ΑΡΙΘΜΗΤΙΚΗ ΑΝΑΛΥΣΗ:1Η ΕΡΓΑΣΙΑ

ΑΝΤΩΝΗΣ ΑΝΔΡΑΣ
ΑΕΜ:2557

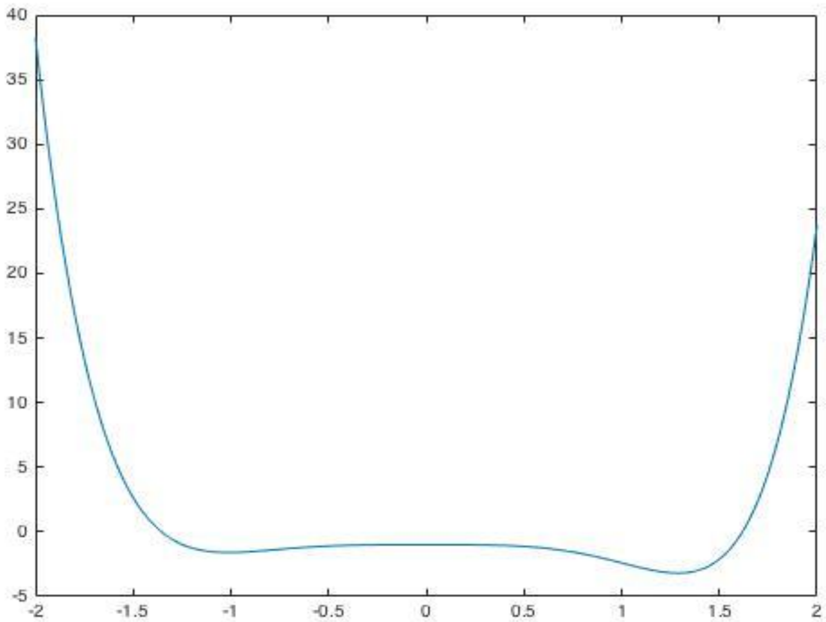
1Η ΑΣΚΗΣΗ

Έχουμε την συνάρτηση $f(x) = e^{\sin^3 x} + x^6 - 2x^4 - x^3 - 1$ στο διάστημα $[-2, 2]$.

- Η plot της f.

```
Command Window
>> x=[-2:0.1:2];
>> y=exp((sin(x).^3))+x.^6-2*(x.^4)-(x.^3)-1;
>> plot(x,y)
fx >>
```

- Η γραφική παρασταση της f



Αρχικά προκειμένου να μην ξαναγράφω τη συνάρτυση με τις παραγώγους της έκανα ένα initialization script με βάση το παρακάτω βίντεο.
https://www.youtube.com/watch?v=NPd13u4i6fM

```
format long;

syms x;
f = exp((sin(x).^3))+x.^6-2*(x.^4)-(x.^3)-1;

f1diff=diff(f);
f2diff=diff(f1diff);

f=inline(char(f));

f1diff=inline(char(f1diff));
f2diff=inline(char(f2diff));
```

α) Μέθοδος διχοτόμησης

```
function b= biscection( func,min,max,tol )

loop=0;
b=0;

if func(min)*func(max)>0
    disp('ERROR');
else
    b=(min+max)/2;
    err=abs(func(b));

    while err>tol
        if func(min)*func(b) < 0
            max=b;
        else
         min=b;
        end
        b=(min+max)/2;
        err=abs(func(b));
        loop=loop+1;
    end
        str=['Numper of loops:',num2str(loop)];
        disp(str);
end
```

Σε αυτή τη μέθοδο προκειμένου να έχω το σωστό αποτέλεσμα χώρισα το[-2,2] σε υπόδιστήματα. Μετά από αρκετές δοκιμές κατέληξα στο [-2,-1],[-1,1] στο οποίο δεν βρήκα λύση και στο [1,2].

Για χ∈[-2,-1] έχουμε:

```
Command Window
>> ASK1
>> biscection(f,-2,-1,10^(-6))
Numper of loops:20

ans =

   -1.197623729705811

fx >>
```

Για χ∈[-1,1] έχουμε ERROR:

```
Command Window
>> ASK1
>> biscection(f,-1,1,10^(-6))
ERROR

ans =

     0

fx >>
```

Για χ∈[1,2] έχουμε:

```
Command Window
>> ASK1
>> biscection(f,1,2,10^(-6))
Numper of loops:21

ans =

   1.530133485794067

fx >>
```

β) Newton-Raphson

```matlab
function x = newton_rap ( f,f1diff,f2diff,x0,x1,tol )

if f(x0)*f(x1) > 0
    disp('ERROR');
else

    loop=0;
    x=x0;
    err=Inf;

    if f(x1)*f2diff(x1)>0
        x=x1;
    end
    while abs(err)>tol
        xold=x;
        x = xold-f(xold)./f1diff(xold);
        err=x-xold;
        loop=loop+1;
    end
        str=['Numper of loops:',num2str(loop)];
        disp(str);
end
end
```

