Лабораторная работа № 1.5
«Исследование сходимости итерационных схем метода Ньютона»

Основной характеристикой методов решения нелинейных уравнений $F(x)=0$ является количество итераций, затрачиваемых этими численными алгоритмами на уточнение корня.

Очевидно, что методы, имеющие наибольшую скорость сходимости, требуют обязательного выполнения некоторых дополнительных условий для функции $F(x)$.

И наоборот, методы с малой скоростью сходимости (например, дихотомии), как правило, никаких дополнительных условий на функцию $F(x)$ не накладывают.

В классической теории приближенного решения нелинейных уравнений известно семейство алгоритмов, называемых методами касательных (или методами Ньютона).

Эти методы обладают самой высокой скоростью сходимости по отношению к другим итерационным алгоритмам. Подтвердим этот факт в рамках данной лабораторной работы.

Цель лабораторной работы

Построить и реализовать алгоритмы для двух итерационных схем из семейства методов касательных и отладить консольную программу определения корней тестового уравнения $F(x)=0$ на интервале $x \in [x_0; x_n]$ с абсолютной погрешностью $\varepsilon \approx 10^{-12}$.

Провести сравнительный анализ скорости сходимости обеих схем (количество затрачиваемых итераций) на заданном тестовом уравнении $F(x)=0$.

***Итерационная схема* 1**:
$$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})}, \qquad (1.5.1)$$

где в качестве нулевого приближения $x^{(0)}$ к искомому корню выбирается та из границ интервала $[x_{i-1}; x_i]$, для которой выполняется условие $F(x^{(0)}) \cdot F''(x^{(0)}) > 0$.

***Итерационная схема* 2**:
$$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(0)})},$$

в ее упрощенном варианте
$$x^{(k+1)} = x^{(k)} - \delta_0 \cdot F(x^{(k)}), \qquad (1.5.2)$$

где $\delta_0 = \dfrac{b-a}{F(b)-F(a)} \approx \left[F'(x^{(0)})\right]^{-1}$, с начальным приближением $x^{(0)} = (b+a)/2$. (1.5.3)

Для обеих указанных схем условием завершения итерационного процесса следует считать выполнение неравенства $\left|F(x^{(k)})\right| \le \varepsilon$.

Для предотвращения возможных зацикливаний следует добавить условия выхода из итерационного процесса: при $k > 15$ для схемы 1 и при $k > 25$ для схемы 2.

Решение поставленной задачи следует выполнять в два этапа.

Первый этап – включение новых методов в состав класса **`Class_Equations`** динамической библиотеки **`MAC_DLL`**, которые будут содержать программные реализации схем уточнения корней нелинейных уравнений, построенных на основе алгоритмов (1.5.1) и (1.5.2)–(1.5.3). Оба алгоритма можно объединить под одним именем **`Tangent()`** и использовать перегрузку, поскольку программные реализации для схем (1.5.1) и (1.5.2)–(1.5.3), очевидно, должны иметь разные наборы входных параметров.

Обязательными параметрами метода, реализующего ***схему (1.5.1)***, должны быть:

- границы интервала $[x_{i-1}; x_i]$, для которого выполняется условие $F(x_{i-1}) \cdot F(x_i) < 0$;
- делегаты **`Function_of_x`**, имеющие сигнатуру функции от действительной переменной, для представления математических функций $F(x)$, $F'(x)$ и $F''(x)$;
- ε – заданная абсолютная погрешность уточнения корня;
- K – количество выполненных итераций.

Обязательными параметрами метода, реализующего ***схему (1.5.2)***, должны быть:

- границы интервала $[x_{i-1}; x_i]$, для которого выполняется условие $F(x_{i-1}) \cdot F(x_i) < 0$;
- делегат **`Function_of_x`**, имеющий сигнатуру функции от действительной переменной, для представления математической функции $F(x)$;
- ε – заданная абсолютная погрешность уточнения корня;
- K – количество выполненных итераций.

