drag force proportional to the speed squared, or higher order
- Analytical solution
- Numerical solution



Examples

- An open rectangular box
- Square base
- To be made from 48 m^2 of material
- What dimensions will result in a box with the largest possible volume?



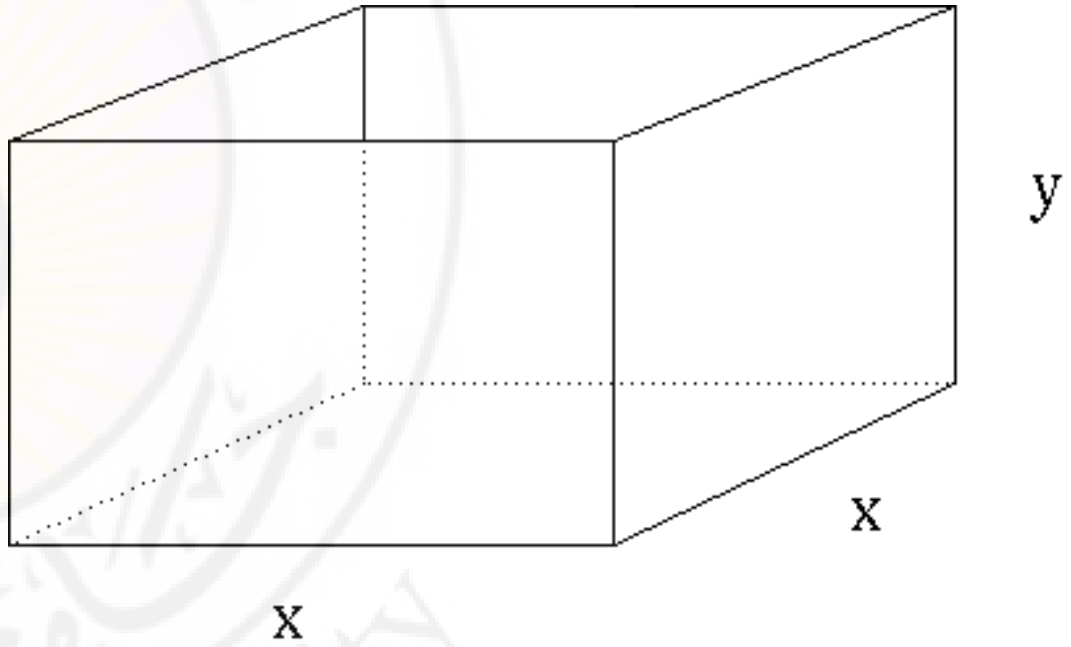
Examples

- Variables x and y
- Objective V(x, y)
- Constraint Area = 48

$$x^2 + 4.x.y = 48$$

$$y = \frac{48 - x^2}{4.x}$$

$$y = \frac{12}{x} - \frac{x}{4}$$



Examples

$$Volme = x^2 \times y$$

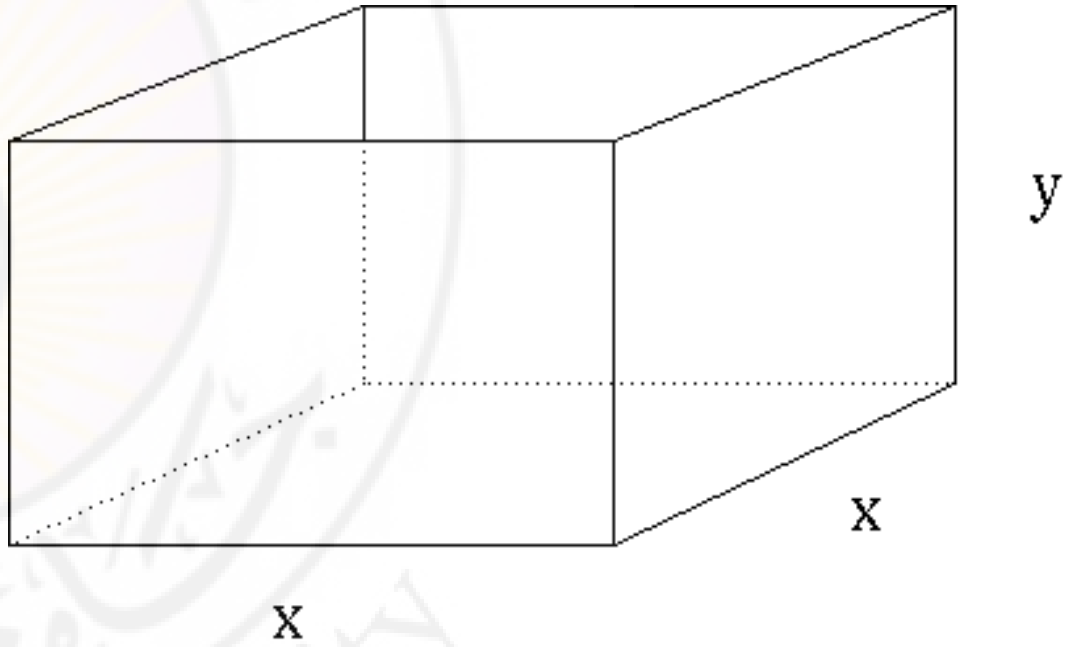
$$= x^2 \times \left(\frac{12}{x} - \frac{x}{4}\right)$$

$$= 12x\ - 0.25x^3$$

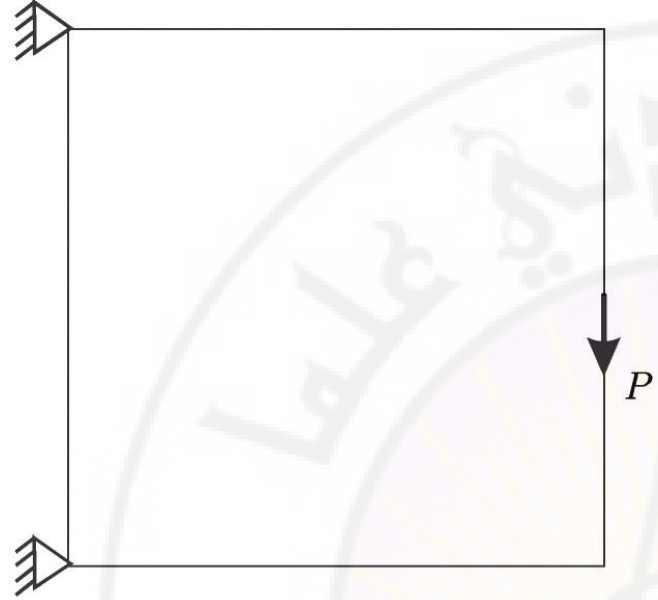
$$Volme' = 12\ - 0.75x^2 = 0$$

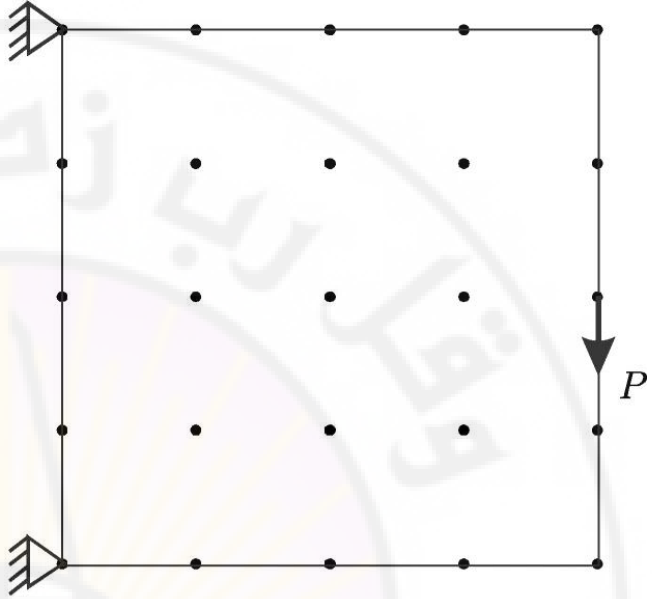
$$x^2 = 16$$

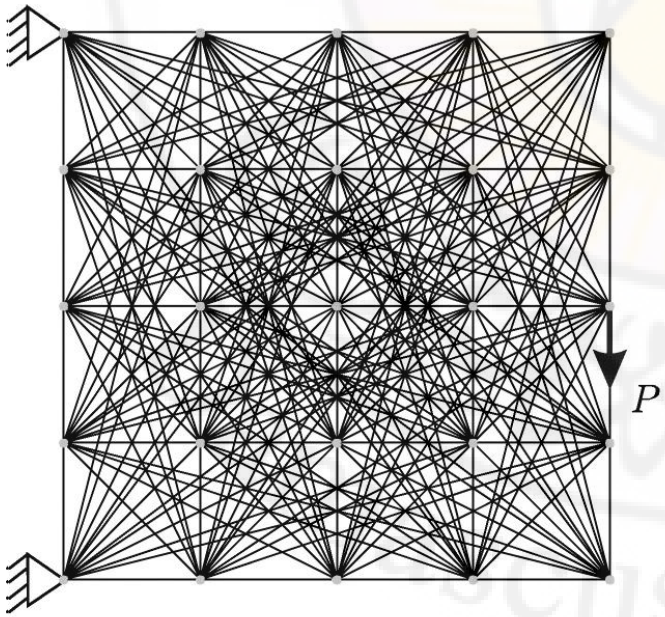
$$x = 4, \qquad V = 32$$

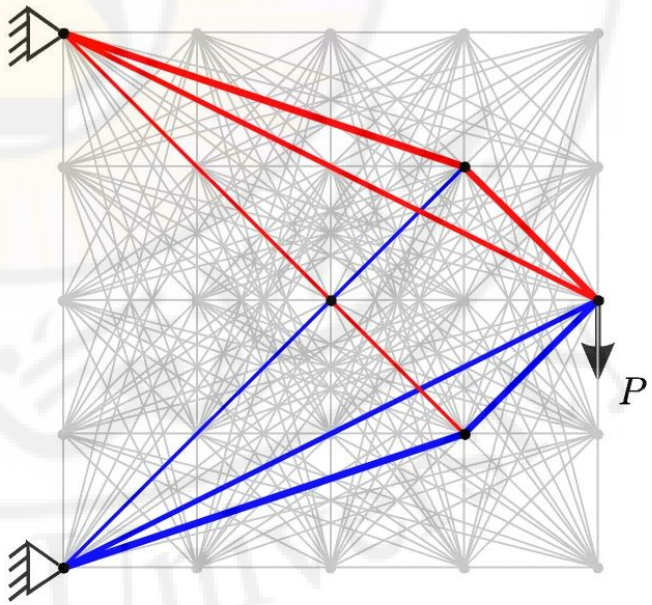


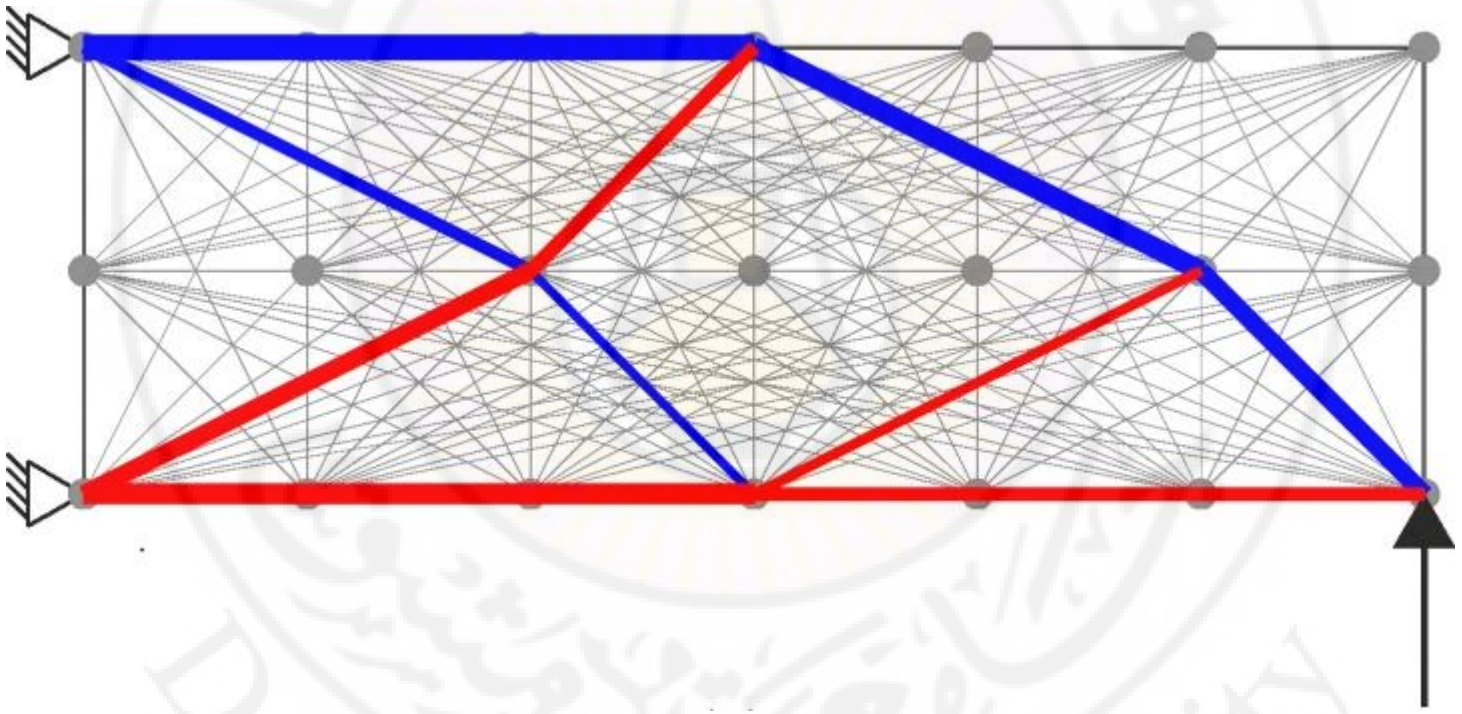
Layout Optimization


P


P


P


P

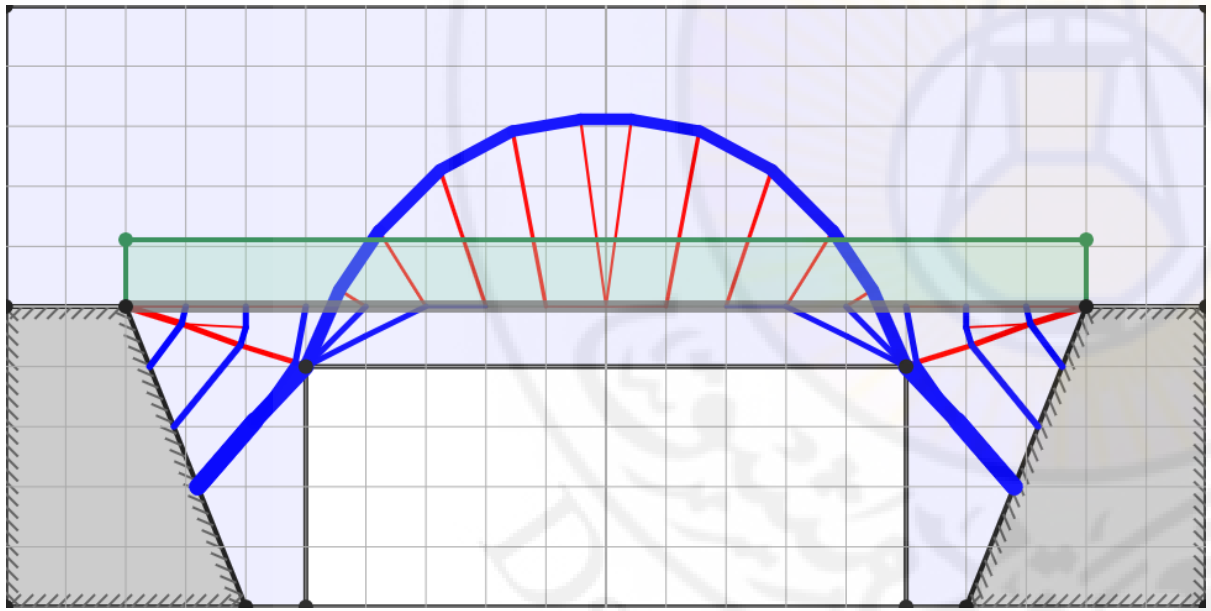


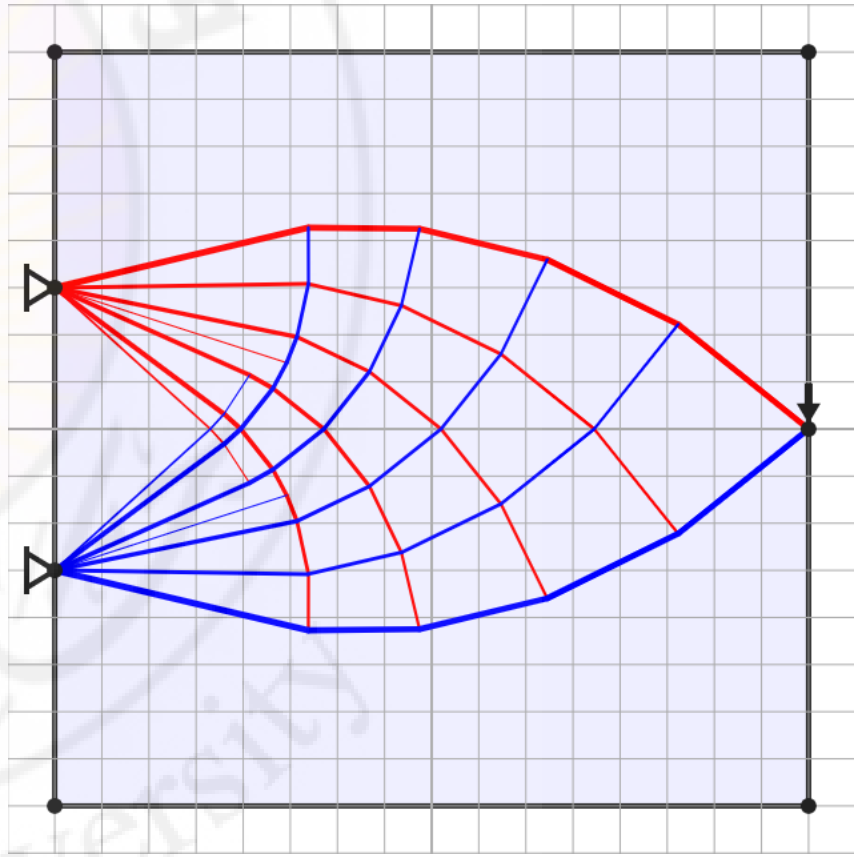
Layout Optimization

- Objective
  - Minimize Volume
- Variables
  - Area of truss members
- Constraints
  - Sum of all forces at each node = 0

