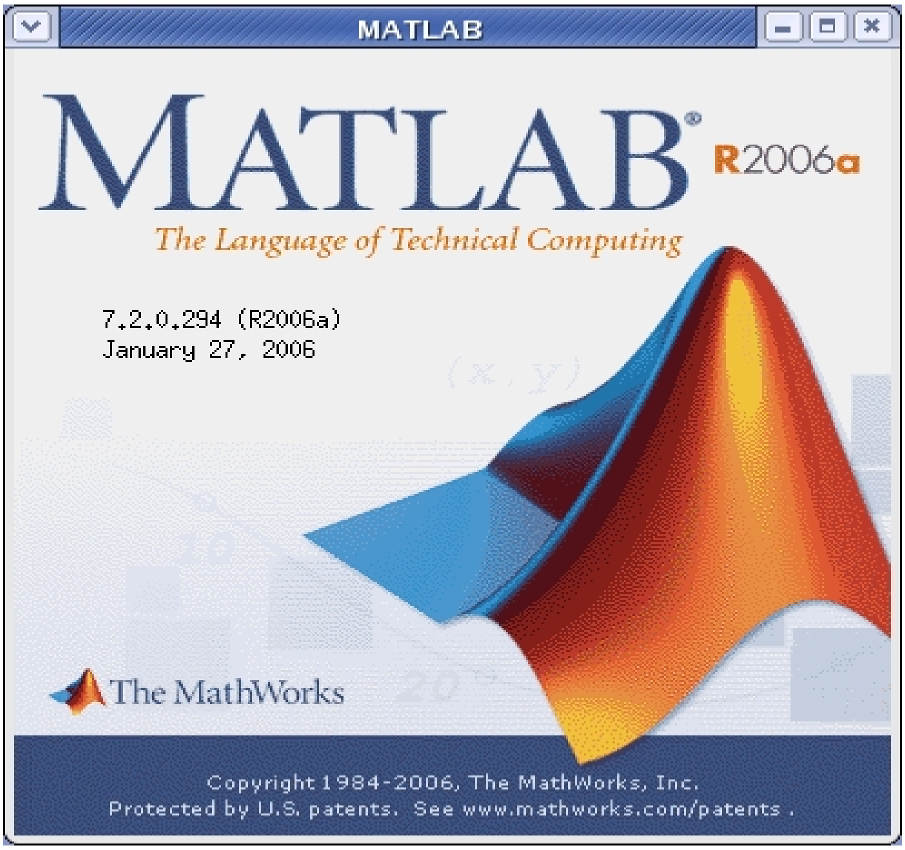
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ MATLAB ΤΟΥ ΜΑΘΗΜΑΤΟΣ

ΜΑΘΗΜΑΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΙΙ

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ 2008- 2009



ΔΙΔΑΣΚΟΝΤΕΣ: Α. Γιαννακόπουλος- Δ. Παππάς

Τα πρώτα 2 μαθήματα του εργαστηρίου MATLAB έγιναν με ένα powerpoint το οποίο βρίσκεται στο e class, folder Matlab.

name : matlab1.ppt

Όποιος - όποια δεν το έχει να το κατεβάσει από την διεύθυνση:

http://eclass.aueb.gr/claroline/document/document.php

Είναι τα βασικά που πρέπει να γνωρίζει κανείς για εισαγωγή.

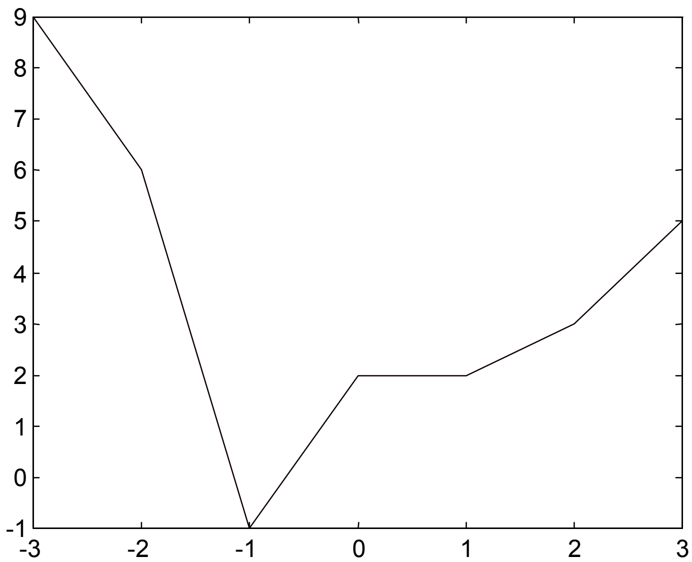
Στη συνέχεια, οι σημειώσεις που δόθηκαν , συμπληρωμένες και βελτιωμένες συγκεντρωθήκαν σε αυτό το φυλλάδιο.

Γραφικά στο Matlab: Εισαγωγή

Τα απλούστερα γραφικά είναι με σημεία στο Καρτεσιανό επίπεδο. Για παράδειγμα.

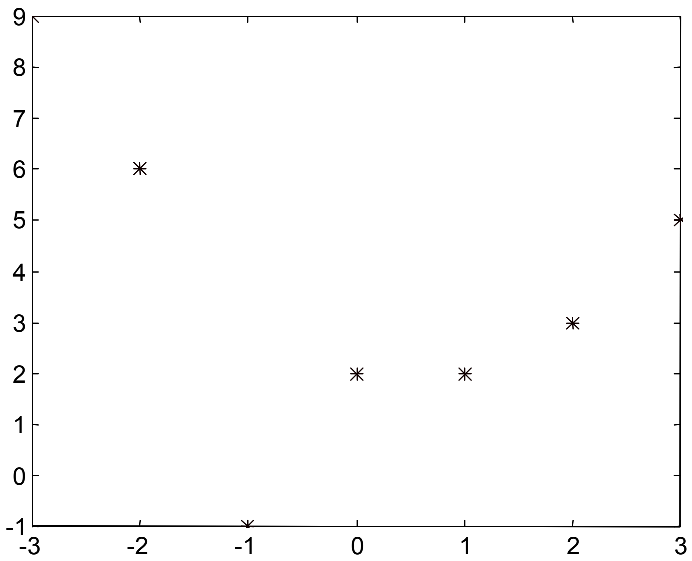
```
>> x = [-3;-2;-1;0;1;2;3];
>> y = [9;6;-1;2;2;3;5];
>> plot(x,y)
```

Το γράφημα φαίνεται στην παρακάτω εικόνα.



Παρατηρείστε ότι το Matlab ενώνει τα σημεία με μία γραμμή. Μία εναλλακτική απεικόνιση είναι η ακόλουθη (δείτε την σχετική εικόνα):

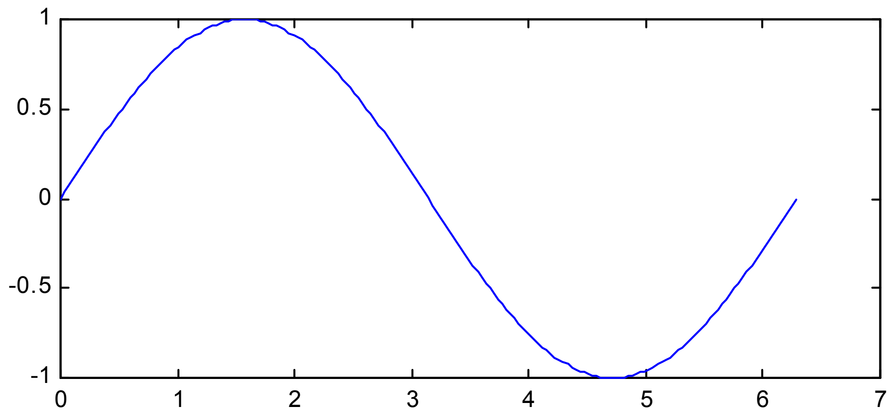
```
>> plot(x,y,'o')
```



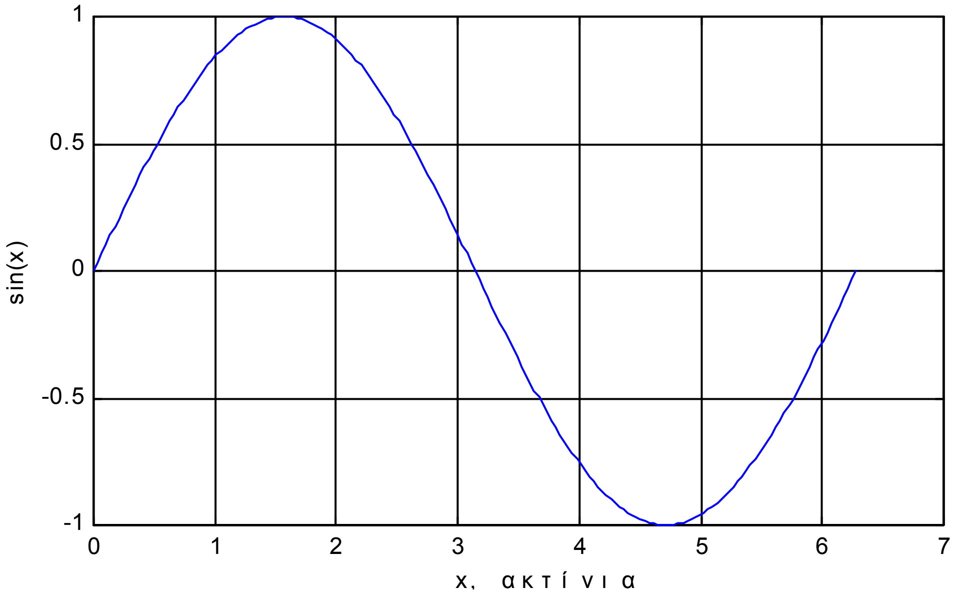
Γραφική παράσταση συνάρτησης στο Matlab

Έστω ότι θέλουμε να κάνουμε τη γραφική παράσταση της συνάρτησης $f(x) = \sin x$ στο διάστημα [0, 2π]. Η βασική συνάρτηση του MATLAB για δυσδιάστατες απεικονίσεις είναι η `plot` (για λεπτομέρειες πληκτρολογήστε `help plot`). Αλλες χρήσιμες συναρτήσεις είναι η `grid` που σχεδιάζει τον κάνναβο , οι `xlabel`, `ylabel` για την εισαγωγή κειμένου στις γραφικές παραστάσεις η axis που βάζει τα διαστήματα των αξόνων.

```
>> x = 0: pi/90: 2*pi;
>> y = sin(x);
>> plot(x,y)
```

