

Методика выполнения контрольного задания № 1.3

«Построение таблицы функции с заданной точностью»

Постановка задачи. Построить алгоритм и отладить программу вычисления таблицы функции, заданной в виде аналитического представления

$$f_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \cdot \left[\cos\left(\frac{b}{k+1}x\right) + \sin\left(\frac{b}{k+a}x\right) \right]}{4 \cdot (2k+a) \cdot (k+1)!} \left(\frac{11x}{7}\right)^{k+1}, \quad (1)$$

на интервале $x \in [x_0, x_n]$ с заданным количеством строк n и абсолютной погрешностью ε , где a и b – некоторые заданные значения вещественных параметров данной функции.

На основе данных, имеющихся в таблице функции, установить наибольшее и наименьшее значения функции на всем заданном интервале $[x_0, x_n]$ изменения аргумента x .

Собственно создание таблицы функции $f = f(x)$, должно быть реализовано в теле конструктора нового класса – **MyTableOfFunction**, наследника от класса **MyTable**.

Указанный конструктор должен принимать конкретную функцию $f(x)$ в виде параметра.

Метод, реализующий вычисление значений функции $f(x)$ в соответствии с представлением (1), должен быть размещен в отдельном классе **MyFunctions** динамической библиотеки **MAC_DLL**:

```

7  namespace MAC_DLL
8  {
9  public class MyFunctions
10 {
11     public static double f0(double x, double a, double b, double eps)
12     {
13         if (x == 0) return 0.0;
14         double xz = 11.0 * x / 7.0, bx = b * x;
15         double A0 = (Math.Cos(bx) + Math.Sin(bx / a)) / a;
16         double pk = 1.0, Ak = 1.0, summa = 0.0;
17         for (int k = 1; Math.Abs(Ak) > eps; k++)
18         {
19             pk = -pk * xz / (k + 1.0);
20             Ak = (Math.Cos(bx / (k + 1.0)) + Math.Sin(bx / (k + a)))
21                 * pk / (2.0 * k + a);
22             summa += Ak;
23         }
24         return 0.25 * xz * (A0 + summa);
25     }
26 }
27 
```

В имеющемся рабочем пространстве **MAC_Petrenko** сгенерируем (добавим) новый проект **MAC_CheckTask_1_3** консольного приложения, который сделаем стартовым проектом.

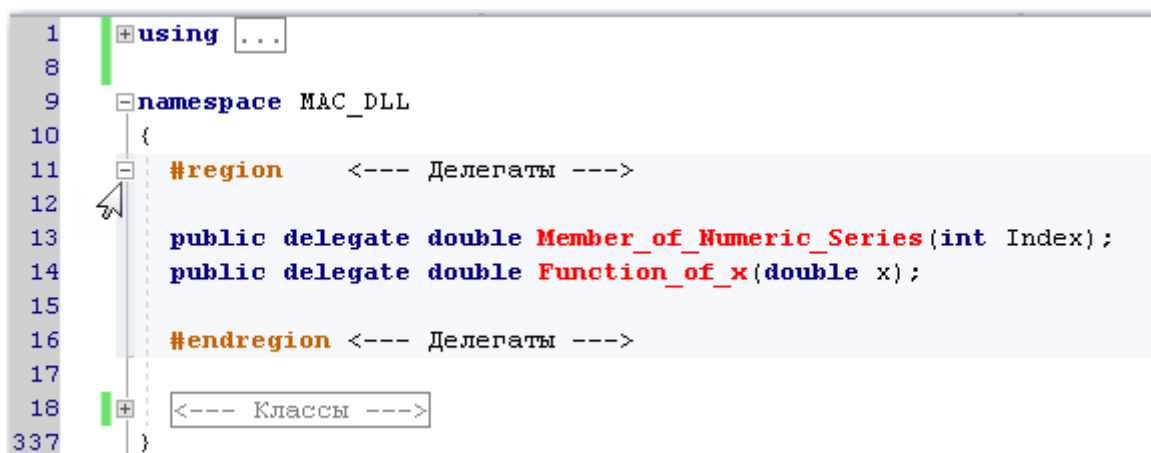
Добавим в раздел **References** проекта **MAC_CheckTask_1_3** ссылку на проект **MAC_DLL**.

Можно приступить к разработке программной реализации класса **MyTableOfFunction**.

По условию задания конструктор **MyTableOfFunction()** должен принимать в качестве входного параметра метод с сигнатурой, соответствующей функции от одной действительной переменной (1), т.е. $f = f(x)$.

Это обязывает нас добавить в раздел делегатов библиотеки **MAC_DLL** определение нового делегата – **Function_of_x(double x)**.

Выполним это в уже имеющемся файле – **MAC_Common.cs** – в разделе «Делегаты»:



```

1  + using ...
8
9  namespace MAC_DLL
10 {
11     #region <--- Делегаты --->
12
13     public delegate double Member_of_Numeric_Series(int Index);
14     public delegate double Function_of_x(double x);
15
16     #endregion <--- Делегаты --->
17
18     <--- Классы --->
337 }
  
```

Алгоритм расчета таблицы функции, когда она задана аналитически, достаточно прост.

Непосредственно в теле конструктора класса **MyTableOfFunction** выполняется цикл по значениям аргумента от левой границы x_0 до правой границы x_n области определения $x \in [x_0, x_n]$ с постоянным шагом $h = (x_n - x_0) / n$.

