МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

КАФЕДРА ИНЖЕНЕРНОЙ КИБЕРНЕТИКИ

ГРУППА БПМ-16-2

ЛАБОРАТОРНАЯ РАБОТА 2

ПО КУРСУ: ЧИСЛЕННЫЕ МЕТОДЫ

СТУДЕНТ МАЛЫНКОВСКИЙ О.В.

ПРЕПОДАВАТЕЛЬ ГОПЕНГАУЗ В.И.

2018г.

ВАРИАНТ 10

Задание 3

А) Подобрать СНУ из трёх уравнений с тремя неизвестными, так чтобы одно из решений было вам заранее известно. С помощью метода покоординатного спуска прийти к этому решению. Фиксируя по очереди каждую из координат построить графики вблизи решения. Изобразить на этих рисунках несколько шагов метода.

$$\begin{cases} 2e^x - \ln y + z = 0 \\ \sin x + 4y - z^2 = 0 \\ x + y^3 + \frac{1}{2}z = 0 \end{cases}$$

Решение системы: x = 0; y = 1; z = -2

Введём целевую функцию Φ(x,y,z):

Φ(x,y,z) = $(2e^x - \ln y + z)^2 + (\sin x + 4y - z^2)^2 + (x + y^3 + \frac{1}{2}z)^2$

Тогда вместо решения СНУ можно минимизировать целевую функцию и так прийти к решению. Будем по очереди фиксировать каждую из координат и сдвигать её на некоторое α, так чтобы целевая функция минимизировалась

Итерации построим таким образом: $x^{(k+1)} = x^{(k)} \pm \alpha p_k$

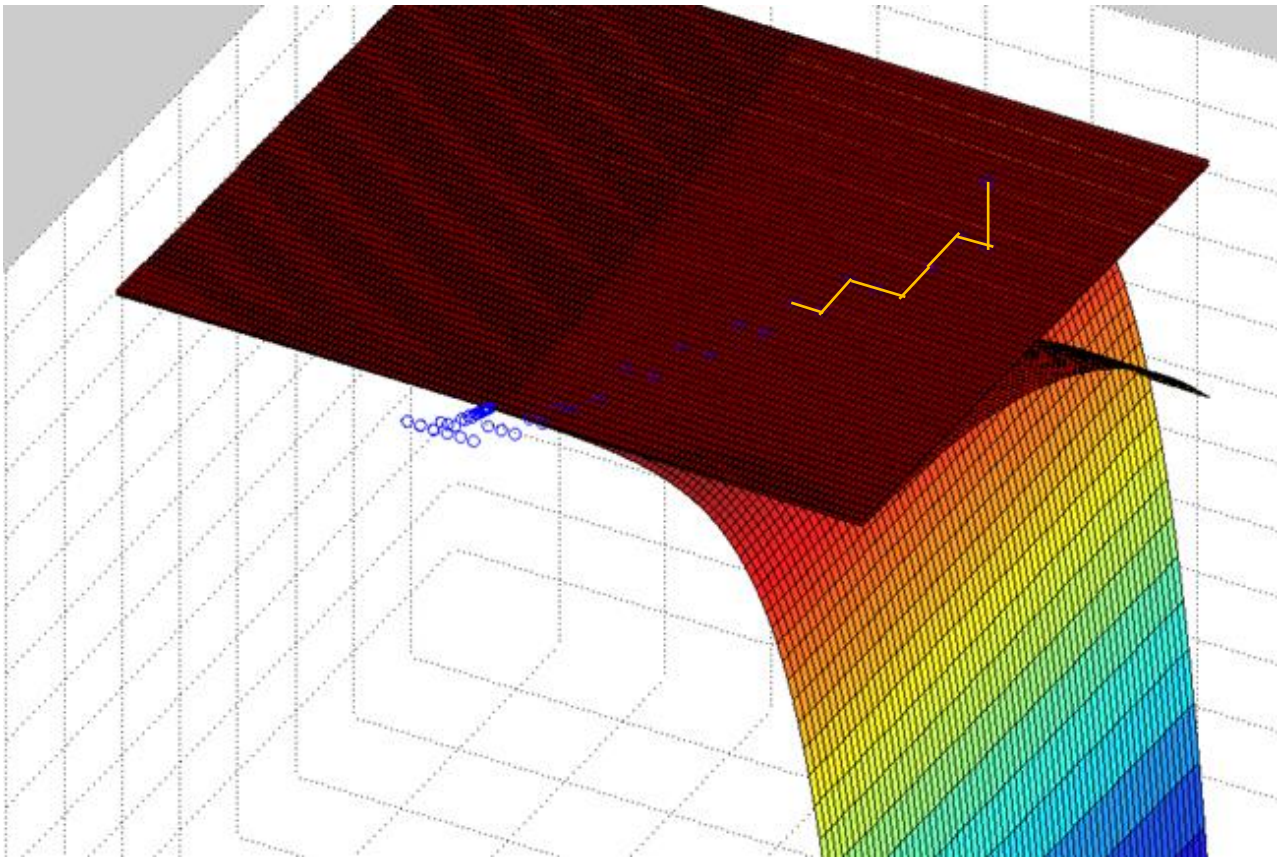
$$p_k = e_{i_k}, \qquad i_k = k - n\left[\frac{k}{n}\right] + 1, \quad e_{i_k} = (0,0,\dots,0,1,0,\dots,0)$$

$x^{(k+1)} = x^{(k)} \pm \alpha p_k$ принимаем в случае, если $\Phi\left(x^{(k)} \pm \alpha p_k\right) < \Phi\left(x^{(k)}\right)$

Если при очередном проходе по всем координатам не удалось провести минимизацию, тогда уменьшаем значение α

```
    class Program
    {
        static double Loss_function(double[] A)//целевая функция
        {
            double x = A[0]; double y = A[1]; double z = A[2];
            double F = (2 * Math.Exp(x) - Math.Log(y) + z) * (2 * Math.Exp(x) -
Math.Log(y) + z) + (Math.Sin(x) + 4 * y - z * z) * (Math.Sin(x) + 4 * y - z * z) + (x + y
* y * y + 0.5 * z) * (x + y * y * y + 0.5 * z);
            return F;
        }
        static double[] MultipleNum(double[] A, double k)//
        static double[] addMatrix(double[] A, double[] B)
        static double[] substractMatrix(double[] A, double[] B)
        static void Main(string[] args)
        {
            double[] X = new double[3];
            for (int i = 0; i < 3; i++)
                X[i] = 5.0;
            double[,] E = new double[3, 3];
            E[0, 0] = 1.0;     E[1, 1] = 1.0;      E[2, 2] = 1.0;
            double alpha = 0.5;  int k = 0;
            double[] previousValues = X; double[] currentValues = X;
            double accuracy = 0.00001; int iterations = 15000; int index;
            double[] P = new double[3];
            double[] temp = new double[3];
            while (true)
```

```
{
    bool t = false;
    index = k - 3 * (k / 3);
    for (int i = 0; i < 3; i++)
        P[i] = E[index, i];
    P = MultipleNum(P, alpha);
    temp = addMatrix(previousValues, P);
    if (Loss_function(temp) < Loss_function(previousValues))
    {
        currentValues = temp;
        t = true;
    }
    else
    {
        temp = substractMatrix(previousValues, P);
        if (Loss_function(temp) < Loss_function(previousValues))
        {
            currentValues = temp;
            t = true;
        }
    }
    k++;
    if ((index == 2) && (currentValues == previousValues))
        alpha /= 2;
    if (t)
    {
        ….
        if ((max <= accuracy) || (k > iterations))
        {
            previousValues = currentValues;
            break;
        }
    }
    previousValues = currentValues;
```



