Εθνικό **Μ**ετσόβιο **Π**ολυτεχνείο
Σχολή **Ε**φαρμοσμένων **Μ**αθηματικών και **Φ**υσικών **Ε**πιστημών

Όνομα : Κάραλης Νικόλας
Α/Μ: 09104042

Αριθμητική Ανάλυση Ι
Εργασία 1^{η} - Μέθοδος Müller

Τμήμα :
Τρίτη 16.00 – 17.30

Ημερομηνία Παράδοσης : 5/5/2006

Αρχικά, η εξίσωση με την οποία θα ασχοληθούμε είναι η $x^4+2x^3+\left(-3a-\frac{5}{4}\right)x^2+2a^2x-a^4-3a^3-5\frac{a^2}{4}=0$. Αλλά επειδή ο αριθμός μητρώου μου όπως φαίνεται και από το εξώφυλλο λήγει σε 2 και ο αλγόριθμος για την άσκηση αυτή είναι α= AM + 1, έχουμε α=3. Άρα, η εξίσωση γίνεται

$$x^4+2x^3+\left(-9-\frac{5}{4}\right)x^2+18x-81-81-5\frac{9}{4}=0 \Leftrightarrow x^4+2x^3-\frac{41}{4}x^2+18x-173-\frac{1}{4}=0$$

Ερώτημα 1ο

Για να κατασκευάσουμε τη γραφική παράσταση της συνάρτησης που επιθυμούμε, δημιουργούμε δύο m-files, ένα με τον ορισμό της συνάρτησης και ένα με τη μέθοδο δημιουργίας της γραφικής παράστασης. Τα περιεχόμενα των αρχείων αυτών φαίνονται παρακάτω :

eq1.m

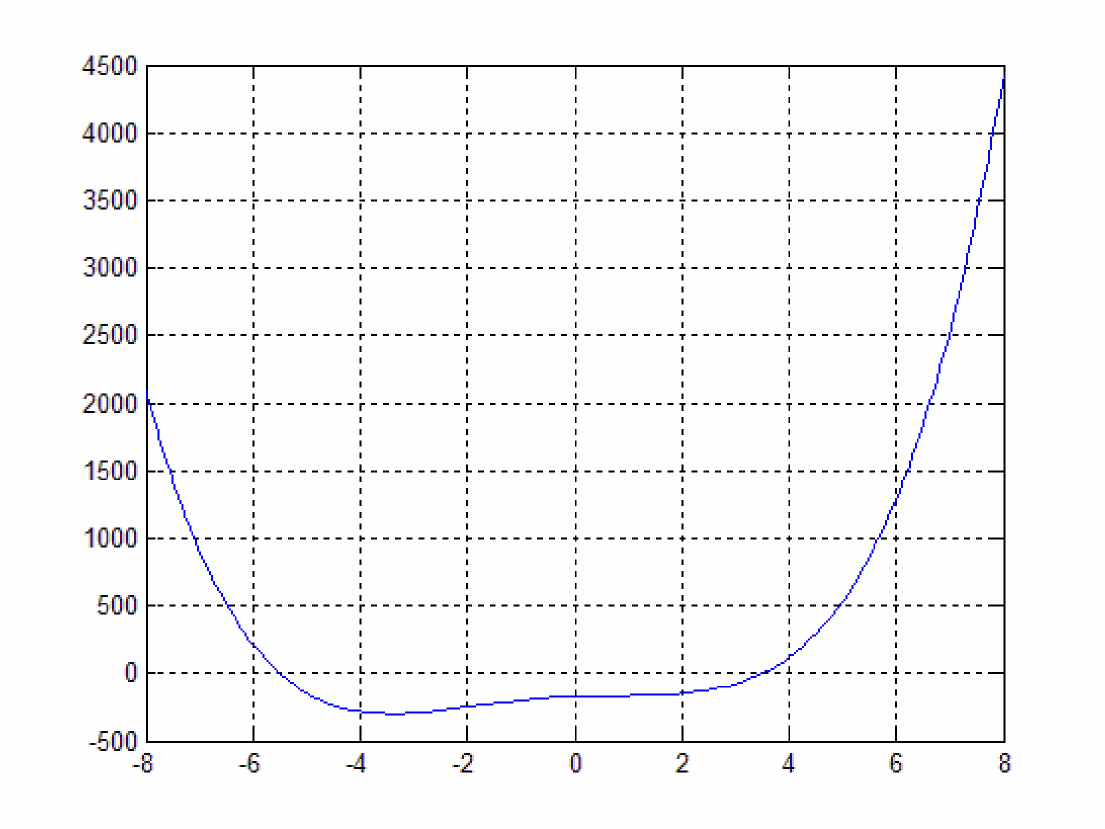
```
function y=eq1(x);
y=x.^4+2*x^3-(41/4)*x.^2+18*x-173-(1/4);
```