В этом же цикле одновременно выполняется отбор узлов таблицы с наименьшим и наибольшим значениями функции.

Также следует переопределить виртуальный метод **ToPrint()**.

Класс MyTableOfFunction. Предлагается следующая реализация класса:

```

1  using System;
2  using System.Collections.Generic;
3  using CI = System.Globalization.CultureInfo;
4  using System.IO;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Linq;
8
9  namespace MAC_DLL
10 {
11     <--- Делегаты --->
12
13     #region <--- Классы --->
14
15     public class Point_xf...
16
17     public abstract class MyTable...
18
19     public class MyTableOfFunction : MyTable
20     {
21         readonly Function_of_x Fx; // Указатель на функцию f(x)
22
23         public MyTableOfFunction(double xo, double xn, int n,
24             Function_of_x f_x, string title)
25         {
26             Title = title; Fx = f_x;
27             Points = new Point_xf[n + 1];
28
29             Minimum = new Point_xf(double.NaN, double.MaxValue);
30             Maximum = new Point_xf(double.NaN, double.MinValue);
31
32             double hx = (xn - xo) / n, xi;
33             for (int i = 0; i <= n; i++)
34             {
35                 xi = xo + i * hx; Points[i] = new Point_xf(xi, Fx(xi));
36                 if (Minimum.f > Points[i].f) { Minimum = Points[i]; }
37                 if (Maximum.f < Points[i].f) { Maximum = Points[i]; }
38             }
39         }
40
41         #region <--- Переопределение методов класса MyTable --->
42         public override string ToPrint(string Comment)
43         {
44             return Comment + "\r\n" + Table_of_Function();
45         }
46         #endregion <--- Переопределение методов класса MyTable --->
47     }
48
49     public class MyTableOfData ...
50
51     #endregion <--- Классы --->
52 }

```

По условиям задания, параметры конструктора **MyTableOfFunction()** имеют следующий смысл:

- **x0** – левая граница интервала, на котором рассчитывается таблица функции;
- **xn** – правая граница этого интервала;
- **n** – количество равных интервалов, на которое требуется разбить отрезок $[x_0, x_n]$;
- **f_x** – делегат, определяющий сигнатуру метода, непосредственно вычисляющего значение заданной функции;
- **title** – строковая переменная, содержащая «идентификатор» таблицы функции.

Теперь можно выполнить первую проверку и вычислить таблицу функции (1) в основной программе **MAC_CheckTask_1_3**:

```
1  using System;
2  using MyF = MAC_DLL.MyFunctions;
3  using MyTF = MAC_DLL.MyTableOfFunction;
4  // using System.Collections.Generic; ...
8
9  namespace MAC_Check_Task_1_3
10 {
11     class Main_CT_1_3
12     {
13         public static double par_a, par_b, par_e;
14
15         static void Main(string[] args)
16         {
17             par_a = 3.0; par_b = 1.8; par_e = 1.0E-9;
18
19             MyTF TF = new MyTF(1.0, 7.0, 50, dummy_fx, "dummy f(x)");
20             Console.Write(TF.ToPrint(" Тестовая Таблица Функции:"));
21         }
22         public static double dummy_fx(double x)
23         {
24             return MyF.f0(x, par_a, par_b, par_e);
25         }
26     }
27 }
```

Имеем следующий результат:

```

C:\Windows\system32\cmd.exe
Тестовая Таблица Функции:

Таблица функции dummy f(x) :
0 ( 1,0000000000, -0,0005153151 )
1 ( 1,1200000000, -0,0227539923 )
2 ( 1,2400000000, -0,0454573978 )
3 ( 1,3600000000, -0,0659410809 )
4 ( 1,4800000000, -0,0813182888 )
5 ( 1,6000000000, -0,0886920525 )
6 ( 1,7200000000, -0,0853595300 )
7 ( 1,8400000000, -0,0690167634 )
8 ( 1,9600000000, -0,0379515945 )
9 ( 2,0800000000, 0,0087872304 )
10 ( 2,2000000000, 0,0712557579 )
11 ( 2,3200000000, 0,1485246995 )
12 ( 2,4400000000, 0,2386406411 )
13 ( 2,5600000000, 0,3386431091 )
14 ( 2,6800000000, 0,4446390331 )
15 ( 2,8000000000, 0,5519330086 )
16 ( 2,9200000000, 0,6552086377 )
17 ( 3,0400000000, 0,7487532819 )
18 ( 3,1600000000, 0,8267159654 )
19 ( 3,2800000000, 0,8833860648 )
20 ( 3,4000000000, 0,9134789409 )
21 ( 3,5200000000, 0,9124139102 )
22 ( 3,6400000000, 0,8765699601 )
23 ( 3,7600000000, 0,8035053962 )
24 ( 3,8800000000, 0,6921291774 )
25 ( 4,0000000000, 0,5428139298 )
26 ( 4,1200000000, 0,3574434628 )
27 ( 4,2400000000, 0,1393908791 )
28 ( 4,3600000000, -0,1065730736 )
29 ( 4,4800000000, -0,3744381223 )
30 ( 4,6000000000, -0,6571722337 )
31 ( 4,7200000000, -0,9469889508 )
32 ( 4,8400000000, -1,2356514479 )
33 ( 4,9600000000, -1,5147983084 )
34 ( 5,0800000000, -1,7762746470 )
35 ( 5,2000000000, -2,0124516605 )
36 ( 5,3200000000, -2,2165180448 )
37 ( 5,4400000000, -2,3827279704 )
38 ( 5,5600000000, -2,5065924158 )
39 ( 5,6800000000, -2,5850035091 )
40 ( 5,8000000000, -2,6162850033 )
41 ( 5,9200000000, -2,6001659301 )
42 ( 6,0400000000, -2,5376786602 )
43 ( 6,1600000000, -2,4309867997 )
44 ( 6,2800000000, -2,2831523666 )
45 ( 6,4000000000, -2,0978553044 )
46 ( 6,5200000000, -1,8790813963 )
47 ( 6,6400000000, -1,6307968639 )
48 ( 6,7600000000, -1,3566292221 )
49 ( 6,8800000000, -1,0595742484 )
50 ( 7,0000000000, -0,7417481572 )

x = [ 1,0000000000 : 7,0000000000 ]
x_Reg = 6,0000000000

Min ( 5,8000000000, -2,616285003267 )
Max ( 3,4000000000, 0,913478940863 )
f_Reg = 3,529763944130
Для продолжения нажмите любую клавишу . . .