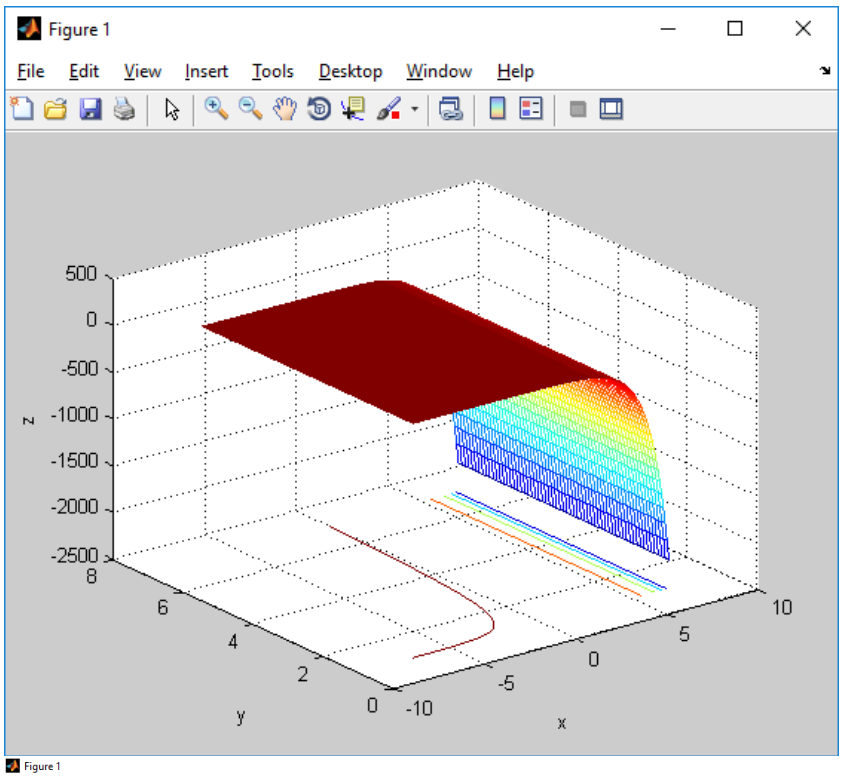


Figure 1
File Edit View Insert Tools Desktop Window Help
500
0
-500
-1000
-1500
-2000
-2500
z
8
6
4
2
0
y
-10
-5
0
5
10
x

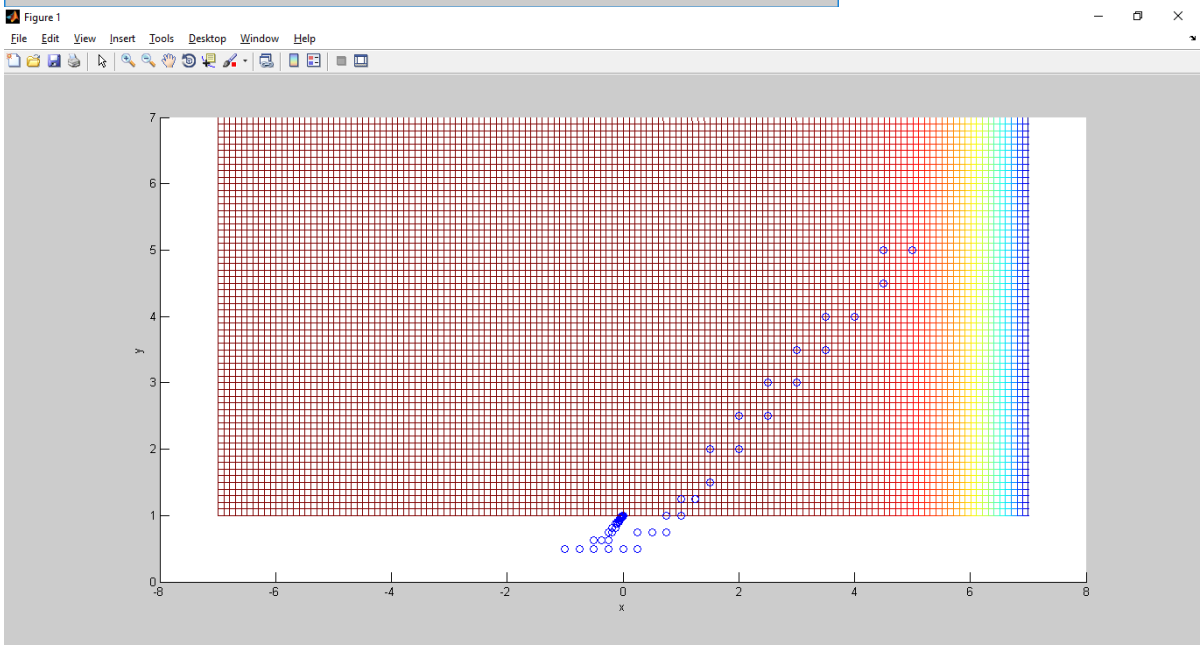


Figure 1
File Edit View Insert Tools Desktop Window Help
7
6
5
4
3
2
1
0
y
-8
-6
-4
-2
0
2
4
6
8
x