Layout Optimization

https://layopt.com/





Computer Aided Design

MATLAB Introduction for Civil Engineers 6

6/12/2022

Wael Darwich

Content

- Mathematical Programming
  - Linear Programming
- Optimization Examples

Mathematical Optimization

- Mathematical programming
- Variables
  - Continuous, e.g. geometry
  - Discrete, e.g. section or material selection (integer, binary)
- Constraints
  - Linear, non linear
  - Objective function
    - Maximization, minimization
- Linear programming

Linear Programming

- Variables (n):

$$x_1, x_2, \cdots, x_n$$

- Minimize:

$$f(x_1, x_2, \cdots, x_n) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

- Constraints (m):

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

$$x_1, x_2, \cdots, x_n \geq 0$$

Linear Programming

- Variables (n):

$$x_1, x_2, \cdots, x_n$$

- Minimize:

$$f(x_1, x_2, \cdots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

- Constraints (m):

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

$$x_1, x_2, \cdots, x_n \geq 0$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

$$a = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Linear Programming

- Variables (n):

$$x$$

- Minimize:

$$f(x) = c^T x$$

- Constraints (m):

$$ax = b$$

$$x \geq 0$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

$$a = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Linear Programming

- Minimization problem
- Maximize

$$f(x) = c^T x$$

Equivalent to minimize

$$-f(x) = -c^T x$$

- Objective value

Equivalent to minimizing error

Linear Programming

- Constraints:
  - Equality:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i$$

  - Inequality constraints:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

Equivalent to $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + x_{n+1} = b_i$

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i$$

Equivalent to $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - x_{n+1} = b_i$

Linear Programming

- Constraints:

$$x \geq 0$$

- For negative variables:

$$x_i = x'_i - x''_i$$

$$x'_i, x''_i \geq 0$$

Linear Programming

- Solution vs no solution
- Single vs multiple solutions
- Bounded vs unbounded solution

Linear Programming

Maximize:

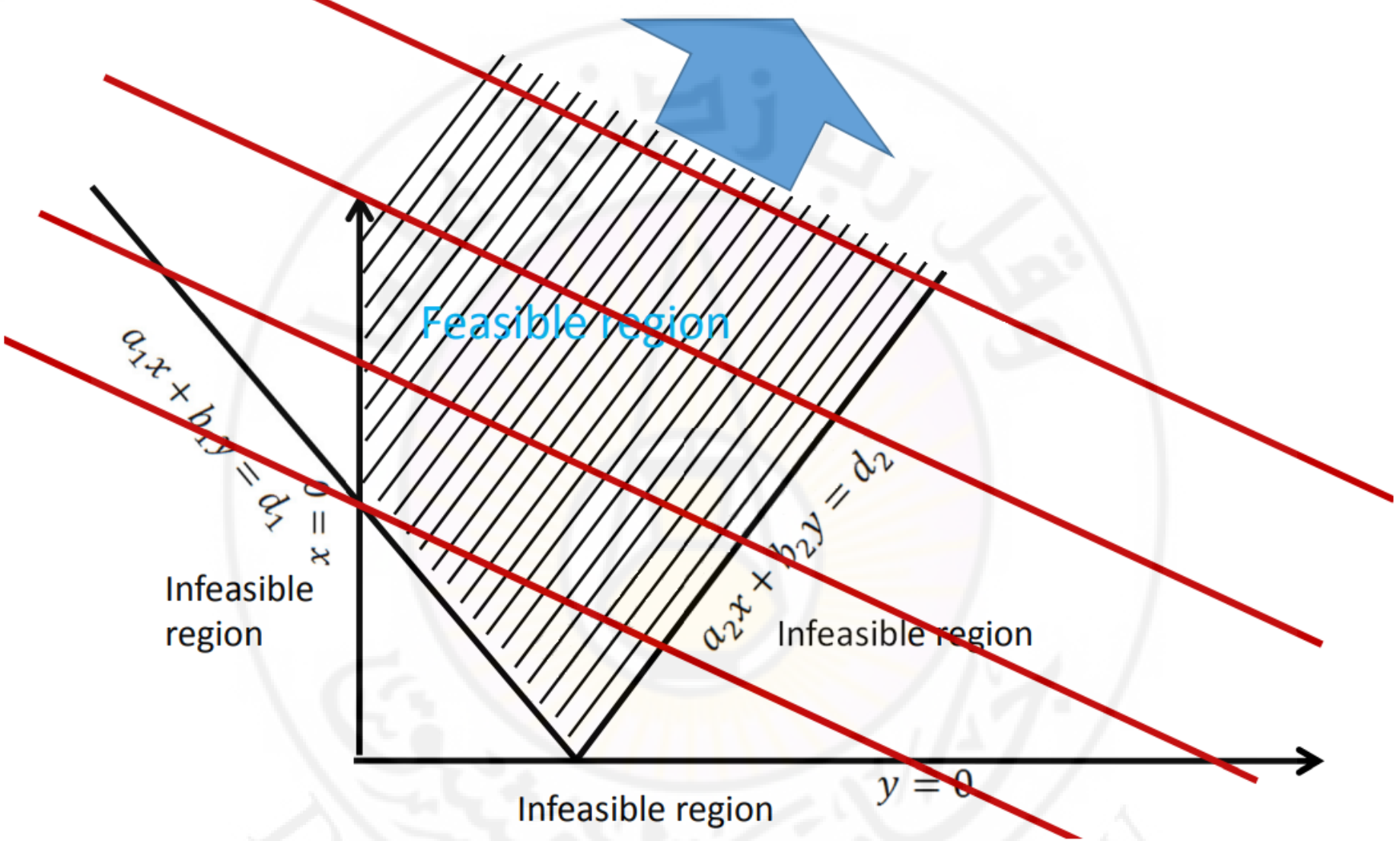
$$z = c_1 x + c_2 y$$

Subject to:

$$a_1 x + b_1 y \leq d_1$$

$$a_2 x + b_2 y \leq d_2$$

$$x, y \geq 0$$


Feasible region
$a_1x + b_1y = d_1$
$x = 0$
Infeasible region
$a_2x + b_2y = d_2$
Infeasible region
$y = 0$
Infeasible region

Mathematical Programming

- Numerical examples

Example 1

- Maximize $f(x,y) = -x^2 + 10.x - y^2 + 8.y$

- Analytically:

$$\frac{\delta f(x,y)}{\delta x} = -2.x + 10 = 0$$

$$\frac{\delta f(x,y)}{\delta y} = -2.y + 8 = 0$$

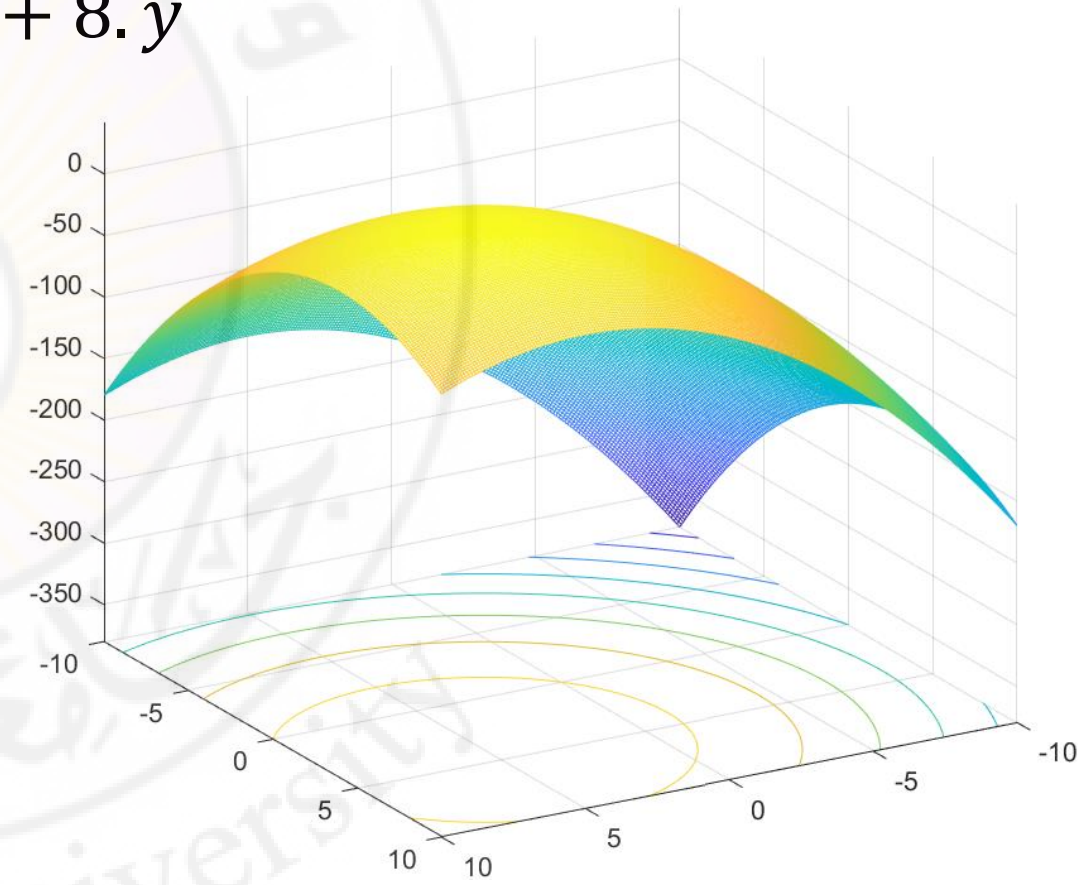
$$x = 5$$
$$y = 4$$
$$f(x,y) = 41$$

Example 1

- Maximize $f(x, y) = -x^2 + 10.x - y^2 + 8.y$

- Numerically:

```
[x, y] = meshgrid(0:0.1:10);
f = -x.^2 + 10*x - y.^2 + 8*y;
meshc(x, y, f)
max(max(f))  % 41
```



Example 1

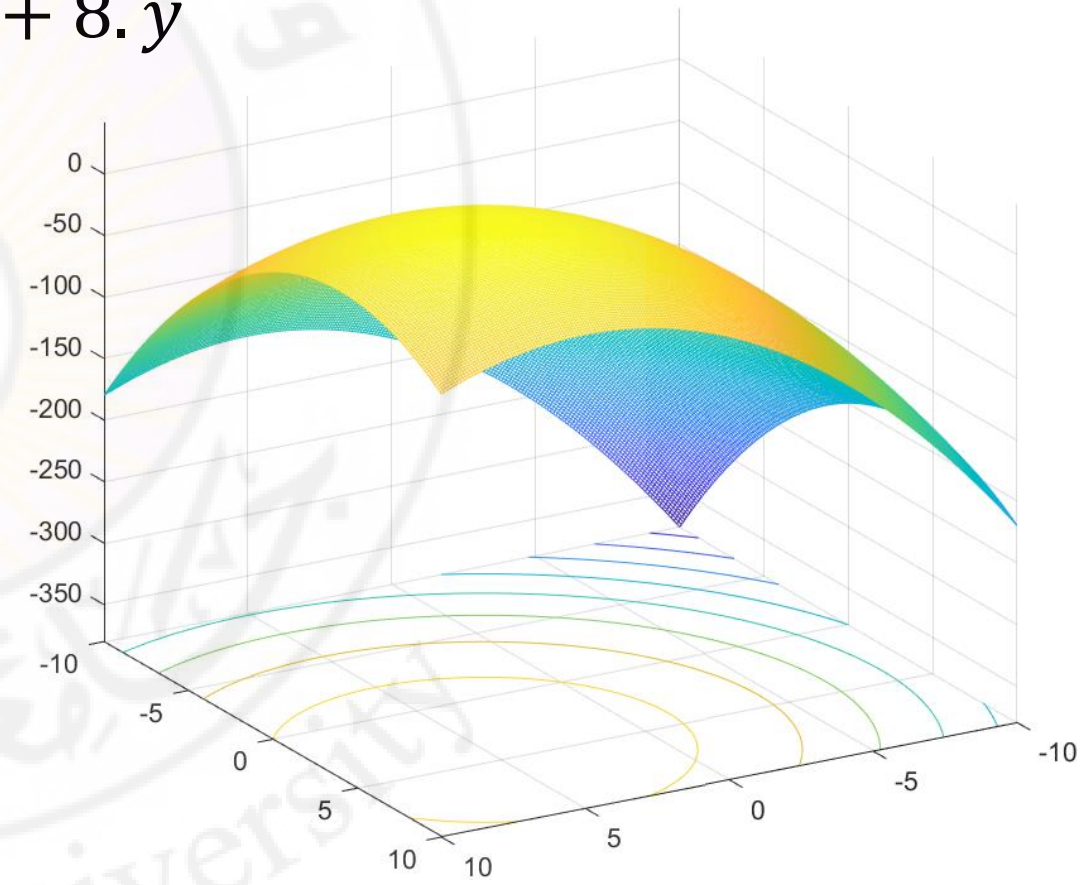
- Maximize $f(x, y) = -x^2 + 10.x - y^2 + 8.y$

- Symbolic:

```
f = @(x)  -x(1).^2+10*x(1)-x(2).^2+8*x(2);
```

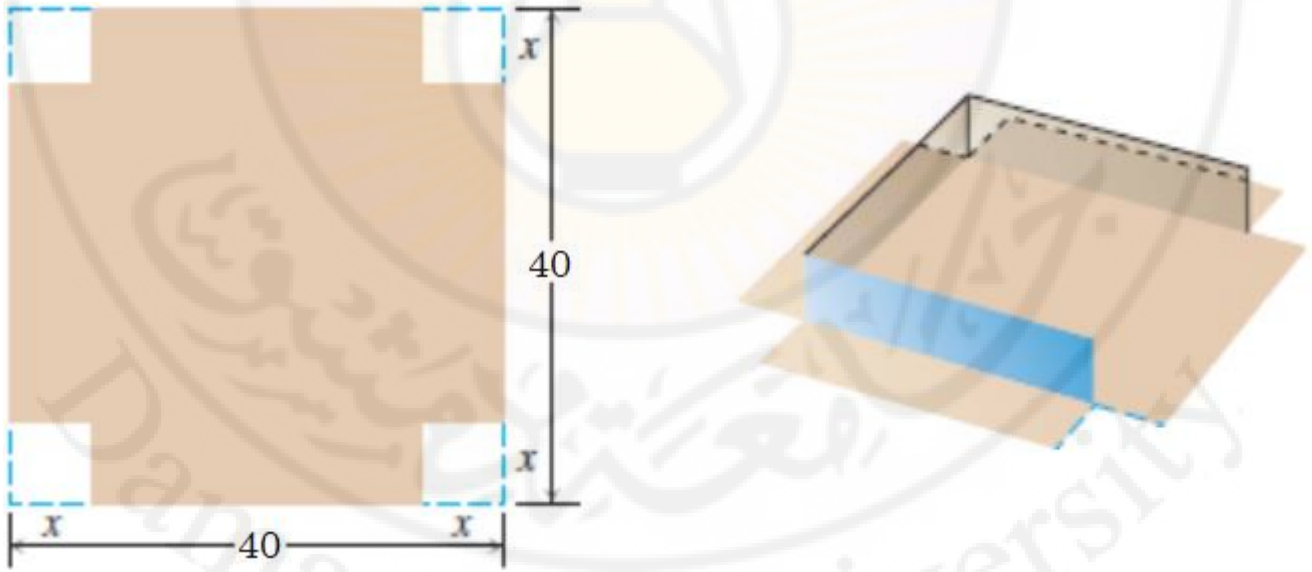
Minimization only

```
f = @(x)-(-x(1).^2+10*x(1)-x(2).^2+8*x(2));
x0 = [0, 0];
sol = fminsearch(f, x0);
disp(sol) % 5, 4
f(sol)    % -41
```



Example 2

- An open rectangular box with square base is made from 40x40cm sheet bended up as shown
- What is the largest possible volume?