```

Визуальный анализ таблицы подтверждает правильность программного определения ее узлов с наибольшим и наименьшим значениями функции, а также длин области определения и области значений для данной функции.

Подведем предварительные итоги проделанной работы.

В нашем распоряжении имеется абстрактный класс **MyTable**, описывающий обобщенную совокупность числовых данных типа «Таблица» и алгоритмы (методы и свойства) их обработки вне зависимости от способа формирования самой таблицы.

На основе этого абстрактного класса мы создали два класса-наследника **MyTableOfData** и **MyTableOfFunction**, которые формируют таблицу двумя принципиально различными способами. При этом классы-наследники пользуются общими методами и свойствами класса **MyTable**, и при необходимости переопределяют их.

Мы и в дальнейшем будем придерживаться такой стратегии в формировании новых свойств и методов для указанных классов.

Общие для двух типов таблиц алгоритмы мы будем размещать в родительском классе **MyTable**. А все специфические алгоритмы – как собственные члены в классах-наследниках.

Продемонстрируем сказанное следующим образом.

Результаты выполнения двух последних заданий **MAC_LabWork_1_3** и **MAC_CheckTask_1_3** содержали достаточно большие объемы числовой информации, которая выводилась непосредственно в окно консольного приложения.

Просмотр и анализ результатов в такой форме нельзя признать удобным, в особенности, когда ее надо сохранить с целью сопоставления с подобными результатами.

Очевидным является способ сохранения результатов в виде текстового именованного файла, который, кроме всего прочего, можно включить в соответствующий проект рабочего пространства в качестве его обновляемого компонента.

Поскольку сохранение результатов обработки таблицы не зависит от способа генерации ее числовых данных, метод, который будет нам обеспечивать вывод результата в файл, мы разместим в абстрактном классе и сделаем виртуальным, чтобы при необходимости его можно было переопределить.

Основным аргументом этого метода является полный путь к файлу – параметр **path**, в который планируется сохранять результаты вычислений.

Если путь к файлу содержит только его имя, то файл размещается в рабочей директории исполняемого файла данного проекта.

Если имя не указано – выбирается имя «по умолчанию» – **My_Table.txt**:

```

40 public abstract class MyTable
41 {
42     <--- Основные Свойства MyTable --->
72
73     #region <--- Основные Методы MyTable --->
74
75     // Метод - возвращает значение x для строки таблицы с номером index
76     public double X(int index) {...}
81
82     // Метод - возвращает значение f для строки таблицы с номером index
83     public double F(int index) {...}
88
89     // Метод, формирующий таблицу функции в текстовой форме
90     public virtual string Table_of_Function() {...}
121
122     // Метод - возвращает два массива со значениями аргумента x и функции f
123     public void ToArrays(out double[] x, out double[] f) {...}
129
130     // Дополнительный переопределяемый метод, предназначенный для операций
131     // вывода в текстовой форме различной информации по данной таблице
132
133     public virtual string ToPrint(string comment) {...}
137
138     public virtual void To_txt_File(string path, string comment)
139     {
140         if (path == "") path = "My_Table.txt";
141         FileInfo file = new FileInfo(path);
142         if (file.Exists) file.Delete();
143         StreamWriter SW = new StreamWriter(file.OpenWrite());
144         SW.Write(comment + "\r\n" + Table_of_Function());
145         SW.Close();
146     }
147
148     #endregion <--- Основные Методы MyTable --->
149 }