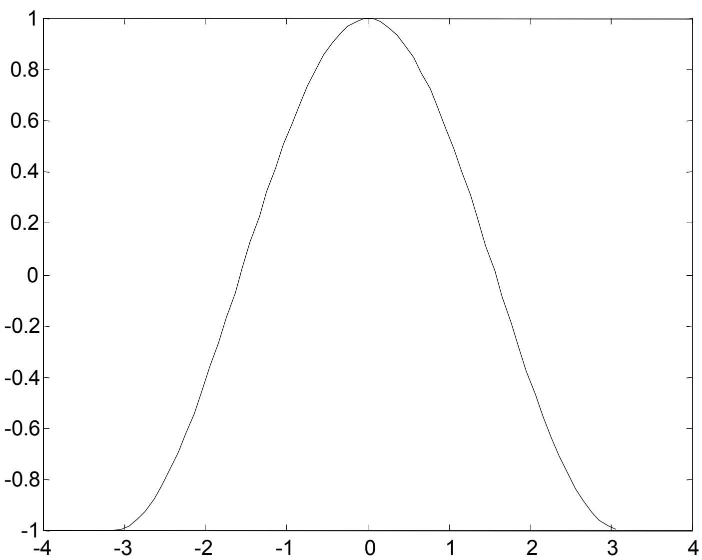


```
>> grid
>> xlabel('x, ακτίνια')
>> ylabel('sin(x)')
```



Είναι πολύ εύκολο να αναπαραστήσουμε γραφικά μια έτοιμη συνάρτηση του Matlab αφού οι συναρτήσεις σε αυτό έχουν την εξής ιδιότητα. Όταν μια συνάρτηση όπως το συνημίτονο εφαρμοσθεί σε ένα διάνυσμα δημιουργείται ένα άλλο διάνυσμα από τις αντίστοιχες τιμές της συνάρτησης. Για παράδειγμα:

```
>> x = (-pi:.1:pi);
>> y = cos(x);
>> plot(x,y)
```



Διαμόρφωση της εμφάνισης και των γραφικών

Όλοι οι υπολογισμοί στο Matlab γίνονται με αριθμητική διπλής ακρίβειας αλλά οι εμφάνιση των αποτελεσμάτων μπορεί να αλλάζει. Έτσι με

`format short` αριθμητική σταθερής υποδιαστολής με 5-ψηφία

`format long` αριθμητική σταθερής υποδιαστολής με 15-ψηφία

`xlabel``(' συμβολοακολουθία '), ylabel('συμβολοακολουθία') :` Τοποθετεί επικεφαλίδες στους άξονες της γραφικής παράστασης.

`title``(' συμβολοακολουθία ')` : Τοποθετεί τίτλο στην γραφική παράσταση.

`axis``([a b c d])` : Ορίζει τους άξονες στην γραφική παράσταση έτσι ώστε $a \le x \le b$, $c \le y \le b$.

`grid` : Προσθέτει ένα τετραγωνικό πλέγμα στην γραφική παράσταση.

`hold on` : «Παγώνει» το παρόν γραφικό για να γίνει μίξη του με τα επόμενα.

`hold off` : Αναιρεί το `hold on`.

`Subplot` : Τοποθετεί πολλά γραγικά σε ένα παράθυρο γραφικών.

Διάφορες συναρτήσεις

`max(x)` : Επιστρέφει τη μεγαλύτερη συντεταγμένη του διανύσματος x.

`min(x)` : Επιστρέφει τη μικρότερη συντεταγμένη του διανύσματος x.

`abs(x)`: Επιστρέφει διάνυσμα με συντεταγμένες τις απόλυτες τιμές των αντίστοιχων συντεταγμένων του διανύσματος x.

`length(x)` : Επιστρέφει το «μήκος» του array, δηλαδή το `max(size(A))`.

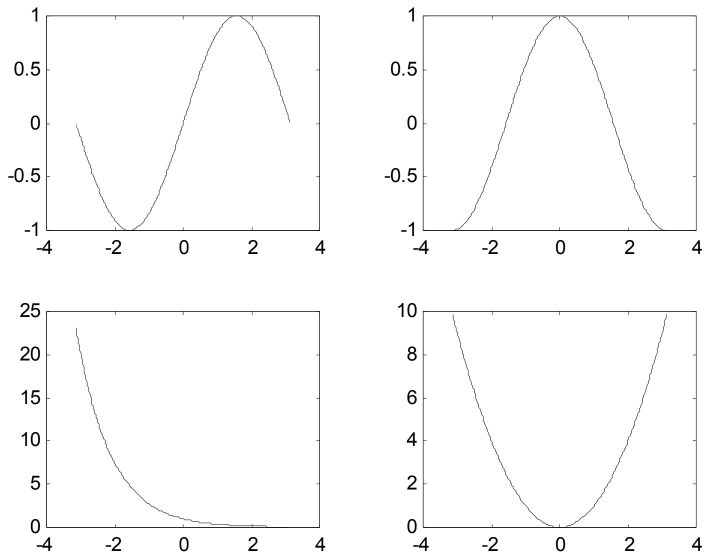
Λεπτομέρειες στις γραφικές παραστάσεις:

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

Για παράδειγμα, η εντολή PLOT(X,Y,'c+:') κάνει plot μια γαλάζια διακεκομμένη γραμμή με ένα «συν» σε κάθε σημείο, ενώ η PLOT(X,Y,'bd') σχεδιάζει μπλε διαμάντι σε κάθε σημείο αλλά δεν τα ενώνει με γραμμές.

Πολλά γραφικά σε ένα παράθυρο

Με την εντολή `subplot` βάζουμε διάφορα γραφικά σε ένα παράθυρο. Για την ακρίβεια, `subplot(m,n,i)` φτιάχνει *mn* γραφήματα, τοποθετημένα σε ένα πίνακα με *m* γραμμές και *n* στήλες. Επίσης τοποθετεί το επόμενο γράφημα στην *i*-η θέση (μετρώντας κατά μήκος των γραμμών). Ακολουθεί ένα παράδειγμα, που παράγει το παρακάτω παράθυρο:



» t=-pi:.02:pi;

» subplot(2,2,1)

» plot(t,sin(t))

» subplot(2,2,2)

» plot(t,cos(t))

» subplot(2,2,3)

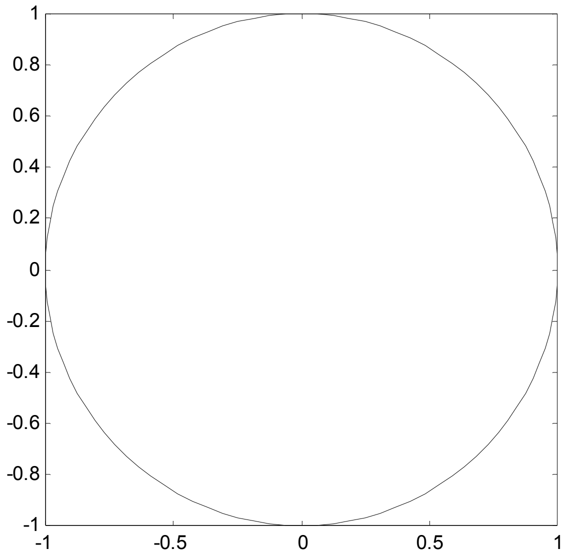
» plot(t,exp(-t))

» subplot(2,2,4)

» plot(t,t.^2)

Παραμετρικές καμπύλες: Γραφική παράσταση

Είναι πολύ εύκολο να φτιάξουμε το γράφημα της καμπύλης ($f(t)$,$g(t)$) στις δύο διαστάσεις. Για παράδειγμα το παρακάτω γράφημα παράγεται γράφοντας

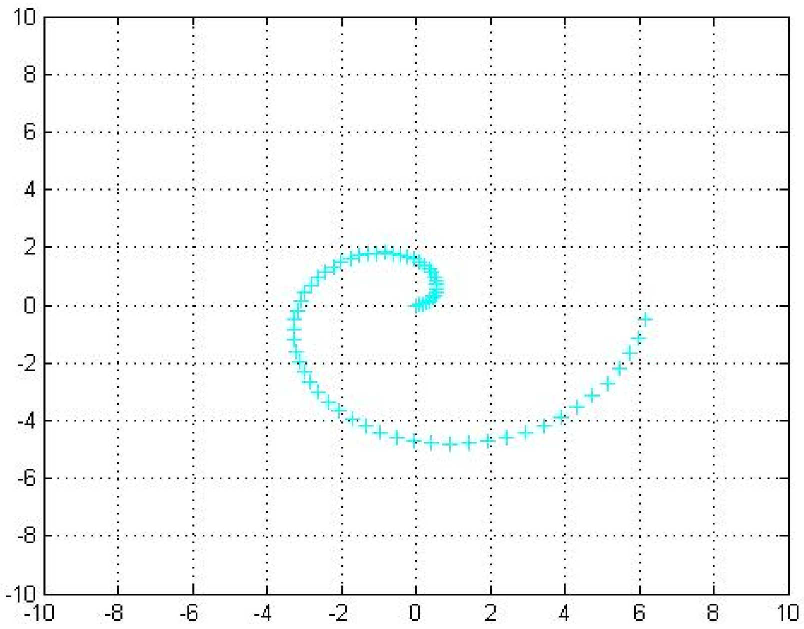



```
>> t = (0:2*pi/100:2*pi)';
>> plot(cos(t),sin(t))
>> axis('square')
```

(Να σημειώσετε ότι η εντολή `axis('square')` φτιάχνει τετραγωνικό πλαίσιο, αλλιώς θα βλέπαμε έλλειψη αντί για κύκλο).

Εναλλακτικά, ανοίγουμε τον editor και γράφουμε το εξής πρόγραμμα:

```
for t = 0:0.1:2*pi;
plot(t*cos(t),t*sin(t),'c+')
grid on
axis([-10 10 -10 10])
hold on
end
hold off
```



Στη συνέχεια μπορούμε να κάνουμε δοκιμές και παραλλαγές του παραπάνω προγράμματος χρησιμοποιώντας τα δεδομένα της παραγράφου «Λεπτομέρειες στις γραφικές παραστάσεις».

