

Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Звіт

про виконання лабораторної роботи № 3

з курсу “Алгоритмізація та програмування”

«Розв’язування нелінійних рівнянь методом ділення навпіл»

Виконав:

ст. гр. ФЕІ-11

Стасів Петро

Перевірив:

доц. Хвищун І.О.

Львів-2021

Звіт

Мета: у середовищі Delphi написати програму яка буде шукати корінь нелінійного рівняння методом ділення навпіл(МДН).

Виконання лабораторної роботи:

- 1) Визначаємо константи, InitialFunctionStep - визначає початковий аргумент для табулювання функції, IterationsLimit - обмеження в ітераціях обрахунку інтервалу та кореня

```
1. const InitialFunctionStep = 0.0;  
2. const IterationsLimit = 1024;
```

- 2) Описуємо функції які будуть використані в програмі,
GetFunctionResult - функція рівняння, FindFunctionInterval - функція для обчислення інтервалу в якому знаходиться корінь

```
1. function GetFunctionResult(root : double) : double;  
2. begin  
3.   GetFunctionResult := root * root * root - 4.0 * root;  
4. end;  
5.  
6. function FindFunctionInterval(stepSize : double) : double;  
7. begin  
8.   var a := InitialFunctionStep;  
9.   var instFunctionV := GetFunctionResult(a);  
10.  
11.   var step := a;  
12.  
13.   var i : integer;  
14.   for i := 0 to (IterationsLimit + 1) do  
15.     begin  
16.       if (GetFunctionResult(step) * instFunctionV < 0)  
17.         then  
18.           begin  
19.             FindFunctionInterval := step;  
20.             break;  
21.           end;  
22.       step := step + stepSize;  
23.     end;
```

24. **end;**

3) Цикл обрахунку кореня

```
1.    var c := 0.0;
2.    var iterationsCount := 0;
3.
4.    while True do
5.    begin
6.       c := (a + b) / 2.0;
7.
8.       if(Abs(GetFunctionResult(c)) <= tolerance)
9.           or (iterationsCount >= IterationsLimit) then
10.        begin
11.            Writeln('Root is ', c, ', computed in ',
iterationsCount, ' iterations');
12.            break;
13.        end;
14.
15.        if(GetFunctionResult(a) * GetFunctionResult(c) < 0.0)
then
16.            begin
17.               b := c
18.            end
19.        else
20.            begin
21.               a := c;
22.            end;
23.
24.        iterationsCount := iterationsCount + 1;
25.    end;
```

Тестування:

```
Equation is x^3 - 4x

Enter result value tolerance: 1e-2

Enter interval: 1 10

Root is: 1.99976
Computed in: 11 iterations

Enter result value tolerance: 1e-10

Enter interval: -1 -20

Root is: -2
Computed in: 24 iterations

Enter result value tolerance: 1e-1

Enter interval: 0 6

Root is: 0

Enter result value tolerance: 1e-25

Enter interval: 0 0

Root is: 0

Enter result value tolerance: 1e-3

Enter interval: -10 10

Root is: 0
Computed in: 0 iterations
```

Текст програми:

1. **uses**
2. `System.SysUtils;`
- 3.
4. *//Константа яка задає початкове значення для табуляції функції*
5. **const** InitialFunctionStep = 0.0;
- 6.

```

7. const IterationsLimit = 1024;
8.
9. function GetFunctionResult(root : double) : double;
10. begin
11.     GetFunctionResult := root * root * root - 4.0 * root;
12. end;
13.
14. function FindFunctionInterval(stepSize : double) : double;
15. begin
16.     var a := InitialFunctionStep;
17.     var instFunctionV := GetFunctionResult(a);
18.
19.     var step := a;
20.
21.     var i : integer;
22.     for i := 0 to (IterationsLimit + 1) do
23.         begin
24.             if (GetFunctionResult(step) * instFunctionV < 0)
25.             then
26.                 begin
27.                     FindFunctionInterval := step;
28.                     break;
29.                 end;
30.                 step := step + stepSize;
31.             end;
32.         end;
33.
34.     begin
35.         Writeln('Equation is x^3 - 4x', #13#10);
36.
37.         while True do
38.             begin
39.                 Writeln('Enter result value tolerance:');
40.
41.                 var tolerance := 1e-1;
42.                 Readln(tolerance);
43.
44.
45.                 Writeln('Enter interval:');
46.
47.                 var a := 0.0;
48.                 var b := 0.0;
49.                 Readln(a, b);
50.
51.                 //Перевірка чи в заданому інтервалі існують корені
52.                 if (GetFunctionResult(a) * GetFunctionResult(b) > 0.0)
53.                 then