plot1.m

```
fplot('eq1',[-10 10]);
grid on;
```

Εκτελώντας τώρα την εντολή : >> plot1 παράγεται η παρακάτω γραφική παράσταση (αφού επιλέξουμε ως βήμα για τις κάθετες ευθείες να είναι το 0.5) :

Έπειτα από παρατήρηση αυτής της καμπύλης, καταλήγουμε στο συμπέρασμα ότι υπάρχουν ρίζες της συνάρτησης αυτής στα διαστήματα (-6,-5) και (3,4).



Ερώτημα 2°

Εκτελώντας την παρακάτω ακολουθία εντολών προσδιορίζουμε τις πραγματικές ρίζες στα διαστήματα που αναφέρθηκαν παραπάνω με χρήση των δύο μεθόδων, Secant και Muller.

```
>> format long
>> [x,l,iter]=secant('eq1',-6,-5,1e-7)
>> [x,l,iter]=secant('eq1',3,4,1e-7)
>> [x,l,iter]=muller('eq1',3,4,3.7,1e-7)
>> [x,l,iter]=muller('eq1',-6,-5,-5.1,1e-7)
```

Η ρίζες που προκύπτουν είναι οι x=-5.5 και η x=3.5 .
Η ανάλυση της σύγκλισης των μεθόδων ακολουθεί στο επόμενο ερώτημα.

Ερώτημα 3°

	Secant	**Muller**
(-6,-5)	-5.49999999999999	-5.50000000000000
# επαναλήψεων	6	4
(3,4)	3.50000000000002	3.50000000000000
# επαναλήψεων	6	4

Στον παραπάνω πίνακα βλέπουμε την προσέγγιση της κάθε μίας από τις ρίζες με τις δύο μεθόδους και τον αριθμό των επαναλήψεων που χρειάστηκε κάθε μία από τις μεθόδους.

Η μέθοδος Secant χρειάστηκε 6 επαναλήψεις για την εύρεση κάθε μιας εκ των 2 πραγματικών ριζών ενώ η μέθοδος Muller χρειάστηκε 4 επαναλήψεις για την εύρεση κάθε μιας εκ των δύο αυτών ριζών. Το γεγονός αυτό αποδεικνύει ότι η τάξη σύγκλισης της μεθόδου Muller είναι μεγαλύτερη αυτής της μεθόδου Secant.

Ερώτημα 4ο

Για την προσέγγιση του ζεύγους των μιγαδικών ριζών, εκτελούμε την παρακάτω ακολουθία εντολών :

```
>> [x,l,iter]=secant('eq1',-100,100,1e-7)
>> [x,l,iter]=muller('eq1',-100,100,20,1e-7)
```

Η προσέγγιση της κάθε μεθόδου καθώς και ο αριθμός των επαναλήψεων που χρειάστηκαν φαίνονται στον παρακάτω πίνακα.

	Secant	**Muller**
(-100,100)	3.5000	0.0000 - 3.0000i
# επαναλήψεων	25	16

Παρατηρούμε ότι η μέθοδος Secant χρειάστηκε 25 επαναλήψεις για να βρει μια πραγματική ρίζα την οποία μάλιστα είχε βρει νωρίτερα σε ένα καλύτερα προσδιορισμένο διάστημα σε μόλις 6 επαναλήψεις, ενώ η μέθοδος Muller χρειάστηκε 16 επαναλήψεις για να βρει μια μιγαδική ρίζα στο ίδιο διάστημα.

Παρακάτω παρατίθεται ο κώδικας των μεθόδων Secant και Muller που χρησιμοποιήθηκε για την προσέγγιση των ριζών, καθώς και το output των μεθόδων αυτών.