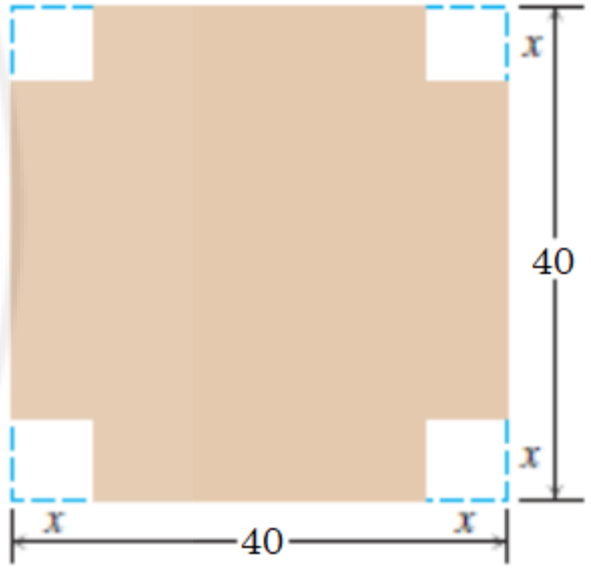
Example 2

- Variable x
- Objective volume:

$$v(x) = x.(40 - 2.x)^2 = 1600x - 160x^2 + 4x^3$$

- Subject to constraint:

$$0 < x < 20$$



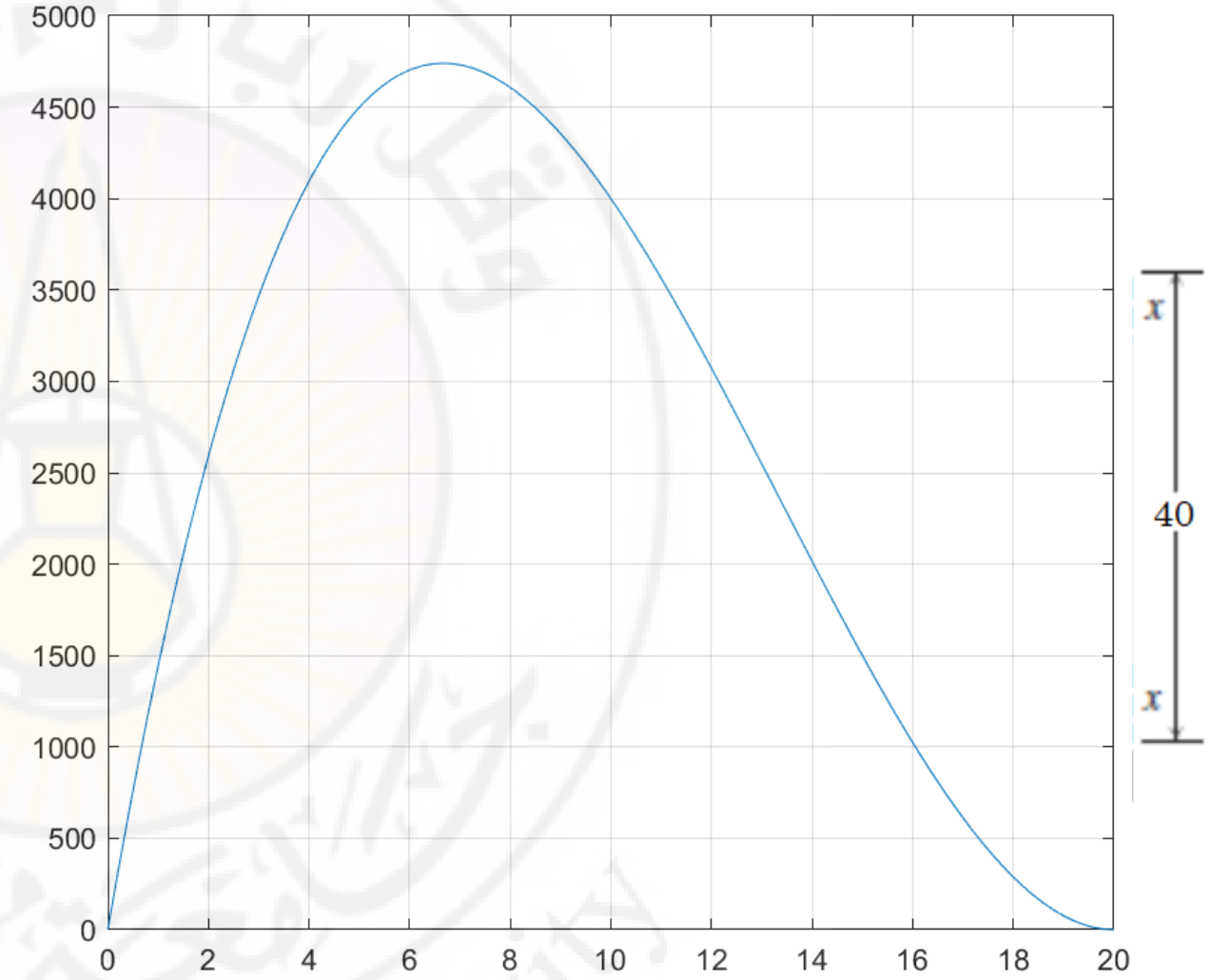
$$\frac{\delta v(x)}{\delta x} = 1600 - 320x + 12x^2 = 0$$

$$133.33 - 26.67x + x^2 = (x - 20).(x - 6.67) = 0$$

Example 2

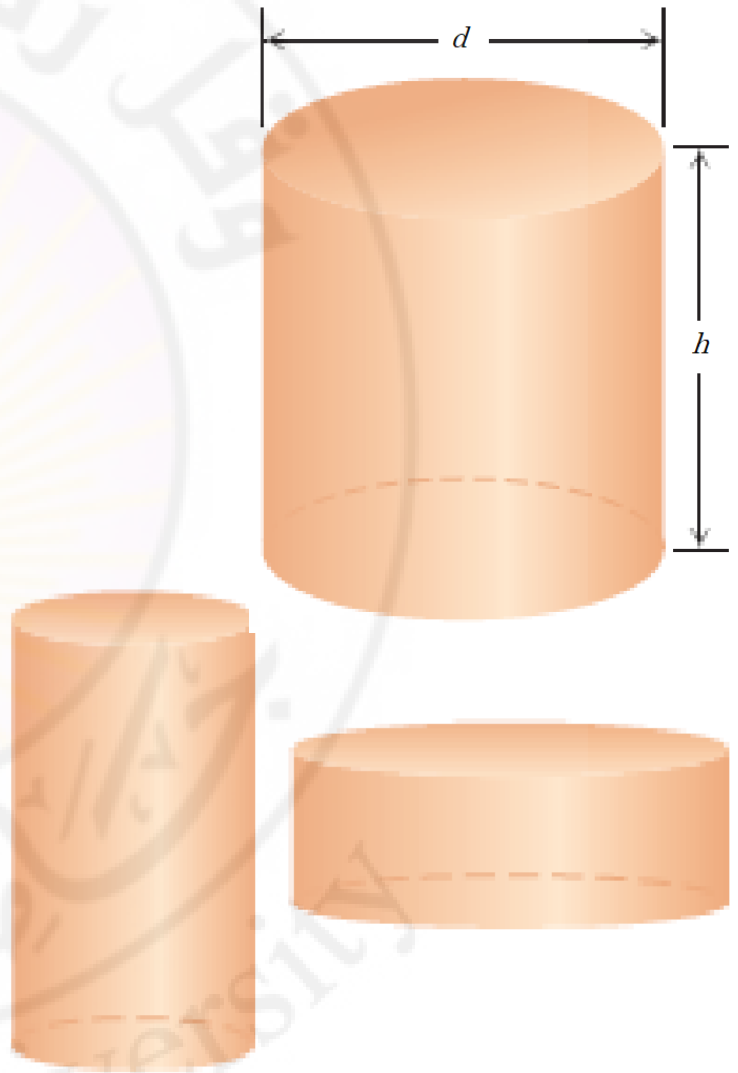
- $v(x) = 1600x - 160x^2 + 4x^3$
- $1600 - 320x + 12x^2 = 0$
  - $x = 20$
  - $x = 6.67$
- Objective volume $= 4740.7$
- Numerical:

```
x = linspace(0, 20);
v = 1600.*x-160.*x.^2+4*x.^3;
plot(x, v), grid
roots([12 -320 1600])
```



Example 3

- Design optimum cylindrical tank with minimum material and volume 1000 litre
- Variables *d* and *h*
- Minimize surface area
- Subject to volume



Example 3

- Variables d, h in metre
- Objective surface:

$$a(d,h) = \pi . d . h + 2 \times 0.25\pi d^2 = \pi d(h + 0.5d)$$

- Subject to constraint:

$$v(d,h) = 0.25\pi d^2 h = 1 \;\Rightarrow\; h = \frac{4}{\pi d^2}$$

$$a(d,h) = \pi d\left(\frac{4}{\pi d^2} + 0.5d\right) = \frac{4}{d} + 0.5\pi d^2$$

$$\frac{\delta a(d)}{\delta d} = \pi d - \frac{4}{d^2} = 0 \Rightarrow d = \sqrt[3]{\frac{4}{\pi}} = 1.0839$$

Example 3

- Variables d, h in metre
- Objective surface:

$$a(d,h) = \pi.d.h + 2 \times 0.25\pi d^2 = \pi d(h + 0.5d)$$

- Subject to constraint:

$$v(d,h) = 0.25\pi d^2 h = 1$$

$$d = 1.0839$$

$$h = \frac{4}{\pi d^2} = 1.0839$$

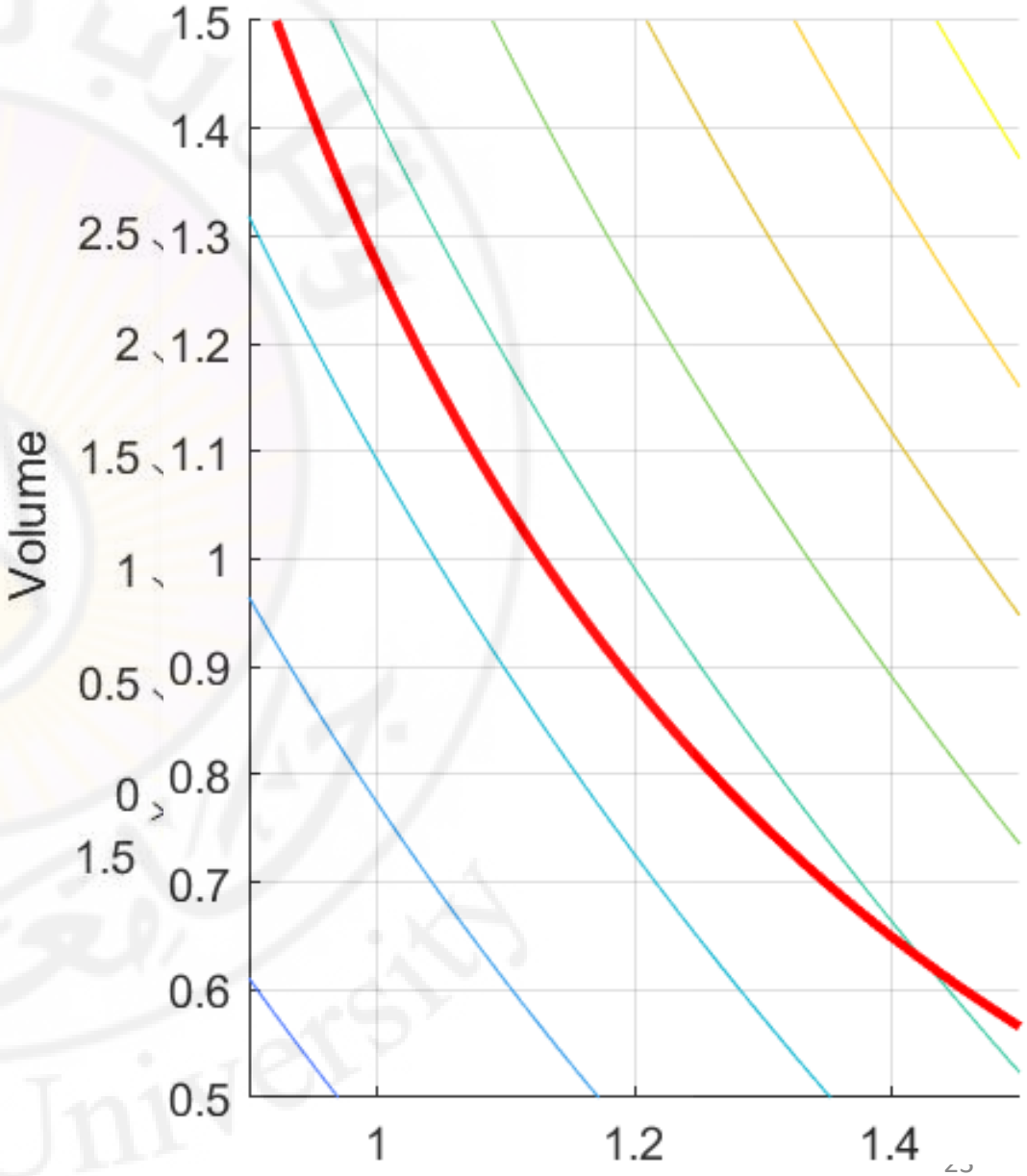
Example 3

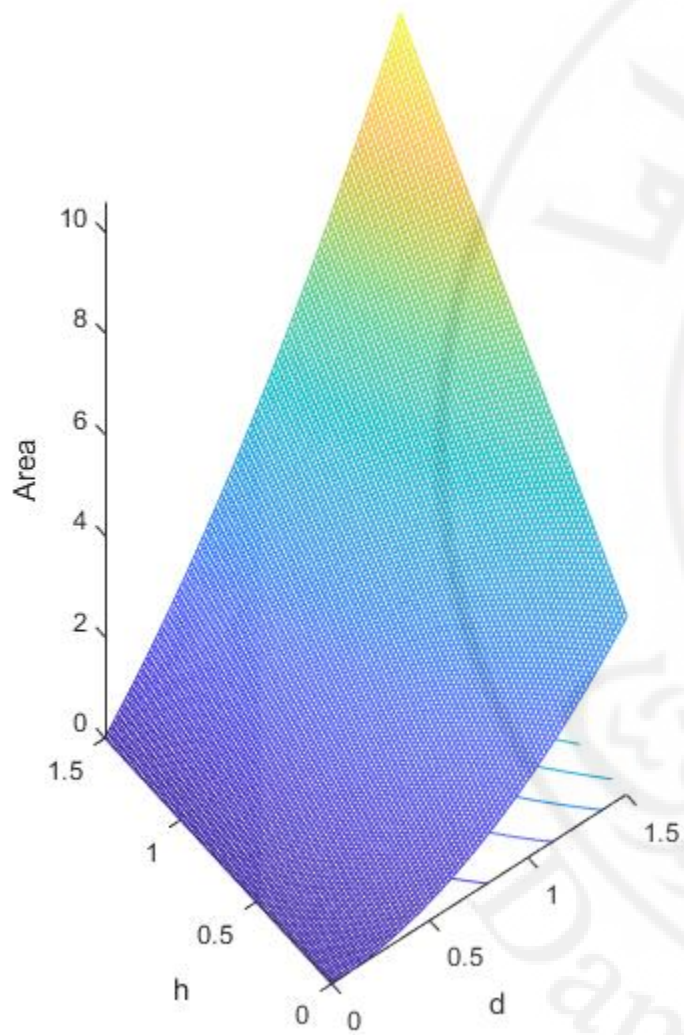
```
d1 = linspace(0, 1.5);
h1 = linspace(0, 1.5);
[d, h] = meshgrid(d1, h1);

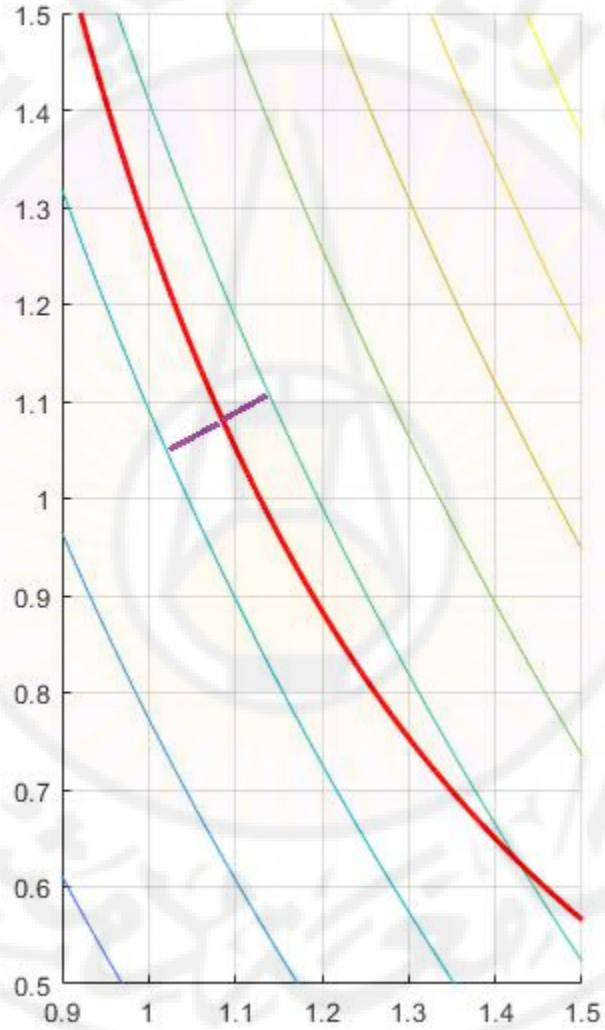
a = pi*d.*(h+0.5*d);
v = 0.25*pi*d.^2.*h;

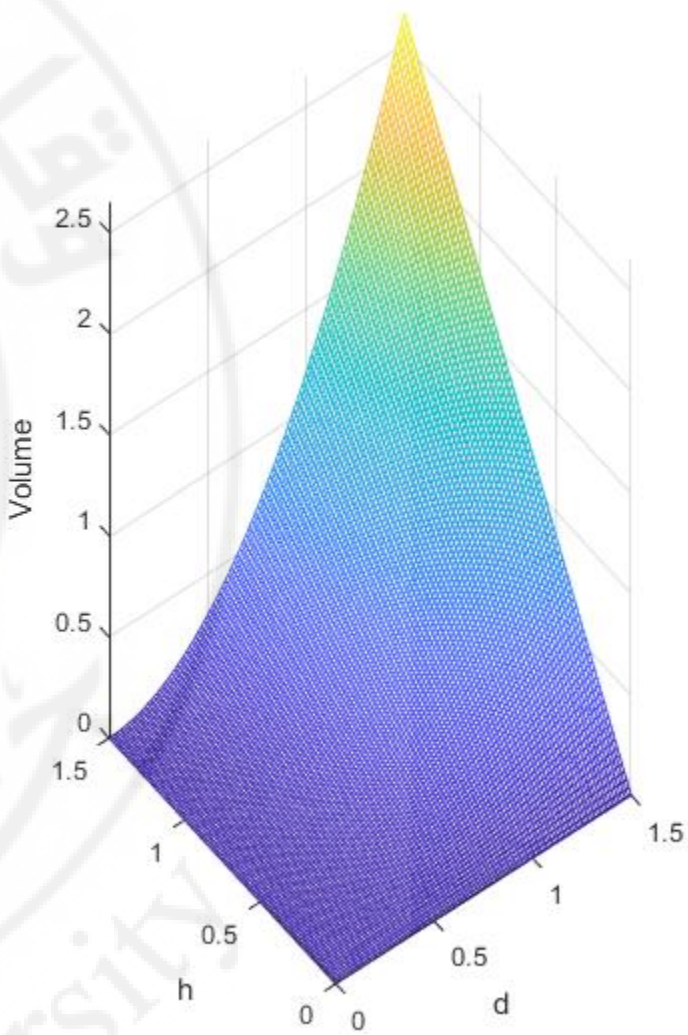
meshc(d, h, a)
meshc(d, h, v)

contour(d, h, v, [1 1], ...
   LineWidth=2, EdgeColor='red')
contour(d, h, a)
```