Για χ∈[-2,-1] έχουμε:

Command Window

```
>> ASK1
>> newton_rap(f,f1diff,f2diff,-2,-1,10^(-6))
Numper of loops:8

ans =

  -1.197623722133590

>>
```

Για χ∈[1,2] έχουμε:

Command Window

```
>> ASK1
>> newton_rap(f,f1diff,f2diff,1,2,10^(-6))
Numper of loops:7

ans =

    1.530133508166615

fx >>
```

γ) Secant/τέμνουσα με βοήθεια από το παρακάτω λινκ: http://math.stackexchange.com/questions/951445/help-with-secant-method-using-matlab

```
function x1 = secant( f,x0,x1,tol )

if f(x0)*f(x1) > 0
    disp('ERROR');
else
     loop=0;
     y1=f(x1);
     y0=f(x0);
     err=abs(y1);

     while err>tol
         tmp = x1;
         x1=x1-y1*(x1-x0)/(y1-y0);
         x0=tmp;
         y0=y1;
         y1=f(x1);
         loop=loop+1;
         err=abs(y1);
     end
end
      str=['Numper of loops:',num2str(loop)];
        disp(str);
end
```

Για χ∈[-2,-1] έχουμε:

```
Command Window
>> ASK1
>> secant(f,-2,-1,10^(-6))
Numper of loops:14

ans =

  -1.197623722469356

>>
```

Για χ∈[1,2] έχουμε:

```
Command Window
>> ASK1
>> secant(f,1,2,10^(-6))
Numper of loops:19

ans =

   0.022812572842858

>>
```

Καταληκτικά μπορόυμε να παρατηρήσουμε ότι η μέθοδος Newton-Rapson υπερβαίνει τις άλλες καθώς παράγει το ίδιο αποτέλεσμα με σχεδόν τις μισές επαναλήψεις από τις άλλες 2.

2^Η ΑΣΚΗΣΗ

Έχουμε την συνάρτηση $f(x) = 54x^6 + 45x^5 - 102x^4 - 69x^3 + 35x^2 + 16x - 4$ στο διάστημα $[-2,2]$.

Όπως στην άσκηση 1 έκανα ένα initialization script που μοιάζει πολύ με το 1^{o}.

```
format long;

syms x;
f = (54*x.^6)+(45*x.^5)-(102*x.^4)-(69*x.^3)+(35*x.^2)+16*x-4;

f1diff=diff(f);
f2diff=diff(f1diff);

f=inline(char(f));

f1diff=inline(char(f1diff));
f2diff=inline(char(f2diff));
```

1α) Νέα Μέθοδος διχοτόμησης,μοιάζει με την προηγούμενη

```
function p = new_bisection(f,x0,x1,tol)

if f(x0)*f(x1)>0
    disp('ERROR');
else
    loop = 0;
    p = (x0 + x1)/2;
    err = abs(f(p));

    while err > tol
        if f(x0)*f(p)<0
            x1 = p;
        else
            x0 = p;
        end

        p = x0 + (x1-x0) * rand();  %Random point!
        err = abs(f(p));
        loop = loop+1;

    end
    str = ['Num of loops:',num2str(loop)];
    disp(str);
end
```

Για $\chi \in [-2,-1]$ έχουμε:

```
Command Window
>> ASK2
>> new_bisection(f,-2,-1,10^(-6))
Num of loops:50

ans =

  -1.381298483242427

fx >>
```

Για χ∈[1,1] έχουμε:

```
Command Window
>> ASK2
>> new_bisection(f,-1,1,10^(-6))
ERROR
>>
```

Για χ∈[1,2] έχουμε:

```
Command Window
>> ASK2
>> new_bisection(f,1,2,10^(-6))
Num of loops:32

ans =

   1.176115559112139

>>
```

1β) Νέα Μέθοδος Newton-Raphson,μοιάζει με την προηγούμενη

```
function x = new_newton(f,f1diff,f2diff,x0,x1,tol)

if f(x0) * f(x1) > 0
    disp('ERROR');
else
    err = Inf;
    loop = 0;
    x = x0;
    if f(x1)*f2diff(x1)>0
        x = x1;
    end

    while abs(err) > tol
        x_old = x;
        x = x_old - f(x_old)./f1diff(x_old)-1/2*( ((f(x_old).^2)  *f2diff(x_old)) / (f1diff(x_old).^3) );
        err = x - x_old;
        loop = loop+1;
    end

  str = ['Num of loops:', num2str(loop)];
  disp(str);

 end
```

Για χ∈[-2,-1] έχουμε:

Command Window

```
>> ASK2
>> newton_rap(f,f1diff,f2diff,-2,-1,10^(-6))
Numper of loops:19

ans =

  -0.666667283561390

fx >>
```

Για χ∈[-1,1] έχουμε ERROR όπως και προηγουμένως.

