Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator Nr.1

Metode și modele de calcul

**Tema :** Rezolvarea numerică a ecuațiilor algebrice și transcendente

Varianta 3

A efectuat : st.gr.TI-214 Buza Cătălin

A verificat : asistent univ. Vadim Struna

Chișinău 2022

## Scopul lucrării:

* + Să se separe toate rădăcinile reale ale ecuaţiei f(x)=0 unde y=f(x)este o funcţie reală de variabilă reală.

## Să se determine o rădăcină reală a ecuaţiei date cu ajutorul metodei înjumătăţirii

intervalului cu o eroare mai mică decît =10-2.

## Să se aprecieze rădăcina obţinută cu exactitatea =10-6, utilizînd :

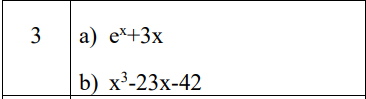
* + - Metoda aproximării succesive ;

## Metoda tangentelor(Newton);

* + - Metoda secantelor;

## Să se compare rezultatele luînd în consideraţie numărul de iteraţii , evaluările pentru funcţii şi derivată.

Sarcina problemei:

Să se găsească rădăcinile ecuaţiei : 

# Separarea rădăcinilor

## Pentru prima ecuație este convenabilă folosirea metodei grafice de separare a rădăcinilor.

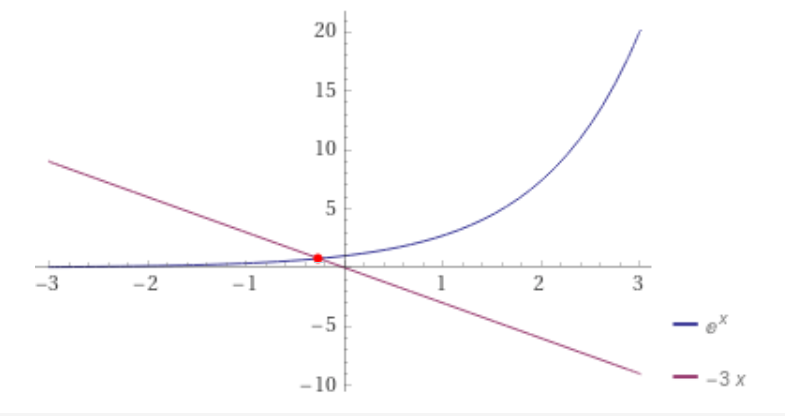
Scriem ecuația

-3*x* =*ex .*

sub forma *φ* ( *x* )=*g*(*x* )

## și obținem:

1. Pentru determinarea punctelor de intersecție a funcțiilor construim graficele :

*y*=*φ*( *x* ) și

*y*=*g* ( *x* )

## Astfel ecuația are o rădăcină reală ξ

pe interval (-0.4,-0.2).

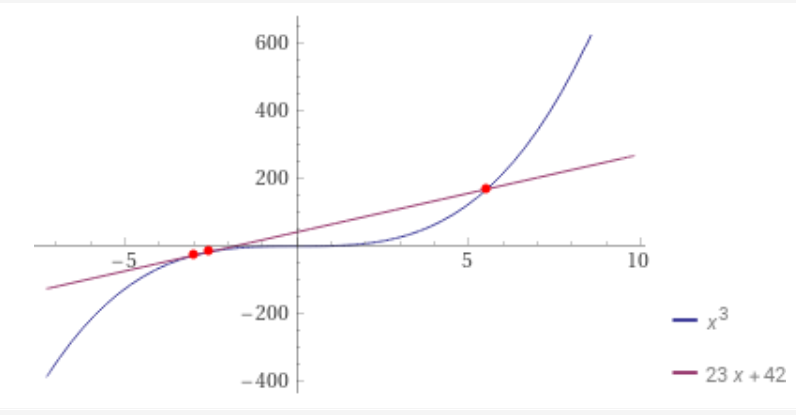
b)Pentru a doua ecuație folosim *metoda șirului lui Rolle.*

Derivata se anuleaza pentru

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | -2.9 | -2.77 | 2.77 | 2.9 |
| y | 0.31 | 0.45 | -84.45 | -84.311 |

Avem o alternanță de semn și respectiv o singură rădăcină reală ξ )

Pentru a determina celelalte rădăcini folosim metoda grafică.