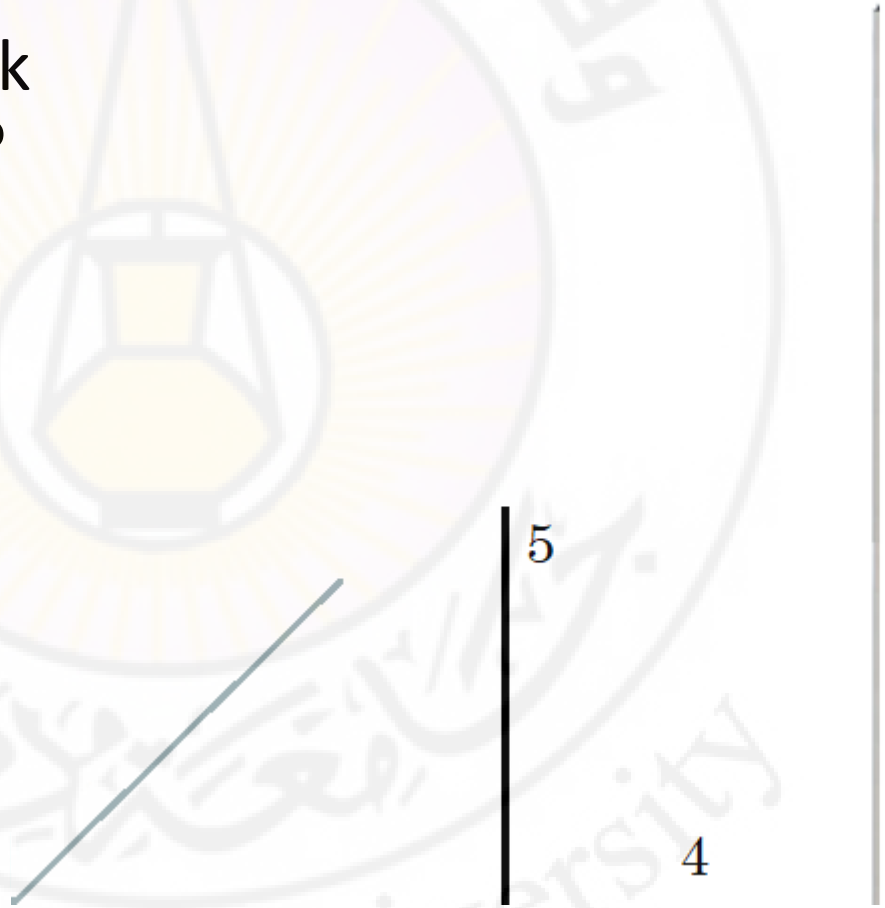

Area
10
8
6
4
2
0
1.5
1
0.5
h
0
0
0.5
1
1.5
d


1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.9
1
1.1
1.2
1.3
1.4
1.5


Volume
2.5
2
1.5
1
0.5
0
1.5
1
0.5
h
0
0
0.5
1
1.5
d

Example 4

- What is the shortest fire truck ladder to get to the building?



Example 4

- Variable x
- Objective minimize L:

$$L^2 = y^2 + (x+4)^2$$

- Constraint:

$$0 < x < \infty$$

$$\tan(\theta) = \frac{y}{4+x} = \frac{5}{x}$$

$$y = 5(4+x)/x$$

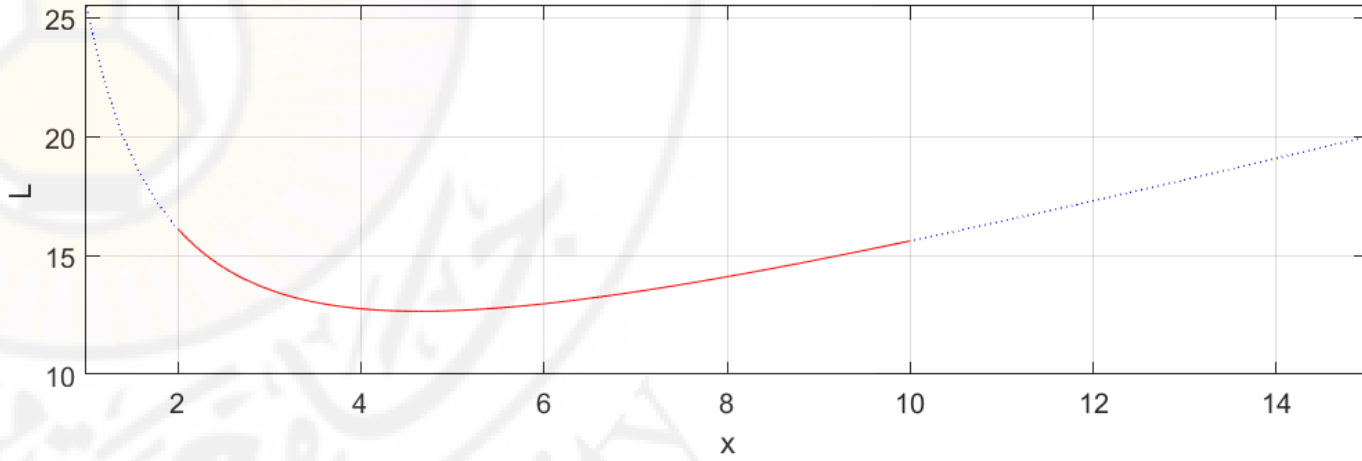
$$L^2 = \left(\frac{5(4+x)}{x}\right)^2 + (x+4)^2$$

Example 4

$$L = \sqrt{(\frac{5(4+x)}{x})^2+(x+4)^2}$$

```
h = 5; l = 4;
x = linspace(1, 20);
L = sqrt((h*(x+l)./x).^2+(x+l).^2);
plot(x, L, ':b')
disp(min(L))   % 12.7017

syms x
f = sqrt((h*(x+l)./x).^2+(x+l).^2);
% fplot(f)
fplot(f, [2 10], 'r')
```

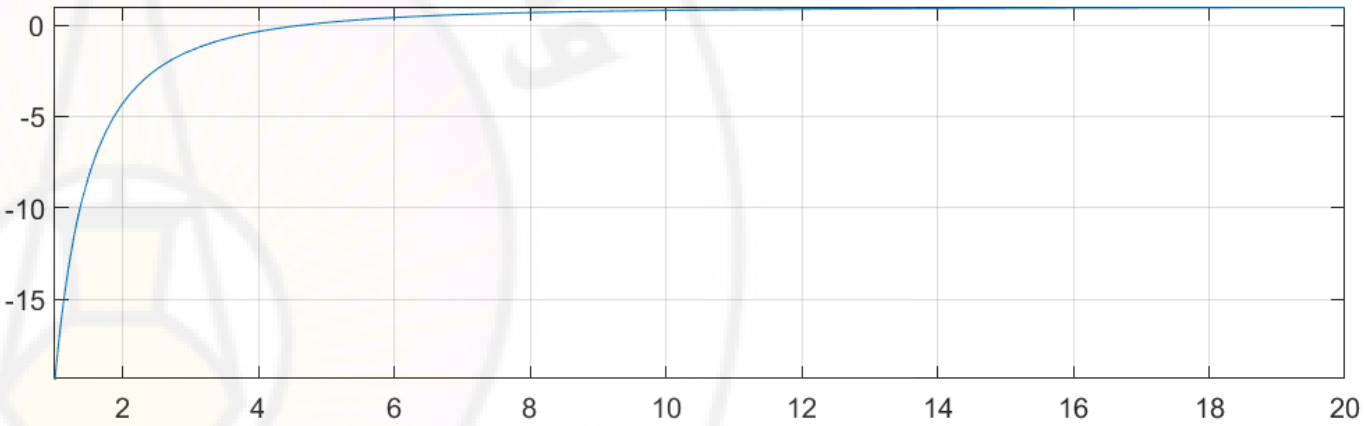


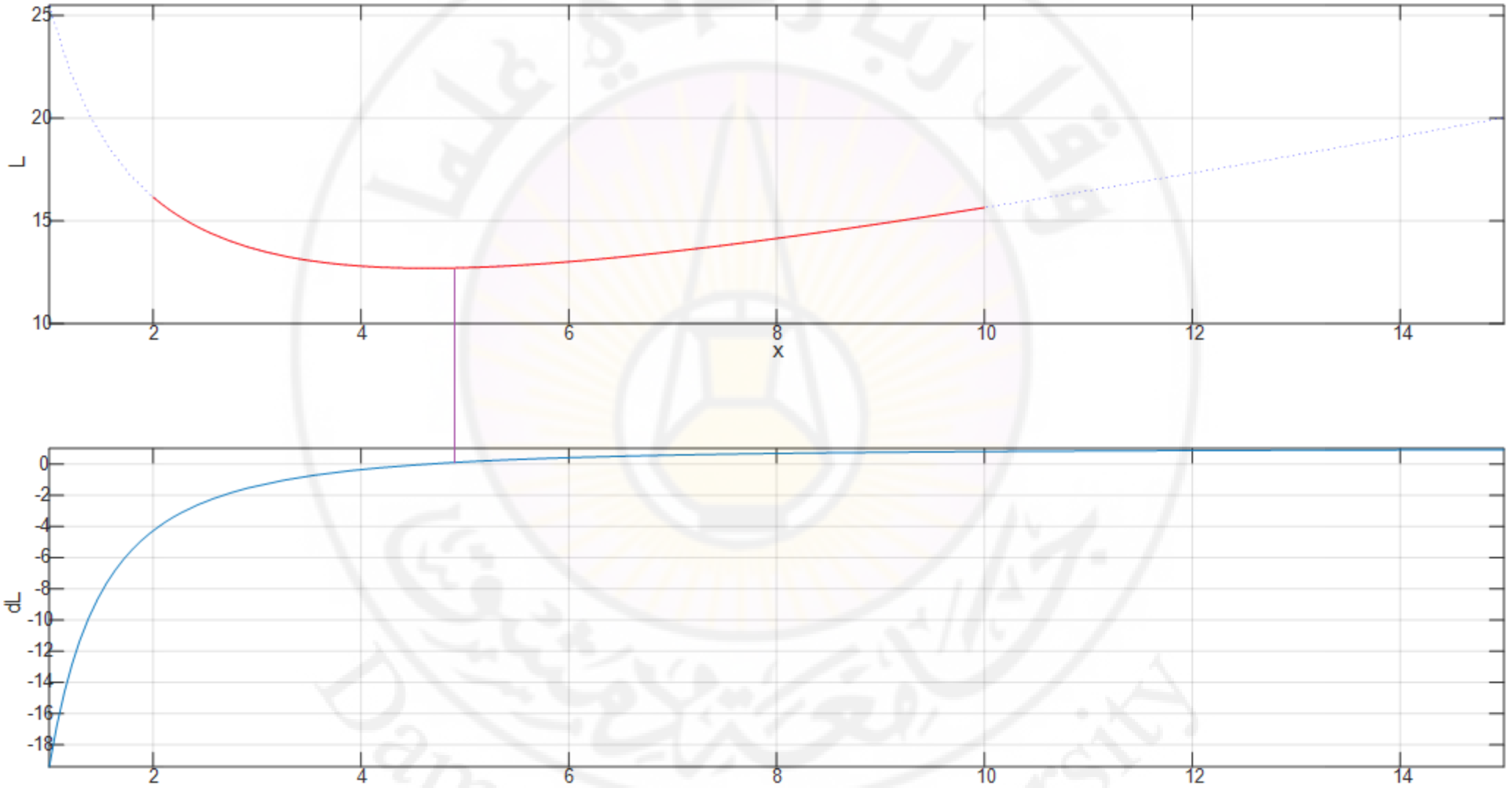
Example 4

```
syms x
f = sqrt((h*(x+l)./x).^2+(x+l).^2)
% fplot(f)
fplot(f, [2 10], 'r')

df = diff(f,x);
fplot(df, [1 15])

eval(solve(df, x))      % 4.6416, -2.3208 + 4.0197i, -2.3208 - 4.0197i
eval(subs(f, 4.6416))   % 12.7017
```




25
20
15
10
L
2
4
6
8
10
12
14
x
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
dL
2
4
6
8
10
12
14

Computer **A**ided **D**esign

MATLAB Introduction for Civil Engineers 7

13/12/2022

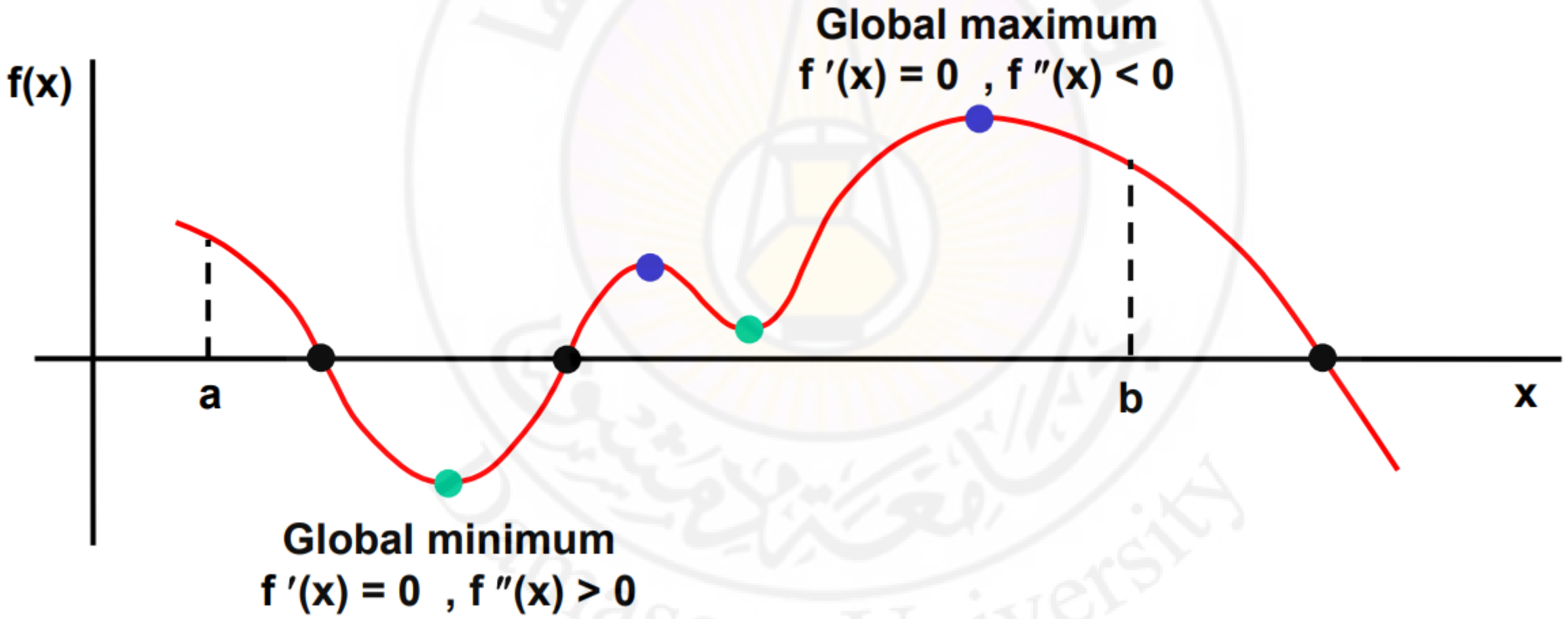
Wael Darwich

Damascus University

Content

- Numerical Solvers
  - Bisection
  - Newton–Raphson
  - Golden–Section Search
- Example

