Лабораторная работа № 1.4
«Вычисление корней нелинейного уравнения методом дихотомии»

Одной из основных проблем вычислительной математики является задача отыскания простых корней для трансцендентных уравнений вида $F(x) = 0$. Эта же задача может быть сформулирована как «*проблема поиска некратных нулей для некоторой функции* $F = F(x)$».

В классической теории приближенных вычислений известно множество разнообразных методов решения данной проблемы, учитывающих специфику и свойства функции $F(x)$.

Самым надежным и, пожалуй, самым простым с точки зрения организации вычислительных итераций, является ***метод дихотомии*** – метод деления отрезка пополам.

Цель лабораторной работы

Построить алгоритм и отладить программу определения корней заданного уравнения $F(x) = 0$ на интервале $x \in [x_0; x_n]$ с абсолютной погрешностью $\varepsilon \approx 10^{-12}$.

Решение данной задачи будем выполнять в два последовательных этапа.

Первый этап:

– разработка программной реализации для метода дихотомии;

– оформление этой реализации в виде библиотечного метода;

– тестирование метода на модельном трансцендентном уравнении.

Второй этап:

– использование метода дихотомии для вычисления всех нулей для таблицы заданной функции $F(x)$;

– расширение функционала для объекта класса `MyTableOfFunction` путем добавления в его состав членов и методов, оперирующих с нулями функции $F(x)$;

– тестирование второго этапа на таблице модельной функции.

Заключительный этап – выполнение вычислений по индивидуальному варианту данной Лабораторной работы.

Требования к программным компонентам

Разрабатываемые программные компоненты должны быть добавлены в Ваше рабочее пространство **MAC_Petrenko**. Программная реализация метода дихотомии должна входить в состав отдельного (нового) класса **MAC_Equations** динамической библиотеки **MAC_DLL**.

Основная программная единица данной лабораторной работы должна быть генерирована в рамках отдельного проекта **Console Application** с именем **MAC_LabWork_1_4**.

Порядок выполнения задания лабораторной работы на языке C#

В соответствии с задачами первого этапа включим в структуру динамической библиотеки **MAC_DLL** новый класс, который будет содержать программные реализации методов уточнения корней нелинейных уравнений, построенных на основе различных численных алгоритмов.

Этот класс будем именовать **MAC_Equations**.

Алгоритм метода дихотомии **Dichotomy()**, который мы первым включим в класс **MAC_Equations**, подробно рассматривался на лекционном занятии и его можно повторить с использованием имеющегося у Вас конспекта в электронной форме.

Обязательными параметрами этого метода должны быть:

- границы интервала $[x_L; x_R]$, на концах которого функция $F(x)$ имеет разные знаки, т.е. обязательно должно выполняться неравенство $F(x_L) \cdot F(x_R) < 0$;
- делегат, имеющий сигнатуру функции $F(x)$ от одной действительной переменной x;
- ε – абсолютная погрешность вычисления искомого решения (корня уравнения);
- K – количество выполненных итераций.

Разрабатываемый метод должен возвращать в основную программу значение $x_*^{(K)}$, которое удовлетворяет условию $|F(x_*^{(K)})| \le \varepsilon$, где K – номер последней итерации.

Для предотвращения возможных зацикливаний в алгоритме метода дихотомии следует ограничить количество итераций K двумя дополнительными условиями (выход из цикла при выполнении следующих условий): а) $K > 70$; б) $10 \cdot \left(x_R^{(K)} - x_L^{(K)}\right) \le \varepsilon$.