## Astfel mai avem 2 rădăcini ξ ∈(−3.5,−2.8)(5,6)

## 2)Calculul rădăcinii reale prin metoda înjumătățirii intervalului;

#include<math.h>

using namespace std;

double f(double x){ //prima functie din varianta 3

return exp(x)+3\*x;

}

double f1(double x){ // a doua functie din varianta3

return pow(x,3)-23\*x-42;

}

/\*

voi folosi o functie pentru a calcula solutiile ambelor ecuatii cu metoda injumatatirii intervalelor

functia primeste ca parametrii valorile intervalului unde se afla solutia si un pointer la o functie care contine ecuatia ce trebuie rezolvata

\*/

void injumatatirea\_intervalului(double a,double b,double(\*func)(double))

{

int k=0; //variabila ce va stoca nr de iteratii ce sau efectuat pina la gasirea solutiei

double c = 0,eps = 0.0001;// c- solutia ,eps-eroare admisibila

while((b-a)>eps ) //atita timp cit se satsfice conditiia

{

k++;

c = a+(b-a)/2;//aflam mijlocu intervalului

if (func(c)==0)//daca valoare functiei in mijlocul intervalului ii egala solutia inseamna ca am gasit solutie

break;// programu se opreste

if (func(a)\*func(c)<0) b = c;//daca nu se verifica conditia daca e adevarata inseamna ca solutia se afla in intervalul (a,c) si repetam procedura cu b=c

else a = c;//daca nu se respecta conditia inseamna ca solutia se afla in intervalul (c,b) si repetam algoritmul cu a=c

}

cout <<"\nRadacina x=" << c<<endl;//afisam radacina gasita

cout<<"Numarul de iteratii: "<<k<<endl;//afisam nr de iterati efectuate pina la gasirea solutiei

}

int main()

{

injumatatirea\_intervalului(-0.4,-0.2,&f);//dam valorile intervalului a si b si adresa functiei unde se afla ecuatia spre rezolvare

cout<<"\nEcuatia 2:";

injumatatirea\_intervalului(-2.77,2.77,&f1);

injumatatirea\_intervalului(-3.5,-2.8,&f1);

injumatatirea\_intervalului(5,6,&f1);

return 0;

}

3) Calculul rădăcinii reale prin metoda aproximațiilor succesive

a) Pentru aplicarea metodei aproximațiilor succesive verificăm condiția de convergență. Scriind ecuația în forma x=φ( x ) obținem :

Prin urmare șirul converge.

1. Verificăm condiția de convergență :

=42-

(x)= ; ; x

double f3(double x){ //prima functie din varianta 3 scrisa sub forma x=fi(x)

return exp(x)/-3;

}

double f4(double x){ //a doua functie din varianta 3 scrisa sub forma x=fi(x)

return (pow(x,3)-42)/23;

}

void aproximatii\_succesive(double x0, double(\*func)(double))//x0 este o valoare initiala aleasa de noi ce apartine intervalului unde se afla solutia noastra

{

int k=0;//variabila ce ne va arata nr de iteratii efectuate pina la gasirea solutiei

double x1,eps=0.000001;//x1-variabila ce va stoca valoare functiei in punctul initial dat de noi; eps-eroarea

while(1) //folosim while pentru agasi solutia ecuatiei

{

x1=func(x0);//x1 ii atribuim valoarea functiei in punctul initial ales de noi ce apartine intervalului (a,b)

k++;//incrementam k

if(abs(x1-x0)<eps) //daca acesta conditie se indeplineste inseamna ca am gasit solutia

{

cout<<"Radacina x= "<<x0<<endl<<"Numarul de iteratii k="<<k<<endl;//afisam radacina si numarul de iteratii

break;//dupa ce am gasit solutia folosim break pentru a iesi din bucla while

}

x0=x1;//in caz contrar repetam algoritmul cu x0=x1;

}

}

int main()

{

cout<<"\nMetoda aproximatiilor succesive:"<<endl;

cout<<"Ecuatia 1:"<<endl;;

aproximatii\_succesive(-0.4,&f3);

cout<<"Ecuatia 2:"<<endl;

aproximatii\_succesive(-0.2,&f4);

return 0;

}

4) Calculul rădăcinii reale prin metoda tangentelor(Newton)

double f(double x){ //prima functie din varianta 3

return exp(x)+3\*x;