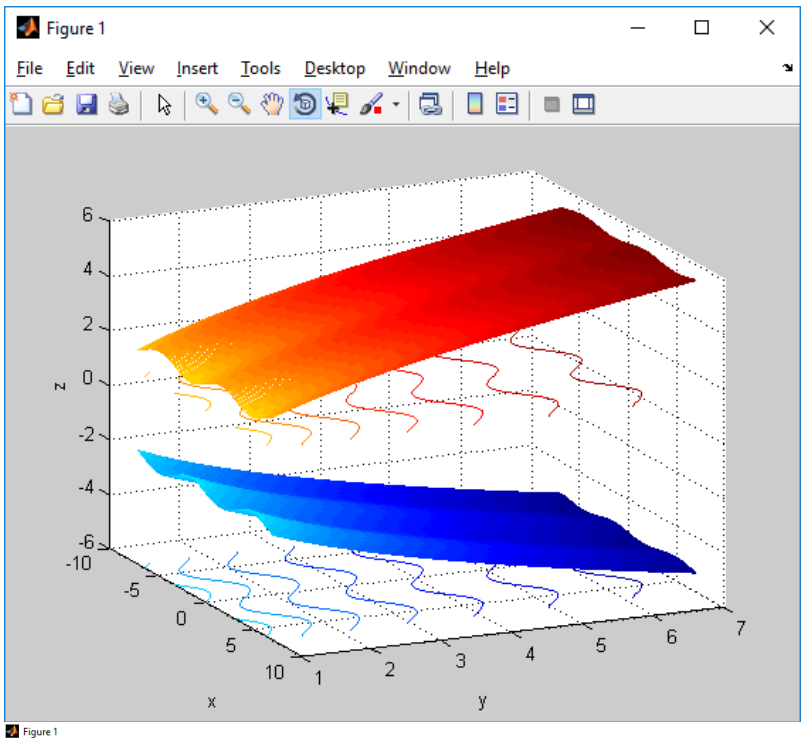


Figure 1
File Edit View Insert Tools Desktop Window Help
z
x
y

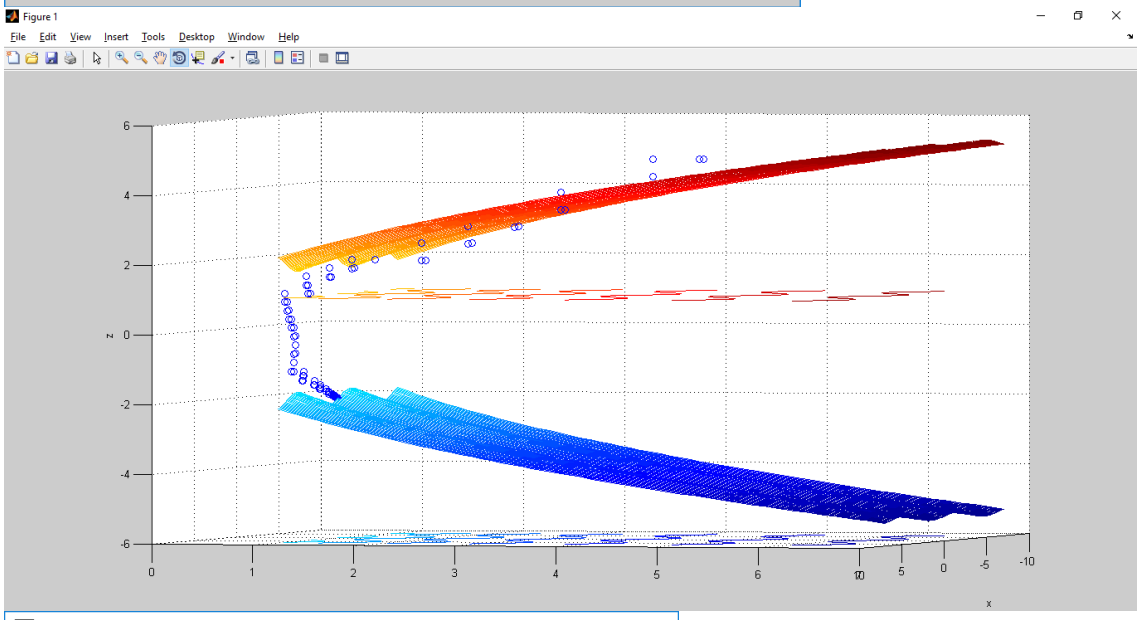


Figure 1
File Edit View Insert Tools Desktop Window Help
z
x

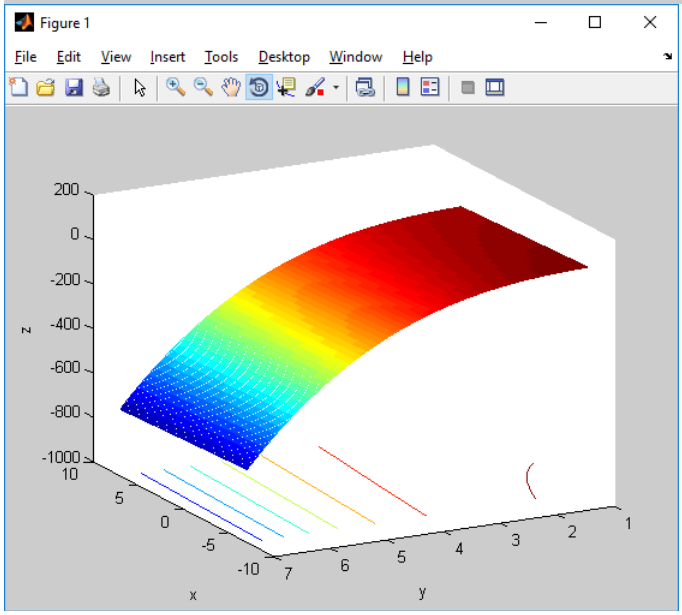
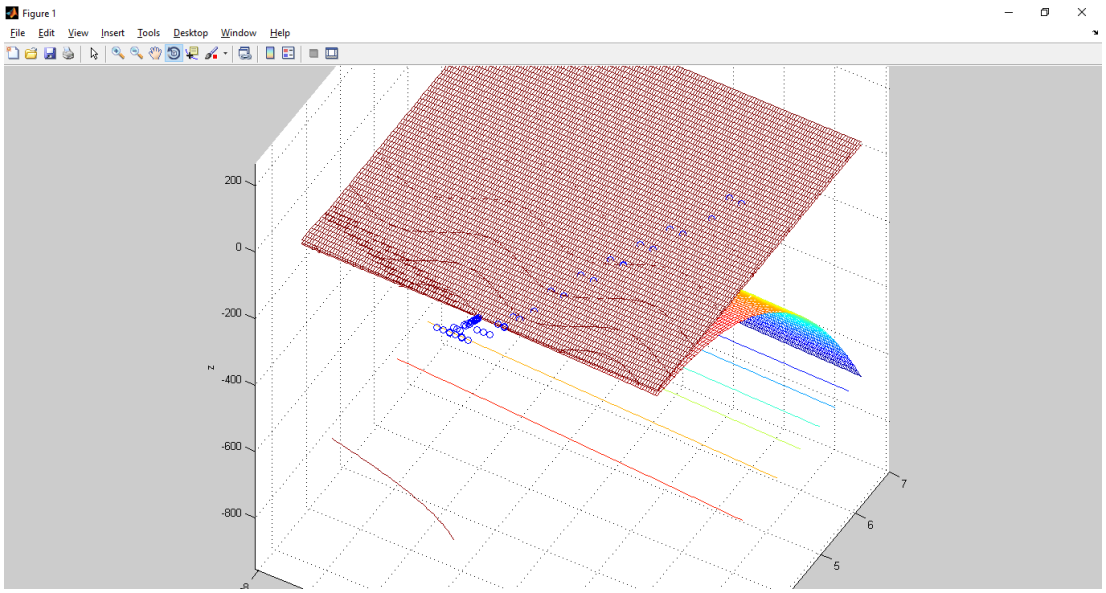
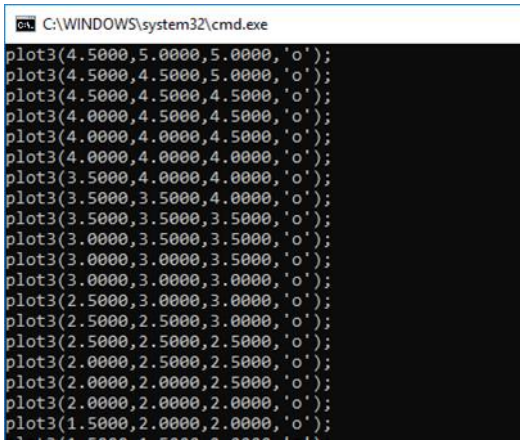
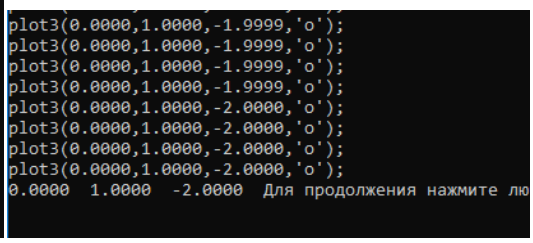


Figure 1
File Edit View Insert Tools Desktop Window Help
z
x
y







Б) Аппроксимировать функцию, заданную на отрезке аналитически, при помощи МНК линейными комбинациями ортогональных полиномов. Результат вывести в виде таблицы и графика.

