# Методика выполнения контрольного задания № 1.5
# «Вычисление корней нелинейного уравнения методом Ньютона (касательных)»

## Постановка задачи

Построить алгоритм и отладить консольное **Windows**–приложение, предназначенное для решения следующей практической задачи (численный эксперимент).

Пусть известны уравнения траекторий $y_1 = f_1(x)$ и $y_2 = f_2(x)$ для двух материальных частиц, одновременно движущихся в плоскости $Oxy$. Необходимо:

1. выяснить, существуют ли точки пересечения у этих траекторий на заданном интервале изменения пространственной координаты $x \in [x_0, x_n]$ (этап отделения корней);

2. при помощи метода Ньютона (схема (1.5.1) из Лабораторной работы 1.5) с абсолютной погрешностью $\varepsilon \sim 10^{-10}$ определить приближенные решения $\tilde{x}_i$ уравнения $f_1(x) = f_2(x)$ на частичных отрезках $[x_{i-1}, x_i]$, найденных на предыдущем этапе;

3. вычислить ординаты $\tilde{y}_i = f_1(\tilde{x}_i) = f_2(\tilde{x}_i)$ точек пересечения траекторий и занести числовые результаты в таблицу с десятью цифрами после десятичной точки.

Для решения поставленной задачи следует воспользоваться уже имеющимися программными компонентами из библиотеки **MAC_DLL** и при необходимости дополнить их.

## Предварительные замечания

Прежде чем приступать к непосредственному программированию, следует привести данную математическую задачу к форме, допускающей применение численного метода касательных (схема (1.5.1)) к ее решению.

Заданное уравнение пересекающихся траекторий $f_1(x) = f_2(x)$ заменяем ему эквивалентным уравнением $F(x) = f_1(x) - f_2(x) = 0$.

Теперь мы можем анализировать таблицу функции $F(x)$ на промежутке $[x_0, x_n]$ на предмет наличия интервалов $[x_{i-1}, x_i]$, содержащих простые нули этой функции.

Поскольку нам предстоит применять метод касательных по схеме (1.5.1), следует предварительно аналитическим путем получить математические выражения (формулы) для производных $F'(x)$ и $F''(x)$. Очевидно, что $F'(x) = f_1'(x) - f_2'(x)$ и $F''(x) = f_1''(x) - f_2''(x)$.

## Требования к программным компонентам

Основная расчетная программа должна быть разработана в виде отдельного проекта **MAC_CheckTask_1_5** консольного приложения **Win32** в имеющемся рабочем пространстве **MAC_Petrenko**.

Собственно создание таблицы функции $F(x)$, должно быть реализовано в экземпляре нового класса **MyExtendedTableOfFunction** (расширенная таблица функции), наследника от класса **MyTableOfFunction**.

В конструкторе нового класса имеются два дополнительных формальных параметра – делегаты типа **Function_of_x**, предназначенные для передачи указателей на функции, вычисляющие первую и вторую производные от основной функции $F(x)$.

Метод уточнения нулей – новый перегруженный метод **Tangent()** из класса **My_Equations**, реализованный по схеме (1.5.1), сигнатура которого будет раскрыта далее.

## Пример выполнения варианта задания на языке C#

В имеющемся рабочем пространстве **MAC_Petrenko** сгенерируем (добавим) новый проект **MAC_CheckTask_1_5** консольного приложения. Добавляем в раздел **References** проекта **MAC_CheckTask_1_5** ссылку на проект **MAC_DLL**.

Пусть имеется следующая математическая постановка задачи:

Уравнения траекторий:

$$y_1 = f_1(x) = \mathrm{th}(x + a)$$
$$y_2 = f_2(x) = 2\cos(\sqrt{10} \cdot x + b)$$

Таблица параметров задания:

| $x_0$ | $x_n$ | $a$ | $b$ |
|---|---|---|---|
| **0.0** | **4.0** | **0.1** | **-0.1** |

Исследуемая функция $F(x)$ представляет собой разность $F(x) = f_1(x) - f_2(x)$ двух заданных функций с параметрами.

То же самое будет касаться и ее первой $F'(x)$ и второй $F''(x)$ производных.

Найдем математические выражения для этих производных:

$$f_1'(x) = \mathrm{ch}^{-2}(x + a), \qquad f_1''(x) = 2 \cdot \mathrm{th}(x + a) \cdot \mathrm{ch}^{-2}(x + a),$$
$$f_2'(x) = -2\sqrt{10} \cdot \sin(\sqrt{10}x + b), \quad f_2''(x) = -20 \cdot \cos(\sqrt{10}x + b).$$

В основной программе следует объявить и проинициализировать параметры задачи (как статические переменные), а также создать коды математических функций, которые будут использованы в вычислениях.