Muller.m

```
function [x,xlist,iter] = muler(f,x0,x1,x2,tol,maxit)
if nargin < 4,
fprintf('You need to provide more options \n');
return;
end;
if nargin < 5,tol = eps; end;
if nargin < 6, maxit = 50; end;
f0 = feval(f,x0);
f1 = feval(f,x1);
iter = 0;
xdiff = inf;
xlist=[x0;x1;x2];
while xdiff >= tol
f2 = feval(f,x2);
c0=(f1-f0)/(x1-x0);
c1=(f2-f1)/(x2-x1);
d0=(c1-c0)/(x2-x0);
s=c1+d0*(x2-x1);
```

```
dd=sqrt(s^2-4*f2*d0);
if (abs(s-dd)<abs(s+dd)),
ss=1;
else
ss=-1;
end;
x = x2 - 2*f2/(s+ss*dd);
xdiff = abs(x-x2)/abs(x);
xlist=[xlist;x];
iter = iter + 1;
if iter >= maxit
disp('Not converged after '+ maxit+' iterations.');
return;
end
x0=x1;f0=f1;
x1=x2;f1=f2;
x2=x;
end
```

Secant.m

```
function [x,xlist,iter] = secant(f,x0,x1,tol,maxit)
if nargin < 3,
fprintf('You need to provide more options \n');
return;
end;
if nargin < 4,tol = eps; end;
if nargin < 5, maxit = 50; end;
f0 = feval(f,x0);
f1 = feval(f,x1);
iter = 0;
xdiff = inf;
xlist=[x0;x1];
x=x1;
while xdiff >= tol
xold=x;
x = x - f1*(x1-x0)/(f1-f0);
xdiff = abs(x-xold)/abs(x);
xlist=[xlist;x];
iter = iter + 1;
if iter >= maxit
disp('Not converged after '+ maxit+' iterations.');
return;
end
x0=x1;
f0=f1;
x1=x;
f1 = feval(f,x);
end
```

Output

```
>>format long

>> [x,l,iter]=secant('eq1',-6,-5,1e-7)

x =

  -5.49999999999999


l =

  -6.00000000000000
  -5.00000000000000
  -5.40334961618981
  -5.52210985631559
  -5.49914817106744
  -5.49999266759308
  -5.50000000244497
  -5.49999999999999


iter =

     6

>> [x,l,iter]=secant('eq1',3,4,1e-7)

x =

   3.50000000000002


l =

   3.00000000000000
   4.00000000000000
   3.39180537772087
   3.47844353586566
   3.50106169368709
   3.49998986610387
   3.49999999526162
   3.50000000000002


iter =

     6
```

```
>> [x,l,iter]=muller('eq1',-6,-5,-5.1,1e-7)

x =

  -5.50000000000000


l =

  -6.00000000000000
  -5.00000000000000
  -5.10000000000000
  -5.49451799961702
  -5.50005992743326
  -5.49999999270995
  -5.50000000000000


iter =

     4

>> [x,l,iter]=muller('eq1',3,4,3.7,1e-7)

x =

   3.50000000000000


l =

   3.00000000000000
   4.00000000000000
   3.70000000000000
   3.50414082240191
   3.49996350467965
   3.50000000256099
   3.50000000000000


iter =

     4

>> [x,l,iter]=muller('eq1',-100,100,20,1e-7)

x =

   0.0000 - 3.0000i
```

```
l =

  1.0e+002 *

  -1.0000
   1.0000
   0.2000
   0.1960
   0.1873 - 0.0309i
   0.1261 - 0.0541i
   0.0982 - 0.0616i
   0.0713 - 0.0651i
   0.0464 - 0.0631i
   0.0278 - 0.0581i
   0.0132 - 0.0516i
   0.0027 - 0.0442i
  -0.0033 - 0.0366i
  -0.0036 - 0.0284i
   0.0002 - 0.0303i
   0.0000 - 0.0300i
   0.0000 - 0.0300i
   0.0000 - 0.0300i
   0.0000 - 0.0300i


iter =

    16

>> [x,l,iter]=secant('eq1',-100,100,1e-7)

x =

    3.5000


l =

  1.0e+003 *

   -0.1000
    0.1000
   -4.9904
    0.1000
    0.1000
    0.0749
    0.0633
    0.0511
    0.0420
```

```
    0.0343
    0.0280
    0.0229
    0.0187
    0.0153
    0.0125
    0.0102
    0.0084
    0.0069
    0.0057
    0.0048
    0.0041
    0.0037
    0.0036
    0.0035
    0.0035
    0.0035
    0.0035


iter =

    25
```