M-File

هي وسيلة لإدخال الأوامر ولكن ليس من خلال نافذة الأوامر، ولكن ماذا قد يختلف في هذه الوسيلة الجديدة في إدخال الأوامر؟

1-في عملية إدخال الأوامر التي كنا نستخدمها، إذا أردنا تعديل عنصر أو أكثر كان يجب إعادة إدخال الأمر من جديد.

2-إذا وجد خطأ، فيجب كتابة الأمر من جديد

3-إذا كتبنا برنامج كبير، وأردنا إعادة العملية مرة أخرى يجب إدخال جميع الأوامر من جديد وبنفس الترتيب.

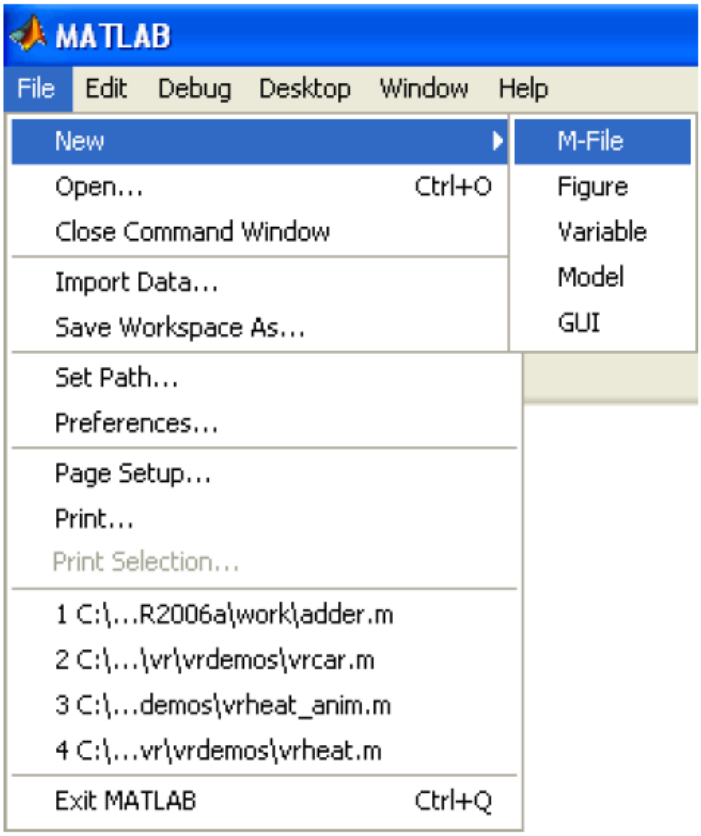
4-إذا حدث خطأ في ترتيب الأوامر لهذا البرنامج الكبير ستقوم بإعادة الإدخال الأوامر من البداية مرة إخرى.

5-يصعب عمل عملية تصحيح للأخطاءDebugging

وهذا بالطبع يستغرق وقتاً كبيراً هذا بالإضافة إلى الملل الذي يحدث للمستخدم

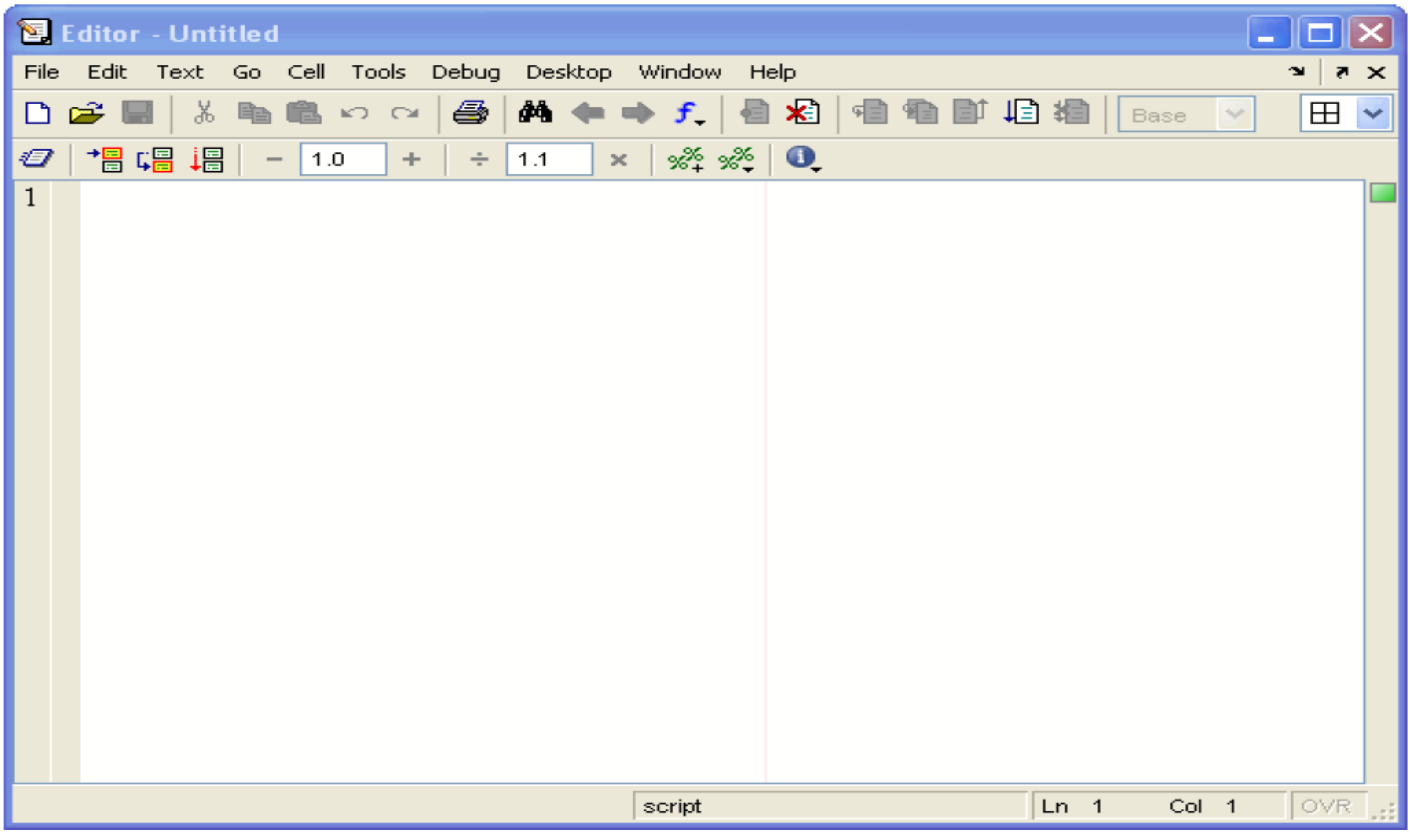
وطبعاً حلاً لهذه المشكلة، تم عمل بما يسمى M-File والتي تعطي القدرة على كتابة البرنامج كاملاً أولاً بدون تشغيل، وبعد الإنتهاء منه يتم تشغيله، هذه الخاصية تعطي القدرة على تعديل القيّم دون الحاجة إلى كتابتها مرة أخرى، أو إعادة إدخال الأوامر التي تعتمد على هذا الأمر.

فكيف يتم تشغيل تلك الخاصية؟ إتبع الصورة التالية



وبالتالي ستظهر نافذه جديدة، تأخذ الشكل التالي

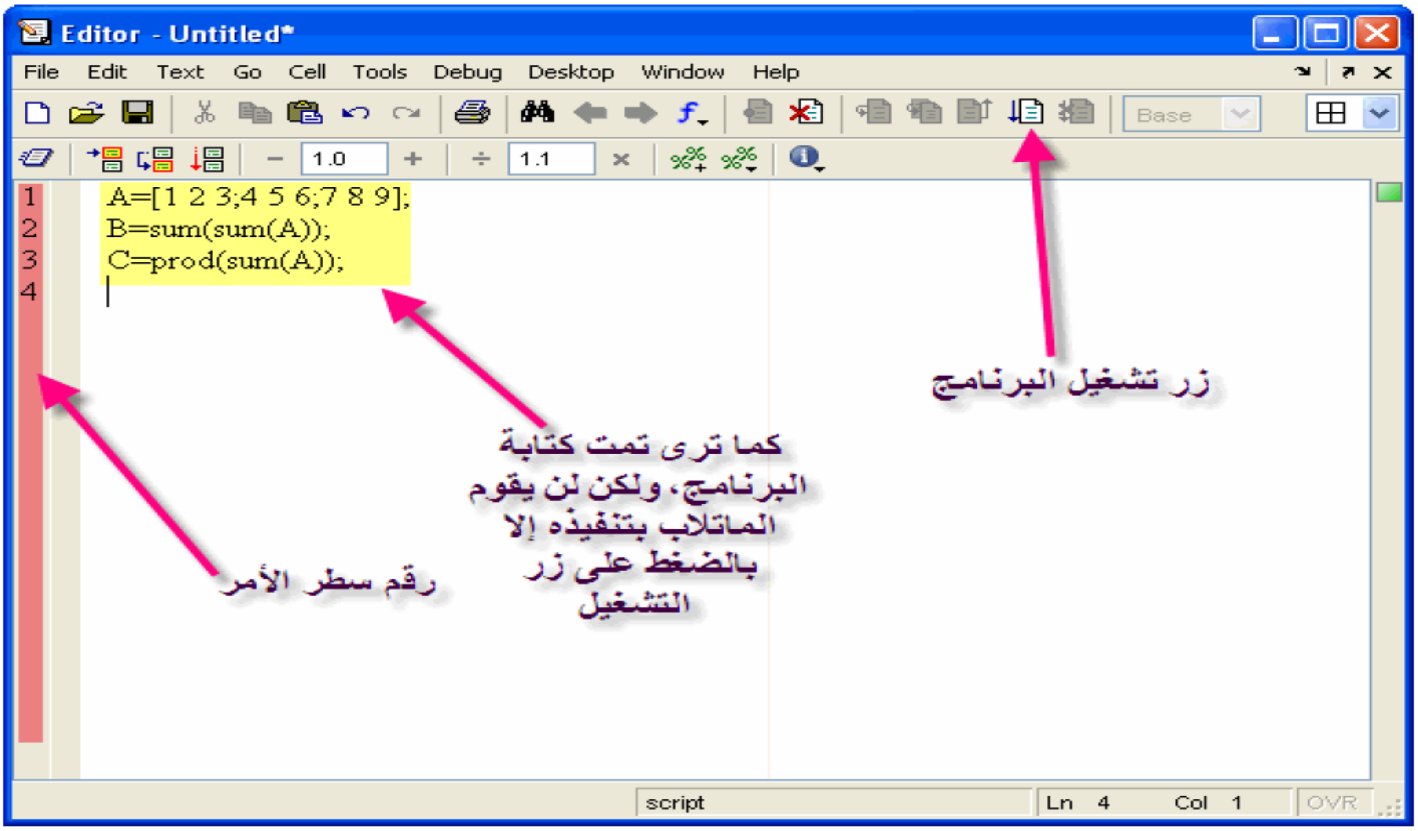
!Error



نافذةM-File

سنقوم الآن بالتعرف على نافذة M-File, أنظر الصورة التالية

!Error



ولكن عند الضغط على زر التشغيل، سيطالبك الماتلاب بحفظ البرنامج، ولكن يشترط الآتي عند حفظ البرنامج

1- أن لايبدأ بأرقام

2- أن لا يكون أمراً معرفاً في الماتلاب

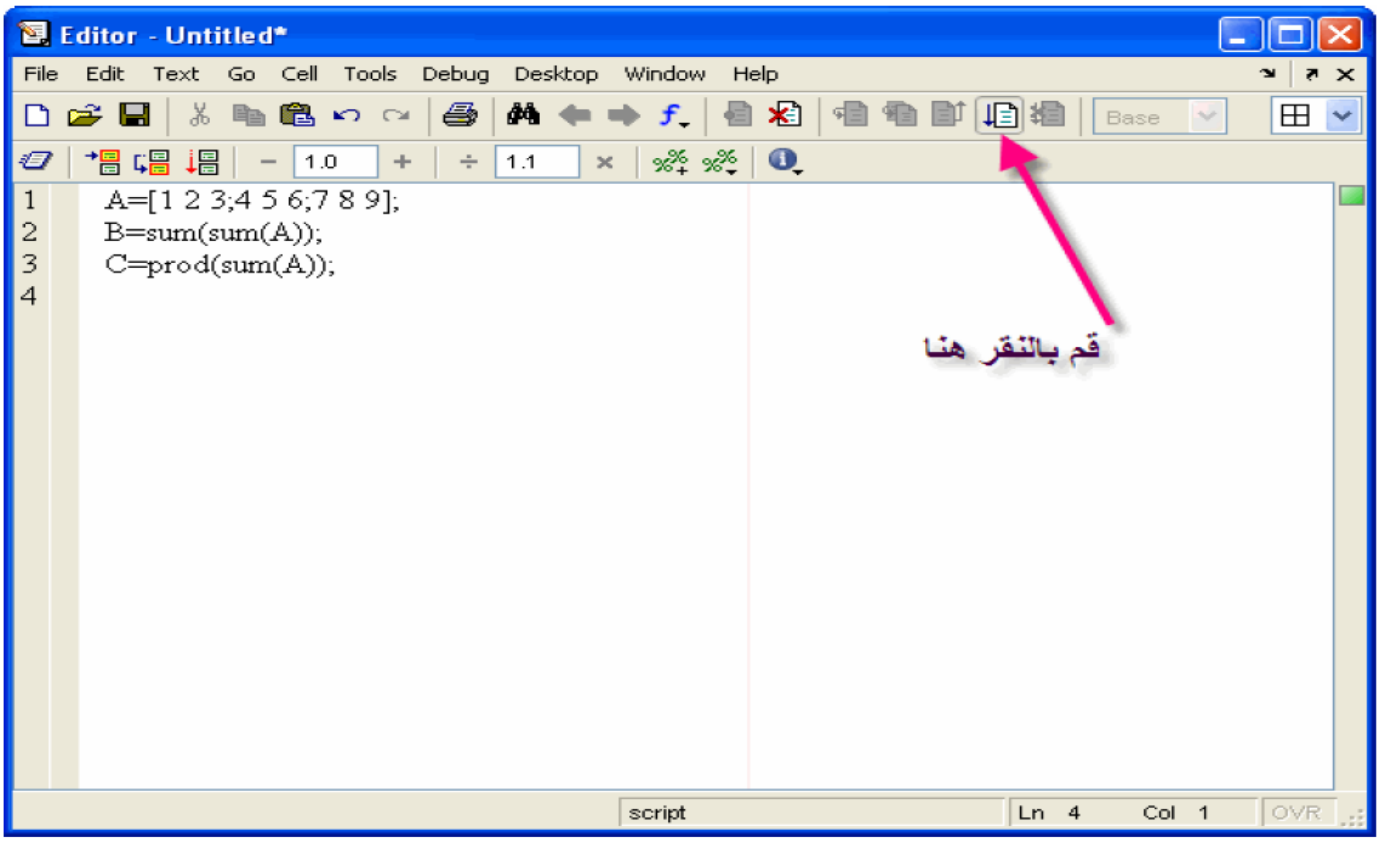
3- أن لا يحتوي الإسم على مسافات فاصلة

٤- أن لاتحتوي على رموز خاصة مثل + ، - ، & ، *

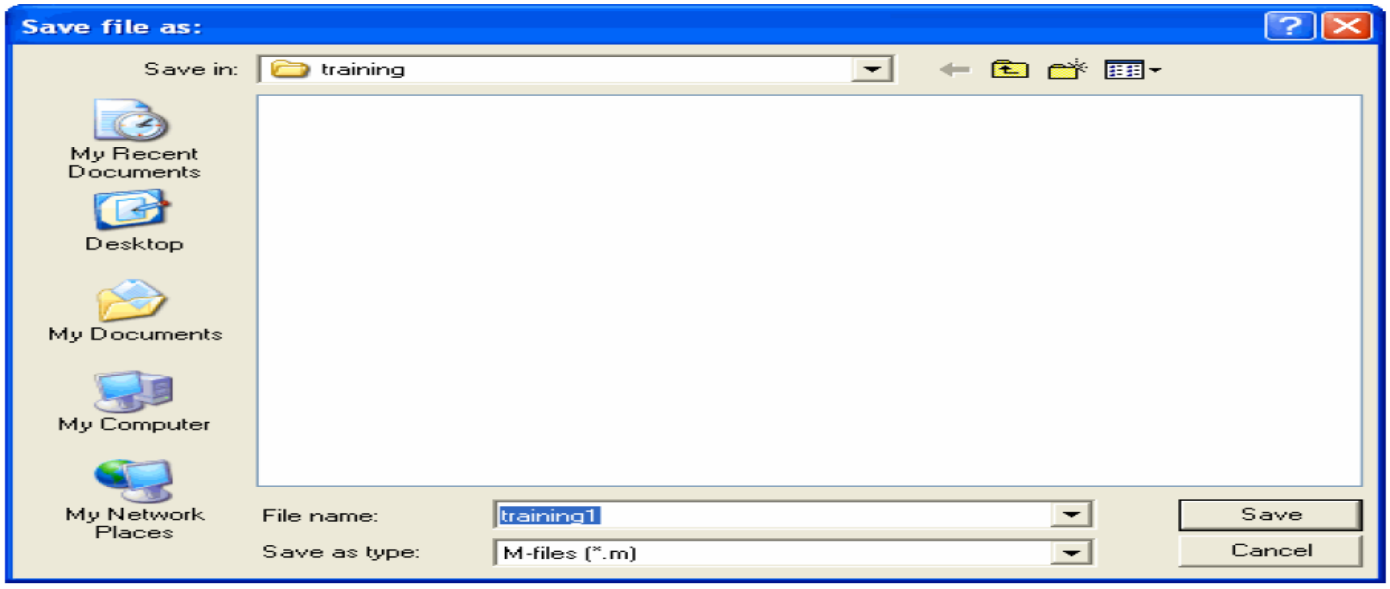
يجب مراعاة تلك الشروط وإلا لن يقوم الماتلاب بتنفيذ البرنامج

فالنقم بتنفيذ المثال المكتوب الآن في النافذة السابقة

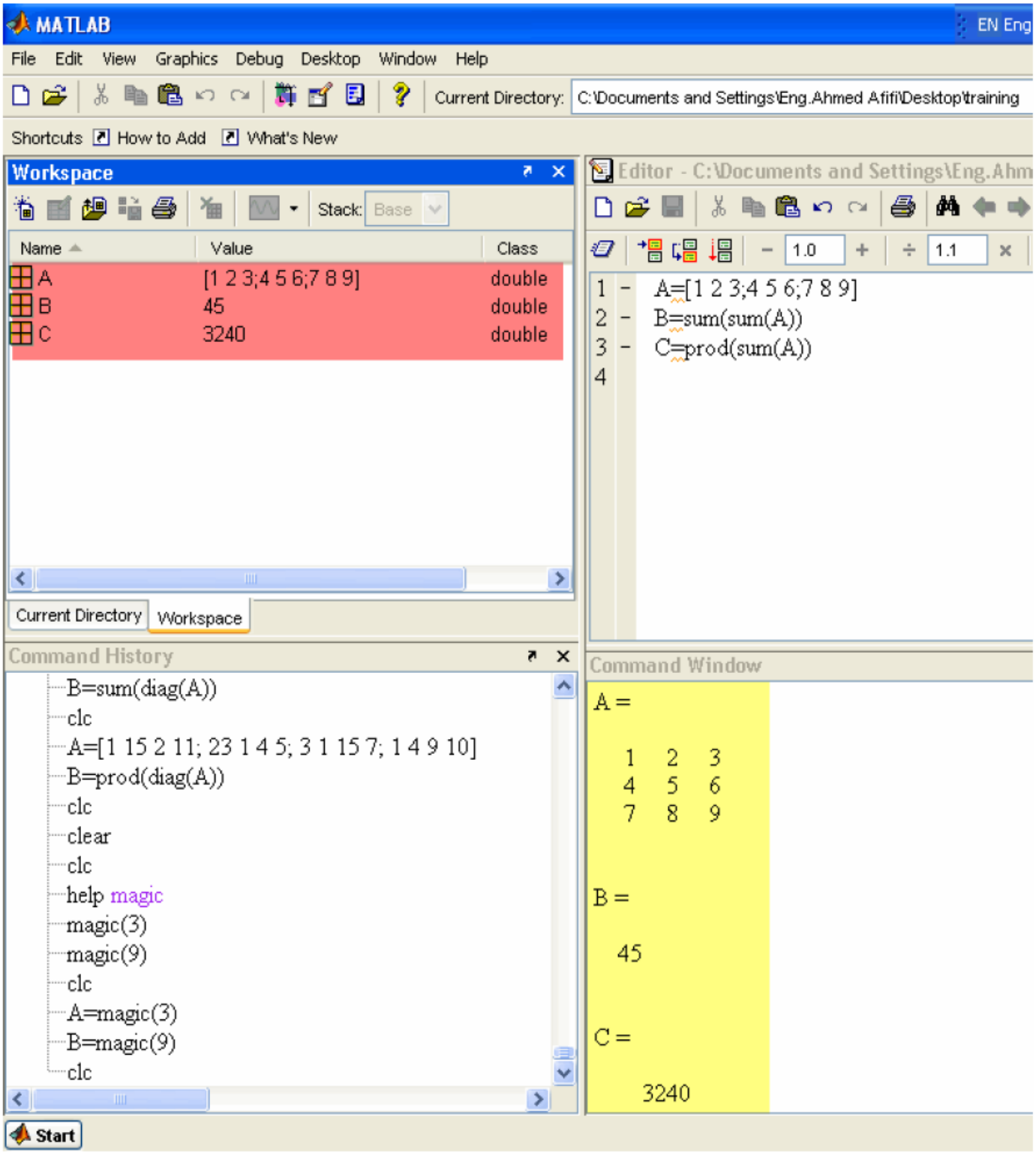
1- يتم الضغط على زر التشغيل كما هو واضح في الصورة التالية



