Universitatea Tehnică a Moldovei

Facultatea *Calculatoare, Informatică și Microelectronică*

Specialitatea *Tehnologii Informaționale*

Raport

la lucrarea de laborator nr. 1

Tema:*“Rezolvarea numerica a ecuatiilor algebrice si transcedente”*

Disciplina: “Metode numerice”

Varianta 3

A efectuat:	Student grupa TI-231 FR	Apareci Aurica
A verificat:	Asistent universitar	Strună Vadim

Chișinău 2024

Cuprins

1. Cadru teoretic

Tema lucrării: Rezolvarea numerica a ecuatiilor algebrice si transcedente

Scopul lucrării:

a) Să se separe toate rădăcinile ecuației *f(x)=0* unde *y=f(x)* este o funcție reală de variabilă reală.

b) Să se determine o rădăcină reală a ecuației date cu ajutorul metodei înjumătățirii intervalului cu o eroare mai mică decât ε=10-2.

c) Să se precizeze rădăcina obținută cu exactitatea ε=10-6 utilizând:

- metoda aproximațiilor succesive
- metoda tangentelor (Newton)
- metoda secantelor

d) Să se compare rezultatele luând în considerație numărul de iterații, evaluările pentru funcția și derivată.

Sarcina (V3): a) e^x+3x b) $x^3-23x-42$

2. Rezolvarea ecuaţiilor

Rezolvarea ecuaţiilor algebrice liniare şi operaţiile de calcul matriceal sunt incluse în domeniul algebrei liniare – implicată în diverse probleme ştiinţifice, de exemplu:

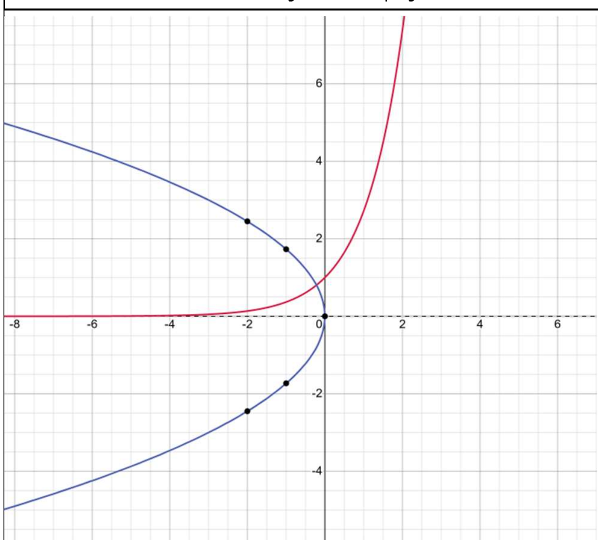
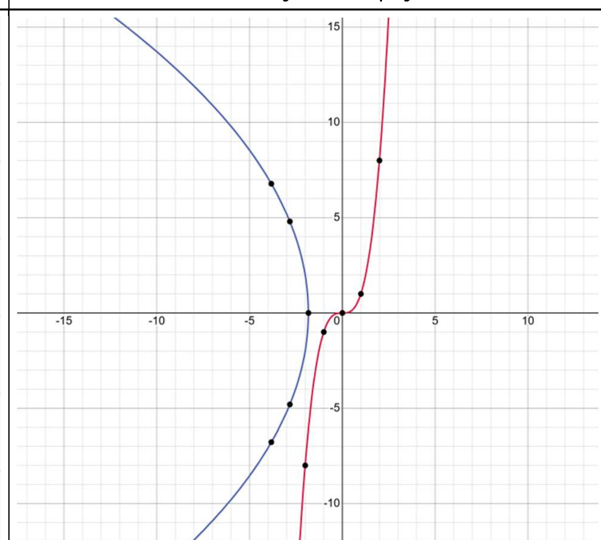
– problemele care depind de un număr finit de grade de libertate, reprezentate prin ecuaţii diferenţiale ordinare sau cu derivate parţiale sunt transformate, cu ajutorul diferenţelor finite

– problemele neliniare sunt frecvent soluţionate (aproximate) prin procese de liniarizare;

– programarea liniară implică rezolvarea unor sisteme de ecuaţii algebrice liniare;

– foarte multe probleme inginereşti din domeniul reţelelor electrice, analiza structurilor, proiectarea clădirilor, vapoarelor, avioanelor, transportul lichidelor şi gazelor prin conducte etc. necesită, pentru soluţionare, rezolvarea unor sisteme liniare.