В коде метода **Dichotomy()**, что приводится далее, использован специальный алгоритм вычисления среднего арифметического для двух действительных чисел с учетом их знаков.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Fx = MAC_DLL.Function_of_x;

namespace MAC_DLL
{
  public class MAC_Equations
  {
    public static double
      Dichotomy(double a, double b, double eps, Fx f, ref int K)
    {
      double fa = f(a), fc, c = 0.0; K = 0;
      while (K < 70)
      {
        if ((a * b) < 0) c = (a + b) * 0.5; else c = a + (b - a) * 0.5;
        fc = f(c); K++;
        if ((fa * fc) < 0) b = c; else a = c;
        if ((Math.Abs(fc) < eps) || ((b - a) * 10 < eps)) break;
      }
      return c;
    }
  }
}
```

Итак, метод дихотомии добавлен в новый класс **MAC_Equations**, и мы можем приступать к его тестированию в рамках консольного приложения **MAC_LabWork_1_4.**

Выполним стандартные действия по взаимодействию проектов **MSVS** в рабочем пространстве **MAC_Petrenko** и добавим ссылку на библиотеку **MAC_DLL** в разделе **References** нового проекта **MAC_LabWork_1_4**.

Для дальнейшей работы нам понадобится простое нелинейное уравнение $F(x) = 0$ с достаточно простой функцией $F(x)$ и известным (очевидным) решением (тестовое уравнение).

При этом желательно, чтобы корень этого уравнения выражался достаточно простым числом, например, целым числом, или простой десятичной дробью.

Пусть нами выбрано следующее тестовое уравнение:

$$\cos(\pi x) = 0\,. \tag{4.1}$$

Это тригонометрическое уравнение имеет бесконечное множество простых корней, которые являются рациональными числами и описываются формулой

$$x_m = \frac{m}{2}, \text{ где } m \in Z\,. \tag{4.2}$$

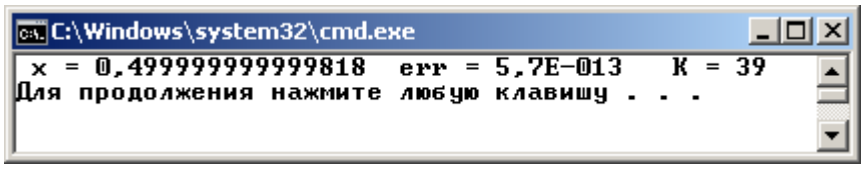
Нас будет интересовать первый положительный корень $x_* = 0.5$ этого уравнения.

Этот корень, очевидно, принадлежит интервалу $[\,0.3, 0.6\,]$, который мы можем использовать в качестве интервала $[x_L; x_R]$, содержащего изолированный простой корень x_*.

Выполним построение кода основного приложения **`MAC_LabWork_1_4`**, в котором вычислим с заданной погрешностью этот корень с использованием имеющегося библиотечного метода **`Dichotomy()`**.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using CLE = MAC_DLL.MAC_Equations;
7
8   namespace MAC_LabWork_1_4
9   {
10    class Main_LW_1_4
11    {
12      static void Main(string[] args)
13      {
14        Test_LW_14_1();
15      }
16
17      static void Test_LW_14_1()
18      {
19        int k = 0; string frmt = " x = {0:F15}    err = {1:E1}    K = {2}";
20        double root = CLE.Dichotomy(0.3, 0.6, 1.0E-12, Cos_pi_x, ref k);
21        Console.WriteLine(frmt, root, Cos_pi_x(root), k);
22      }
23      static double Cos_pi_x(double x)
24      {
25        return Math.Cos(Math.PI * x);
26      }
27    }
28  }
```

Результат выполнения теста:



Анализ результата свидетельствует о том, что разработанный нами алгоритм метода дихотомии затратил 39 шагов итераций на уточнение искомого корня x_* до значения, обеспечивающего абсолютную погрешность 10^{-12}.

Такой результат можно считать вполне удовлетворительным и первый этап Лабораторной работы 1.4 успешно завершенным.

Переходим ко второму этапу Лабораторной работы 1.4.

Суть этого этапа нам поможет уяснить следующая задача.

Пусть нам задано трансцендентное уравнение (4.3), для которого необходимо: сначала выяснить само наличие интервалов, содержащих корни уравнения, на заданном отрезке изменения аргумента $x \in [\,0.0, 15.0\,]$, а затем уточнить все эти корни с погрешностью 10^{-12}:

$$F(x) = \mathrm{th}(x) - 2\cos(\sqrt{10}x) = 0\,. \qquad (4.3)$$

Для выполнения этой задачи мы должны будем использовать объект класса **`MyTableOfFunction`**, который представляет собой упорядоченную таблицу функции $F(x)$, вычисленную в соответствии с ее функциональным представлением (4.3).

