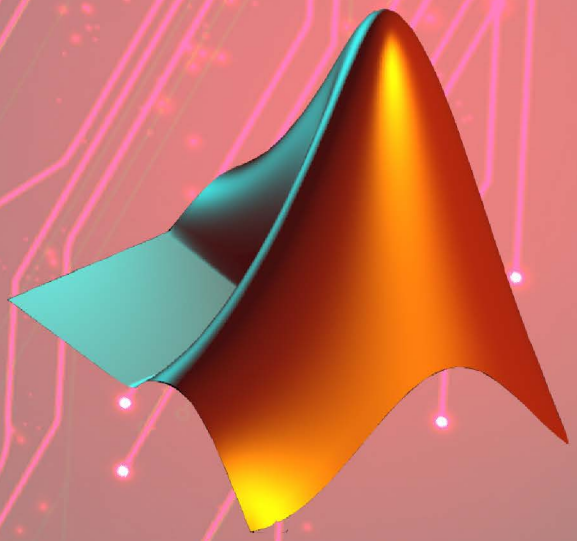


Basics of MATLAB

Mid-term
for mechanical enginering students

By:
Omar Fiasal Nimer

Mech Family
كن أنت التغيير

Important notes:

نضع بين ايديكم شرح لمادة الماتلاب تم كتابة هذه الدوسية بناءً على المادة المقررة للهندسة الميكانيكية وأيضا بحيث تكون مرجاً لك ليس في كورس الماتلاب فقط بل حتى تفيدك باستعمال البرنامج بشكل عام لذلك ان كنت طالباً من قسم اخر يمكنك الاستفادة منها لكن الأفضل الالتزام بما هو مقرر عندك.

الدوسية مشروحة باللغة العربية ومع الحفاظ على المسميات باللغة الإنجليزية وذلك حتى يتم تسهيل الفهم على الطالب من دون احداث خلل او ارتباك بالمسميات.

تم حل جميع الأمثلة في الدوسية وتنفيذها على نسخة

MATLAB 2019a

لذلك اذا كنت تستعمل اصداراً قديما تأكد من انه لا يوجد اختلاف في الأوامر بين الإصدارات

المواضيع التي تم شرحها هنا هي نفسها المقررة في خطة المادة لكن تم الشرح بزيادة في بعض التفاصيل ولذلك لتحقيق فهم جيد للمادة وايضاً لأن بعض الأوامر قد تفيدك في مواد القسم القادمة لذلك من الضروري جدا ان تتأكد مما هو مطلوب من مدرس المادة.

مع تمنياتي لكم بالتوفيق والنجاح

لتقديم أي ملاحظة يرجى التواصل :

- عمر فيصل طه نمر

Omar Nimer

+962797976058

Contents

Overview

What is MATLAB?

ال ماتلاب هو برنامج يستعمله معظم المهندسين بمختلف تخصصاتهم بالإضافة الى طلاب المواد العلمية ويستخدم لتحليل البيانات والمعلومات وصنع برامج تسهل عليهم القيام بكثير من مهامهم وتستطيع حل مسائل معقدة في وقت قصير وبدقة عالية.

أكثر ما يميز الماتلاب هو لغة البرمجة الخاصة به والمبنية على أساس المصفوفات (matrix) ومنها اخذ اسمه (matrix laboratory → MATLAB).

ويتيح لك الماتلاب استخدام ما يسمى بال Tool boxes والتي تحتوي على أدوات وأوامر جاهزة للقيام بوظائف معينة تسهل كتابة البرامج عليك.

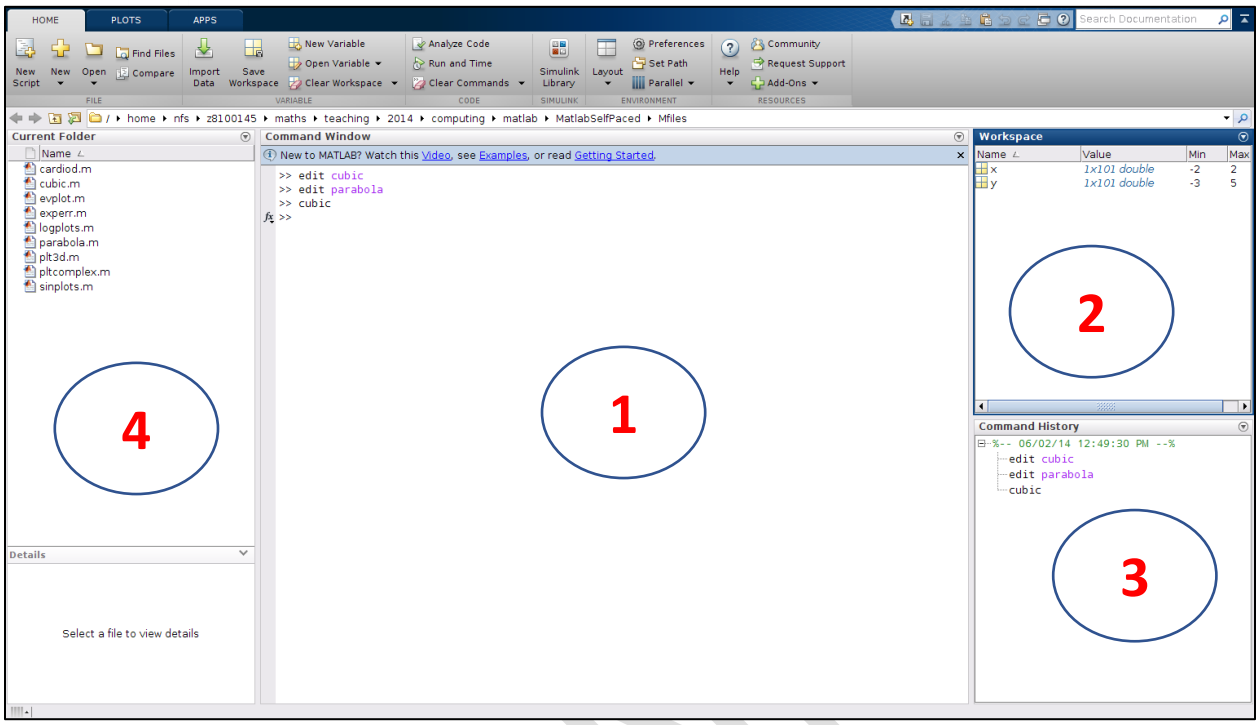
الماتلاب يمكن استخدامه في كثير من المجالات وأهمها:

1- Data analyses
2- Image processing
3- Sound processing
4- Control systems
5- Wireless communication
6- Mathematical uses
7- Deep learning
8- Machine learning
9- Robotics
10-Simulations

وكما تبين لنا ان الماتلاب يستطيع القيام بالكثير فليس هذا ما يميزه فقط بل ويتميز أيضا بسهولة تعلمه وذلك بسبب أنه أوامره تكون قريبة من طريقة تفكير الانسان وتستطيع كتابتها بشكل قريب من الذي يخطر في بالك وتفكر فيه.

والان لتعرف على هذا البرنامج من قرب:

❖ أهم مكونات الشاشة الرئيسية للماتلاب:



1- Command window

هي النافذة الرئيسية وأهم نافذة في الماتلاب والتي يمكننا ان نكتب فيها الأوامر ويقوم بعرض النتائج بصورة مباشرة.

عندما نكتب الأوامر داخل ال command window يقوم بتنفيذها مباشرة و اظهار النتائج ولا يمكننا التعديل على الأوامر التي قد ادخلناها سابقا لذلك عند كتابة البرامج سنقوم بكتابتها في نافذة ال (script) التي سنشرح عنها لاحقا.

2- Workspace

يتم تخزين جميع المتغيرات المخزنة في البرنامج داخل ال workspace ويمكننا رؤية المتغيرات و نوعها و حجمها من هناك.

واذا قمنا بالنقر نقرا مزدوجا على متغير ما تظهر شاشة تمكننا من التعديل في قيمه

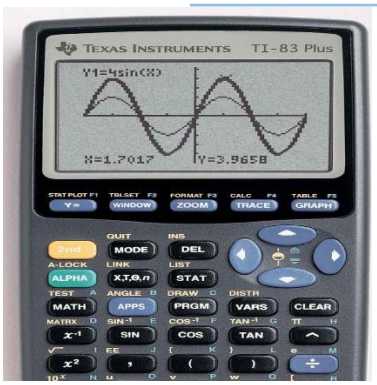
3- Command history

تظهر جميع العمليات التي قام بها الماتلاب والتي تم تنفيذها في ال command history حيث نستطيع مراجعة العمليات التي قمنا بها حتى لو قمنا بمسح ال command window مثلا.

4- Current folder

يقوم بإظهار جميع ملفات الماتلاب (M file) المخزنة في نفس الملف الذي نقوم بالعمل داخله لتسهيل التنقل بين هذه الملفات.

❖ Simple math operations



يستطيع الماتلاب القيام بالعمليات الرياضية البسيطة والمعقدة

حيث يمكننا اعتباره الة حاسبة قوية جدا(super calculate)

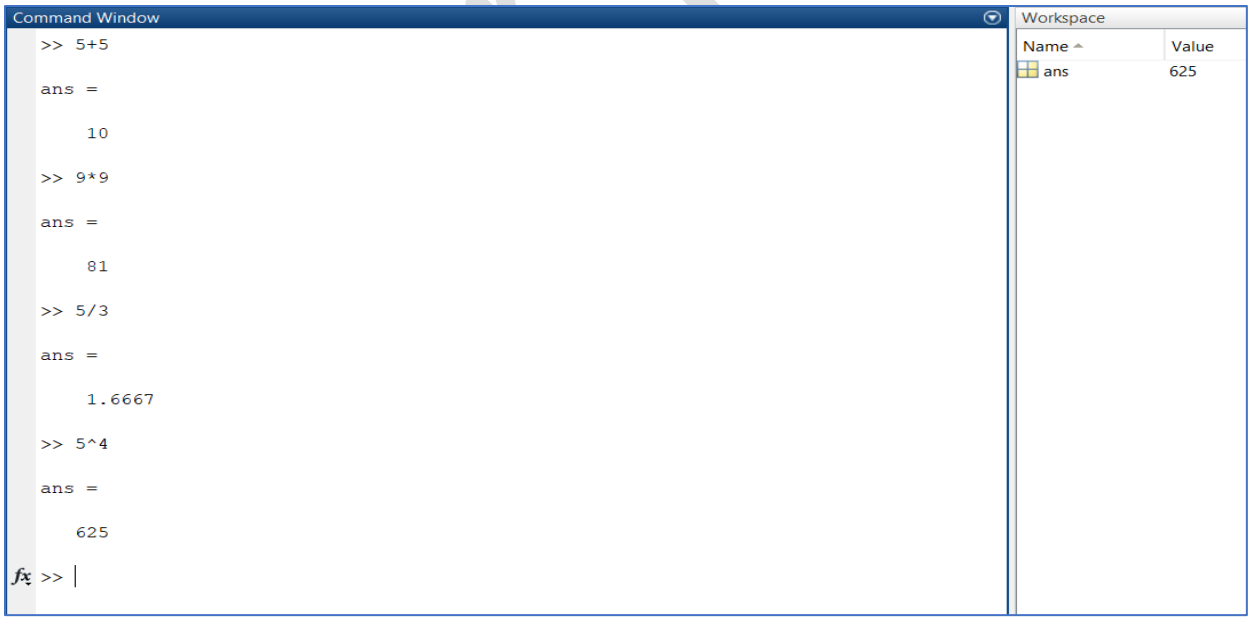
سنقوم بدراسة العمليات المعقدة لاحقا مثل الاشتقاق و التكامل

اما الان سنقوم بتوضيح العمليات البسيطة:

MATLAB can solve simple operations:

Operator	MATLAB	algebra
+	+	4+2=6
-	-	4-2=2
*****	*	2*3=6
/	/	8/2=4
$\mathbf{a^b}$	a^b	2^3=8

Example:



❖ Built in Math constants

In MATLAB	constant
Pi	π
I	$\sqrt{-1}$: imaginary unit
J	$\sqrt{-1}$: imaginary unit
Inf	∞: Infinity
NaN	Not a number
Intmax	Largest value of integer type
Intmin	smallest value of integer type
Ans	Temporary variable containing the most recent answer

المتغير ans

*المتغير **ans** هو متغير مؤقت يقوم البرنامج بافتراضه ويخزن داخله قيمة آخر عملية حسابية قام بها البرنامج .*

**هنا قيمة ans =6 وهي اخر عملية حسابية

**** نلاحظ انه بعد اجراء عملية حسابية أخرى تغيرت قيمة المتغير و أصبحت قيمة اخر عملية حسابية و يمكن ملاحظة ذلك من تغير قيمة ans في نافذة workspace.**

Some Built in functions in MATLAB

Function	MATLAB syntax*
e^x	`exp(x)`
$\sqrt{x}$	`sqrt(x)`
$\ln x$	`log(x)`
$\log_{10} x$	`log10(x)`
$\cos x$	`cos(x)`
$\sin x$	`sin(x)`
$\tan x$	`tan(x)`
$\cos^{-1} x$	`acos(x)`
$\sin^{-1} x$	`asin(x)`
$\tan^{-1} x$	`atan(x)`

- في الماتلاب العمليات مثل sin ,cos تحسب بال radiant أي ان cos(π) =0 لتحويها ل degree نكتبها بصيغة cosd() او sind() .

Variables

قواعد التسمية في الماتلاب بسيطة وي القواعد التي تعلمناها في مادة ال c++ وهي:

1- تستطيع تسمية المتغير باستخدام الحروف الكبيرة والصغيرة والأرقام و (_).

2- يجب ان لا يبدأ اسم المتغير ب رقم او (_).

3- يجب أن لا يكون اسم المتغير محجوزاً.(ليس اسم أمر او function بـ الماتلاب مثل sum, max ...)

4- الماتلاب يستطيع التفريق بين الحروف الصغيرة والكبيرة (المتغير D يختلف تماما عن المتغير d).

Examples:

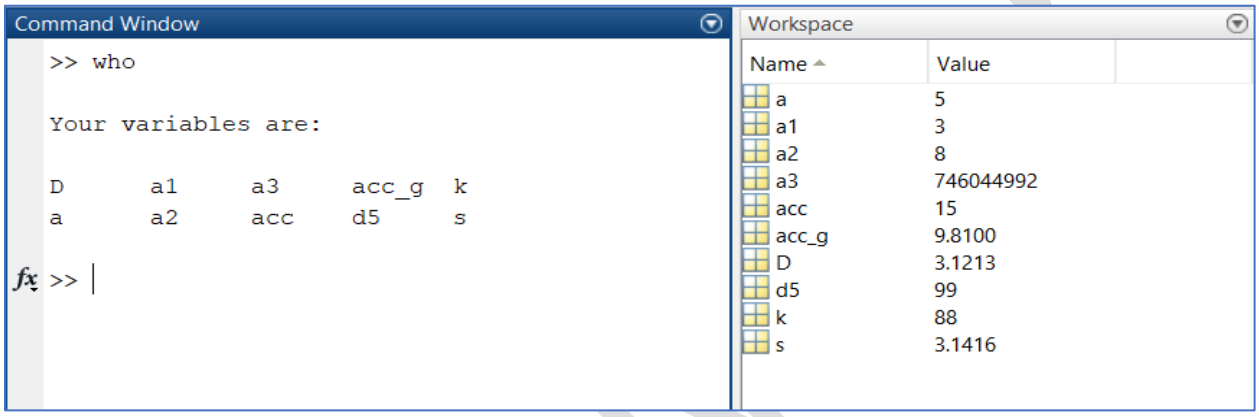
- **X1 , x2 , x3 _____>** **متغيرات صحيحة**
- **Mass_1 , acceleration _____>** **متغيرات صحيحة**
- **_time , 2abc _____>** **متغيرات غير صحيحة لأنها بدأت ب رقم او (_)**
- **s-b _____>** **متغير خاطئ: رمز غير مسموح (-)**
- **clc , max , min _____>** **متغيرات غير صحيحة لأنها محجوزة مسبقا من قبل الماتلاب**

Who /whos

في كثير من الأحيان عندما نكتب برامج كبيرة تصل ال 1000 سطر مثلا ويحتوي على عشرات المتغيرات او برنامج قد كتبناه من وقت طويل نحتاج الى معرفة المتغيرات التي تم استعمالها في البرنامج و أنواعها و قيمها المخزنة لذلك نلجأ الى الأمرين who , whos

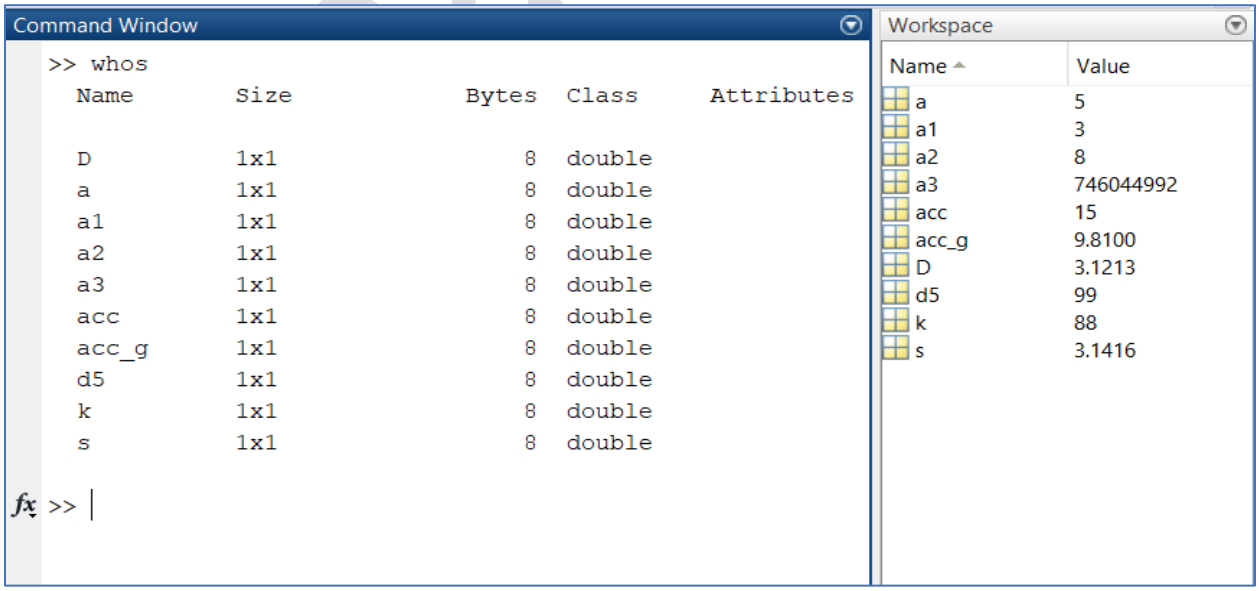
1- Who

يقوم الامر who بإظهار جميع المتغيرات المخزنة داخل البرنامج دون ذكر أي تفاصيل لها :



2- Whos

- **يقوم بإظهار جميع المتغيرات التي تم تعريفها داخل البرنامج مع ذكر كامل تفاصيلها (اسمها ونوعها و وحجمها في الذاكرة):**



clc and clear and close commands

❖ clc

بعد كتابة برامج طويلة واجراء عمليات حسابية عدة على الcommand window قد تريد مسح ما عليها من اجل تنظيم افكارك وزيادة الترتيب لكن لا تريد ان تمسح قيم المتغيرات المخزنة أو تأثر على البرنامج الذي كتبته فتلجأ للأمر clc.

يقوم هذا الأمر بمسح ال command window من غير التأثير على البرنامج او التغيرات.

❖ clear commands

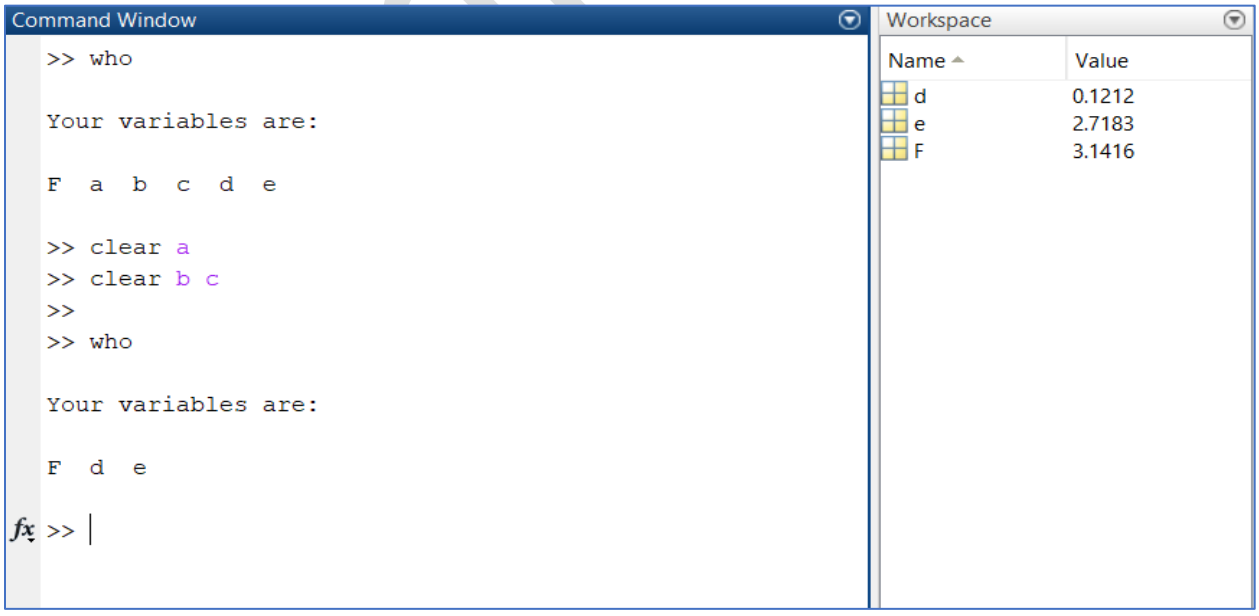
عند الانتهاء من استخدام متغير قد تريد ان تخذف هذا المتغير لإعادة استعماله مثلا ولزيادة التنظيم او التأكد من عدم وقوع أخطاء عند إعادة استخدامه في البرنامج

نستطيع استخدامه بصيغتين:

1- Clear variable

نستخدم امرclear بهذه الصيغة من اجل مسح متغير واحد او عدة متغيرات عن طريق ذكر اسمائهم

(clear a , clear b , clear acceleration ,....)



** نلاحظ انه تم حذف المتغيرات التي حددناها (a,b,c) بعد الامر clear وانه يمكننا حذف اكثر من متغير بنفس الامر.

2- Clear all

نستخدم clear all من اجل مسح جميع المتغيرات المخزنة في البرنامج.

