**Отчет**

**по Методам численного анализа**

Исполнитель

студент 161 группы

специальности

Прикладная математика

Борис Д. Ю.

«30» Марта 2018 г.

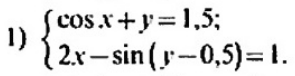
Гродно, 2018

**Лабораторная работа №4-5**

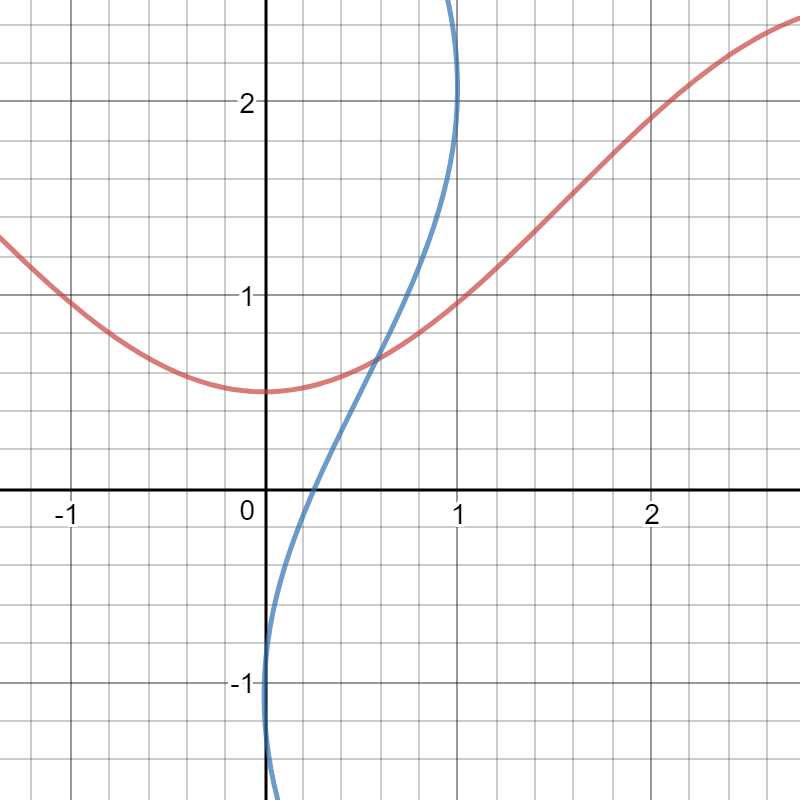
**Тема. Численное решение систем нелинейных уравнений**

Решить нелинейную систему методом итерации и методом Ньютона

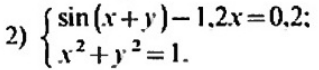
Первая нелинейная система:



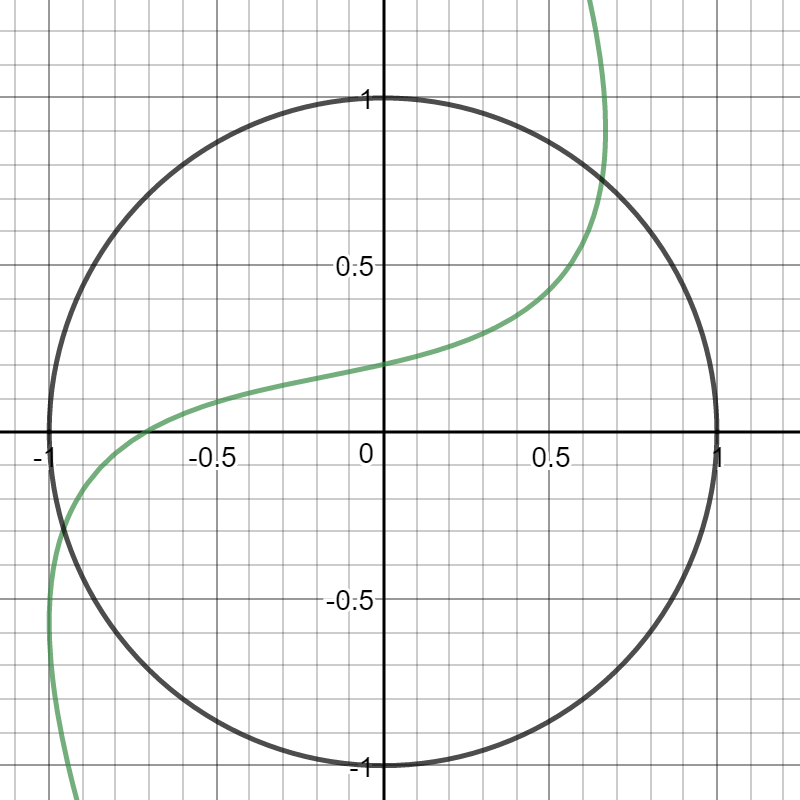
И её график, где красная – первая функция, синяя – вторая:



Вторая система:

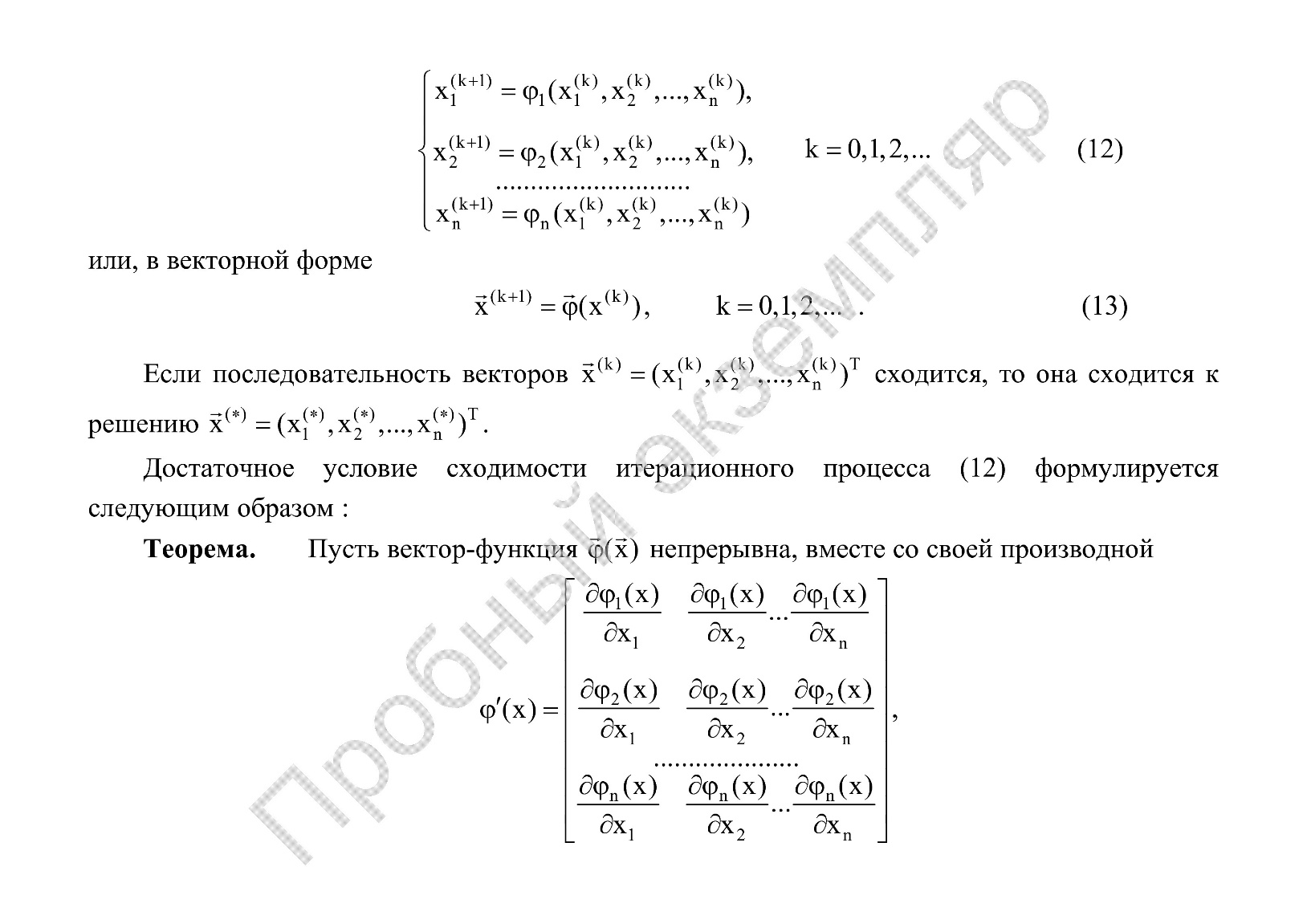