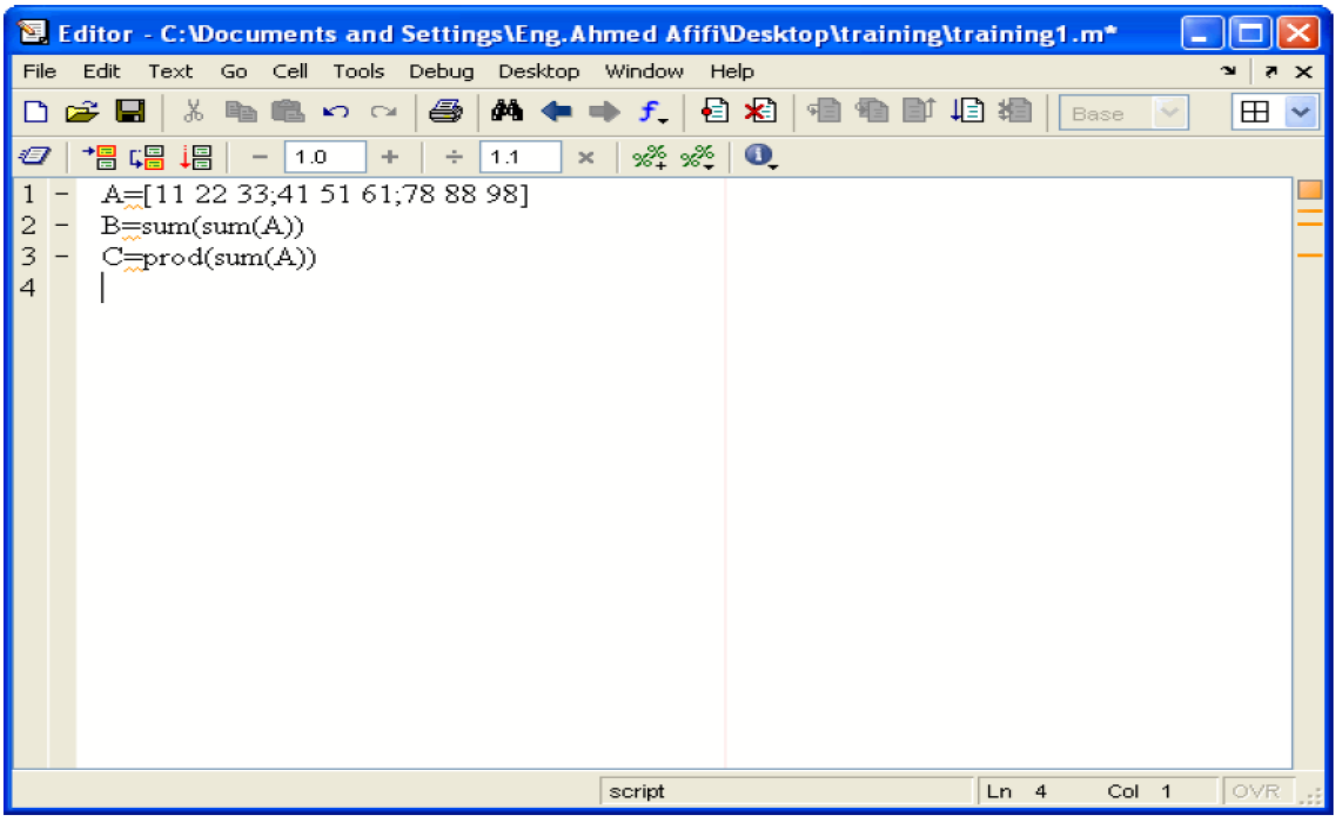
2- سيطالبنا الماتلاب بحفظ البرنامج أولاً، ولنسميه training1



3-ستظهر القيم في كلاً من Command Window and Workspace



4-لنعود إلى M-File ونقوم بتغيير بعض القيم للمصفوفة، كما في الشكل التالي



5-سنقوم الآن بتشغيل البرنامج، وسيقوم الماتلاب الآن بالحفظ تلقائياً دون الحاجة لإعادة التسمية، ثم شاهد نافذة الأوامر Command Window

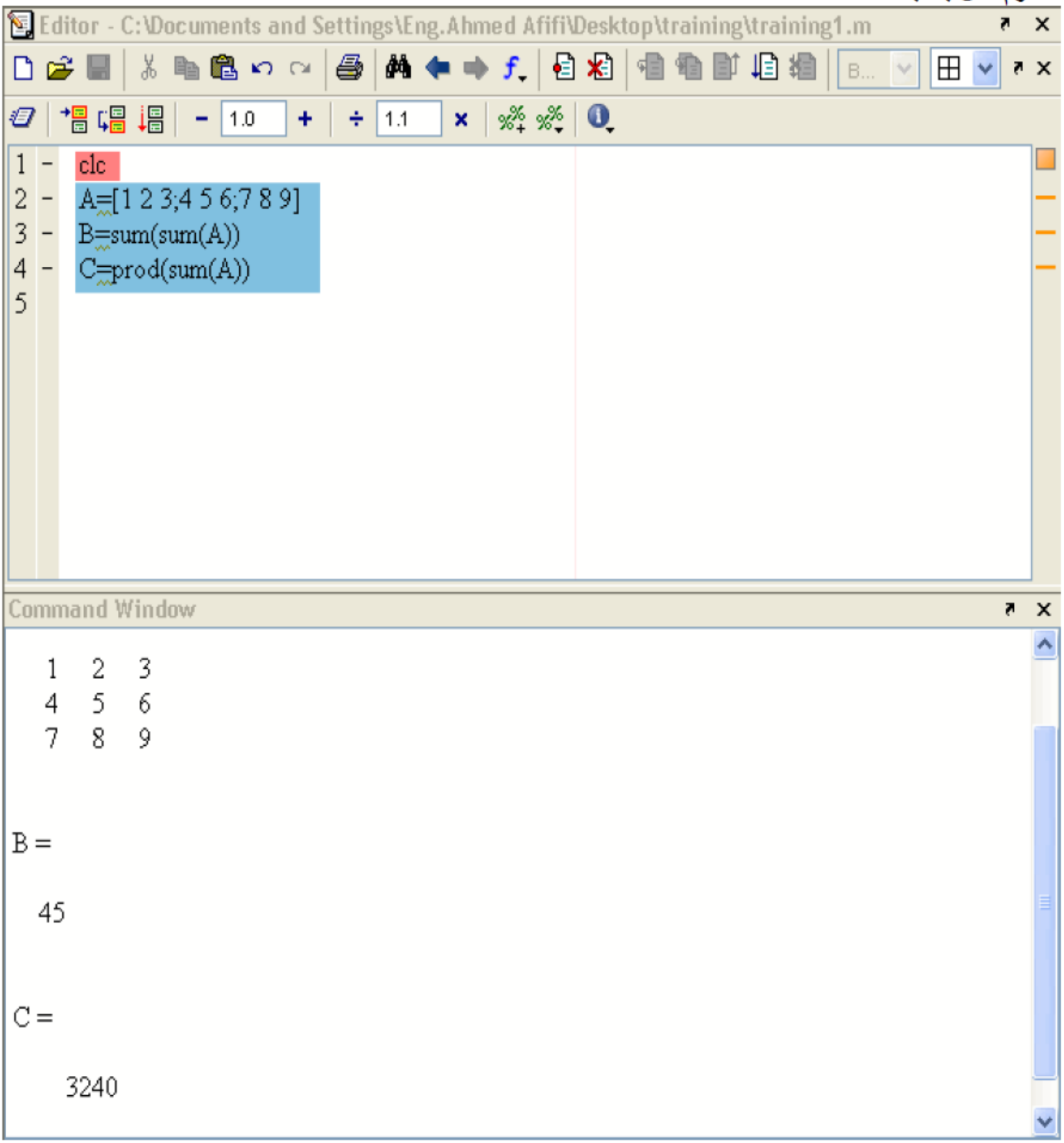
```
Command Window

    1     2     3
    4     5     6
    7     8     9

B =

   45

C =

        3240

A =

   11    22    33
   41    51    61
   78    88    98

B =

  483

C =

     4018560
```

هذه قيم البرامج التي قد حصلنا عليها منذ قليل

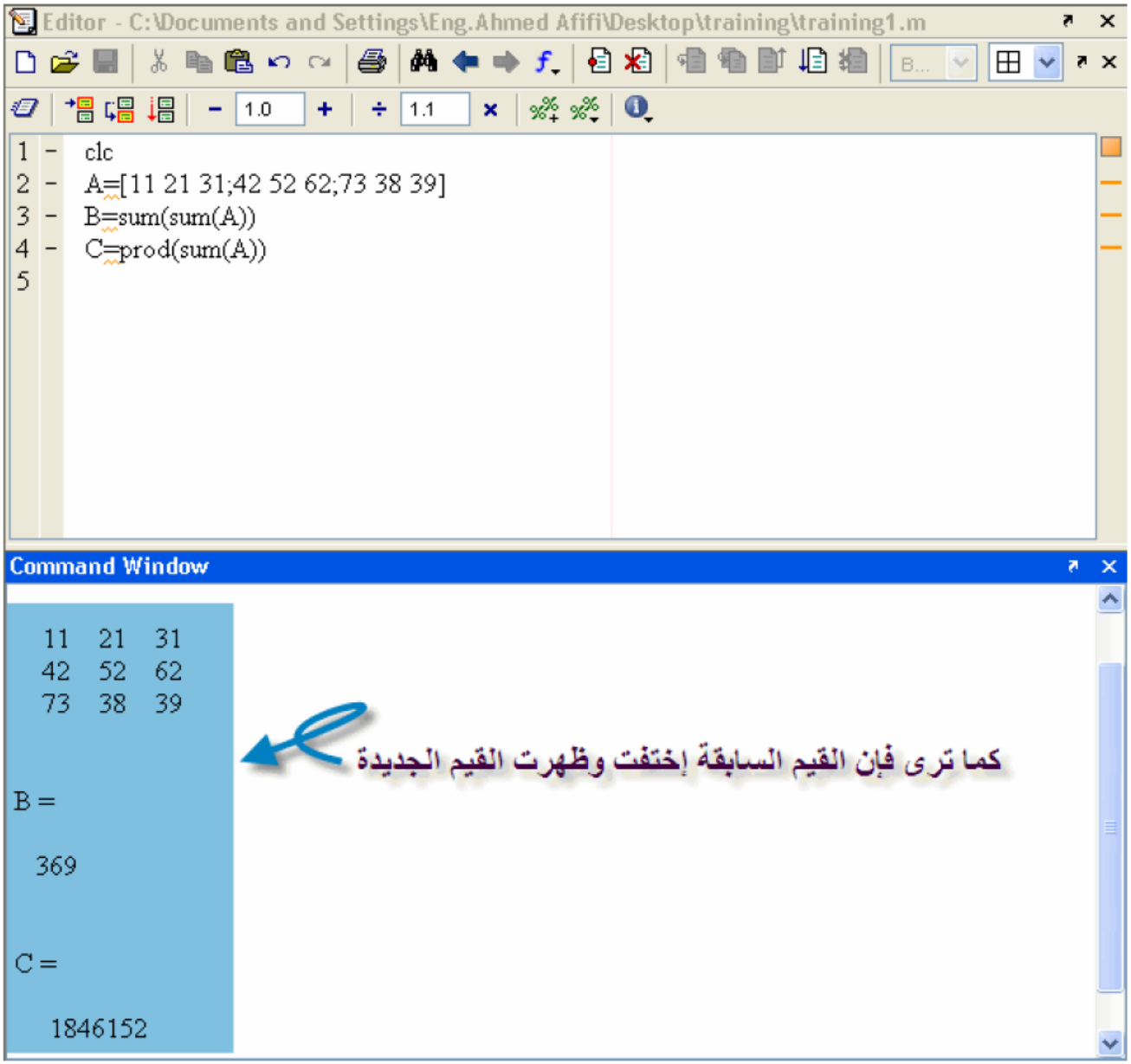
وهذه قيم البرنامج بعد عمل التعديلات عليه

وكما تلاحظ فإنه في كل عملية تحديث للبرنامج ستظل قيم البرنامج القديم موجودة ,فحلاً لهذه المشكلة، يتم وضع الأمر CLCفي أول كل برنامج، وهذا يكون مبدأ في جميع البرامج التي نقوم بعملها لابد من أن تبدأ بهذا الأمر.
ودعونا نقوم بمثال يوضح لنا ذلك

سنقوم الآن بكتابة الـــM-File

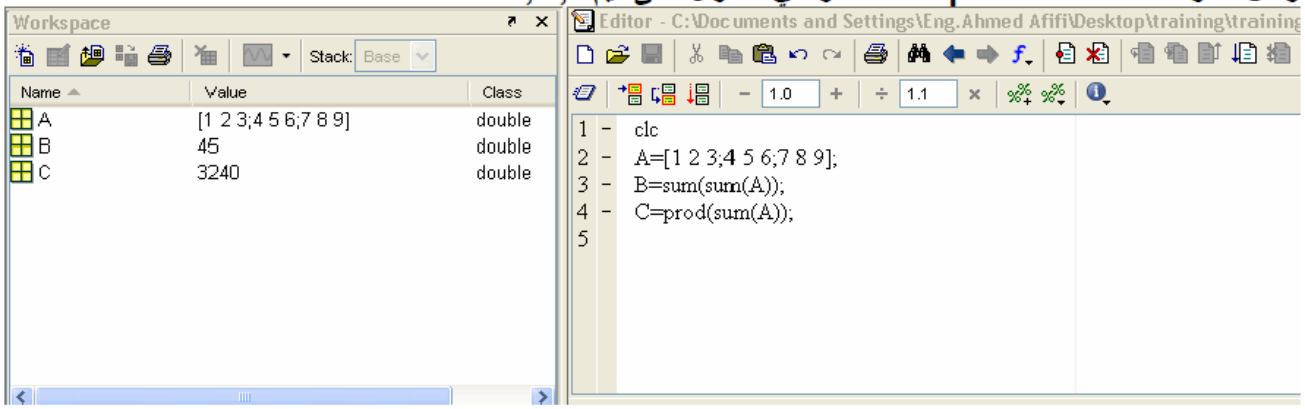


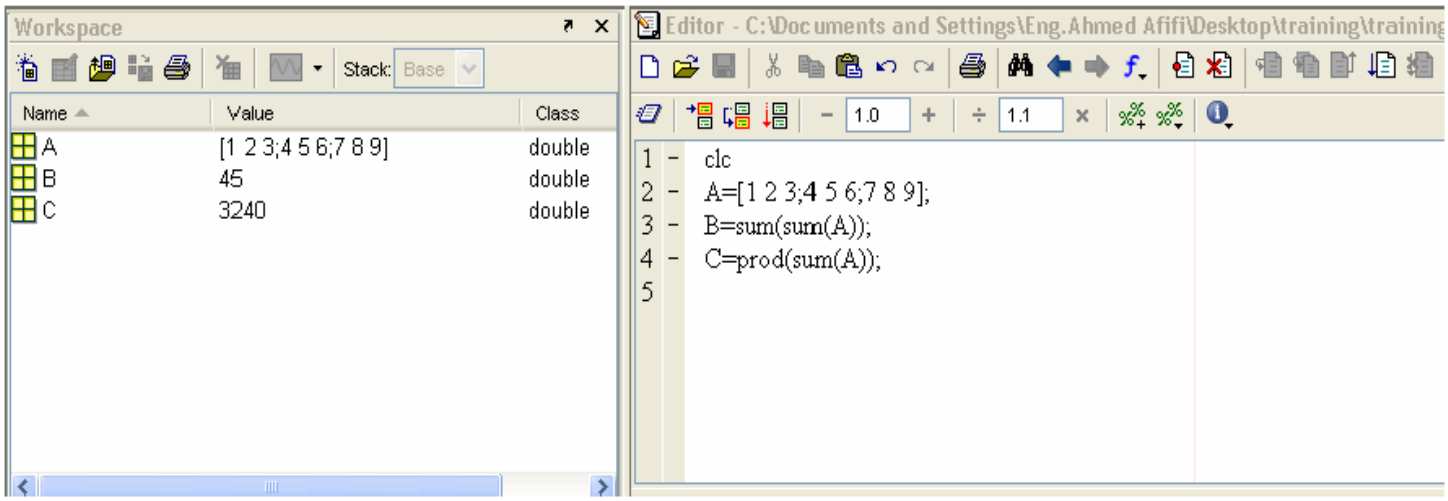
سنقوم الآن بتعديل المثال، وحتى نتأكد أن أمر CLC يعمل، ستختفي القيم من Command Window وتظهر القيم الجديدة



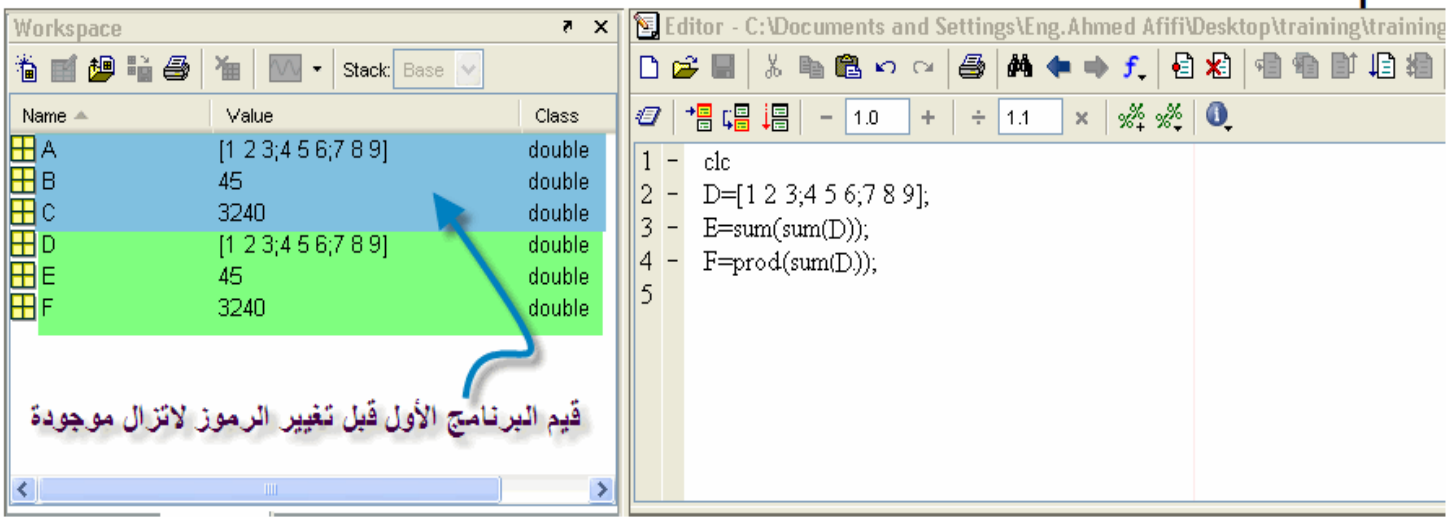
وبهذا نتأكد من أن الأمر CLC يعمل بكفاءة

ولكن دعونا نشاهد نافذة Workspace والتي تحتوى على قيم A,B,C

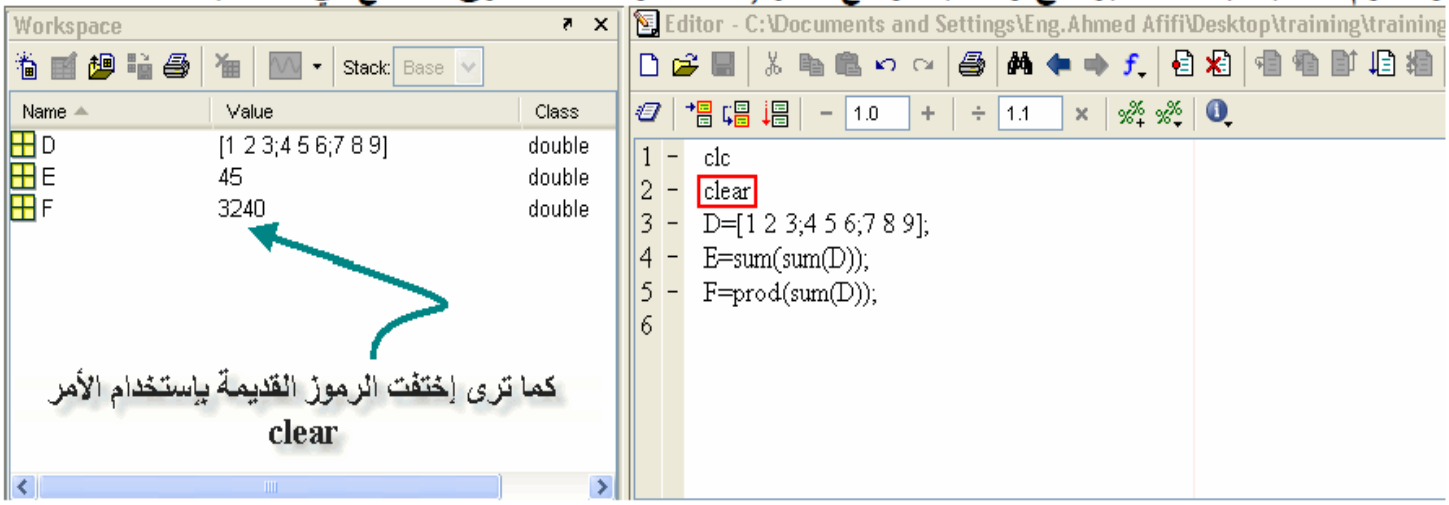




لنقم بتعديل بسيط في البرنامج عن طريق تغيير الرموز فقط من A,B,C إلى D,E,F ومشاهدة النافذة Workspace