** نلاحظ انه تم نسح جميع المتغيرات بعد clear all.

❖ Close commands

نستخدم امر close من اجل إغلاق الصور والجداول المفتوحة في البرنامج ونستطيع استخدامه بالصور المذكورة في الأعلى.

1- Close variable

نستخدم أمر close بهذه الطريقة من اجل إغلاق الصور والجداول المفتوحة في البرنامج عن طرق ذكر اسمها.

2- Close all

نستخدم امر close all من اجل إغلاق جميع الصور والجداول المفتوحة في البرنامج دفعة واحدة.

help commands

الماتلاب برنامج ضخم ويحتوي على الاف الأوامر والخيارات التي يحتاجها الشخص عند كتابته لاي برنامج ... ومن غير المعقول ان يستطيع أي شخص بحفظها وحفظ هذه الأوامر وصيغتها لذلك تعد لائحة help من اهم الموجودات في برنامج MATLAB واكثرها فائدة لذلك يجب تعلم استخدامها لتذكر صيغة امر في حال نسيانك لهذا الأمر.

يوجد ثلاث طرق لاستفادة من help وهي:

1- To write in the command window

" Help the thing that we want to search about"

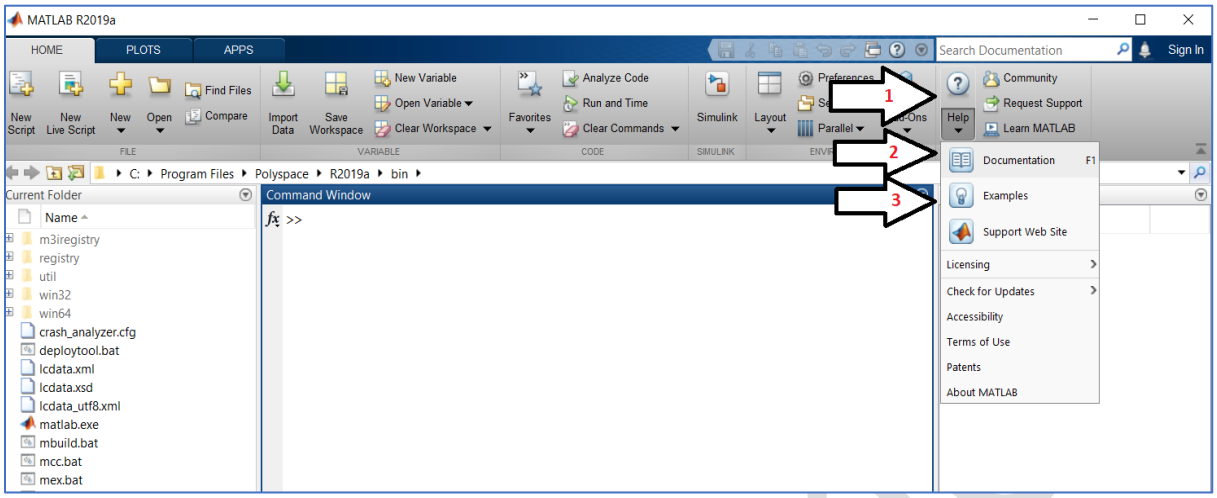
- If we want to search about the exponential function as an example:

Command Window

```
>> help exponential
--- help for xregusermod/exponential ---

 exponential exponential user defined model for MBC

  varargout= exponential(U,x,varargin)

  To use this function, the command
    U2= checkin(xregusermod,'exponential',xtest)
  must be run. The last argument is a column based input matrix to
  test the function.

fx >>
```

**Same thing goes for (sin , cosine , tangent and any other function).

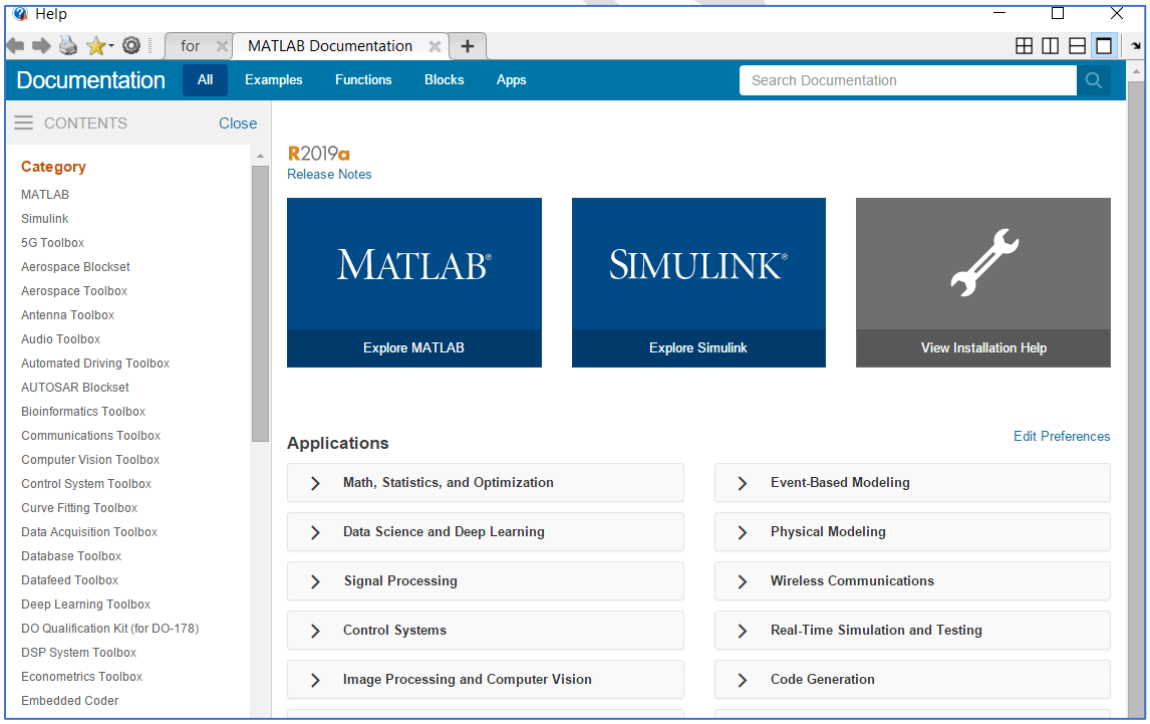
2- من خيار help في لائحة HOME.



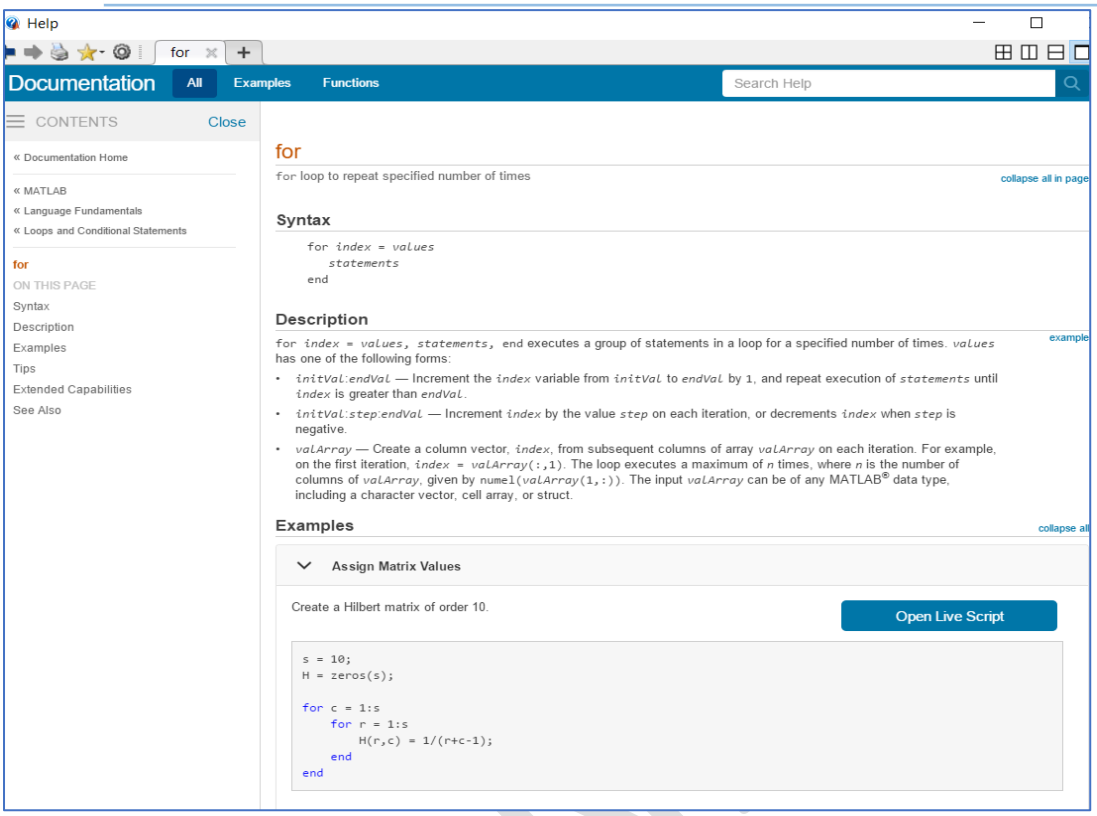
1- لائحة help

2- للبحث في أوامر البرنامج و ال functions مثل ال (if else, for ,while, plot)

3- للبحث عن امثلة عن الاقترانات او الأوامر لمعرفة كيفية تنفيذها.



(لائحة help)

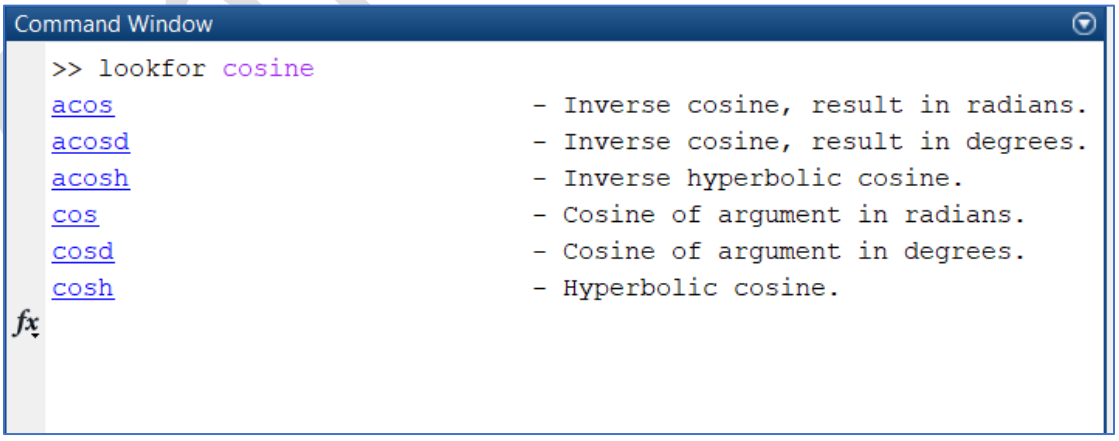


**ومن الخصائص الجميلة في الماتلاب انه يمكنك النسخ من لائحة help حيث يمكنك نسخ صيغ الأوامر والاقترانات ووضعها في البرنامج (ويمكنك استخدامها من اجل توفير الوقت أحيانا او من اوجل التأكد من عدم الوقوع في خطأ اثناء كتابة الأمر).

**مثلا في المثال الموجود في الأعلى يمكنك نسخ صيغة ال for ووضعها في البرنامج الخاص بك.

3) lookfor " the thing that we want to search about" .

-Lets lookfor cosine as an example:



** نلاحظ انه قام بإعطاء جميع أوامر ال cos الممكنة (inverse, in degree, in rad, cosh)

rounding

عندما تقوم بكتابة برنامج قد تحتاج أحيانا للتعامل فقط مع اعداد صحيحة integers فتريد ان تكون مدخلات برنامجك او مخرجاته اعداداً صحيحة، وفي هذه الحالة نلجأ الى التقريب (rounding).

نستطيع التقريب باستخدام أوامر عدة ولكل امر منها غرضه الخاص وهي:

1- round (a) ⟶ round "a" toward the nearest integer

- **يقوم بتقريب العدد إلى اقرب عدد صحيح له.. مثال :**

round (3.4) = 3

round (3.9) = 4

2- fix (a) ⟶ round "a" toward zero

- **يقوم بتقريب العدد إلى العدد الأقرب إلى الصفر.. مثال:**

fix (3.9) = 3 **لأنها الأقرب للصفر**

fix (-5.8) = -5 **لأنها الأقرب للصفر**

3- ceil (a) ⟶ round "a" toward +∞

- **يقوم بتقريب العدد إلى العدد الأقرب إلى (+∞) أي العدد الأكبر.. مثال:**

ceil (3.1) = 4

ceil (-5.7) = -5

4- floor (a) ⟶ round "a" toward -∞

- **يقوم بتقريب العدد إلى العدد الأقرب إلى (-∞) أي العدد الأكبر.. مثال:**

floor (3.1) = 3

floor (-5.7) = -6

Imaginary and real numbers

- **MATLAB can deal with complex numbers and do operations on it without any problem:**

- **Some operations we can do on them:**

let's we assume a = 5 -2i

1- abs (a)=

this function gives us the absolute value of the variable.

what MATLAB dose ⟶ $\sqrt{(5^2 + 2^2)}$ **= 5.3852**

examples:

value of 'a'	abs (a)
a = 6 +4i	7.2111
a = 0.2 – 5i	5.0040
a = 2 +2j	2.8284
a= -5-4i	6.4031

2- angle(a)=

this function gives the angle that results from ($\tan^{-1}\frac{imaginary\ part}{real\ part}$ **) .**

in our example ⟶ $(\tan^{-1}\frac{-2}{5})$

examples:

value of 'a'	angle (a) (in rad)
a = 6 +4i	-0.3805
a = 0.2 – 5i	0.5880
a = 2 +2j	-1.5308
a= -5-4i	-2.4669

****لاحظ انه جواب الزوايا بال "راديان" إي من (π+،π-).**

3- real (a)=

this function gives the value of the real part of the complex number.

examples:

value of 'a'	real (a)
a = 6 +4i	6
a = 0.2 – 5i	0.2
a = 2 +2j	2
a= -5-4i	-5

4- imag (a)=

this function gives the value of the imaginary part of the complex number.

examples:

value of 'a'	imag (a)
a = 6 +4i	4
a = 0.2 – 5i	-5
a = 2 +2j	2
a= -5-4i	-4

- **There are two was to define complex numbers in MATLAB:**

1- a= real number + imaginary number

a= 2 + 2 i

2- a= complex (,)

the value of the real — the value of imaginary

a= complex (3 , 5) ⟶ a= 3.000 + 5.000 i

math operations and format

- ولان قبل التعريف بال matrix يجب علينا التعرف على بعض العمليات الرياضية المهمة التي يتم استعمالها بكثرة.

In MATLAB	Function
log10(a)	**Logarithm base 10 for (a)**
log (a)	**ln (a)**
sqrt (a)	**Square root ($\sqrt{a}$)**
expo(a)	**Exponential function $e^{(a)}$**
factorial (a)	**a Factorial = a!**
sin (a)	**Sine (a)**
cos (a)	**Cosine (a)**
tan(a)	**Tangent (a)**
sind (a)	**Sin (a) in degree**
cosd (a)	**Cos (a) in degree**
tand(a)	**tan(a) in degree**
asin (a)	**Arc Sine (a) (inverse)**
acos (a)	**Arc Cosine (a) (inverse)**
atan(a)	**Arc Tangent (a) (inverse)**
acosd(a)	**Arc Cosine(a) in degrees**

** في امر log يكون الرقم بعد كلمة log و قبل القوس يدل على أساس اللوغاريثم وفي حال عدم كتابة أي رقم يعتبر بان الأمر هو ln.

Format long /short

يستخدم الأمر format بصيغتي من اجل التحكم بعدد المنازل العشرية التي يظهرها الماتلاب في المتغيرات حيث نجعله يظهر عدد منازل أكثر او اقل.

Format long

Example:

- جميعنا نعرف ان باي (π) هو عدد غير نسبي غير منتهي وفي العادة يقوم الماتلاب بإظهار 4 منازل بعد الصفر فإذا اردنا أن نظهر عدد منازل أكثر نستعمل هذا التنسيق.

```
Command Window
>> a= pi

a =

    3.1416

>> format long
>> a= pi

a =

   3.141592653589793

fx >> D
```

Format short

- نستخدم Format short من اجل اظهار الأرقام بصورة أقصر

```
Command Window
>> a= pi

a =

   3.141592653589793

>> format short
>> a= pi

a =

    3.1416

fx >>
```

- **اذا تم وضع حرف e بعد الصيغة مثلا: format long e سيقوم بكتابة النتيجة مع ال exponent مثال:

```
Command Window
>> format long e
>> pi

ans =

     3.141592653589793e+00
```

Semicolon: (;)

- نقوم باستخدام الفاصلة المنقوطة (semicolon)(;) في نهاية أي امر من اجل تنفيذ ذلك الأمر من دون ان نظهره على ال command window ونقوم بذلك من اجل زيادة الترتيب عندما نقوم بكتابة برامج طويلة وعندما نريد إخفاء كل شيء لا يتعلق بالنتيجة النهائية.

```
Command Window
>> a=5;
>> b=0.2525;
>> c=9*2/3;
>> d=5

d =

     5

fx >>
```

Workspace

Name	Value
a	5
b	0.2525
c	6
d	5

**نلاحظ انه تم اجراء أول 3 عمليات من دون اظهار الناتج بسبب وجود ال(;) اما في العملية الرابعة تم تنفيذها وعدم إظهار الناتج بسبب عدم وجود فاصلة منقوطة.

Section 1 end

Section 2

Matrix

Declaring a matrix

Matrices is the base of MATLAB in fact MATLAB is a short for matrix laboratory

- **تعريف ال matrix او ال array :**

1- تعريف الصفوف عن طريق مسافة او فاصلة بين المدخلات

- **A= [1 2 3 4 5 6] او A=[1,2,3,4,5,6]**

Result:

```
A =

     1     2     3     4     5
```

2- لتعريف صفوف جديدة نستخدم الفاصلة المنقوطة

- **A=[1;2;3;4;5]**

Result:

```
A =

     1
     2
     3
     4
     5
```

- **والان مثال لتعريف مصفوفة 3x3 :**

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

A =

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

- **لتعريف المصفوفات الطويلة نستخدم الصيغة التالية**

A=[starting number : jump : final number]

البداية **مقدار القفزة بين كل عنصر و عنصر** **النهاية**

Example: a= [0 : 2 : 12] , b=[1:2:]

```
>> a=[0:2:12]

a =

     0     2     4     6     8    10    12

>> b=[1:2:10]

b =

     1     3     5     7     9
```

**نلاحظ انه في المصفوفة(b) توقفت عند الرقم 9 وليس 10 لأنه عندما نضيف لأخر رقم وهو 9 سيصبح الناتج 11 (يعني سيتجاوز ال 10) لذلك توقفت المصفوفة عند 9.

- **أمر(, ,) linspace**

نستخدم هذا الأمر عندما نريد انشاء مصفوفة ونعرف البداية و النهاية و عدد العناصر الموجودة في المصفوفة(ولا نعرف مقدار القفزة بين العناصر) فيقوم بإنشاء مصفوفة ويقوم بتقسيم الطول بشكل متساوي بين العناصر.

Example: a = linspace (1 , 5 , 9)

البداية **النهاية** **عدد العناصر التي نريدها بالمجموعة**

في هذا المثال حددنا البداية (1) و النهاية (5) و عدد العناصر في المصفوفة (9) لذلك سيقوم البرنامج بتقسيمها ل تسعة أجزاء متساوية (والتي سيكون البعد بين عناصرها 0.5).

```
>> a= linspace (1 , 5 , 9)

a =

    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000    5.0000
```

- **أمر logspace(, ,):**

نستخدم هذا الأمر عندما نريد انشاء مصفوفة عناصرها مرفوعة للأساس 10 بمعلومية عدد العناصر المصفوفة في المصفوفة وأول قيمة وآخر قيمة.

Example: a = logspace (1 , 2 , 5)

1: البداية — 2: النهاية — 5: عدد العناصر التي نريدها بالمجموعة

يعني أول عنصر راح يكون $10^{(\text{البداية})}$ والي هو في المثال $10^0 = 1$ واخر عنصر هو $10^{(\text{النهاية})}$ والي في المثال $10^2 = 100$.

```
>> a=logspace ( 0 , 2 , 5 )

a =

    1.0000    3.1623   10.0000   31.6228  100.0000
```

10^0 $10^{0.5}$ 10^1 $10^{1.5}$ 10^2

Operations on matrices

- **الجمع و الطرح**

يمكننا القيام بذلك ببساطة حيث سيقوم بجمع او طرح كل عنصر بالعنصر المقابل له ولذلك يجب ان تكون احجام ال matrix متساوية عند القيام بالجمع او الطرح

Example 1:

$$A=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix} \quad \text{and} \quad B=\begin{bmatrix}2&2&2\\2&2&2\\2&2&2\end{bmatrix}$$

Then:

$$A+B=\begin{bmatrix}3&4&5\\6&7&8\\9&10&11\end{bmatrix} \quad \text{and} \quad A-B=\begin{bmatrix}-1&0&1\\2&3&4\\5&6&7\end{bmatrix}$$

- **الضرب**

يمكننا اجراء عملية ضرب المصفوفات في الماتلاب أيضا ولكن يجب ان ننتبه لشرط الضرب و هو ان يتساوى عدد أعمدة اول مصفوفة مع صفوف المصفوفة الثانية.

Example 2:

C =[1 2 3] and D= $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

Then:

C * D = [22 28]

لكن اذا اردنا قلب العملية و اجراء العكش راح يعطينا error لأنه عدد أعمدة الأولى لا يساوي عدد صفوف الثانية.

D*C=

```
Error using  *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix
matches the number of rows in the second matrix. To perform elementwise multiplication, use '.*'.
```

➢ القسمة

لا يوجد هناك شيء في المصفوفات اسمه قسمة المصفوفات ولكن نستخدم مكانه عملية أخرى وهي الضرب بال inverse فعندما نقوم بقسمة matrices هذا يعني انه البرنامج يقوم بضربها بال inverse لهذه المصفوفة.