```

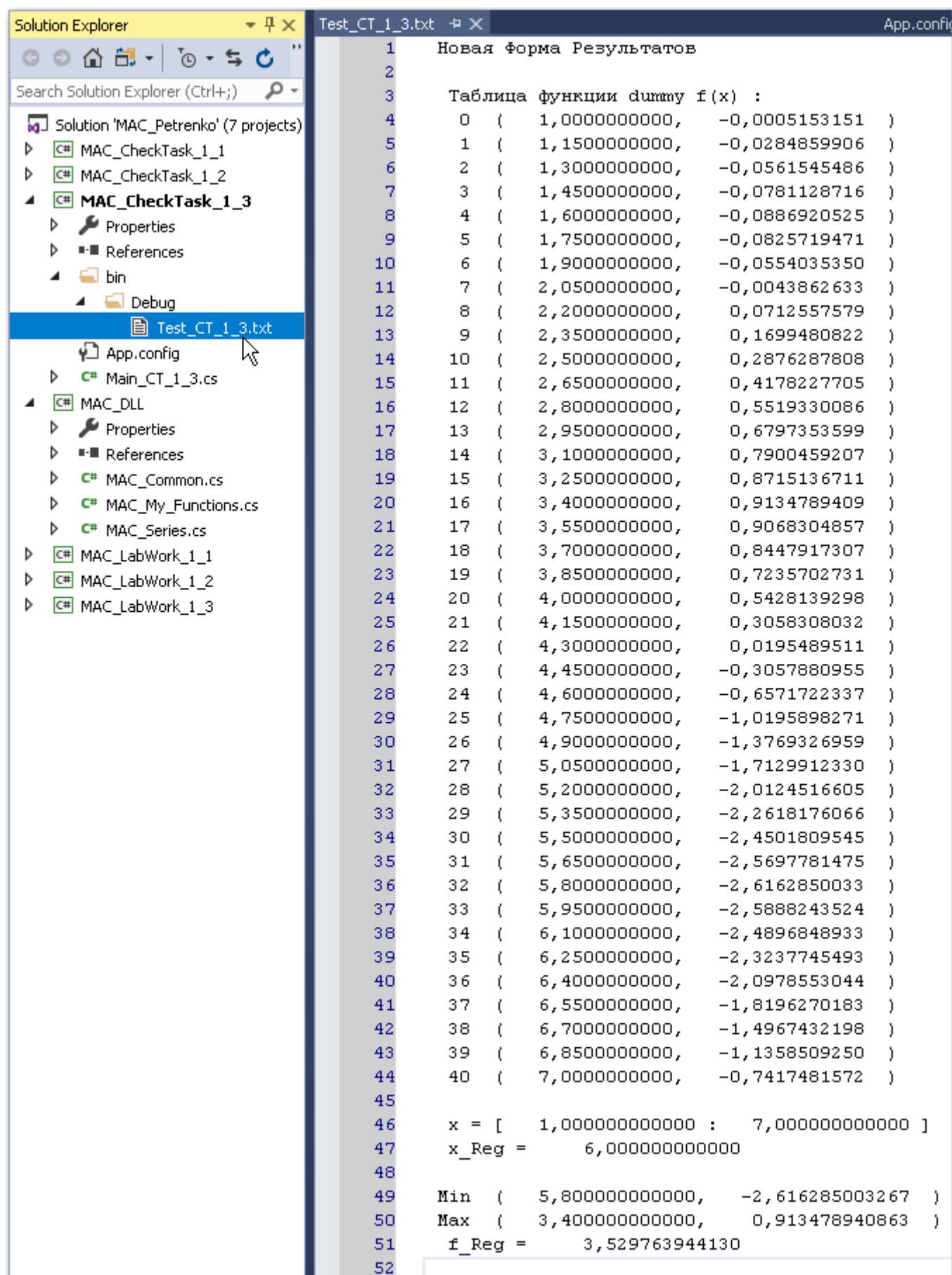
Добавим вызов нового метода в приложение **MAC_CheckTask_1_3**:

```

11 class Main_CT_1_3
12 {
13     public static double par_a, par_b, par_e;
14
15     static void Main(string[] args)
16     {
17         par_a = 3.0; par_b = 1.8; par_e = 1.0E-9;
18
19         MyTF TF = new MyTF(1.0, 7.0, 40, dummy_fx, "dummy f(x)");
20         //Console.Write(TF.ToPrint(" Тестовая Таблица функции:"));
21         TF.To_txt_File("Test_CT_1_3.txt", " Новая Форма Результатов");
22     }
23     public static double dummy_fx(double x)
24     {
25         return MyF.f0(x, par_a, par_b, par_e);
26     }
27 }

```

Файл **Test_CT_1_3.txt** присоединим к проекту и откроем для просмотра:



Переходим к первой задаче математической обработки данных в таблице функции.

Из курса математического анализа известно, что если функция $f(x)$ строго монотонна на интервале $[x_{i-1}, x_i]$ и выполняется неравенство $f(x_{i-1}) \cdot f(x_i) < 0$, то на этом интервале имеется точка с координатой x_* , в которой функция обращается в нуль, т.е. $f(x_*) = 0$.

Поставим перед собой задачу программно установить для уже созданной таблицы все интервалы $[x_{i-1}, x_i]$ изменения аргумента, на концах которых функция $f(x)$ имеет разные знаки, т.е. выполняется неравенство $f(x_{i-1}) \cdot f(x_i) < 0$.

Очевидно, что мы не можем наперед угадать количество таких интервалов.

Нам придется последовательно «обрабатывать» узлы таблицы и определять те из них, для которых выполняются указанные выше условия.

Как только такой интервал $[x_{i-1}, x_i]$ будет обнаружен – мы должны сохранить информацию о нем в какой-то переменной и перейти к следующим узлам таблицы.