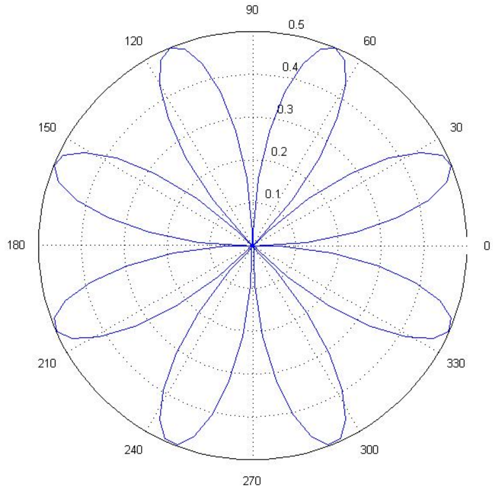
Πχ :

```
for t = 0:0.1:2*pi;
plot(t*cos(t)-4,t*(sin(t))^2,'ro')
grid on
axis([-10 10 -10 10])
hold on
end
hold off
```

Πολικές συντεταγμένες

Θα σχεδιάσουμε την καμπύλη με τύπο r = sin2φcos2φ.

t =linspace(0,2*pi);
>> r=sin(2*t).*cos(2*t);
>> polar(t,r)



Σειρές Taylor- Mc Laurin . Σειρές Fourier.

1. Με χρήση του πακέτου Symbolic toolbox :

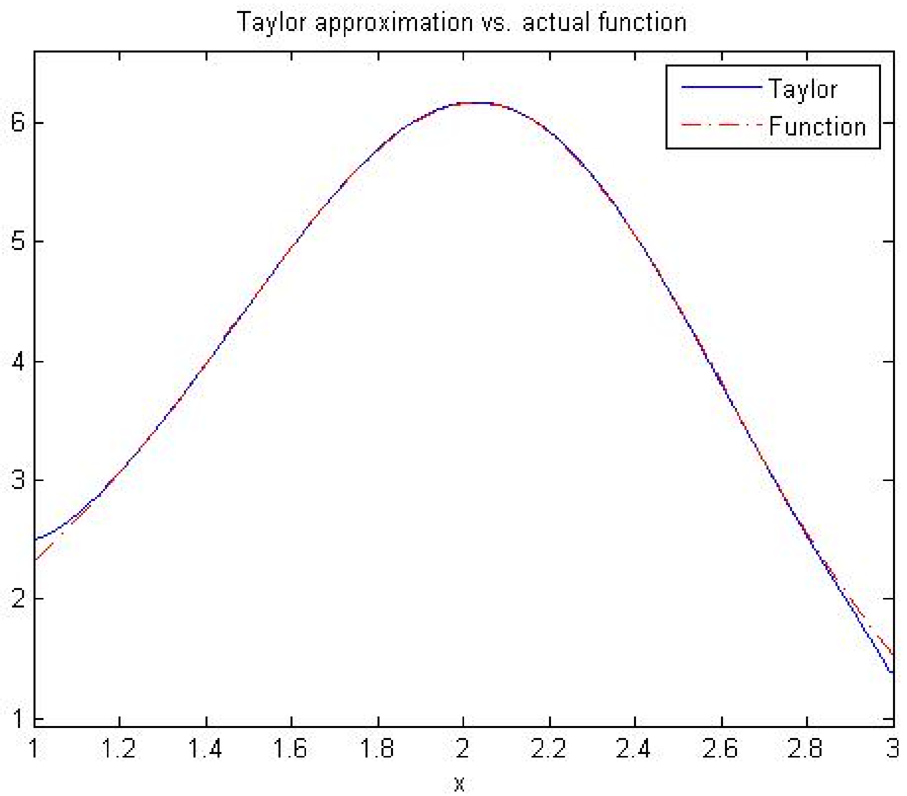
```
>> syms x
>> f=sin(5*x)

>> T = taylor(f,8)  ή, εναλλακτικά,
>> t = taylor(f,8,2) ( γύρω από το σημείο 2)
>> pretty(T)  ( για να φανεί το πολυώνυμο)

                       3   625   5   15625   7
        5 x - 125/6 x  + ---   x  -  -----    x
                          24          1008
```

Παράδειγμα 1:

```
syms x
g = exp(x*sin(x))
t = taylor(g,12,2);
```

Θα κάνουμε την γραφική παράσταση και των 2 μαζί για να φανεί η προσέγγιση:

```
xd = 1:0.05:3; yd = subs(g,x,xd);
ezplot(t, [1,3]); hold on;
plot(xd, yd, 'r-.')
title('Taylor approximation vs. actual function');
legend('Taylor','Function')
```



Τώρα, θα δούμε την αριθμητική διαφορά του πολυωνύμου Taylor και της τιμής της συνάρτησης:

Παράδειγμα 2:

syms x
f = 1/(5+4*cos(x))
T = taylor(f,8)

T = 1/9+2/81*x^2+5/1458*x^4+49/131220*x^6

Ορίζω το πολυώνυμο που πήραμε παραπάνω:

>> p=[49/131220 0 5/1458 0 2/81 0 1/9]

P =0.0004   0   0.0034   0   0.0247   0   0.1111

>> v=polyval(p,0) % Η τιμή του πολυωνύμου στο σημείο 0 :

v =   0.111111

>> v1=1/(5+4*cos(0)) % Η τιμή της συνάρτησης στο σημείο 0:

v1 =   0.11111

Παράδειγμα 3:
Μπορούμε να κατασκευάσουμε ένα loop όπως το παρακάτω:

```
f = exp(x)
for i=1:10
a = taylor(f,i)
end
```

και να πάρουμε τα διαδοχικά πολυώνυμα βαθμού i :
a =1+x
a =1+x+1/2*x^2
a =1+x+1/2*x^2+1/6*x^3
a =1+x+1/2*x^2+1/6*x^3+1/24*x^4
a =1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5
a =1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+1/720*x^6
a =1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+1/720*x^6+1/5040*x^7
a =
1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+1/720*x^6+1/5040*x^7+1/40320*x^8

a = 1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+1/720*x^6+
1/5040*x^7+1/40320*x^8+1/362880*x^9

Κλπ…

2. Χωρίς την χρήση του symbolic toolbox:

Παράδειγμα 4:
Με την βοήθεια του παρακάτω κώδικα, εισάγουμε :

```
% x: power of e to be solved.
% d: desired accuracy evaluated as the value of the n-term Taylor
series
%       approximation - the (n-1)-term Taylor series approximation.
% n: number of terms in the Taylor series.
% t_n: nth term in Taylor series.
% e_approx: solution to the n-term Taylor series.
```

Και παίρνουμε την απάντηση (παράδειγμα)

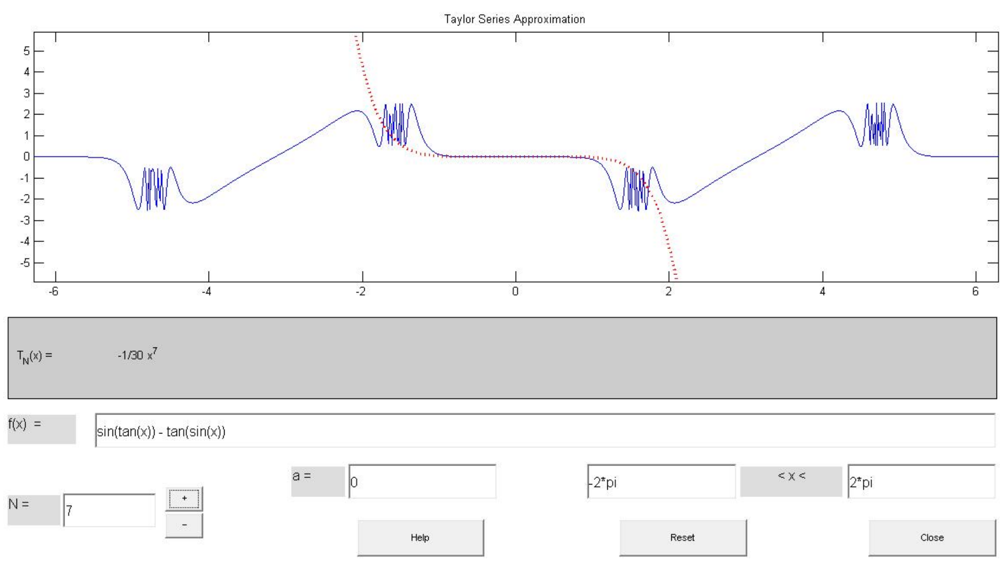
Enter x: 4
Enter accuracy of approximation: 0.001
Approximate solution of e^x:
54.597882905650103
Number of terms in Taylor series:
16
Accuracy achieved:
8.211079533830857e-004

Κώδικας: (Βρίσκεται στο server του εργαστηρίου, και στο e-class: Taylor code.m)

```
% Variable definitions:
% x: power of e to be solved.
% d: desired accuracy evaluated as the value of the n-term Taylor
series
%       approximation - the (n-1)-term Taylor series approximation.
% n: number of terms in the Taylor series.
% t_n: nth term in Taylor series.
% e_approx: solution to the n-term Taylor series.

x=input('Enter x: ');
d=input('Enter accuracy of approximation: ');

n=1;            % approximation begins with single-term solution
t_n=x^(n-1)/factorial(n-1);
e_approx=t_n;   % initial approximation
while t_n > d                       % accuracy reached?
    n=n+1;                          % add another term.
    t_n=x^(n-1)/factorial(n-1);     % evaluate the new term.
    e_approx=e_approx+t_n;          % add the new term to the
approximation.
end
disp('Approximate solution of e^x:');
disp(e_approx);
disp('Number of terms in Taylor series:');
disp(n);
disp('Accuracy achieved:');
disp(t_n);
```