2.1 Metoda grafică

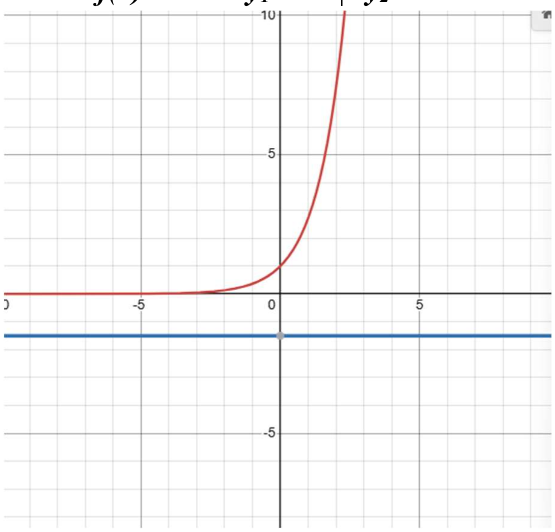
Separarea grafica a soluțiilor	
f(x) = $e^x + 3x$ e^x = *-3x* => ***y₁***= e^x | ***y₂= -3x***	*f(x)* = $x^3 - 23x - 42$ x^3=*-23x-42* => ***y₁***= $\boldsymbol{x^3}$ | ***y₂= -23x-42***
	
Ecuația are o singura rădăcină: r_1 (0,1) Abscisa punctului de intersecție a graficelor $\mathbf{r_1}$ se afla pe intervalul sau ***(0;1)***	Cele două grafice se intersectează în **trei puncte distincte**, ceea ce indică existența a trei soluții reale. Rădăcinile ecuației se află în următoarele intervale: **(−∞,−3), (−3,0), (0,∞).**

2.2 Metoda analitică

Metoda şirului lui Rolle. Se ştie din cursul de analiză matematică că între două rădăcini reale consecutive ale derivatei funcţiei y=f(x) există cel mult o rădăcină reală a ecuaţiei f (x) = 0 . De asemenea între două rădăcini consecutive ale ecuaţiei f(x)=0 există o rădăcină a ecuaţiei f (′ x). Ecuaţia f(x)=0 are atâtea rădăcini reale câte alternanţe de semn prezintă şirul lui Rolle.

<table>
<tr><th>f(x) = e^x+3x</th><th>f(x) = $x^3 - 23x - 42$</th></tr>
<tr><td>

1. Determinăm derivata funcției
   Df(x) = e^x+3
2. Împărțim ecuația în doua părți și găsim rădăcinile prin metoda grafică
   Df(x) = 0 => $y_1 = e^x$ | $y_2 = -3$



Nu există soluții.

</td><td>

1. Determinăm derivata funcției:
   Df(x) = 3x^2-23
2. **2.** Determinarea punctelor critice:
   Df(x) = 0 => $x_1 \approx -2.77, \quad x_2 \approx 2.77$
3. **3.** Împărțim ecuația în doua părți și găsim rădăcinile prin metoda intervalelor:
   $y_1 = x^3$ | $y_2 = 23x+42$

<table>
<tr><th>x</th><th>$f(x) = x^3 - 23x - 42$</th></tr>
<tr><td>-5</td><td>$(-5)^3 - 23(-5) - 42 = -125 + 115 - 42 = -52$</td></tr>
<tr><td>-3</td><td>$(-3)^3 - 23(-3) - 42 = -27 + 69 - 42 = 0$</td></tr>
<tr><td>0</td><td>$0^3 - 23(0) - 42 = -42$</td></tr>
<tr><td>5</td><td>$5^3 - 23(5) - 42 = 125 - 115 - 42 = -32$</td></tr>
</table>

Rezultatul arată că există rădăcini în intervalele: (−5,−3), (−3,0), (0,5)

4. Micșorarea intervalelor:
   (-5, -3) *=> f(−4) = (−4)³ −23*(−4) −42 = −14.*
   (-3, 0) *=> f(−1.5)=(−1.5)³−(−1.5)−42 = −10.87.*
   (0, 5) *=> f(2.5)=(2.5)³−23*(2.5)−42 = −83.875.*
5. Determinarea soluțiilor aproximative:
   $x_1 \approx -4$, $x_2 \approx -1.5$, $x_3 \approx 2.5$