Numerical Solvers



Bisection method

- Root finding tool
- Continuous functions
- Start with initial values of opposite signs
- Reduce range
- Get root within numerical tolerance

Bisection

Find root of:

$$f(x) = 0.15x^3 - x^2 + x + 10$$

```
y = @(x)0.15*x.^3-x.^2+x+10;
fzero(y, -5)
-2.3720
```

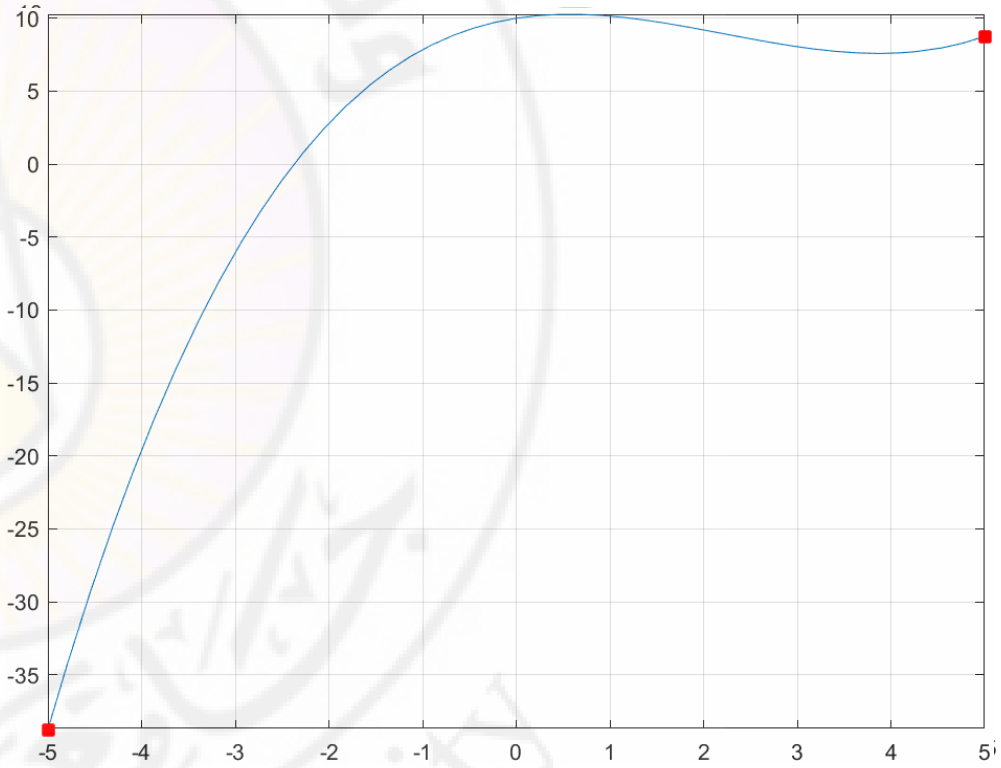
Bisection

Find root of:

$$f(x) = 0.15x^3 - x^2 + x + 10$$

```
y = @(x)0.15*x.^3-x.^2+x+10;
x1 = -5; x2 = 5;

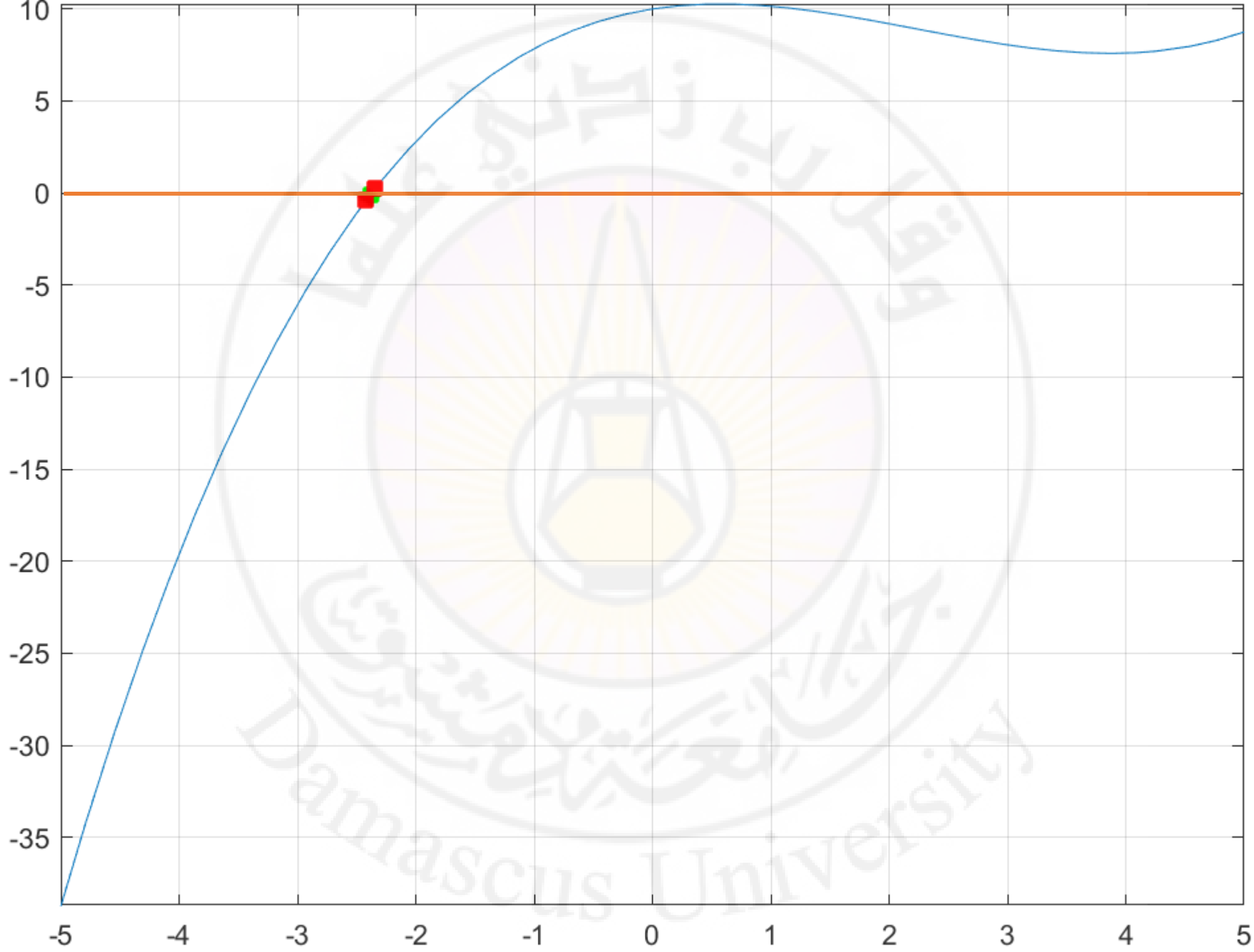
step = 0.1;
for i = 1:100
        y1 = y(x1);
        y2 = y(x2);
        plot([x1 x2], [y1 y2], 'r+')
        if (y1 < 0)
                x1 = x1 + step;
        end
        if (y2 > 0)
                x2 = x2 - step;
        end
end
```

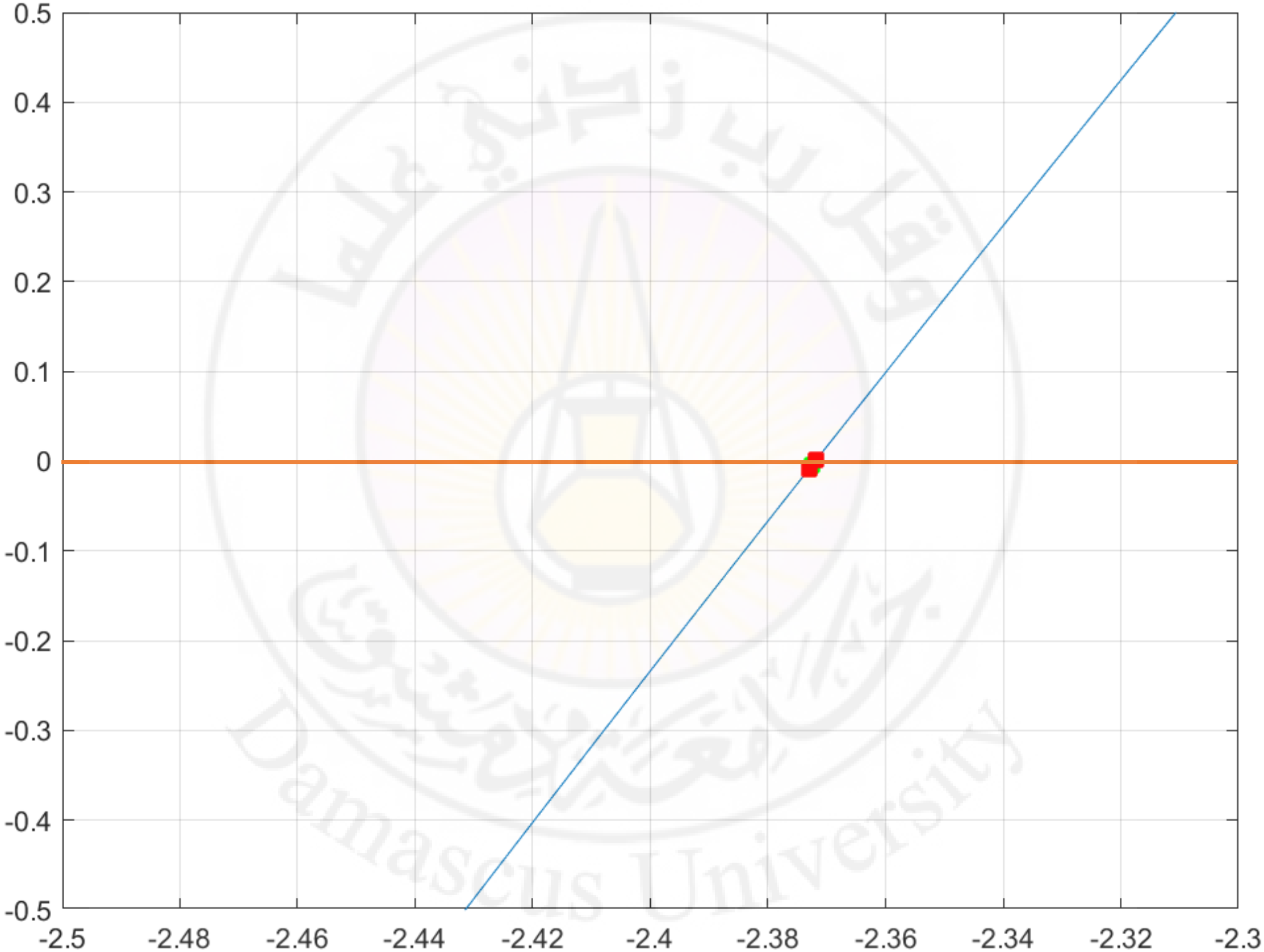


Bisection

- Number of steps
- Root position in initial range
- Step size, fixed vs variable

- Bisection: split half range each iteration
  - Keep range end points on different y signs


10
5
0
-5
-10
-15
-20
-25
-30
-35
-5
-4
-3
-2
-1
0
1
2
3
4
5


0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-2.5
-2.48
-2.46
-2.44
-2.42
-2.4
-2.38
-2.36
-2.34
-2.32
-2.3

Bisection

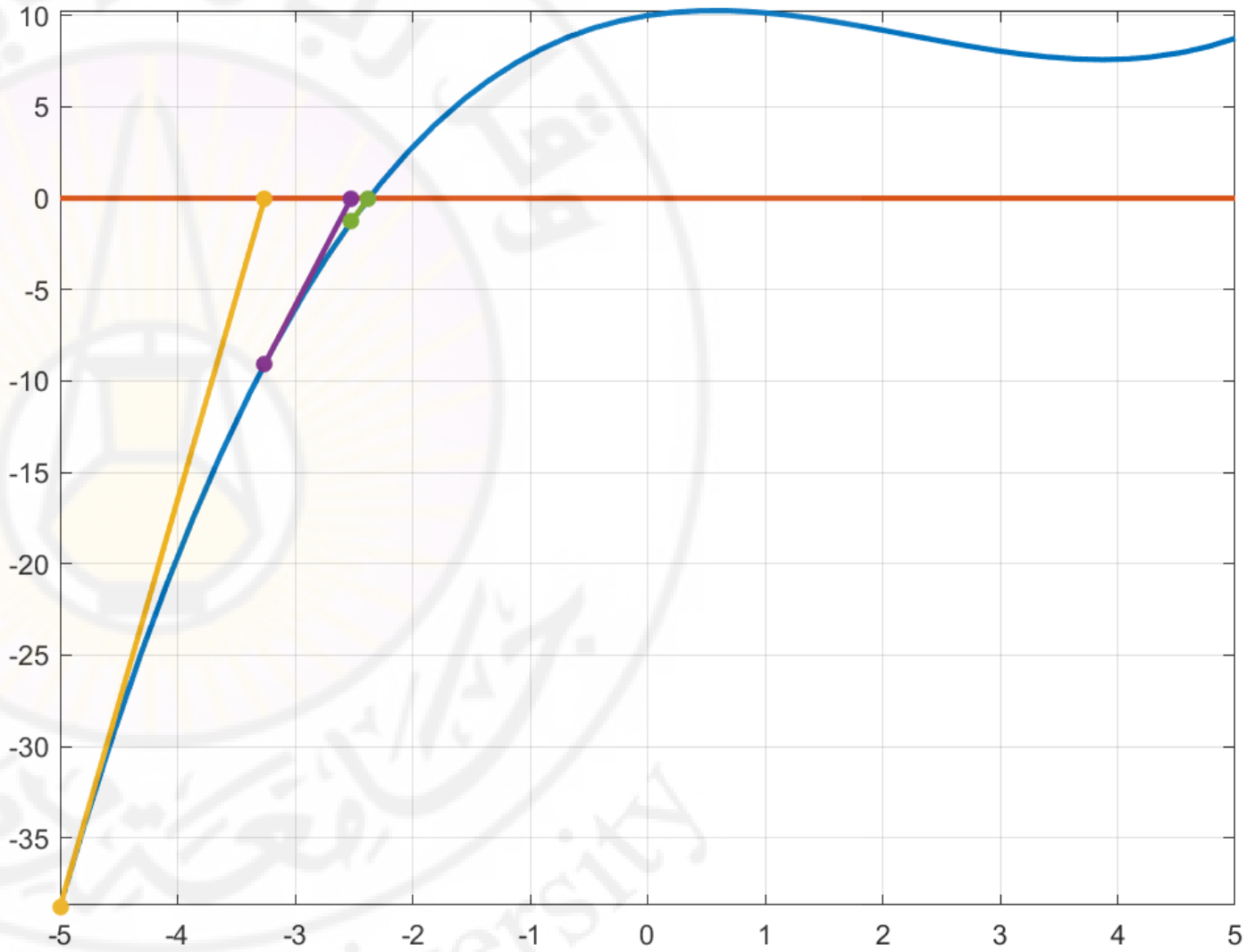
Itr.	x1	x2	y
1	-5.00	5.00	10.00000
2	-5.00	0.00	-1.09375
3	-2.50	0.00	6.89453
4	-2.50	-1.25	3.62061
5	-2.50	-1.88	1.45721
6	-2.50	-2.19	0.23190
7	-2.50	-2.34	-0.41817
8	-2.42	-2.34	-0.08998
9	-2.38	-2.34	0.07175
10	-2.38	-2.36	-0.00892
11	-2.37	-2.36	0.03146
12	-2.37	-2.37	0.01129
13	-2.37	-2.37	0.00119
14	-2.37	-2.37	-0.00386
15	-2.37	-2.37	-0.00134
16	-2.37	-2.37	-0.00008

Bisection

Itr.	x1	x2	y
1	-5.00	5.00	10.00000
2	-5.00	0.00	-1.09375
3	-2.50	0.00	6.89453
4	-2.50	-1.25	3.62061
5	-2.50	-1.88	1.45721
6	-2.50	-2.19	0.23190
7	-2.50	-2.34	-0.41817
8	-2.42	-2.34	-0.08998
9	-2.38	-2.34	0.07175
10	-2.38	-2.36	-0.00892
11	-2.37	-2.36	0.03146
12	-2.37	-2.37	0.01129
13	-2.37	-2.37	0.00119
14	-2.37	-2.37	-0.00386
15	-2.37	-2.37	-0.00134
16	**-2.37**	**-2.37**	-0.00008

Newton–Raphson

- Root finding tool
- Continuous differentiable function
- Start with initial guess, e.g. `x=-5`
- Iterate for better solution
  - linear approximation