Ортогональные полиномы построим следующим образом:

$$q_0 = 1, (q_k, q_j) = 0, j \neq k, (q_k, q_k) = \|q_k\|^2 \neq 0$$

$$q_1 = x - \frac{(x, q_0(x))}{(q_0(x), q_0(x))}$$

$$q_{k+1} = xq_k(x) - \frac{(xq_k(x), q_k(x))}{(q_k(x), q_k(x))} q_k(x) - \frac{(xq_k(x), q_{k-1}(x))}{(q_{k-1}(x), q_{k-1}(x))} q_{k-1}(x), \quad k = 1,2,3, \ldots$$

Согласно методу МНК:

$$\int_a^b (c_0\varphi_0(x) + c_1\varphi_1(x) + \cdots + c_m\varphi_m(x) - f(x))^2 dx \rightarrow \min$$

Необходимые (и достаточные) для этого условия выражаются системой:

$$\begin{cases} (\varphi_0(x), \varphi_0(x)c_0 + (\varphi_0(x), \varphi_1(x)c_1 + \cdots + (\varphi_0(x), \varphi_m(x)c_m) = (\varphi_0(x), f), \\ (\varphi_1(x), \varphi_0(x)c_0 + (\varphi_1(x), \varphi_1(x)c_1 + \cdots + (\varphi_1(x), \varphi_m(x)c_m) = (\varphi_1(x), f), \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ (\varphi_m(x), \varphi_0(x)c_0 + (\varphi_m(x), \varphi_1(x)c_1 + \cdots + (\varphi_m(x), \varphi_m(x)c_m) = (\varphi_m(x), f) \end{cases}$$

В силу того, что у нас ортогональные полиномы, слева у нас останется диагональная матрица, остальные элементы равны 0, откуда сразу можно найти необходимые коэффициенты.

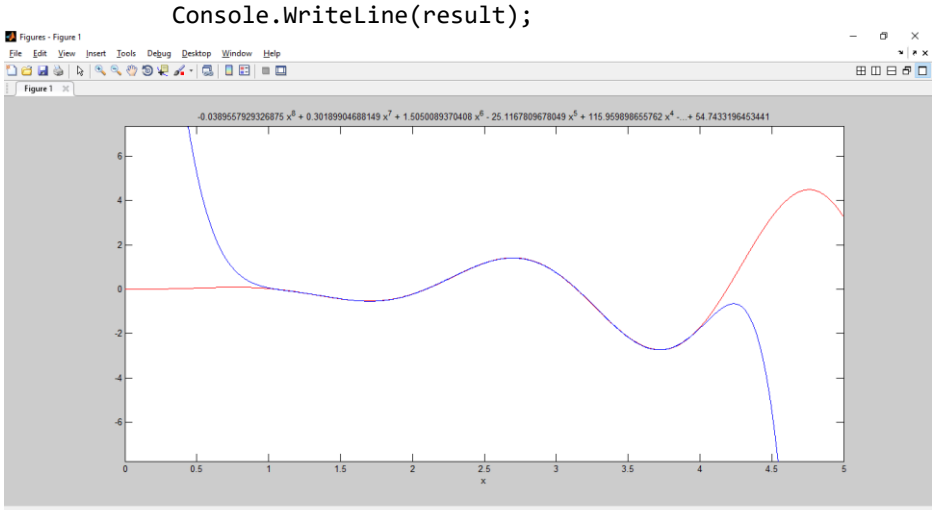
$$c_i = \frac{(\varphi_i(x), f)}{(\varphi_i(x), \varphi_i(x))} = \frac{(\varphi_i(x), f)}{\|\varphi_i(x)\|^2}$$

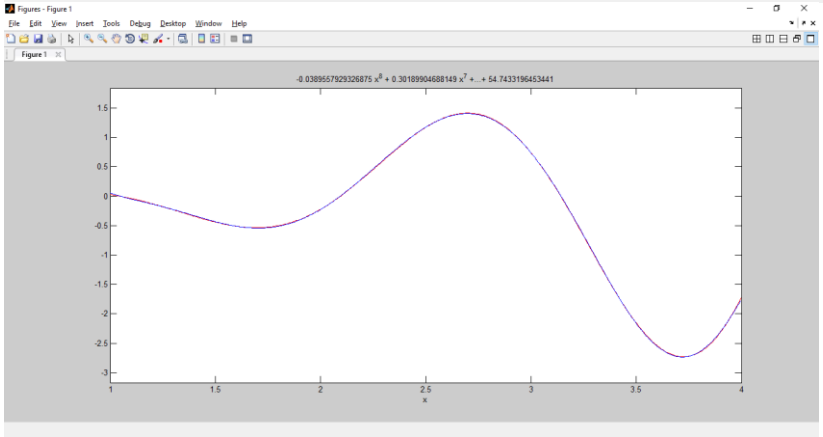
Возьмём такую аналитическую функцию:

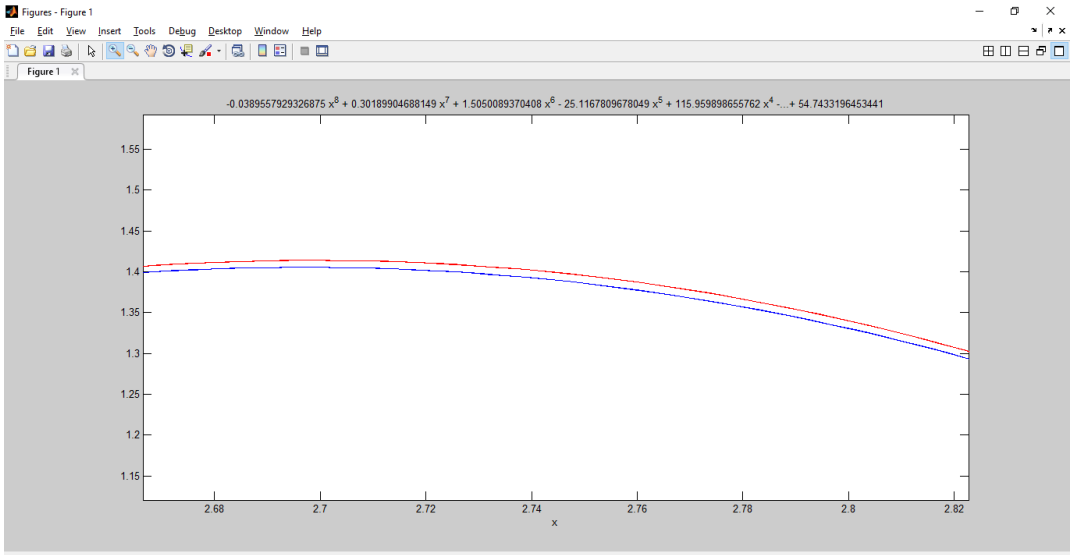
$$y = \frac{\sin(3x) * x^2}{5}$$

```
    public class Polynomial

    class Program
    {
        static double Fun(double x)
        {
            return Math.Sin(3*x) * x * x / 5;
        }
        static double Scalar(Polynomial A, Polynomial B)
        {
            Polynomial p = A * B;
            //Console.WriteLine(p);
            double result = p.Integral().GetSolution(4) - p.Integral().GetSolution(1);
            return result;
        }
        static Polynomial[] Make_ort_system(int m)//m число замеров функции
        {
            Polynomial[] pol = new Polynomial[m];
            Polynomial x = new Polynomial(new double[] { 1, 0 });
            pol[0] = new Polynomial(new double[] { 1 });
            pol[1] = x + new Polynomial(new double[1] { -
Scalar(x,pol[0])/Scalar(pol[0],pol[0])});
            for (int i = 2; i < m ; i++)
            {
                pol[i] = x * pol[i - 1] + (pol[i - 1] * (-Scalar(x * pol[i - 1], pol[i -
1]) / Scalar(pol[i - 1], pol[i - 1]))) + (pol[i - 2] * (-Scalar(x * pol[i - 1], pol[i -
2]) / Scalar(pol[i - 2], pol[i - 2])));
            }
            return pol;
        }
        static Polynomial SLAU(double[,] A,double[] B)
        {

            double[] t = new double[A.GetLength(0)];
            for (int i = 0; i < A.GetLength(0); i++)
            {
                t[i] = B[i] / A[i, i];
            }
            return new Polynomial(t);
        }
        static double Integral_(Polynomial n)
        {
            double step = 1000000;
            double h = 3.0 / step;
```

```
        double res = Fun(1.0)* n.GetSolution(1.0)*h;
        for (int i = 1; i < step; i++)
            res += Fun(1.0 + h * i) * n.GetSolution(1.0 + h * i) * h;
        return res;

    }
    static void Main(string[] args)
    {
        int step = 9;
        Polynomial[] n = Make_ort_system(step);
        double[,] A = new double[step, step];
        double[] B = new double[step];
        for (int i = 0; i < A.GetLength(0); i++)
        {
            for (int j = 0; j < A.GetLongLength(1); j++)
                A[i, j] = Scalar(n[i], n[j]);
            B[i] = Integral_(n[i]);
        }
        double[] t = new double[A.GetLength(0)];
        Polynomial[] res = new Polynomial[step];
        double[] ttt = new double[step];
        for (int i = 0; i < step; i++)
            ttt[i] = 0;
        Polynomial result = new Polynomial(ttt);
        for (int i = 0; i < A.GetLength(0); i++)
        {
            t[i] = B[i] / A[i, i];
            res[i] = n[i] * t[i];
            result += res[i];
        }
        Console.WriteLine(result);
```







C:\WINDOWS\system32\cmd.exe

```
3,00  0,00  0,00  0,00  0,00  0,00  0,00  0,00  0,00          -1,04
0,00  2,25  0,00  0,00  0,00  0,00  0,00  0,00  0,00          -1,44
0,00  0,00  1,35  0,00  0,00  0,00  0,00  0,00  0,00          -1,65
0,00  0,00  0,00  0,78  0,00  0,00  0,00  0,00  0,00          -0,49
0,00  0,00  0,00  0,00  0,45  0,00  0,00  0,00  0,00          0,58
0,00  0,00  0,00  0,00  0,00  0,25  0,00  0,00  0,00          0,32
0,00  0,00  0,00  0,00  0,00  0,00  0,14  0,00  0,00          -0,03
0,00  0,00  0,00  0,00  0,00  0,00  0,00  0,08  0,00          -0,04
0,00  0,00  0,00  0,00  0,00  0,00  0,00  0,00  0,05          0,00
-0,0389557929326875*x^8 + 0,30189904688149*x^7 + 1,5050089370408*x^6 + -25,11678096
78049*x^5 + 115,959898655762*x^4 + -265,390435819631*x^3 + 327,679201508534*x^2 + -
209,587407253363*x + 54,7433196453441

Analytic function                    Approximant
X         Y                          X         Y
1,0000    0,0282                   1,0000    0,0557
1,1579    -0,0874                   1,1579    -0,0980
1,3158    -0,2498                   1,3158    -0,2440
1,4737    -0,4160                   1,4737    -0,4079
1,6316    -0,5236                   1,6316    -0,5258
1,7895    -0,5075                   1,7895    -0,5165
1,9474    -0,3238                   1,9474    -0,3290
2,1053    0,0289                   2,1053    0,0328
2,2632    0,4968                   2,2632    0,5059
2,4211    0,9736                   2,4211    0,9791
2,5789    1,3211                   2,5789    1,3176
2,7368    1,4038                   2,7368    1,3944
2,8947    1,1307                   2,8947    1,1245
3,0526    0,4915                   3,0526    0,4952
3,2105    -0,4233                   3,2105    -0,4130
3,3684    -1,4277                   3,3684    -1,4231
3,5263    -2,2742                   3,5263    -2,2827
3,6842    -2,7103                   3,6842    -2,7188
3,8421    -2,5462                   3,8421    -2,5335
4,0000    -1,7170                   4,0000    -1,7529
Для продолжения нажмите любую клавишу . . .
```

Задание 9

А) Подобрать СНУ из трёх уравнений с тремя неизвестными, так чтобы одно из решений было вам заранее известно. С помощью метода градиентного спуска прийти к этому решению. Фиксируя по очереди каждую из координат построить графики вблизи решения. Изобразить на этих рисунках несколько шагов метода.

$$\begin{cases} 2e^x - \ln y + z = 0 \\ \sin x + 4y - z^2 = 0 \\ x + y^3 + \frac{1}{2}z = 0 \end{cases}$$

Решение системы: x = 0; y = 1; z = -2

Введём целевую функцию Ф(x,y,z):

Ф(x,y,z) = $(2e^x - \ln y + z)^2 + (\sin x + 4y - z^2)^2 + (x + y^3 + \frac{1}{2}z)^2$