Example 3:

Assume A and B the same as in the example 1.

a/b=

```
ans =

   -0.7500    0.8333   -0.0833
   -0.7500    0.8333   -0.0833
   -0.7500    0.8333   -0.0833
```

➢ الضرب و القسمة عنصر بعنصر:

اذا كان يوجد لدينا مصفوفتان في الماتلاب و اردنا أن نضرب احداهما بالأخرى ولكن ليس بطرقة ضرب المصفوفات و انما كل عنصر مع العنصر المقابل له نستطيع فعل ذلك بسهولة

- فقط نقوم بوضع نقطة قبل الضرب او القسمة هكذا (.*) (./)

A= $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ and B = $\begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}$

Then

A.*B =

```
ans =

     2     4     6
    12    15    18
    28    32    36
```

and A./B =

```
ans =

    0.5000    1.0000    1.5000
    1.3333    1.6667    2.0000
    1.7500    2.0000    2.2500
```

- Predefined functions for vectors:

لنأخذ المثال التالي :

a = [1 : 10]

```
a =

     1     2     3     4     5     6     7     8     9    10
```

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

```
b =

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

In MATLAB	العملية	الناتج من المثال في الأعلى
min(a)	إيجاد أصغر قيمة	1
max(a)	إيجاد أكبر قيمة	10
mean(a)	المتوسط الحسابي	5.500
median(a)	الوسيط الحسابي	
sum(a)	مجموع عناصر ال vector	55
prod(a)	حاصل ضرب عناصر ال vector	
cumsum(a)		ans= 1 3 6 10 15 21 28 36 45 55
cumprod(a)		
length(a)	اكبر عدد لصفوف او اعمدة المصفوفة يعني اذا الاعمدة اكثر بظهر عددهم و هو الي 10	10
Size (b)	حجم المصفوفة (عدد الاعمدة، عدد الصفوف)	3 3
b’	Transpose	سنقوم بشرح الامر مع امثلة

In MATLAB	العملية	الناتج من المثال في الأعلى
det (b)	Determinant for b المحددة	-9.5162e-16
triu (b)	Upper triangle for b	سنقوم بشرح الامر مع امثلة
tril (b)	Lower triangle for b	سنقوم بشرح الامر مع امثلة
eig( )	eigen value	
std(a)	الانحراف المعياري للقيم	ans = 3.0277
Sort(a)	ترتيب القيم تصاعدياً	ans= 1 2 3 4 5 6 7 8 9 10
Sort(a, 'descend')	ترتيب القيم تنازلياً	ans= 10 9 8 7 6 5 4 3 2 1

➢ Size ( )

- يقوم الامر بإعطائك حجم مصفوفة ما بترتيب صف ثم عامود و يمكننا الاستفادة منها عندما نعرف مصفوفات بحجم ضخم و نريد معرفة ابعادها.

ويمكننا تعريف النتائج في متغيرات كما في المثال التالي:

```
>> [ raws , columns ] = size(b)

raws =

     3


columns =

     3
```

➢ Det ( )

- يستخدم الأمر من اجل إيجاد محددة ما لمصفوفة (determinant)

Example:

```
b =

     1     2    31     7
     4     5    12     7
     7     8     9     9
     5    56     5     5

>> det(b)

ans =

   5.6920e+03
```

➢ Transpose ( ) , x'

- يقوم هذا الامر بقلب المصفوفة (بخلي الصف عامود و العامود صف) ونسطيع تنفيذه عن طريق الامر ( ) Transpose او نضع اسم المصفوفة وبعدها (').

```
>> b'

ans =

     1     4     7     5
     2     5     8    56
    31    12     9     5
     7     7     9     5
```

```
>> transpose(b)

ans =

     1     4     7     5
     2     5     8    56
    31    12     9     5
     7     7     9     5
```

➢ Triu ( )

اختصار (triangle upper)

- نستخدم هذا الامر عندما نريد اخذ المثلث العلوي من المصفوفة ما ويقوم بفرض باقي القيم الموجودة في المصفوفة اصفار.

➢ Tril ( )

اختصار (triangle lower)

- نستخدم هذا الامر عندما نريد اخذ المثلث السفلي من المصفوفة ما ويقوم بفرض باقي القيم الموجودة في المصفوفة اصفار.

Example:.

```
>> b=[1 2 31 7 ; 4 5 12 7; 7 8 9 9 ; 5 56 5 5]

b =

     1     2    31     7
     4     5    12     7
     7     8     9     9
     5    56     5     5

>> triu(b)

ans =

     1     2    31     7
     0     5    12     7
     0     0     9     9
     0     0     0     5

>> tril(b)

ans =

     1     0     0     0
     4     5     0     0
     7     8     9     0
     5    56     5     5
```

➢ Max & Min ( )

- نستخدم الأوامر ( )min و ( )max عندما نريد إيجاد أكبر أو أصغر قيمة في مصفوفة ما ولنتعرف على طريقة عمل الاقتران لننظر الأمثلة التالية:

1. اذا كانت المصفوفة تتكون من صف واحد أو عامود واحد فقط يقوم الأمر بأخذ اكبر/ أصغر قيمة قي ال مصفوفة.

```
b =

   112    75    67    96    95    94    94    98    97    94    79     8     8

>> max (b)

ans =

   112

>> min (b)

ans =

     8
```

2. اذا كانت المصفوفة تتكون من عدة صفوف و أعمدة فإنه يقوم بأخذ أكبر/أصغر قيمة من كل عامود على حدى.

```
a =

     6    -4     8
    -5    -3     7
    14     9    -5

>> max (a)

ans =

    14     9     8
```

❖ **يمكننا استخدام الامر أيضا بصيغة أخرى وهي " `[v,k] = max(A)` " حيث سيقوم بتخزين اكبر قيمة من كل عامود في المتغير v و يقوم بتخزين موقعها(في أي صف تقع) في المتغير k. (نفس الاشي بالنسبة ينطبق للامر min).**

```
>> [v,k]= max(a)

v =

    14     9     8


k =

     3     3     1
```

Matrix editing:

يمكننا استدعاء عنصر معين في ال matrix عن طريق ذكر اسم المتغير و الصف والعامود الذي يقع فيه العنصر:

Example:

```
>> a=[1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]

a =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

a (2 , 3) = 7

صف عامود

a (3 , 1) = 9

لاستدعاء صف كامل او عامود كامل نقوم بالتالي

- لاستدعاء صف نقوم بتحديد الصف نقوم بتحديد رقمه ونضعه في خانة الصفوف ونضع مكان الأعمدة (:)
ولنأخذ الصف الثالث كمثال:

- a (3 , :) = 9 10 11 12

- لاستدعاء عامود نقوم بتحديد العامود نقوم بتحديد رقمه ونضعه في خانة الاعمدة ونضع مكان اللصفوف (:)
ولنأخذ العامود الثاني كمثال:

- a (: , 2) = 2
6
10
14

- لاستدعاء صفوف و أعمدة معينة في نفس الوقت:
نحدد الصفوف و الأعمدة التي نريدها بالشكل التالي (النهاية : البداية , النهاية : البداية) وكمثال فلنأخذ اول 3 أعمدة و الصفين الثالث و الرابع:

- a (3:4 , 1:3) = 9 10 11
13 14 15

❖ التعديل على ال matrices

في كثير من الأحيان نحتاج للتعديل على المعلومات الموجودة داخل عنصر في المصفوفة او في صف ما داخلها لذلك يجب ان نتعرف على كيفية التعديل عليها.

❖ يمكننا التعديل على عنصر ببساطة عن طريق استدعائه ووضع قيمته الجديدة
مثلا لو اردنا تعديل العصر الموجود في الصف الرابع و العامود الثاني سوف نعدله بهذه الطريقة

- a(4 , 2)= 20

```
>> a( 4 ,2)= 20

a =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    20    15    16
```

❖ لو اردنا تعديل صف نقوم باستدعائه ثم تغيير قيمه

- a (1 , :) = [5 5 5 5]

```
>> a( 1 , : )=[ 5 5 5 5 ]

a =

     5     5     5     5
     5     6     7     8
     9    10    11    12
    13    20    15    16
```

❖ لو اردنا تعديل عامود نقوم باستدعائه و تغيير قيمته ايضاً

- a (: , 4) = [3; 3 ; 3 ; 3]

```
>> a( : , 4)=[ 3 ; 3 ; 3 ; 3]

a =

     5     5     5     3
     5     6     7     3
     9    10    11     3
    13    20    15     3
```

** يجب ان ننتبه لعدد العناصر المدخلة بالصف او العامود الجديد ان تكون مساوية لما قبلها يعني (اذا كان عندك الصفوف كلها فيها 5 عناصر الصف الجديد يجب ان يحتوي على 5 أيضا و الا راح يعطيك error).

❖ حذف وإضافة الصفوف و الأعمدة:

1- إضافة صف او عمود

يمكننا حد صف او عامود من مصفوفة تم تعريفها عن طريق الخطوات التالية:

- لنفرض وجود المصفوفة 'a' 3X3 واردنا ان نضيف لها عامود رابع يحتوي على القيم (1 2 3) نقوم بذلك بهذه الصورة:

```
a =

     1     2     3
     4     5     6
     7     8     9

>> a( : , 4 )= [1 2 3]

a =

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

** يعني زي كانه بدنا نستدعي العامود الرابع و بنعرف داخله قيم جديدة .

ملاحظة:

- نفس المثال ينطبق على الصفوف لكن بدل تعريف عامود رابع نعرف صفاً رابعاً.
- اذا ضفنا صف او عامود الفرق بينه و بين اخر صف او عامود اكثر من واحد راح يفرض الصفوف و الأعمدة الي بينهم اصفار.

ولنأخذ المصفوفة السابقة 3X3 مثال ونضيف لها صفاً خامسا:

```
a =

     1     2     3
     4     5     6
     7     8     9

>> a(5 , :)=[ 1 2 3 ]

a =

     1     2     3
     4     5     6
     7     8     9
     0     0     0   <-----
     1     2     3
```

** نستطيع ان نلاحظ ان تم فرض الصف الرابع اصفار والي هي بين اخر سطر وبين السطر الي احنا اضفناه.

2- حذف صف او عامود:

يمكننا حد صف او عامود من مصفوفة تم تعريفها كالتالي:

- نقوم باستدعاء الصف او العامود الذي نريد ان نقوم بحذفه ونساويه بقوسين فارغات " [ ] ".

لنرجع للمثال السابق ونقوم بحذف الصف الثاني ثم العامود الثاني:

```
a =

     1     2     3
     4     5     6
     7     8     9

>> a( 2 , : )= []

a =

     1     2     3
     7     8     9

>> a( :, 2 )= []

a =

     1     3
     7     9
```

Pre-defined matrices

أحينا نحتاج الي استخدام matrix معينة أكثر من مرة في البرنامج و تعريفها في متغيرات مختلفة وهذه المصفوفات معروفة ومحددة لدينا مثل identity matrix, zeros matrix لذلك توجد أوامر محددة في الماتلاب تساعدنا على تعريف هذه المصفوفات بسرعة ومن دون الحاجة الى تعريفها عنصرا عنصراً .

ومن اهم هذه الأوامر:

❖ Zeros( )

يقوم هذا الامر بإنشاء مصفوفة جميع عناصرها أصفار (zero) ونستطيع تحديد حجم المصفوفة داخل الأقواس من وتتبع للقواعد التالية:

1- اذا وضعنا رقم واحد فقط داخل الأقواس يعتبر ان هذا الرقم هو عدد الصفوف و هو نفسه عدد الأعمدة.

Example:

```
>> a= zeros(3)

a =

     0     0     0
     0     0     0
     0     0     0
```

** لأننا وضعنا الرقم 3 بين الأقواس قام بافتراض انها 3X3

2- اذا اردنا ان نعرف مصفوفة ب عدد صفوف و أعمدة مختلف نضع رقمين داخل الأقواس بينهم فاصلة حيث يشير الرقم الأول لعدد الصفوف و الثاني لعدد الأعمدة.

مثلا لو افترضنا اننا نريد انشاء مصفوفة كل عناصرها اصفار و تحتوي على صفين واربع أعمدة.

Example:

```
>> b= zeros(2,4)

b =

     0     0     0     0
     0     0     0     0
```

** zeros (2 , 4)

صف عامود

❖ Ones ( )

نستخدم هذا الأمر من اجل انشاء مصفوفة كل عناصرها "1"

Example:

```
A =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> B= ones(3,2)

B =

     1     1
     1     1
     1     1
```

❖ Eye ( )

نستخدم هذا الأمر من اجل انشاء مصفوفة الوحدة (identity matrix) وهي المصفوفة التي ذا ضربناها ب أي مصفوفة أخرى لا تغير على قيمتها شيء وتتكون عناصرها من اصفار باستثناء ال diagonal تكون قيمة عناصره 1.

```
>> eye (3)

ans =

     1     0     0
     0     1     0
     0     0     1

>> eye(5)

ans =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

❖ random

امر نستخدمه من اجل انشاء مصفوفة جميع عناصرها عشوائية حيث نحدد حجم المصفوفة وثم يقوم البرنامج بفرض عناصرها قيم عشوائية بين الصفر و الواحد.

Example:

```
>> rand(3,6)

ans =

    0.1455    0.5797    0.8530    0.5132    0.2399    0.2400
    0.1361    0.5499    0.6221    0.4018    0.1233    0.4173
    0.8693    0.1450    0.3510    0.0760    0.1839    0.0497
```

** اذا اردنا ان نجعل القيم العشوائية اعداداً صحيحة نستطيع استعمال أمر( ) randi اذا احتجتها في أي وقت يمكنك البحث عنها في لائحة help.

❖ Diagonal

نستخدم هذا الأمر لإنشاء مصفوفة نقوم نحن بوضع قيم ال diagonal كما نريد و هو يقوم بانشاء مصفوفة و يضع باقي القيم اصفار.

Example:

```
>> v=[2 1 8 9];
>> a=diag(v)

a =

     2     0     0     0
     0     1     0     0
     0     0     8     0
     0     0     0     9
```

يمكننا استعمال امر () diag بشكل عكسي(اذا كانت عنا مصفوفة و بدنا نوخذ ال diagonal لاتبعها بنستعمل الأمر).

```
c =

     1     2     3
     4     5     6
     7     8     9

>> d=diag(c)

d =

     1
     5
     9
```

❖ Repmat

نستخدم هذا الامر عندما نريد انشاء مصفوفة جميع عناصرها نفس الرقم، فنقوم بتحديدالرقم ثم حجم المصفوفة بالترتيب.

Example:

```
>> repmat(pi,6,4)

ans =

    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416

>> repmat(2.555,2,6)

ans =

    2.5550    2.5550    2.5550    2.5550    2.5550    2.5550
    2.5550    2.5550    2.5550    2.5550    2.5550    2.5550
```

Solving equations by MATLAB

حل المعدلات سهل وسريع نسبيا على الانسان وحلها يدويا يتطلب وقتاً أقل من كتابة برنامج لحلها لكن هذا يتوقف عند المعادلات من متغيرين او ثلاث متغيرات بحد أقصى، أما اذا احتجنا الى حل معادلة من 10 متغيرات مثلاً (نحتاج إلى وقت كبير جدا لحلها و نكون عرضة للوقوع بأخطاء في الحل ايضاً) لذلك من الأسهل استخدام الماتلاب الذي بإمكانه حلها في ثواني معدودة ويمكنه القيام بذلك من دون الوقوع في الأخطاء.

$$x_1 - 2x_2 + 3x_3 = 0$$
$$-2x_1 - 3x_2 - 4x_3 = 0$$
$$2x_1 - 4x_2 + 4x_3 = 0$$

← **Easy for human**

$$\begin{aligned}
2x_1 - x_2 + 3x_4 - x_6 + 2x_7 + 3x_9 + x_{10} &= 1\\
x_1 + x_3 + 3x_4 + 2x_5 + x_6 + 3x_9 - x_{10} &= 2\\
3x_1 + 3x_2 - x_3 - x_4 + 2x_5 + 3x_6 - x_7 + 2x_8 + 3x_9 + x_{10} &= 1\\
2x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 + 2x_6 + x_7 + x_{10} &= 3\\
3x_1 - x_2 - x_3 + 2x_5 - x_6 + x_7 + 3x_8 + x_9 + 2x_{10} &= 2\\
x_1 - x_3 + x_4 + 2x_5 - x_7 + 3x_8 - x_9 + 2x_{10} &= 3\\
x_1 + x_2 + x_4 - x_5 + x_6 + x_7 + 2x_8 + x_9 + 2x_{10} &= 1\\
3x_1 + x_2 - x_3 + 3x_4 - x_5 + 3x_6 - x_{10} &= 0\\
-x_1 + 2x_2 + x_3 + x_4 + 3x_5 - x_6 + x_8 - x_9 - x_{10} &= -1\\
-x_1 + 2x_2 + 3x_4 - x_5 + 3x_6 + x_7 - x_8 - x_9 &= 2
\end{aligned}$$

← **Hard for human**

❖ مراجعة سريعة لموضوع **solving linear system of equations :**

أخذنا الموضوع ب (رياضيات هندسية1 او "الديف")

لنفرض انه يوجد لدينا المعادلة التالية التي تتكون من matrices فقط.

معاملات x1 معاملات x2 معاملات x3 المتغيرات بالترتيب النتائج بالترتيب

$$\begin{bmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \end{bmatrix} \times \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} d1 \\ d2 \\ d3 \end{bmatrix}$$

A x **B** = **C**

A: معروف — B: متغير — C: معروف

في العادة سوف نقوم بقسمة A على طرفي المعادلة من اجل إيجاد قيمة المتغير B لكن لا نستطيع ان نفعل ذلك لأننا نتعامل مع matrix لذلك نقوم بضرب كلا الطرفين ب A inverse (قسمة ال matrix هي نفسها الضرب بال inverse).

$$A^{-1} \times A \times B = A^{-1} \times C$$

$$B = A^{-1} \times C \longleftarrow \text{Solution}$$

❖ **خطوات الحل:**

1- نقوم بترتيب المعادلة اذا لم تكن مرتبة (المتغيرات المتشابهة فوق بعضها و الثابت بعد المساواة)
2- اذا كان هناك متغير غي موجود في المعادلة هذا يعني ان معامله = 0
3- نقوم بنشاء مصفوفتين الأولى تحتوي على **المعاملات بالترتيب** والثانية تحتوي على **الناتج من كل معادلة بالترتيب**
4- نعرف متغير جديد و نساويه بناتج ضرب ال inv للمصفوفة الأولى ب المصفوفة الثانية (او باستعمال القسمة ما بتفرق بالماتلاب).

****هذه الخطوات يمكننا ان نحل بها أنظمة قد تصل الى آلاف المعادلات ب آلاف المتغيرات.**

Example:

Use MATLAB to solve this system of three equations:

3*X1 - 2*X2 + 0.5*X3 = 12

-X1 + 2*X2 + 2*X3 = 2

X1 + X2 - 4*X3 = -16

Editor - D:\Omar\1.Eng\matlab\random\Untitled.m

Untitled.m

```
1 -    A=[3 -2 0.5 ;-1 2 2; 1 1 -4];
2
3 -    B=[12; 2; -16];
4
5 -    c= inv(A)*B
```

Command Window

```
>> Untitled

c =

    2.0000   <-- X1
   -2.0000   <-- X2
    4.0000   <-- X3
```

Practice problems

Problem 1

Suppose x takes on the value **x= 1 , 1.2 , 1.4 , 1.6 …… 5**. Use MATLAB to compute the array y that results from the function **y = sin (4x), (y is in degrees)**. then use MATLAB to determine **how many elements** are in the array 'y' and the third value of the **third element** in array y.

Solution:

Code:

```
x=[ 1: 0.2 : 5] ;
y = sind (4*x); %to make it in degrees
length(y)
y(3)

```

Result:

1- Y is defined,
2- The number of elements of array y is 21
3- the third value of y in degrees is 0.0976

Problem 2

Use MATLAB to solve this set of equations:

6x – 4y + 8z = 112

-5x - 3y + 7z =75

14x + 9y – 5z = 67

Solution:

Code:

```
a=[6 -4 8; -5 -3 7; 14 9 -5];
b=[112;75;67];
c = inv(a)*b

```

Result:

X= 2.8874 Y= 13.1921 Z=18.4305

Problem 3

For the matrix a=[1 4 2 ; 2 4 100 ; 7 9 7 ; 3 pi 42] and b = ln(a) do the following :

a) Evaluate the sum of the third raw of **b.**
b) Multiply element by element the second column of **b** and the first column of **a** and save the result in a variable named " **s1** ".
c) Evaluate the maximum value of in vector " **s1** ".
d) Use element by element division to divide the first raw of **a** by the first three elements of the third column of **b** and save the result vector in variable named **s2**. then evaluate the sum of the elements of the resulting vector.

Solution:

Code:

```
a = [1 4 2; 2 4 100; 7 9 7; 3 pi 42];
b = log(a);
sum( b(2,:) )
s1=b( : , 2 ) .* a( : , 1 )
max(s1)
s2 = a( 1 , :) ./ b( 1:3 , 3 )'
sum(s2)

```

Result:

a) 6.6846

b) 1.3863
2.7726
15.3806
3.4342

c) 15.3806

d) s2= 1.4427 0.8686 1.0278
Sum (s2) = 3.3391

Section 3

Plot

Introduction:

بعد كتابة البرنامج قد تريد اظهار النتائج على شكل رسم بياني وقد نريد بعض الأحيان استعراض المتغيرات ومقارنة قيمها ببعضها وأفضل طريق لذلك هي الرسم (plot) وكما يقال (الصورة تساوي الف كلمة).

ويعد الماتلاب أداة فعالة جدا من اجل هذا الغرض حيث يستطيع انشاء رسمات دقيقة لأنه يقوم بملايين العمليات الحسابية في أجزاء من الثانية. ونستطيع استخدامها في رسم الاقترانات والعلاقات الرياضية.