Для подобного «сбора» информации подходит тип данных – список.

Но нам еще предстоит обобщить понятие «интервал, содержащий нуль функции» в виде некоторого пользовательского типа данных (класса) и определить количество и качество составляющих его членов.

Будем исходить из того, что интервал, содержащий нуль функции, описывается тремя действительными числами: это координаты концов интервала $[x_{i-1}, x_i]$ и собственно координата x_* , в которой функция обращается в нуль, т.е. $f(x_*) = 0$.

В численных методах координата x_* вычисляется приближенно (с использованием определенного алгоритма) с некоторой погрешностью, которая характеризует «качество» найденного значения x_* . Эту погрешность можно отождествить со значением $\varepsilon = |f(x_*)|$.

Чем меньше эта величина, тем «качественнее» нами вычислено значение x_* .

Мы включим эту величину в качестве члена в создаваемый класс.

Кроме того, алгоритмы уточнения нулей функции, как правило, выполняются итерационным путем. Нас будет интересовать количество итераций, затраченных тем или иным численным методом на вычисление нуля функции с заданной погрешностью.

Чем меньше таких итераций – тем лучше численный метод.

Количество итераций – целое число – также будет членом нового класса.

Класс Root.

Ниже предлагается код нового класса **Root**, который разместим в разделе общих классов файла **MAC_Common.cs**:

```

9      namespace MAC_DLL
10     {
11         <--- Делегаты --->
12
13
14
15
16
17
18         #region <--- Классы --->
19
20         public class Point_xf...
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
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()
49             {
50                 return
51                     string.Format(" [{0,10:F5},{1,10:F5} ]", xL, xR) +
52                     string.Format(" root={0,16:F12}", x) +
53                     string.Format(" err={0,10:E1}", err) +
54                     string.Format(" iters = {0}", iters);
55             }
56         }
57
58         public abstract class MyTable...
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Добавим в совокупность членов абстрактного класса **MyTable** новый элемент (свойство) – **List<Root> Roots** – список объектов типа **Root**.

Алгоритм инициализации этого списка и заполнения его соответствующими экземплярами класса **Root** реализуем в виде закрытого метода **Roots_Location()** абстрактного класса **MyTable**.

Одновременно создадим вспомогательный метод **Table_of_Roots()**, формирующий таблицу искомых интервалов.

Оба этих метода разместим в отдельном регионе «Дополнительные Методы **MyTable**»:

```

58 public abstract class MyTable
59 {
60     <--- Основные Свойства MyTable --->
90
91     #region <--- Дополнительные Свойства MyTable --->
92
93     protected List<Root> Roots { get; set; }
94
95     #endregion <--- Дополнительные Свойства MyTable --->
96
97     <--- Основные Методы MyTable --->
173
174     #region <--- Дополнительные Методы MyTable --->
175
176     protected void Roots_Location()
177     {
178         int counter = 0;
179         for (int i = 1; i < Length; i++)
180         {
181             if (Points[i - 1].f * Points[i].f < 0)
182             {
183                 counter++;
184                 if (counter == 1) { Roots = new List<Root>(); }
185                 Roots.Add(new Root(Points[i - 1].x, Points[i].x));
186             }
187         }
188     }
189
190     public string Table_of_Roots(string comment)
191     {
192         string table = comment + "\r\n";
193         if (Roots != null)
194         {
195             table += " Таблица нулей " + Title + " функций: \r\n";
196             for (int j = 0; j < Roots.Count; j++)
197             {
198                 table += string.Format("{0,3}", j) + Roots[j].ToPrint() + "\r\n";
199             }
200         }
201         else { table += " Таблица нулей " + Title + " пустая!\r\n"; }
202         return table;
203     }
204
205     #endregion <--- Дополнительные Методы MyTable --->
206
207 }

```

Расширим функционал метода **Table_of_Function()**:

```

97  #region <--- Основные Методы MyTable --->
98
99  // Метод - возвращает значение x для строки таблицы с номером index
100 public double X(int index) ...
105
106 // Метод - возвращает значение f для строки таблицы с номером index
107 public double F(int index) ...
112
113 // Метод, формирующий таблицу функции в текстовой форме
114 public virtual string Table_of_Function()
115 {
116     string txt = "\r\n Таблица функции " + Title + " : \r\n";
117
118     for (int i = 0; i < Length; i++)
119     {
120         txt += string.Format("{0}", i).PadLeft(4)
121             + Points[i].ToPrint() + "\r\n";
122     }
123
124     txt += "\r\n x = ["
125         + string.Format("{0:F12}", Points[0].x).PadLeft(17) + " : "
126         + string.Format("{0:F12}", Points[Length - 1].x).PadLeft(17)
127         + " ] \r\n";
128
129     txt += " x_Reg = "
130         + string.Format("{0:F12}", Region_x).PadLeft(18) + "\r\n";
131
132     txt += "\r\n Min  ("
133         + string.Format("{0:F12}", Minimum.x).PadLeft(18)
134         + string.Format("{0:F12}", Minimum.f).PadLeft(18) + " )";
135
136     txt += "\r\n Max  ("
137         + string.Format("{0:F12}", Maximum.x).PadLeft(18)
138         + string.Format("{0:F12}", Maximum.f).PadLeft(18) + " )";
139
140     txt += "\r\n f_Reg = "
141         + string.Format("{0:F12}", Region_f).PadLeft(18) + "\r\n";
142
143     return txt + Table_of_Roots("");
144 }
145
146 // Метод - возвращает два массива со значениями аргумента x и функции f
147 public void ToArrays(out double[] x, out double[] f) ...
153
154 // Дополнительный переопределяемый метод, предназначенный для операций ...
155
156
157 public virtual string ToPrint(string comment) ...
161
162 public virtual void To_txt_File(string path, string comment) ...
171
172 #endregion <--- Основные Методы MyTable --->

```

Добавим теперь вызов метода **Roots_Location()** в код конструкторов классов-наследников **MyTableOfData** и **MyTableOfFunction**:

```

240 public class MyTableOfData : MyTable
241 {
242     // Свойство - Таблица данных непосредственно в файле
243     public string Table_in_File { get; }
244
245     // <--- Конструктор экземпляра --->
246     public MyTableOfData(string path, string title)
247     {
248         List<Point_xf> Temp = new List<Point_xf>();
249         FileInfo file = new FileInfo(path);
250         Table_in_File = "\r\n Таблица " + title +
251             " в файле " + file.Name + " :\r\n";
252
253         <--- Чтение бинарного файла в список данных --->
274
275         <--- Чтение форматного файла в список данных --->
304
305         <--- Формирование Таблицы Данных --->
324
325         Roots_Location();
326     }
327
328     <--- Переопределение методов класса MyTable --->
334 }
    
```

```

209 public class MyTableOfFunction : MyTable
210 {
211     readonly Function_of_x Fx; // Указатель на функцию f(x)
212
213     public MyTableOfFunction(double xo, double xn, int n,
214         Function_of_x f_x, string title)
215     {
216         Title = title; Fx = f_x;
217         Points = new Point_xf[n + 1];
218
219         Minimum = new Point_xf(double.NaN, double.MaxValue);
220         Maximum = new Point_xf(double.NaN, double.MinValue);
221
222         double hx = (xn - xo) / n, xi;
223         for (int i = 0; i <= n; i++)
224         {
225             xi = xo + i * hx; Points[i] = new Point_xf(xi, Fx(xi));
226             if (Minimum.f > Points[i].f) { Minimum = Points[i]; }
227             if (Maximum.f < Points[i].f) { Maximum = Points[i]; }
228         }
229         Roots_Location();
230     }
231
232     <--- Переопределение методов класса MyTable --->
238 }
    
```

Перекомпилируем проект **MAC_DLL** и убедимся в отсутствии ошибок.

Выполним заново приложение **MAC_CheckTask_1_3** (не внося в него никаких изменений) и получим новый результат – таблицу функции с выделенными интервалами, на концах которых функция меняет знак (нижняя часть таблицы):

```

9      5 ( 1,7500000000, -0,0825719471 )
10     6 ( 1,9000000000, -0,0554035350 )
11     7 ( 2,0500000000, -0,0043862633 )
12     8 ( 2,2000000000, 0,0712557579 )
13     9 ( 2,3500000000, 0,1699480822 )
14    10 ( 2,5000000000, 0,2876287808 )
15    11 ( 2,6500000000, 0,4178227705 )
16    12 ( 2,8000000000, 0,5519330086 )
17    13 ( 2,9500000000, 0,6797353599 )
18    14 ( 3,1000000000, 0,7900459207 )
19    15 ( 3,2500000000, 0,8715136711 )
20    16 ( 3,4000000000, 0,9134789409 )
21    17 ( 3,5500000000, 0,9068304857 )
22    18 ( 3,7000000000, 0,8447917307 )
23    19 ( 3,8500000000, 0,7235702731 )
24    20 ( 4,0000000000, 0,5428139298 )
25    21 ( 4,1500000000, 0,3058308032 )
26    22 ( 4,3000000000, 0,0195489511 )
27    23 ( 4,4500000000, -0,3057880955 )
28    24 ( 4,6000000000, -0,6571722337 )
29    25 ( 4,7500000000, -1,0195898271 )
30    26 ( 4,9000000000, -1,3769326959 )
31    27 ( 5,0500000000, -1,7129912330 )
32    28 ( 5,2000000000, -2,0124516605 )
33    29 ( 5,3500000000, -2,2618176066 )
34    30 ( 5,5000000000, -2,4501809545 )
35    31 ( 5,6500000000, -2,5697781475 )
36    32 ( 5,8000000000, -2,6162850033 )
37    33 ( 5,9500000000, -2,5888243524 )
38    34 ( 6,1000000000, -2,4896848933 )
39    35 ( 6,2500000000, -2,3237745493 )
40    36 ( 6,4000000000, -2,0978553044 )
41    37 ( 6,5500000000, -1,8196270183 )
42    38 ( 6,7000000000, -1,4967432198 )
43    39 ( 6,8500000000, -1,1358509250 )
44    40 ( 7,0000000000, -0,7417481572 )
45
46    x = [ 1,000000000000 : 7,000000000000 ]
47    x_Reg = 6,000000000000
48
49    Min ( 5,800000000000, -2,616285003267 )
50    Max ( 3,400000000000, 0,913478940863 )
51    f_Reg = 3,529763944130
52
53    Таблица нулей функции f(x) функции:
54    0 [ 2,05000, 2,20000 ] root = NaN err = NaN iters = 0
55    1 [ 4,30000, 4,45000 ] root = NaN err = NaN iters = 0

```

Теперь вернемся к проекту **MAC_LabWork_1_3** и внесем в код соответствующие изменения, после чего выполним проверку корректности работы новых методов:

```

1  using System;
2  using MyTD = MAC_DLL.MyTableOfData;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace MAC_LabWork_1_3
9  {
10     class Main_LW_1_3
11     {
12         static void Main(string[] args)
13         {
14             MyTD T1 = new MyTD("MAC_LW_1_3_v00.bin", "binary file");
15             string txt = "\r\n";
16             txt += string.Format(" Прочитано строк: {0}", T1.Length);
17             txt += T1.ToPrint(" Обработка файла *.bin");
18             Console.WriteLine(txt);
19
20             MyTD T2 = new MyTD("MAC_LW_1_3_v00.txt", "text file");
21             txt = "\r\n";
22             txt = string.Format(" Прочитано строк: {0}", T2.Length);
23             txt += T2.ToPrint(" Обработка файла *.txt");
24             Console.WriteLine(txt);
25             T2.To_txt_File("Test_LW_1_3.txt", " Новая Форма Результаты:");
26         }
27     }
28 }

```

Не забудьте включить файл результатов **Test_LW_1_3.txt** в состав проекта **MAC_LabWork_1_3** и открыть его для просмотра в **MS VS**:

```

1  Новая Форма Результатов:
2
3  Таблица функции text    file :
4  0  (  -3,100000000000,   -1,58100000000 )
5  1  (  -2,87916666667,    1,6974586950 )
6  2  (  -2,65833333333,    4,2309959491 )
7  3  (  -2,43750000000,    6,0842285156 )
8  4  (  -2,21666666667,    7,3217731481 )
9  5  (  -1,99583333333,    8,0082466001 )
10  6  (  -1,77500000000,    8,2082656250 )
11  7  (  -1,55416666667,    7,9864469763 )
12  8  (  -1,33333333333,    7,4074074074 )
13  9  (  -1,11250000000,    6,5357636719 )
14 10  (  -0,89166666667,    5,4361325231 )
15 11  (  -0,67083333333,    4,1731307147 )
16 12  (  -0,45000000000,    2,8113750000 )
17 13  (  -0,22916666667,    1,4154821325 )
18 14  (  -0,00833333333,    0,0500688657 )
19 15  (   0,21250000000,   -1,2202480469 )
20 16  (   0,43333333333,   -2,3308518519 )
21 17  (   0,65416666667,   -3,2171257957 )
22 18  (   0,87500000000,   -3,8144531250 )
23 19  (   1,09583333333,   -4,0582170862 )
24 20  (   1,31666666667,   -3,8838009259 )
25 21  (   1,53750000000,   -3,2265878906 )
26 22  (   1,75833333333,   -2,0219612269 )
27 23  (   1,97916666667,   -0,2053041811 )
28 24  (   2,20000000000,    2,2880000000 )
29
30  x = [  -3,100000000000 :   2,200000000000 ]
31  x_Reg =      5,300000000000
32
33  Min  (   1,09583333333,   -4,058217086227 )
34  Max  (  -1,775000000000,    8,208265625000 )
35  f_Reg =      12,266482711227
36
37  | Таблица нулей text    file функции:
38  0  [  -3,10000,   -2,87917 ] root =      NaN  err =      NaN  iters = 0
39  1  [  -0,00833,    0,21250 ] root =      NaN  err =      NaN  iters = 0
40  2  [   1,97917,    2,20000 ] root =      NaN  err =      NaN  iters = 0

```


Дополнительные замечания.

Основное приложение **MAC_CheckTask_1_3** должно нам продемонстрировать корректность работы разработанного конструктора **MyTableOfFunction()**.

Тестируемая функция (1) уже была нами разработана и проверена в ходе выполнения Контрольного задания 1.2, и находится в классе **MAC_My_Functions** библиотеки **MAC_DLL**.

Основной проблемой является несовместимость этой функции (с четырьмя действительными переменными) с определением делегата **Function_of_x** (для функции от одной переменной), использованного в конструкторе класса **MyTableOfFunction()**.

Решение этой проблемы может быть следующим.

В классе **Main_CT_1_3** основной программы мы объявляем статические члены, которые будут играть роль параметров ε , a и b для функции (1).

Добавляем в этот же класс **Main_CT_1_3** вспомогательный метод **dummy_fx(x)**, «превращающий» вызов функции четырех переменных в вызов функции от одной переменной.

Именно этот вспомогательный метод мы и будем передавать в конструктор **MyTableOfFunction()**, поскольку его сигнатура теперь уже будет соответствовать делегату **Function_of_x**.

Остальные параметры вычислений подберем таким образом, чтобы в рассчитываемой таблице оказались два контрольных узла для функции из Контрольного задания 1.2.

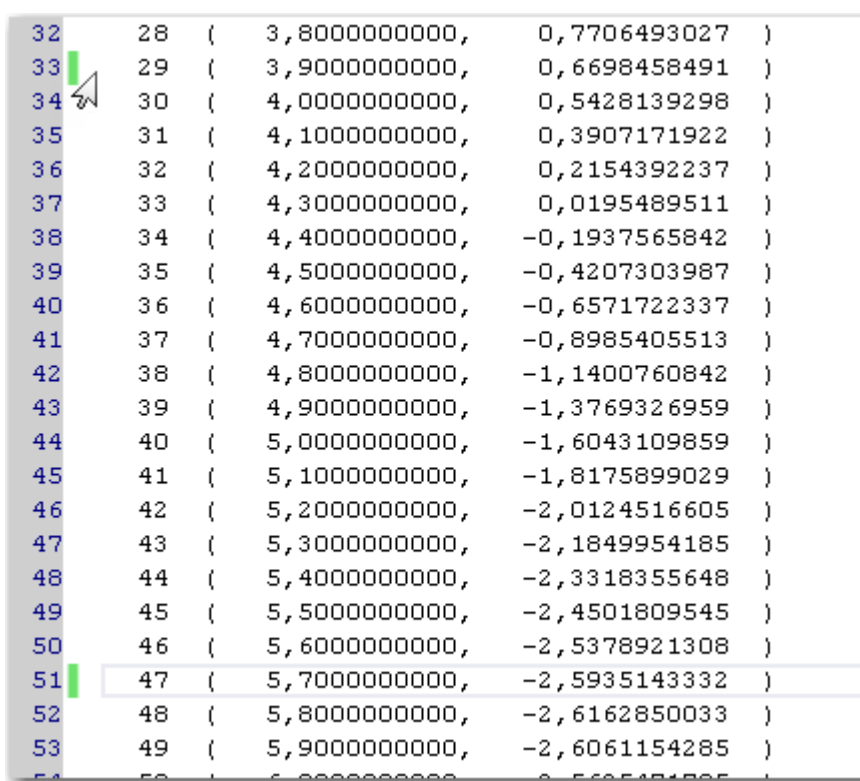
В Контрольном задании 1.2 вычисления функции выполнялись для значений $x_1 = 3.9$ и $x_2 = 5.7$ ее аргумента, при значениях $\varepsilon = 10^{-9}$, $a = 3.0$ и $b = 1.8$ ее параметров. Поэтому в настоящем расчете зададим $x_0 = 1.0$, $x_n = 7.0$ и $n = 60$, что обеспечит нам вычисление значений функции (1) в требуемых точках $x_1 = 3.9$ и $x_2 = 5.7$.

Итак, в функции **Main()** при инициализации экземпляра **MyTableOfFunction** в конструктор передаются соответствующие значения :