}

double f1(double x){ // a doua functie din varianta3

return pow(x,3)-23\*x-42;

}

double fderiv(double x){ //derivata functiei f de mai sus

return 3+exp(x);

}

double f1deriv(double x){ //derivata functiei f1 de mia sus

return 3\*pow(x,2)-23;

}

/\*x0=valoarea initiala a functiei alesa de noi arbitrar din intervalul unde se afla solutia cautata

func,func2 pointeri la functii prima e la la functia unde se afla ecuatia careia trebuie sai gasim solutie func2- derivata func

\*/

void metoda\_tangentelor(double x0,double (\*func) (double),double(\*func2)(double))

{

int k=0;//variabila care ne va arata de cite ori am folosit algorimul pina la gasirea solutie

double x1,eps=0.000001;//x1-punctul de pe axa ox obtinut la intersectarea tangentei prin x0 eps-exactitatea cu care sa obtinut solutia

while(1)//incepem algortmul

{

x1=x0-func(x0)/func2(x0);//formula prin care aflam punctul x1 ce se apropie rapid de solutia cautata

k++;//incrementam valoarea lui k

if(abs(x1-x0)<eps)// daca se respecta conditia data inseamna ca am gasit solutia

{

cout<<"Radacina x= "<<x0<<endl;//daca am gasit solutia o afisam si mai afisam nr de iteratii a algoritmului pina la gasirea solutiei

cout<<"Numarul de iteratii :"<<endl;

break;//iesim din bucla while dupa ce am gasit solutia

}

x0=x1;//daca nu sa gasit solutia repetam algoritmul cu x0=x1;

}

}

int main()

{

cout<<"\nMetoda tangentelor (Newton):"<<endl;

cout<<"Ecuatia 1:"<<endl;

metoda\_tangentelor(-0.4,&f,&fderiv);

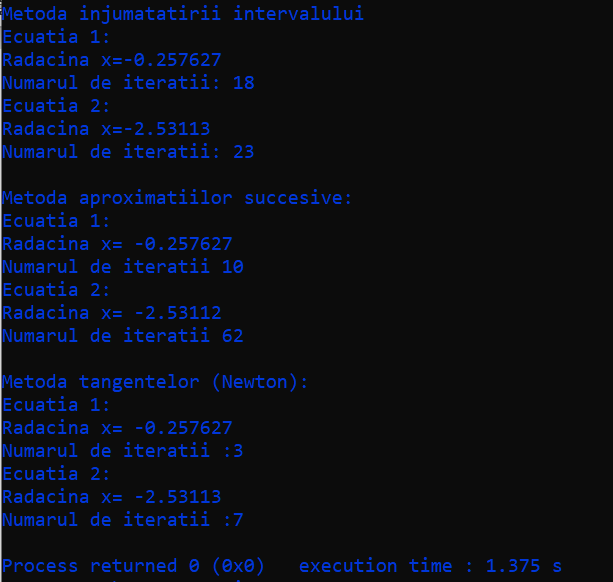
cout<<"Ecuatia 2:"<<endl;

metoda\_tangentelor(-0.5,&f1,&f1deriv);

return 0;

}

Rezultatul(toate cele 3 metode intr-un program):



6) Compararea rezultatelor și concluzia

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Metoda | Rădacina | | Numărul de iterații | | Eroarea |
| f(x) | f1(x) | f(x) | f1(x) |
| Înjumătățirii intervalului | -0.257627 | -2.53113 | 18 | 23 | 0.0001 |
| Aproximării succesive | -0.257627 | -2.53112 | 10 | 62 | 0.000001 |
| Tangentei | -0.257627 | -2.53113 | 3 | 7 | 0.000001 |

Concluzii:

În urma efectuării lucrării de laborator am realizat în practică rezolvarea numerică a ecuațiilor algebrice și transcendente. Putem concluziona că cea mai eficientă metoda este metoda tangentelor, calculatorul efectuând un număr minim de iterații , însă numărul acestor iterații este dependent de aproximația inițială aleasă. Acest număr este cu atât mai mic cu cât aproxima ia inițială este mai aproape de rădăcina căutată. O vulnerabilitate a acestei metode este necesitatea calculului derivatei, ceea ce în unele cazuri poate fi dificil sau practic imposibil.