ولتلافي هذه المشكلة، يجب وضع أمر Clear بعد الأمر clc بحيث يقوم بمسح أي قيمة سابقة من أي برنامج آخر في Workspace , ويجب تثبيت هذا الأمر أيضاً في جميع البرامج والتي سيتم عملها لاحقاً بإذن الله.
وسنقوم الآن بتنفيذ نفس البرنامج ولكن بعد وضع الأمر ,clear وستلاحظ الفرق الشاسع في الماتلاب الآن



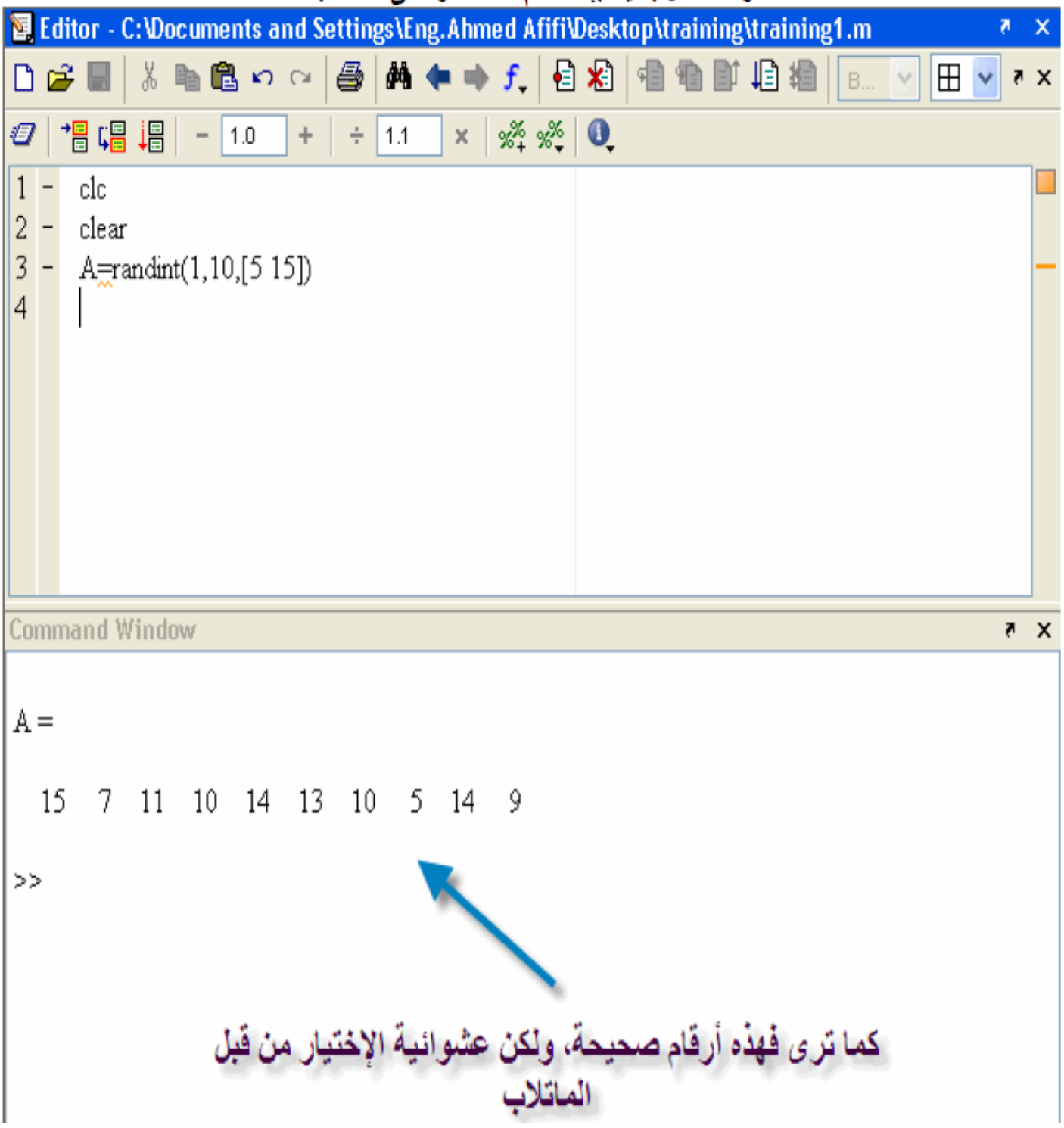
randint

هذا الأمر من ضمن الأوامر والتي تنشأ نظام عشوائي للأرقم، ولكن ليس نظام نظام عدد صحيح وليس على هيئة كسور مثل الأمر السابق، كما أن نظام الأرقام به ليس تزايدياً أو تناقصياً بل عشوائياً

ويأخذ الصورة التالية

randint(number of rows,number of column,[minimum number,maximum number])

وهذا مثال بسيط بإستخدام هذا الأمر على الماتلاب



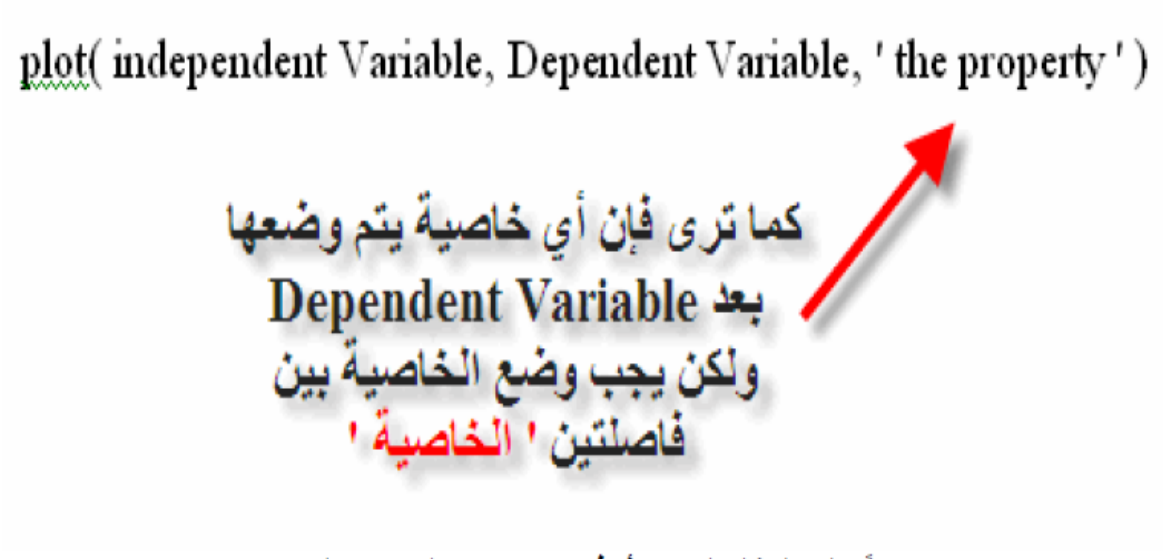
إضافة خصائص إلى الرسومات داخل الماتلاب

في بعض الأحيان يكون من الضروري جداً تغيير بعض الخواص لدى الرسومات التي نحصل عليها مثل تغيير الألوان، وتغيير الرسمة من خطوط متصلة إلى نجوم ونقاك وغيرها، وهذه هي مجموعة الخصائص التي تتم من خلال الماتلاب

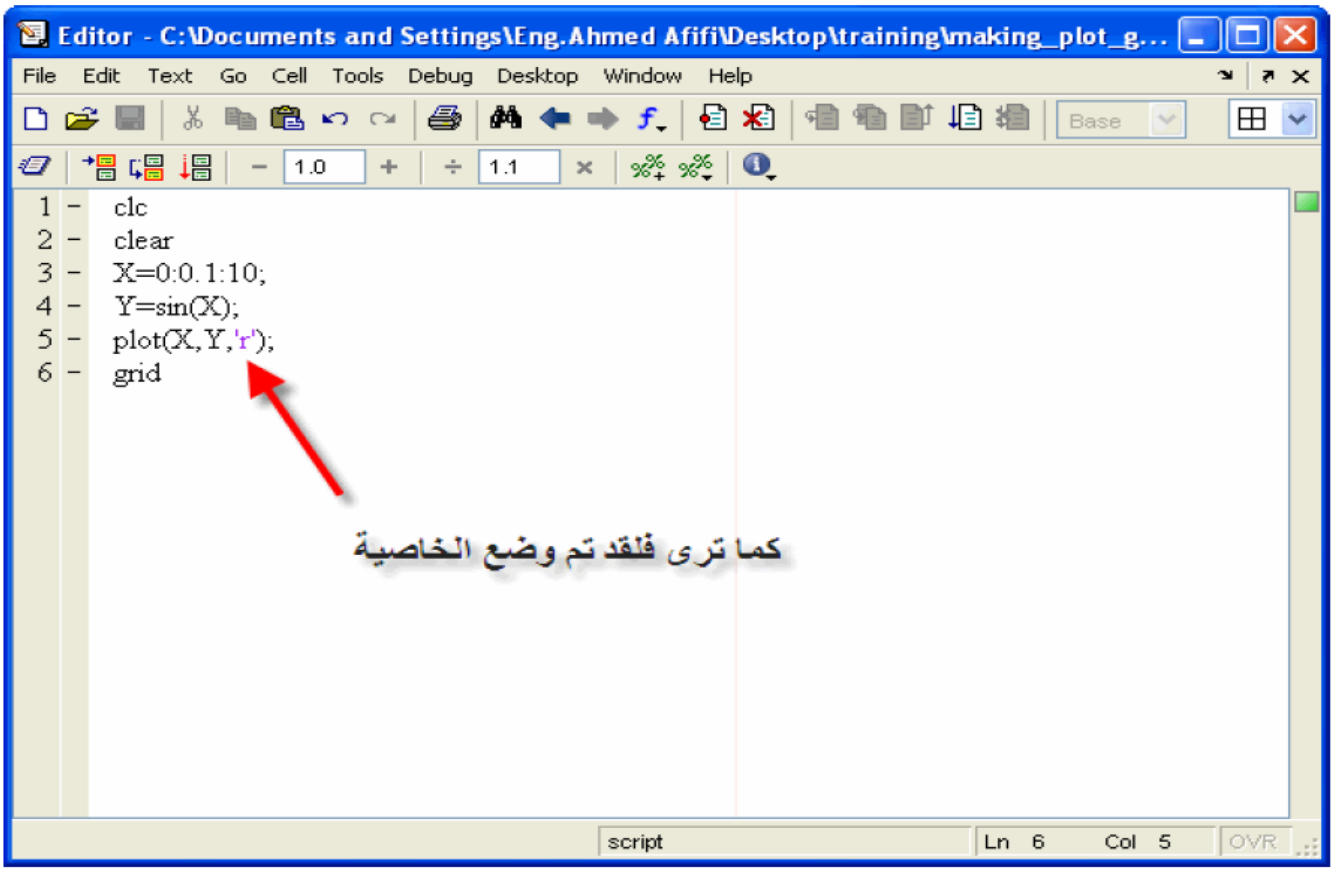
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

فكيف يتم وضع تلك الخصائص داخل الماتلاب، تكون هذه الخصائص متضمنة في الأمر plot حيث تأخذ الصورة التالية

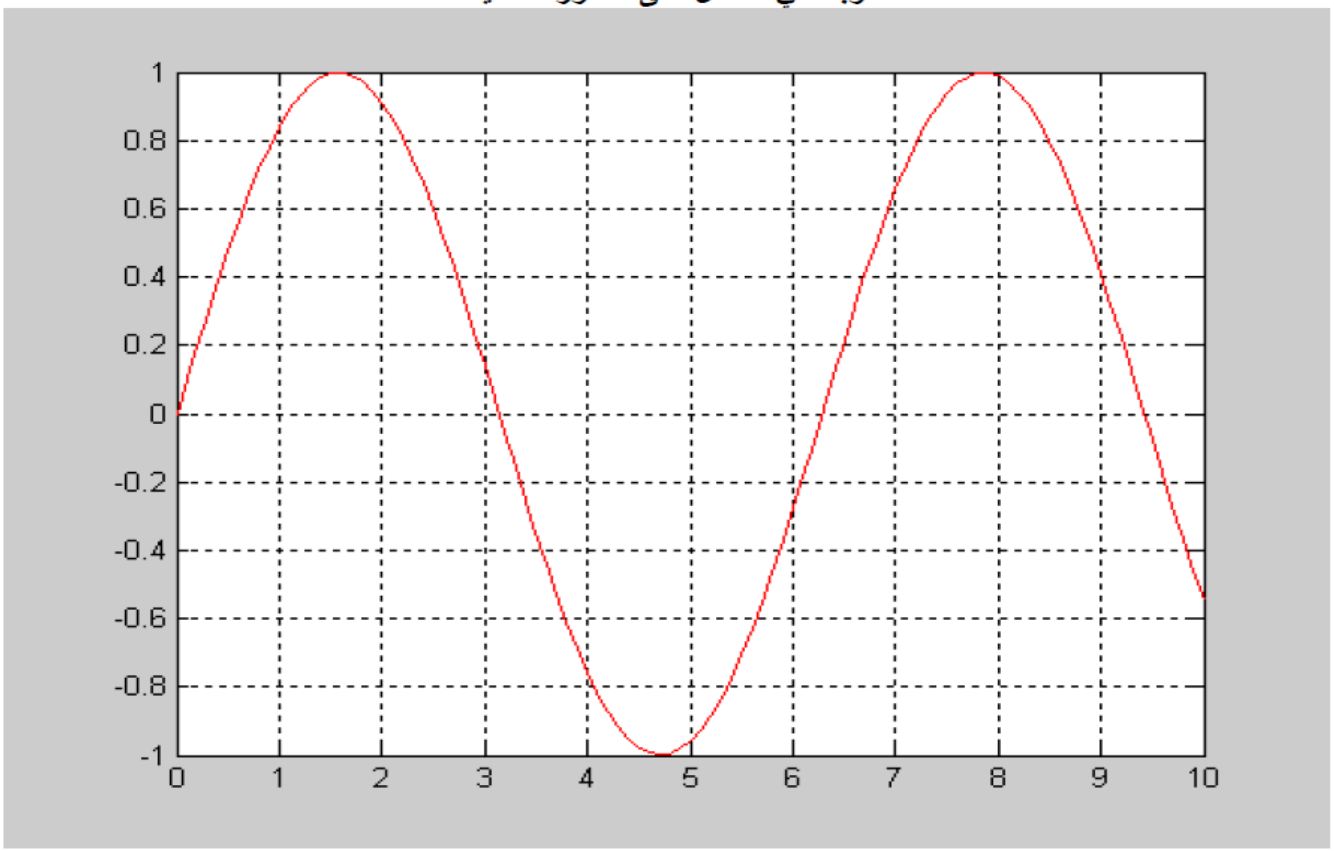
```
plot( independent Variable, Dependent Variable, ' the property ' )
```



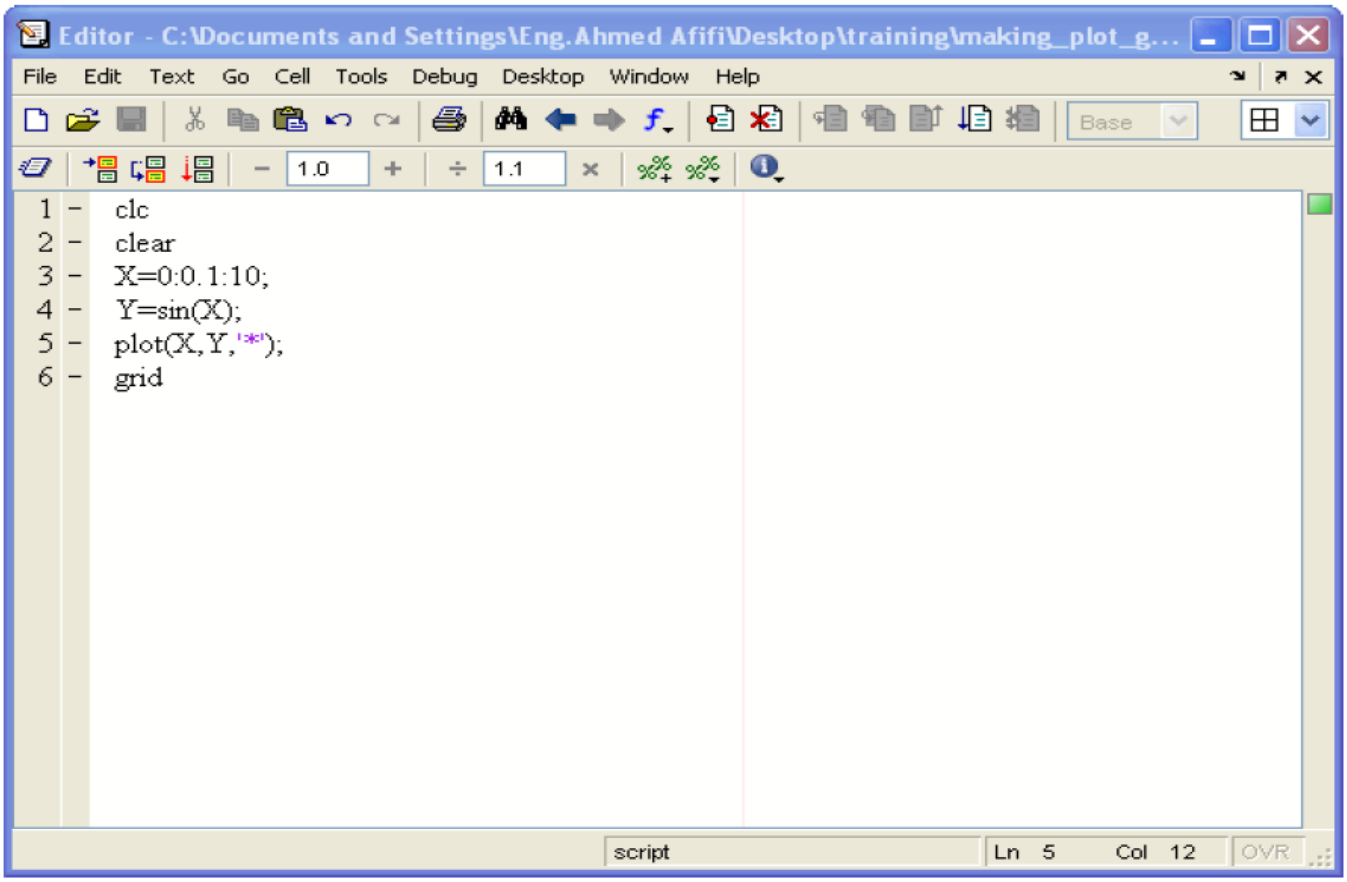
إعتماداً على المثال السابق أخذه سنقوم بتعديل بعض الخصائص
سنقوم مثلاً بتغيير لون الخط إلى الأحمر