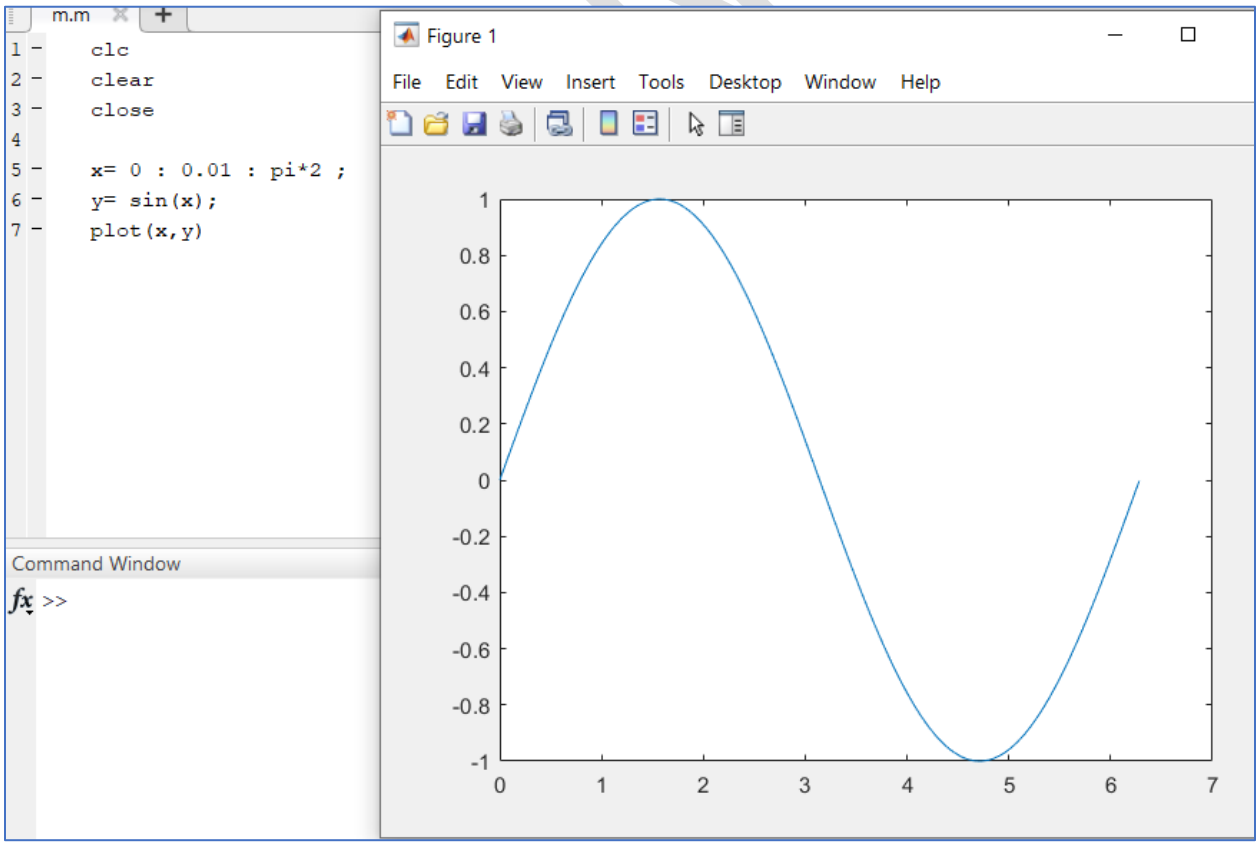
الماتلاب يتيح لك ان تكون الرسمة 2D او 3D ويمكن التعديل على مظهرها العام ولون الخطوط واضافة grid وتغير العنوان واسم كل محور و.... وكل هذا من اجل الصورة واضحة وسهلة القراءة.

2d plot:

عندما يتم رسمها على محورين فقط. وأبسط صيغة يكتب بها امر plot هي:

Plot (x-axis value , y-axis value)

سنبدأ بمثال بسيط لشرح كيف نرسم رسمة ال sin ... وسنقوم بشرح الأوامر والعناصر بناءً عليها:



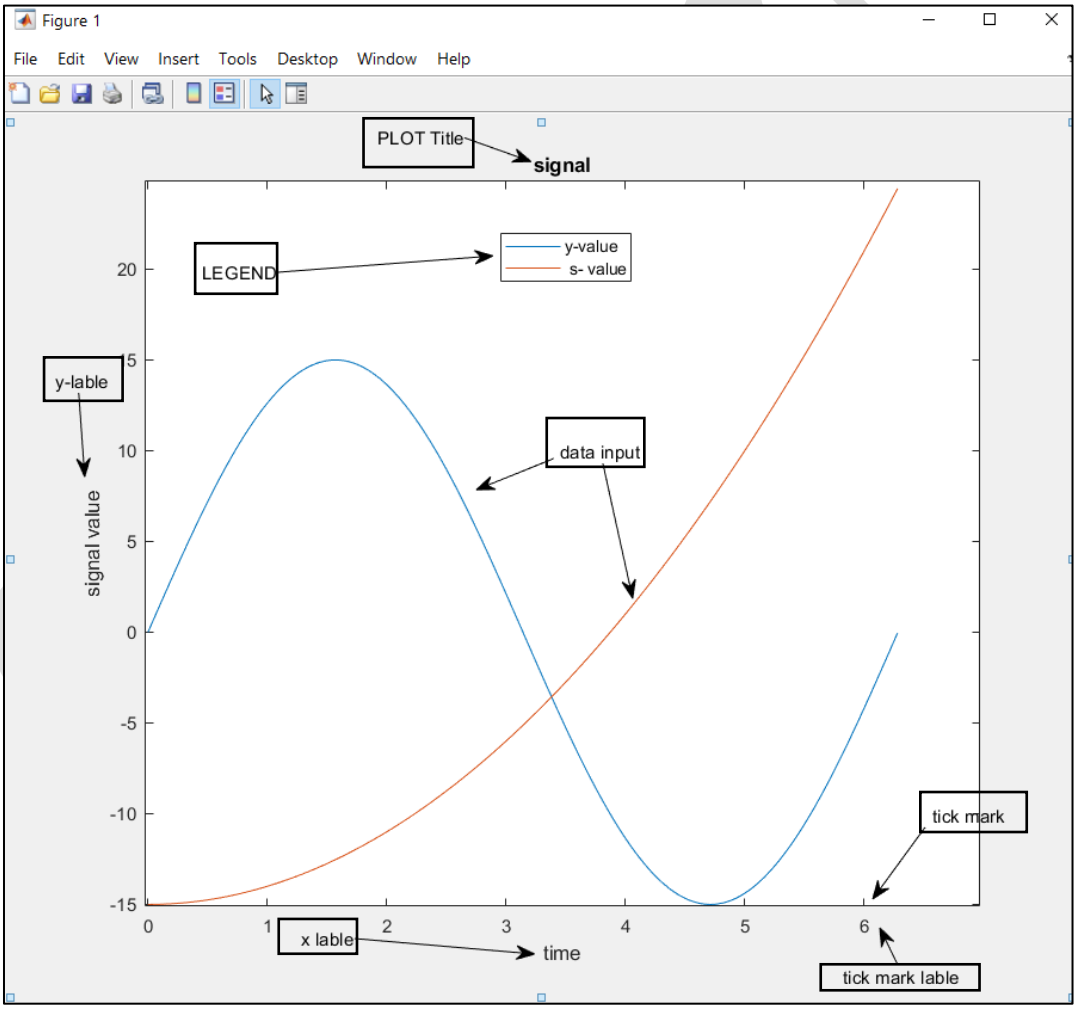
في المثال السابق:

1- في البداية قمنا بتعريف ال vector اسمه x ويحتوي على جميع القيم بين 0 و 2π .
2- قمنا بتعريف vectorاخر اسمه y وقيمه تساوي sin(x) .
3- قمنا بكتابة امر plot بين ال x و ال y .

** ملاحظة يجب ان يتساوى عدد عناصر xمع عدد عناصر y وإلا ستظهر إشارة error.

** بعد ان تظهر ال plot تستطيع طباعتها (print) وتستطيع ان تحفظها كملف بي دي اف او صورة لتستخدمها في وقت لاحق.

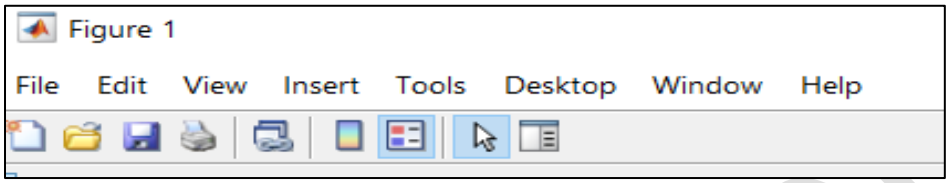
❖ يتيح لك الماتلاب ان تقوم بتعديل ال plot والعناصر داخلها مثل شكل الخط لون الخط شكل النقاط وأسماء المحاور وإضافة وسنتعرف على كيفية القيام بذلك:



Plot editing

ولان سوف نتعلم كيفية إضافة هذه العناصر والتعديل عليها:

لكن قبل البدء يجدر بنا الذكر ان اغلب هذه العناصر يمكننا اضافتها والتعديل عليها من اللوائح في شاشة ال plot وهذه التعديلات سهلة ومباشرة تستطيع استكشافها بنفسك.



1- تعديل الألوان والنقاط وأنواع الخطوط (colors, data markers, line types)

Plot (x , y, ’ ’)

هنا نضع رموز تعبر عن اللون والعلامة ونوع الخط الذي نريده

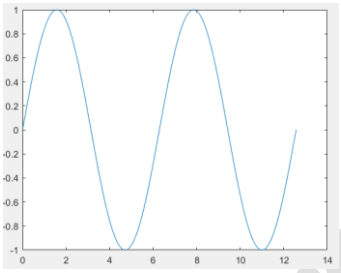
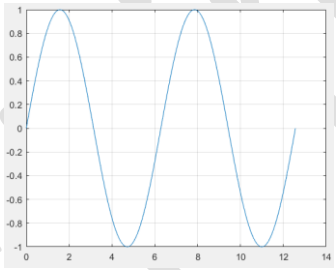
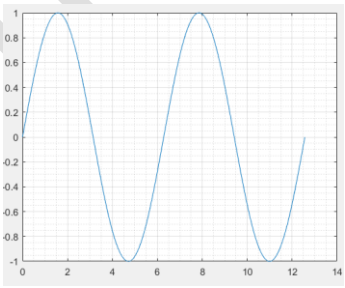
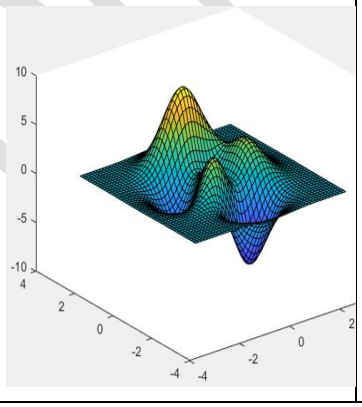
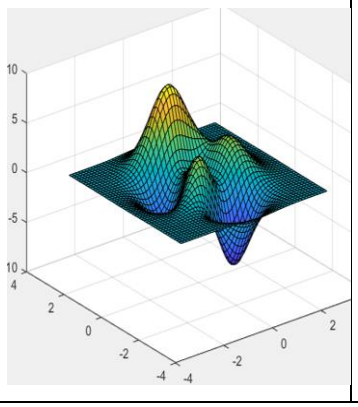
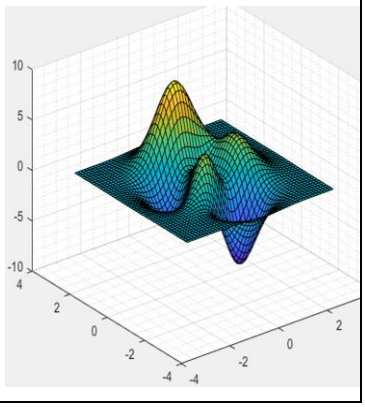
وها هو جدول بجميع هذه الرموز:

Symbol	Color		Symbol	Marker (points)		Symbol	Line style
b	Blue		.	Point		-	Solid line
g	Green		o	Circle		:	Dotted line
r	Red		*	Asterisk		-.	Dash dot line
c	Cyan		+	Plus		--	Dash line
m	Magnet		x	Cross		(None)	No line
y	Yellow		s	Square			
k	Black		d	Diamond			
w	white		v	Triangle down			
			^	Triangle up			
			<	Triangle left			
			>	Triangle right			
			P	Pentagram			
			N	Hexagram			

Grid & grid minor

يتيح لك الماتلاب إضافة خطوط بيانية للرسمة عن طريق امر grid on واذا اردت ان تضيف المزيد من الخطوط للقياس بشكل ادق يمكنك أيضا استعمال امر grid minor بعد ال grid on.
واذا اردت الرسمة من دون أي خطط تستطيع ايقافها عن طريق grid off.

والجدول التالي يوضح لنا تأثير كل أمر فيهم:

drawing	2D		
Grid type:	Grid off	Grid on	Grid minor
Figures			
drawing	3D		
Grid type:	Grid off	Grid on	Grid minor
figures			

Title and labels

يمكنك إضافة عنوان وأيضا تسمية المحاور للدلالة لتستطيع توضيح ما تمثله

يمكنك إضافتهم عن طريق كتابة الأوامر التالية بعد ال plot:

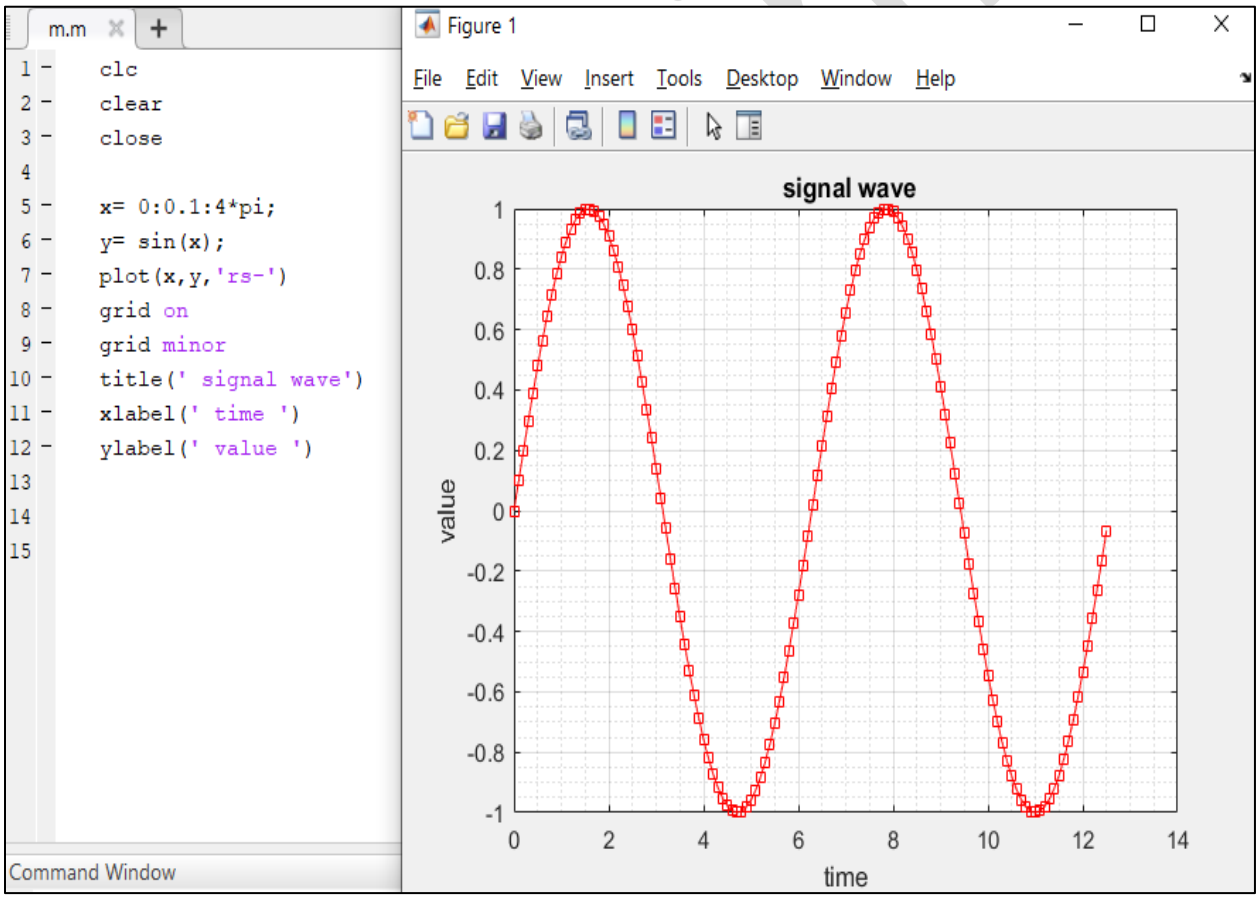
title (‘ ‘)

xlabel (‘ ‘)

ylabel (‘ ‘)

Example:

Draw a sine wave from **0 to ‘4*pi’** with **red line** and **square markers** and **straight line,** then add a **minor grid** to the figure then add a **title**, **x-label** and **y-label.**



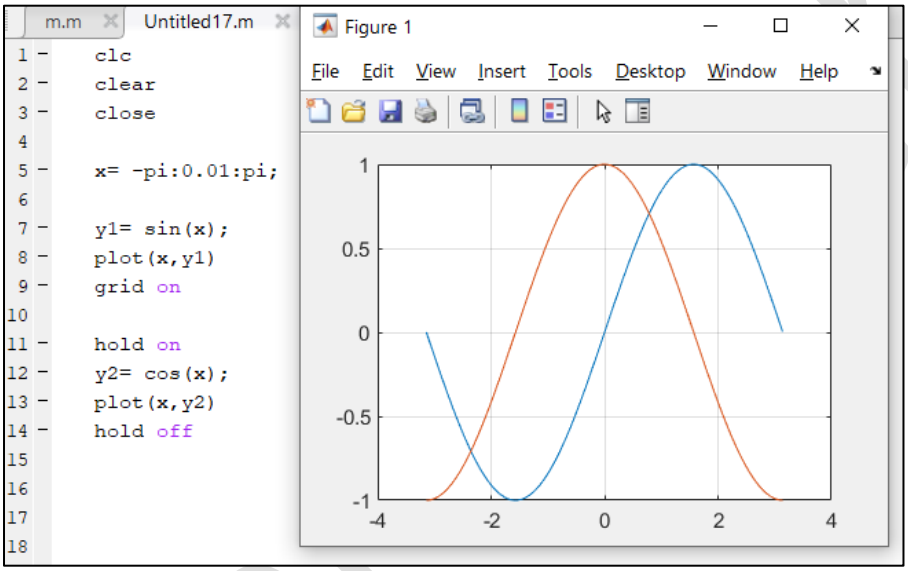
Drawing multiple data in one drawing

نستطيع رسم أكثر من اقتران او مجموعة بينات في نفس الرسمة بطريقتين:

1- Hold

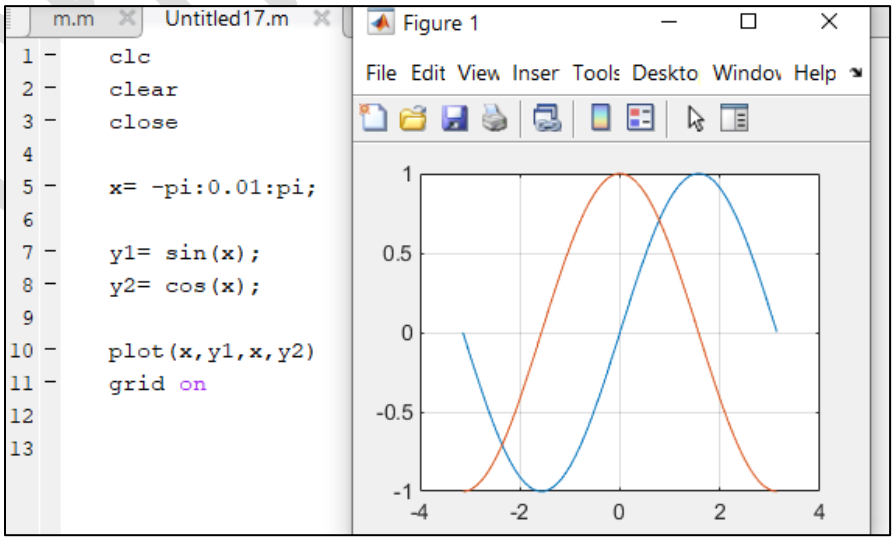
بعد كتابة اول امر plot نقوم بكتابة **hold on** ونكتب بقية أوامر plot التي نريد رسمها وحينما ننتهي منها نقوم بكتابة **hold off**

Example:



2- Plot(x,y1, x,y2 ,..... ,x,yn)

تستطيع رسم اكثر من رسمة أيضا بطريقة مباشرة حيث تضع داخل أمر plot اول متغيرين ومن ثم المتغير الأول والمتغير الثالث وهكذا... (لاحظ المتغير الأول يتكرر في كل زوج).



Legends

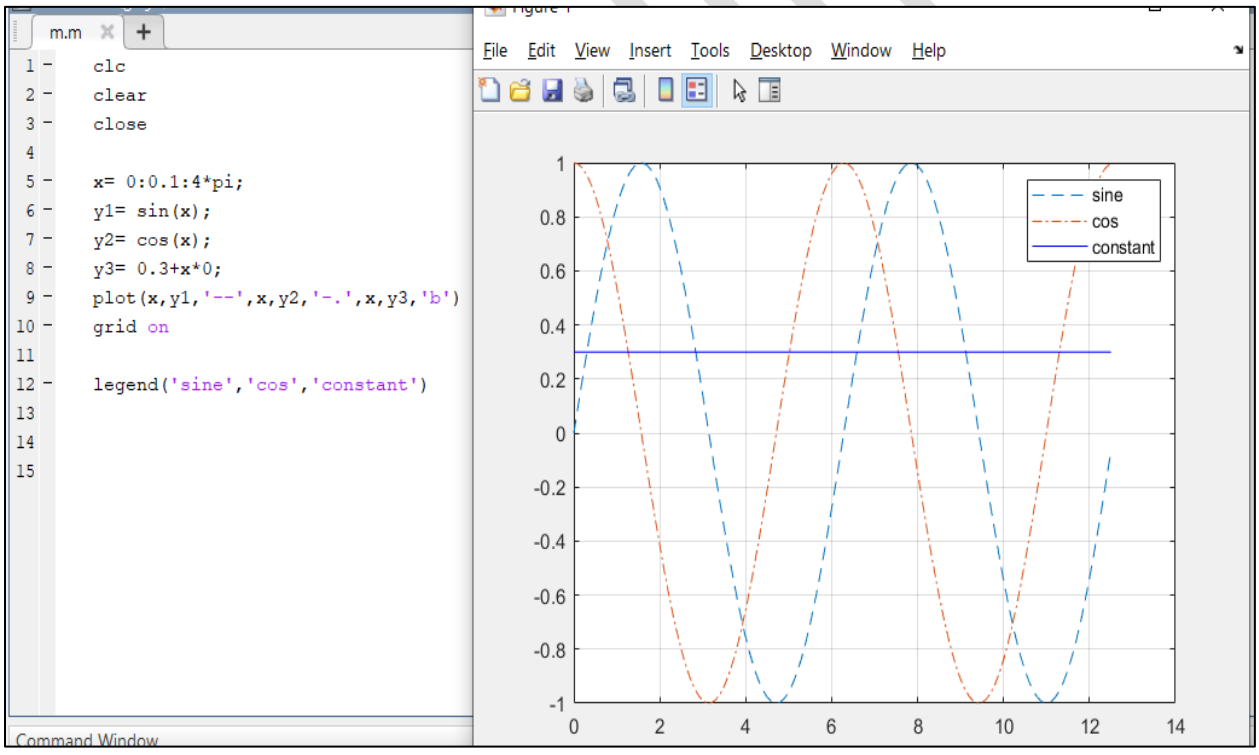
عندما نرسم أكث من خط او علاقة معا يستحسن ان تضع legend معها للدلالة علو ما يمثله كل خط و سيقوم الماتلاب بإظهار ما يمثله كل خط مع شكله ولونه.