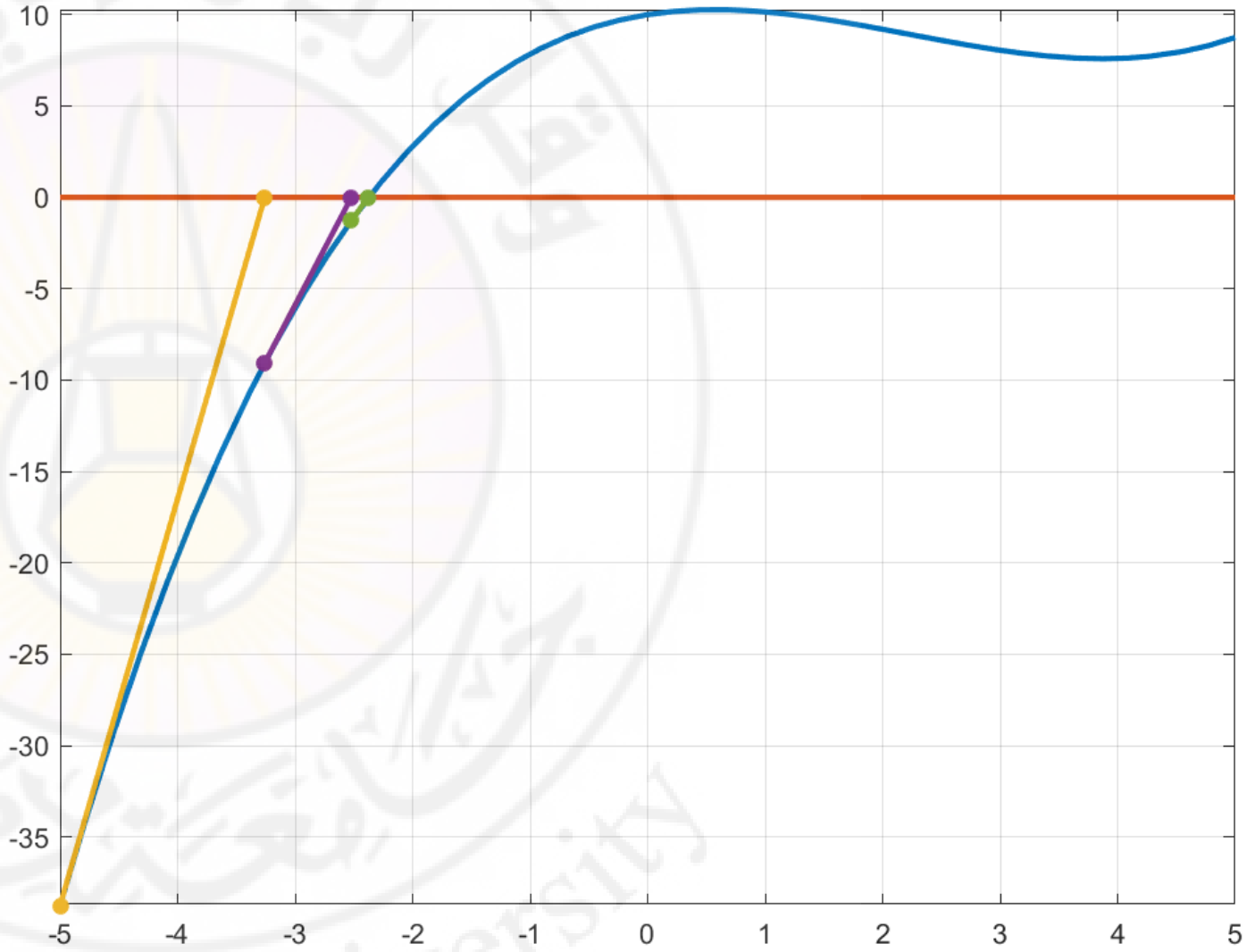
Newton–Raphson

$$f'(x_n) = \frac{\Delta y}{\Delta x}$$

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

$$f'(x_n) = \frac{0 - f(x_n)}{x_{n+1} - x_n}$$

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

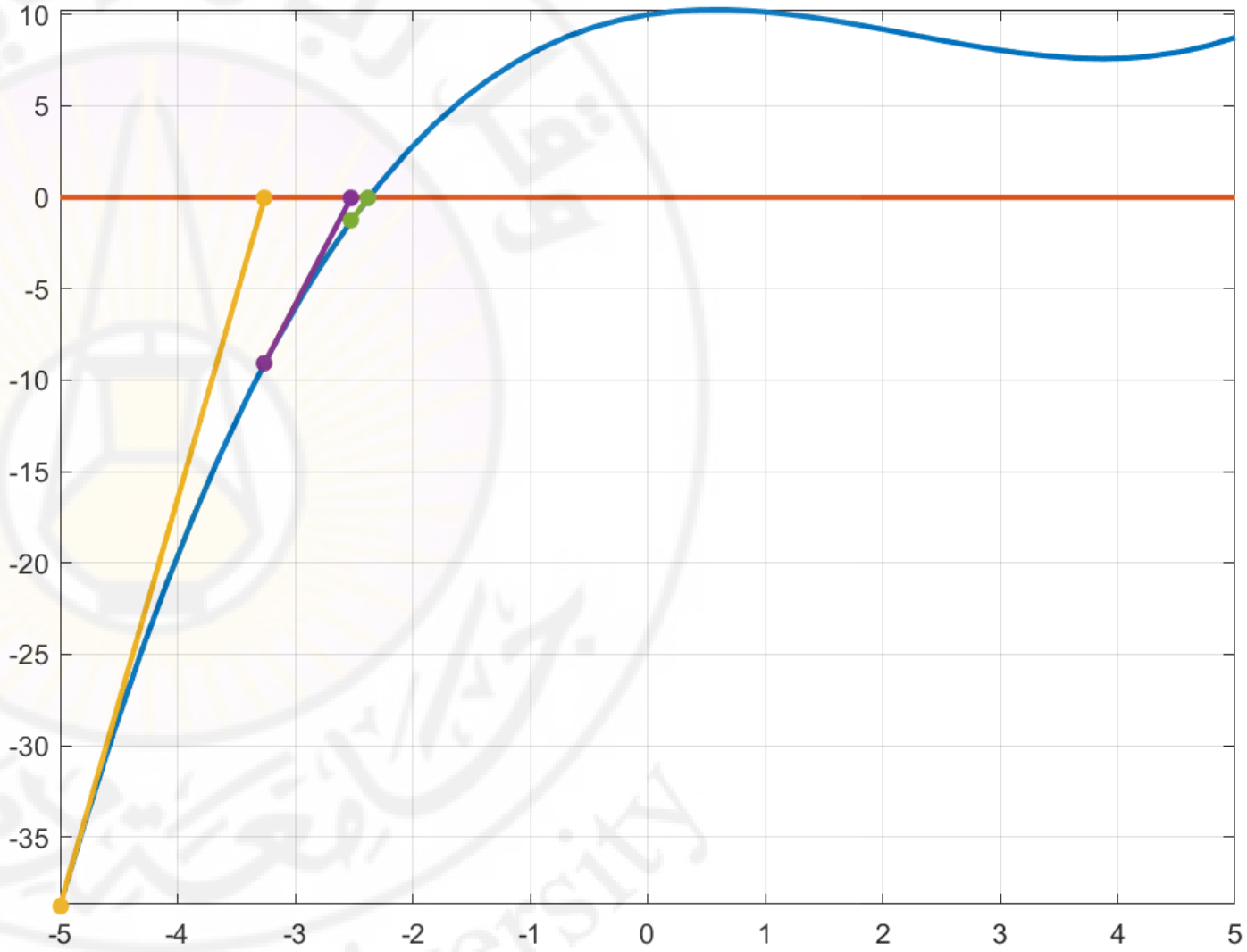


Newton–Raphson

```
syms x
y = 0.15*x.^3-x.^2+x+10;
dy = diff(y,x);

x1 = -5;
fplot(y)

y1 = 1;
while abs(y1) > 1e-4
        y1 = subs(y, x1);
        dy1 = subs(dy, x1);
        x2 = x1 - y1/dy1;
        plot([x1 x2], [y1 0], '-*')
        x1 = x2;
end
```



Newton–Raphson

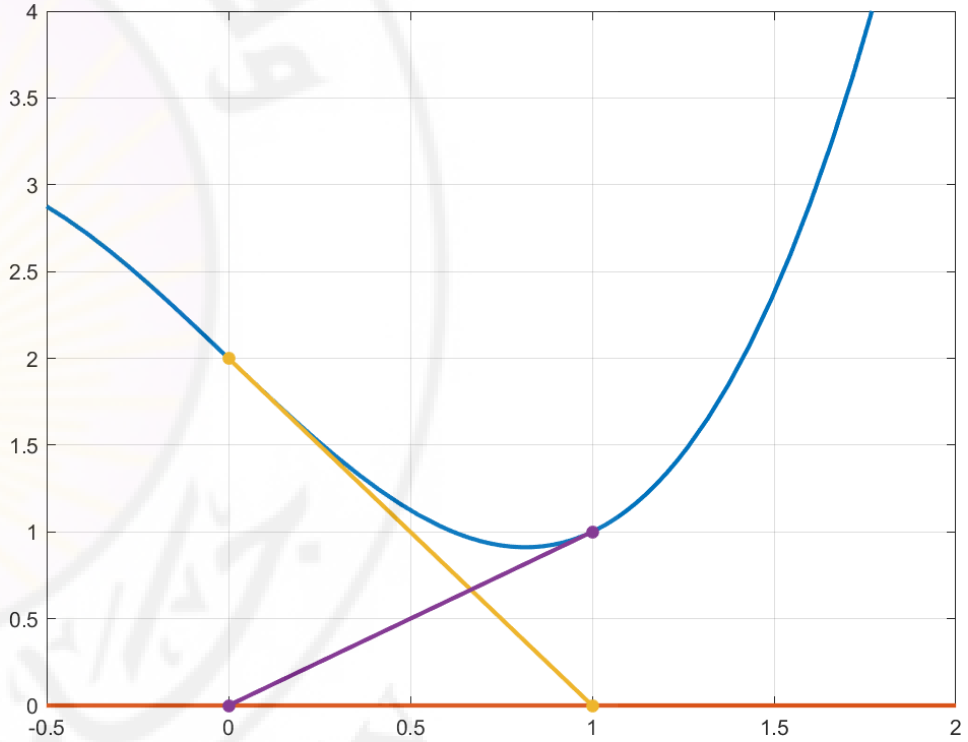
- Needs function derivative
  - no vertical tangent line
  - can get numerical slope using two points
  - secant method: slower, easier
- Derivative cannot be zero

$$f'(x) = 0$$

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

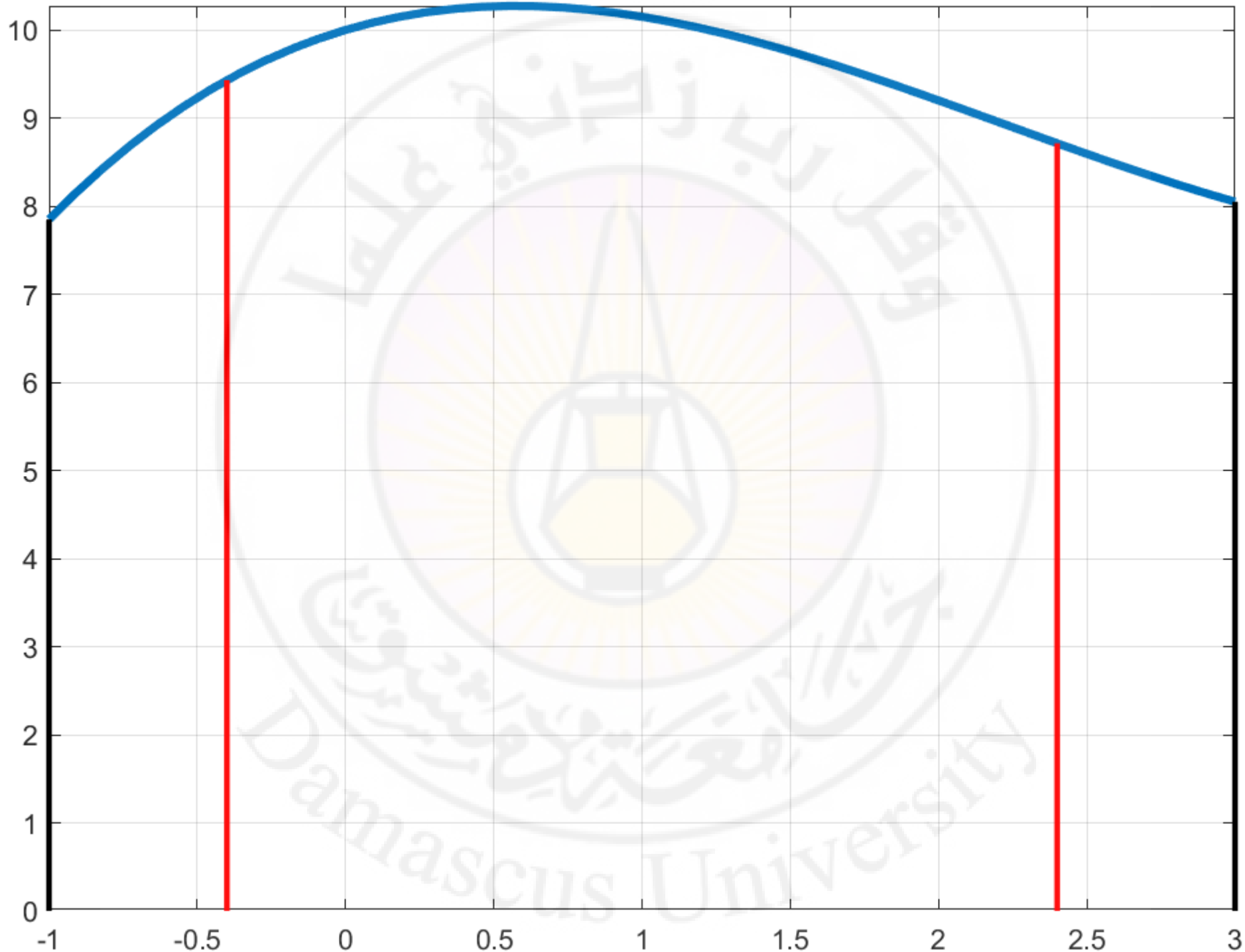
- Oscillating forever

$$x^3 - 2x + 2$$



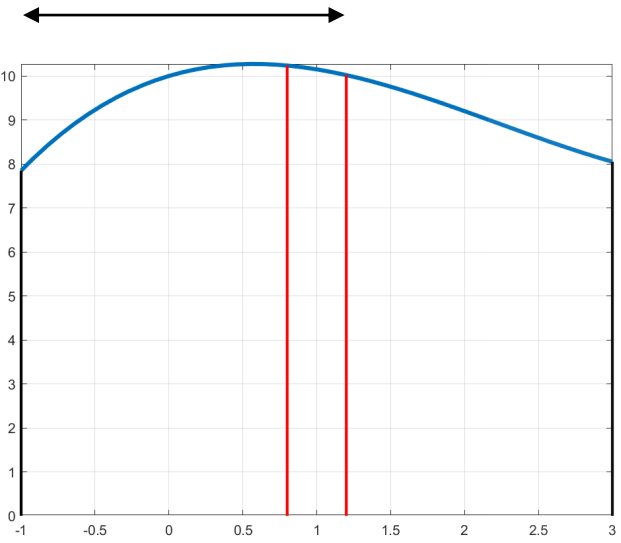
Golden–Section Search

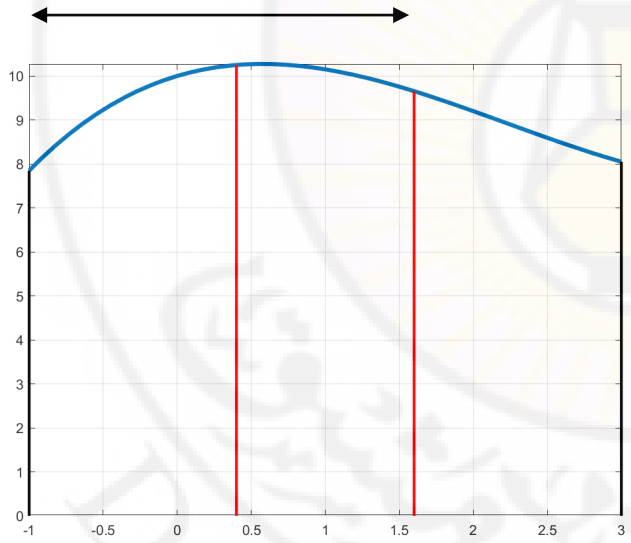
- Extremum finding tool
- In a given range
- Only for unimodal function
  - Single solution in range
- Reduce range size
  - y is unknown (unlike root)

