**Ministеrul Еducаțiеi, Culturii și Cеrcеtării аl Rеpublicii Mоldоvа**

**Univеrsitаtеа Tеhnică а Mоldоvеi**

**Fаcultаtеа Cаlculаtоаrе, Infоrmаtică şi Micrоеlеctrоnică**

**Dеpаrtаmеntul Ingineria Software și Automatică**

**RAPORT**

**Lucrare de laborator nr.1**

**la disciplina Metode Numerice (MN)**

A efectuat: Trifan Iaroslav

st. gr. SI-221

A verificat: Strună Vadim

**Chișinău – 2023**

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

**Varianta:** 24

**Scopul lucrării:**

1. Să se separe toate rădăcinile reale ale ecuației f(x) = 0 unde y=f(x) este o funcție reală de variabilă reală.
2. Să se determine o rădăcină reală a ecuației date cu ajutorul metodeo înjumătățirii intervalului cu o eroare mai mică decât ε = 10-2.
3. Să se precizeze rădăcina obținută cu exactitatea ε = 10-6 utilizând

* Metoda aproximațiilor succesive
* Metoda tangentelor (Newton)

1. Să se compare rezultatele luând în considerație număruil de iterații, evaluările pentru funcția și derivată.

**Descrierea metodelor**

Rezolvarea ecuației f(x) = 0 implică parcurgerea a două etape importante:

* separarea rădăcinilor, care constă în determinarea unui interval [a, b] în care este situată o rădăcină reală a ecuației;
* calculul aproximativ ai fiecărei rădăcini și evaluarea erorii care s-a comis considerând că separarea deja s-a efectuat.

**Sarcina lucrarii:**

Problema dată spre rezolvare

|  |  |
| --- | --- |
| **Nr.** | **Funcția** |
| 24 | a) 2-x^2-2\*e^x |
| b) x^3-25\*x+11 |

Să parcurgem direct la separarea rădăcinilor.

Pentru exemplul a) f(x)=2-x^2-2\*e^x, calculăm domeniul de definiție (DDF).

### DDF:  x ∈ (-∞; +∞) => x ∈ R

### Se vor preciza acum intervalele:

### Pentru x=-2 => f(-2)=2-(-2)^2-2\*e^(-2)=-2.27 (-) – nr. este negativ

### Pentru x=-1 => f(-1)=2-(-1)^2-2\*e^(-1)=-0.26 (+) – nr. este pozitiv

### Între valorile (-2, -1) există cel minim un zerou. (fig. 1.)

### 

Fig. 1. Graficul funcției f(x)=2-x^2-2\*e^x

De asemenea se observă un alt zerou al funcției, în (0, 0).

Separarea rădăcinilor utilizând **metoda grafică**.

De exemplu ecuația *2-x^2-2\*e^x = 0* se poate pune sub forma echivalentă *2-x^2=2\*e^x*

Atunci rădăcinile ei sunt abscisele punctelor de intersecție ale curbelor:

**y = 2-x^2** și **y = 2\*e^x** (fig. 2.)

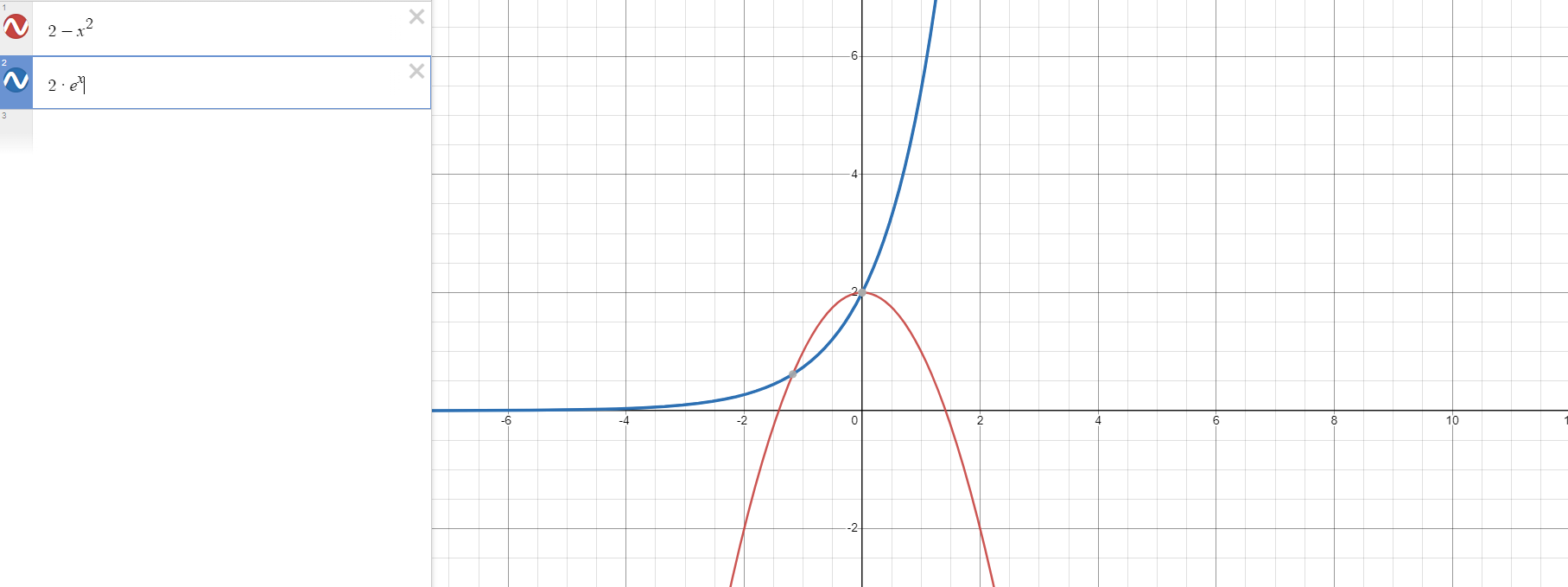
****

Fig. 2. Graficul intesecție funcțiilor y=2-x^2 și y=2\*e^x

Funcțiile se intersectează în două puncte: A(-1.176, 0.617) și B(0, 2)

**Metoda Bisecției (Înjumătățirii intervalelor)**

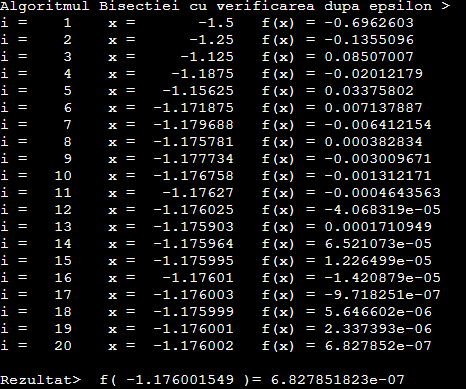
****

Fig. 3. Rezultatul programului prin bisecție(Anexa nr. 1) pentru metoda de verificare epsilon(10-6)