3. Σειρές Taylor- Mc Laurin με την βοήθεια GUI

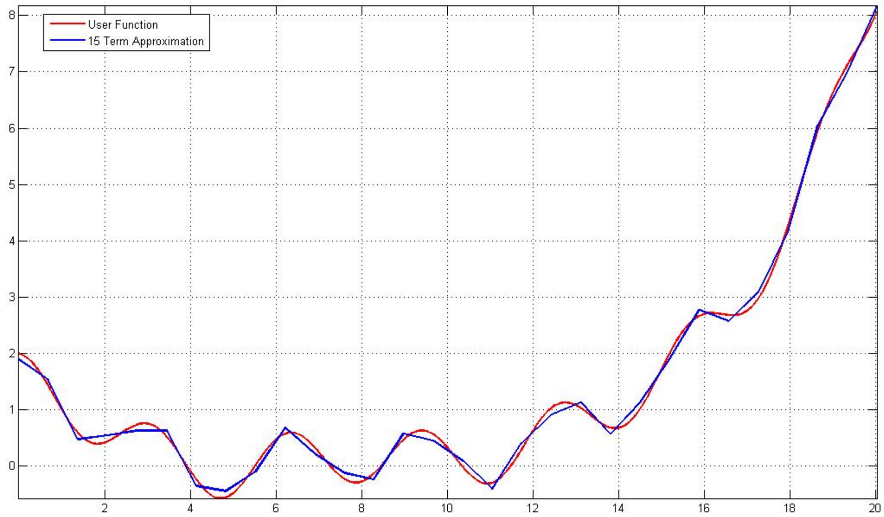
Για να πάρουμε μια πρώτη ιδέα (γραφικά) της προσέγγισης Taylor, θα δούμε την εντολή taylortool:
Η εντολή taylortool κατασκευάζει ένα GUI (Grafical User Interface – διασύνδεση με χρήστη μέσω γραφικών) το οποίο μας δείχνει την συνάρτηση και το πολυώνυμο Taylor βαθμού N, γύρω αποτο σημείο α.
Παράδειγμα:
taylortool('sin(tan(x)) - tan(sin(x))')



Fourier series:

Κώδικας: (Βρίσκεται στο server του εργαστηρίου, και στο e class, fseriesdemo.m)

```
fseriesdemo(f,20,100)
Εισάγουμε την συνάρτηση f, στο διάστημα [0, 20], με 100 όρους :

f = @(x) sin(x)./x+(cos(x)).^2-exp(x/8)+x.^(x/20);
```



Προχωρημένα Γραφικά

Το Matlab μπορεί να παράγει διάφορους τύπους γραφικών: 2-διάστατες καμπύλες, 3-διάστατες επιφάνειες, ισοϋψείς 3-διάστατων επιφανειών, παραμετρικές καμπύλες σε δύο ή τρεις διαστάσεις. Τα περισσότερα μπορεί να τα βρει κανείς από τα εγχειρίδια ή το σύστημα βοήθειας του Matlab. Εδώ, με διάφορα παραδείγματα, θα επιδείξουμε κάποιες δυνατότητες. Θα εξηγήσουμε κάποια βασικά στοιχεία για την παραγωγή 3-διάστατων γραφικών.

3-διάστατα γραφικά

Για να σχεδιάσουμε το γράφημα μιας επιφάνειας στις τρεις διαστάσεις (ή και τις ισοϋψείς μιας επιφάνειας), είναι απαραίτητο να υπολογίσουμε την συνάρτηση σε ένα ορθογώνιο πλέγμα σημείων. Γι' αυτό χρησιμοποιούμε την εντολή meshgrid. Πρώτα φτιάχνουμε διανύσματα που περιγράφουν το πλέγμα κατά μήκος της *x*- και της *y*-διάστασης: Δηλαδή τις διαμερίσεις x και y.

<u>Παράδειγμα 1 :</u>

```
» x=-pi/2:pi/20:pi/2;
» y=0:pi/30:pi/2;
```

Μετά ανοίγουμε το πλέγμα με την εντολή meshgrid:

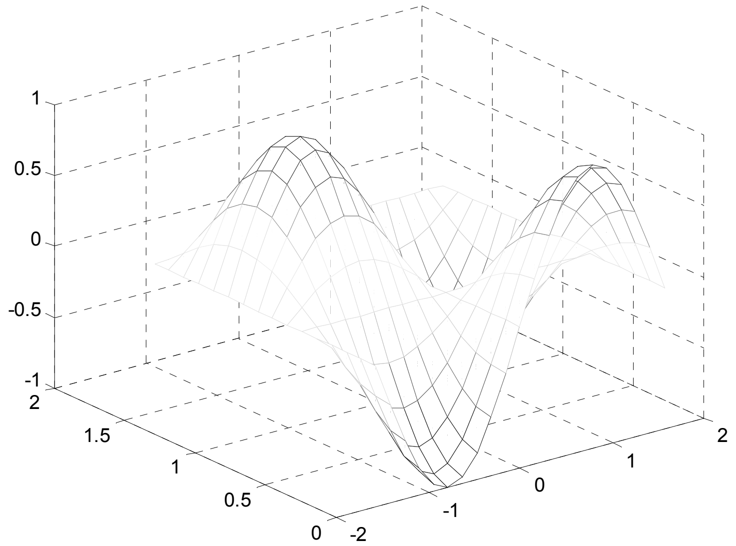
```
» [X,Y]=meshgrid(x,y);
» whos
  Name      Size        Bytes  Class
  X        16x21        2688  double array
  Y        16x21        2688  double array
  x         1x21         168  double array
  y         1x16         128  double array
Grand total is 709 elements using 5672 bytes
```

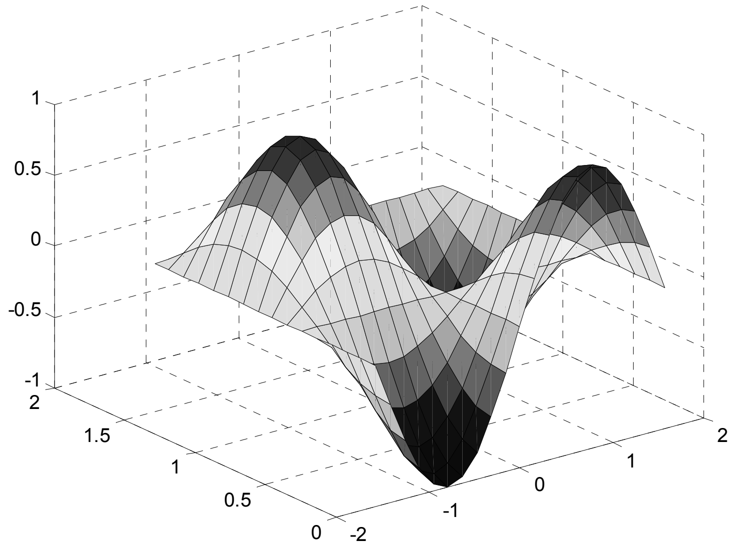
Με την meshgrid δημιουργούμε τον πίνακα X με τη διαμέριση *x* σε κάθε του γραμμή, και τον πίνακα Y με τη διαμέριση *y* σε κάθε του στήλη. Μετά μπορούμε να υπολογίσουμε την συνάρτηση δύο μεταβλητών $z = f(x,y)$ στο προκαθορισμένο πλέγμα:

```
» z=sin(2*X).*cos(3*Y);
```

Έχοντας δημιουργήσει τον πίνακα Z, με τιμές της συνάρτησης χρησιμοποιώντας την εντολή mesh ή την εντολή surf:

```
» mesh(x,y,z)
» surf(x,y,z)
```

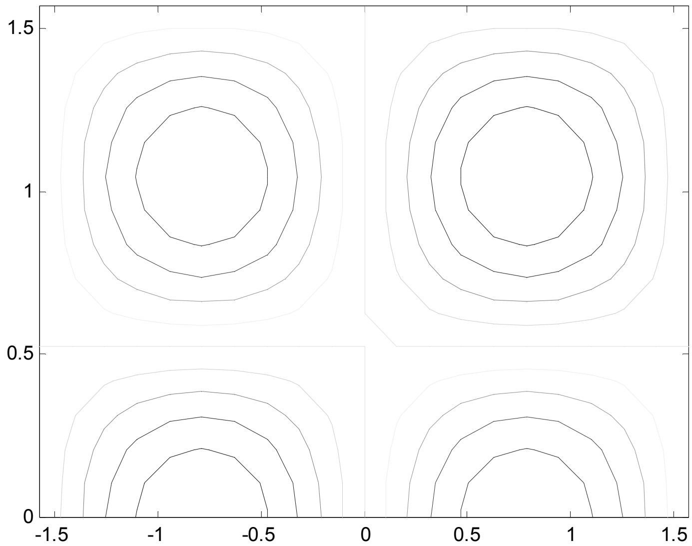




(Η διαφορά είναι ότι η surf βάζει σκιά στην επιφάνεια ενώ η mesh όχι.)

Μπορούμε επίσης να φτιάξουμε τις ισοϋψείς:

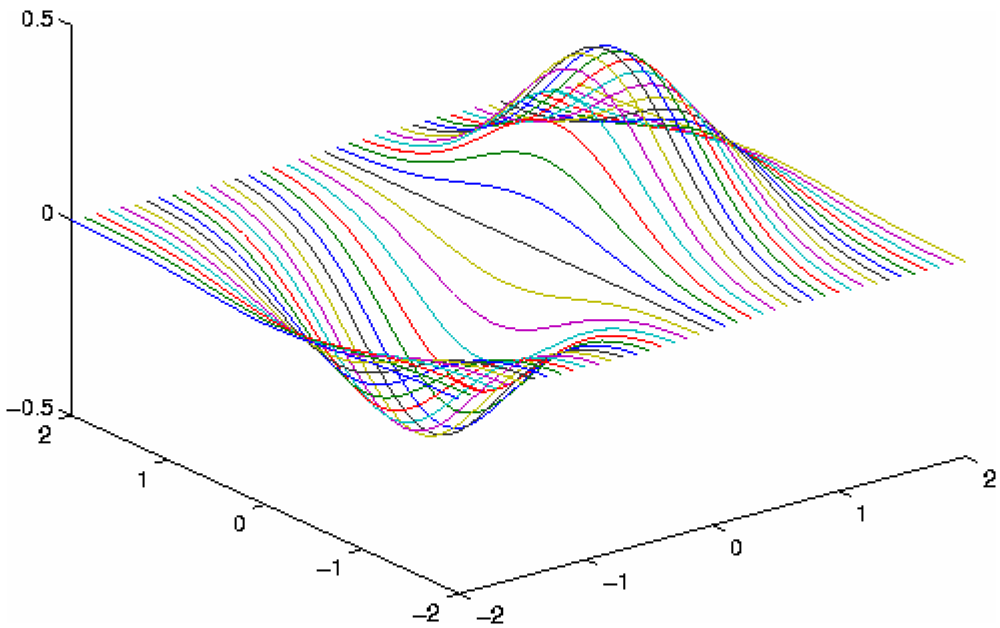
```
>> contour(x,y,z)
```



Χρησιμοποιείστε το help για να μάθετε περισσότερα πάνω στο θέμα. Πάντως οι εντολές αυτές γίνονται πολύ χρονοβόρες όσο πυκνώνει το πλέγμα.