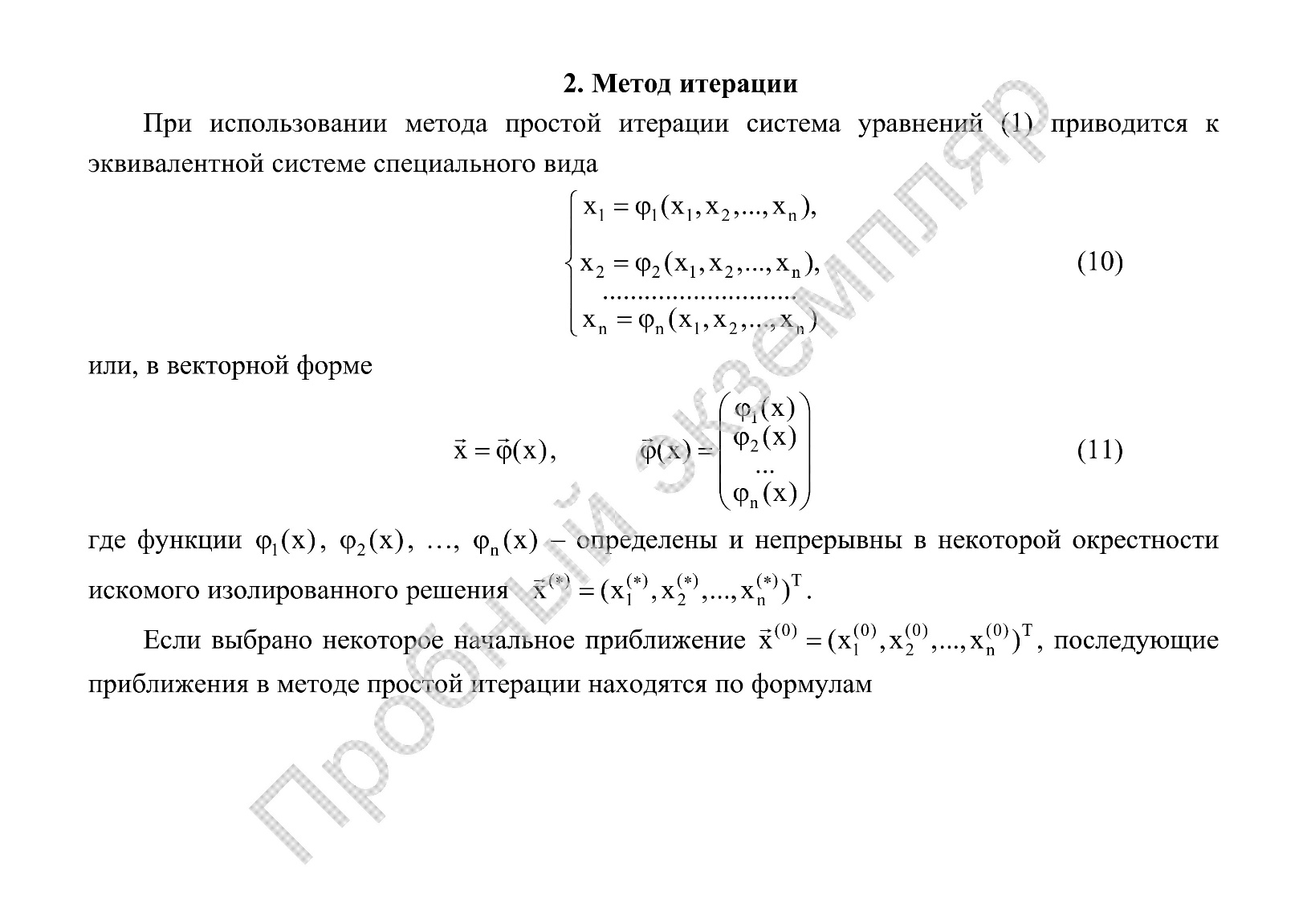


