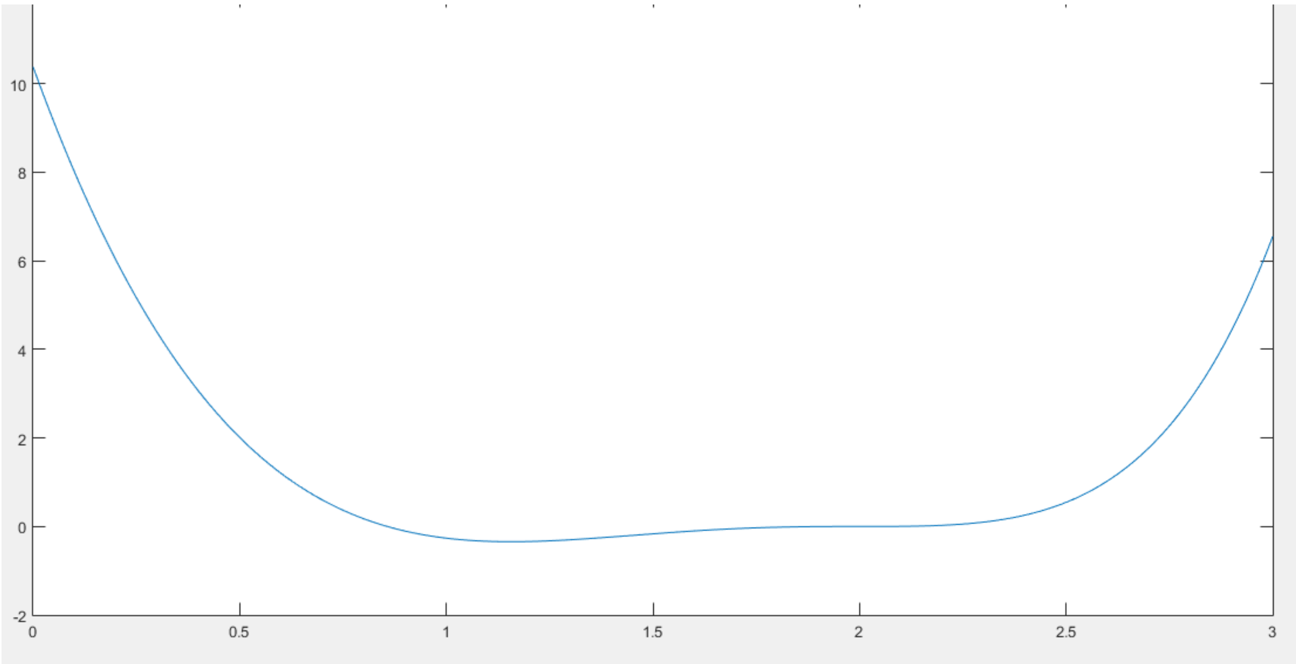
ΑΡΙΘΜΗΤΙΚΗ ΑΝΑΛΥΣΗ
1η ΕΡΓΑΣΙΑ

ονοματεπώνυμο: Παντελής Κυριακίδης
ΑΕΜ: 2551

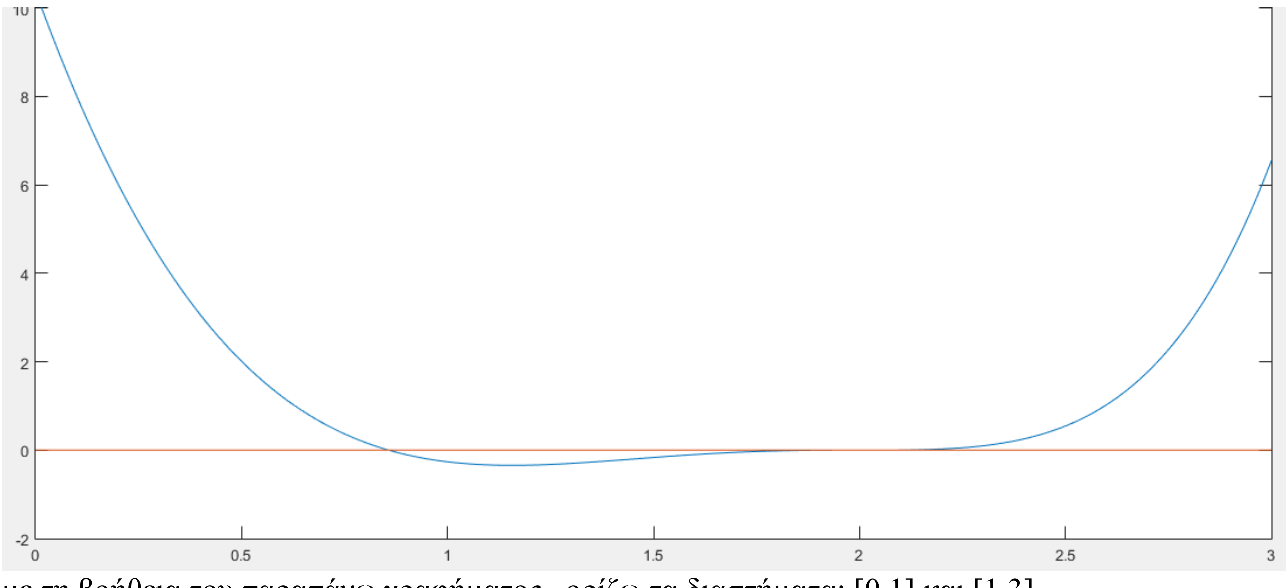
Ασκηση1:

Συνάρτηση: f(x) = 14*x*e^(x-2)−12*e^(x−2)−7*x^3 +20*x^2−26x+12 στο διάστημα [0,3].

Η γραφική παράσταση της συνάρτησης:



για να χωρίσω τα διαστήματα στα οποία βρίσκονται οι ρίζες της , κοιτάω περίπου σε ποια χ μηδενίζεται:



με τη βοήθεια του παραπάνω γραφήματος , ορίζω τα διαστήματα: [0,1] και [1,3].

α) Μέθοδος Διχοτόμησης:

κώδικας matlab:

```
function [m,steps]= dixotomisi( a,b )
%D Summary of this function goes here
if(f(a)*f(b)<0) %Bolzano
    steps=0; %step counter
    while(abs(b-a)>10^(-6)/2)
        m=(a+b)/2;
        if (f(m)==0)
            break;
        end
        if(f(m)*f(a)<0)
            b=m;
        else
            a=m;
        end
        steps=steps+1;
    end
else
    disp('no root')
end

end
```

<u>Αποτελέσματα:</u>

I=[0,1]

```
>> [m,steps]=dixotomisi(0,1)

m =

    0.8571


steps =

    21
```

<u>Παρατήρηση:</u> στο διάστημα [1,3] βρίσκει με την μία ρίζα της συνάρτησης οπότε ο αλγόριθμος σταματά στην 1η επανάληψη.

I=[1,3]

```
>> [m,steps]=dixotomisi(1,3)

m =

     2


steps =

     1
```

β)Μέθοδος Newton-Raphson:

κώδικας matlab:
(Αλλάζω μόνο το a και το b και το ξανατρέχω.)

```
%changing initial x value.
a=1; b=3; % [0,3]
x=b;
if(f(x)*ddf(x)<=0)
    x=a;
end

%loops
k=0;

lastx=x;
x=lastx-f(lastx)/df(lastx);
e=abs(x-lastx);
k=1;
while(e>(10^(-6)/2))
    lastx=x;
    x=lastx-f(lastx)/df(lastx);
    e=abs(x-lastx);
    k=k+1;
end
```

Αποτελέσματα: **I=[0,1]** **I=[0,1]**

```
>> x

x =

   0.857142857142856

>> k

k =

     7
```

```
>> x

x =

   2.000017497293807

>> k

k =

    29
```

γ)Μέθοδος Τέμνουσας:

κώδικας matlab:

```
% initial values
x0=0; x1=1;

%loops
k=0;

x=0;
e=abs(x1-x0);
while(e>(10^(-6)/2))
    if(k~=0)
        x0=x1; x1=x;
    end
    x=x1-(f(x1)*(x1-x0))/(f(x1)-f(x0));
    e=abs(x-x1);
    k=k+1;
end
```

Αποτελέσματα: **I=[0,1]** **I=[1,3]** **I=[1.5,3]**

```
>> x

x =

   0.857142857142854

>> k

k =

     8
```

```
>> x

x =

   0.857142857142860

>> k

k =

    23
```

```
>> x

x =

   1.999988860756149

>> k

k =

    38
```

Το τελευταίο διάστημα το πήρα γιατί δεν μου εμφανίστηκε η 2η ρίζα στο [1,3]

- **για χ=0.8571 :**
    - **διχοτομηση(21 επαναληψεις)**
    - **newton-raphson(7)**
    - **τεμνουσα(8)**
- **για χ=2:**
    - **διχοτόμηση(1 γιατί βρήκε τη ρίζα με την πρώτη επανάληψη)**
    - **newton-raphson(29)**
    - **τέμνουσα(38)**

τετραγωνική σύγκλιση: θα πρέπει η παράγωγος της (χ-f(x)/df(x)) να ειναι ιση με 0 για χ: ριζα. Συγκλίνουν τετραγωνικά οι απλές ρίζες. Αν η α είναι ρίζα πολλαπλότητας μεγαλύτερης ή ίσης του δύο, τότε η μέθοδος δεν συγκλίνει τετραγωνικά.
Στα παραδείγματα μας καμία ρίζα δεν συγκλίνει τετραγωνικά.