Первоначальная структура приложения **MAC_CheckTask_1_5** может быть следующей:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using MyETF = MAC_DLL.MyExtendedTableOfFunction;
7   using System.IO;
8
9   namespace MAC_CheckTask_1_5
10  {
11    class Main_CT_1_5
12    {
13      static double a, b, s;
14
15      static void Main(string[] args)
16      {
17        a = 0.1; b = -0.1; s = Math.Sqrt(10.0);
18        StreamWriter SW = new StreamWriter("MAC_CheckTask_1_5.txt");
19
20        <-- Решение поставленной задачи -->
36
37        SW.Close();
38      }
39
40      public static double Fx(double x) { return f1(x) - f2(x); }
41      public static double d1Fx(double x) { return d1f1(x) - d1f2(x); }
42      public static double d2Fx(double x) { return d2f1(x) - d2f2(x); }
43
44      public static double f1(double x)
45      { return Math.Tanh(x + a); }
46      public static double d1f1(double x)
47      { return 1.0 / Math.Cosh(x + a) / Math.Cosh(x + a); }
48      public static double d2f1(double x)
49      { return -2.0 * Math.Tanh(x + a) / Math.Cosh(x + a) / Math.Cosh(x + a); }
50
51      public static double f2(double x)
52      { return 2.0 * Math.Cos(x * s + b); }
53      public static double d1f2(double x)
54      { return -2.0 * s * Math.Sin(x * s + b); }
55      public static double d2f2(double x)
56      { return -2.0 * s * s * Math.Cos(x * s + b); }
57    }
58  }
```

Теперь займемся формированием нового класса **MyExtendedTableOfFunction** – расширенной таблицы функции.

Сначала добавим в определения абстрактного класса **MyTable** виртуальный метод **Roots_Correction()**, который мы будем переопределять в классах-наследниках:

```
58   public abstract class MyTable
59   {
60     <--- Основные Свойства MyTable --->
90
91     <--- Дополнительные Свойства MyTable --->
96
97     <--- Основные Методы MyTable --->
173
174    #region    <--- Дополнительные Методы MyTable --->
175
176    protected void Roots_Location()...
189    public string Table_of_Roots(string comment)...
203
204    public virtual void Roots_Correction(double eps)
205    { }
206
207    #endregion <--- Дополнительные Методы MyTable --->
208  }
```

Новый класс **MyExtendedTableOfFunction** является наследником класса **MyTableOfFunction** и размещается с ним в общем файле **MAC_Common**.

В новом классе переопределяем виртуальный метод **Roots_Correction()** уточнения нулей функции с использованием перегруженного метода Ньютона:

```
254  public class MyExtendedTableOfFunction : MyTableOfFunction
255  {
256    protected Function_of_x d1Fx, d2Fx;
257
258    public MyExtendedTableOfFunction
259      (double xo, double xn, int n,
260       Function_of_x f_x, Function_of_x d1_x, Function_of_x d2_x, string title)
261    : base(xo, xn, n, f_x, title)
262    {
263      d1Fx = d1_x; d2Fx = d2_x;
264    }
265    public override void Roots_Correction(double eps)
266    {
267      if (Roots != null)
268      {
269        foreach (Root root in Roots)
270          MAC_Equations.Tangent(Fx, d1Fx, d2Fx, root, eps);
271      }
272    }
273  }
```

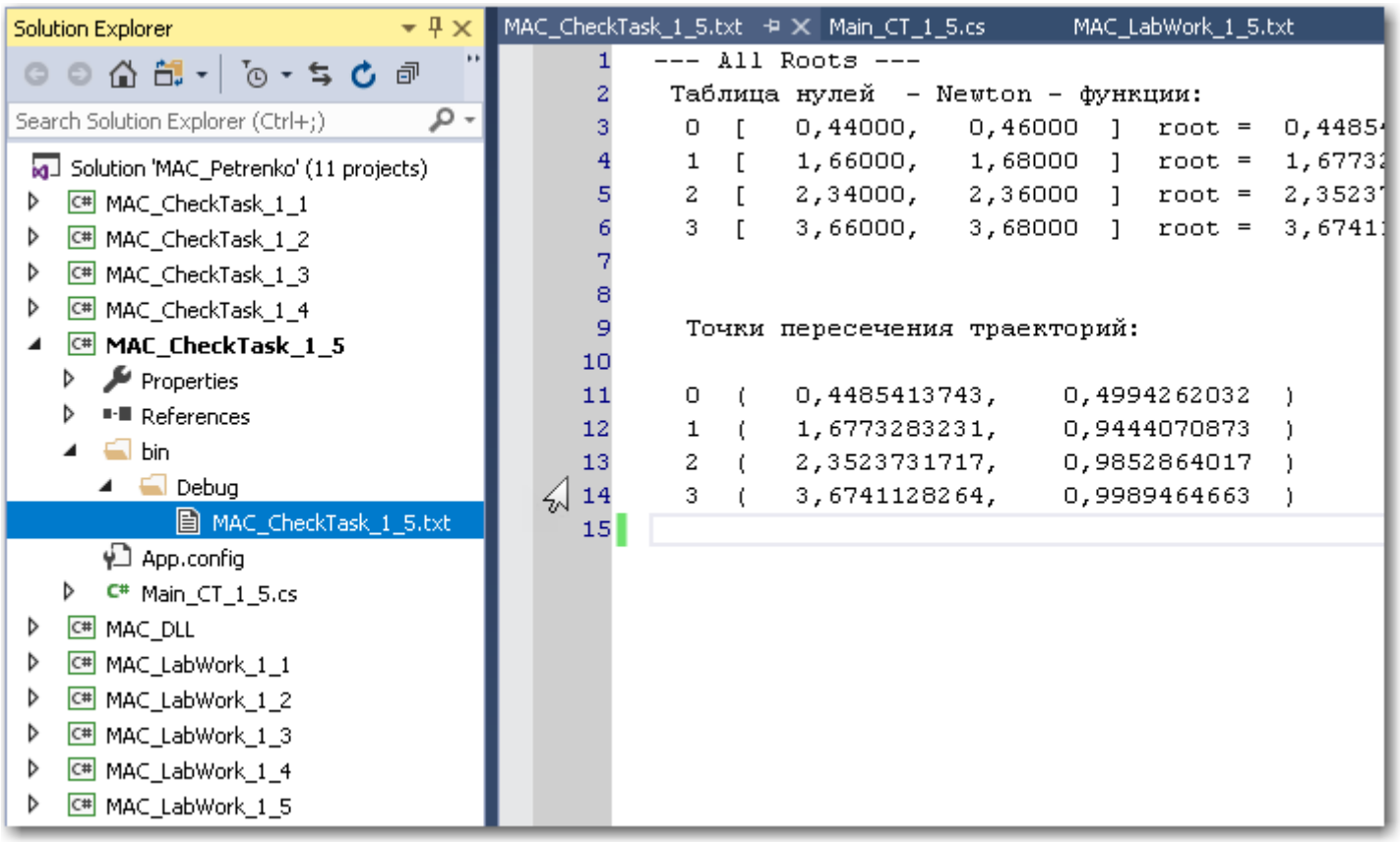
Перегруженный метод (на рисунке ниже на его сигнатуру указывает стрелка мыши) Вам следует запрограммировать самостоятельно, взяв за основу уже отлаженный аналогичный метод на основе алгоритма (1.5.1):