Её график, зелёная – первая, чёрная – вторая:

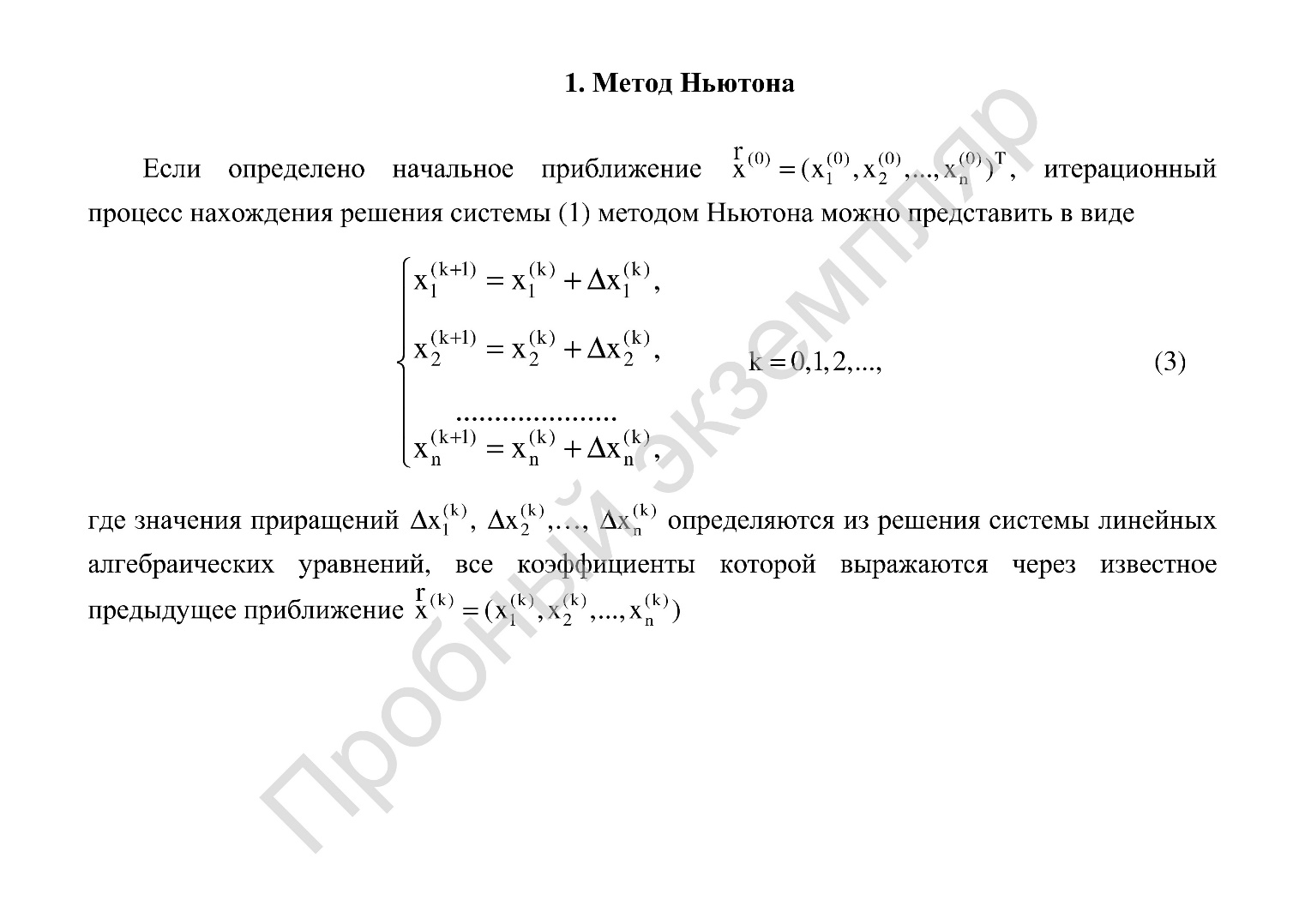
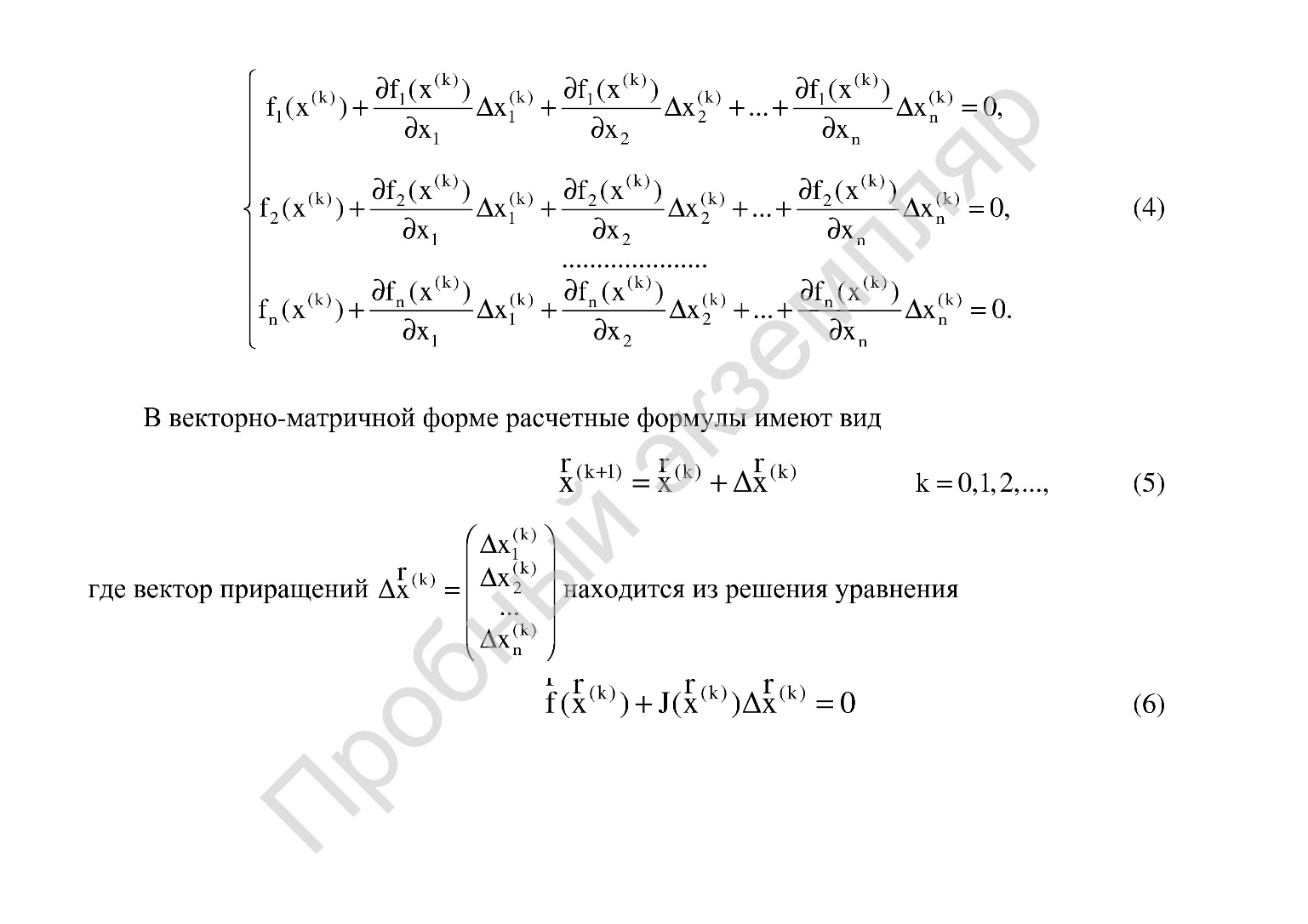