</td></tr>
</table>

2.3 Metoda tangentelor (Newton)

Fie ecuaţia algebrică sau transcendentă f(x)=0 care admite o singură rădăcină reală r în intervalul [a, b]. Presupunem în plus că derivatele f ′(x) şi f ′′(x) păstrează un semn constant pe intervalul [a, b]. Metoda lui Newton este definită de următoarea formulă: $x_{k+1} = x_k - \frac{f(x_k)'}{f'(x)}$, unde x_0 este aproximaţia iniţială a rădăcinii din intervalul [a, b]. Punctul x $_{k+1}$ este abscisa punctului de intersecţie a tangentei dusă la curba y=f(x) în punctul x_k cu axa OX. De aceea această metodă se mai numeşte metoda tangentelor.

<table>
<tr><td>

$f''(x) > 0\ sau\ f''(x) < 0$

$f(x) = e^x + 3x$

$f'(x) = e^x + 3 > 0$

$f''(x) = e^x > 0$

$f'(x) * f''(x) > 0$

$x_0 = b = 0{,}5$

$[a, b] = [-0{,}5; 0{,}5]$

</td><td>

$f''(x) > 0\ sau\ f''(x) < 0$

$f(x) = x^3 - 23x - 42$

$f'(x) = 3x^2 - 23 < 0$

$f''(x) = 6x < 0$
$\frac{x}{f(x)} < 0$
$x_{1,2} = \pm\sqrt{10} = \pm 3.16227 \ldots$
$[a, b] = [-5, -4]$
$f(-5) < 0, f(-4) > 0$

$x_0 = -5$

</td></tr>
</table>

Listingul programului

NewtonsMethod / Program.cs

```
namespace NewtonsMethod
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Metoda lui Newton");

            // Function A: e^x+3x
            Console.WriteLine("A: e^x+3x = 0");
            Newton newtonA = new Newton(1e-8, 100, -0.5);
            double rootA = newtonA.Solve(Data.fx, Data.dfx);
            Console.WriteLine($"Radacina pentru A: {rootA:f4}\n");

            // Function B: x^3 - 23x - 42
            Console.WriteLine("B: x^3 - 23x - 42 = 0");
            Newton newtonB = new Newton(1e-8, 100, -5);
            double rootB = newtonB.Solve(Data.gx, Data.dgx);
            Console.WriteLine($"Radacina pentru B: {rootB:f4}");
        }
    }
}
```

NewtonsMethod / Newton.cs

```
namespace NewtonsMethod
{
  public class Newton : Template
  {
      public Newton(double eps, int maxIter, double x0)
      {
          Eps = eps;    MaxIter = maxIter;   X0 = x0;
      }

      public double Solve(Func<double, double> f, Func<double, double> df)
      {
          double x = X0;
          double x1;
          int i = 0;
          Console.WriteLine($"Iteratia: {i}  x = {x:f4}  f(x) = {f(x):f4}
df(x) = {df(x):f4}");
          do
          {
              double dfx = df(x);
              if (Math.Abs(dfx) < 1e-10)
              {
                  Console.WriteLine("Derivata este aproape de zero. Metoda
Newton a esuat.");
                  return double.NaN;
              }

              x1 = x - f(x) / dfx;
              Console.WriteLine($"Iteratia: {++i}  x = {x1:f4}  f(x) =
{f(x1):f4}  df(x) = {df(x1):f4}");

              if (Math.Abs(f(x1)) < Eps || Math.Abs(x1 - x) < Eps) break;
              x = x1;
          } while (i < MaxIter);
          if (i == MaxIter)
          Console.WriteLine("Numarul maxim de iteratii a fost atins.");
          return x1;
        }
    }
}
```

NewtonsMethod / Template.cs

```
namespace NewtonsMethod
{
    public class Template
    {
        public double Eps { get; set; }
        public int MaxIter { get; set; }
        public double X0 { get; set; }
    }
}
```