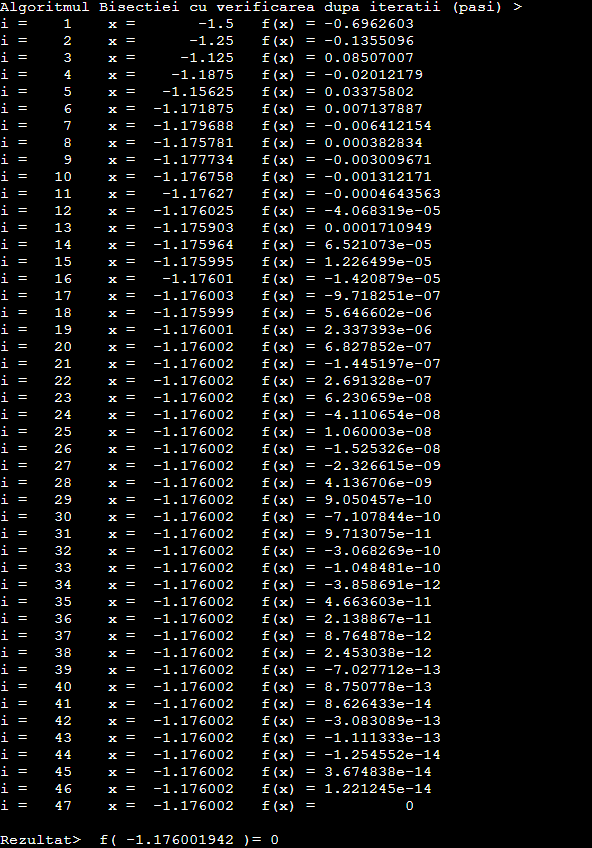
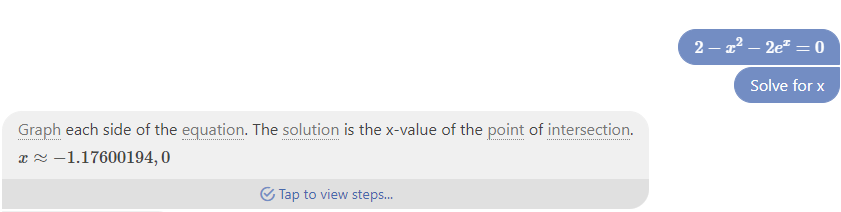


Fig. 4. Rezultatul programului prin bisecție cu nr. de iterații/pași (Anexa nr. 1) pentru metoda de verificare epsilon(10-6)



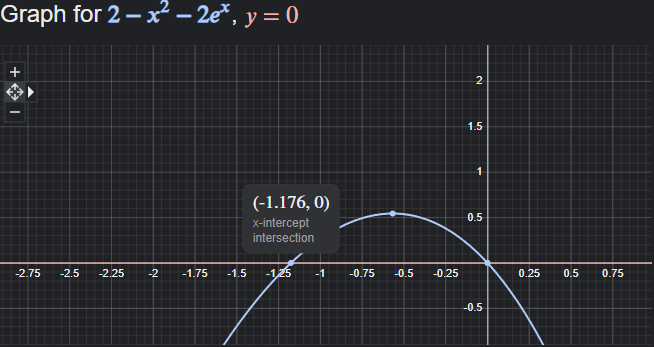


Fig. 5. Ilustrarea graficului a valorii lui x cu indicarea a 8 cifre semnificative după virgulă.

Pentru exemplul b) x^3 – 25\*x + 11;

Folosim metoda șirului lui Rolle pentru a separa rădăcinele.

Fie ecuația F(x) = x^3 - 25\*x + 11 = 0

Derivata funcției F(x) = 3\*x^2 – 25

3\*x^2 – 25 = 0 => x^2 = 25/3 => x1 = - sqrt(25/3) și x2 = sqrt(25/3)

Se anulează pentru x = - (5\*sqrt(3)) / 3, x = (5\*sqrt(3)) / 3

Șirul lui Rolle este următorul:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | -6 | - (5\*sqrt(3)) / 3 | -2 | 0 | 2 | (5\*sqrt(3)) / 3 | 6 |
| y | -55 | 59.1125 | 53 | 11 | -31 | -37.1125 | 77 |

Prin  urmare avem trei alternanții de semn, deci ecuația dată are pe intervalul (-6, 6) trei rădăcini reale r ∈ (-6, - (5\*sqrt(3)) / 3) , r ∈ (0,2) și r ∈ ((5\*sqrt(3)) / 3 , 6).