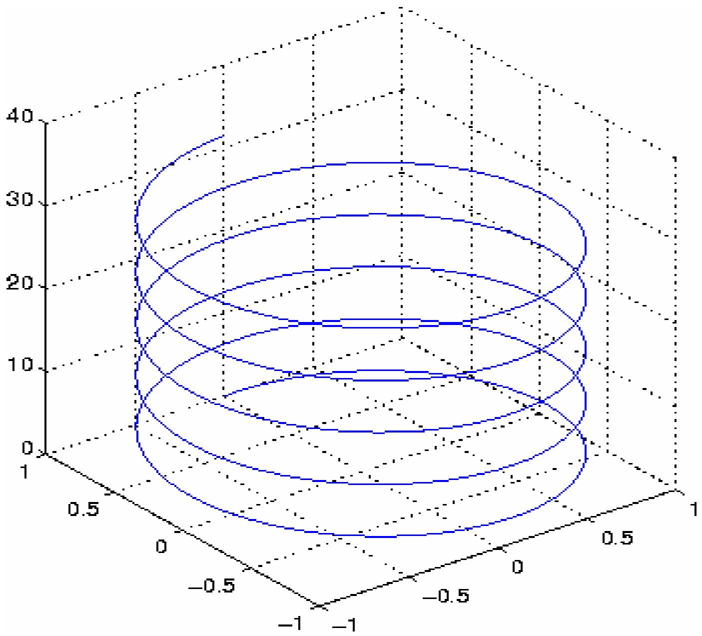
<u>Παράδειγμα 2 :</u>

```
[X,Y] = meshgrid([-2:0.1:2]);
Z = X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z)
grid on
```



Παράδειγμα 3 :

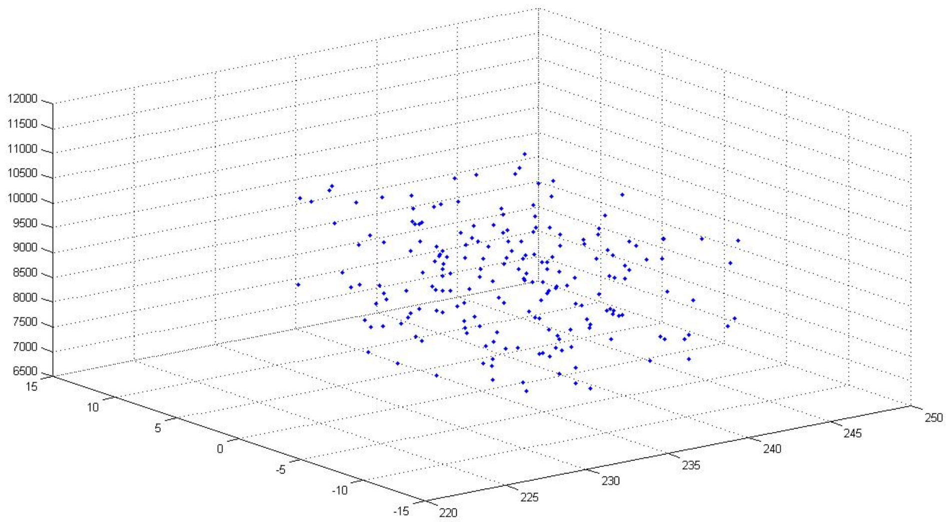
```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
axis square; grid on
```



Παράδειγμα 4 :

Μπορούμε να πάρουμε διάφορα τυχαία σημεία στον χώρο με την εντολή rand (m,n) η οποία μας δίνει έναν πίνακα m επι n με στοιχεία από το διάστημα [0,1] κατανεμημένα ομοιόμορφα :

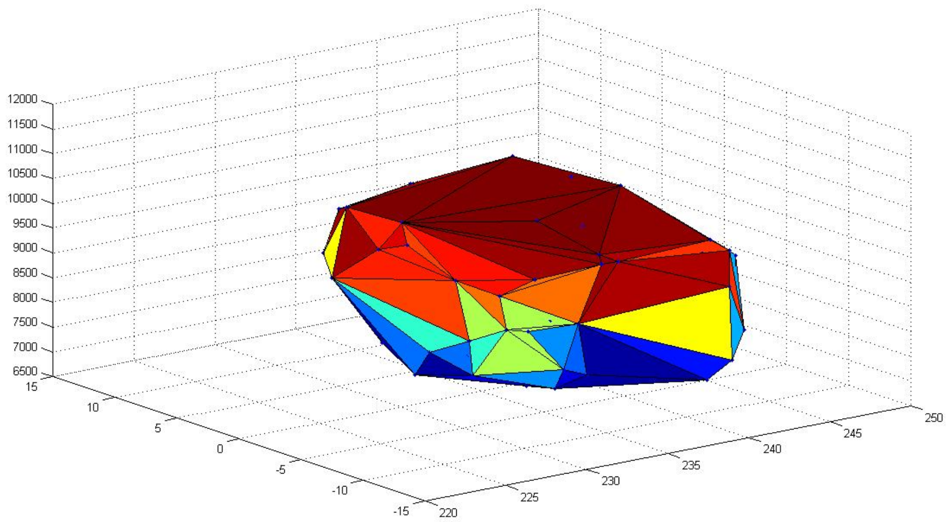
```
h=7000+(3000)*rand(200,1);
v=230+(15)*rand(200,1);
d=-10+(20)*rand(200,1);
plot3(v,d,h,'.')
hold
axis([220 250 -15 15 6500 12000]);
grid on
```



Στη συνέχεια μπορούμε να τα καλύψουμε με μια επιφάνεια:

K = convhulln([v d h]);

trisurf(K,v,d,h);

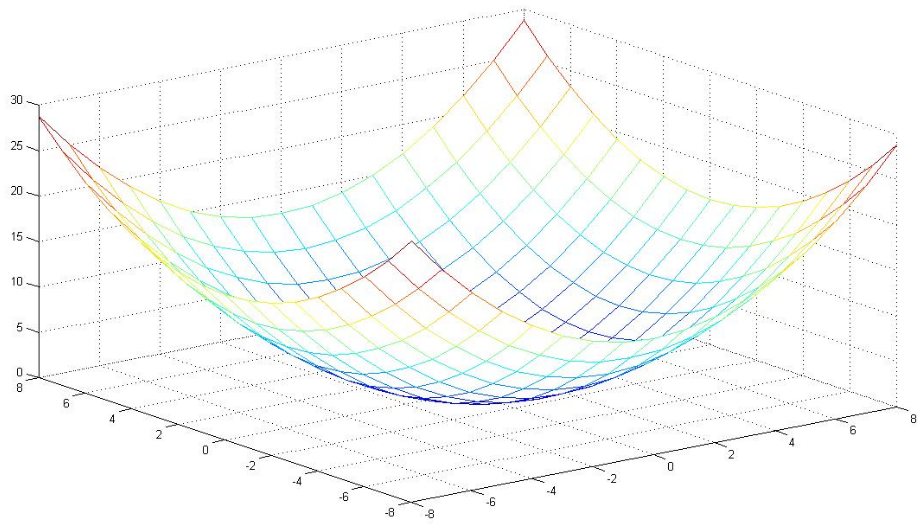


Convex hull: Η κυρτή γραμμική θήκη του συνόλου των σημείων.

Παράδειγμα 5 Ε<u>πιφάνειες 2^{ου} βαθμού:</u>**:**

Η εντολή mesh μας βοηθά να δούμε τις επιφάνειες αυτές:

```
>> x=-8:8;
>> y=-8:8;
>> [X,Y]=meshgrid(x,y);
>> Z=[X.^2/4+Y.^2/5];
>> mesh(X,Y,Z)
```



Μπορούμε να δούμε πολλές παραλλαγέστων επιφανειών 2^{ου} βα8μού (πχ παραβολοειδές μονόχωνο ή δίχωνο , ελλειψοειδές κλπ..)

ΟΙ ΣΥΝΑΡΤΗΣΕΙΣ ΤΟΥ MATLAB

Το MATLAB μας παρέχει ένα πλήθος ενσωματωμένων συναρτήσεων όπως τετραγωνική ρίζα, εκθετικές και λογαριθμικές συναρτήσεις, τριγωνομετρικές και αντίστροφες τριγωνομετρικές συναρτήσεις κ.ά.:

```
>> sqrt(2)          % τετραγωνική ρίζα
ans =
   1.4142
>> exp(1)           % εκθετική συνάρτηση
ans =
   2.7183
>> log(exp(1))      % φυσικός λογάριθμος
ans =
   1
```

```
>> log10(10^2)    % δεκαδικός λογάριθμος
ans =
   2


% Τριγωνομετρικές συναρτήσεις

>> pi                          % η σταθερά π
ans =
   3.1416
>> sin(pi/4)                   % ημίτονο
ans =
   0.7071
>> cos(pi/2)                   % συνημίτονο
ans =
   6.1230e-017      ← πρακτικά το αποτέλεσμα είναι 0
>> tan(pi/4)                   % εφαπτομένη
ans =
   1.0000
>> asin(0.5)                   % τόξο ημιτόνου
ans =
   0.5236
>> atan(1)          % τόξο εφαπτομένης
ans =
   0.7854
```

όπου το σύμβολο '%' χρησιμοποιείται για την εισαγωγή σχολίων. Αν η γωνία δίνεται σε μοίρες, τότε την μετατρέπουμε σε ακτίνια πολλαπλασιάζοντας με το π/180. Παράδειγμα υπολογισμού του cos(60^o):

```
>> cos(60*pi/180)
ans =
   0.5000
```

Οι παρακάτω συναρτήσεις δρούν σε μια μεταβλητή, αλλά επίσης και σε πίνακες ανα στοιχείο:

sin	asin	exp	abs	round
cos	acos	log (natural log)	sqrt	tan
atan	rem (remainder)		sign	

ceil (στρογγυλοποίηση προς τα πάνω)

floor (στρογγυλοποίηση προς τα κάτω)

Οι παρακάτω συναρτήσεις δρούν σε διανύσματα, αλλά επίσης και σε πίνακες ανα στήλη:

max sum median any (παραβλέπει τα NAN)

min prod mean all (δείχνει 1 σε μηδενικούς πίνακες) sort std

Ορισμός συναρτήσεων

Στον editor, γράφουμε την συνάρτηση που θα ορίσουμε και την σώζουμε στον φάκελο που δουλεύουμε.
π.χ
```
function f1=function1(t)
f1=t*cos(t);

function f2=function2(y)
f2=y(1)+y(2);

function f3=function3(y)
f3=[y(1)+y(2) y(2)]';            {το ' ορίζει διάνυσμα στήλη}

function f4=function4(t,y)
f4=[t*y(2) (t+1)*y(1)]';
```

Το Matlab μας παρέχει έτοιμες συναρτήσεις για την επίλυση κάποιων συνηθισμένων προβλημάτων όπως η αριθμητική παραγώγιση και ολοκλήρωση. Επίσης υπάρχουν και πολλά εργαλεία (τα Toolboxes) που παρέχουν πολύ μεγάλη ποικιλία συναρτήσεων σε διάφορα θέματα. Για παράδειγμα στην προσέγγιση με splines, στατιστική, επεξεργασία σήματος, τα νευρωνικά δίκτυα κ. ά. Υπάρχουν πάνω από 20 Toolboxes, με πέραν των εκατό ειδικών συναρτήσεων το καθένα.