Άσκηση2:

1. a)

```
%changing initial x value.
a=0; b=1; % [0,3]
x=b;
if(f(x)*ddf(x)<=0)
    x=a;
end

%loops
k=0;

lastx=x;
x=lastx-f(lastx)/df(lastx)-(0.5*(f(lastx)^2)*ddf(lastx))/(df(lastx)^3);
e=abs(x-lastx);
k=1;
while(e>(10^(-6)/2))
    lastx=x;
    x=lastx-f(lastx)/df(lastx)-(0.5*(f(lastx)^2)*ddf(lastx))/(df(lastx)^3);
    e=abs(x-lastx);
    k=k+1;
end
```

newton raphson . Αποτελεσματα: **I=[0,1] και I=[1,3]** **I=[2, 2.6]**
με τροποποίηση

ίδια αποτελέσματα

```
x =

   1.047190244231963

>> k

k =

    23
```

```
x =

   2.300523983021863

>> k

k =

     4
```

β)
μέθοδος διχοτόμησης με τροποποίηση:

```
a=1; b=2; %I=[0,3]--> I1=[0,1] , I2=[1,2] , I3=[2,3]

if(f(a)*f(b)<0) %Bolzano
    steps=0; %step counter
    while(abs(b-a)>10^(-6)/2)
        m= rand * (b-a) + a ;
        steps=steps+1;
        if (f(m)==0)
            break;
        end
        if(f(m)*f(a)<0)
            b=m;
        else
            a=m;
        end
    end
else
    disp('no root')
end
```

Αποτελέσματα:

I=[0,1]

```
m =

   0.841068812376971

>> steps

steps =

    24
```

I=[1,2]

```
>> m

m =

   1.047201946810505

>> steps

steps =

    22
```

I=[2,3]

```
m =

   2.300523977822029

>> steps

steps =

    30
```

2. Καθε φορά που το τρέχω βγαίνουν διαφορετικές επαναλήψεις.

γ) τέμνουσα τροποποιημένη:

κώδικας:

```
% initial values
x0=0; x1=1; x2=0.5;

%loops
k=0;

x=0;
e=abs(x2-x1);
while(e>(10^(-6)/2))

    if(k~=0)
        x0=x1; x1=x2; x2=x;
    end

    q=f(x0)/f(x1); r=f(x2)/f(x1); s=f(x2)/f(x0);
    x=x2-(r*(r-q)*(x2-x1)+(1-r)*s*(x2-x0))/((q-1)*(r-1)*(s-1));

    e=abs(x-x2);
    k=k+1;

end
```

Αποτελέσματα:

x0=0 x1=1 x2=0.5

```
>> x

x =

   1.047193816333577

>> k

k =

    29
```

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

Άσκηση3:

α) L U decomposition

returning x

```
function [ L,U,P,x ] = ex3a(A,b)
%   exercise 3a)
[n n1] = size(A);
L=eye(n);
P=eye(n);
U=A;
%decomposition
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
%return x :
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
```

β)

Cholesky decomposition

```
function L = ex3b( M )
%EX3B Summary of this function goes here
%   Detailed explanation goes here
n = length( M );
L = zeros( n, n );
for i=1:n
    L(i, i) = sqrt(M(i, i) - L(i, :)*L(i, :)');
    for j=(i + 1):n
        L(j, i) = (M(j, i) - L(i,:)*L(j ,:)')/L(i, i);
    end
end

end
```

γ) Gauss-seidel

```
function X = ex3c( A,b )
%EX3C Summary of this function goes here
%   Detailed explanation goes here
n = length(b);
X = zeros(n,1);
Error_eval = ones(n,1);

%% Check if the matrix A is diagonally dominant
for i = 1:n
    j = 1:n;
    j(i) = [];
    B = abs(A(i,j));
    Check(i) = abs(A(i,i)) - sum(B); % Is the diagonal value greater than the remaining row values combined?
    if Check(i) < 0
        fprintf('The matrix is not strictly diagonally dominant at row %2i\n\n',i)
    end
end

%% Start the Iterative method

iteration = 0;
while max(Error_eval) > (10^(-4)/2)
    iteration = iteration + 1;
    Z = X;  % save current values to calculate error later
    for i = 1:n
        j = 1:n; % define an array of the coefficients' elements
        j(i) = [];  % eliminate the unknow's coefficient from the remaining coefficients
        Xtemp = X;  % copy the unknows to a new variable
        Xtemp(i) = [];  % eliminate the unknown under question from the set of values
        X(i) = (b(i) - sum(A(i,j) * Xtemp)) / A(i,i);
    end
    Xsolution(:,iteration) = X;
    Error_eval = sqrt((X - Z).^2);
end

GaussSeidelTable = [1:iteration;Xsolution]
MaTrIx = [A X b]
end
```