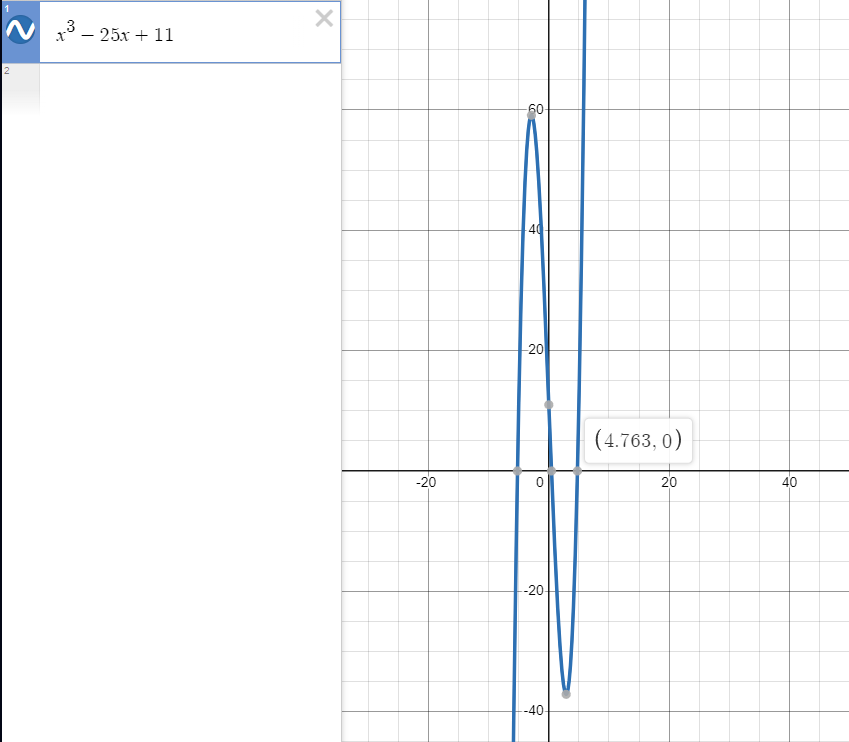


Fig. 6. Graficul funcției y=x^3 – 25\*x + 11

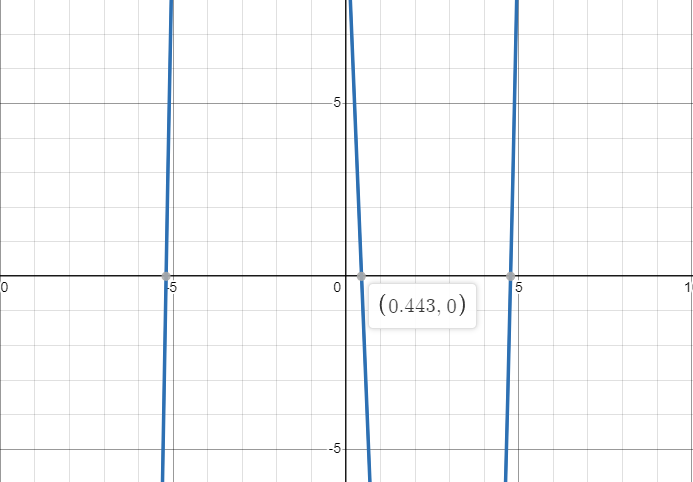


Fig. 7. Graficul funcției y = x^3 – 25\*x + 11, de unde se pot vedea grafic soluțiile ecuației

Pentru intervalul (-6, - (5\*sqrt(3)) / 3), aleg un alt interval ce va include un domeniu mai mare, și va conține ca valori numere întregi.  R ∈ (-6,-2)

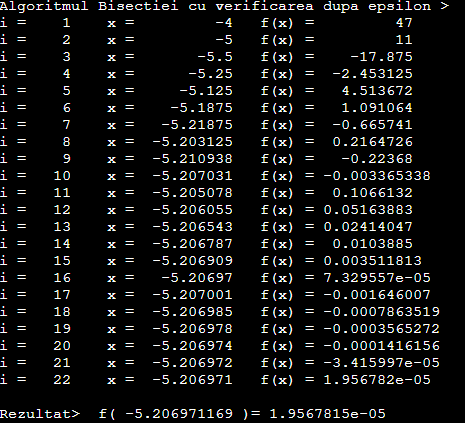


Fig. 8. Rezultatul programului prin bisecție pentru funcția y = x^3 – 25\*x + 11 cu verificarea după epsilon