NewtonsMethod / Data.cs

```
namespace NewtonsMethod
{
    public static class Data
    {
        // Function A: e^x+3x
        public static double fx(double x) => Math.Pow(Math.E, x) + 3 * x;
        public static double dfx(double x) => Math.Pow(Math.E, x) + 3;
        // Function B: x^3 - 23x-42
        public static double gx(double x) => Math.Pow(x, 3) - 23 * x - 42;
        public static double dgx(double x) => 3 * Math.Pow(x, 2) - 23;
    }
}
```

Rezultatele testării

```
Metoda lui Newton
A: e^x+3x = 0
Iteratia: 0  x = -0.5000  f(x) = -0.8935  df(x) = 3.6065
Iteratia: 1  x = -0.2523  f(x) = 0.0202  df(x) = 3.7770
Iteratia: 2  x = -0.2576  f(x) = 0.0000  df(x) = 3.7729
Iteratia: 3  x = -0.2576  f(x) = 0.0000  df(x) = 3.7729
Radacina pentru A: -0.2576

B: x^3 - 23x - 42 = 0
Iteratia: 0  x = -5.0000  f(x) = -52.0000  df(x) = 52.0000
Iteratia: 1  x = -4.0000  f(x) = -14.0000  df(x) = 25.0000
Iteratia: 2  x = -3.4400  f(x) = -3.5876  df(x) = 12.5008
Iteratia: 3  x = -3.1530  f(x) = -0.8263  df(x) = 6.8244
Iteratia: 4  x = -3.0319  f(x) = -0.1369  df(x) = 4.5777
Iteratia: 5  x = -3.0020  f(x) = -0.0081  df(x) = 4.0363
Iteratia: 6  x = -3.0000  f(x) = -0.0000  df(x) = 4.0002
Iteratia: 7  x = -3.0000  f(x) = -0.0000  df(x) = 4.0000
Radacina pentru B: -3.0000
```

2.4 Metoda secantelor

Metoda secantelor se deduce din metoda lui Newton înlocuind derivate. $f'(x) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$ Pentru startul iteraţiilor în metoda secantelor avem nevoie de două aproximaţii iniţiale x0 şi x1. Valoarea *xk+1* este abscisa punctului de intersecţie dintre secanta care trece prin punctele *(xk-1,f(xk-1))* şi *(xk,f(xk))* şi OX; de aici şi denumirea metodei.

Listingul programului

SecantMethod / Program.cs

```
static void Main(string[] args)
      {
          Console.WriteLine("Metoda Secantei");
          // Function A: e^x+3x
          Console.WriteLine("A: e^x+3x = 0");
          Secant secantA = new Secant(1e-8, 100, -0.5, 0.0);
          double rootA = secantA.Solve(Data.fx);
          Console.WriteLine($"Radacina pentru A: {rootA:f5}\n");
          // Function B: x^3 - 23x-42
          Console.WriteLine("B: x^3 - 23x-42 = 0");
          Secant secantB = new Secant(1e-8, 100, -5.0, -4.0);
          double rootB = secantB.Solve(Data.gx);
          Console.WriteLine($"Radacina pentru B: {rootB:f5}");
      }
```

SecantMethod / Secant.cs

```
using NewtonsMethod;
namespace SecantMethod
{
    public class Secant : Template
    {
        public double X1 { get; set; }
        public Secant(double eps, int maxIter, double x0, double x1)
        {
            Eps = eps;
            MaxIter = maxIter;
            X0 = x0;
            X1 = x1;
        }
        public double Solve(Func<double, double> f)
        {
            double f0 = f(X0);
            double f1 = f(X1);

            for (int i = 0; i < MaxIter; i++)
            {
                double x2 = X1 - f1 * (X1 - X0) / (f1 - f0);
                double f2 = Math.Pow(Math.E, x2) + 3 * x2;

                if (Math.Abs(x2 - X1) < Eps)
                {
                    return x2;
                }
                Console.WriteLine($"Iteratia: {i + 1}  x = {x2:f5} f(x) =
{f(x2):f5}");

                X0 = X1;
                f0 = f1;
                X1 = x2;
                f1 = f2;
            }

            Console.WriteLine("Metoda secantei nu converge.");
            return double.NaN;
        }
    }
}
```