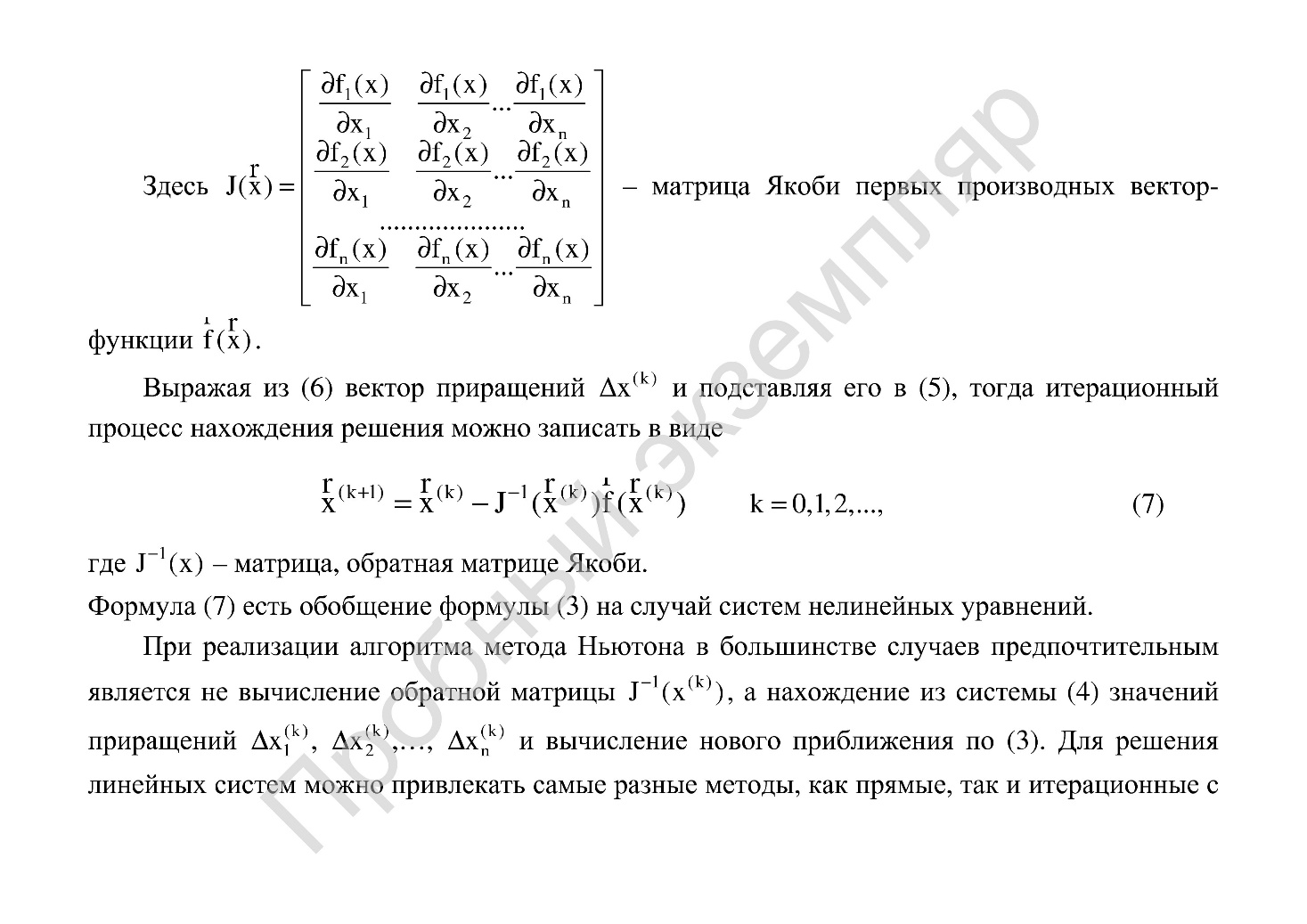
1. **Теоретический материал**

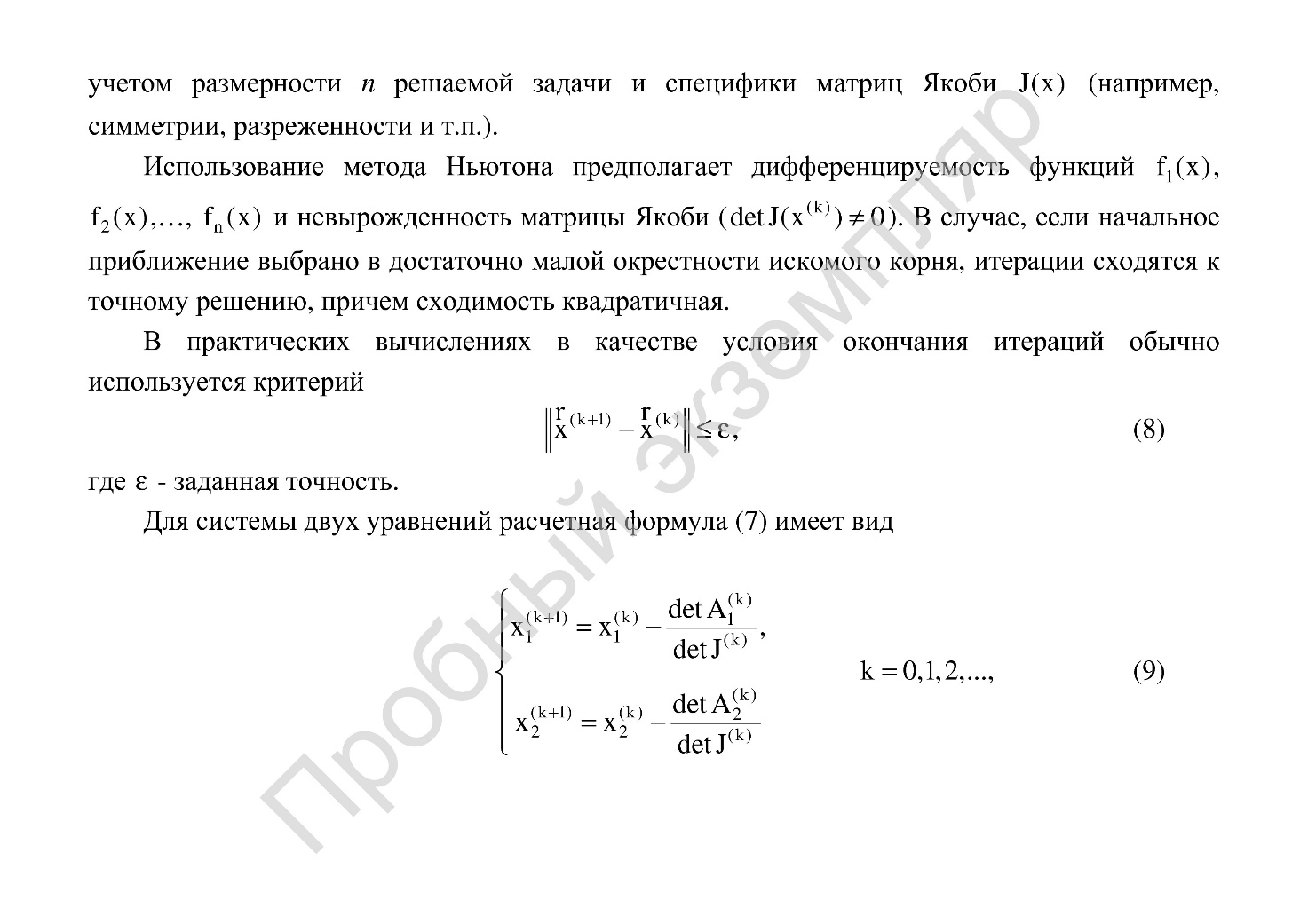
*Алгоритм метода итерации*

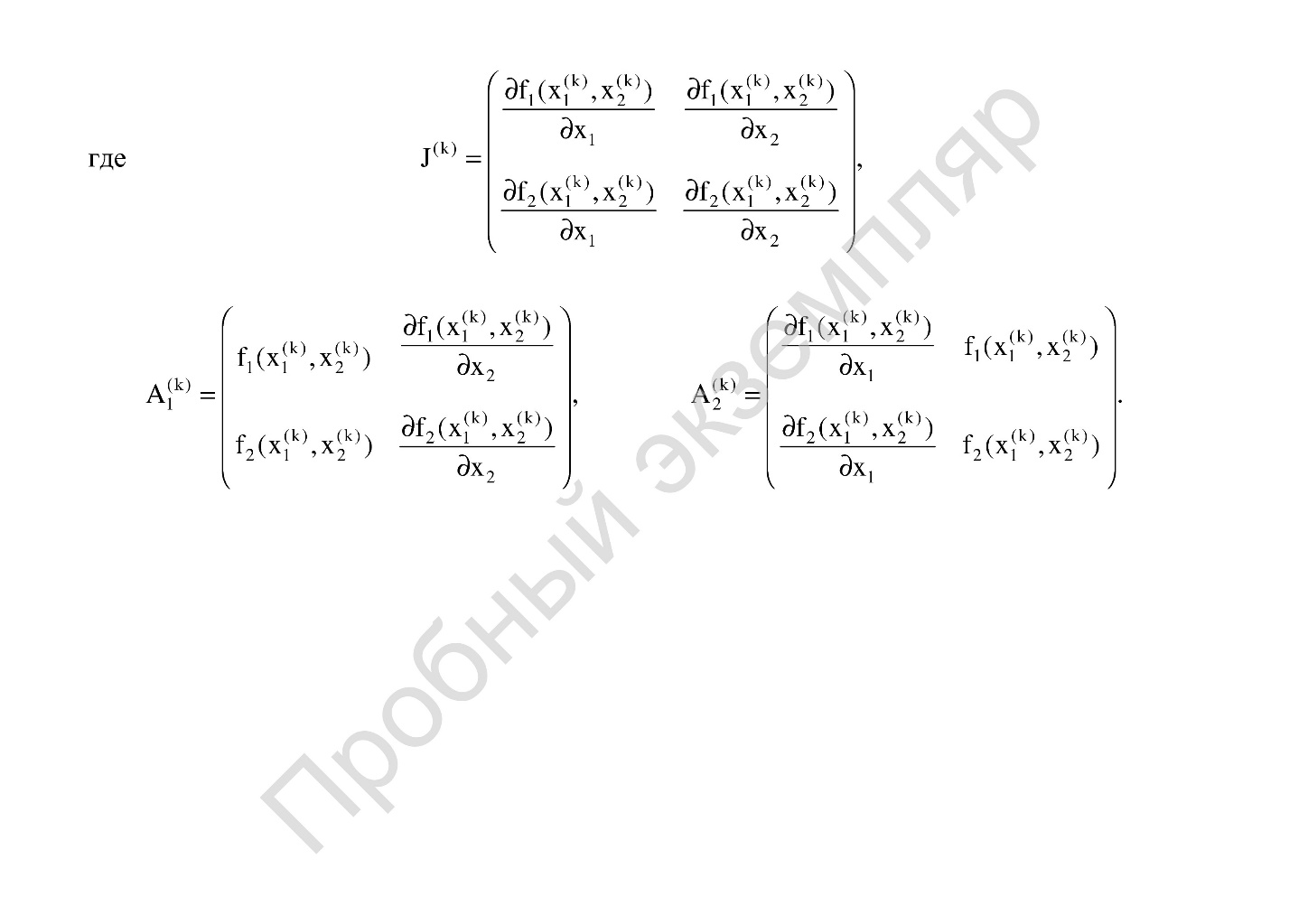
**

*Алгоритм метода Ньютона*

** 

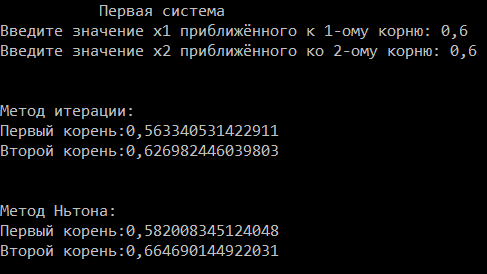




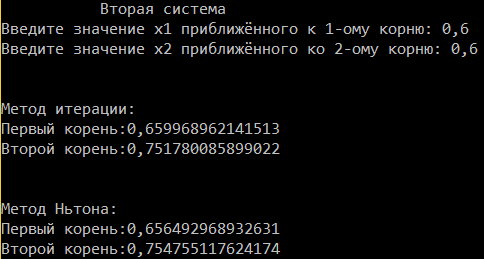


1. **Результат вычислений**

Результат вычислений для первой системы уравнений:



Для второй системы уравнений:



1. **Текст программы**

*Метод итерации*

static double iter\_function\_x1(System system, double x1, double x2, double[] c)

{

return x1 + c[0] \* system.f1(x1, x2) + c[1] \* system.f2(x1, x2);

}

static double iter\_function\_x2(System system, double x1, double x2, double[] c)

{

return x2 + c[0] \* system.f1(x1, x2) + c[1] \* system.f2(x1, x2);

}