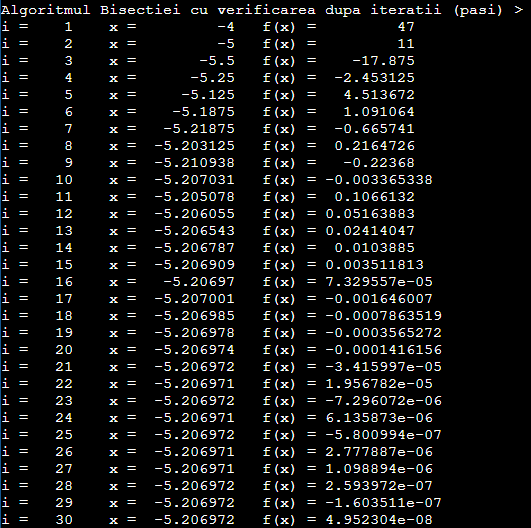


Fig. 9. Rezultatul programului prin bisecție pentru funcția y = x^3 – 25\*x + 11 cu verificarea după epsilon și cu nr. de iterații

Pentru intervalul (0, 2), avem:

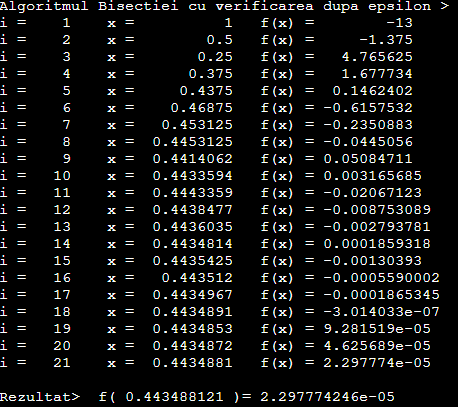


Fig. 10. Rezultatul programului prin bisecție(Anexa nr. 1) prin epsilon

Pentru intervalul ((5\*sqrt(3)) / 3, 6), aleg un alt interval ce va include un domeniu mai mare, și va conține ca valori numere întregi.  R ∈ (2, 6)

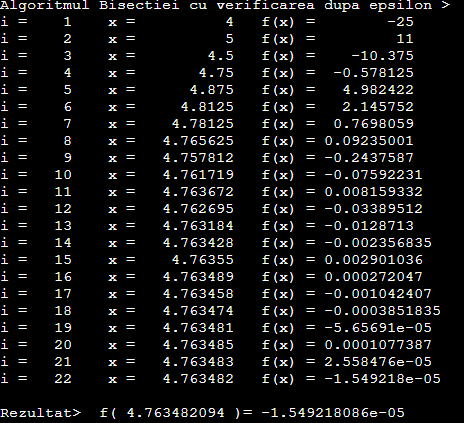


Fig. 11. Rezultatul programului prin bisecție(Anexa nr. 1) prin epsilon

**Metoda lui Newton**

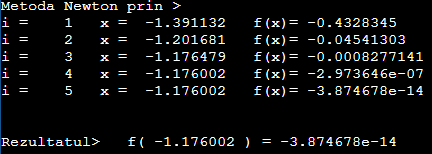
****

Fig. 12. Funcția y = 2-x^2-2\*e^x prin metoda Newton (epsilon)

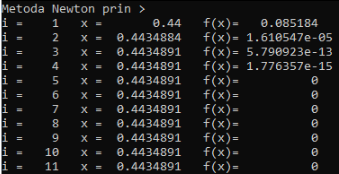




Fig. 13. Funcția y = x^3 – 25\*x + 11 prin metoda Newton (iterații) pentru intervalul (0, 2)