نكتبها بالصيغة التالية:

Legend (‘ ‘ , ‘ ‘, …)

Example

Write program to plot the three functions y1= sin x , y2= cos x , y3 = 0.3 from x=0 to x=4π use grid on , and add legends to the figure.

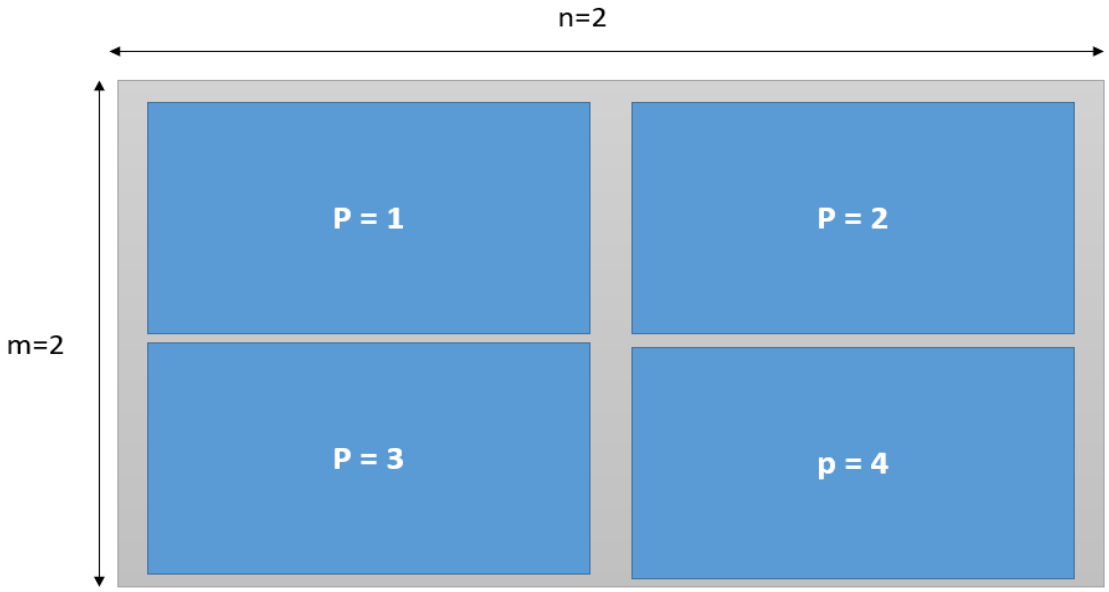


Drawing more than one plot in the same figure

في نفس ال figure يمكنك رسم اكثر من plot واحد وذلك عن طريق امر **subplot**

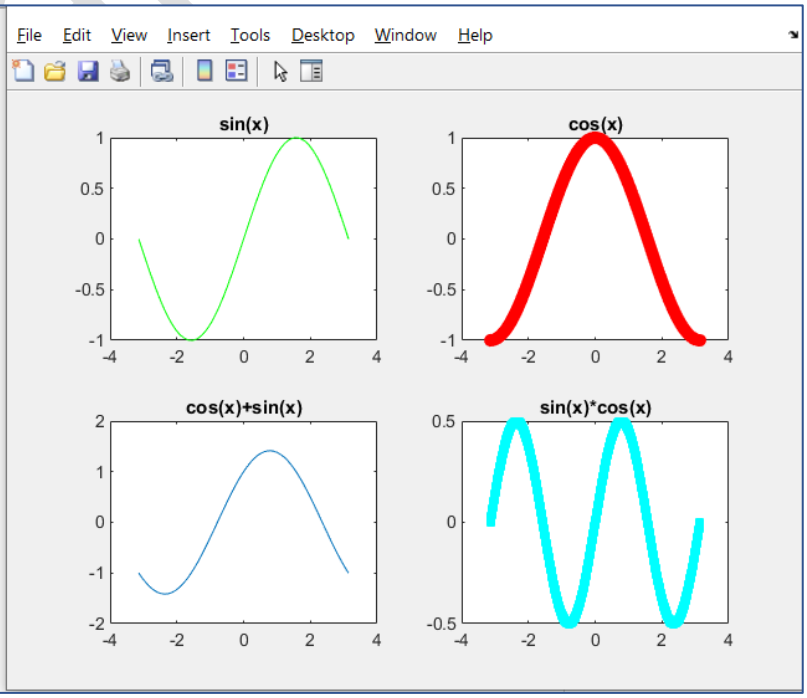
Subplot (m, n ,p)

عدد الأعمدة:m عدد الصفوف:n رقم ال plot المراد تحديده:p

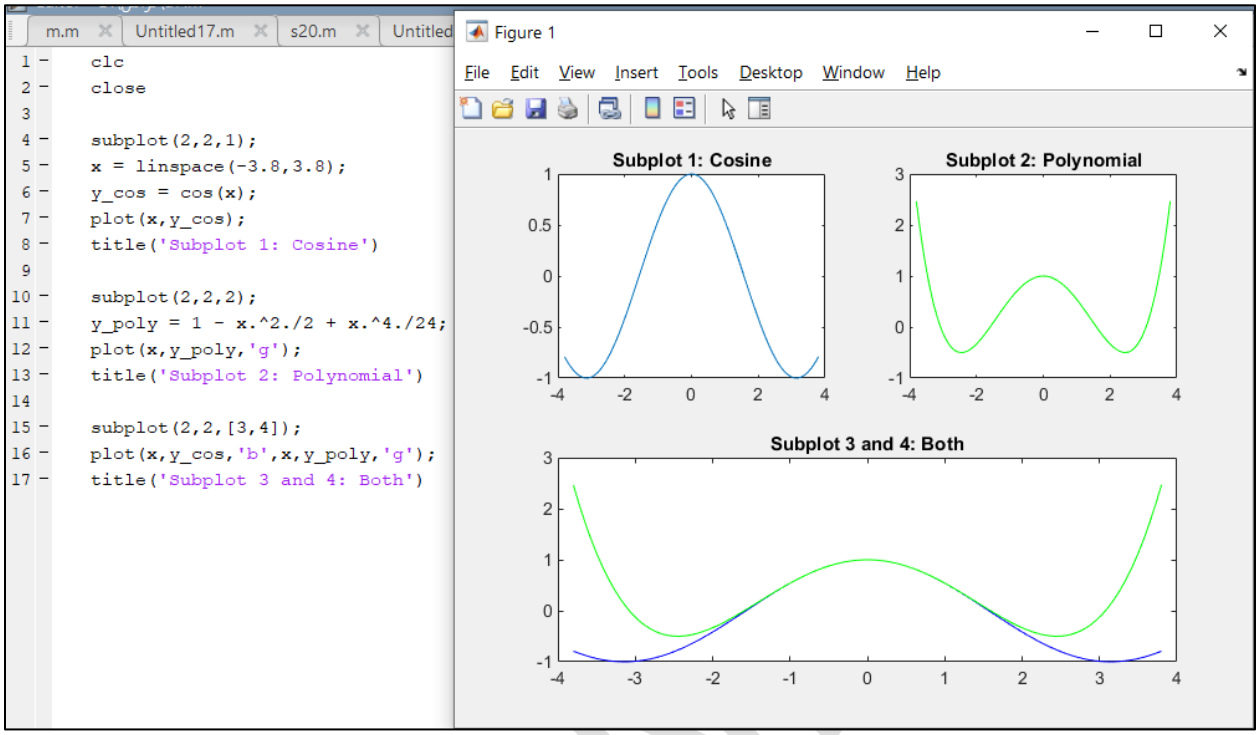


Example

```
clc
clear
close

x = -pi:2*pi/1000:pi;
y1 = sin(x);
y2 = cos(x);
y3 = y1+y2;
y4 = sin(x).*cos(x);

subplot(2,2,1);
plot(x,y1,'g-');
title('sin(x)');

subplot(2,2,2);
plot(x,y2,'ro:');
title('cos(x)');

subplot(2,2,3);
plot(x,y3);
title('cos(x)+sin(x)');

subplot(2,2,4)
plot(x,y4,'cs--')
title('sin(x)*cos(x)')
```



You can also use subplot like this:



Using more than one figure

ال figure هي النافذة التي تظهر داخلها ال plot واستخدام اكثر من figure يعني ان تستخدم أكثر من نافذة وهذا الأمر بسيط وتم عن طريق كتابة ما يأتي:

Figure (1)

Plot (x, y)

.

.

Figure (2)

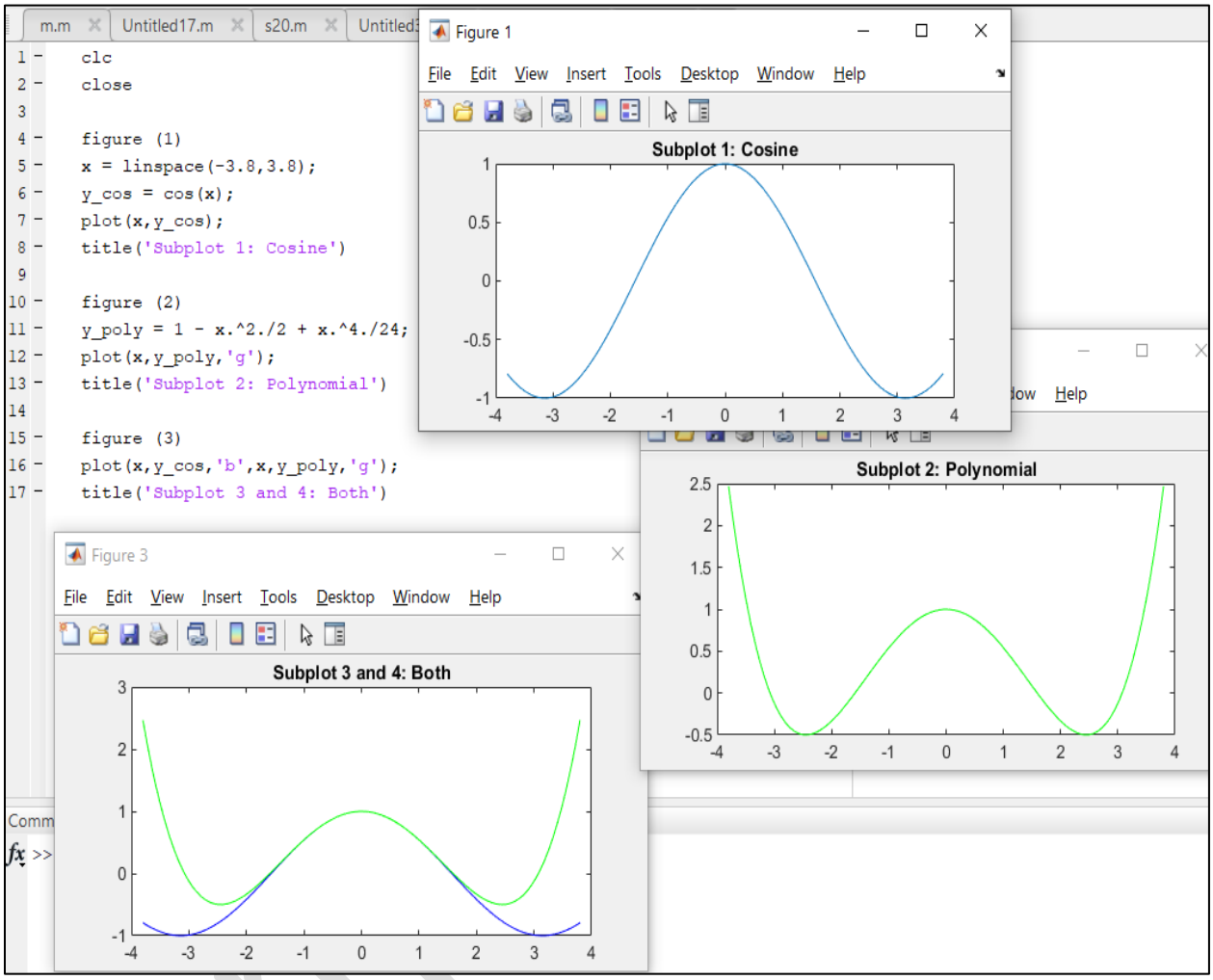
Plot (x_1 ,y_1)

…

ال plot الموجودة تحت ال figure رقم 1 سيتم رسمها في اول نافذة و plot الموجودة تحت 2 سيتم رسمها في النفاذة الثانية وهكذا.... وتستطيع استخدام العدد الذي تريده من الfigures .

Example

Plot the last example in subplot but use 3 figures instead of subletting.



Three-dimensional plotting

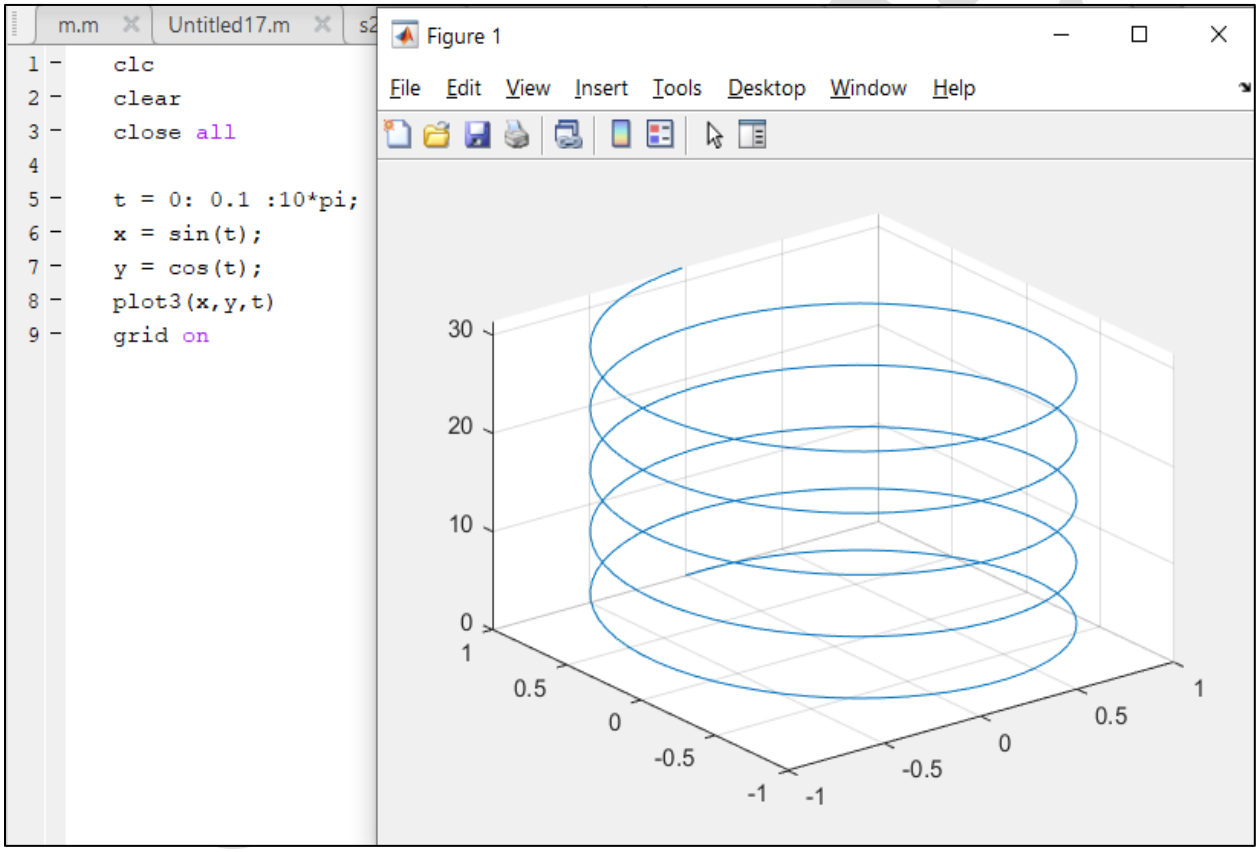
يتيح لك الماتلاب الرسم على بشكل ثلاثي الأبعاد (x, y, z) عن طريق أمر اسمه **plot3**

يكتب الأمر بالصيغة التالية:

Plot3(x-axis, y-axis, z-axis)

Example

Plot a helix in a 3D plot, the helix rule is =(sin(t), cos(t), t) and add a grid to the plot



**تنطبق على ال 3d ما ينطبق على ال 2d من ناحية رسم أكثر من مجموعة بينات أو تسمية المحاور أو إضافة legends وغيرها من الأوامر.

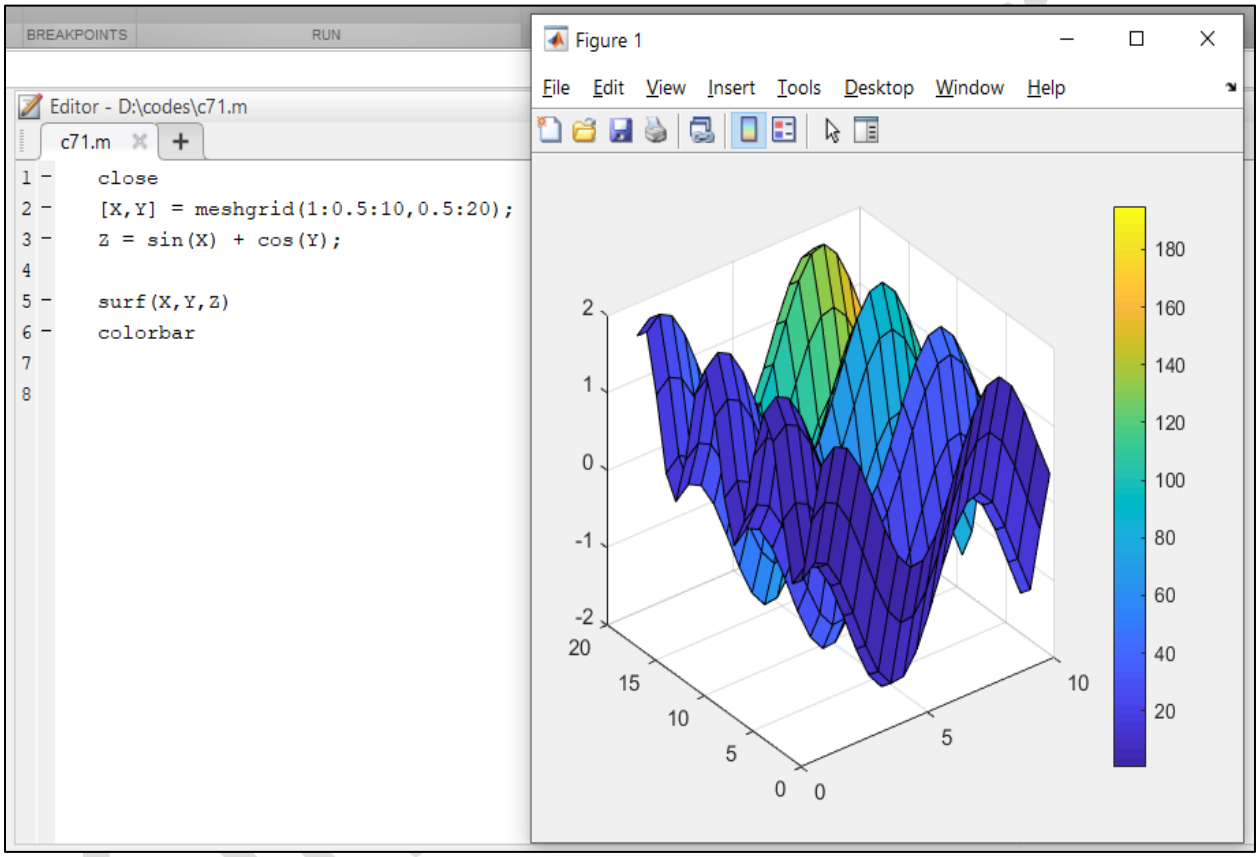
Other commands for plotting:

1- Surf

يستخدم لرسم الأسطح بشكل ثلاثي الأبعاد وليس كخط كما في plot3:

Surf(x, y, z)

Example:



ملاحظة:

أمر meshgrid يستخدم من اجل تحويل 2vectors الى surfaces اذا أحببت ان تعرف أكثر عنه و عن الية عمله بإمكانك البحث في قائمة help.