Αριθμητική και Συμβολική Παραγώγιση και Ολοκλήρωση

Παραγώγιση: Έστω η **g(x) = x^2 cos x**.
syms x

g = x^2*cos(x)

diff(g, x) % (Δηλ. $g'(x)$)

ans =2*x*cos(x)-x^2*sin(x)

Για την 2^η παράγωγο: $g''(x)$
diff(g, x, 2)

ans = 2*cos(x)-4*x*sin(x)-x^2*cos(x)

diff(x^3-x^2+2, x) % Χωρίς να ορίσω συνάρτηση

ans = 3*x^2-2*x

Εισάγουμε μια σταθερά α :
syms a
diff(x^3 - a*x^2 + 2, x)

ans = 3*x^2-2*a*x

Παραγωγίζω ως προς α:
diff(x^3 - a*x^2 + 2, a)

ans = -x^2

Έστω ότι θέλω να υπολογίσω την **dg/dx** για x = 2.1.
gp=diff(g, x)

gp = 2*x*cos(x)-x^2*sin(x)

subs(gp, x, 2.1)

ans = -5.9271

Εναλλακτικά:
subs(diff(g, x), x , 2.1)

ans = -5.9271

Ομοίως, θα υπολογίσουμε αόριστο ολοκλήρωμα:

syms x
int(x*sin(x), x)
ans = sin(x)-x*cos(x)

Βλέπουμε ότι όντως η παράγωγος της απάντησης μας δίνει την αρχική συνάρτηση:
diff(ans, x)
ans = x*sin(x)

Εισάγω άλλη μια μεταβλητή, α, για να δούμε πως συμπεριφέρεται η ολοκλήρωση με 2 μεταβλητές:
syms a
int(a*sin(x), x)
ans = -a*cos(x)

Εάν γράψω:
int(a*sin(x), a)

Ποιος είναι ο ρόλος της μεταβλητής μετά το κόμμα? Κάντε μερικές δοκιμές για να πάρετε απάντηση.

Στη συνέχεια θα υπολογίσουμε ορισμένα ολοκληρώματα: πχ, **x sin(x)** στο διάστημα i **[0,pi/2]**:

int(x*sin(x), x, 0, pi/2)
ans = 1

Αριθμητική ολοκλήρωση:

Για τον υπολογισμό του ορισμένου ολοκληρώματος $\int_a^b f(x)dx$, το Matlab μας παρέχει τις συναρτήσεις quad, quadl. Η quadl δίνει αποτελέσματα υψηλότερης ακρίβειας με μικρότερο υπολογιστικό κόστος από την quad.

Παράδειγμα 1: Έστω ότι θέλουμε να ολοκληρώσουμε την συνάρτηση e^{-x}, στο διάστημα [−1,1]. Το αποτέλεσμα θα πρέπει να είναι $e - e^{-1}$. Η quad μας δίνει τα παρακάτω:

» q=quad(inline('exp(-x)'),-1,1,1e-9)

q =   2.35040238728779

» exp(1)-exp(-1)-q

ans =   -1.896260926059767e-013   Δηλαδή, είχαμε σφάλμα της τάξης του 10^{-13}.

Παράδειγμα 2:

```
>> x = -1:0.1:2;

>> y=humps(x);

>> plot(x,y)

>> area =trapz(x,y)

 area =   25.9174
```

(Η humps είναι μια κλασσική έτοιμη συνάρτηση για παραδείγματα στο matlab
Η trapz υπολογίζει εμβαδά με την μέθοδο των τραπεζίων, όχι με μεγάλη ακρίβεια.)

```
>> H_humps =@humps;
>> quad(H_humps,-1,2)
 ans =   26.3450
>> quadl(H_humps,-1,2)
 ans =   26.3450
```

Σε αντίθεση με την ολοκλήρωση, η αριθμητική παραγώγιση είναι πιο δύσκολη.

Η ολοκλήρωση περιγράφει μια μακροσκοπική ιδιότητα μιας συνάρτησης, ενώ η παραγώγιση περιγράφει την κλίση της συνάρτησης σε ένα σημείο, που είναι μικροσκοπική ιδιότητα.

Έτσι, η ολοκλήρωση δεν είναι ευαίσθητη σε μικρές αλλαγές του σχήματος μιας συνάρτησης ενώ η παραγώγιση είναι.

Λόγω των παραπάνω, η αριθμητική παραγώγιση αποφεύγεται αν είναι δυνατόν.

ΣΥΝΑΡΤΗΣΕΙΣ ΠΟΛΛΩΝ ΜΕΤΑΒΛΗΤΩΝ

<u>Παραδείγματα:</u>

● syms x
syms y
g=x^2+x*y-6*y^3

>> diff(g, x, 2) %   ($\frac{\partial^2 g}{\partial x^2}$)
 ans = 2

Βρίσκουμε πρώτα την μερική παράγωγο ως προς χ ($\frac{\partial g}{\partial x}$):

z=diff(g, x)
z = 2*x+y

Στη συνέχεια βρίσκουμε την 2^η παράγωγο, ως προς ψ ($\frac{\partial^2 g}{\partial y \partial x}$) :

>> diff(z,y)
 ans = 1

● syms x y
f = x*cos(x+y)-exp(x*y)
fx = diff(f,x)
f = x*cos(x+y)-exp(x*y)
 fx = cos(x+y)-x*sin(x+y)-y*exp(x*y)
int(f,y)
ans = x*sin(x+y)-1/x*exp(x*y)
subs(f,{x y},{0 1})
ans =   -1

Η ΕΝΤΟΛΗ SOLVE :

```
>> solve(g, x)
ans =
 (-1/2+1/2*(1+24*y)^(1/2))*y
 (-1/2-1/2*(1+24*y)^(1/2))*y
```

SOLVE Symbolic solution of algebraic equations.
    SOLVE('eqn1','eqn2',...,'eqnN')
    SOLVE('eqn1','eqn2',...,'eqnN','var1,var2,...,varN')
    SOLVE('eqn1','eqn2',...,'eqnN','var1','var2',...'varN')

Τα eqns είναι symbolic εκφράσεις εξισώσεων.
Τα vars είναι symbolic μεταβλητές.
Η SOLVE βρίσκει τις λύσεις των εξισώσεων.

Υπάρχουν 3 πιθανά είδη output.

Όταν έχουμε 1 εξίσωση και ένα output, παίρνουμε την λύση.
Για πολλές εξισώσεις και ίσο αριθμό output, τα αποτελέσματα δίνονται αντίστοιχα.
Για πολλές εξισώσεις και ένα output, παίρνουμε μια παράσταση που περιέχει τις λύσεις.

<u>Παραδείγματα:</u>

● solve('p*sin(x) = r') chooses 'x' as the unknown and returns

ans =   asin(r/p)

solve('5*sin(x) = 0.3')
ans = 0.60036058445278422499751163440541e-1

● [x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0') μας δίνει :

```
x =
[ 1]
[ 3]
```

```
y =
[    1]
[ -3/2]
```

● [a,u,v] = solve('a*u^2 + v^2','u - v = 1','a^2 - 5*a + 6')

Λύνει τις 3 εξισώσεις ως προς a, u, v.

```
a =
 2
 2
 3
 3

u =
 1/3+1/3*i*2^(1/2)
 1/3-1/3*i*2^(1/2)
 1/4+1/4*i*3^(1/2)
 1/4-1/4*i*3^(1/2)

v =
 -2/3+1/3*i*2^(1/2)
 -2/3-1/3*i*2^(1/2)
 -3/4+1/4*i*3^(1/2)
 -3/4-1/4*i*3^(1/2)
```

● S = solve('x + y = 1','x - 11*y = 5')

```
S =
    x: [1x1 sym]
    y: [1x1 sym]

>> S.x
ans = 4/3
>> S.y
ans = -1/3
```

● solve('x^2-2*x+1')

```
ans =
 1
 1
```

● syms x y

```
f = x - 3/y
g = 3*y+x-7
[a b] = solve(f,g)
 f = x-3/y
 g = 3*y+x-7
 a =
 7/2+1/2*13^(1/2)
 7/2-1/2*13^(1/2)
```

b =
7/6-1/6*13^(1/2)
7/6+1/6*13^(1/2)

```
>> h = cos(x+y)-2^x+3
[a b]=solve(f,h)
 h =
 cos(x+y)-2^x+3
```

Warning: solutions may have been lost

a = 1.1348617854768247235753168714563
b = 2.6434937173776770454822567397744

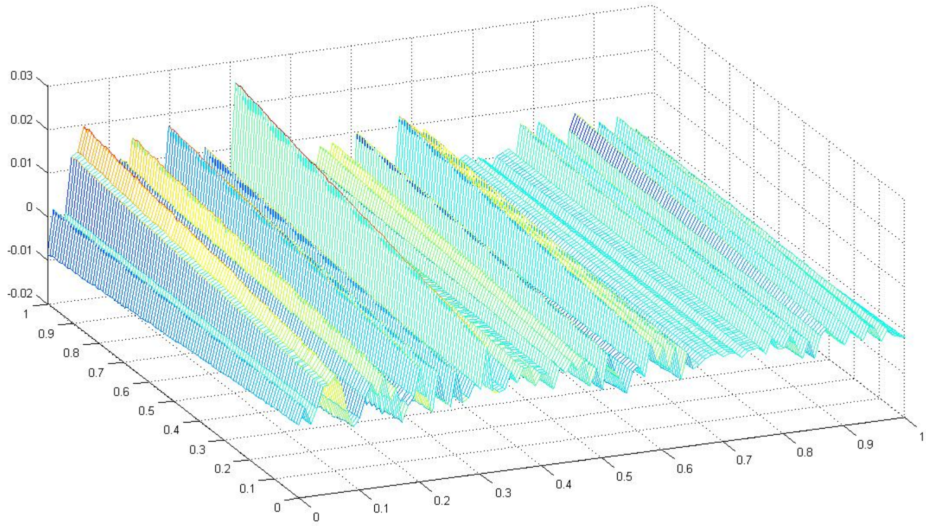
****

Στη συνέχεια, σαν εφαρμογή της θεωρίας του μαθήματος Μαθ. Λογισμός II, θα δούμε το εξής ενδιαφέρον παράδειγμα:

Παράδειγμα μιας συνάρτησης που δεν είναι φραγμένη σε μια περιοχή του (0,0), και επομένως δεν είναι ολοκληρώσιμη:

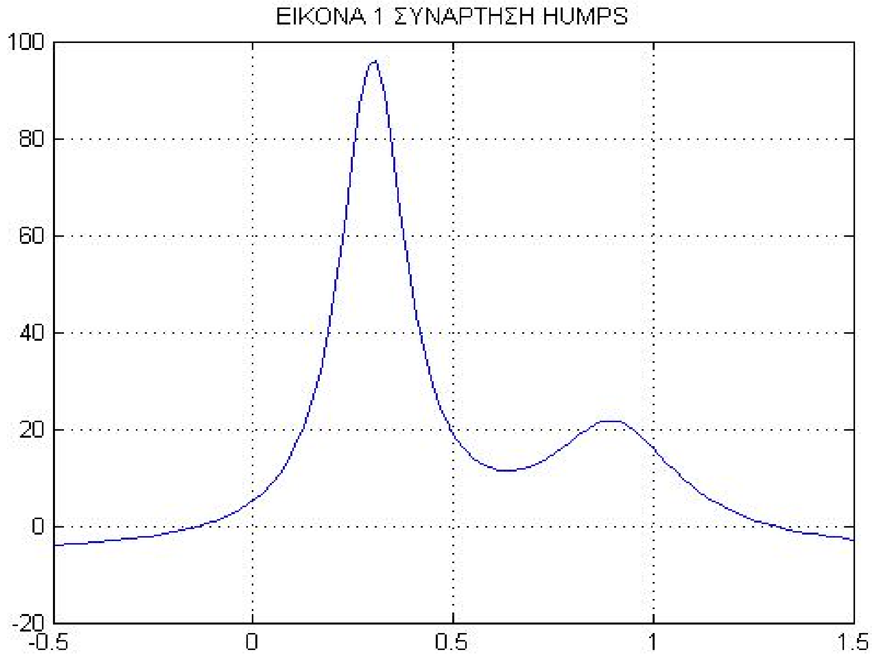
$$f(x,y)=\begin{cases}\dfrac{x-y}{(x+y)^3}, & xy\neq 0\\ 0, & xy=0\end{cases}$$

```
>> x=0:0.01:1;
>> y=0:0.01:1;
>> [X,Y]=meshgrid(x,y);
>> Z=[(X-Y)/(X+Y)^3];
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.399228e-020.
>> mesh(X,Y,Z)
```



<u>Σημεία μηδενισμού Συνάρτησης</u>

```
x=linspace(-0.5,1.5);
>> y=humps(x);   % καμπούρες: humps
>> plot(x,y)
>> grid on
>> title('ΕΙΚΟΝΑ 1 ΣΥΝΑΡΤΗΣΗ HUMPS')
```



Αυτή είναι μια κλασσική συνάρτηση του Matlab, με τύπο:

$$h(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

Οι ρίζες της είναι κοντά στις τιμές χ = -0.2 και χ = 1.3

Η συνάρτηση fzero υπολογίζει αριθμητικά τις τιμές αυτές:

```
>> format long
>> H_humps =@humps;
>> x=fzero(H_humps, 1.3)
x =   1.299549682584822
```

Για την άλλη τιμή, κοντά στο -0.2 γράφουμε:

```
>> [x,value]=fzero(H_humps,-0.2)
x =  -0.131618018099606
value =   8.881784197001252e-016
```

Εναλλακτικά, μπορούμε να δώσουμε διάστημα στο οποίο να ψάξει η εντολή fzero για λύσεις (Θεώρημα Bolzano)

```
>> [x,value]=fzero(H_humps,[-2,0])
x =  -0.131618018099606
value =    0
```

Βελτιστοποίηση- Ακρότατα.

Για να βρούμε το ελάχιστο χρησιμοποιούμε την εντολή fminbnd

Για να βρούμε το μέγιστο, βρίσκουμε το ελάχιστο της - f(x).

Παράδειγμα 1

Θα συνεχίσουμε με τα ακρότατα του προηγούμενου παραδείγματος:

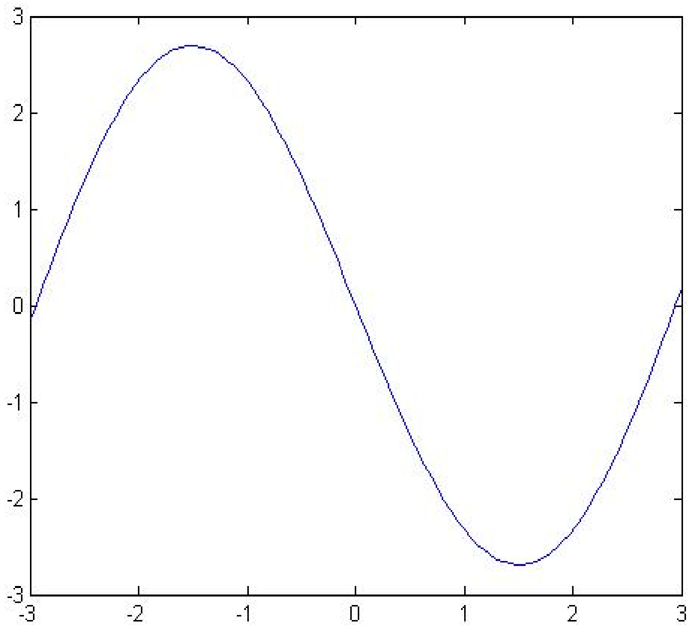
Θα βρώ για παράδειγμα ένα ακρότατο που βλέπουμε ότι υπάρχει στο διάστημα (0.5, 0.8)

```
>> [xmin,value]=fminbnd(H_humps,0.5,0.8)
xmin =   0.637008211963619
value =  11.252754125877694
```

Παράδειγμα 2

Ορίζω μια άλλη συνάρτηση, (Zsin) για να βρώ το μέγιστο της:

```
>> Zsin=@(x) 0.2*x-3*sin(x);
>> x=linspace(-3,3);
>> plot(x,Zsin(x))
```



Όπως βλέπουμε, παρουσιάζει μέγιστο στο διάστημα (-2, -1).

Ορίζω την αντίθετη της (Xsin) και βρίσκω το ελάχιστο της:

>> Xsin=@(x) -0.2*x+3*sin(x);

>> [xmin,value]=fminbnd(Xsin,-2,-1)

xmin = -1.504079968461076

value = -2.692509873742153

Επομένως, το μέγιστο της Xsin ισούται με 2.692509873742153.

Βελτιστοποίηση- Ακρότατα σε περισσότερες διαστάσεις

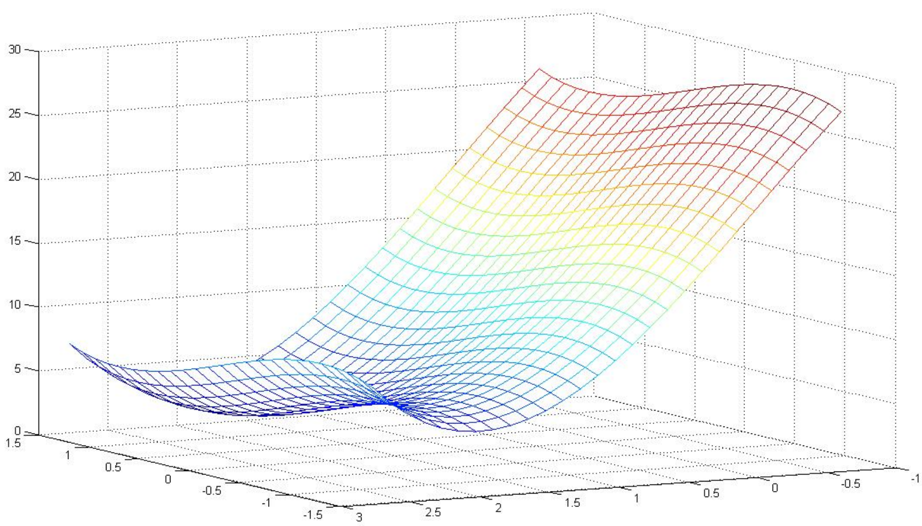
Σαν πρώτο παράδειγμα, θα σχεδιάσουμε την επιφάνεια

$z = f(x,y) = x^3 + y^3 - 3x - 12y + 20$ (Tutorial 9, άσκηση 6)

Τα κρίσιμα σημεία της είναι τα Α(1,2), Β(1,-2), Γ(-1, -2), Δ(-1,2).

Τα σημεία Α και Γ είναι ελάχιστο και μέγιστο αντίστοιχα, τα Β και Δ σαγματικά σημεία.

```
X=[-1.5:0.125:1.5];
y=[-2.5:0.125:2.5];
[X,Y]=meshgrid(x,y);
Z=X.^3 +Y.^3 -3*X -12*Y +20;
mesh(X,Y,Z)
```



Στη συνέχεια, αλλάζοντας τους συντελεστές των χ και ψ, βλέπουμε πως αλλάζει η επιφάνεια και τα ακρότατα της.

Την ελαχιστοποίηση συναρτήσεων πολλών μεταβλητών την κάνει η συνάρτηση fminsearch.

Δηλαδή, η εντολή αυτή προσπαθεί να βρεί το ελάχιστο μιας συνάρτησης f(x,y,z).

Ο αλγόριθμος αυτός δεν είναι τοσο αποδοτικός σε λείες συναρτήσεις, αλλά από την άλλη πλευρά δεν απαιτεί πληροφορίες για την κλίση, ο υπολογισμός των οποίων κοστίζει πολύ.