```
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)...
40
41    public static double
42      Tangent(Fx Fx, Fx D1Fx, Fx D2Fx,
43              double xL, double xR, double eps, out int K)...
54
55    public static void
56      Tangent(Fx Fx, Fx D1Fx, Fx D2Fx, Root root, double eps)...
68
69    public static double
70      Tangent(Fx Fx, double xL, double xR, double eps, out int K)...
80  }
```

После того, как эта часть работы будет Вами выполнена, можно воспользоваться разработанными компонентами и с их помощью выполнить поставленную задачу:

```
15  static void Main(string[] args)
16  {
17    a = 0.1; b = -0.1; s = Math.Sqrt(10.0);
18    StreamWriter SW = new StreamWriter("MAC_CheckTask_1_5.txt");
19
20    #region    <-- Решение поставленной задачи -->
21
22    MyETF ET_Fx = new MyETF(0.0, 4.0, 200, Fx, d1Fx, d2Fx, " - Newton -");
23    ET_Fx.Roots_Correction(1.0E-12);
24
25    SW.WriteLine(ET_Fx.Table_of_Roots("--- All Roots ---"));
26
27    SW.WriteLine("\r\n  Точки пересечения траекторий: \r\n");
28    MAC_DLL.Point_xf xy;
29    for (int i = 0; i < ET_Fx.Roots.Count; i++)
30    {
31      xy = new MAC_DLL.Point_xf(ET_Fx.Roots[i].x, f1(ET_Fx.Roots[i].x));
32      SW.WriteLine(string.Format("{0,3}", i) + xy.ToPrint());
33    }
34
35    #endregion <-- Решение поставленной задачи -->
36
37    SW.Close();
38  }
```

Результаты вычислений сохраняются в текстовом файле **`MAC_CheckTask_1_5.txt`**:




```
--- All Roots ---
 Таблица нулей  - Newton - функции:
  0  [   0,44000,   0,46000  ]  root =  0,448541374334  err =  5,6E-017  iters = 3
  1  [   1,66000,   1,68000  ]  root =  1,677328323143  err =  9,7E-015  iters = 3
  2  [   2,34000,   2,36000  ]  root =  2,352373171678  err =  1,6E-015  iters = 3
  3  [   3,66000,   3,68000  ]  root =  3,674112826428  err =  2,6E-015  iters = 3
```

Если Вы успешно выполнили данное программное проектирование, и получили представленные выше числовые результаты, то Вы можете приступать к непосредственному выполнению своего индивидуального варианта данного Контрольного задания.

Для этого используйте уже имеющийся проект **`MAC_CheckTask_1_5`**.