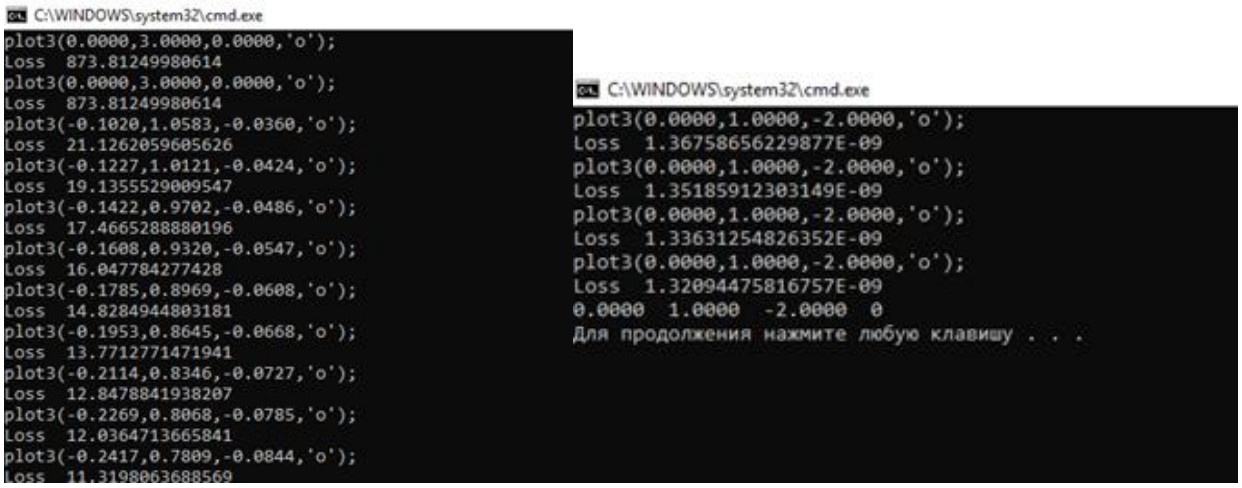
Тогда вместо решения СНУ можно минимизировать целевую функцию и так прийти к решению. Будем по очереди фиксировать каждую из координат и сдвигать её на некоторое α, так чтобы целевая функция минимизировалась

Итерации построим таким образом: $x^{(k+1)} = x^{(k)} - \alpha\, grad(x^{(k)})$

То есть здесь, отличии от покоординатного спуска мы будем спускать не по отдельным координатам, а одновременно по всем двигаясь по направлению антиградиента с некоторым шагом.

```
    class Program
    {
        static double Loss_function(double[] A)//целевая функция
        {
            double x = A[0];
            double y = A[1];
            double z = A[2];
            double F = (2 * Math.Exp(x) - Math.Log(y) + z) * (2 * Math.Exp(x) -
Math.Log(y) + z) + (Math.Sin(x) + 4 * y - z * z) * (Math.Sin(x) + 4 * y - z * z) + (x + y
* y * y + 0.5 * z) * (x + y * y * y + 0.5 * z);
            return F;
        }

        static double[] Gradient_analitic(double[] A)
        {
            double x = A[0];
            double y = A[1];
            double z = A[2];
            double X = 4 * Math.Exp(x) * (2 * Math.Exp(x) - Math.Log(y) + z) + 2 *
Math.Cos(x) * (Math.Sin(x) + 4 * y - z * z) + 2 * (x + y * y * y + 0.5 * z);
            double Y = 2 * ((-1.0) / y) * (2 * Math.Exp(x) - Math.Log(y) + z) + 8 *
(Math.Sin(x) + 4 * y - z * z) + 6 * y * y * (x + y * y * y + 0.5 * z);
            double Z = 2 * (2 * Math.Exp(x) - Math.Log(y) + z) - 4 * z * (Math.Sin(x) + 4
* y - z * z) + (x + y * y * y + 0.5 * z);
            double[] res = new double[3];
            res[0] = X; res[1] = Y; res[2] = Z;
            return res;
        }

        static double[] MultipleNum(double[] A, double k)//        {}
        static double[] addMatrix(double[] A, double[] B)        {}
        static double[] substractMatrix(double[] A, double[] B)        {}
```

```
        static void Main(string[] args)
        {
            double[] X = new double[3];
            for (int i = 0; i < 3; i++)
                X[i] = 0.0;
            X[1] = 3.0; double alpha = 0.005; int k = 0;
            double[] previousValues = X; double[] currentValues = X;
            double accuracy = 0.0000001; int iterations = 15000;
            double[] P = new double[3];
            double[] temp = new double[3];
            while (true)
            {
                bool t = false;
                temp = substractMatrix(previousValues, MultipleNum(
Gradient_analitic(previousValues),alpha));
                if (Loss_function(temp) < Loss_function(previousValues))
                {
                    currentValues = temp;
                    t = true;
                }
                k++;
                if (currentValues == previousValues)
                    alpha /= 2;
                if (t)
                {
                    …
                    if ((max <= accuracy) || (k > iterations))
                    {
                        previousValues = currentValues;
                        break;
                    }
```



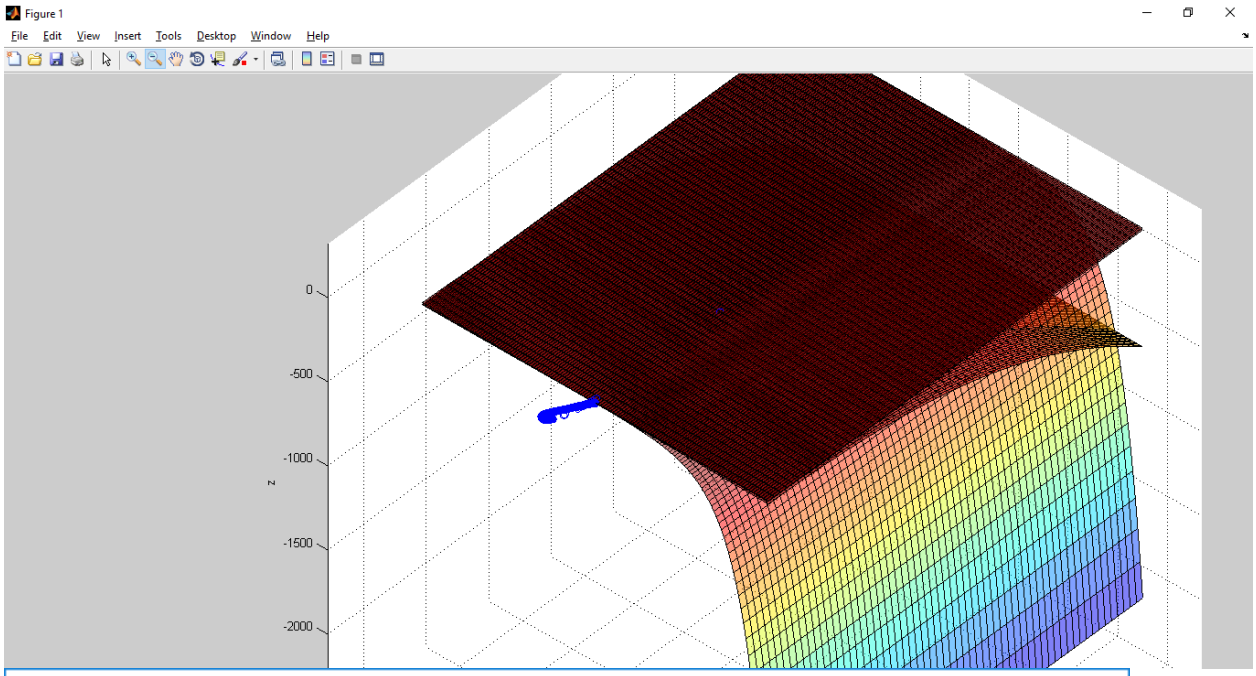


Figure 1
File Edit View Insert Tools Desktop Window Help
0
-500
-1000
-1500
-2000
z