```
static void Main(string[] args)
{
    par_a = 3.0; par_b = 1.8; par_e = 1.0E-9;

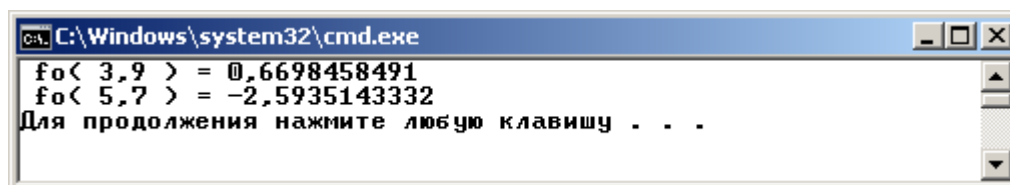
    MyTF TF = new MyTF(1.0, 7.0, 60, dummy_fx, "dummy f(x)");
    //Console.Write(TF.ToPrint(" Тестовая Таблица функции:"));
    TF.To_txt_File("Test_CT_1_3.txt", " Новая Форма Результатов");
}
```

Проверим теперь «содержимое» текстового файла с обработанной таблицей функции.



32	28	(3,8000000000,	0,7706493027)
33	29	(3,9000000000,	0,6698458491)
34	30	(4,0000000000,	0,5428139298)
35	31	(4,1000000000,	0,3907171922)
36	32	(4,2000000000,	0,2154392237)
37	33	(4,3000000000,	0,0195489511)
38	34	(4,4000000000,	-0,1937565842)
39	35	(4,5000000000,	-0,4207303987)
40	36	(4,6000000000,	-0,6571722337)
41	37	(4,7000000000,	-0,8985405513)
42	38	(4,8000000000,	-1,1400760842)
43	39	(4,9000000000,	-1,3769326959)
44	40	(5,0000000000,	-1,6043109859)
45	41	(5,1000000000,	-1,8175899029)
46	42	(5,2000000000,	-2,0124516605)
47	43	(5,3000000000,	-2,1849954185)
48	44	(5,4000000000,	-2,3318355648)
49	45	(5,5000000000,	-2,4501809545)
50	46	(5,6000000000,	-2,5378921308)
51	47	(5,7000000000,	-2,5935143332)
52	48	(5,8000000000,	-2,6162850033)
53	49	(5,9000000000,	-2,6061154285)
54	50	(6,0000000000,	-2,5695451505)

Мы можем сопоставить эти результаты с результатами Контрольного задания 1.2:



```
C:\Windows\system32\cmd.exe
fo< 3,9 > = 0,6698458491
fo< 5,7 > = -2,5935143332
Для продолжения нажмите любую клавишу . . .
```

Очевидно, что они совпадают.

Кроме того, нами выявлены те интервалы таблицы функции, на которых она меняет знак, а, следовательно, ее график пересекает ось абсцисс (нуль функции).