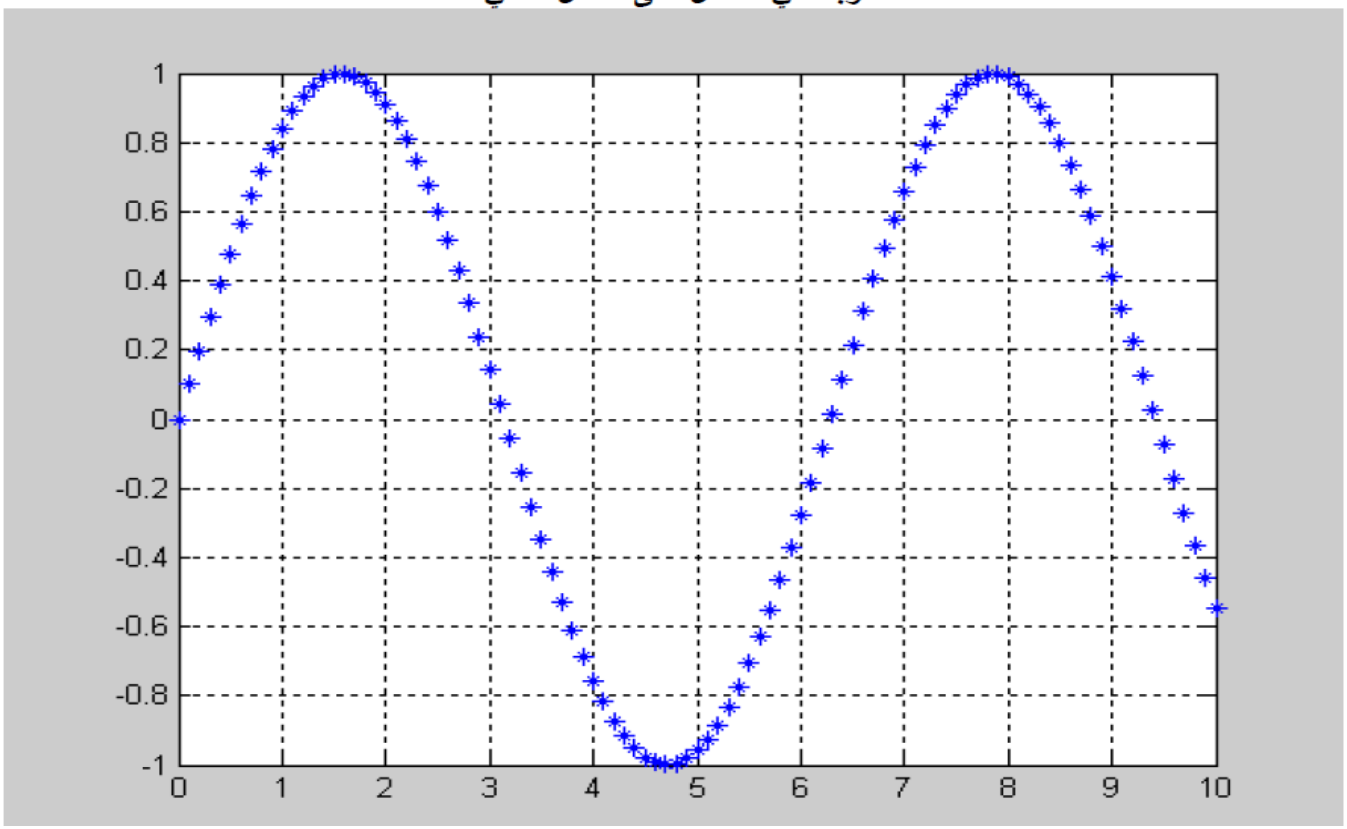
وبالتالي نحصل على الصورة التالية



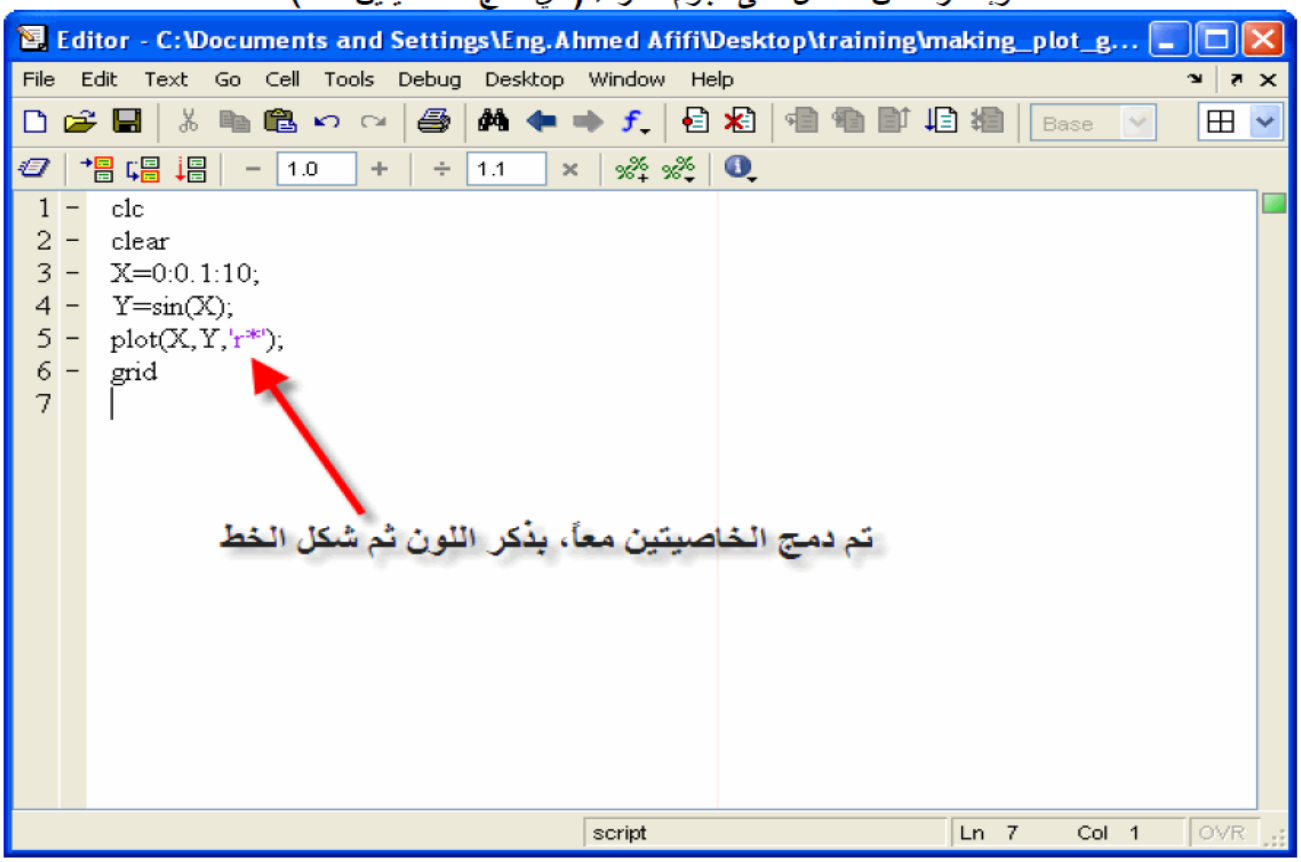
ولنقم الآن بإضافة خاصية جديدة بأن يكون الخط ليس خطاً متصل وإنما عبارة عن نجوم



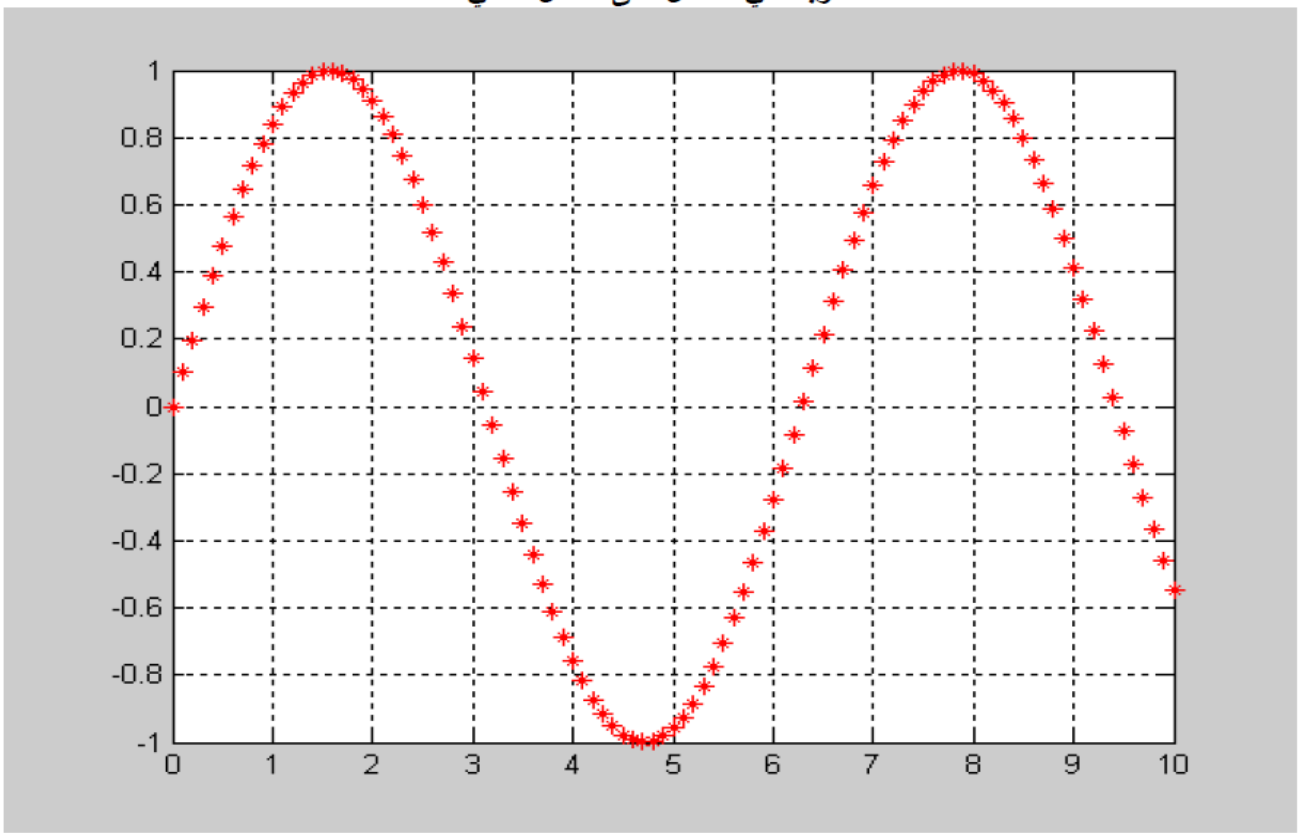
وبالتالي نحصل على الشكل التالي



وإذا أردنا أن نحصل على نجوم حمراء (أي دمج الخاصيتين معاً)



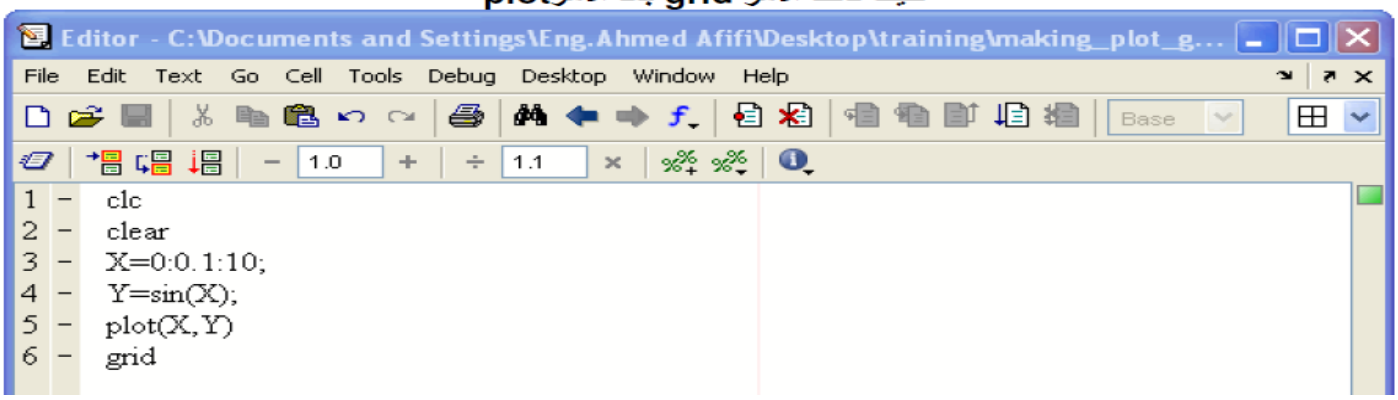
وبالتالي نحصل على الشكل التالي

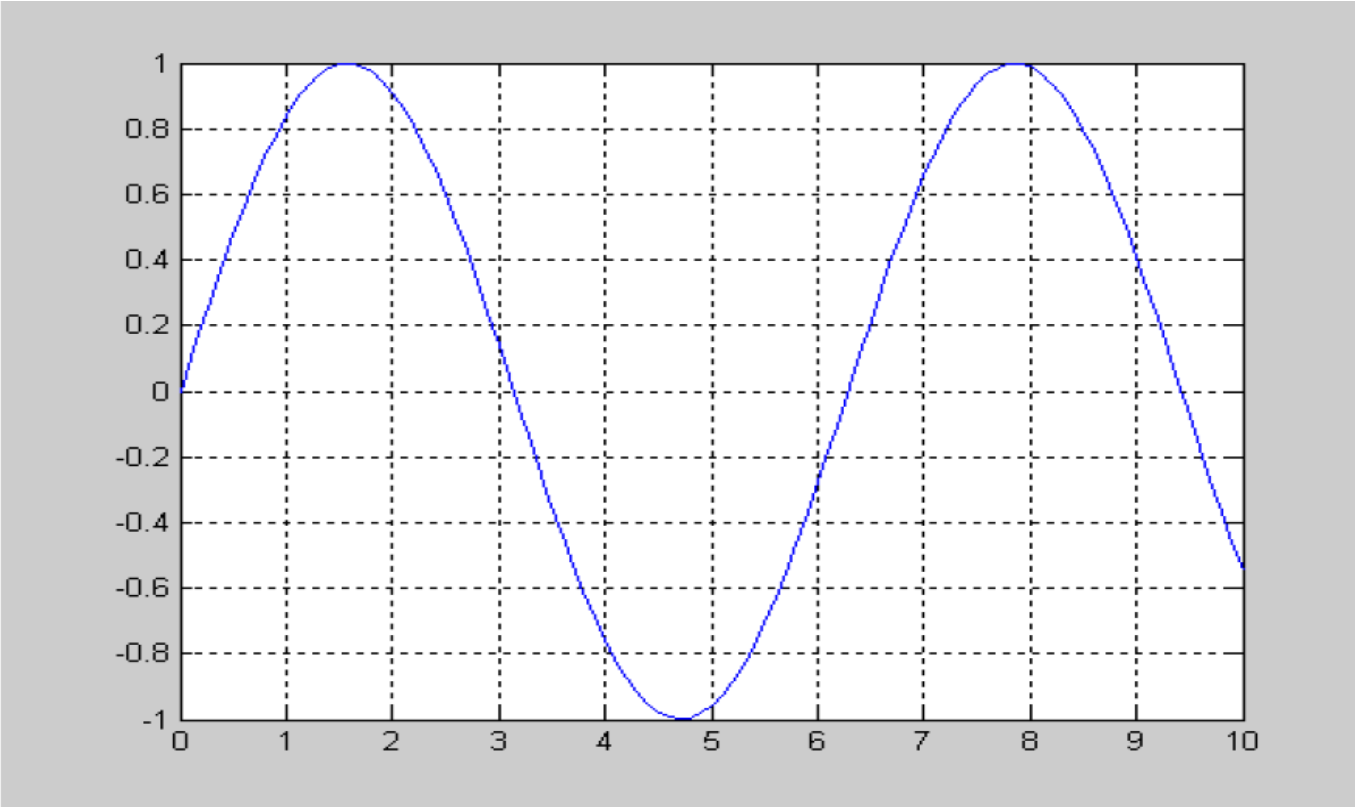


وهنا نكون قد شرحنا خصائص الرسومات داخل الماتلاب

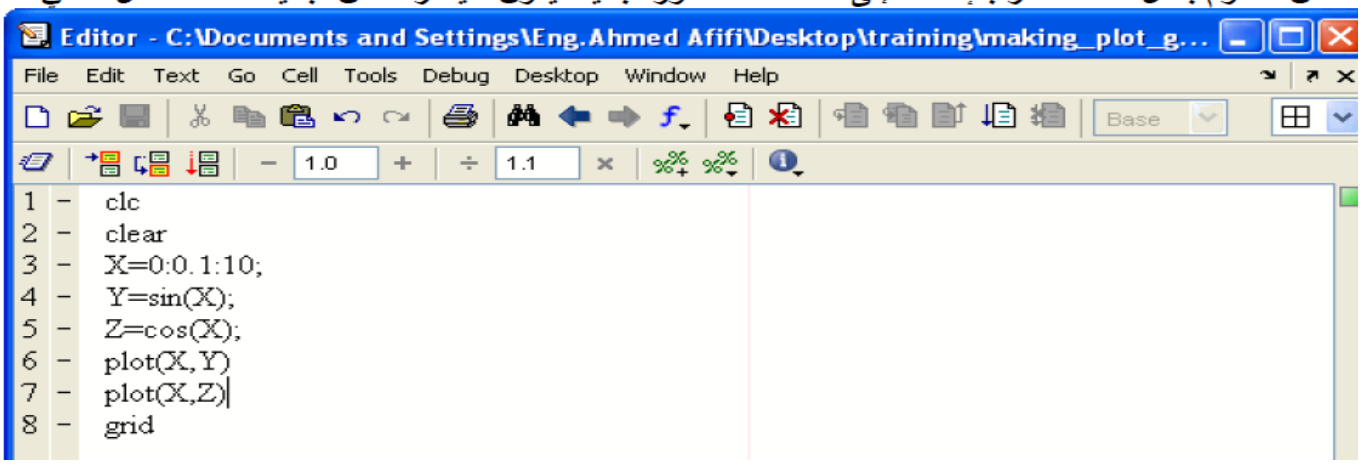
عملية وضع شبكة على الرسم

يقوم الماتلاب بوضع شبكة على الرسم، بحيث يكون من السهل تحديد القيم من على الرسم
حيث تأخذ الأمر grid بعد الأمرplot