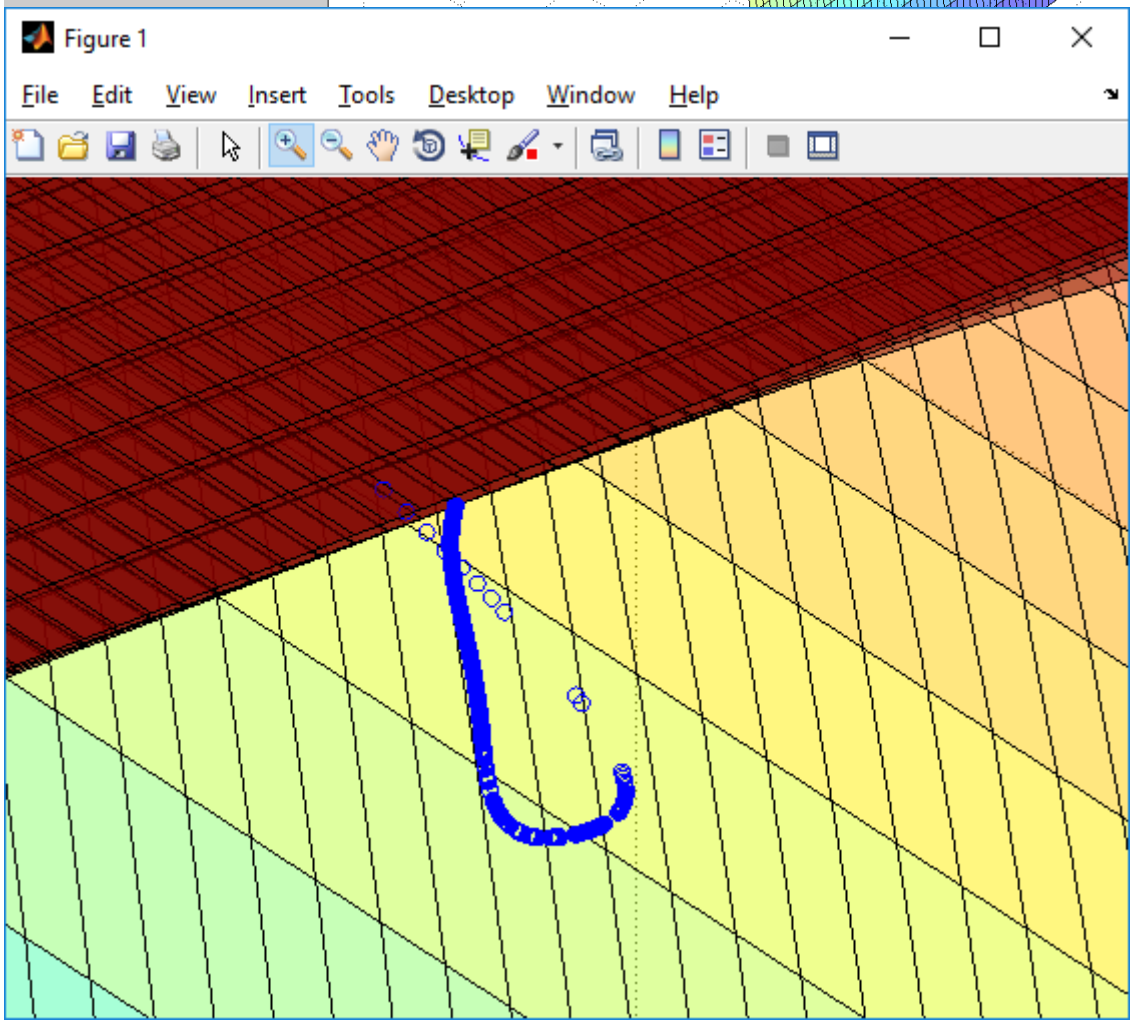
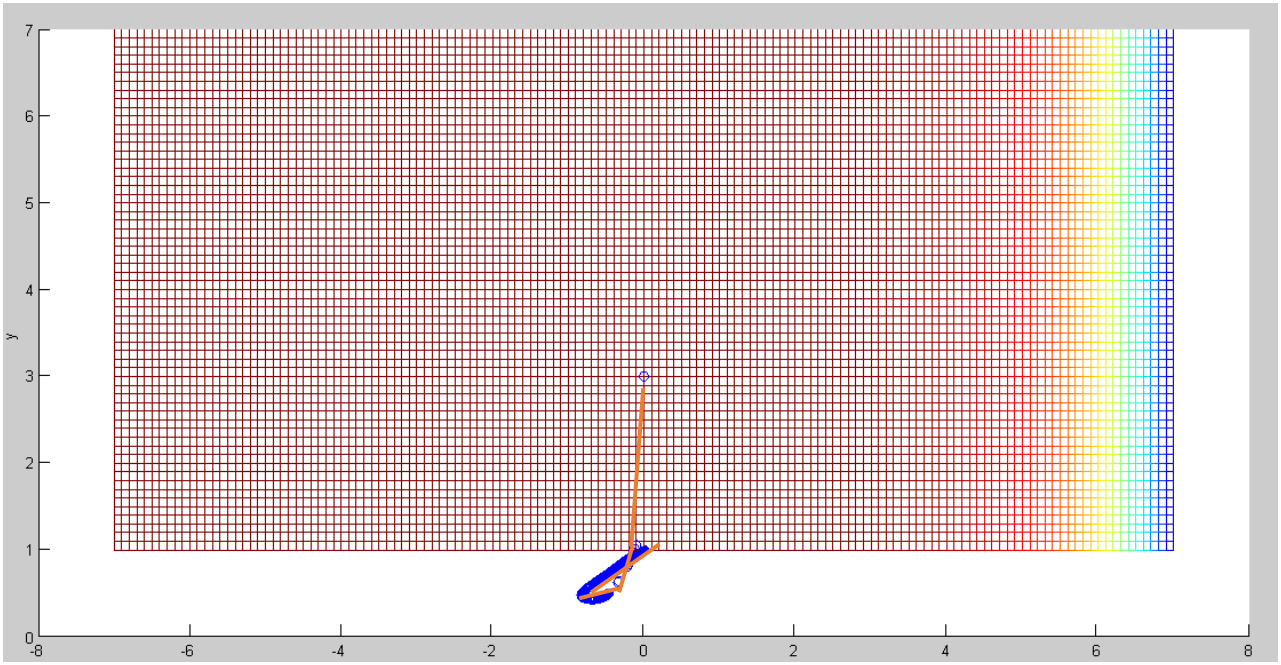


Figure 1
File Edit View Insert Tools Desktop Window Help



Б) Подобрать нелинейное уравнение с одним неизвестным, так чтобы одно из решений было известно заранее. Прийти к этому решению обратным интерполированием, взяв за основу метода первую интерполяционную формулу Ньютона по не равностоящим узлам.