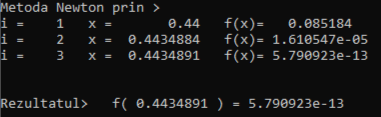


Fig.14. Funcția y = x^3 – 25\*x + 11 prin metoda Newton (epsilon) pentru intervalul (0, 2)

**Metoda aproximațiilor succesive**

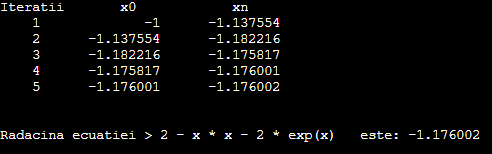


Fig. 15. Funcția y = 2-x^2-2\*e^x prin metoda Aproximațiilor Succesive (epsilon) pentru intervalul (-1.5 , -1)

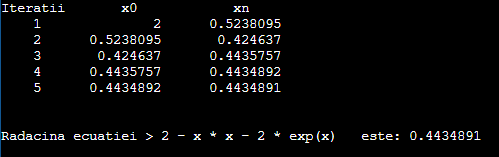
****

Fig. 16. Funcția y = x^3 – 25\*x + 11 prin metoda Aproximațiilor Succesive (epsilon) pentru intervalul (0, 2)

Tabel 1. Tabelul comparării rezultatelor prin diferite metode

|  |  |  |
| --- | --- | --- |
| **Funcția** | 2 - x^2 - 2\*e^x | x^3 – 25\*x + 11 |
| **Intervalul** | [-2, -1] | [0, 2] |
| **Metoda Bisecției** | -1.176001549 | 0.443488121 |
| 47 iterații | 21 iterații |
| **Metoda Aproximațiilor Succesive** | -1.176001 | 0.4434892 |
| 5 iterații | 5 iterații |
| **Metoda Newton** | -1.176002 | 0.4434891 |
| 5 iterații | 4 iterații |
| **Epsilon** | 10^-6 | 10^-6 |

**Concluzie**

În această lucrare de laborator am studiat rezolvarea numerică a ecuațiilor algebrice și transcendente. Pentru aceasta ne-am aprofundat cunoștințele în înțelegea ecuațiilor ce înseamnă ecuația, cum se separă rădăcinile și am studiat 3 metode de calculare a rădăcinilor ecuației. Precum și am prevăzut criteriile de oprire în metodele iterative.

Deci pentru a calcula rădăcina unei ecuației e nevoie de ales anume o metodă potrivită pentru a afișa rezultatul într-un timp scurt și cât mai precis. Astfel e nevoie de prevăzut și condiția de stopare. În metodele iterative oprirea procesului de calcul se face prin trunchierea șirului de iterare {xk} la un indice m, astfel încât termenul xm să constituie aproximația satisfăcătoare a rădăcinii exacte.

Anexa nr. 1

Metoda Bisectiei

#include <math.h>  
#include <iostream>  
#include <iomanip>  
  
/\*  
 \* a) 2 - x^2 - 2\*e^x = 2 - x\*x - 2\*exp(x)  
 \* b) x^3 - 25\*x + 11 = pow(x, 3) - 25\*x + 11  
 \*/  
  
using namespace std;  
  
  
double a = -6;  
double b = -2;  
double eps = pow(10, -6);  
  
  
double Function\_a(double x)  
{  
 return pow(x, 3) - 25\*x + 11;  
}  
  
  
int main()  
{  
 double c;  
 int k = 500;  
 int i = 0;  
  
  
 cout << "Algoritmul Bisectiei cu verificarea dupa epsilon > " << endl;  
 do  
 {  
 i++;  
 c = (a + b) / 2;  
 cout << "i = " << setw(4) << i << " x = " << setw(10) << setprecision(7) << c << " f(x) = " << setw(10) << setprecision(7) << Function\_a(c) << endl;  
 if (Function\_a(c) == 0)  
 {  
 cout << endl << "Rezultat> f( " << c << " )= " << Function\_a(c); break;  
 }  
 else  
 if (Function\_a(c) \* Function\_a(a) > 0) a = c; else b = c;  
 } while (abs(b - a) > eps);  
 cout << endl << "Rezultat> f( " << setprecision(10) << c << " )= " << Function\_a(c);  
 cout << endl << "######################################################################" << endl;  
  