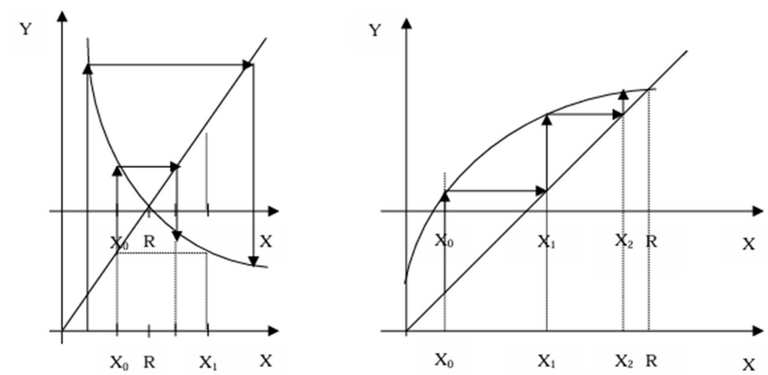
Rezultatele testării

```
Metoda Secantei
A: e^x+3x = 0
Iteratia: 1  x = -0.26407 f(x) = -0.02427
Iteratia: 2  x = -0.25781 f(x) = -0.00068
Iteratia: 3  x = -0.25763 f(x) = 0.00000
Iteratia: 4  x = -0.25763 f(x) = -0.00000
Radacina pentru A: -0.25763

B: x^3 - 23x-42 = 0
Iteratia: 1  x = -3.63158 f(x) = -6.36828
Iteratia: 2  x = -2.35302 f(x) = -0.90848
Iteratia: 3  x = -0.07248 f(x) = -40.33323
Iteratia: 4  x = -0.28419 f(x) = -35.48661
Iteratia: 5  x = -0.25815 f(x) = -36.07975
Iteratia: 6  x = -0.25763 f(x) = -36.09170
Iteratia: 7  x = -0.25763 f(x) = -36.09166
Radacina pentru B: -0.25763
```

2.5 Metoda aproximarilor successive

Ecuaţia $f(x)=0$ o punem sub forma echivalentă $x=\phi(x)$. Plecând de la o valoare iniţială arbitrară x0 generăm şirul xk după regula: $xk+1=\phi(xk)$, $k=0,1,2\ldots$, adică $x2=\phi(x0)$, $x2=\phi(x1),\ldots,xk=\phi(xk\text{-}1),\ldots$ Din punct de vedere geometric, rădăcina reală r este abscisa punctului de intersecţie a curbei $y=\phi(x)$ cu dreapta $y=x$. Modul cum şirul aproximaţiilor succesive x0, x1,…,xk,…conduce spre soluţia exactă este ilustrat (în funcţie de forma curbei $y=\phi(x)$).



$f(x)\varphi \rightarrow x = e^x + 3x = \varphi(x)$ $\lvert\varphi'(x)\rvert \le q < 1$ $pe[-0{,}5; 0{,}5] \rightarrow q = e^{0.5} = 0{,}48$ $x = \varphi(x)$ $f(x) = e^x + 3x \rightarrow x = \varphi(x)$ $\varphi(x) = e^x + 3x$	$f'(x) = 3x^2 - 30 > 0$ $q = \frac{m}{M}$ $\varphi(x) = x - \frac{1}{M} * f(x)$ $M = max f'(x) \rightarrow f'(-5), x \in [-5; -4]$ $m = min f'(x) \rightarrow f'(-4), x \in [-5; -4] M = 45, m = 18 q = \frac{18}{45} = 0{,}4$ $\varphi(x) = x - \frac{1}{45} * f(x)$ $\varphi(x) = x - \frac{1}{45} * (x^3 - 23x - 42)$

Listingul programului

SuccesiveApprox / Program.cs

```
using NewtonsMethod;

namespace SuccessiveApprox
{
  internal class Program
  {
      static void Main(string[] args)
      {
          Console.WriteLine("Metoda aproximarilor succesive");
          SuccessiveApprox successiveApprox = new SuccessiveApprox(0.0000001,
100, -0.5);
          Console.WriteLine("A: e^x+3x = 0");
          double x = successiveApprox.Solve(Data.fx,SuccesiveApproxData.FiFx);
          Console.WriteLine("B: x^3 - 23x - 42 = 0");
          successiveApprox = new SuccessiveApprox(0.0000001, 100, -5);
          double y = successiveApprox.Solve(Data.gx,SuccesiveApproxData.FiGx);
      }
  }
}
```