Результатом вызова обоих методов в основном консольном приложении должно быть уточненное значение корня $\tilde{x}^{(K)}$ уравнения $F(x) = 0$, удовлетворяющее условиям $x_{i-1} \le \tilde{x}^{(\mathrm{K})} \le x_i$ и $\mid F(x^{(\mathrm{K})}) \mid \le \varepsilon$.

Второй этап (основная программа) – этап вычисления списка всех корней нелинейного уравнения $F(x) = 0$ на заданном интервале аргумента $x \in [x_0; x_n]$.

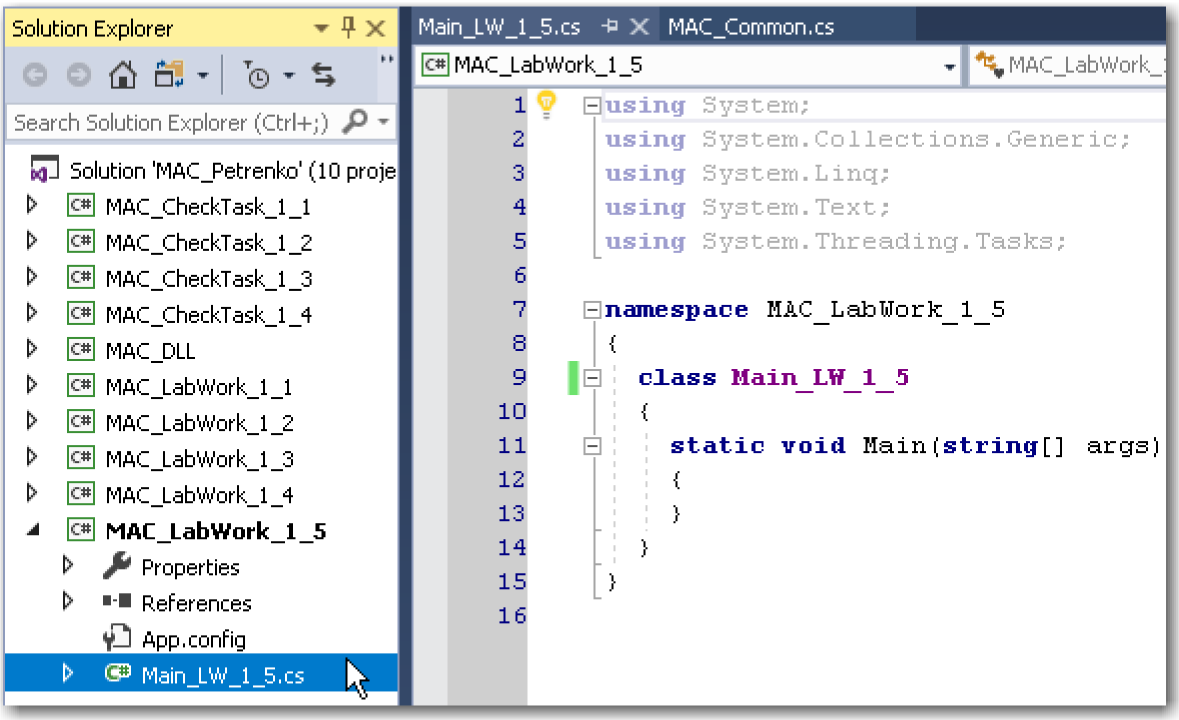
Уточнение каждого корня $\tilde{x}$ производится заданным численным методом.

Этот этап выполняется практически так же, как и в предыдущей Лабораторной работе.

Требования к программным компонентам

Разрабатываемые программные компоненты должны быть реализованы в составе имеющейся динамической библиотеки **MAC_DLL**, проект которой уже размещен в Вашем рабочем пространстве **MAC_Petrenko**.

Основная программная единица данной лабораторной работы должна быть генерирована в рамках отдельного проекта (**Console Application** с именем **MAC_LabWork_1_5** в рабочем пространстве **MAC_Petrenko**), имеющего доступ к динамической библиотеке, в которой определены классы **MAC_MyTableOfFunction** и **MAC_Equations**.