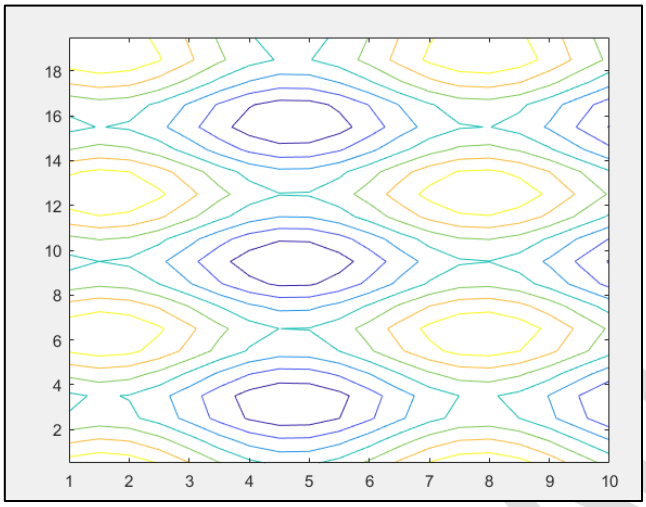
2- Contour

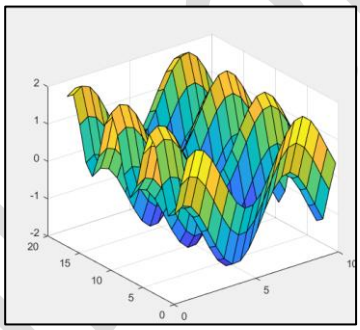
يستخدم لرسم خرائط ال كونتور

Example

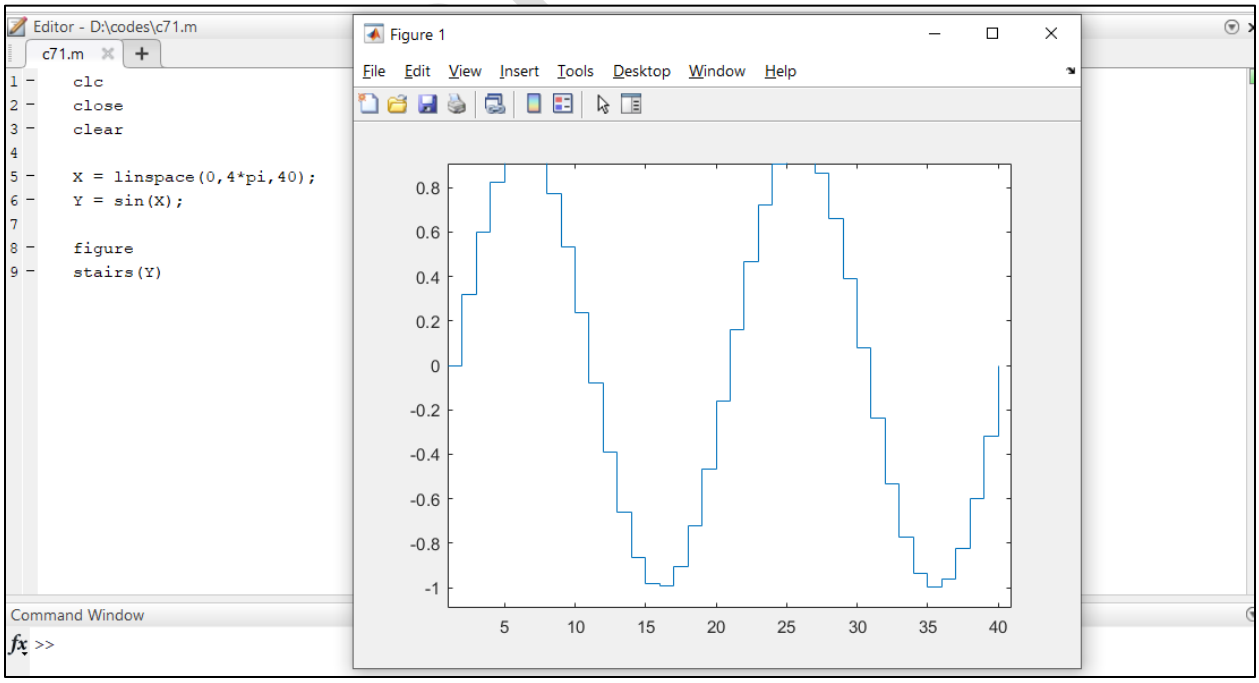
Draw the last example but with contour lines:

(نكتب نفس الأمر تماما ولكن بدل surfنضع contour) وهذه مقارنة بين النتيجتين

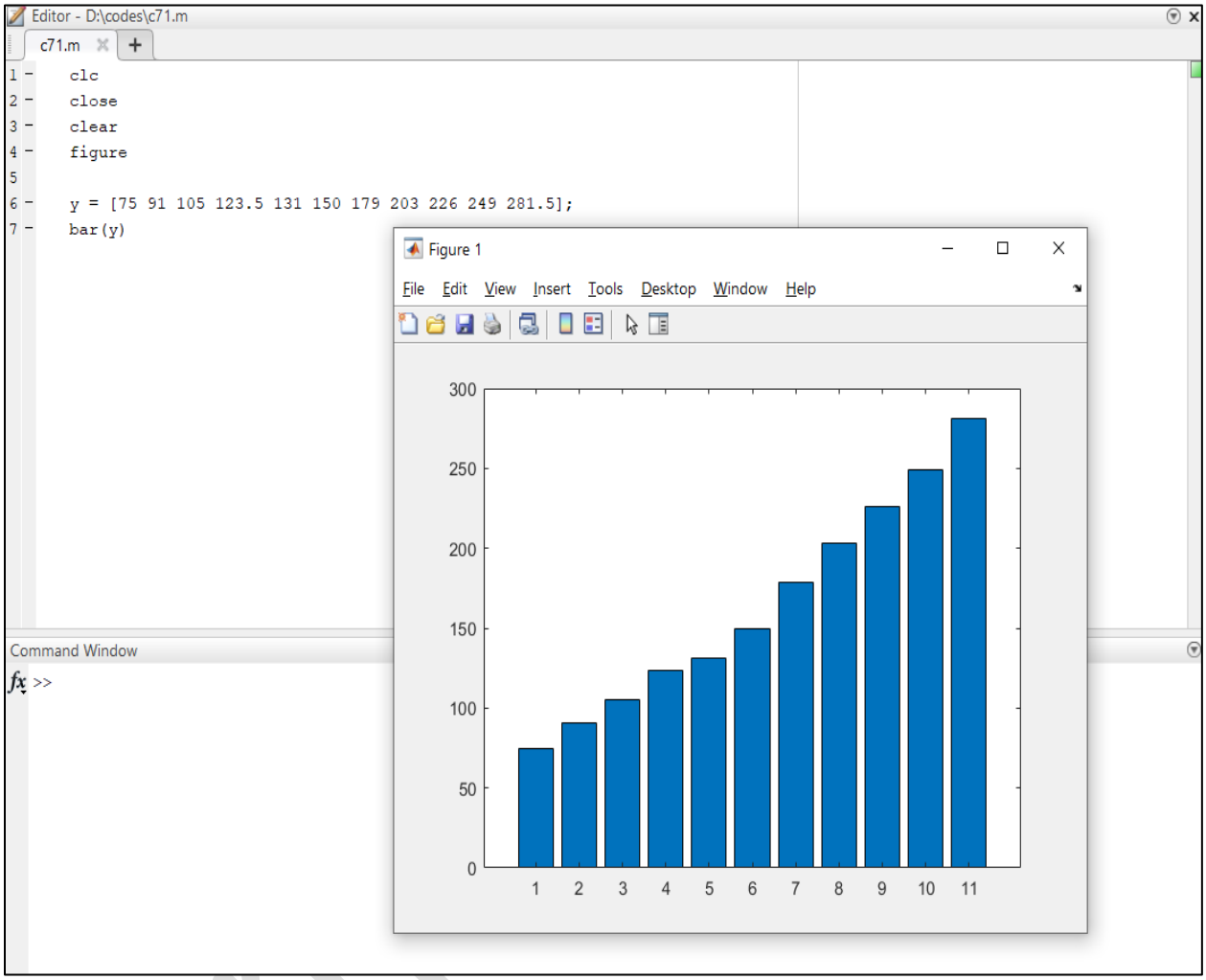




3- Stairs



4- Bar



يمكنك البحث بشكل مفصل في قائمة help عن جميع هذه الخيارات وعن كيفية عملها بشكل مفصل

Section 4

Input &output

Introduction:

في كثير من البرامج نحن لا نريد ان يطبق القيم التي وضعناها فقط بل نحن نريد للمستخدم ان يستطيع ادخال أي قيمة يريدها ويستطيع تنفيذ البرامج بهذه القيم عدد ما يشاء من المرات بدون الدخول و التعديل على الكود لهذا يمكننا الاستعانة بأمر **input** الذي يمكن للمستخدم ادخال قيمة متغير ما في كل مرة نقوم بتنفيذ البرنامج.

وأحيانا نريد ان نظهر الناتج من تنفيذ البرنامج على الشاشة وقد لا يفهم المستخدم على الرموز التي وضعناها لذلك نحتاج لطباعة كلام على الشاشة مع الأرقام و نريد وضع توضيحات تظهر على ال command window لذلك نضطر الى استخدام أوامر ال **output** وهم **disp( )** و **fprintf( )** ومن الأفضل ان نقوم بشرح هذه الأوامر قبل ان نقوم بشرح أوامر ال if وال loops .

Input command

- أهمية تمكين المستخدم من ادخال متغير(variable) عندما يقوم بتشغيل البرنامج الذي كتبته انت امر مهم مثل ما وضحنا لذلك يجب ان نتعلم كيف نفعل ذلك ولا يوجد في الماتلاب غير صيغة واحدة وهي:

Variable_name = input (' some message ')

اسم المتغير الذي نريد تعرفه

رسالة ما للمستخدم تخبره عن القيمة التي سوف يدخلها

- **Example:**
- لنفرض أننا كنا نقوم بكتابة برنامج و اردنا من المستخدم ان يقوم بإدخال طوله ووزنه وعمره في ثلاث متغيرات .

```
1 -    clc
2 -    clear
3
4 -    age= input(' enter your age please:  ')
5 -    weight = input(' enter your weight please:  ')
6 -    height = input(' enter your height please:  ')
```

```
Command Window
 enter your age please:  19

age =

    19

 enter your weight please:  70

weight =

    70

 enter your height please:  172

height =

   172
```

** في معظم الأوقات ما بهم المستخدم شو اسم المتغير الي راح نخزن فيه القيمة للمعلومة الي قام بإدخالها عشان هيك راح نحط دايما " ; " بعد كل امر "input" ولأنه ارتب كمان.

```
3
4 -    age= input(' enter your age please:  ');
5 -    weight = input(' enter your weight please:  ');
6 -    height = input(' enter your height please:  ');
```

```
Command Window
   enter your age please:  19
   enter your weight please:  70
   enter your height please:  172
```

**لاحظ انه لو ادخلنا حروف مع الأرقام راح يعطينا error لأنه هو بتوقع انه يكون المتغير رقم في هذه الحالة فلو كتبنا لما 19 years بدل من 19 فقط راح يعطينا ال error زي هيك:

```
1 -    clc
2 -    clear
3
4 -    age= input(' enter your age please ')
5 -    weight = input(' enter your weight please ')
6 -    height = input(' enter your height please ')
```

```
Command Window
   enter your age please 19 years
Error using g (line 4)
Error: Invalid expression. Check for missing multiplication operator, missing or unbalanced delimiters,
or other syntax error. To construct matrices, use brackets instead of parentheses.
```

** لذلك يجب أن نتعلم كيف يمكننا ادخال المتغيرات التي تحتوي على احرف.

** للفائدة: اذا اردنا ان نطبع(') مثلا في كلمة can't نكتبها مرتين داخل الأقواس بهذا الشكل:

Input (' what''s your name ')

نفعل ذلك لكي يميز البرنامج عندما نريد ان نطبع ' وعندما نريد ان نستعملها لكي ننهي الجملة.

❖ String input:

لنفترض اننا لا نريد من المستخدم ان يدخل في داخل المتغير أرقاماً بل نريد منه ان يدخل أحرف **"متغير من نوع STRING"** مثل ان يقوم بإدخال اسمه او مكان سكنه او عمله.

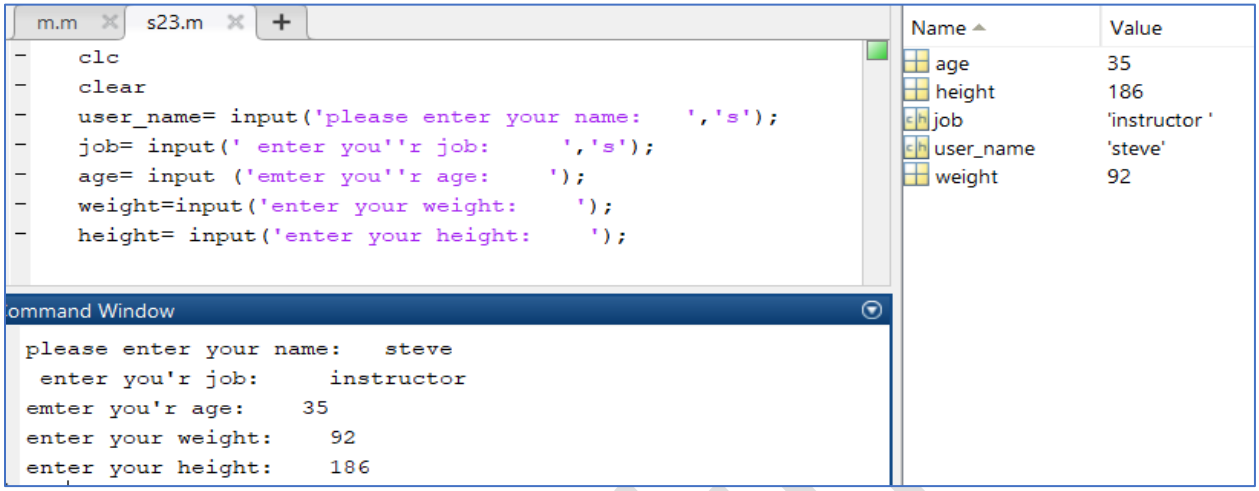
الحل في هذه الحالة بسيط وهو كتابة " ,s " في اخر القوس بعد الرسالة التي تريد اظهارها للمستخدم.

Variable_name = input (' any message ', 's')

- Variable_name: اسم المتغير الذي نريد تعرفه
- ' any message ': رسالة ما للمستخدم تخبره عن القيمة التي سوف يدخلها
- 's': للدلالة على ان المتغير string

Example:

ولان لنأخذ نفس المثال السابق لكن نريد من المستخدم ان يدخل اسمه معهم ووظيفته ونريده ان يدخل "year 25" ان يحفظه كما هو(يعني انه يحفظها string).



** لاحظ انه ال **age** وال **name** وال **job** تم حفظهم ك **string** في ال **workspace**.

Output commands

عند كتابة البرامج الطويلة يكون لدينا فيها الكثير من المتغيرات وكثير مما نريد توضيحه للمستخدم وأيضا عندما نريد اظهار النتيجة النهائية لذلك يجب ان نتعرف على أوامر الoutput وهما امران الأول ( )disp والثاني ( )fprintf

disp ( )

امر ( )disp و اختصار ل display ويقوم هذا الامر بطباعة ما في داخله على الشاشة سواء كان نص او متغير من نوع double او string او أي انوع.

Example:

لنفرض انك كنت تقوم بكتابة برنامج ليقوم بحساب سرعة سيارة تسير حسب علاقة ما وكنت تريد السرعة عند الثانية 4 وكانت القيمة هذه مخزنة في متغير اسمه car_speed المطلوب: قم بطباعة النتيجة على شاشة الماتلاب.

```
26 -    disp( 'speed when time is equal to 4 seconds = ' )
27 -    disp( car_speed )
28
```

```
Command Window
speed when time is equal to 4 seconds =
   25.3362
```

طبعا لاحظ انه لما طبعنا الجملة كتبناها بين (' ') عشان تنطبع زي ما هي اما لما كنا بدنا نطبع المتغير كتبناه عطول بين الأقواس بدون ال ' ' .

❖ **سؤال ممكن يخطر على البال!**

- ليش استعمل امر ( )disp و انا بقدر احط اسم المتغير وراح يطبعلي إياه زي هيك:

```
23
24 -     car_speed
25
```

Command Window

```
car_speed =

   25.3362
```

- الجواب بسيط، اول اشي الأمر ساعدني انه اكتب الجملة الي وضحت إيش معنى الرقم او الجملة الي راح تطلع (واسم المتغير لحاله مرات ما راح يكون كافي) ثاني اشي انه هو ما راح يكتبلي اسم المتغير وهيك راح يكون الحل ارتب واريح للمستخدم وراح يمنعك من الوقوع في الأخطاء الناجمة عن استعمال اسم المتغير بعشوائية.

❖ **طيب سؤال ثاني**

- ايش لو كنا بدنا نطبع الجواب على نفس السطر؟ يعني زي هيك:

speed when time is equal to 4 seconds = 25.3362

شو بدنا نعمل ؟؟

- **اذا جربنا نكتبه بنفس الامر راح يعطينا error**

```
25
26 -    disp( 'speed when time is equal to 4 seconds = ', car_speed )
27
28
```

Command Window

```
Error using disp
Too many input arguments.
```

➢ ببساطة البرنامج بحكيلك انه انت أدخلت مدخلات اكثر من الي بقدر يستحملها أمر disp عشان هيك ما بنفع نطبعهم باستخدام هذا الامر.
وهون هي نقطة الضعف عند أمر disp وهي إنه ما بنقدر نطبع أكثر من جملة واحدة أو متغير واحد في نفس الأمر.
مما يضطرنا في بعض الحالات الى استعمال امر اخر أكثر تعقيدا لكنه قادر على فعل هذه المهام وهو:

fprintf( )

fprintf ( )

يمكننا أن نعتبر أن امر fprintf مثل امر disp لكنه متقدم اكثر ويمكنه القيام بمهام اكثر حيث نستطيع ان نطبع text مع اكثر من متغير وهو الامر الذي لم يسطع امر dips ان يفعله

صيغة الامر:

fprintf (' text ' , variable1 , variable2 ,....)

**وفي كل مرة تريد أن تكتب فيها متغير داخل النص عليك ان تقوم بوضع إشارة % بعدها نوع المتغير

يعني اذا المتغير double بتكتب %d و اذا كان من نوع string بتكتب %s واذا كان flout بتكتب %f

في غيرهم اكيد لكن هذول اهم 3

Example:

Write a code to define 3 variables 'name, age and gpa ' let his name be Mohammad hussein and his age is 20 years and gpa is 4 , and print them in one command .

بده اياك تكتب اسم الشخص وعمره ومعدله وتطبعهم بأمر واحد.

```
1 -    clc
2 -    clear
3
4 -    name = 'mohammed hussein';
5 -    age = 20;
6 -    gpa = 4;
7
8 -    fprintf(' my name is %s and my age is %d and my gpa is %d ' ,name,age,gpa)
```

Command Window

```
fx  my name is mohammed hussein and my age is 20 and my gpa is 4  >>
```

** قمنا بوضع %s بدل الاسم لأنه string و %d بدل العمر و المعدل لانهم من نوع double و رجعنا كتبناهم بالأخر **بالترتيب** بعد ما خلصنا ال text

نستنتج من المثال السابق انه الطالب الي حكينا عنه مش طالب هندسة و مش طالب ميكانيك اكيد لأنه معدله 4 ... 😐

وإذ كنا بدنا نعمل سطر جديد او نعمل مسافة بنفس الامر الحل بسيط :

1- نكتب \n من اجل البدئ بسطر جديد
2- نكتب \t ممن اجل ترك مسافة صغيرة "زي كانه كبسنا Tab"
3- نكتب %% من اجل طباعة %

** في نفس المثال السابق

```
1 -    clc
2 -    clear
3
4 -    name = 'mohammed hussein';
5 -    age = 20;
6 -    gpa = 4;
7
8 -    fprintf(' my name is %s \n  \t %% my age is %d and my gpa is %d ' ,name,age,gpa)
```

```
Command Window
  my name is mohammed hussein
fx      % my age is 20 and my gpa is 4 >>
```

** لاحظ انه تم اضفة سطر بعد \n ووضع مسافة مكان \t وطباعة % عند %% .

Section 5

If ,else ,switch conditions

Introduction

الجمل الشرطية ليست جديدة علينا (المفروض) فقد اخذناها بال c++ ، حيث تقوم بأمر معين عندما يتحقق شرط نحن نحدده، توجد لها استخدامات غير محدودة يمكننا الاستفادة منها. سنقوم بشرح أوامر ال if و ال switch في هذا الدرس.

ولكن قبل ان نبدأ بالشر لا بد لنا من مراجعة بعض الموضوعات والأساسيات التي تعلمناها بال c++

Relational operators:

نستخدمها للمقارنة بين قيمتين والنتيجة منها logical value :

Operator	meaning	بالعربي
==	Is equal to	يساوي
~=	Is NOT equal to	لا يساوي
>	Grater than	اكبر من
<	Less than	اصغر من
>=	Grater or equal to	اكبر او يساوي
<=	Less or equal to	اصغر او يساوي

Logical operators:

نستعملهم للمقارنات المنطقية وبعطيك ناتج (true(1) او false (0))

*أي قيمة يتم إدخالها غير الصفر تعتبر (true) والناتج اما (0) او (1)

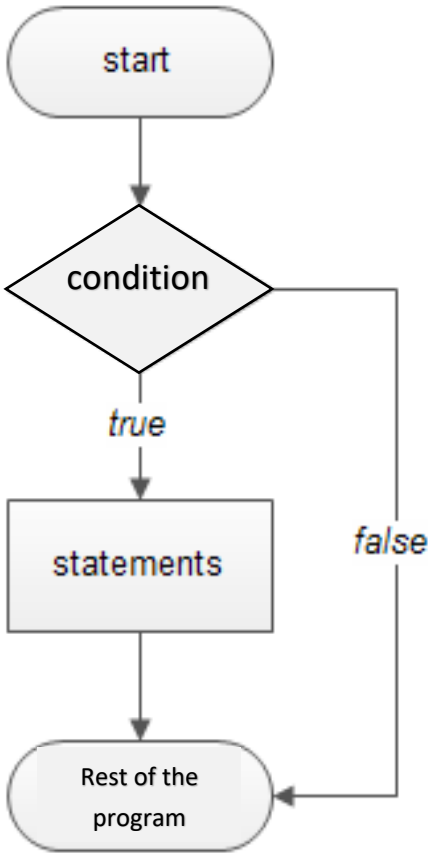
Operator	meaning	طريقة العمل
&&	AND	الناتج (true) اذا كان الطرفين (true) فقط
||	OR	الناتج (true) اذا كان **احد** الطرفين (true)
~	NOT	تعكس القيمة: اذا كانت القيمة (true) تصبح (true) والعكس صحيح

input		&&	||
1	1	**1**	**1**
1	0	**0**	**1**
0	1	**0**	**1**
0	0	**0**	**0**

If

- اول امر سوف نقوم بشرحه حيث تضع شرطا بعد ال if واذا كان الشرط صحيح يقوم بتنفيذ ما داخل امر if (ما بين if و end) توجد عدة صور لاستعماله:

1- if, end



- أبسط صور الاستعمال (إذا كان الشرط صحيح بنفذ ما داخل ال if ثم يكمل باقي البرنامج اما إذا كان الشرط خاطئ يكمل بقية البرنامج من دون ان ينفذ ما داخل if).