SuccesiveApprox / SuccesiveApprox.cs

```
using NewtonsMethod;

namespace SuccessiveApprox
{
    public class SuccessiveApprox:Template
    {
        public SuccessiveApprox(double eps, int maxIter, double x0)
        {
            Eps = eps;
            MaxIter = maxIter;
            X0 = x0;
        }
        public double Solve(Func<double, double> f, Func<double, double> g)
        {
            double x = X0;
            double x1 = g(x);
            int i = 0;
            while (Math.Abs(x1 - x) > Eps && i < MaxIter)
            {
                x = x1;
                x1 = g(x);
                i++;
                Console.WriteLine($"Iteratia: {i}  x = {x1} f(x) = {f(x1)}");
            }
            return x1;
        }
    }
}
```

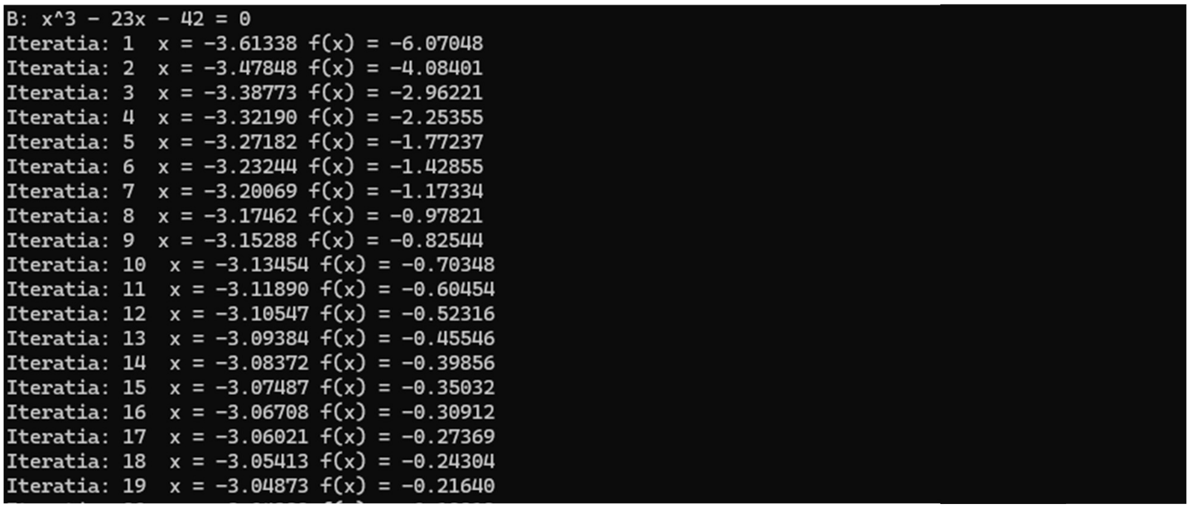
SuccesiveApproxData / SuccesiveApprox.cs

```
using NewtonsMethod;

namespace SuccessiveApprox
{
    public class SuccesiveApproxData:Data
    {
        public static double FiFx(double x)
        {
            double res = Math.Pow(Math.E, x) + 3 * x;
            return res;
        }
        public static double FiGx(double x)
        {
            double res = x - gx(x)/45;
            return res;
        }
    }
}
```

Rezultatele testării

```
Metoda aproximarilor succesive
A: e^x+3x = 0
Iteratia: 1  x = -2.27117 f(x) = -6.71033
Iteratia: 2  x = -6.71033 f(x) = -20.12978
Iteratia: 3  x = -20.12978 f(x) = -60.38934
Iteratia: 4  x = -60.38934 f(x) = -181.16802
Iteratia: 5  x = -181.16802 f(x) = -543.50405
Iteratia: 6  x = -543.50405 f(x) = -1630.51214
Iteratia: 7  x = -1630.51214 f(x) = -4891.53643
Iteratia: 8  x = -4891.53643 f(x) = -14674.60929
Iteratia: 9  x = -14674.60929 f(x) = -44023.82787
Iteratia: 10  x = -44023.82787 f(x) = -132071.48360
Iteratia: 11  x = -132071.48360 f(x) = -396214.45079
Iteratia: 12  x = -396214.45079 f(x) = -1188643.35238
Iteratia: 13  x = -1188643.35238 f(x) = -3565930.05713
Iteratia: 14  x = -3565930.05713 f(x) = -10697790.17140
Iteratia: 15  x = -10697790.17140 f(x) = -32093370.51419
Iteratia: 16  x = -32093370.51419 f(x) = -96280111.54258
Iteratia: 17  x = -96280111.54258 f(x) = -288840334.62775
```