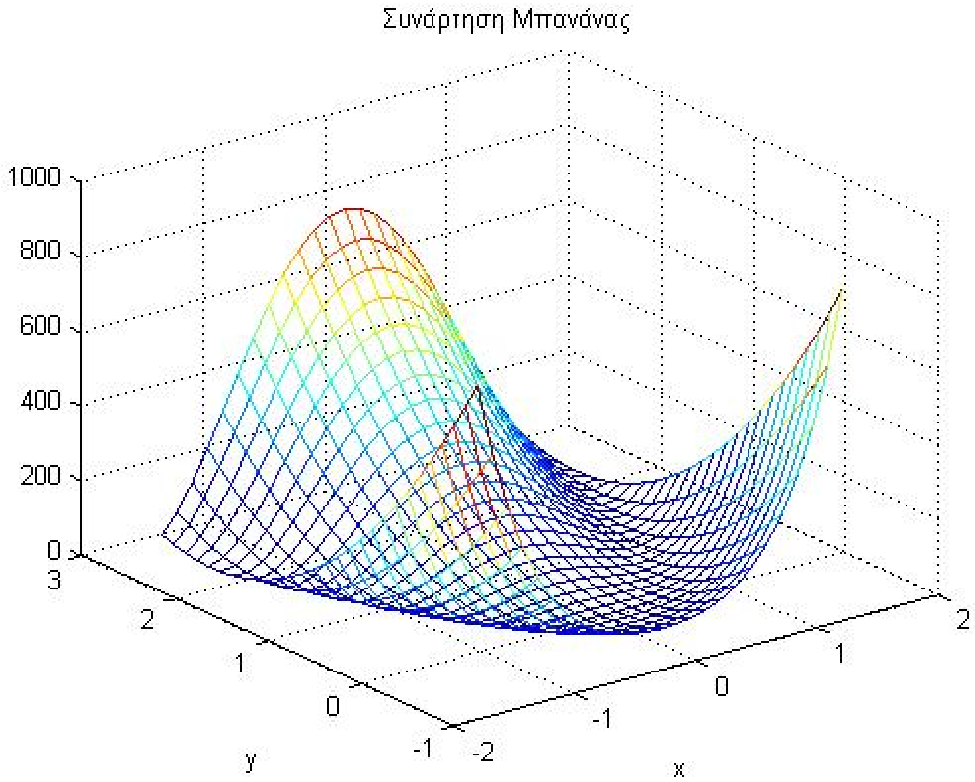
<u>Παράδειγμα:</u>

Η "συνάρτηση μπανάνα" του Rosenbrock:

$$f(x,y) = 100(y - x^2)^2 + (1 - x)^2$$

Ας κάνουμε πρώτα την γραφικά παράσταση:

```
>> x=[-1.5:0.125:1.5];
>> y=[-0.6:0.125:2.8];
>> [X,Y]=meshgrid(x,y);
(κάνναβος όλων των χ, ψ)
>> Z=100*(Y-X.^2).^2+(1-X).^2;
>> mesh(X,Y,Z)
>> hidden off
(για να μην κρύβει η επιφάνεια τα πίσω σημεία)
>> xlabel('x')
>> ylabel('y')
>> title{'Συνάρτηση Μπανάνας')
>> title('Συνάρτηση Μπανάνας')
```



Η συνάρτηση αυτή έχει ελάχιστο στο σημείο Α(1,1,0).

Θα χρησιμοποιήσουμε τώρα την εντολή fminsearch:

```
AH_banaan=@(x) 100*(x(2)-x(1)^2)^2+(1-x(1))^2;
>> [xmin,value,flag,output]=fminsearch(AH_banaan,[-1.9,2])
xmin =    1.0000    1.0000
value =   4.0686e-010
flag =     1
output =
    iterations: 114
     funcCount: 210
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x196 char]
```

<u>όπου:</u> xmin = η τιμή των χ,ψ

value = η τιμή της συνάρτησης στο σημείο αυτό

flag = σημαία (1 = επιτυχία)

iterations: επαναλήψεις

funcCount: πόσοι υπολογισμοί της συνάρτησης χρειάστηκαν

Επιλογές της συνάρτησης fminsearch:

Options=optimset('ονομα', τιμή, 'ονομα', τιμή,…)

ΟΝΟΜΑ ΕΠΙΛΟΓΗΣ	ΠΕΡΙΓΡΑΦΗ	ΑΡΧΙΚΗ ΤΙΜΗ
Display	Εμφάνιση συχνότητας Iter,final,notify,off	Notify, αν δεν βρεθεί λύση
MaxFunEvals	Μέγιστοι υπολογισμοί συνάρτησης	200*length(x)
MaxIter	Μέγιστες επαναλήψεις αλγορίθμου	200*length(x)
TolFun	Ανοχή λύσης συνάρτησης	1.e^-4
TolX	Ανοχή λύσης μεταβλητής	1.e^-4

Για να φανεί η χρήση των επιλογών, θα βρούμε την λύση στο παραπάνω πρόβλημα με πιο μικρές ανοχές συναρτήσεων και μεταβλητών:

```
>> options=optimset('TolFun',1e-8,'TolX',1e-8);
>> [xmin,value,flag,output]=fminsearch(AH_banaan,[-1.9,2],options)
xmin =    1.0000    1.0000
value =  6.1539e-018
flag =     1
output =
    iterations: 144
     funcCount: 266
     algorithm: 'Nelder-Mead simplex direct search'
```

Ακρότατα υπο συνθήκη: Πολλαπλασιαστές Lagrange

Να βρεθούν τα ακρότατα της συνάρτησης $f(x,y)=x^3+2y^3$ υπο τη συνθήκη $x^2+y^2=1$.

Θα χρησιμοποιήσουμε το συμβολικό πακέτο, για να υπολογίσουμε τα σημεία μηδενισμού της συνάρτησης του Lagrange. $F(x,y,\lambda)=x^3+2y^3+\lambda(x^2+y^2-1)$

Άγνωστοι θα είναι οι χ, y, λ.

```
>> syms x y lambda
eq1=3*x^2-2*x*lambda
eq2=6*y^2-2*y*lambda
g=x.^2+y.^2-1
[lambda x y]=solve(eq1,eq2,g);
[lambda x y]

eq1 =
3*x^2-2*x*lambda

eq2 =

6*y^2-2*y*lambda

 g =

x^2+y^2-1

ans =

[            3,            0,            1]
[           -3,            0,           -1]
[          3/2,            1,            0]
[         -3/2,           -1,            0]
[  3/5*5^(1/2),  2/5*5^(1/2),  1/5*5^(1/2)]
[ -3/5*5^(1/2), -2/5*5^(1/2), -1/5*5^(1/2)]
```

Στη συνέχεια, για να δούμε και γραφικά την ερμηνεία των ακροτάτων υπο συνθήκη, θα χρησιμοποιήσουμε τον αλγόριθμο lagrange, τον οποίο θα βρείτε αποθηκευμένο στο server του εργαστηρίου.

```
>> f= inline('x.^3+2*y.^3', 'x','y')
g= inline('x.^2+y.^2-1', 'x', 'y')
corners = [-1.5,1.5,-1.5,1.5]
lagrange (f,g,corners)
f =
    Inline function:
    f(x,y) = x.^3+2*y.^3

g =
    Inline function:
    g(x,y) = x.^2+y.^2-1

corners =

   -1.5000    1.5000   -1.5000    1.5000

   x          y         f(x,y)
ans =
   -0.0307    1.0044    2.0264
```

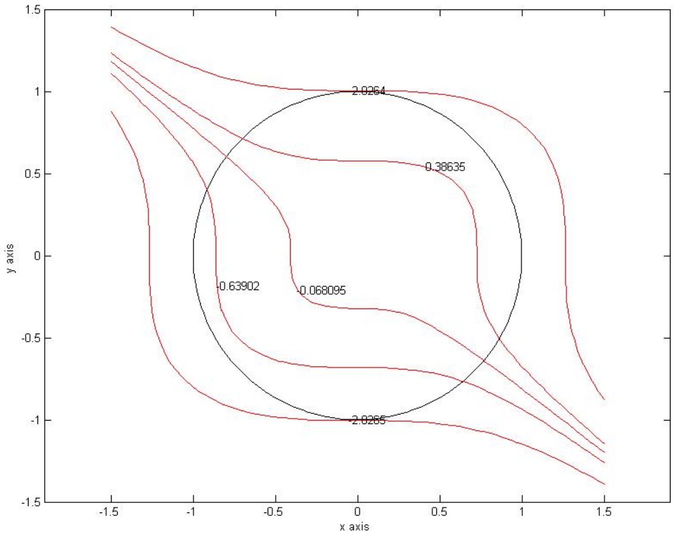
ans =

   -0.0482   -1.0044   -2.0265
ans =

   -0.3640   -0.2149   -0.0681
ans =

    0.4167   0.5395   0.3863
ans =

   -0.8553   -0.1886   -0.6390



Στη συνέχεια βλέπουμε το matlab code της συνάρτησης lagrange, αν κάποιος θέλει να την τροποποιήσει κλπ να κάνει copy- paste στον editor του Matlab τα παρακάτω:

```
%  Function mfile,  lagrange
%
%    Given a function f(x,y) in an mfile or as an inline function.
%    Given a constraint function g(x,y) in the same way.
%    User provides a vector corners = [a b c d] which define the
%    corners of a rectangle.
%
%                  (a,d)-----------(b,d)
%                    |               |
%                    |               |
%                    |               |
%                  (a,c)-----------(b,c)
%
%   The rectangle should be chosen to include the level curve
%   g(x,y) = 0.
%         The call is lagrange(f,g,corners) for f and g given as
inline
%   functions, and lagrange('f', 'g', corners)  for f and g given in
%   mfiles.
%          User then clicks on points on this zero level curve
%   of g.  The level curve of f through point is displayed along with
%   the value of f on this curve. This can be done 5 times.
```

```
function out = lagrange(f,g,corners)

a = corners(1);
b = corners(2);
c = corners(3);
d = corners(4);

x = linspace(a,b,50);
y = linspace(c,d,50);

[X,Y] = meshgrid(x,y);
Z = feval(f,X,Y);
W = feval(g,X,Y);
disp(' ')
disp('  x           y          f(x,y)   ')
disp('   ')

contour(X,Y,W, [0 0], 'k')
axis equal
hold on

for n = 1:5
   [p,q] = ginput(1);
   z0 = feval(f, p,q);
   contour(X,Y,Z, [z0 z0], 'r')
   z1 = num2str(z0);
   text(p,q, z1)
   [p,q,z0]
end
 xlabel('x axis ')
 ylabel('y axis ')

hold off
```

<u>ΒΙΒΛΙΟΓΡΑΦΙΑ:</u>

1) http://www.mathworks.com/matlabcentral/

2) Μάθετε το Matlab 7, D. Hanselman- B. Littlefield. εκδόσεις Κλειδάριθμος 2005