Экземпляр класса **`MyTableOfFunction`** (как наследник класса **`MyTable`**) уже имеет свойство **`Roots`** – список объектов типа **`Root`**, представляющих собой интервалы, на концах которых заданная функция $F(x)$ меняет свой знак:

```
40  public class Root
41  {
42      public double xL, xR, x, err; public int iters;
43      public Root(double x_left, double x_right)
44      {
45          xL = x_left; xR = x_right;
46          x = double.NaN; err = double.NaN; iters = 0;
47      }
48      public string ToPrint()...
56  }
```

Каждый элемент списка **`Roots`** имеет поле **`Roots.x`**, предназначенное для «хранения» абсциссы нуля функции (корня уравнения), соответствующего данному интервалу **`[xL,xR]`**.

Поле **`Roots.err`** предназначено для записи погрешности, допущенной при вычислении этого нуля, а поле **`Roots.iters`** – для количества итераций, выполненных конкретным численным методом для уточнения этого нуля.

Следовательно, нам необходимо в рамках класса **`MAC_Equations`** переопределить метод дихотомии так, чтобы он мог «работать» с объектом типа **`Root`** и определять для заданной функции $F(x)$ корень уравнения с требуемой точностью на «своем» интервале **`[xL,xR]`**.

Имеющийся и уже отлаженный алгоритм этого метода позволит Вам без особых усилий выполнить эту задачу.

Далее приводится только сигнатура заголовка перегруженного метода:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using Fx = MAC_DLL.Function_of_x;
7
8   namespace MAC_DLL
9   {
10    public class MAC_Equations
11    {
12      public static double
13        Dichotomy(double a, double b, double eps, Fx f, ref int K)...
25
26      public static void Dichotomy(Fx f, Root root, double eps)
27      {
28        double a = root.xL, b = root.xR, c = 0.0, fc, fa = f(a);
29        root.iters = 0;
30        while (root.iters < 70)...
38        root.x = c;  return;
39      }
40    }
41  }
```

Далее. В код класса **MyTableOfFunction** добавляем регион «Методы обработки таблицы функции».

В этом регионе формируем новый метод, выполняющий уточнение всех нулей заданной функции перегруженным методом дихотомии:

```
208   public class MyTableOfFunction : MyTable
209   {
210     readonly Function_of_x Fx; // Указатель на функцию f(x)
211
212     public MyTableOfFunction(double xo, double xn, int n,
213                              Function_of_x f_x, string title)...
230
231     <--- Переопределение методов класса MyTable --->
237
238     #region    <--- Методы обработки таблицы функции --->
239
240     public void Roots_Correction(double eps)
241     {
242       if (Roots != null)
243       {
244         foreach (Root root in Roots)
245           MAC_Equations.Dichotomy(Fx, root, eps);
246       }
247     }
248     #endregion  <--- Методы обработки таблицы функции --->
249   }
```

Возвращаемся к основному приложению **MAC_LabWork_1_4** данной Лабораторной работы. Добавим новый тестирующий метод, в котором будет производиться:

1) генерация таблицы функции (4.3) на общем интервале **[** 0.0**,** 15.0 **]**;
2) уточнение всех нулей функции с помощью библиотечного метода дихотомии с погрешностью 10^{-12} и вывод результатов в текстовый файл для анализа.

Помимо этого, добавим в основное приложение статический метод, вычисляющий значения функции (4.3) и соответствующий сигнатуре делегата **Function_of_x**:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using CLE = MAC_DLL.MAC_Equations;
7   using CTF = MAC_DLL.MyTableOfFunction;
8
9   namespace MAC_LabWork_1_4
10  {
11    class Main_LW_1_4
12    {
13      static void Main(string[] args)
14      {
15        Test_43(1.0E-12);
16        // Test_LW_14_1();
17      }
18
19      static void Test_43(double epsilon)
20      {
21        CTF table_43 = new CTF(0.0, 15.0, 300, F_43, "Test_43");
22        table_43.Roots_Correction(epsilon);
23        table_43.To_txt_File("MAC_LW_1_4.txt", " Test F_43");
24      }
25
26      static double F_43(double x)
27      {
28        return Math.Tanh(x) - 2.0 * Math.Cos(Math.Sqrt(10.0) * x);
29      }
30
31      static void Test_LW_14_1()...
37      static double Cos_pi_x(double x)...
41    }
42  }
```