  
 a = -6;  
 b = -2;  
 cout << endl << endl << "Algoritmul Bisectiei cu verificarea dupa iteratii (pasi) > " << endl;  
 for (int i = 1; i <= k; i++)  
 {  
 c = (a + b) / 2;  
 cout << "i = " << setw(4) << i << " x = " << setw(10) << setprecision(7) << c << " f(x) = " << setw(10) << setprecision(7) << Function\_a(c) << endl;  
 if (Function\_a(c) == 0)  
 {  
 cout << endl << "Rezultat> f( " << setprecision(10) << c << " )= " << Function\_a(c); break;  
 }  
 else  
 if (Function\_a(c) \* Function\_a(a) > 0) a = c; else b = c;  
 }  
 return 0;  
}

Anexa nr.2

Metoda Newton

#include <math.h>  
#include <iostream>  
#include <iomanip>  
  
/\*  
 \* a) 2 - x^2 - 2\*e^x = 2 - x\*x - 2\*exp(x)  
 \* b) x^3 - 25\*x + 11 = pow(x, 3) - 25\*x + 11  
 \*/  
  
using namespace std;  
  
  
double a = -2;  
double b = -1;  
double eps = pow(10, -6);  
  
  
double Function(double x)  
{  
 return 2 - x\*x - 2\*exp(x);  
}  
  
  
double Function\_Derivata(double x)  
{  
 return -2\*x - 2\*exp(x);  
 // return 3\* pow(x,2) – 25;  
}  
  
  
int main()  
{  
 int k = 50;  
  
  
 double c = a - (Function(a)) / (Function(b) - Function(a)) \* (b - a);  
 double x;  
  
  
 cout << "Metoda Newton prin > " << endl;  
  
  
 if (Function(c) \* Function(a) < 0)  
 x = a;  
 else  
 x = b;  
  
  
 double h;  
 double i = 0;  
 do  
 {  
 i++;  
 h = Function(x) / Function\_Derivata(x);  
 x = x - h;  
 if (i > k) break;  
 cout << "i = " << setw(4) << i << " x = " << setprecision(7) << setw(10) << x << " f(x)= " << setprecision(7) << setw(10) << Function(x) << endl;  
 } while ( /\*true\*/ abs(h) >= eps);  
  
  
 cout << endl << endl << "Rezultatul> f(" << setprecision(7) << setw(10) << x << " ) = " << setprecision(7) << setw(10) << Function(x) << endl;  
  
  
 return 0;  
}

Anexa nr.3

Metoda Aproximatiilor Succesive

#include <iostream>  
#include <cmath>  
#include <iomanip>  
  
//pow(x, 3) - 25\*x + 11  
  
using namespace std;  
  
double eps = 0.000001;  
  
double function(double x)  
{  
 return 2 - x \* x - 2 \* exp(x);  
}  
  
int main()  
{  
 double x0, x1, xn;  
 int iteratii = 0;  
  
 cout << "Iteratii x0 xn\n";  
  
 x0 = -1.5;  
 x1 = -1.0;  
  
 while (fabs(x1 - x0) > eps)  
 {  
 xn = x1 - (function(x1) \* (x1 - x0)) / (function(x1) - function(x0));  
 x0 = x1;  
 x1 = xn;  
 iteratii++;  
  
 cout << setw(5) << iteratii << setw(15) << setprecision(7) << x0 << setw(15) << setprecision(7) << xn << "\n";  
 }  
  
 cout << "\n\nRadacina ecuatiei > 2 - x \* x - 2 \* exp(x) este: " << xn << endl << endl;  
  
 return 0;  
}