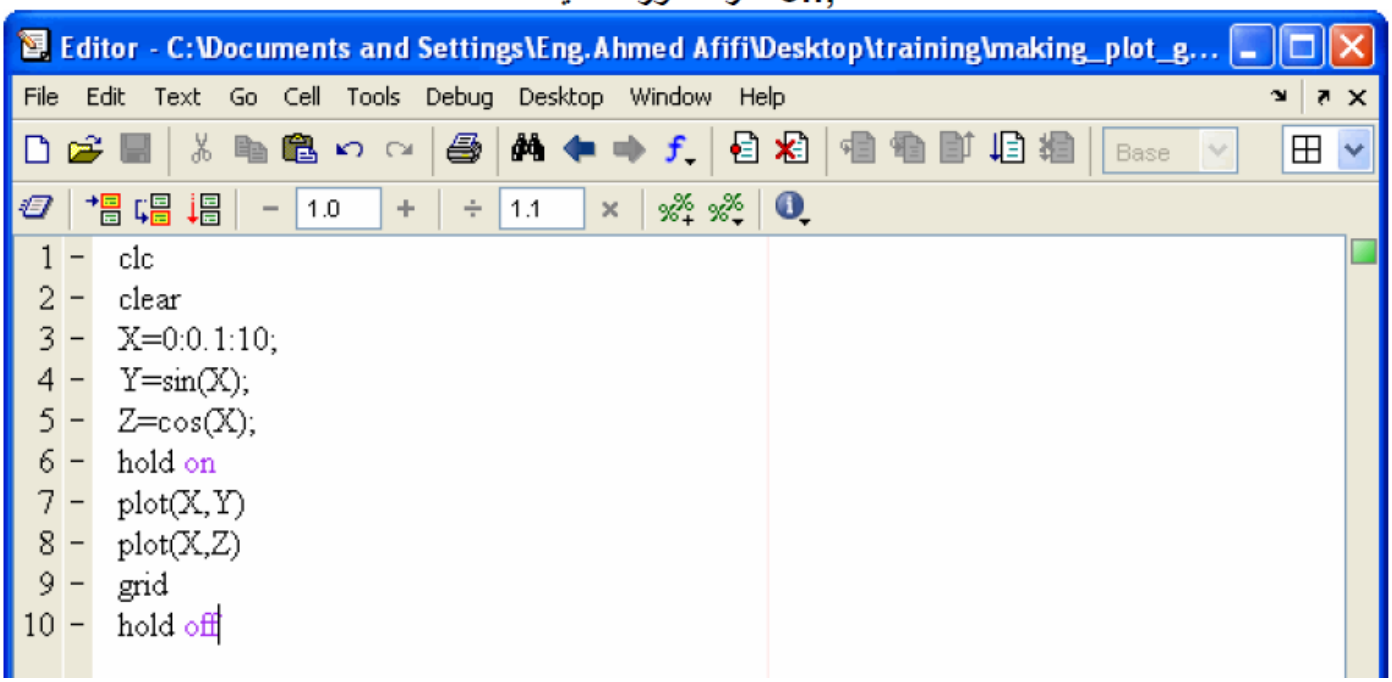


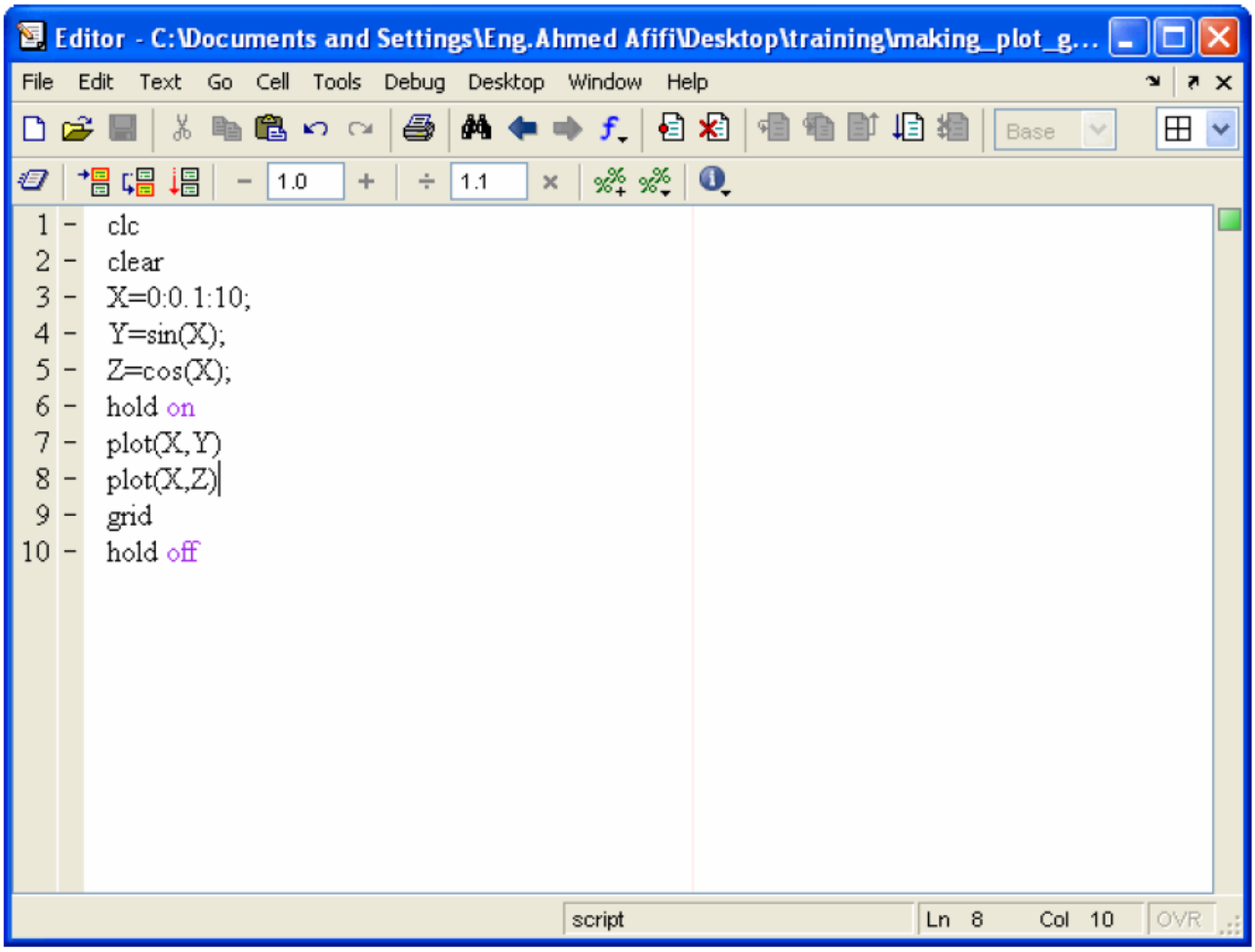


الآن سنقوم بعمل معادلة أخر بالإضافة إلى المعادلة المذكورة بحيث يكون لدينا رسمتان، بحيث تأخذ الشكل التالي



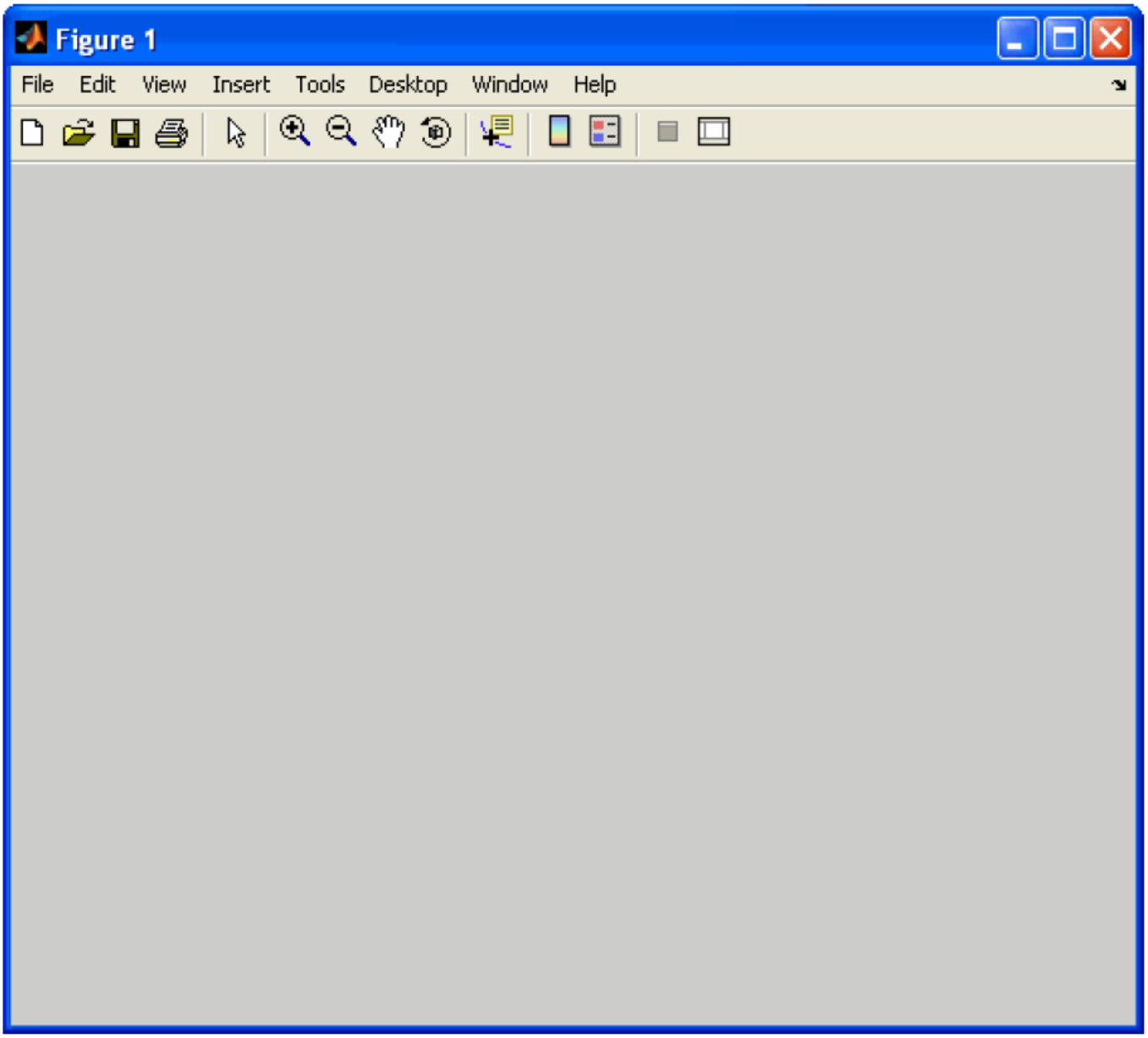
ولكن عند تشغيل البرنامج، سيقوم الماتلاب بإظهار الرسمة الأخيرة فقط، فكيف يتم إظهار الرسمتين، يتم ذلك بإستخدام الأمر Hold on قبل الأمر plot لكي يتم وضع الرسمتين في نافذة واحدة، وفي نهاية الأمر يتم وضع الأمر hold off, أنظر الصورة التالية



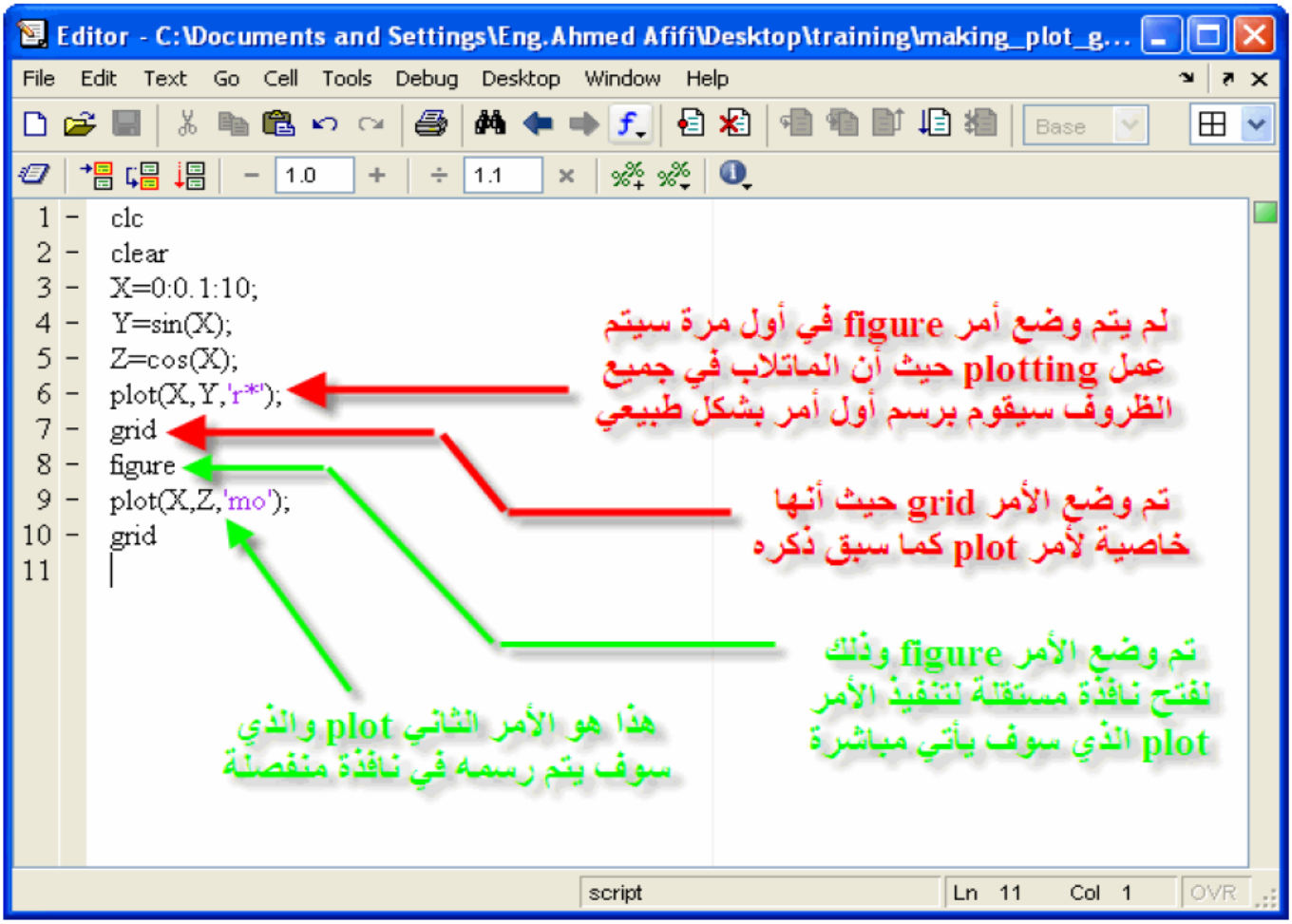


علمية وضع الرسومات في نوافذ منفصلة

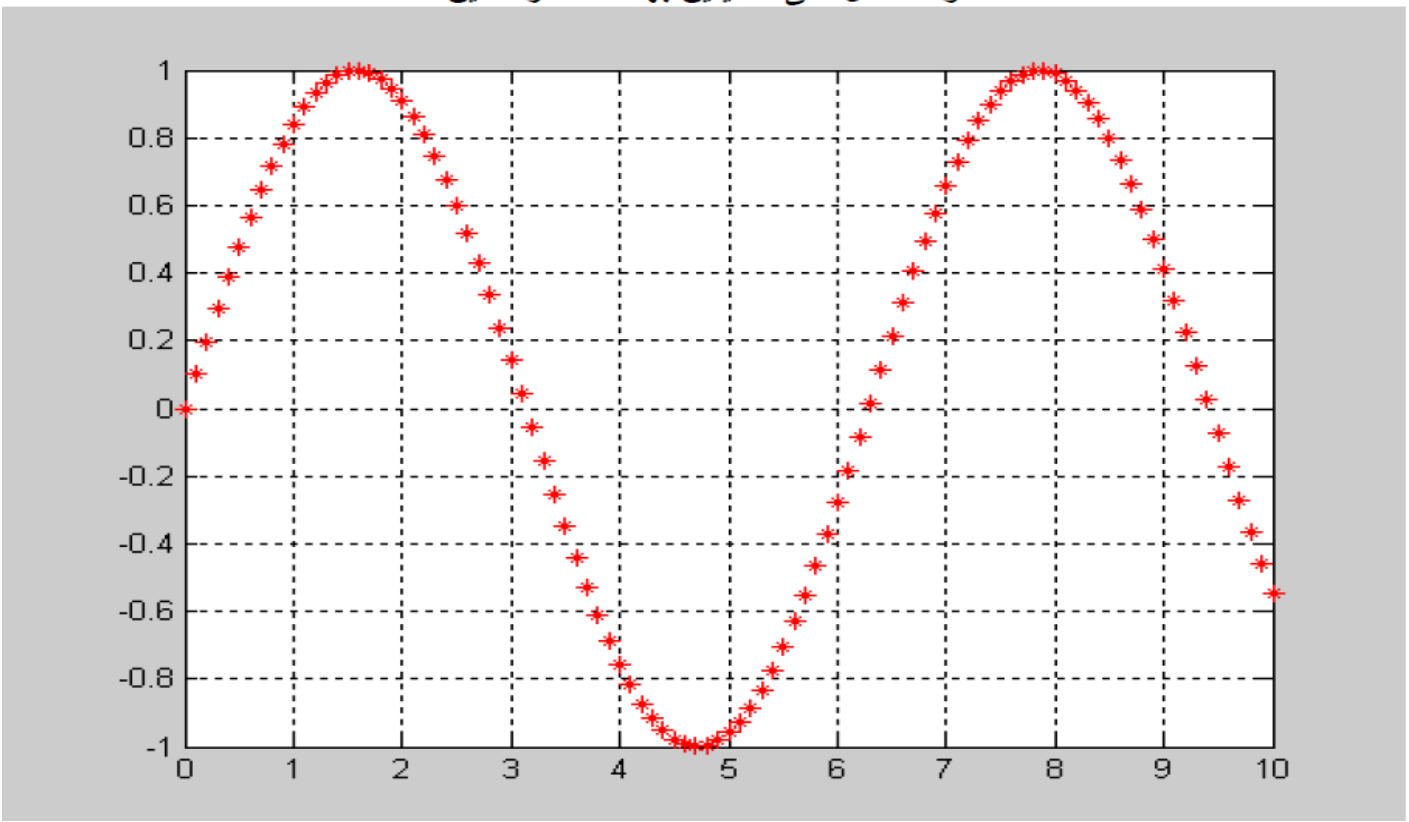
سنقوم الآن بدلاً من وضع الرسومات في نفس النافذة سنقوم بوضعها في نوافذ مختلفة
وعلى نحتاج إلى الأمر **figure** والذي يقوم بفتح نافذة فارغة إذا تم وضعه منفصلاً، جرب ذلك في نافذة الأوامر
ستلاحظ ان الماتلاب قام بإظهار نافذة رمادية اللون فارغة شاهد الصورة التالية

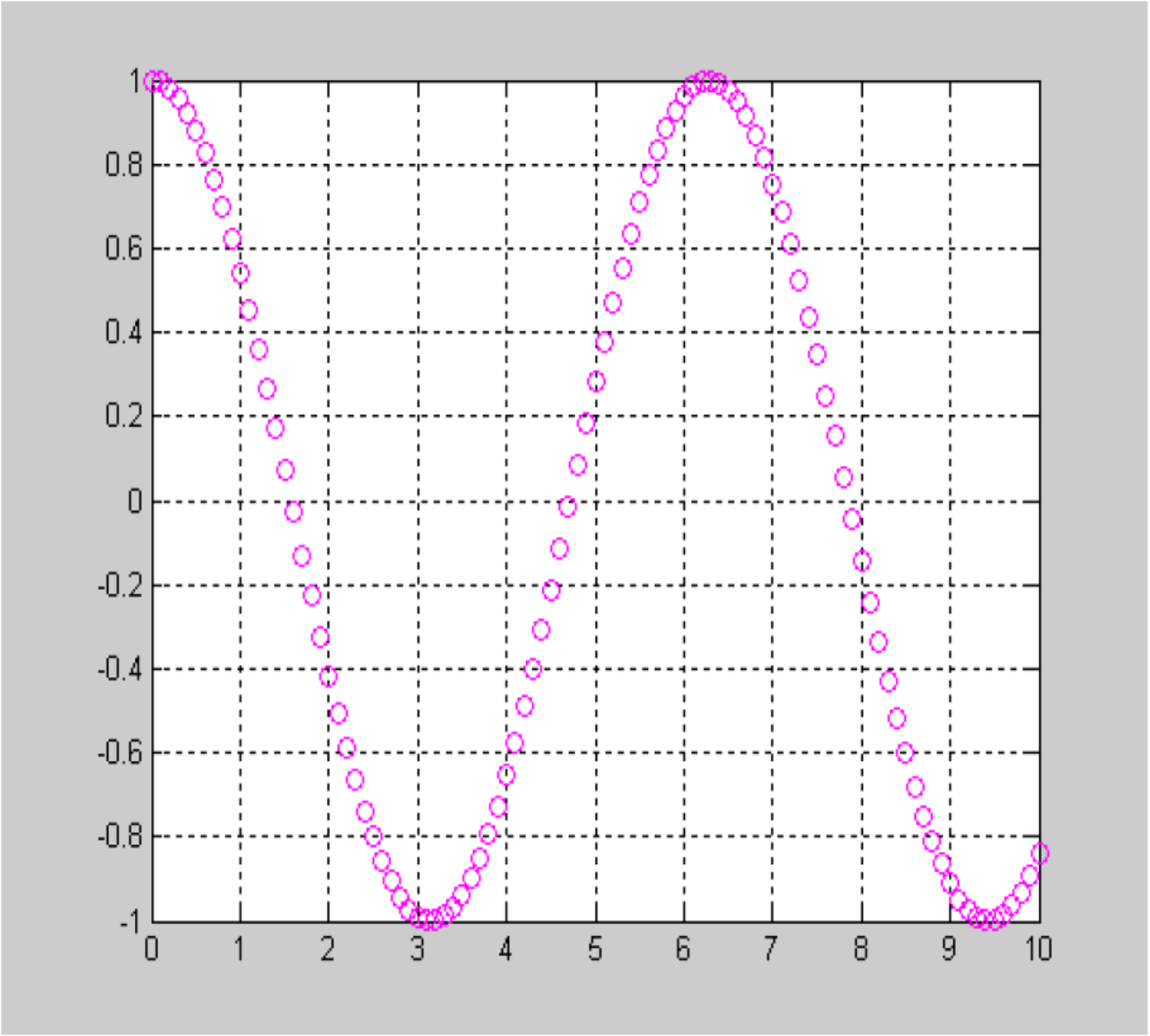


حيث وجود تلك النافذة يعني انه سيتم تنفيذ أمر الرسم **plot** الذي بعد أمر **figure** علماً أنه بعد كل أمر **figure** يتم وضع الخصائص التي تختص بهذه الرسمة مثل أمر **grid** الذي سبق شرحه.
وهذا مثال بسيط على ذلك