Результат выполнения приложения **MAC_LabWork_1_4** должен быть следующим (нижняя часть таблицы функции):

```
    282  (  14,1000000000,   -0,6440733118  )
    283  (  14,1500000000,   -0,4442435090  )
    284  (  14,2000000000,   -0,2083827767  )
    285  (  14,2500000000,    0,0576246408  )
    286  (  14,3000000000,    0,3471424011  )
    287  (  14,3500000000,    0,6529476267  )
    288  (  14,4000000000,    0,9674111010  )
    289  (  14,4500000000,    1,2826876019  )
    290  (  14,5000000000,    1,5909116238  )
    291  (  14,5500000000,    1,8843936061  )
    292  (  14,6000000000,    2,1558117721  )
    293  (  14,6500000000,    2,3983947921  )
    294  (  14,7000000000,    2,6060907147  )
    295  (  14,7500000000,    2,7737179504  )
    296  (  14,8000000000,    2,8970945415  )
    297  (  14,8500000000,    2,9731424937  )
    298  (  14,9000000000,    2,9999645659  )
    299  (  14,9500000000,    2,9768916019  )
    300  (  15,0000000000,    2,9044992252  )

     x = [   0,000000000000 :  15,000000000000 ]
     x_Reg =    15,000000000000

    Min  (   0,000000000000,   -2,000000000000  )
    Max  (  14,900000000000,    2,999964565856  )
     f_Reg =     4,999964565856

    Таблица нулей Test_43 функции:
     0  [   0,40000,   0,45000  ]  root =  0,431934615246  err =  8,3E-013  iters = 36
     1  [   1,60000,   1,65000  ]  root =  1,642743997228  err =  6,9E-013  iters = 35
     2  [   2,30000,   2,35000  ]  root =  2,321541775472  err =  8,5E-013  iters = 37
     3  [   3,60000,   3,65000  ]  root =  3,642432159058  err =  6,6E-014  iters = 37
     4  [   4,30000,   4,35000  ]  root =  4,305054782074  err =  2,9E-013  iters = 36
     5  [   5,60000,   5,65000  ]  root =  5,629595311224  err =  5,1E-013  iters = 37
     6  [   6,25000,   6,30000  ]  root =  6,291907153042  err =  5,1E-013  iters = 37
     7  [   7,60000,   7,65000  ]  root =  7,616517581963  err =  9,3E-014  iters = 37
     8  [   8,25000,   8,30000  ]  root =  8,278823578358  err =  9,6E-013  iters = 22
     9  [   9,60000,   9,65000  ]  root =  9,603435321939  err =  5,7E-013  iters = 35
    10  [  10,25000,  10,30000  ]  root = 10,265741208432  err =  8,0E-013  iters = 37
    11  [  11,55000,  11,60000  ]  root = 11,590352976731  err =  6,6E-013  iters = 37
    12  [  12,25000,  12,30000  ]  root = 12,252658861157  err =  9,3E-013  iters = 34
    13  [  13,55000,  13,60000  ]  root = 13,577270629921  err =  8,0E-013  iters = 37
    14  [  14,20000,  14,25000  ]  root = 14,239576514308  err =  8,6E-014  iters = 37
```

Математический результат проделанной работы можно сформулировать так:

На интервале $[\,0,15\,]$ *тестовая функция (4.3) имеет пятнадцать нулей, которые нам удалось вычислить методом дихотомии с абсолютной погрешностью* 10^{-12}.