2.6 Metoda înjumătățirii intervalului / bisecției

Metoda bisecției este o metodă iterativă de căutare a soluției, în care un interval este înjumătățit în repetate rânduri. Dacă funcția schimbă semnul pe un anumit interval, valoarea funcției pentru punctul de la mijlocul intervalului este determinată. Poziția rădăcinii este apoi considerată ca aflându-se în interiorul subintervalului unde apare schimbarea de semn. Subintervalul devine astfel intervalul folosit pentru noua iterație. Procesul se repetă până când rădăcina respectă standardele de precizie dorite (toleranța).

Listingul programului

BisectionMethod / Program.cs

```
using BisectionMethod.BisectionMethod;

using NewtonsMethod;

namespace BisectionMethod
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Metoda Bisectiei (Injumatatirii Intervalu-
lui)");

            // Function A: e^x + 3x
            Func<double, double> fx = x => Math.Exp(x) + 3 * x;
            Console.WriteLine("A: Solving e^x + 3x = 0");
            Bisection bisectionA = new Bisection(1e-7, 100, -1, 0);
            double rootA = bisectionA.Solve(Data.fx);
            Console.WriteLine($"Radacina pentru A: {rootA:f5}\n");

            // Function B: x^3 - 23x - 42
            Func<double, double> gx = x => Math.Pow(x, 3) - 23 * x - 42;
            Console.WriteLine("B: Solving x^3 - 23x - 42 = 0");
            Bisection bisectionB = new Bisection(1e-7, 100, -5, 5);
            double rootB = bisectionB.Solve(Data.gx);
            Console.WriteLine($"Radacina pentru B: {rootB:f5}");
        }
    }
}
```

BisectionMethod / Bisection.cs

```
using NewtonsMethod;

namespace BisectionMethod
{
    namespace BisectionMethod
    {
        internal class Bisection
        {
            private double _a;
            private double _b;
            private double _eps;
            private int _maxIter;

            public Bisection(double eps, int maxIter, double a, double b)
            {
                _eps = eps;
                _maxIter = maxIter;
                _a = a;
                _b = b;
            }

            public double Solve(Func<double, double> f)
            {
                double x = (_a + _b) / 2;
                int i = 0;

                if (f(_a) * f(_b) > 0)
                {
                    Console.WriteLine("Invalid interval: f(a) and f(b) must
have opposite signs.");
                    return double.NaN;
                }

                while (Math.Abs(_b - _a) > _eps && i < _maxIter)
                {
                    x = (_a + _b) / 2;

                    if (f(x) == 0 || Math.Abs(f(x)) < _eps)
                    {
                        break;
                    }

                    if (f(_a) * f(x) < 0)
                    {
                        _b = x;
                    }
                    else
                    {
                        _a = x;
                    }

                    i++;
                    Console.WriteLine($"Iteratia: {i}  x = {x} f(x) =
{f(x)}");
                }

                return x;
            }
        }
    }
}
```