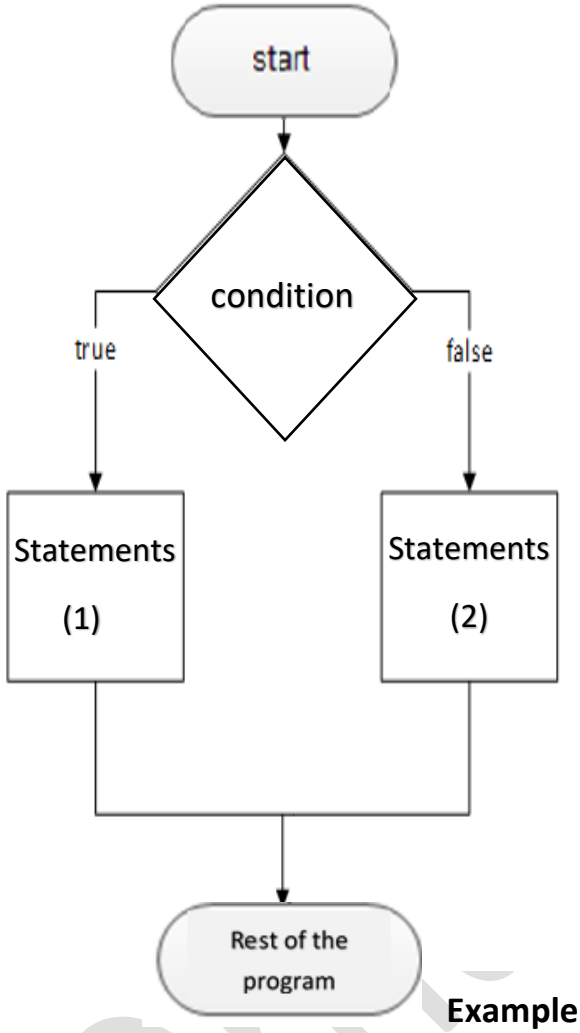
Example 1:

```
1 -    clear
2 -    x= input('enter any number  ');
3 -    if x==5
4 -        disp( 'your number is 5' )
5 -    end
6
```

** في هذا المثال كود يقوم باستقبال متغير اسمه x وبعدها جمله if الشرط فيها x تساوي العدد 5.

1- اذا تم ادخال العدد 5 سيقوم بتنفيذ ما داخل الامر وهو طبع الجملة "your number is 5"
2- اذا تم ادخال أي عدد غير 5 ببساطة لن يقوم بإدخال الجملة وسيكمل باقي البرنامج

If, else, end



- نفس الصيغة السابقة لكنا قمنا بإضافة Hمر else حيث يصبح مبدأ عمل الجملة كالتالي:
  1- اذا كان الشرط ناتجه صحيح (true) يقوم البرنامج بتنفيذ مجموعة الأوامر داخل جملة if.
  2- اذا كان الشرط ناتجه خطأ (false) يقوم بتنفيذ مجموعة الأوامر داخل else.

```
If condition
...
...
else
...
...
end
```

Example 2:

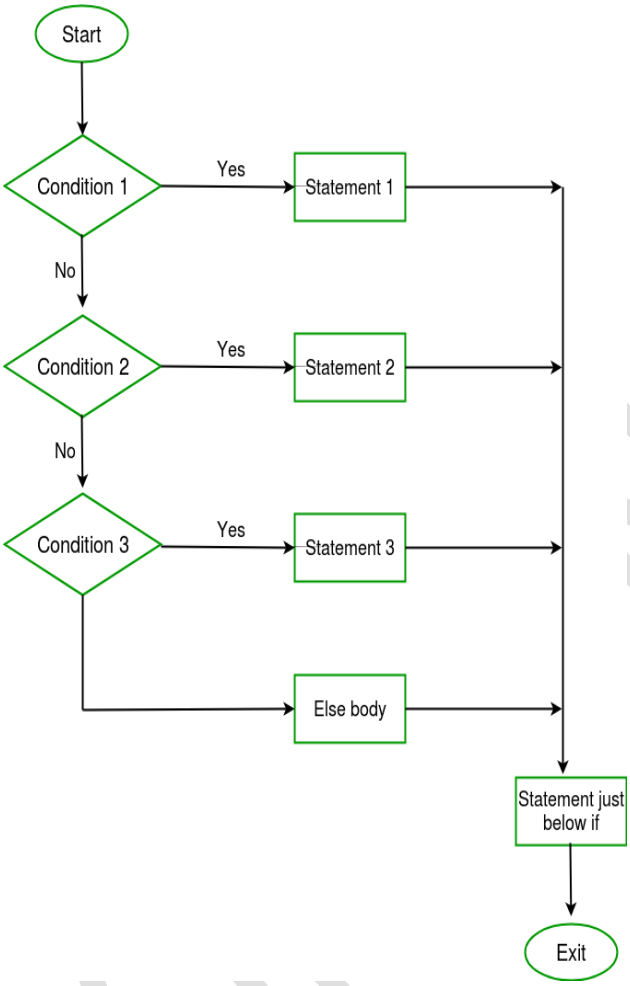
```
3 -    n= input('enter any number  ');
4 -    if n > 0
5 -        disp('your number is positive')
6 -    else
7 -        disp('your numer is either zero or negative')
8 -    end
```

البرنامج المكتوب في الأعلى يعمل بالشكل التالي:

1- يستقبل متغير اسمه **n**
2- اذا كان هذا المتغير(**n**) اكبر من الصفر سيتحقق شرط **if** وسيقوم بطباعة " your number is positive "
3- اذا كان (**n**) صفر او اقل (**يعني لم يتحقق الشرط الي ب if**) سيقوم بتنفيذ ما داخل else ويقوم بطباعة "your is number is either zero or negative"

If, elseif, else, end

- طيب لو فرضنا لنه بدنا نحط اكثر من شرط بنفس الجملة واذا تحقق شرط منهم ينفذه وينهي الجملة ببساطة بنقدر نضيف على الجملة أمر elseif (بنضيف شرط جديد اذا تحقق بنفذ مجموع من الأوامر احنا كتبناها) .



انتبه:

1- البرنامج راح يبلش بالجمل بالترتيب واذا تم تحقيق أي جملة راح ينفذها ويخرج من امر if كامل.

2- اذا لم يتم تحقيق الشرط في أي جملة ينفذ البرنامج ما داخل امر else

3- امر else مش ضروري يكون موجود

4- يمكن تكرار elseif داخل الجملة اكثر من مرة

```
If condition(1)
   ...
   ...
elseif condition(2)
   ...
   ...
else
   ...
   ...
end
```

Example 3:

* سنأخذ نفس المثال السابق مع تعديل بسيط

```
clear

n= input('enter any number  ');
if n > 0
    disp('your number is positive')
elseif n==0
    disp('your number is zero')
else
    disp('your number is negative')
end
```

1- يستقبل البرنامج متغير اسمه n
2- يدخل البرنامج امر if
3- اول شرط: n اكبر من صفر اذا كان صحيح راح يطبع " your number is positive "
وبعدها بطلع من امر if
4- اذا كان اول شرط خطا راح يروح على ثاني شرط (ال elseif) والي هو اذا كانت n=0 او لا
واذا تحقق الشرط بنفذ البرنامج وبخرج من if
5- اذا كان الشرط خطأ بتكون انتهت جميع جمل الشروط وما تحقق ولا واحد منها فبنفذ ما داخل else

Example 4:

```
n= 5;
if n>0
    disp('first one')
elseif n == 5
    disp('second')
else
    disp('none')
end
```

Command Window

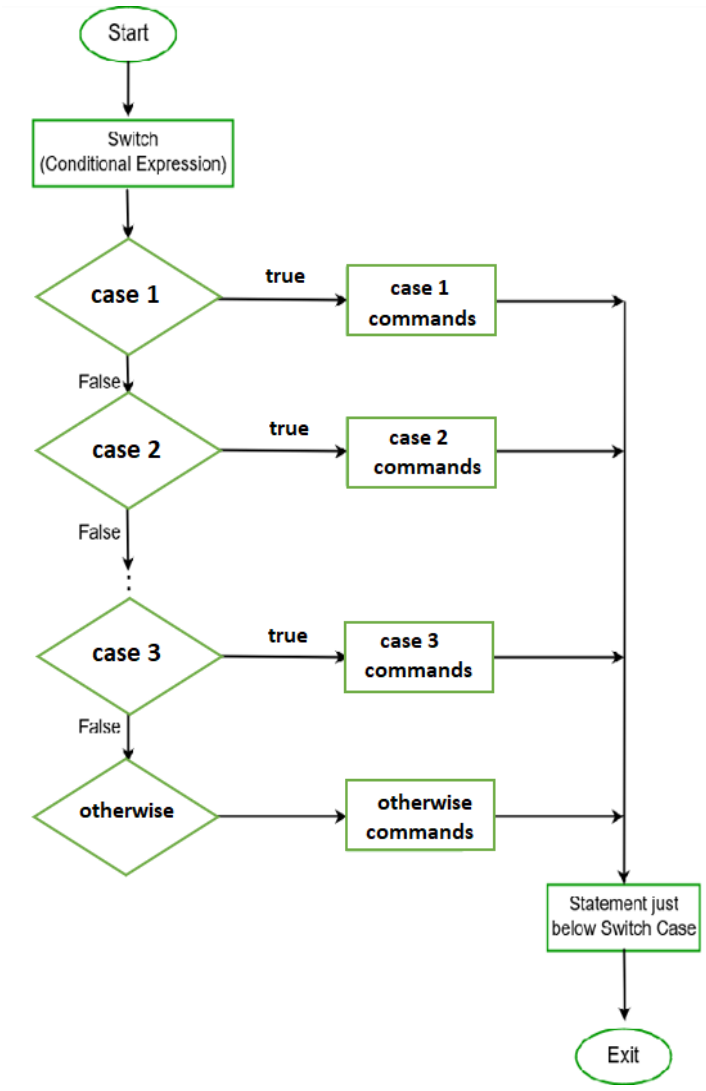
```
first one
```

نلاحظ في هذا المثال ان المتغير n قيمته 5 وان الشرط الأول ان تكون n اكبر من صفر والشرط الثاني بأن تكون n تساوي 5... وهذا يعني بأن الشرطان صحيحان
لكن عندما نضغط على run ونشغل البرنامج نجد انه قد قام بطباعة ناتج اول جملة فقط(first)
وذلك لان البرنامج يخرج من امر if كاملا بمجرد تحقق احد الشروط.

Switch, otherwise, end

منطق الأمر هو نفسه من ال c++ حيث يقوم بحساب ناتج تعبير معين وحسب نناتج هذا التعبير يقوم بالاختيار من بين عدة حالات (cases) واذا كان الناتج لا يطابق أي حالة ينفذ ما في داخل otherwise.

وكما الأمر في elseif يقوم البرنامج بفحص ال cases بالترتيب وفي حال تحققت واحدة فانه ينفذ ما في داخلها ويخرج من امر switch.



Switch expression

Case ( )

...

...

Case ( )

...

...

Case ( )

...

...

Otherwise

...

...

end

Example 5:

```
n = input('Enter a number: ');
switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

في هذا الكود:

1- يستقبل البرنامج متغير اسمه n
2- يدخل جملة switch والتي ناتج التعبير داخلها هو n (لم يتغير عليه شيء)
3- يقوم بفحص ال cases بالترتيب وفي حال تحقق منها أي ينفذ الكود داخلها ويخرج من الجملة فاذا كان 1 او 0 او -1 راح ينفذ ال case الخاصة فيها.
4- وفي حال عدم تحقق أي وحدة من الجمل سينفذ otherwiseويقوم بطباع 'other value' .

Example 6:

Write a code that takes number as an input and save it in 'x' , assume the input will be from 1-12, the code must return the name of the month of that number(for example 4 is April) and if the number is not an integer form 1-12 return the statement **"you must enter integer form 1 to 12"**. **Note**: use "switch case".

بده اياك تكتب كود بستقبل رقم وبده اياك تطبع اسم الشهر الي بوافق هاض الرقم واذا كان الرقم مش عدد صحيح بين ال 1 و ال 12 يطلعلك المسج المكتوب فوق.

```
n = input('Enter the number of the month : ');
switch n
    case 1
        disp('january')
    case 2
        disp('february')
    case 3
        disp('march')
    case 4
        disp('April')
    case 5
        disp('may')
    case 6
        disp('june')
    case 7
        disp('july')
    case 8
        disp('August')
    case 9
        disp('september')
    case 10
        disp('october')
    case 11
        disp('november')
    case 12
        disp('december')
    otherwise
        disp('you must enter integer form 1 to 12')
end
```

Section 6

Loops

Introduction:

في اغلب الأحيان عند كتابة البرامج نحتاج لتكرار امر معين عدد كبير من المرات لهاذا ليس من المنطق ان نقوم بكتابة نفس الأوامر مراراً لذلك يمكننا ان نستعمل ال loops بأوامرها for و while من اجل الاختصار وتسهيل كتابة البرامج.

For loops:

- نستعمل أمر for عندما نعلم عدد المرات التي نريد تكرار الأمر فيها

For variable = start number : step : end number

.

.

.

End

- بفضل ان تكون ال step=1 وذلك لتجنب الوقوع في الأخطاء أثناء كتابة البرنامج.
- اذا لم توضع ال step وكتبت كالتالي variable = start number : end number يعتبر البرنامج انها 1 تلقائياً.

Examples:

(1)

```
clc
clear

for i=6:2:12
    disp(i)
end
```

Command Window

```
     6
     8
    10
    12
```

(2)

```
clc
clear

for i=1:4
    x=i*2;
   disp(x)
end
```

Command Window

```
     2
     4
     6
     8
```

- لاحظ في المثال رقم 2 تم تعريف x بقيمة i*2 لذلك كانت قيمتها في كل دورة 2 ثم 4 ثم 6 ثم 8 بالترتيب ويمكنك ان تلاحظ ان x في كل مرة كانت تأخذ قيمة جديدة بدل قيمتها السابقة ولم تعرفها ك matrix بالصورة التالية x=[2 4 6 8] وهذا منطقي نظرا للصيغة التي تم تعريف x بها.

- لذلك من اجل تعريف matrix داخل loop نقوم بذلك بالصيغة التالية

Variable (counter) =

ففي المثال السابق اذا اردنا ان نعرفها ك matrix نكتبها كالتالي x(i) = i*2 حيث x هنا هي ال variable و i هي ال counter حيث انها تزداد في كل دورة بمقدار 1 وبالتالي ستنتج لنا x=[2 4 6 8]

Example 1:

Write program to store numbers between 1 and 10 using loop.

```
clc
clear

for i=1:10
    x(i) = i;
end
x
```

Command Window

```
x =

     1     2     3     4     5     6     7     8     9    10
```

Example 2:

Write a program to store even numbers between 1 and 20 in a variable named ‘even’ using loop.

```
clc
clear

counter=1;
for i=1:20
    if mod(i,2)==0
        even(counter)=i;
        counter=counter+1;
    end
end
even
```

Command Window

```
even =

     2     4     6     8    10    12    14    16    18    20
```

Example 3:

Writer program to define two matrices in the same for loop

X = [1 3 5 7 9 11 13 15]

Y = [15 13 11 9 7 5 3 1]

```
clc
clear

for i=1:8
    x(i)= 2*i-1 ;
    y(i)= 17 -2*i ;
end
x
y
```

Command Window

```
x =
     1     3     5     7     9    11    13    15
y =
    15    13    11     9     7     5     3     1
```

حل آخر:

```
clc
clear

counter=1;
for i=1:15
    if mod(i,2)~=0
        x(counter)= i ;
        y(counter)= 16 -i ;
        counter = counter+1 ;
    end
end
```

Command Window

```
x =
     1     3     5     7     9    11    13    15
y =
    15    13    11     9     7     5     3     1
```

While loops

- نستخدم امر while عندما نريد تكرار امر ما لكن لا نعرف كم عدد المرات التي نريد تكراره فيها وإنما نريد انهائها عند تحقق شرط معين.

While logical expression

.

.

.

End

ملاحظات هامة حول الwhile loop :

1- يجب تعريف المتغير الموجود داخل ال logical expression قبل البدء بال loop.
2- يجب ان تتغير قيمة المتغير داخل loop في كل دورة.
3- تأكد من أن المتغير سيصل الى قيمة تؤدي الى انهاء ال loop تجنبا لإحداث ال infinite loop

Example:

Write a program to find all full-squared numbers between 1 and 150 using while loop.

```
1 -   clc
2 -   clear
3
4 -   x=1;
5 -   while x^2 <=150
6 -       num=x^2;
7 -       disp(num)
8 -       x=x+1;
9 -   end
10
```

Command Window

```
     1
     4
     9
    16
    25
    36
    49
    64
    81
   100
   121
   144
```

هنا قمنا باستعمال ال loop while لأننا لا نعرف عدد المرات التي ستتكرر فيها ال loop لكننا نعرف الشرط المراد تحقيقه وهو ان لا يتجاوز مربع العدد قيمة معينة وهي 150.

Example: factorial

Write a MATLAB script to calculate the factorial of number 10 using while loop (don't use any predefined commands).

```
1 -    n = 10;
2 -    f = n;
3 -    counter = n ;
4 -    while counter > 1
5 -        counter = counter-1;
6 -        f = f*counter;
7 -    end
8 -    fprintf('    %d! = %d \n',n,f)
```

Command Window

```
>> factor
   10! = 3628800
```

Infinite loops:

يحدث في بعض الأحيان أن ننسى كتابة سطر من اجل قيمة المتغير الموجود داخل الشرط او ان قيمة المتغير ستظل تحقق الشرط الى ال لانهاية وتسمى هذه الحالة **infinite loop** حيث تعني ان البرنامج سيظل ينفذ بال loop مرارا دون الوصول الى نهاية وبطبيعة الحال نحن لا نريد حدوث لذلك يجب ان نتأكد من:

1- ان نحرض على وجود كود يغير من قيمة المتغير في داخل ال loop
2- ان نتأكد بأن المتغير سيصل الى مرحلة لا ينطبق شرط ال while عليه
3- في حال حدوث infinite loop يمكنك ايقافها عن طريق pause بعدها quit من ال editor او تضغط ctrl + C داخل ال workspace

```
1 -    n = 10;
2 -    f = n;
3 -    counter = n ;
4 -    while counter > 1
5
6 -        f = f+1
7 -    end
8 -    fprintf('    %d! = %d \n',n,f)
```

**في المثال أعلاه infinite loop لان counter لا تتغير داخل ال while loop

Break

عندما نقوم بكتابة بعض البرامج مستخدمين ال for loop او ال while loop نحتاج الى ان يقوم البرنامج بالخروج من ال loop في حال تحضض شرط معين وذلك لتجن الوقوع في الأخطاء او حدوث error، وهذا بالضبط ما يساعدنا امر break على فعله حيث يقوم الأمر بإيقاف ال loop التي يجري تنفيذها حاليا ويخرج منها نهائياً

Example:

Write a script to define a vector named x which contains 10 random numbers and make it stop when there is element is grater then 0.9(use for loop)

```
1 -    clc
2 -    clear
3 -    for i=1:10
4 -        x(i)= rand;
5 -        if x(i)>0.9
6 -            break
7 -        end
8 -    end
```

Command Window

```
x =

    0.7287    0.7378    0.0634    0.8604    0.9344
```

Nested loops

عندما نستخدم الماتلاب قد نحتاج الى استخدام اكثر من loop داخل بعضها في نفس الوقت وفي الغالب نحتاج الى ذلك عند انشاء او تعديل على matrix تتكون من عدة صفوف وأعمدة.

Example:

Write a program to define a matrix named x contains **5 columns** and **5 rows** and the elements inside it is the numbers from 1 to 25 using for loop.

```
1 -    clc
2 -    clear
3 -    s=1;
4 -    for i=1:5
5 -        for j=1:5
6 -            x(i,j)=s;
7 -            s=s+1;
8 -        end
9 -    end
```

body of the outer loop

body of the inner loop

Command Window

```
x =

     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
    16    17    18    19    20
    21    22    23    24    25
```

في المثال أعلاه استعملنا ال for loop مرتين

المرة الأولى (عندما عرفنا ال i) كانت من اجل التنقل بين الصفوف

المرة الثانية (عندما عرفنا ال j) كانت من اجل التنقل بين الأعمدة

Example 2:

Create a 5X5 matrix called “mat1” and let the elements of the diagonal be equal to 1 and every other element is 0.

```
for i=1:5
    for j=1:5

        if i==j
            mat1(i,j)= 1;
        else
            mat1(i,j) = 0;
        end

    end
end
```

Command Window

```
mat1 =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

تطبيق هندسي على ال loop (self-study)

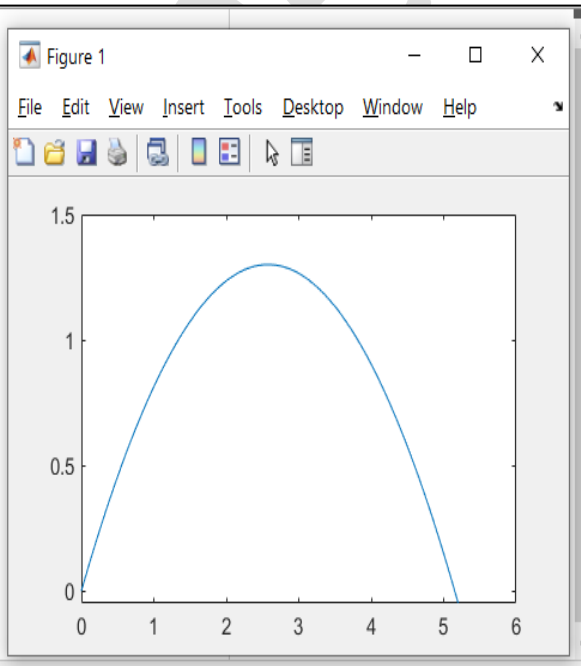
Projectile motion

بالتأكيد يمكننا استخدام ال loops والاستفادة من العمليات الحسابة التي تستطيع اجراءها في أجزاء من الثانية في العديد من التطبيقات الهندسية وواحدة منها هي حسابات ال projectile motion (المقذوفات) ولنأخذ المثال التالي:

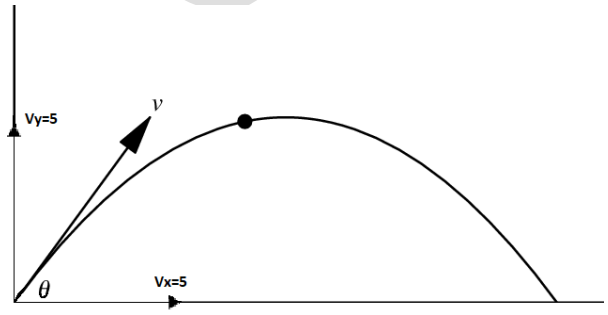
Example: MATLAB code for projectile motion

Write a MATLAB code to find the flying time and the distance in x direction for a projectile thrown at initial speed Vx=5 and Vy=5 , assume x(1)=0 and y(1)=0 then plot the graph of x and y distance.

```
clear all;
close
x(1)=0;
y(1)=0;
vx(1)=5;
vy(1)=5;
dt=0.01;
t(1)=0;
i=1;
while y>=0
x(i+1)=x(i)+vx(i)*dt;
y(i+1)=y(i)+vy(i)*dt;
vx(i+1)=vx(i);
vy(i+1)=vy(i)-9.8*dt;
t(i+1)=t(i)+dt;
i=i+1;
end
fprintf('the distance in x directon = %d \n ',x(i));
fprintf('the total time in the air = %d \n',t(i));
plot(x,y);
```




```
Command Window
>> dfd
the distance in x directon = 5.200000e+00
 the total time in the air = 1.040000e+00
```



Section 7

Functions

Introduction:

سنتعرف في هذا الفصل على ال functions وكيفية تعريفها واستخدامها وانواعها.