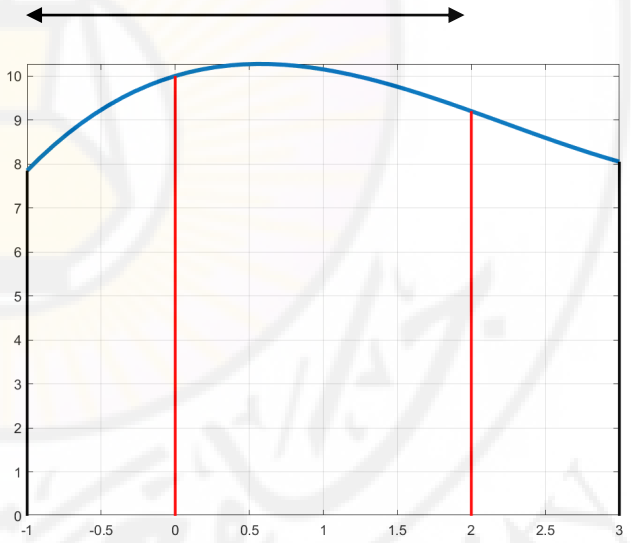

10
9
8
7
6
5
4
3
2
1
0
-1
-0.5
0
0.5
1
1.5
2
2.5
3

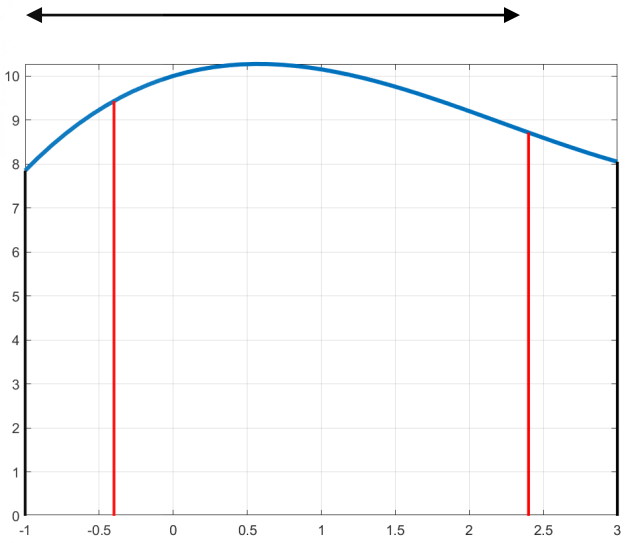
Golden–Section Search

- $y_1 > y_2 \Rightarrow$ maximum on the left







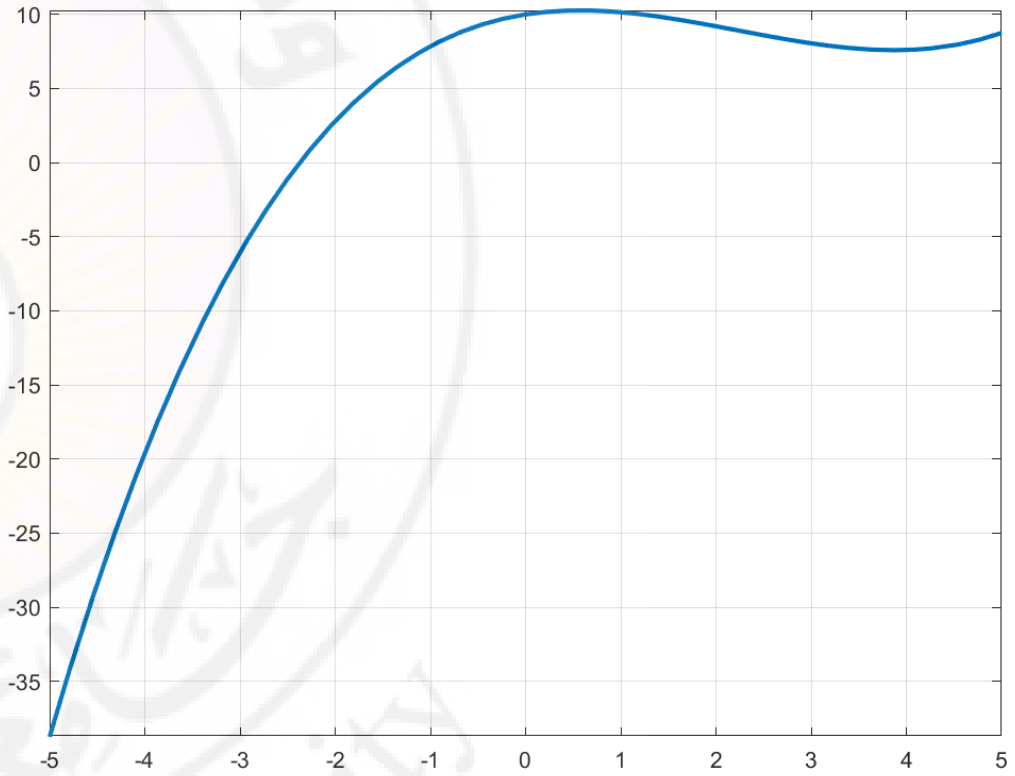


Golden–Section Search

Find maximum value of:

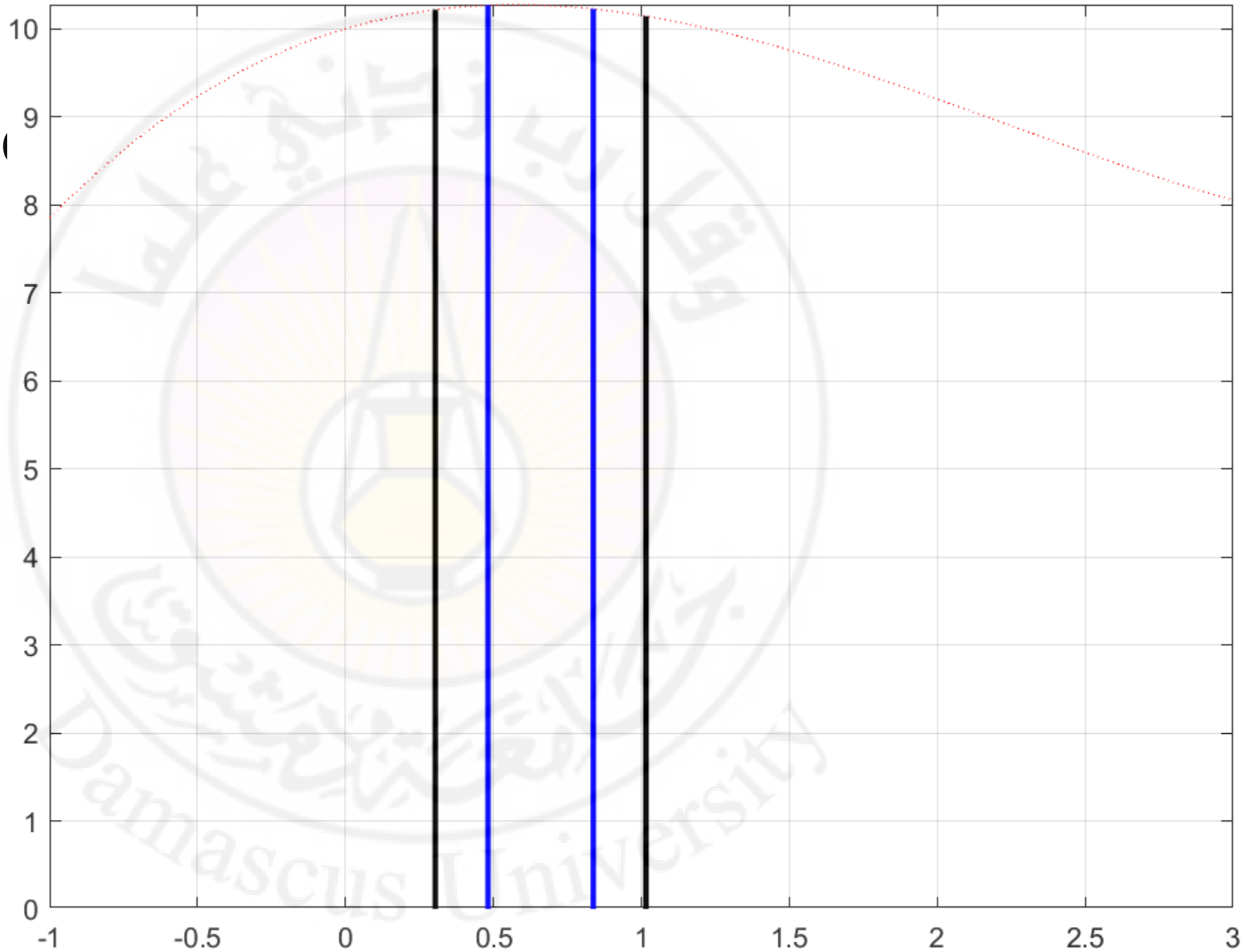
$$f(x) = 0.15x^3 - x^2 + x + 10$$

```
y = @(x)    0.15*x.^3-x.^2+x+10;
fplot(y, [-1 3])
ym= @(x) -(0.15*x.^3-x.^2+x+10);
x = fminbnd(ym, -1,3) % 0.5742
y(x)                  % -10.2729
```

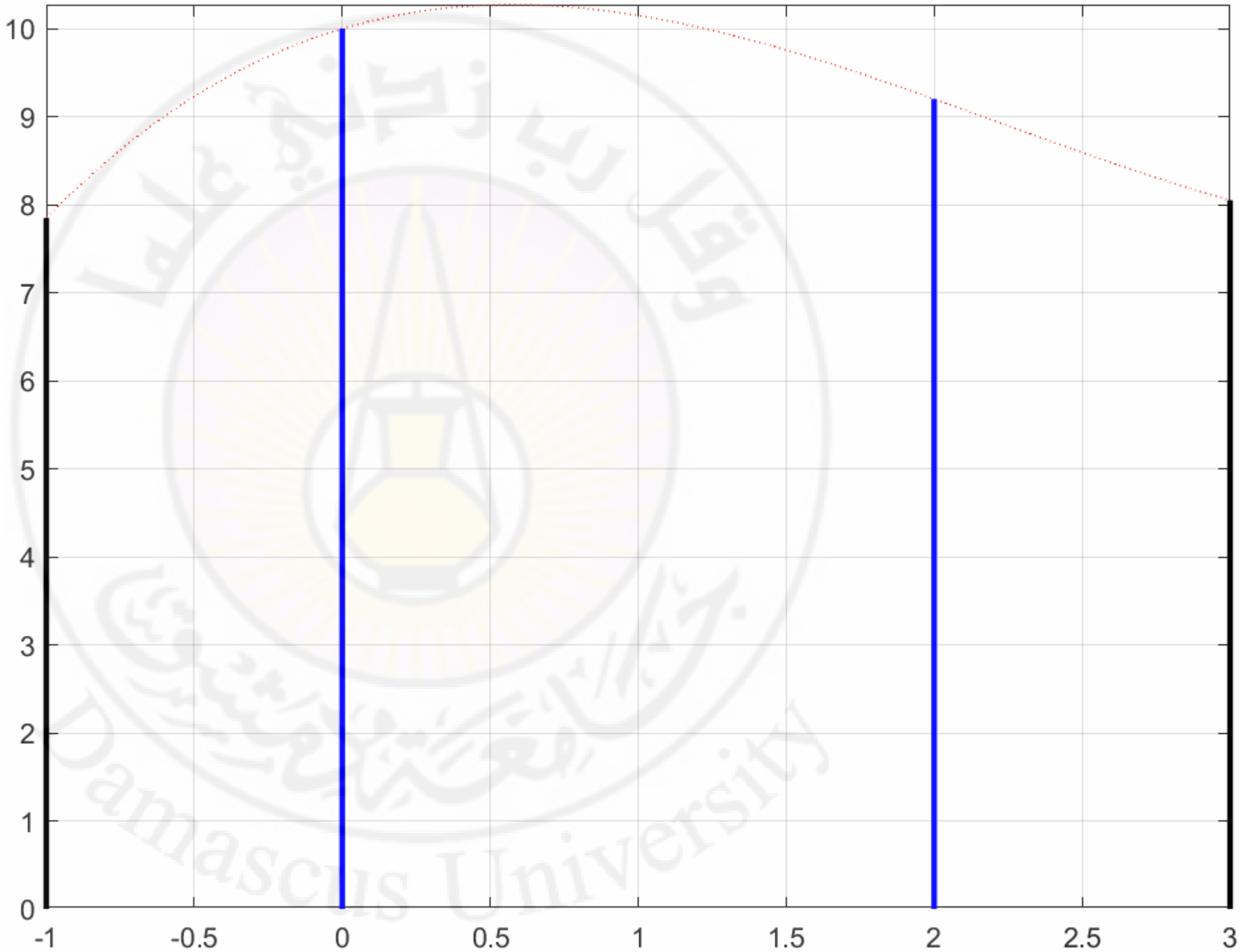


Golden–S

-1.0000	3.0000
-1.0000	2.0000
-0.2500	2.0000
-0.2500	1.4375
-0.2500	1.0156
0.0664	1.0156
0.3037	1.0156
0.3037	0.8376
0.4372	0.8376
0.4372	0.7375
0.4372	0.6624
0.4935	0.6624
0.4935	0.6202
0.5252	0.6202
0.5489	0.6202
0.5489	0.6024
0.5489	0.5890
0.5590	0.5890

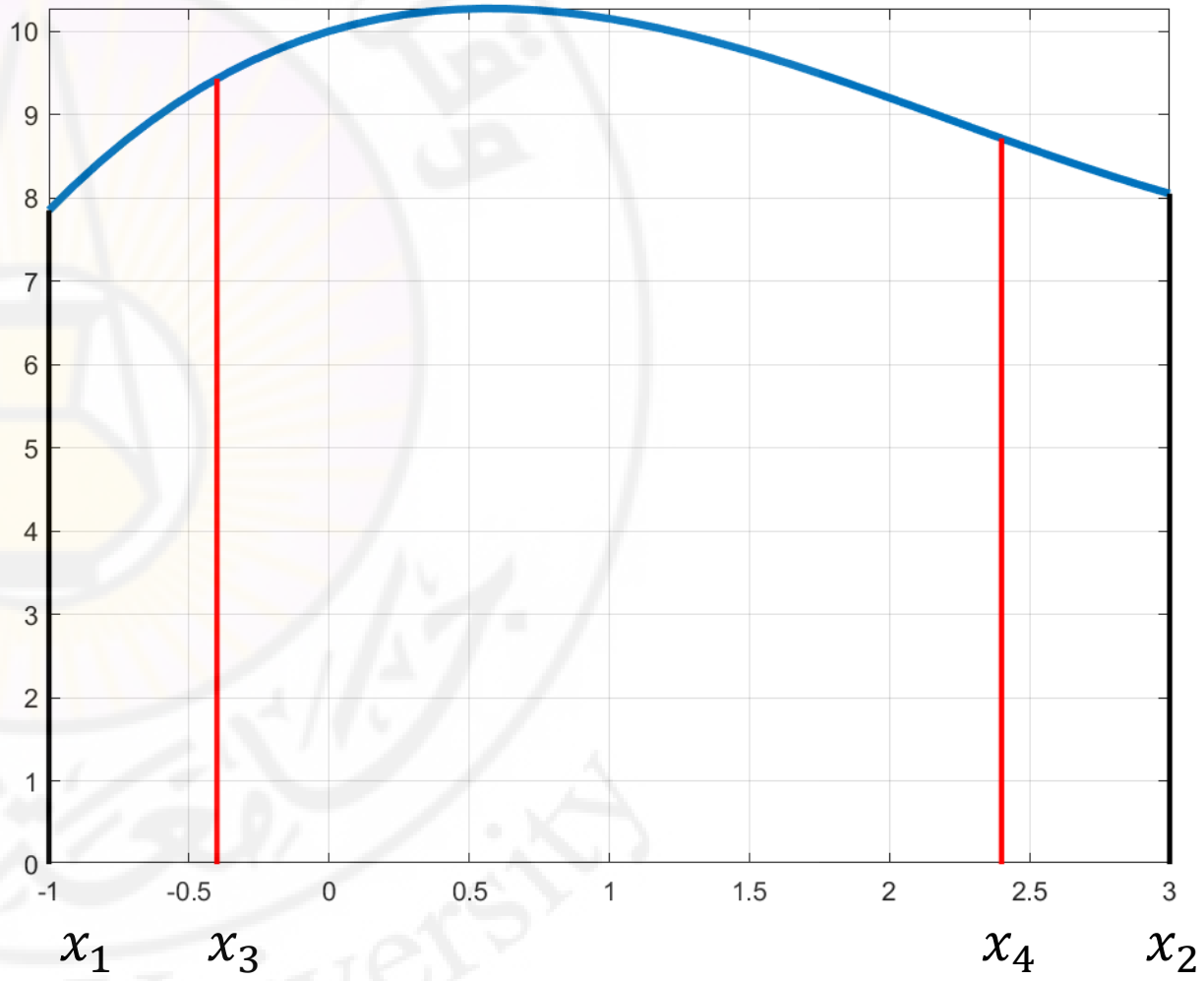


```
   x   =   0.5742
y(x)   =  10.2729
```