وستحصل على نافذيتين بهما كلتا الرسمتين

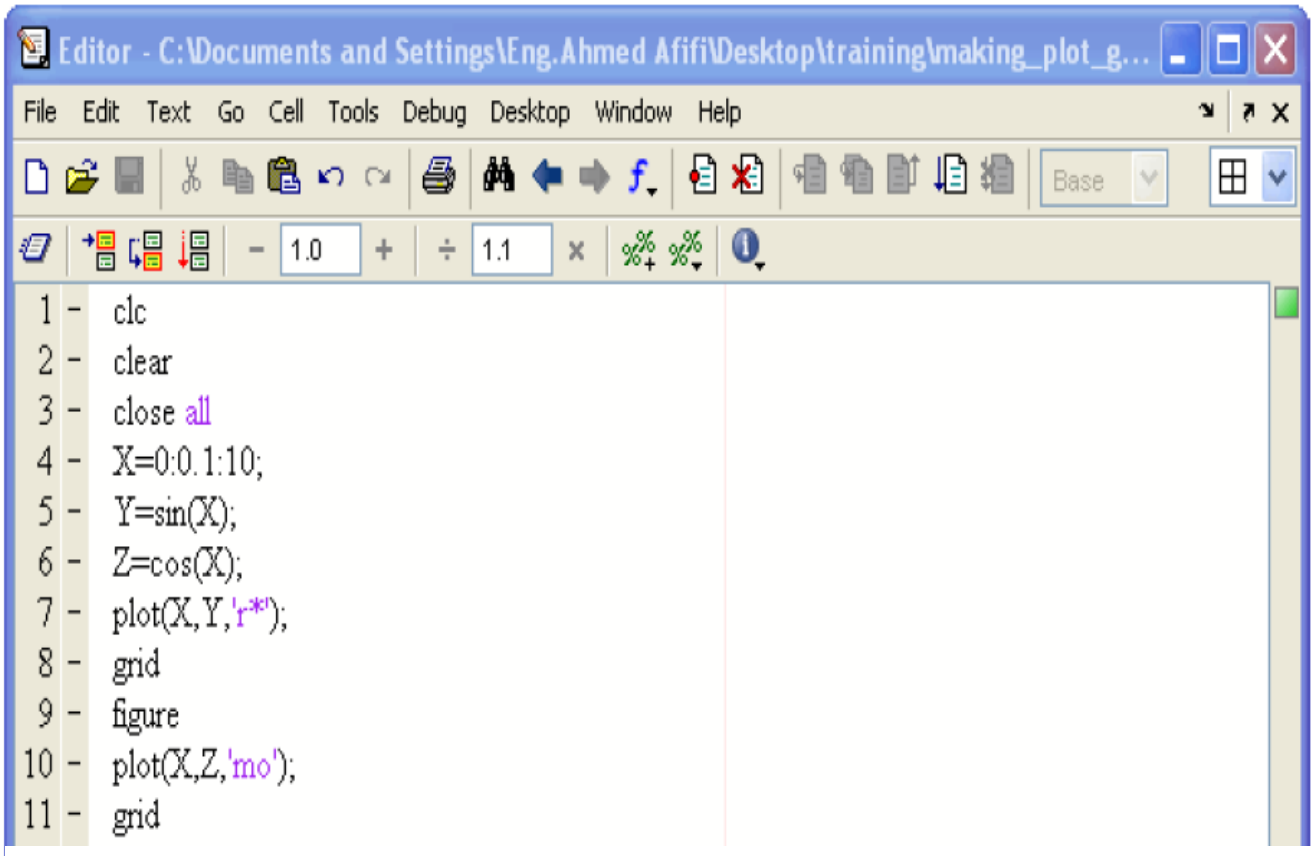


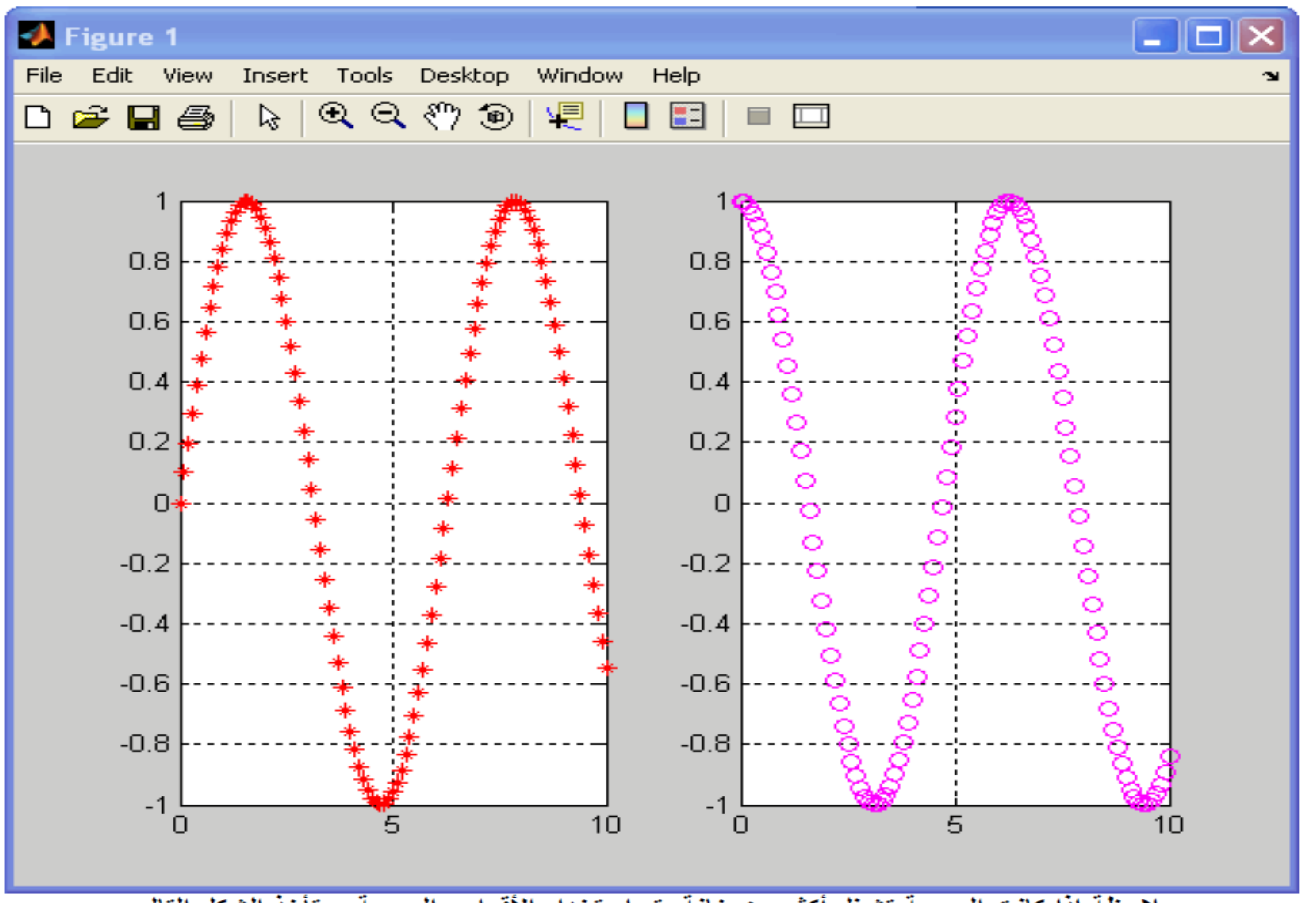


والآن قم بتشغيل البرنامج مرة أخرى، ستلاحظ أن عدد النوافذ قد زاد نافذة واحدة ,فكيف حدث هذا؟
يقوم الماتلاب برسم أول دالة على النافذة الأخيرة التي تم رسم الدالة الثانية بها، ثم يقوم برسم الدالة الثانية في نافذة جديدة بسبب وجود الأمر figure ولحل هذه المشكلة قم بإستخدام الأمر close all بعد الأمر clear بحيث يتم إغلاق أي نوافذ كانت مفتوحة قبل ذلك عند تشغيل البرنامج كل مرة وبالتالي سيكون هنالك ثلاثة أوامر لابد من إستخدامها في كل مرة يتم عمل أي برنامج وهم

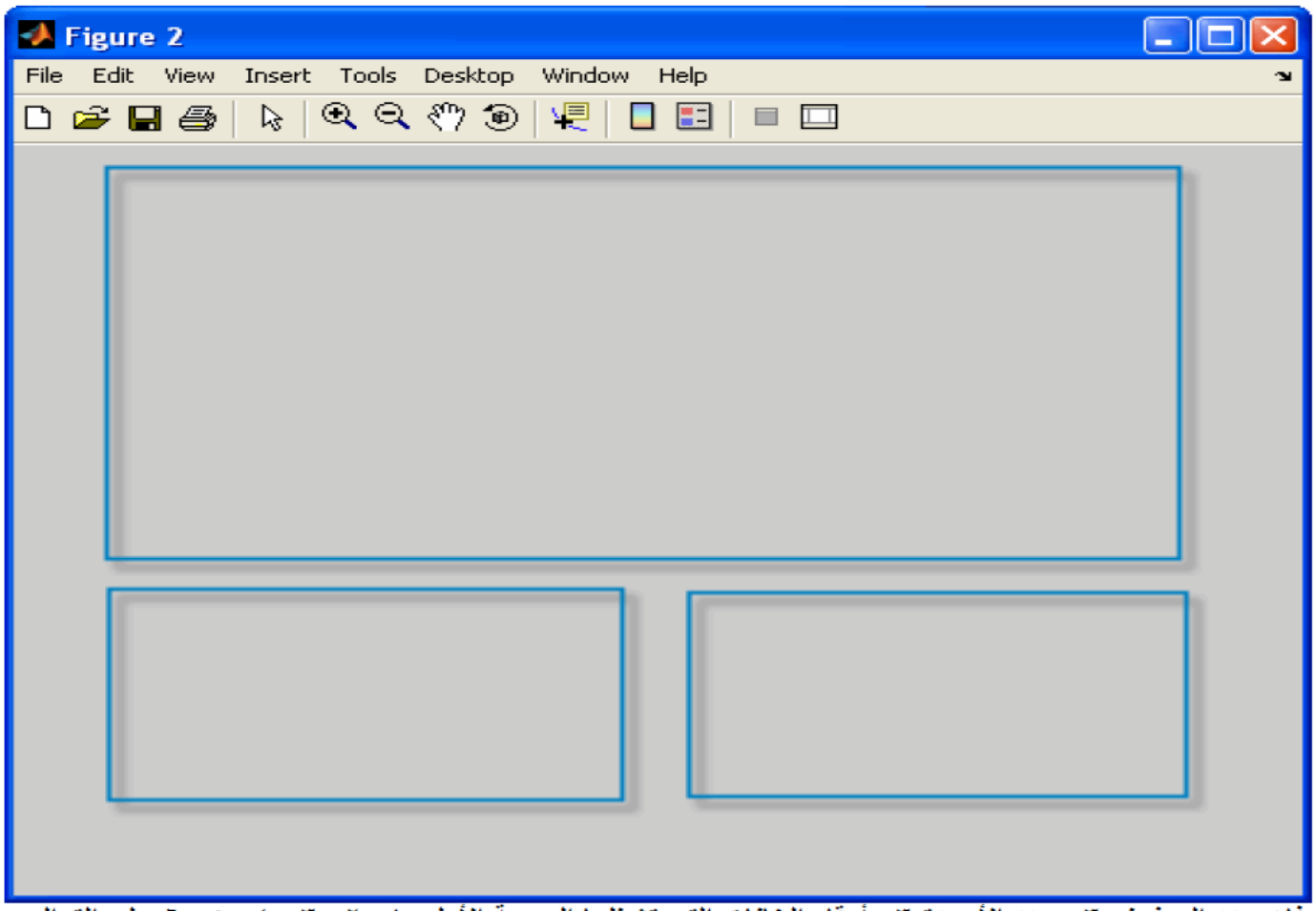
clc
clear
close all

وهذا هو المثال الذي تم عمله منذ قليل بعد التعديل





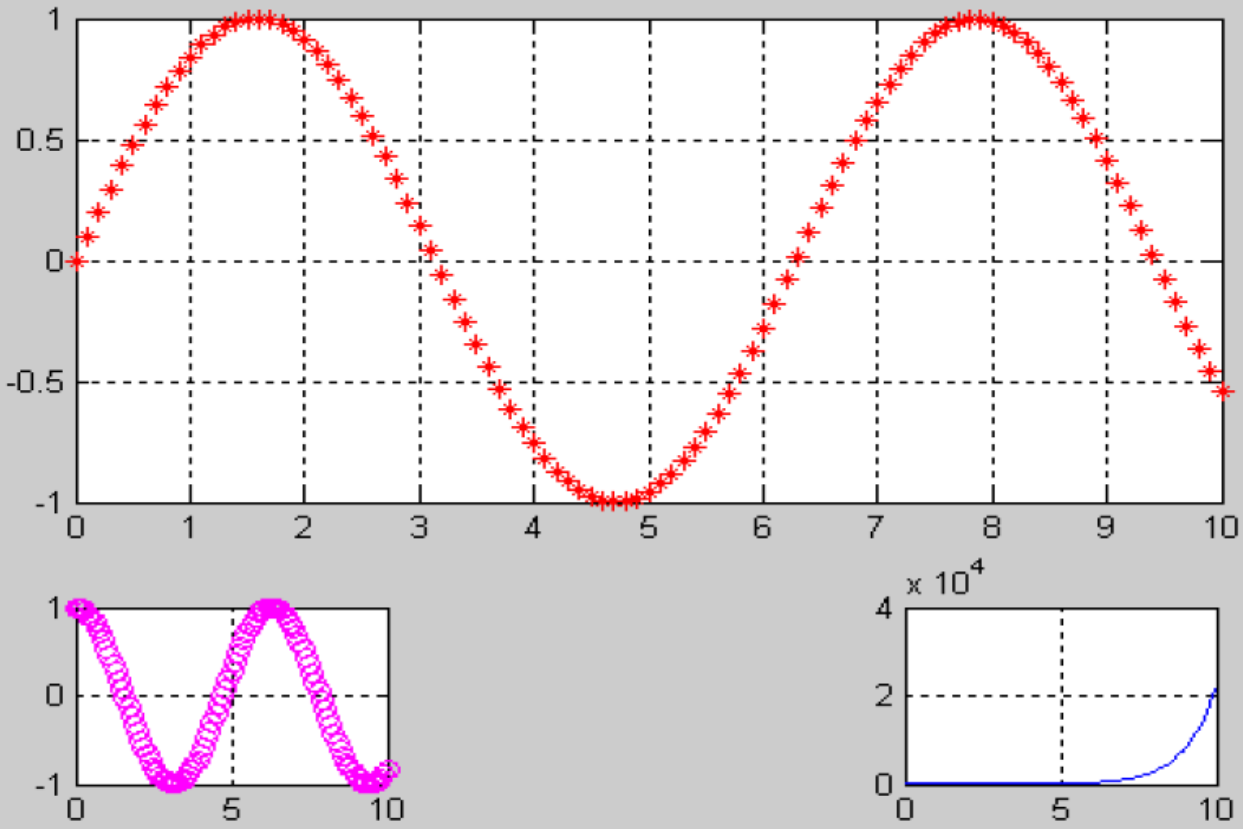
ملاحظة إذا كانت الرسمة تشغل أكثر من خانة يتم إستخدام الأقواس المربعة، وتأخذ الشكل التالي
[أرقام جميع الخانات التي تشغلها الرسمة]
وسنقوم بإعطاء مثال
نريد أن يكون الشكل الخارج على شكل الصورة التالية



فإن عدد الصفوف ٣ وعدد الأعمدة ٣ وأرقام الخانات التي تشغلها الرسمة الأولى ١ و٢ و٣ و٤ و٥ و٦ على التوالي، وأرقام الخانات التي تشغل الرسمة الثانية ٧ وأرقام الخانات التي تشغل الرسمة الثالثة هي ٩
والبرنامج يكون بالشكل التالي

```
clc
clear
close all
X=0:0.1:10;
Y=sin(X);
Z=cos(X);
V=exp(X);
subplot(3,3,[1 2 3 4 5 6])
plot(X,Y,'r*');
grid
subplot(3,3,7)
plot(X,Z,'mo');
grid
subplot(3,3,9)
plot(X,V);
grid
```

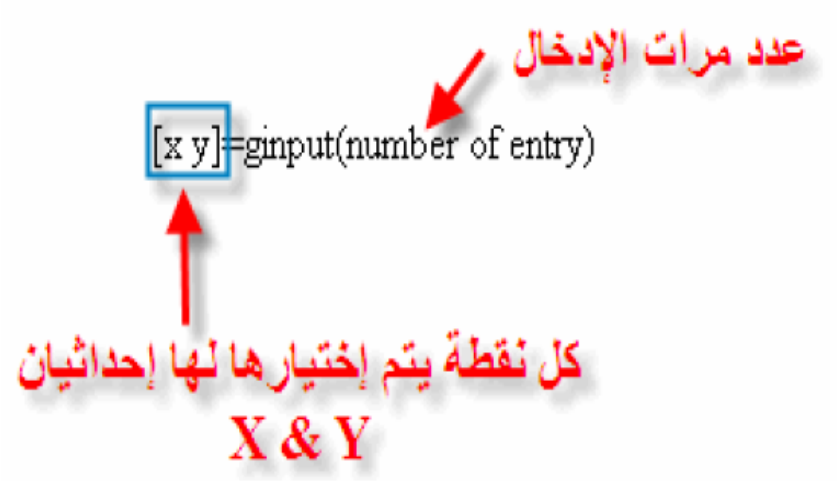
وستكون النتيجة كالتالي



كيفية إدخال النقاط من خلال الماوس

تعلمنا أنه يمكننا إدخال القيم بإستخدام المتجهات أو المصفوفات، ولكن يوفر الماتلاب قدرة في إدخال النقاط من خلال الرسم بإستخدام الماوس، ونظراً لأننا نقوم بإختيار النقاط من على الرسم فهذا يعني أن النقاط التي يتم إختيارها يتم تمثيلها في قيمة في محور السينات وقيمة في محور الصادات، ويتم وضع قيم محاور السينات والصادات في صورة متجه.

يستخدم الأمر **ginput** في عملية إدخال النقاط بإستخدام الماوس، ويتم كتابة ذلك الأمر في الصورة التالية



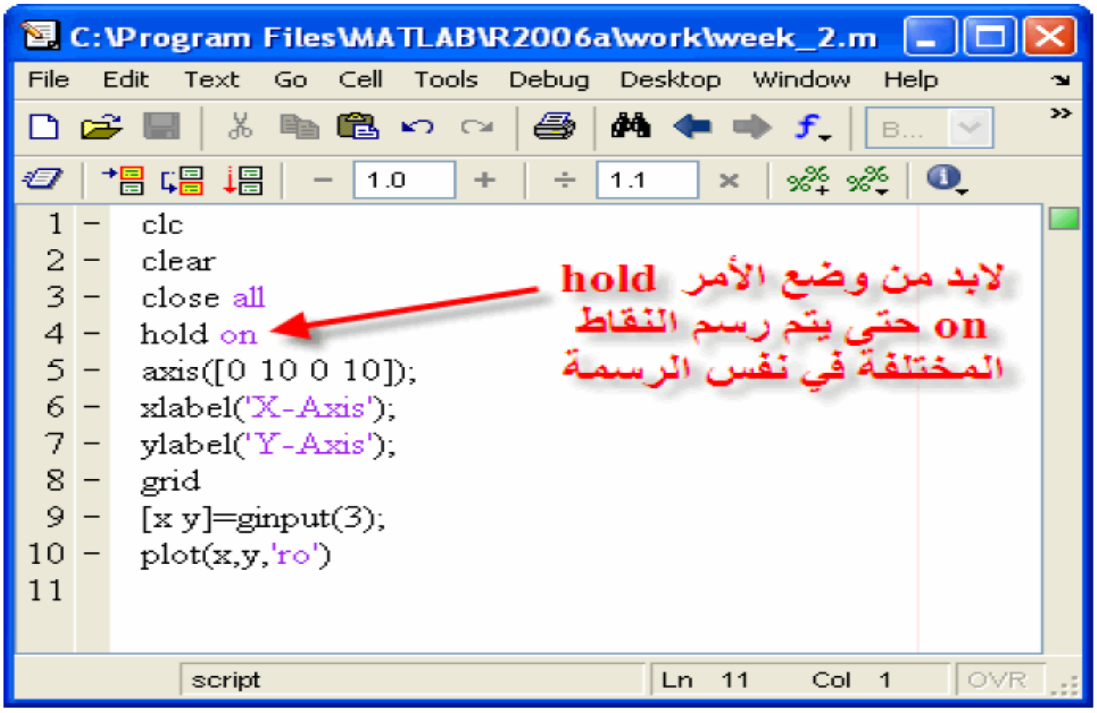
أما إذا أردنا إدخال عدد لا نهائي من النقاط يمكن ذلك بعدم ذكر عدد نقاط الإدخال ,كما في الشكل التالي

وبعد الإنهاء من إدخال النقاط كل ما عليك هو الضغط على مفتاح **Enter** في لوحة المفاتيح.