عندما نستعمل الماتلاب نحتاج أحيانا ان نستعمل سلسلة من الأوامر أكثر من مرة مثل (التحويل بين الوحد، او سلسلة عمليات حسابية،...) وليس من المنطقي ان نقوم بتكرار نفس الأوامر في كل مرة لذلك يتيح لك الماتلاب ان تعرف ما يسمى بال function بعد ان تعرفه للقيام بالأوامر التي كتبتها داخله.

أولاً: ما هو ال function؟

هو ملف يشبه ال script لكنه يحتوي على مجموعة من الأوامر يقوم باستقبال متغيرات inputs ويقوم بإنتاج outputs.

- ال functions في الماتلاب تستقبل متغير او اثين الى أي عدد قد تحدده انت وأيضا يمكن ان لا يوجد هناك أي عنصر مُدخل (input) وكذلك الأمر بالنسبة للمخرجات تستطيع وتحديد العدد الذي تريده وقد لا يكون هناك أي (output).

❖ **طيب ليش بدنا نستعمل ال functions؟؟**

1- تساعدك على تقسيم عملك الى أجزاء أصغر (بدل ما يكون عندك برنامج واحد كبير تستطيع تقسيمه الى أجزاء أصغر وإخفاء أجزاء من الكود انت ليست بحاجة لرؤيتها مما يجعله ابسط للفهم والتعديل).
2- يسهل عملية ال debugging او تحديد الأخطاء في كتابة الكود لأنه مقسم لأجزاء أصغر
3- تستطيع استعمال ال function أكثر من مرة في البرنامج وفي برامج أخرى أيضا مما يوفر الوقت.

❖ ويجدر بنا الذكر بان هناك نوعين من ال functions وهما ال pre-defined functions والذي يكون معرف من قبل البرنامج والمكتبات المخزنة داخله. والنوع الآخر هو ال user-defined functions والتي يقوم المستخدم بتعريفها وتحديد وظيفتها. وتنقسم أيضاً الى local التي يمكنك استخدامها فقط داخل ال script الذي يتم العمل داخله و global التي يمكن استخدامها دائما.

Pre-defined functions

وهي ال functions التي تكون موجودة مع الماتلاب ومخزنة في مكتباته ولا دخل للمستخدم بكتابتها او تعريفها مثل (sin, cos, tan, sqrt, max, min, …)

❖ **أمثلة على predefined functions:**

Command Window

```
>> what('lang')

MATLAB Code files in folder C:\Program Files\Polyspace\R2019a\toolbox\matlab\lang

Contents                        genvarname                  parfor
ParallelException               global                      parfor_M_check
ans                             handle                      parfor_colon_range_check
assert                          if                          parfor_endpoint_check
assignin                        input                       parfor_nested_for_endpoint_check
break                           inputname                   parfor_nested_for_range_check
builtin                         iskeyword                   parfor_nested_for_range_step_check
case                            isvarname                   parfor_range_check
catch                           javachk                     parfor_range_step_check
checkSyntacticWarnings          keyboard                    parfor_reduction_variables_set_check
classdef                        lasterr                     parfor_sliced_fcnhdl_check
clearAllMemoizedCaches          lasterror                   parfor_sliced_offset_check
consume_assign                  lastwarn                    persistent
continue                        lists                       precedence
details                         localfunctions              rethrow
disp                            memoize                     return
display                         message                     reverse_binary_operator
doclink                         mfilename                   run
else                            mislocked                   script
elseif                          mlock                       spmd_feval
end                             munlock                     switch
error                           nargchk                     try
eval                            nargin                      validateattributes
evalc                           narginchk                   validatestring
evalin                          nargout                     varargin
exist                           nargoutchk                  varargout
feval                           online_concatenator         warning
for                             otherwise                   while
function                        parallel_function
```

❖ **كيفية استدعاء ال function.**

نقوم باستدعاء ال function داخل ال command window او ال script بالطريقة التالية:

Function name (input1, input2, …)

في معظم الأحيان نحتاج الى تخزين الناتج داخل متغير لاستعماله لاحقاً فنكتبه بهذه الطريقة:

Variable = function name (input1, input2, …)

User-defined functions

هي ال functions التي يقوم المستخدم بتعريفها واختيار اسمها وتحديد وظيفتها ويتم حفظها في ملف ينتهي بـ (.m).

يجب ان تنتبه عند تسمية ال functionالى عدة نقاط:

1- مسموح فقط بأن يتكون الاسم من الأحرف (a-z) والأرقام (0-9) وال underscore (_).
2- يجب أن يبدأ الاسم بحرف.
3- يجب أن لا يكون الاسم مستخدم محجوز (اسم ل function او أمر آخر بالماتلاب).

** الماتلاب يُفرق بين الحروف الصغيرة والكبيرة(max ليست نفسها MAX).

❖ **يوجد نوعان من ال user-defined وهما ال local و ال global.**

Local functions:

هي ال functionsالمعرفة داخل script معين ولا تعمل خارجه تتميز بسهولة كتابتها لكنها غير عملة في كثير من الأحيان.... تكتب بالصيغة التالية:

Function name = @ (variable) function statement

**يمكنك تسمية المتغير بأي اسم تريده حتى لو تكرر خارج ال function ولن يؤثر ذلك بشيء على البرنامج

Example

Write a local function called y with a value equals to '' x2+1" then find y (5), y (2) and y (3.77)

Solution: **y = @(x) x^2+1**

```
2 -    y=@(x)x^2+1;
3 -    y(5)
4 -    y(2)
5 -    y(3.77)
```

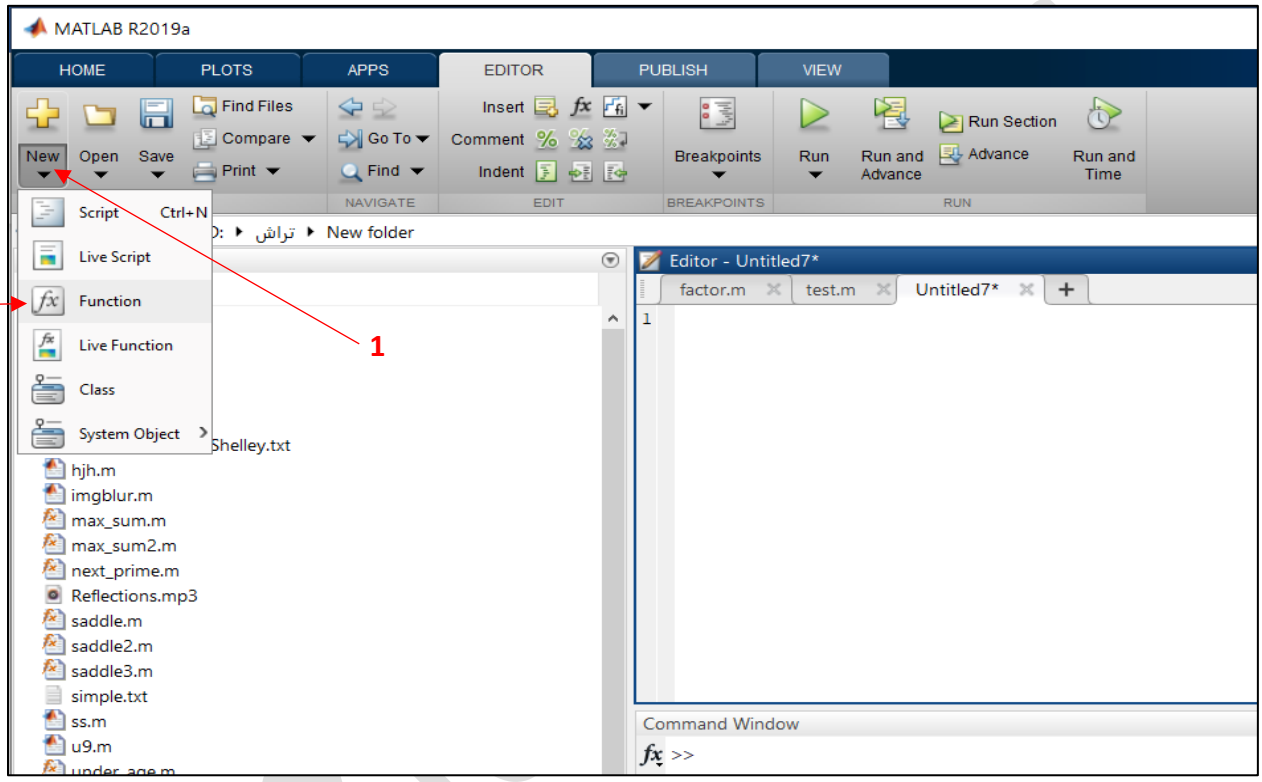
```
Command Window
  ans =
      26
  ans =
       5
  ans =
     15.2129
```

Global functions

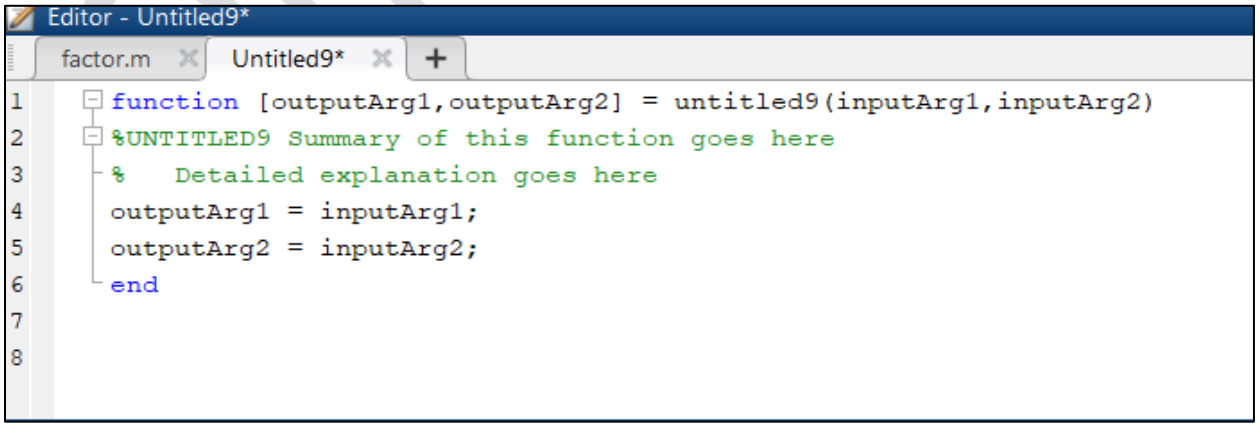
هي ال functions التي نستطيع استعمالها في أكثر من script حيث تكون مخزنة في m.file خاص فيها موجود في نفس الملف مع ال scripts الأخرى وهي ما نتعامل معه في الغالب.

كيفية إنشاء global function:

- من لائحة new اختار function



- ستظهر هذه الشاشة امامنا وكل ما علينا هو التعديل على ما في داخلها كما سنتعلم بعد قليل



Functions formula

Function [out1 , out2 , …, out n] =function_ name (inp 1, inp 2 , …., inp n)

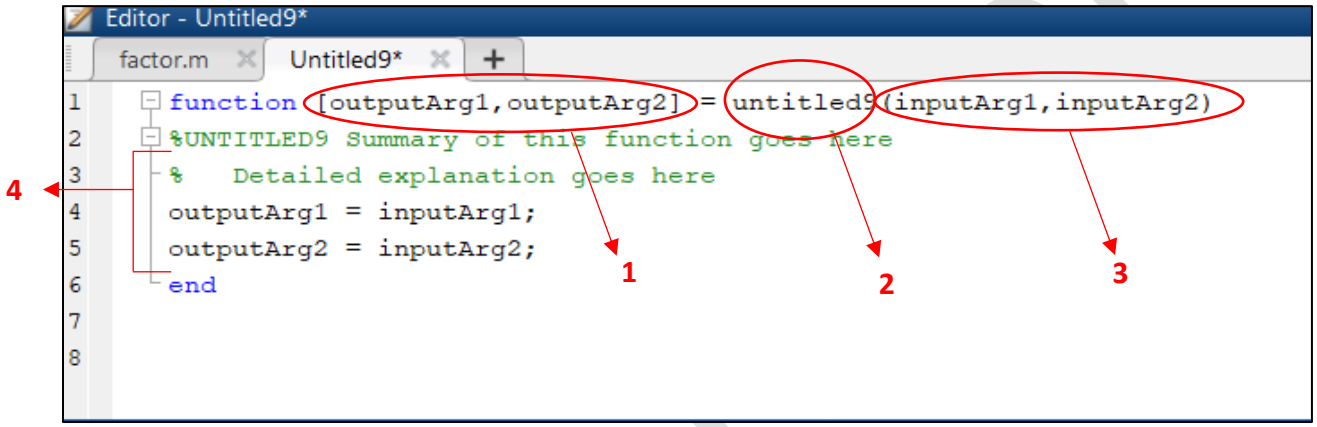
Perform task 1

:

:

Perform task n

End



1. Output argument

نضع هنا اسم متغير للمخرج او الناتج الذي نريده من ال function ويمكن ان يكون مخرج واحد او أكثر من مخرج، وتسميته ليست مرتبطة بال script الذي يتم استعماله فيه. يعني بإمكانك انه تسمي المتغير الاسم الي بناسبك داخل ال function اما خارجه فلن يؤثر او يظهر هاذ الاسم.

2. Function name

هو الاسم الذي ستختاره لل function وتكتبه بال script من اجل استدعائه وتنطبق عليه نفس القواعد التي تنطبق على تسمية المتغيرات.

3. Input argument

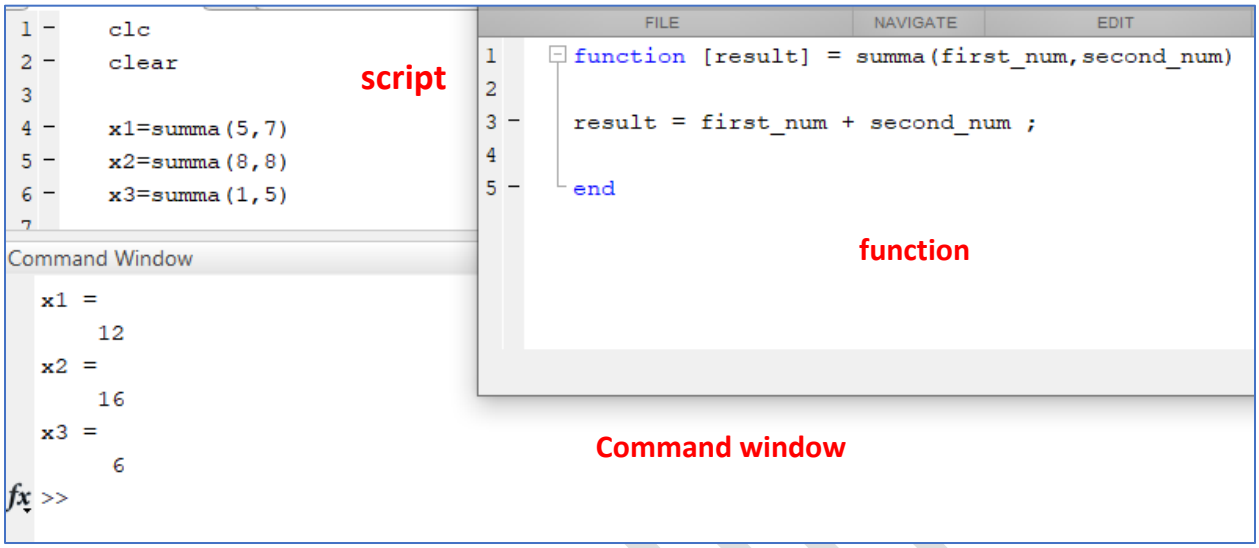
هي المدخلات أو المتغيرات التي يدخلها المستخدم في ال script ليتم حساب الناتج بناءً عليها ويفضل ان يتم تسميتها بأسماء مفيدة لها معنى وتوضح ما هو استخدامها في البرنامج.
على سبيل المثال اذا كانت سرعة يفضل تسميتها v او velocity واذا كانت درجة حرارة T وذلك لأنها ستساعد المستخدم في معرفة ما عليه ان يدخله عند استعمال ال function مثلما سنقوم بتوضيحها لاحقا.

4. Function tasks

وهي مجموعة الأوامر التي نريد لل function أن يقوم بها ويجب أن ينتج منها قيم المخرجات او ال outputs والا سيقوم بإعطاء error عند تشغيل ال function.

Example 1

سنبدأ بأبسط مثال ممكن وهو انشاء function يقوم باستقبال رقمين وقوم بإخراج ناتج جمعهما وسنسمي هذا ال function ب summa.




```
clc
clear

x1=summa(5,7)
x2=summa(8,8)
x3=summa(1,5)
```

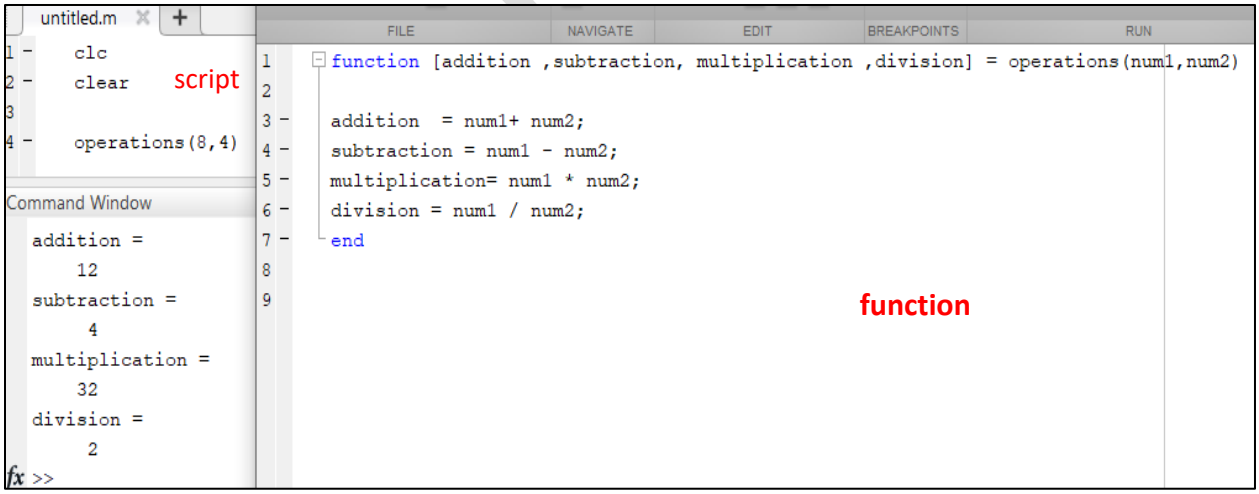
```
function [result] = summa(first_num,second_num)

result = first_num + second_num ;

end
```

```
Command Window
x1 =
    12
x2 =
    16
x3 =
     6
fx >>
```

Example 2

Write a function named operations that gives you the result of the four operations (+ , - , * , /) in order between two numbers.

**** الفرق عن المثال الأول انه يوجد هنا اكثر من output**




```
clc
clear

operations(8,4)
```

```
function [addition ,subtraction, multiplication ,division] = operations(num1,num2)

addition  = num1+ num2;
subtraction = num1 - num2;
multiplication= num1 * num2;
division = num1 / num2;
end
```

```
Command Window
addition =
    12
subtraction =
     4
multiplication =
    32
division =
     2
fx >>
```

ملاحظة مهمة:

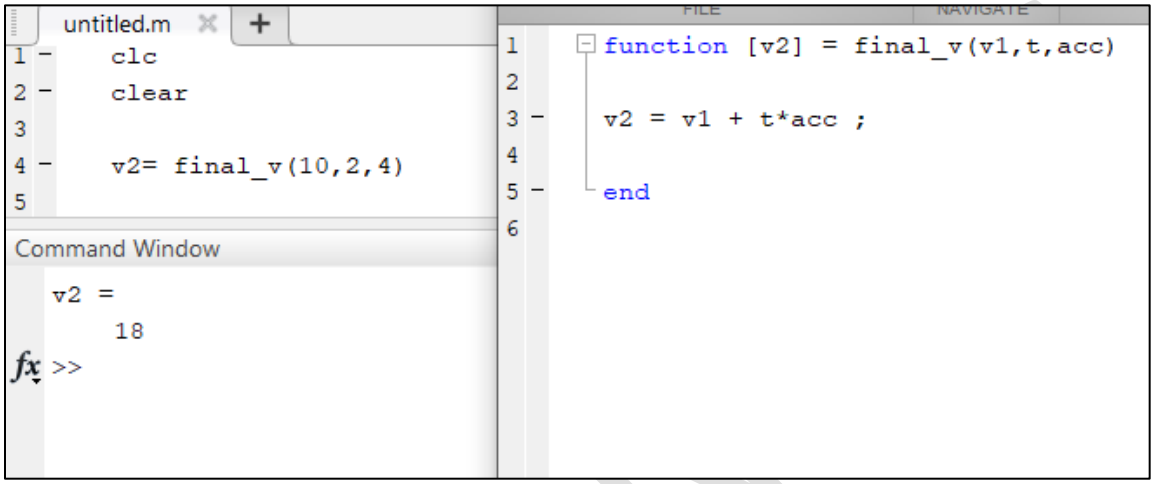
لو اردنا تخزين الناتج من ال function داخل متغيرات وهناك اكثر من output نقوم بكتابته كالتالي: (سنقوم باستخدام المسالة في الأعلى ك مثال)

[x1, x2, x3, x4] = operations (8,4)

Example 3:

Write a function called “final_v” that calculates the final velocity of an object knowing its initial velocity and total time and the acceleration.
To test your function assume v1= 10 , t=2 , acc=4 and the result should be 18

Hint: use the relation “ V2= V1 + a*t “



ملاحظة مهمة

تظهر أهمية تسمية ال inputs هنا حيث انه في هذا المثال حيث انه مطلوب منك ادخال v1 و t و acc بالترتيب ولنفرض انك نسيت ما هو الترتيب او انك اردت استعمال البرنامج بعد فترة طويلة في هذه الحالة كل ما عليك فعله هو كتابة اسم ال functionوبعده قوس مفتوح “ final_v(“ بعدها سيقوم الماتلاب ب إظهار أسماء متغيرات ال inputs كما تم انشائهم بال function (كما هو موضح بالصورة في الأسفل) وتعتبر هذه الخاصية مفيدة اذا اردت ان تتأكد من ان function موجود او ان تتذكر ترتيب المدخلات وغيرها من الأمر.