```

```

53.      begin
54.          //Якщо інтервал є недійсним інтервал буде визначений
           методом табулювання функції
55.          a := InitialFunctionStep;
56.          b := FindFunctionInterval(1.0);
57.
58.          if(GetFunctionResult(a) * GetFunctionResult(b) >
0.0) then
59.              begin
60.                  Writeln('Coudlnt find any root in this
equation');
61.                  continue;
62.              end;
63.          end;
64.
65.          if(Abs(GetFunctionResult(a)) <= tolerance) then
66.              begin
67.                  Writeln('Root is ', a, #13#10);
68.                  continue;
69.              end
70.          else if(Abs(GetFunctionResult(b)) <= tolerance) then
71.              begin
72.                  Writeln('nRoot is ', b, #13#10);
73.                  continue;
74.              end;
75.
76.          var c := 0.0;
77.          var iterationsCount := 0;
78.
79.          while True do
80.              begin
81.                  c := (a + b) / 2.0;
82.
83.                  if(Abs(GetFunctionResult(c)) <= tolerance)
84.                      or (iterationsCount >= IterationsLimit) then
85.                      begin
86.                          Writeln('Root is ', c, ', computed in ',
iterationsCount, ' iterations');
87.                          break;
88.                      end;
89.
90.                  if(GetFunctionResult(a) * GetFunctionResult(c) < 0.0)
then
91.                      begin
92.                          b := c
93.                      end
94.                  else
95.                      begin

```

```

96.         a := c;
97.     end;
98.
99.     iterationsCount := iterationsCount + 1;
100. end;
101.
102.     Writeln(#13#10);
103. end;
104. end.

```

Додаткове завдання, програма на C++:

```

1. #include <iostream>
2.
3. //Константа яка задає початкове значення для табуляції
   функції
4. constexpr float IntialFunctionStep = 0.0f;
5.
6. constexpr int IterationsLimit = 1024;
7.
8. inline float GetFunctionResult(const float root)
9. {
10.     return root * root * root - 4.0f * root;
11. }
12.
13. inline std::pair<float, float> FindFunctionInterval(const
   float stepSize)
14. {
15.     const float a = IntialFunctionStep;
16.     const float instFunctionV = GetFunctionResult(a);
17.
18.     float step = a;
19.
20.     for(int i = 0; i <= IterationsLimit; ++i)
21.     {
22.         if (GetFunctionResult(step) * instFunctionV < 0)
23.             return { a, step };
24.
25.         step += stepSize;
26.     }
27. }
28.
29. int main()
30. {
31.     std::cout << "Equation is x^3 - 4x\n\n";
32.
33.     while (1)

```

```

34.     {
35.         std::cout << "\nEnter result value tolerance: ";
36.
37.         float tolerance = 1e-1f;
38.         std::cin >> tolerance;
39.
40.
41.         std::cout << "\nEnter interval: ";
42.
43.         float a = 0.0f;
44.         float b = 0.0f;
45.         std::cin >> a >> b;
46.
47.         //Перевірка чи в заданому інтервалі існують корені
48.         if (GetFunctionResult(a) * GetFunctionResult(b) >
49.             0.0f)
50.         {
51.             //Якщо інтервал є недійсним інтервал буде
52.             визначений методом табулювання функції
53.             auto interval = FindFunctionInterval(1.0f);
54.             a = interval.first;
55.             b = interval.second;
56.
57.             if (GetFunctionResult(a) *
58.                 GetFunctionResult(b) > 0.0f)
59.             {
60.                 std::cout << "\nOn this interval there
61.                 isn't any root or tolerance value is too small!\n\n";
62.
63.                 std::cin.clear();
64.
65.                 std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
66.                                 '\n');
67.
68.                 continue;
69.             }
70.         }
71.
72.         //Перевірка чи коренем може бути одне з чисел яке
73.         задає інтервал
74.         if (std::abs(GetFunctionResult(a)) <= tolerance)
75.         {
76.             std::cout << "\nRoot is: " << a << "\n\n";
77.             continue;
78.         }
79.         else if (std::abs(GetFunctionResult(b)) <=
80.             tolerance)

```



```

74.         {
75.             std::cout << "\nRoot is: " << b << "\n\n";
76.             continue;
77.         }
78.
79.
80.         float c = 0.0f;
81.         int iterationsCount = 0;
82.
83.         while (1)
84.         {
85.             c = (a + b) / 2.0f;
86.
87.             if (std::abs(GetFunctionResult(c)) <=
tolerance
88.                 || iterationsCount >= IterationsLimit)
89.             {
90.                 std::cout << "\nRoot is: " << c <<
"\nComputed in: " << iterationsCount << " iterations";
91.                 break;
92.             }
93.
94.             if (GetFunctionResult(a) *
GetFunctionResult(c) < 0.0f)
95.                 b = c;
96.             else
97.                 a = c;
98.
99.             iterationsCount++;
100.        }
101.
102.        std::cout << "\n\n";
103.    }
104.
105.    return(0);
106. }

```

Код лабораторної: <https://github.com/ptrstasiv/Lab3.git>

Висновок: при виконанні даної лабораторної роботи ми ознайомилися з особливостями чисельного розв'язання нелінійних рівнянь методом ділення навпіл. Реалізувавши цей алгоритм в середовищі Delphi.