Rezultatele testării

```
Metoda Bisectiei (Injumatatirii Intervalului)
A: Solving e^x + 3x = 0
Iteratia: 1  x = -0.5 f(x) = -0.8934693402873666
Iteratia: 2  x = -0.25 f(x) = 0.02880078307140488
Iteratia: 3  x = -0.375 f(x) = -0.43771072120902776
Iteratia: 4  x = -0.3125 f(x) = -0.20588437105335822
Iteratia: 5  x = -0.28125 f(x) = -0.08891039801099265
Iteratia: 6  x = -0.265625 f(x) = -0.030148403929179945
Iteratia: 7  x = -0.2578125 f(x) = -0.0006973927054274576
Iteratia: 8  x = -0.25390625 f(x) = 0.014045776561826262
Iteratia: 9  x = -0.255859375 f(x) = 0.006672715161448517
Iteratia: 10  x = -0.2568359375 f(x) = 0.002987292396773533
Iteratia: 11  x = -0.25732421875 f(x) = 0.0011448576828816392
Iteratia: 12  x = -0.257568359375 f(x) = 0.00022370945365401962
Iteratia: 13  x = -0.2576904296875 f(x) = -0.00023684738395202132
Iteratia: 14  x = -0.25762939453125 f(x) = -6.570404753158954E-06
Iteratia: 15  x = -0.257598876953125 f(x) = 0.00010856916453838572
Iteratia: 16  x = -0.2576141357421875 f(x) = 5.099928991592062E-05
Iteratia: 17  x = -0.25762176513671875 f(x) = 2.2214420087429687E-05
Iteratia: 18  x = -0.2576255798339844 f(x) = 7.822002043633702E-06
Iteratia: 19  x = -0.2576274871826172 f(x) = 6.257972393619582E-07
Iteratia: 20  x = -0.2576284408569336 f(x) = -2.9723041083951074E-06
Iteratia: 21  x = -0.2576279640197754 f(x) = -1.1732535223352158E-06
Iteratia: 22  x = -0.2576277256011963 f(x) = -2.737281634690447E-07
Iteratia: 23  x = -0.25762760639190674 f(x) = 1.7603453239534161E-07
Radacina pentru A: -0.25763

B: Solving x^3 - 23x - 42 = 0
Invalid interval: f(a) and f(b) must have opposite signs.
Radacina pentru B: NaN
```

3. Concluzii

În cadrul acestei lucrări, am dezvoltat programe pentru rezolvarea numerică a ecuațiilor algebrice și transcendente, utilizând metodele înjumătățirii intervalului, aproximațiilor succesive, tangentelor (Newton) și secantelor. Obiectivele stabilite au fost îndeplinite cu succes, iar rezultatele obținute au fost analizate comparativ pentru a identifica metoda cea mai eficientă în contextul dat.

Lucrarea a fost implementată utilizând limbajul de programare C#, cu ajutorul mediului de dezvoltare Visual Studio, cunoscut pentru flexibilitatea și eficiența sa. Alegerea C# s-a bazat pe capacitățile avansate de manipulare a ecuatiilor și suportul excelent pentru programarea orientată pe obiecte, facilitând implementarea logicii algoritmilor pentru rezolvarea ecuațiilor.

1. **Metoda înjumătățirii intervalului**

   Aceasta a fost utilizată pentru a garanta separarea rădăcinilor ecuației și obținerea unei soluții cu o eroare mai mică decât ε = 10^-2. Avantajele metodei includ robustitatea și faptul că convergența este garantată dacă ecuația este continuă pe intervalul ales. Totuși, convergența este liniară, iar numărul de iterații poate fi semnificativ mai mare comparativ cu alte metode.

2. **Metoda aproximațiilor successive**

   Această metodă a fost eficientă pentru obținerea unei soluții cu precizia ε = 10^-6, cu condiția ca funcția să fie contracție pe intervalul considerat. Convergența este mai lentă decât în cazul metodei tangentelor, însă metoda are avantajul simplității implementării și nu necesită calcularea derivatei.

3. **Metoda tangentelor (Newton)**

   Aceasta s-a dovedit a fi cea mai rapidă metodă din punct de vedere al numărului de iterații datorită convergenței de ordinul doi. Totuși, metoda necesită calcularea derivatei funcției, ceea ce poate reprezenta un dezavantaj pentru ecuații complexe. De asemenea, convergența nu este garantată dacă punctul inițial nu este ales în apropierea rădăcinii.

4. **Metoda secantelor**

   Comparativ cu metoda tangentelor, metoda secantelor nu necesită calculul explicit al derivatei, ci folosește o aproximație numerică. Deși convergența este mai lentă decât a metodei Newton, aceasta este superioară metodei înjumătățirii intervalului, iar cerințele computaționale sunt mai reduse decât în cazul metodei Newton.

Prin urmare, alegerea metodei optime depinde de specificul ecuației analizate și de resursele disponibile.

4. Webobrafie

- Curs ***Metode numerice*** https://else.fcim.utm.md/course/view.php?id=1689
- Inteligență artificială https://chatgpt.com/
- Suport curs ***Metode numerice*** https://elth.ucv.ro/fisiere/probleme%20studentesti/Cursuri/Metode%20numerice/curs_met_num.pdf