static void iteration\_method(System system, double clone\_x1, double clone\_x2, double fault = 0.001)

{

double x1 = clone\_x1;

double x2 = clone\_x2;

double[,] a = { { system.df1dx1(x1, x2), system.df2dx1(x1, x2) }, { system.df2dx1(x1, x2), system.df2dx2(x1, x2) } };

double[] solve\_for\_x = { -1, 0 };

alglib.rmatrixsolve(a, 2, solve\_for\_x, out int t, out alglib.densesolverreport s, out double[] c1);

double[] solve\_for\_y = { 0, -1 };

alglib.rmatrixsolve(a, 2, solve\_for\_y, out int t1, out alglib.densesolverreport s1, out double[] c2);

double x1\_temp, x2\_temp;

while ( ( Math.Abs(system.f1(x1, x2)) > fault ) && ( Math.Abs(system.f2(x1, x2)) > fault ) )

{

x1\_temp = x1;

x2\_temp = x2;

x1 = iter\_function\_x1(system,x1\_temp,x2\_temp,c1);

x2 = iter\_function\_x2(system, x1\_temp, x2\_temp, c2);

}

Console.WriteLine("Первый корень:" + x1);

Console.WriteLine("Второй корень:" + x2);

}

*Метод Ньютона*

static void newton\_method(System system, double clone\_x1, double clone\_x2, double fault = 0.001)

{

double x1 = clone\_x1;

double x2 = clone\_x2;

while ( ( Math.Abs(system.f1(x1, x2)) > fault ) && ( Math.Abs(system.f2(x1, x2)) > fault ) )

{

x1 = x1 - ( system.f1(x1, x2) \* system.df2dx2(x1, x2) - system.df1dx2(x1, x2) \* system.f2(x1, x2) ) /

( system.df1dx1(x1, x2) \* system.df2dx2(x1, x2) - system.df2dx1(x1, x2) \* system.df1dx2(x1, x2) );

x2 = x2 - ( system.df1dx1(x1, x2) \* system.f2(x1, x2) - system.f1(x1, x2) \* system.df2dx1(x1, x2) ) /

( system.df1dx1(x1, x2) \* system.df2dx2(x1, x2) - system.df2dx1(x1, x2) \* system.df1dx2(x1, x2) );

}

Console.WriteLine("Первый корень:" + x1);

Console.WriteLine("Второй корень:" + x2);

}