Για χ∈[1,2] έχουμε:

Command Window

```
>> ASK2
>> newton_rap(f,f1diff,f2diff,1,2,10^(-6))
Numper of loops:8

ans =

   1.176115557355612

fx >>
```

1γ)Νέα μέθοδος σύγκλισης

```
function x1 = new_secant( f,x0,x1,x2,tol )

if f(x0)*f(x1) > 0
    disp('ERROR');
else
     loop=0;
     y2=f(x2);
     y1=f(x1);
     y0=f(x0);
     err=abs(y1);
    q=f(x0)/f(x1);
    r=f(x2)/f(x1);
    s=f(x2)/f(x0);

     while err>tol
         x2=x2-y2*(x2-x1)/(y2-y1);
         x1=x1-y1*(x1-x0)/(y1-y0);
         x0=x2-(r*(r-q)*(x2-x1)+(1-r)*s*(x2-x0));
         y0=y1;
         y1=y2;
         y1=f(x1);
         y2=f(x2)
         loop=loop+1;
         err=abs(y2);
     end
end
      str=['Numper of loops:',num2str(loop)];
        disp(str);
end
```

2)Εκτελόντας τον αλγόριθμο 10 φορές παρατηρούμε ότι το αποτέλεσμα του αριθμού των επαναλήψεων διαφέρει και έχει κάποια απόκλιση από 27 έως 36.

Τα αποτλέσματα που προέκυψαν είναι τα εξής:

27, 36, 35, 36, 36, 36, 29, 33, 33, 33

Βλέπουμε ότι υπάρχει απόκλιση αλλά παρατηρούμε ότι κολλάει σε κάποια νούμερα όπως το 33(3 φορές) και το 36(4 φορές).

3)Βάση των απότελεσματων θεωρώ πως οι τροποποιημένες μέθοδοι χρειάζονται περισσότερες επαναλήψεις προκείμενου να έχουμε το ίδιο αποτέλεσμα, επομένως είναι πιο χρονοβόρες και αναποτελεσματικές.

3Η ΑΣΚΗΣΗ

1)Ο παρακάτω κώδικας δέχεται ως είσοδο τον πίνακα A και παράγει τους L,U&P. Βοηθήθηκα αρκετά από το παρακάτω λινκ http://www.mathworks.com/matlabcentral/answers/1351-l-u-decomposition.

```
function [L, U, P] = decomposition(A)
[n n1] = size(A);
L=eye(n);
P=eye(n);
U=A;

for j = 1:n
  [pivot m] = max(abs(U(j:n, j)));
  m = m+j-1;
  if m ~= j
    U([m,j],:) =  U([j,m], :);
    P([m,j],:) =  P([j,m], :);
    if j >= 2;
      L([m,j],1:j-1) =  L([j,m], 1:j-1);
    end;
  end
  for i = j+1:n
    L(i, j) = U(i, j) / U(j, j);
    U(i, :) =  U(i, :) - L(i, j)*U(j, :);
  end
end
```

Αφού εκτελέσουμε τη decomposition με αυτόν τον αλγόριθμο υπολογίζουμε το x.

```
function x = returnx( L,U,P,b)
y=size(L,1);
[m,n]=size(L)
for i=1:m
    sum=0;
    for j=1:n
        sum=sum+P(i,j)*b(j);
    end
    for j=1:i-1
        sum=sum-y(j)*L(i,j);
    end
    sum=sum/L(i,i);
   y(i)=sum;
end
for i=1:m
    sum=0;
    for j=1:n
        sum=sum+y(j);
    end
    for j=1:i-1
        sum=sum-x(j)*U(i,j);
    end
    sum=sum/U(i,i);
   x(i)=sum;
end

end
```

2) Με βάση το βιβλίο και κάποια λινκ αυτός ο αλγόριθμος τριδιαγωνοποιεί τους πίνακες και επιστρέφει τα L & U.

```
function  [L,U] = trid( A )
[m,n]=size(A);
L=zeros(size(A));
U=zeros(size(A));
L(1,1)=A(1,1);
U(1,2)=A(1,2)/l(1,1);
for i=2:m-1
    U(i,i)=1;
    L(i,i)=A(i,i)-A(i,i-1)*U(i-1,i);
    L(i,i-1)=A(i,i-1);
    U(i-1,i)=A(i-1,i)/L(i,i-1);
end
L(m,m-1)=A(m,m-1);
L(m,m)=A(m,m)-L(m,m-1)*U(m-1,m);
end
```

Ενώ ο παρακάτω μας επιστρέφει το x.

```
function [ x ] = returnx2( L,U,f )
y=zeros(size(L),1);
x=zeros(size(L),1);
y(1)=f(1)/L(1,1);
for k=2:m
    y(k)=(f(k)-y(k-1))/L(k,k);
end
x(m)=y(m);
for k=1:n-1
    x(n-k)=y(n-k)-x(n-k+1)*U(n-k,n-k+1);
end

end
```