Результат выполнения основной программы – две таблицы корней тестового уравнения – должны сохраняться в текстовом файле **MAC_LabWork_1_5.txt**, который должен быть включен в проект **MAC_LabWork_1_5**.

Порядок выполнения задания лабораторной работы на языке C#

Начнем с разработки программных реализаций алгоритмов (1.5.1) и (1.5.2)–(1.5.3) уточнения корней для нелинейного уравнения $F(x) = 0$.

Эти методы добавим в уже имеющийся класс **Class_Equations** динамической библиотеки **MAC_DLL**.

Первый метод, именуемый **Tangent()** будет соответствовать алгоритму (1.5.1), а второй – уже перегруженный – будет выполнять итерации согласно схеме (1.5.2)–(1.5.3).

```
public class MAC_Equations
{
    public static double
      Dichotomy(double a, double b, double eps, Fx f, ref int K)...

    public static void Dichotomy(Fx f, Root root, double eps)...

    public static double
      Tangent(Fx Fx, Fx D1Fx, Fx D2Fx,
              double xL, double xR, double eps, out int K)
    {
      double xK = (xR + xL) * 0.5; K = 0;
      if ((Fx(xL) * D2Fx(xL)) > 0) xK = xL;
      if ((Fx(xR) * D2Fx(xR)) > 0) xK = xR;
      while (Math.Abs(Fx(xK)) > eps)
      {
        xK = xK - Fx(xK) / D1Fx(xK); K++; if (K > 15) break;
      }
      return xK;
    }

    public static double
      Tangent(Fx Fx, double xL, double xR, double eps, out int K)
    {
      double xK = (xR + xL) * 0.5; K = 0;
      double dF = (Fx(xR) - Fx(xL)) / (xR - xL);
      while (Math.Abs(Fx(xK)) > eps)
      {
        xK = xK - Fx(xK) / dF; K++; if (K > 25) break;
      }
      return xK;
    }
}
```

Теперь переходим к основному консольному приложению Лабораторной работы.

Задача этой программы – тестирование методов **`Tangent()`** на модельном уравнении:

$$F(x) = \mathrm{th}(x) - 2\cos(\sqrt{10}\cdot x) = 0, \;\; [x_0 ; x_n] = [\;0\;;15\;], \;\; \varepsilon \approx 10^{-12}. \qquad (1.5.4)$$

Это уравнение нам удобно тем, что мы уже вычислили его корни в ходе выполнения предыдущей Лабораторной работы и нам есть с чем сравнивать новые результаты.

Для реализации вызова схемы (1.5.1) помимо функции $F(x)$ потребуются первая $F'(x)$ и вторая $F''(x)$ производные в виде соответствующих статических методов в классе основного приложения. Поэтому предварительно находим выражения для функций $F'(x)$ и $F''(x)$:

$$F'(x) = \mathrm{ch}^{-2}(x) + 2\sqrt{10}\sin(\sqrt{10}\cdot x), \quad F''(x) = 2\cdot\left(10\cos(\sqrt{10}\cdot x) - \mathrm{th}(x)\,\mathrm{ch}^{-2}(x)\right). \qquad (1.5.5)$$

Выполним «подготовительную работу»:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MyTF = MAC_DLL.MyTableOfFunction;
using Eq = MAC_DLL.MAC_Equations;
using System.IO;

namespace MAC_LabWork_1_5
{
    class Main_LW_1_5
    {
        static void Main(string[] args)...