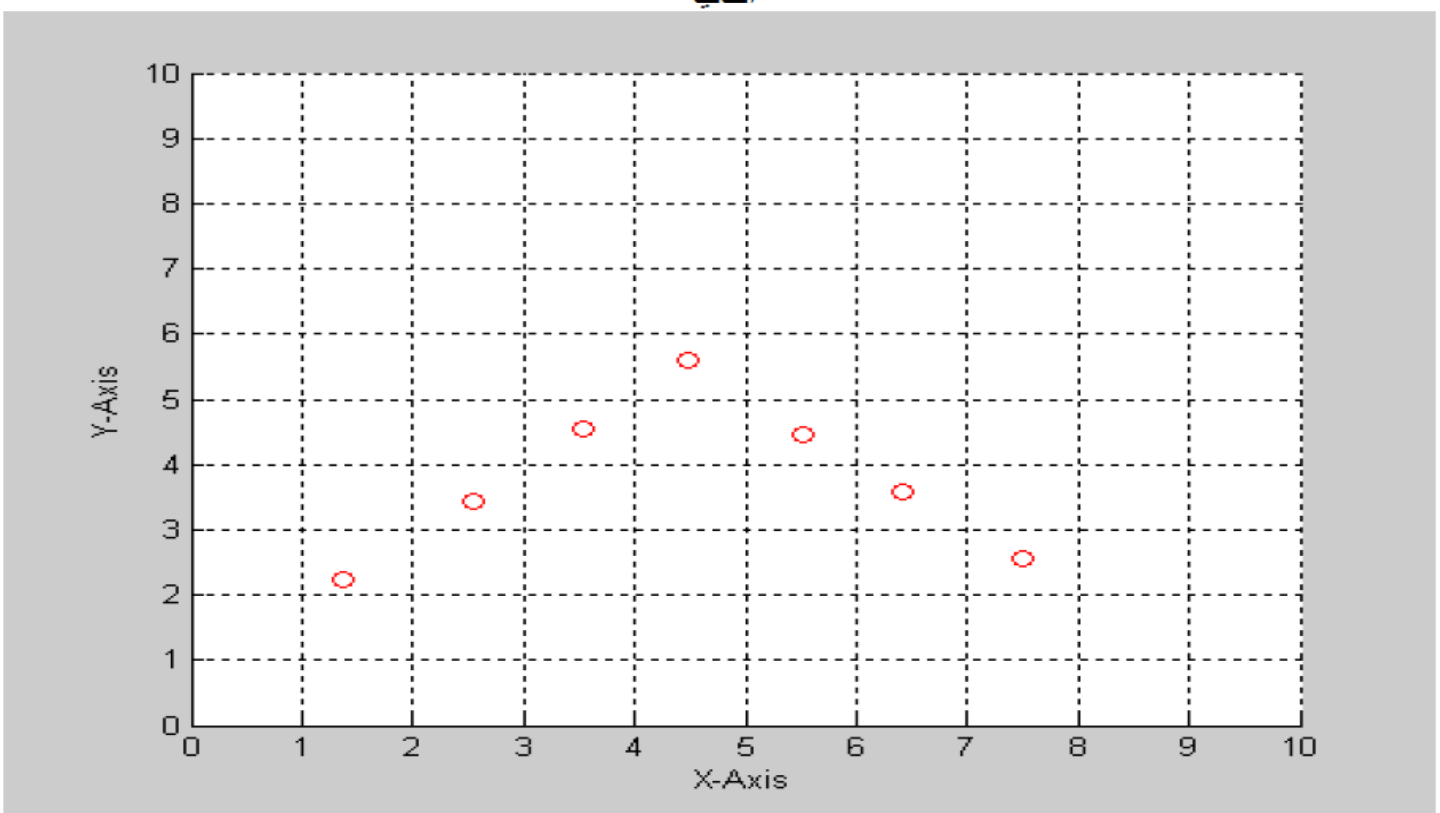
مثال تطبيقي

سنقوم بفتح نافذة للرسم بها شبكة، وأقل قيمة لمحور السينات هي صفر وأكبر قيمة لمحور السينات هي ١٠ وكذلك بالنسبة لمحور الصادات، ثم إدخال عدد كبير من النقاط على الرسم بإستخدام الأمر **ginput**, وهذه النقاط يتم طباعتها على شكل دوائر حمراء.

ويتم كتابة الأوامر بالشكل التالي



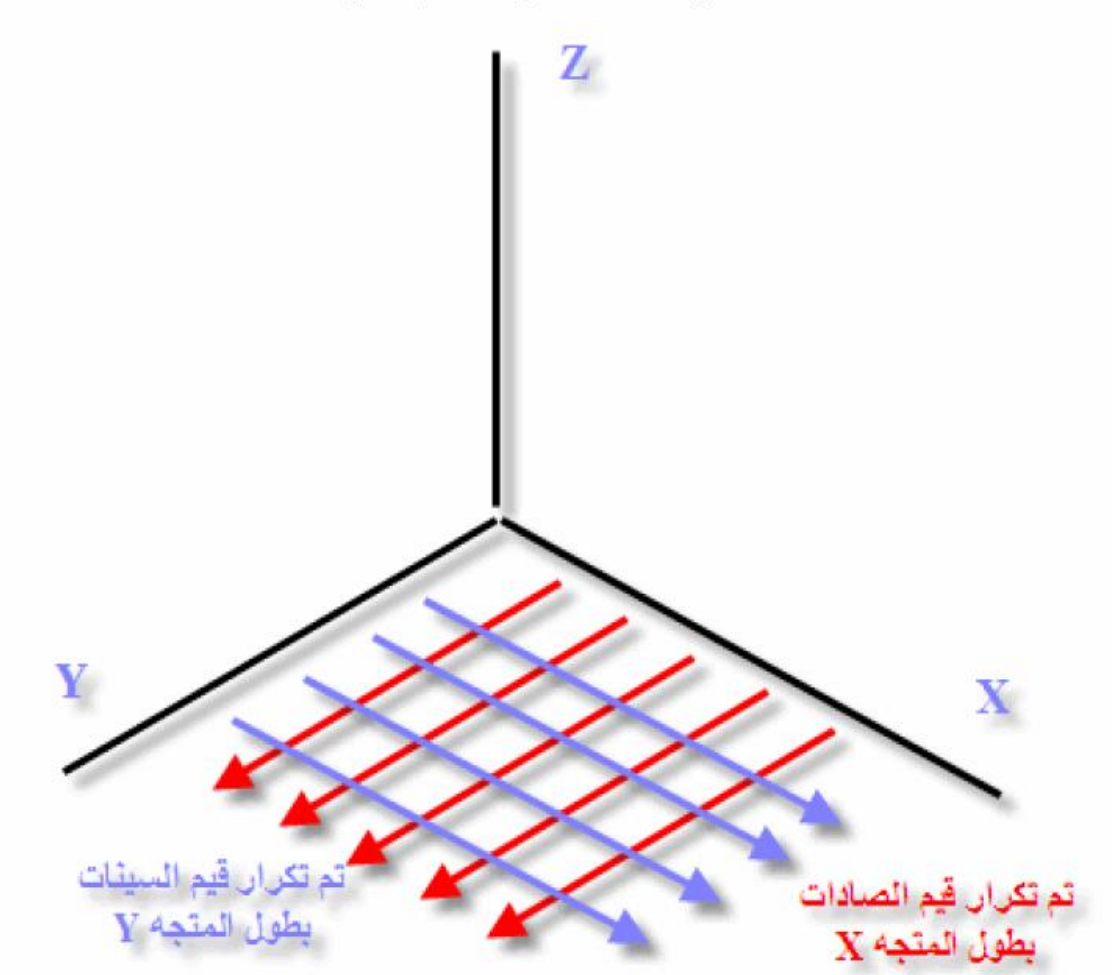
وستظهر لك نافذة لإدخال النقاط ، وبعد إتمام عملية الإدخال إضغط على **Enter** لإتمام الإدخال وستظهر لك النافذة التالية



وبالتالي نكون قد أتممنا شرح هذه الجزئية بنجاح، وسيتم التطرق في دورة البرمجة بإستخدام الماتلاب إلى كيفية إظهار النقاط بمجرد الضغط عليها.

الرسم ثلاثي الأبعاد

كما تعلمنا أن الرسم ثلاثي الأبعاد يعتمد على ثلاثة محاور لرسمها، محور X , Y & Z وأن كلاً من X & Y يمثلان المستوى الأفقي، وأن المحور Z يمثل الإرتفاع، ولكن تلك القيم هي قيم النقاط الموجودة المحاور، ولكن حتى يتم رسم أي نقطة في المستوى الأفقي يجب أن نقوم تعريف ذلك للماتلاب وذلك بإستخدام الأمر meshgrid حيث يقوم الماتلاب بإنتاج مصفوفة يتم تكرار قيم محور السينات X-Axis بنفس طول محور الصادات Y-Axis, كما يقوم بتكرار قيم محور الصادات Y-Axis بنفس طول قيم السينات X-Axis, وبهذا تكون المصفوفة المتكونة هي المستوى الأفقي كما هو واضح بالرسم التالي



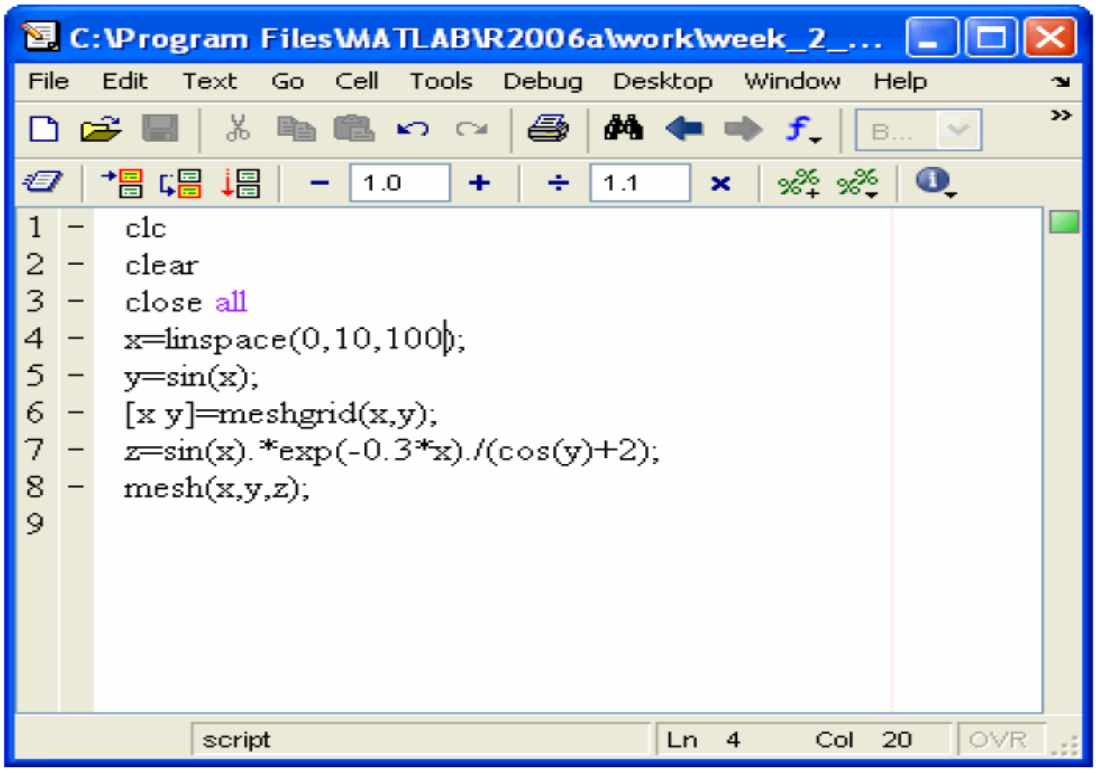
علما أن الأمر meshgrid يأخذ الصورة التالية في كتابته

[x y]=meshgrid(x,y)

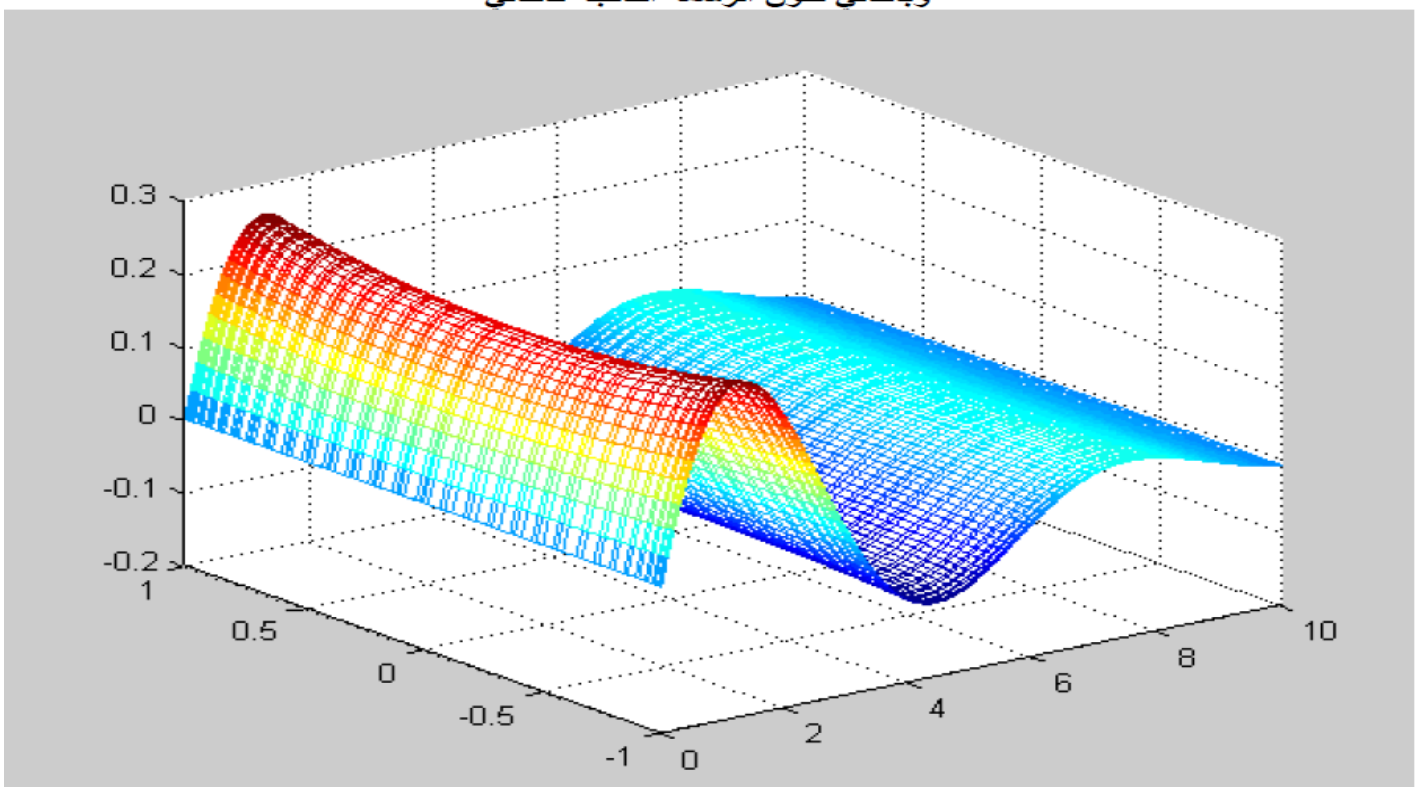
وبعد إستخدام الأمر meshgrid يتم إستخدام الأمر mesh والذي يستخدم كبديل الأمر plot ولكن في الرسم ثلاثي الأبعاد

مثال تطبيقي

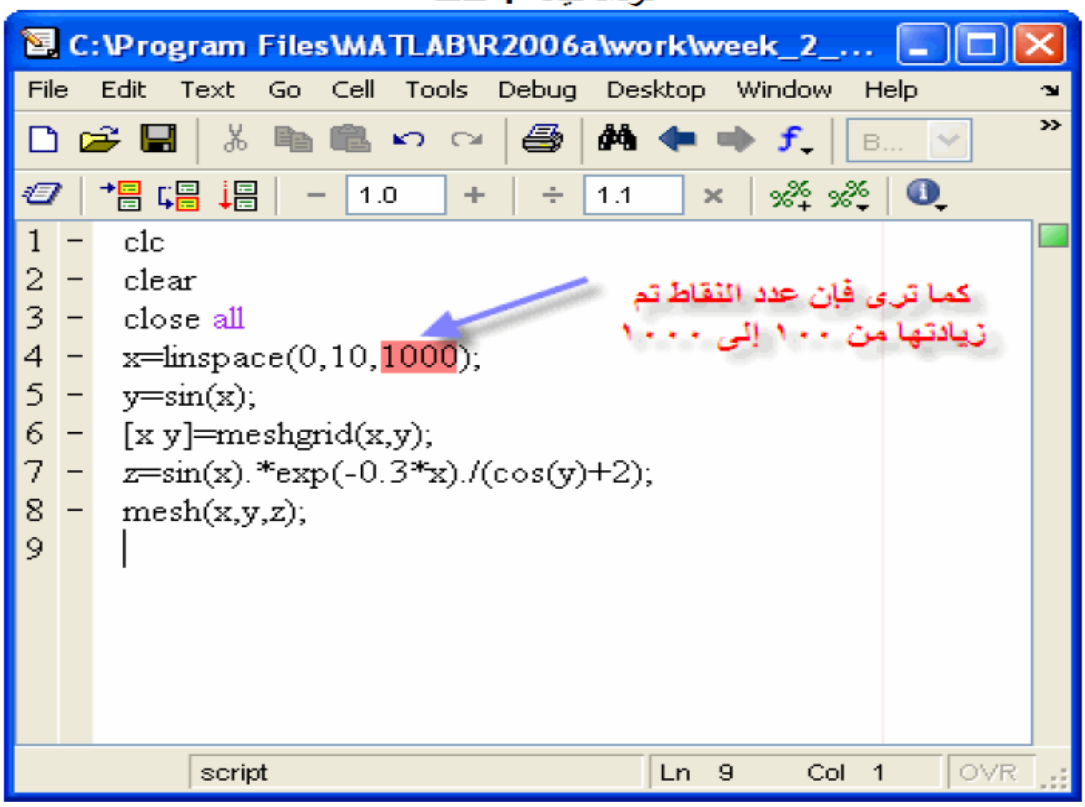
نقوم في هذا المثال بتعريف قيم محور السينات X-Axis وسنقوم بوضع المعادلة التي تصف محور الصادات وعلاقته بمحور السينات، أخيراً وليس آخراً نقوم بوضع العلاقة التي تربط بين محور السينات والصادات.