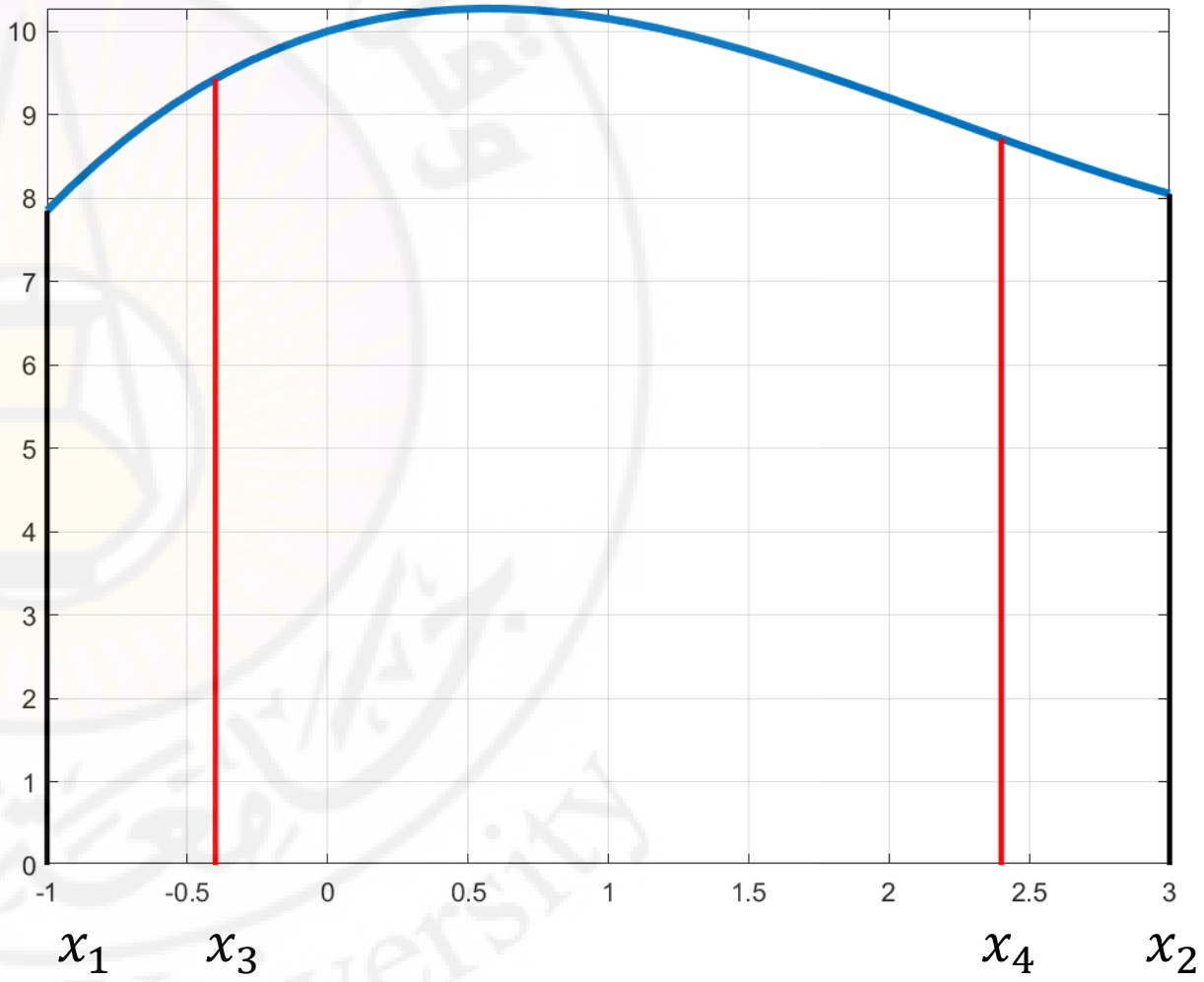
Golden–Section Search

```
y = @(x)0.15*x.^3-x.^2+x+10;
x1 = -1; x2 = 3;
ratio = 0.25;
epsilon = 1e-4;
while abs(x2-x1) > epsilon
    d = x2-x1;
    x3 = x1 + ratio * d;
    x4 = x2 - ratio * d;
    y3 = y(x3);
    y4 = y(x4);
    if (y3 > y4)
        x2 = x4;
    else
        x1 = x3;
    end
end
```



Golden–Section Search

```
y = @(x)0.15*x.^3-x.^2+x+10;
x1 = -1; x2 = 3;
ratio = 0.25;
epsilon = 1e-4;
while abs(x2-x1) > epsilon
    d = x2-x1;
    x3 = x1 + ratio * d;
    x4 = x2 - ratio * d;
    y3 = y(x3);
    y4 = y(x4);
    if (y3 > y4)
        x2 = x4;
    else
        x1 = x3;
    end
end
```



Golden–Section Search

First iteration

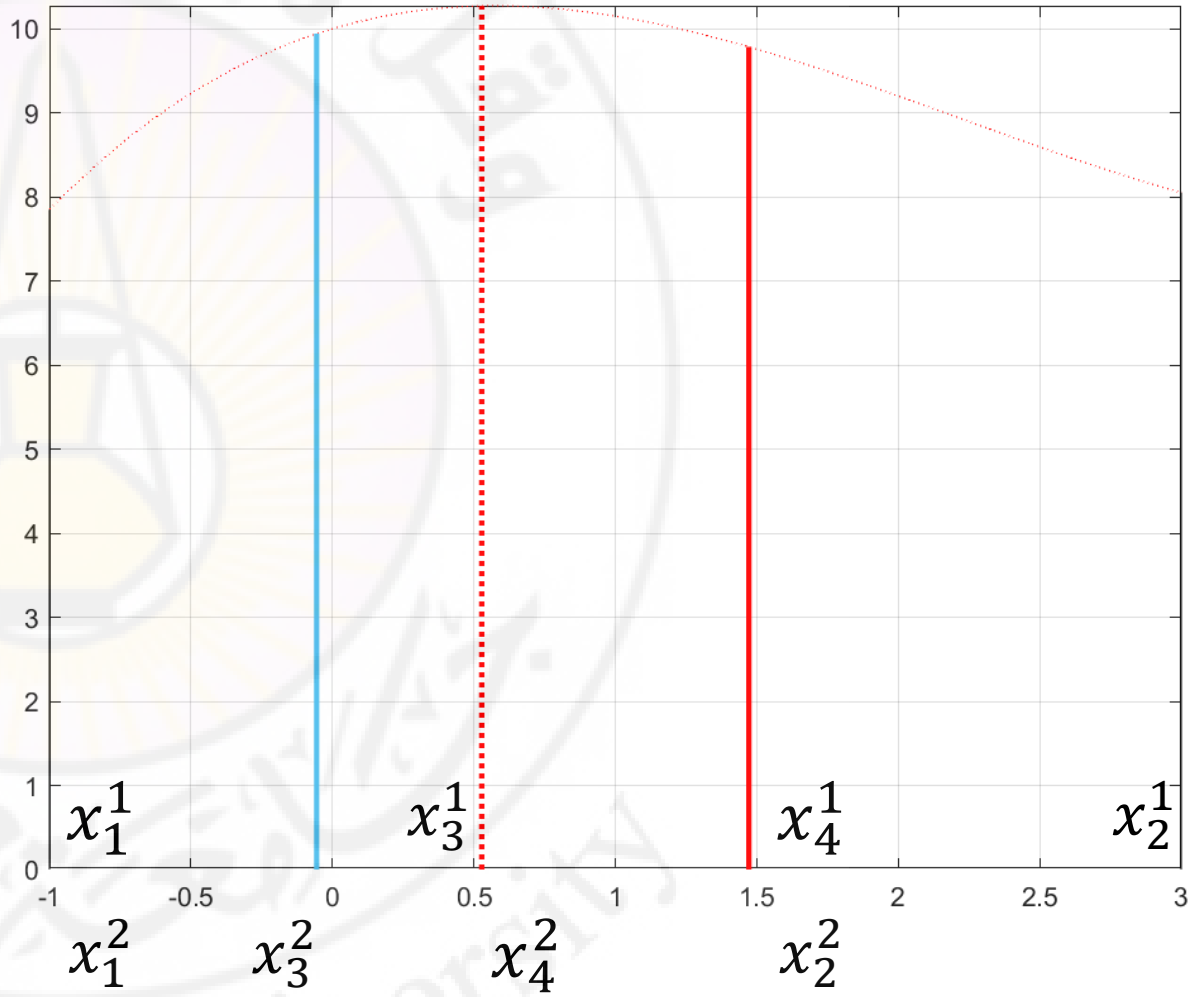
$$\frac{x_4^1 - x_1^1}{x_2^1 - x_1^1}$$

Second iteration

$$\frac{x_2^2 - x_3^2}{x_2^2 - x_1^2} = \frac{x_2^2 - x_3^2}{x_4^1 - x_1^1}$$

Golden ratio when:

$$x_2^2 - x_3^2 = x_2^1 - x_4^1$$



Golden–Section Search

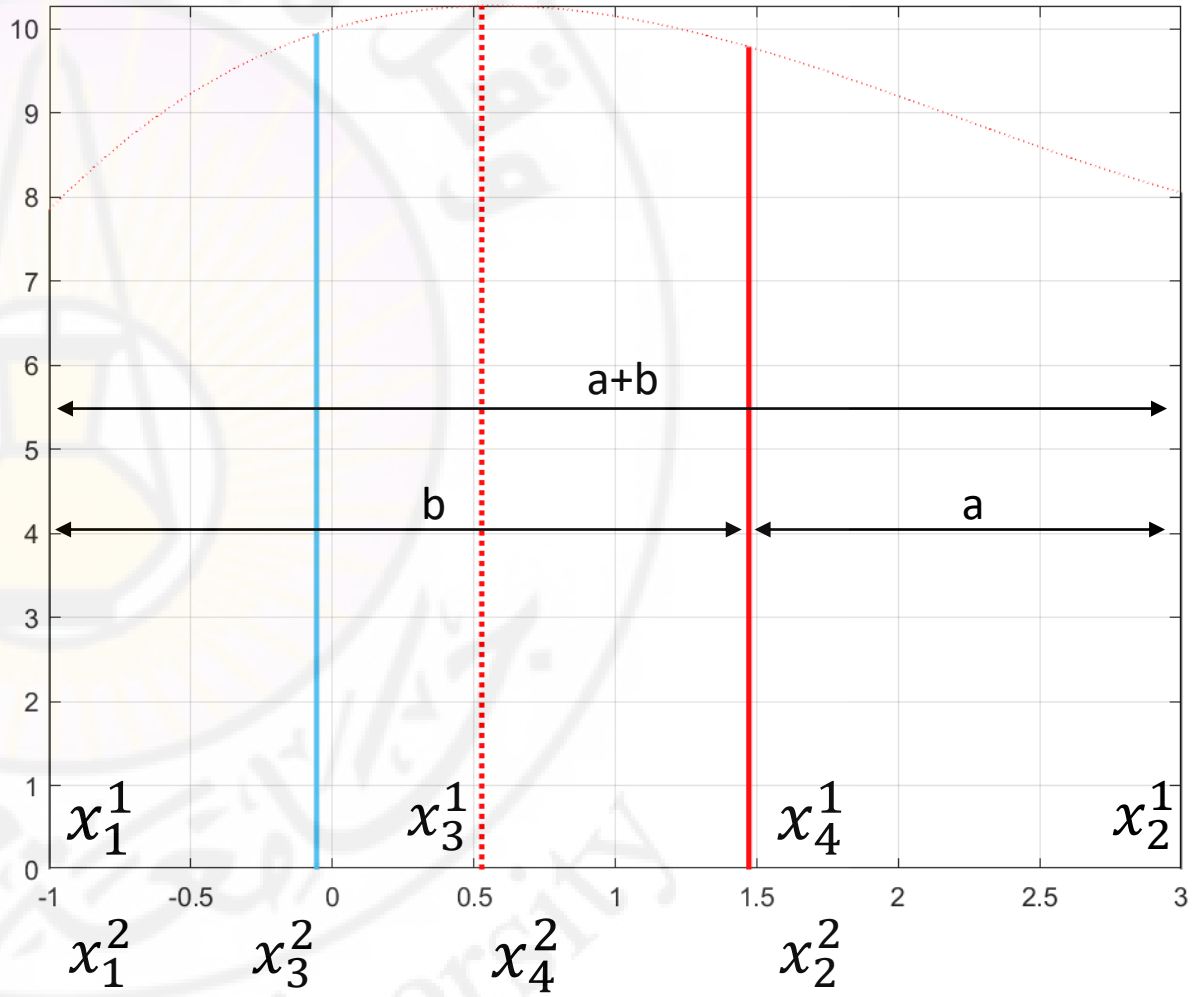
Same ratio

$$\frac{x_4^1 - x_1^1}{x_2^1 - x_1^1} = \frac{x_2^1 - x_4^1}{x_4^1 - x_1^1}$$

$$\frac{b}{a+b} = \frac{a}{b} = Ratio$$

$$\frac{a+b}{b} = \frac{b}{a}$$

$$R + 1 = \frac{1}{R}$$



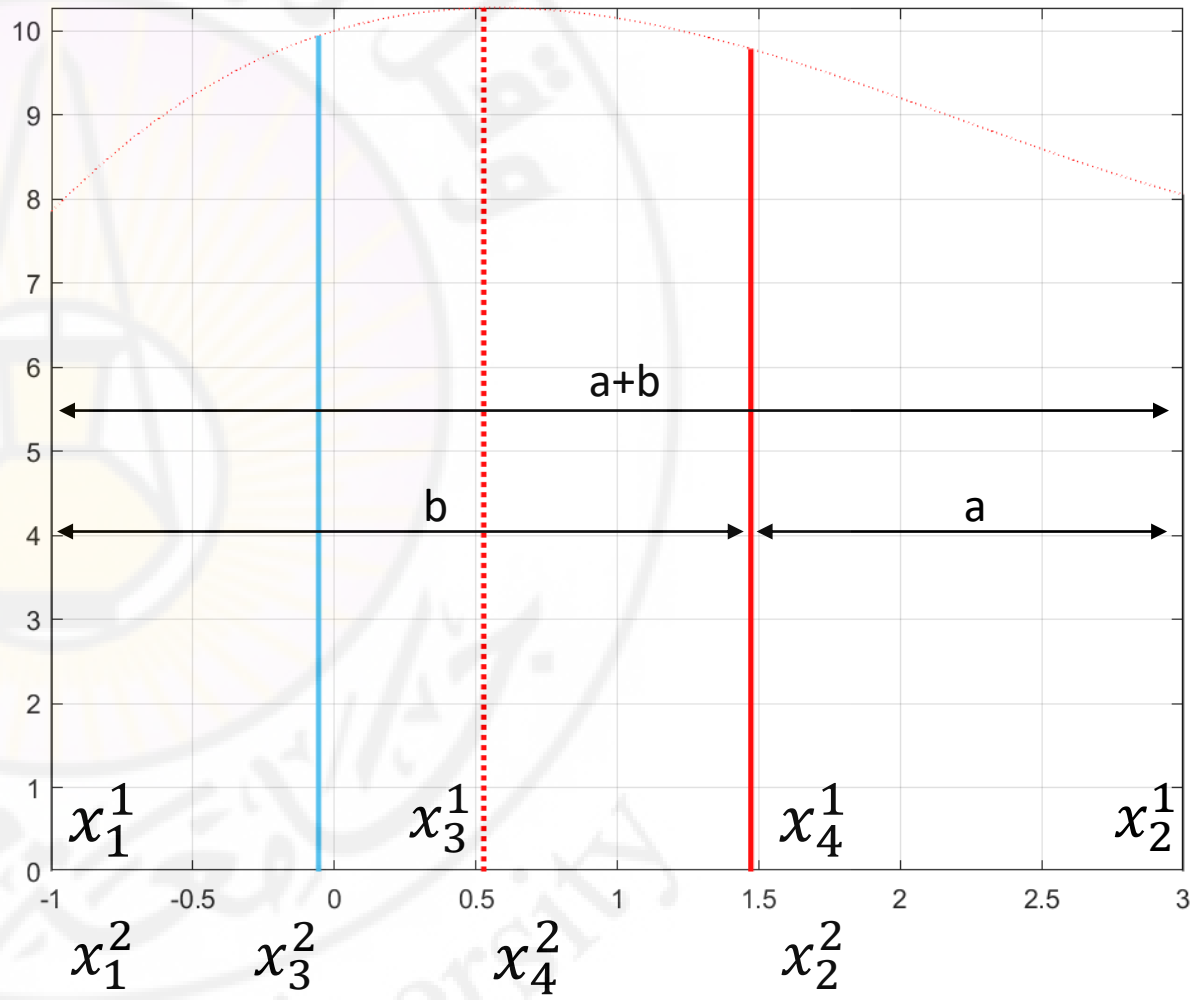
Golden–Section Search

$$R + 1 = \frac{1}{R}$$
$$R^2 + R - 1 = 0$$

$$R_1 = -1.618$$
$$R_2 = +0.618$$

$$\frac{a}{b} = 0.618$$

$$a = 0.382(a + b)$$

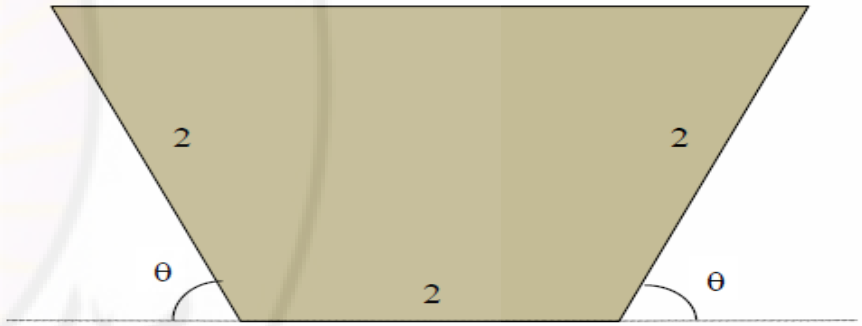


Example

- For gutter with this section:
- Find θ for maximum area

$$A = 2 \times \sin(\theta) \times 2 \times \cos(\theta) + 4 \times \sin(\theta)$$

$$A = 4 \times \sin(\theta)\,(1 + \cos(\theta))$$



Example

```
y = @(x) 4*sin(x)*(1+cos(x));

x1 = 0;
x2 = pi/2;

x rad = 1.0472
x deg = 59.9999
y = 5.1962
```