        public static double Fx(double x)
        {
            return Math.Tanh(x) - 2.0 * Math.Cos(Math.Sqrt(10.0) * x);
        }
        public static double d1F(double x)
        {
            return 1.0 / Math.Cosh(x) / Math.Cosh(x) +
                   2.0 * Math.Sqrt(10.0) * Math.Sin(Math.Sqrt(10.0) * x);
        }
        public static double d2F(double x)
        {
            return 2.0 * (10.0 * Math.Cos(Math.Sqrt(10.0) * x) -
                          Math.Tanh(x) / Math.Cosh(x) / Math.Cosh(x));
        }
    }
}
```

В функции **Main()** класса **MAC_LabWork_1_5** выполним следующие действия:

1) создадим объект **T_Fx** класса **MAC_MyTableOfFunction**, соответствующий таблице функции (1.5.4);

2) выполним уточнение нулей для полученной таблицы функции с использованием метода дихотомии и запишем в файл результатов **MAC_LabWork_1_5.txt** соответствующую таблицу **Table_of_Roots()**;

3) воспользуемся списком **Roots** таблицы **T_Fx** для организации циклов вычислений тех же нулей, но с использованием разработанных методов **Tangent()**;

4) результаты будем дописывать в файл **MAC_LabWork_1_5.txt**.

Код основной функции **Main()** может быть следующим:

```
static void Main(string[] args)
{
  StreamWriter SW = new StreamWriter("MAC_LabWork_1_5.txt");
  MyTF T_Fx = new MyTF(0.0, 15.0, 500, Fx, "Fx");
  T_Fx.Roots_Correction(1.0E-12);
  SW.Write(T_Fx.Table_of_Roots("- Dichotomy -"));

  int K, M = T_Fx.Roots.Count;
  double xa, xb, xr = double.NaN, fr, eps = 1.0E-12;

  SW.WriteLine("\r\n Таблица нулей, вычисленная по схеме (1.5.1):");
  for (int j = 0; j < M; j++)
  {
    xa = T_Fx.Roots[j].xL; xb = T_Fx.Roots[j].xR;
    xr = Eq.Tangent(Fx, d1F, d2F, xa, xb, eps, out K);
    fr = Math.Abs(Fx(xr));
    SW.WriteLine(string.Format("{0,3}", j) +
                 string.Format("{0,17:F12}", xr) +
                 string.Format("{0,10:E1}", fr) +
                 string.Format("   {0}", K));
  }

  SW.WriteLine("\r\n Таблица нулей, вычисленная по схеме (1.5.2):");
  for (int j = 0; j < M; j++)
  {
    xa = T_Fx.Roots[j].xL; xb = T_Fx.Roots[j].xR;
    xr = Eq.Tangent(Fx, xa, xb, eps, out K);
    fr = Math.Abs(Fx(xr));
    SW.WriteLine(string.Format("{0,3}", j) +
                 string.Format("{0,17:F12}", xr) +
                 string.Format("{0,10:E1}", fr) +
                 string.Format("   {0}", K));
  }
  SW.Close();
}
```

Файл результатов должен содержать следующие числовые данные:

```
 - Dichotomy -
  Таблица нулей Fx функции:
   0  [   0,42000,   0,45000  ]  root =  0,431934615246  err =  2,0E-013  iters = 35
   1  [   1,62000,   1,65000  ]  root =  1,642743997228  err =  9,8E-014  iters = 36
   2  [   2,31000,   2,34000  ]  root =  2,321541775473  err =  3,6E-013  iters = 34
   3  [   3,63000,   3,66000  ]  root =  3,642432159058  err =  6,6E-014  iters = 37
   4  [   4,29000,   4,32000  ]  root =  4,305054782074  err =  1,3E-013  iters = 37
   5  [   5,61000,   5,64000  ]  root =  5,629595311224  err =  8,9E-013  iters = 36
   6  [   6,27000,   6,30000  ]  root =  6,291907153042  err =  6,8E-013  iters = 33
   7  [   7,59000,   7,62000  ]  root =  7,616517581963  err =  3,0E-013  iters = 36
   8  [   8,25000,   8,28000  ]  root =  8,278823578358  err =  1,8E-013  iters = 36
   9  [   9,60000,   9,63000  ]  root =  9,603435321939  err =  1,7E-013  iters = 37
  10  [  10,26000,  10,29000  ]  root = 10,265741208432  err =  3,8E-013  iters = 35
  11  [  11,58000,  11,61000  ]  root = 11,590352976731  err =  2,8E-013  iters = 33
  12  [  12,24000,  12,27000  ]  root = 12,252658861157  err =  9,8E-013  iters = 34
  13  [  13,56000,  13,59000  ]  root = 13,577270629921  err =  3,9E-013  iters = 36
  14  [  14,22000,  14,25000  ]  root = 14,239576514308  err =  7,4E-014  iters = 37