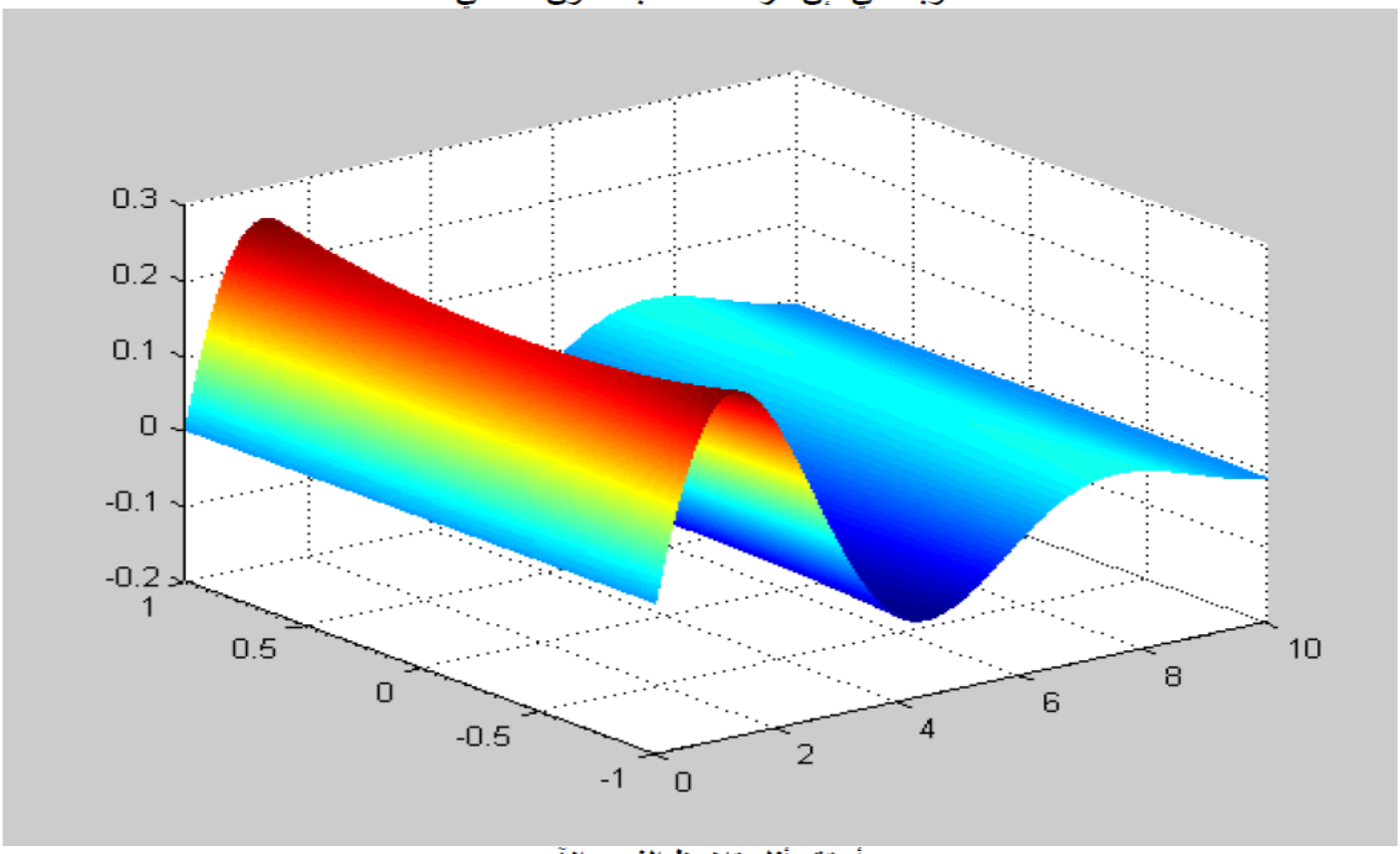
وبالتالي تكون الرسمة الناتجة كالتالي



كما ترى فإن الرسمة الناتجة عبارة عن شبكة تعتمد مجموعة النقاط لكلاً من X & Y فإذا أكثرنا عدد نقاط X وبالتالي تزداد قيمة Y كذلك



وبالتالي فإن الرسمة الناتجة تكون كالتالي



أعتقد أنك تلاحظ الفرق الآن

ملاحظة كلما زادت عدد النقاط كلما زاد الوقت المستغرق لإظهار النتائج في الماتلاب.

إيجاد المساحة تحت المنحنى

هذا المثال من التطبيقات الهامة، حيث سنقوم بتعريف المدخلات ورسم الدالة، ثم سنختار نقطتان نقطتان من على الرسم، ثم سنقوم بإيجاد المساحة بين تلك النقطتين، ونقوم بتظليل الجزء المختار، ولكن سنقوم في هذا المثال بإستخدام أمرين جديدين وهما

trapz لإيجاد المساحة تحت المنحنى

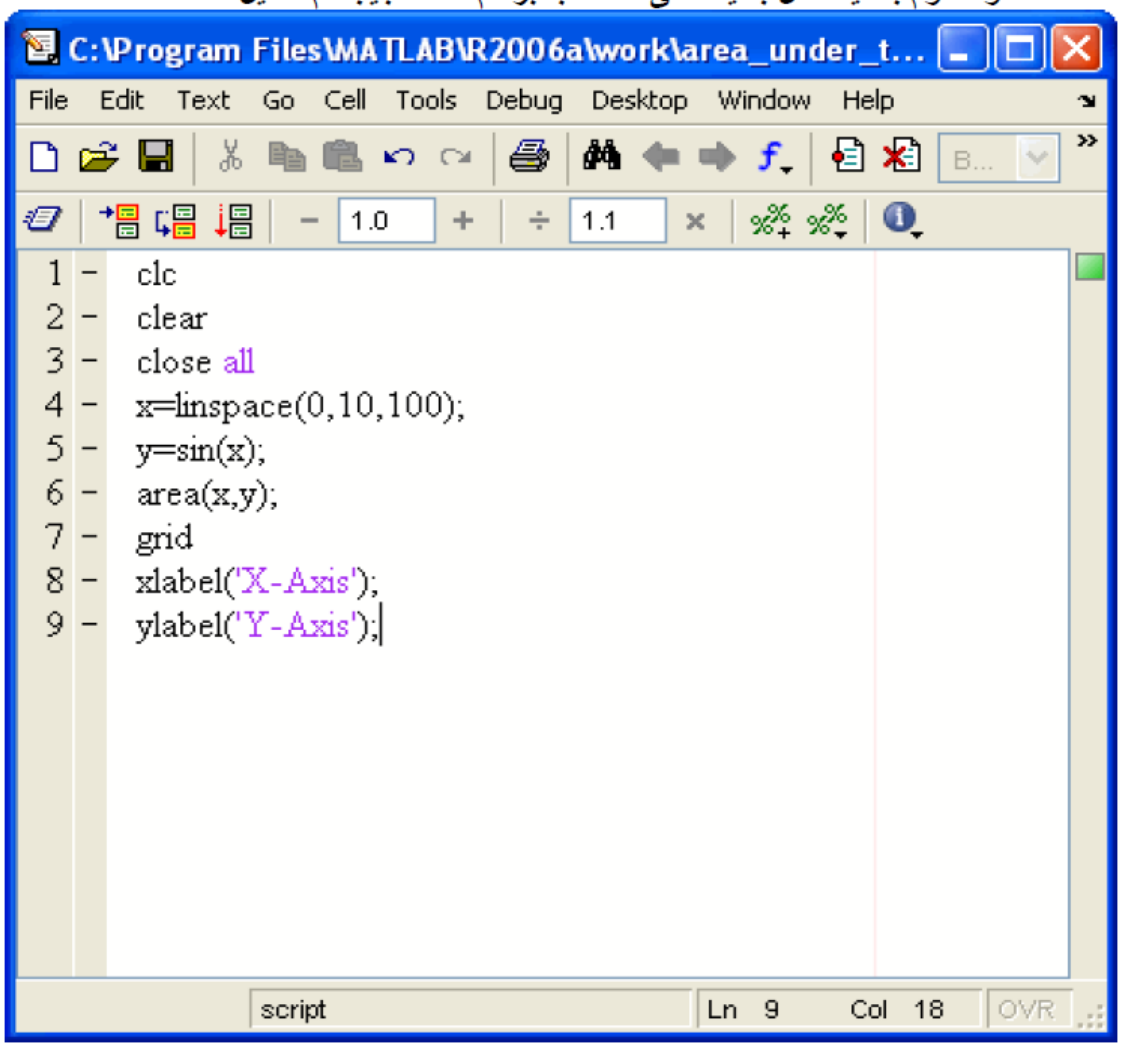
area لتظليل تلك المساحة من الدالة

وسنقوم بشرح الأمر **area**

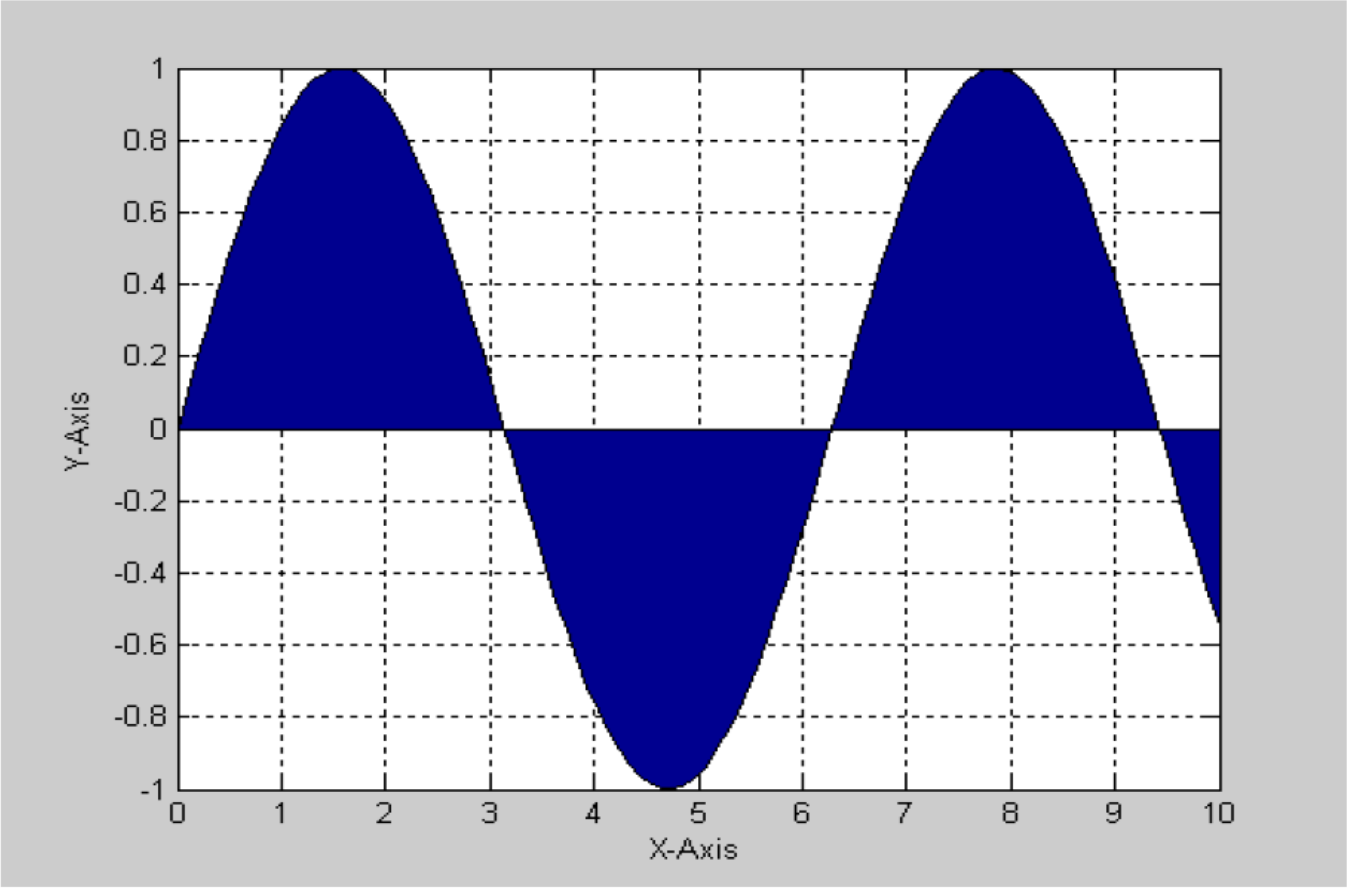
حيث يأخذ الصورة التالية

area(x, y)

وسنقوم بتنفيذ مثال بسيط على الماتلاب برسم دالة الجيب ثم تظليل تلك الدالة



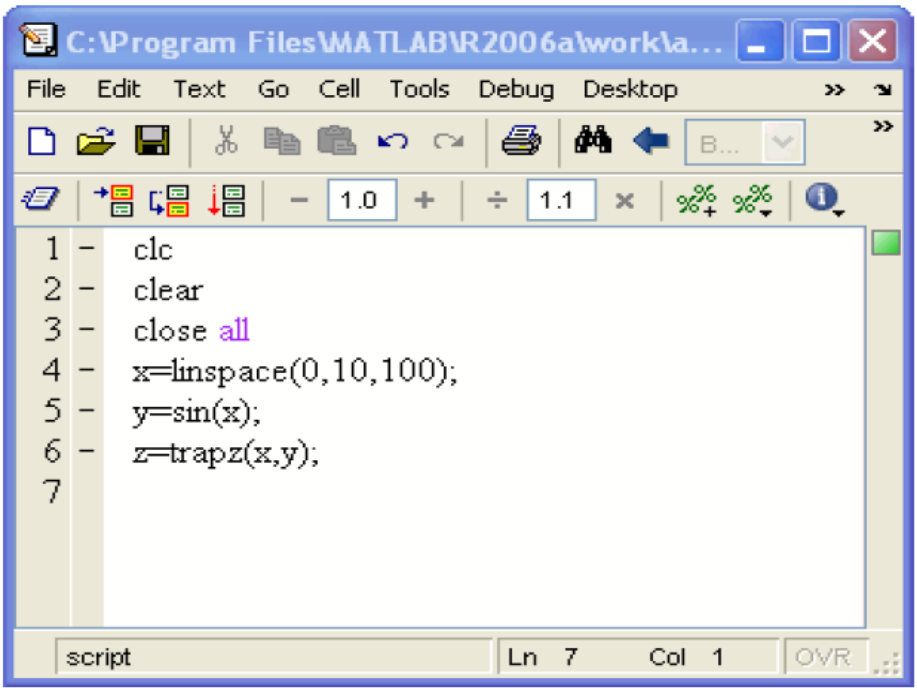
وستظهر الرسمة كالتالي



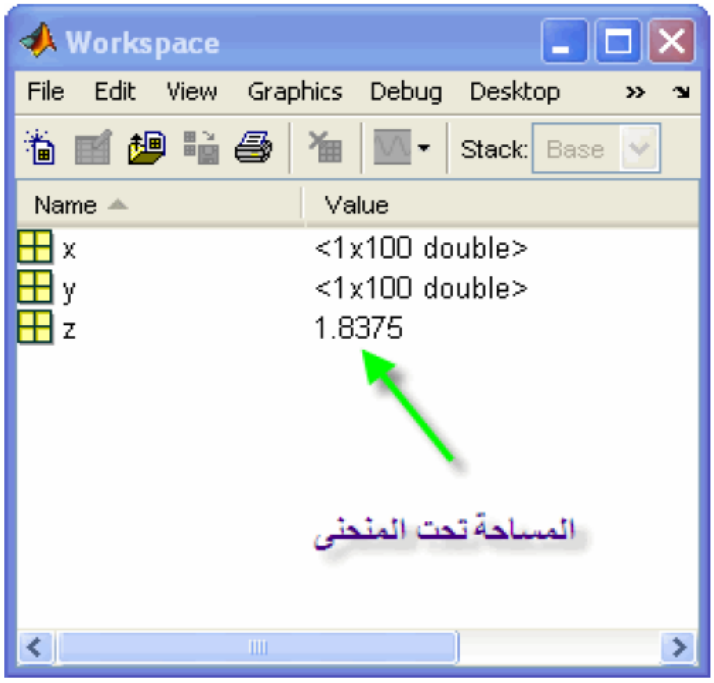
أما بخصوص الأمر **trapz** فيستخدم في إيجاد المساحة تحت المنحنى، حيث يأخذ الصورة التالية

```
trapz( x , y )
```

وسنقوم بعمل برنامج بسيط في إيجاد المساحة تحت منحنى دالة الجيب



وبالتالي ستلاحظ قيمة المساحة من خلال نافذة**Workspace**



الآن نتوجه إلى البرنامج الذي نريد تنفيذه، سنقوم بإدخال المدخلات **inputs** ثم سنقوم بالتعويض بها في المخرجات **outputs**ثم سنقوم بإختيار النقطتان من على الرسمة، ثم سنوجد المساحة تحت المنحنى بين تلك النقطتين، ثم سنقوم بتظليل المساحة بين النقتطين.

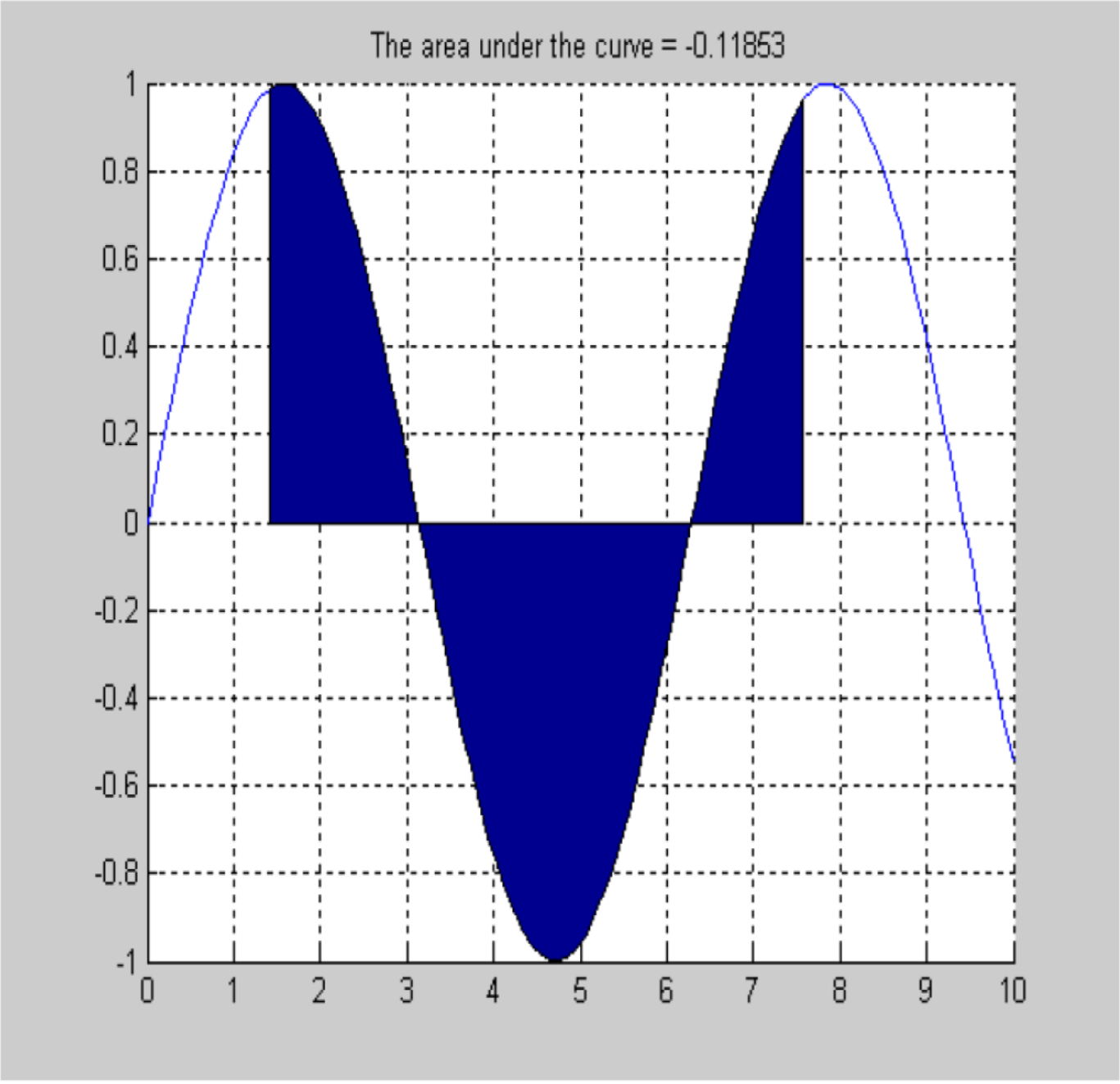
C:\Program Files\MATLAB\R2006a\work\area_under_the_curve.m

File Edit Text Go Cell Tools Debug Desktop Window Help

```
1  clc
2  clear
3  close all
4  hold on
5  x=linspace(0,10,100);              % By defining the inputs
6  y=sin(x);                          % the function to be drawn
7  plot(x,y)                          % By plotting the function
8  grid;
9  [xp yp]=ginput(2);                 % By delecting two points from the drawing
10 ind=find(x>=xp(1) & x<=xp(2));     % By selecting the region in between the two points
11 x=x(ind);                          % By finding the values of x at that region
12 y=y(ind);                          % By finding the values of y at that region
13 area(x,y);                         % By shading that region
14 z=trapz(x,y);                      % By getting the area under the curve
15 str=['The area under the curve = ',num2str(z)];
16 title(str);
```

script Ln 16 Col 12 OVR

وتم إختيار نقتطان عشوائيتان، وظهرت الرسمة كالتالي


The area under the curve = -0.11853
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9
10