  Таблица нулей, вычисленная по схеме (1.5.1):
   0   0,431934615246  1,7E-016  3
   1   1,642743997228  5,8E-014  3
   2   2,321541775473  1,9E-014  3
   3   3,642432159058  1,9E-015  3
   4   4,305054782074  7,2E-015  3
   5   5,629595311224  4,0E-014  3
   6   6,291907153042  7,8E-016  3
   7   7,616517581963  3,9E-013  3
   8   8,278823578358  5,3E-015  3
   9   9,603435321939  5,3E-015  3
  10  10,265741208432  2,0E-013  3
  11  11,590352976731  6,9E-015  3
  12  12,252658861157  3,9E-015  3
  13  13,577270629921  2,9E-014  3
  14  14,239576514308  0,0E+000  3

  Таблица нулей, вычисленная по схеме (1.5.2):
   0   0,431934615246  1,1E-014  4
   1   1,642743997228  5,8E-014  6
   2   2,321541775473  4,9E-014  5
   3   3,642432159058  2,3E-014  5
   4   4,305054782074  2,6E-014  3
   5   5,629595311223  3,5E-013  5
   6   6,291907153042  1,2E-013  6
   7   7,616517581963  3,1E-014  7
   8   8,278823578358  4,4E-013  7
   9   9,603435321939  1,0E-013  7
  10  10,265741208432  3,8E-013  6
  11  11,590352976731  8,2E-013  5
  12  12,252658861157  3,9E-015  5
  13  13,577270629921  4,6E-015  5
  14  14,239576514308  7,4E-013  5
```

Таблица корней тестового уравнения (1.5.4), полученная в результате применения метода Ньютона по схеме (1.5.1), демонстрирует сходимость к корню за 3 итерации (всего 45 итераций при последовательном расчете 15 корней).

При этом достигнутая точность вычисления корня оказалась выше затребованной.

Схема Ньютона (1.5.2)–(1.5.3) показала «в среднем» в 1.8 раза больше итераций на каждом из корней, а общее число их составило 81. При этом длина подотрезка $[x_{i-1}; x_i]$, на котором производилось уточнение корней, была одна и та же для обеих схем.

Метод дихотомии, тестированный нами на этом же уравнении, показал в среднем по 35 итераций на корень при той же длине подотрезка $[x_{i-1}; x_i]$.

Это больше чем в 10 раз по сравнению со схемой (1.5.1) метода касательных, и больше чем в 6 раз по отношению к схеме (1.5.2)–(1.5.3).

Итак, новые методы решения нелинейного уравнения $F(x) = 0$ отлажены и протестированы.

После того, как тестовые вычисления дали требуемый результат, Вам следует произвести аналогичные вычисления для Вашего персонального варианта Лабораторной работы 1.5.

Отличия будут заключаться не только в математическом виде самой функции $F(x)$ и двух ее производных.

На предлагаемом Вам интервале $x \in [x_0; x_n]$ функция $F(x)$ будет иметь только один нуль x_*. Поэтому этот корень уравнения $F(x) = 0$ будет и «первым» и «старшим».

Для найденного значения этого корня x_* следует дополнительно вычислить значения функций $F'(x_*)$ и $F''(x_*)$, и свести эти результаты в «единую таблицу».

Бланки индивидуальных заданий по данной лабораторной работе выдаются преподавателем после выполнения Вами тестовой части задания.

Рекомендация. При реализации индивидуального задания следует ***обязательно сохранить*** код основного приложения, разработанный для тестовых уравнений.