Пусть нелинейное уравнение имеет такой вид:$x^3 - 7e^{x-2}$ Найдём x, для которого $\hat{y} = 1$ (при x = 2)

$$x^{(k+1)} = \varphi(x^{(k)})$$

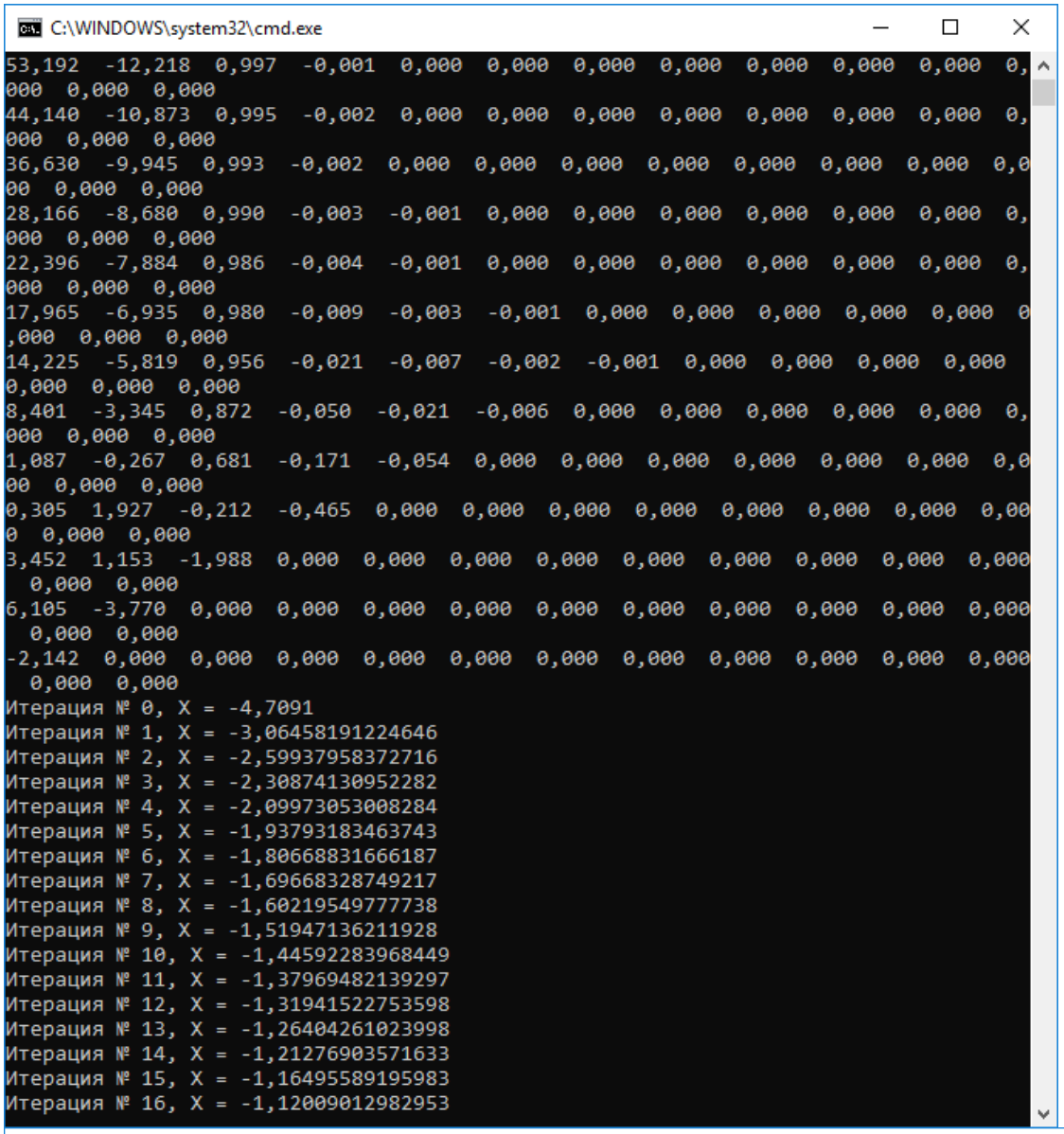
$$\varphi(x) = x_0 - \frac{1}{f(x_0, x_1)}(f(x_0) - \hat{y} + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \cdots + f(x_0, x_1, \dots, x_n)(x - x_0)(x - x_1)\dots(x - x_{n-1}))$$

$$f(x_{i-1}, x_i, \dots, x_{i+k}) = \frac{f(x_i, x_{i+1}, \dots, x_{i+k}) - f(x_{i-1}, x_i, \dots, x_{i+k-1})}{x_{i+k} - x_{i-1}}$$

Разделённые разности хранятся в массиве A (n - 1 x n - 1), после их просчитывания для вычисления очередной итерации нужно испольхователь элементы только первой строки матрицы A

```
public class Polynomial…

class Program
{
    static double Fun(double x)
    {
        return (x * x * x - 7*Math.Exp(x-2));
    }

    static Polynomial Iteration(double[] X,double[,] A, double y_)
    {
        Polynomial result = new Polynomial(new double[] { X[0] });
        Polynomial[] temp = new Polynomial[A.GetLength(0)];
        for (int i = 1; i < A.GetLength(0); i++)
        {
            temp[i] = new Polynomial(new double[1] { 1 });
```

```
            for (int j = 0; j <= i; j++)
            {
                temp[i] =  temp[i] * (new Polynomial(new double[2] { 1, -X[j] }));
            }
            temp[i] = temp[i] * A[0, i];
        }
        Polynomial t = new Polynomial(new double[] { Fun(X[0])- y_});
        for (int i = 1; i < temp.Length; i++)
            t = t + temp[i];
        t = t / (-A[0, 0]);
        result = result + t;
        return result;

    }

    static void Main(string[] args)
    {
        int n = 15;  double Y_ = 1.0;
        double[] X = new double[n];     double[] Y = new double[n];
        Random rand = new Random();
        for (int i = 0; i < n; i++)
            X[i] = rand.Next(-5, 5) + rand.NextDouble();
        Array.Sort(X);
        for (int i = 0; i < n; i++)
            Y[i] = Fun(X[i]);
        double[,] A = new double[n - 1, n - 1];
        for (int i = 0; i < n - 1; i++)
            A[i, 0] = (Y[i + 1] - Y[i]) / (X[i + 1] - X[i]);
        for (int j = 1; j < n-1; j++)
        {
            for (int i = 0; i < n - 1 - j; i++)
            {
                A[i, j] = (A[i + 1, j - 1] - A[i, j - 1]) / (X[i + j + 1] - X[i]);
            }
        }
        Polynomial a = Iteration(X, A, Y_);
        int iterations = 1500;  double epsilon = 0.00000001; int k = 0;
        double res = X[0];
        Console.WriteLine("Итерация № {0}, X = {1:N4}", k, res);
        double temp = X[0];    double t;
        while (true)
        {
            t = a.GetSolution(temp);
            k++;
            Console.WriteLine("Итерация № {0}, X = {1}", k, t);

            if ((Math.Abs(t - temp) <= epsilon) || (k > iterations))
            {
                temp = t;
                break;
            }
            temp = t;
        }
```


C:\WINDOWS\system32\cmd.exe
53,192 -12,218 0,997 -0,001 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
44,140 -10,873 0,995 -0,002 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
36,630 -9,945 0,993 -0,002 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
28,166 -8,680 0,990 -0,003 -0,001 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
22,396 -7,884 0,986 -0,004 -0,001 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
17,965 -6,935 0,980 -0,009 -0,003 -0,001 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
14,225 -5,819 0,956 -0,021 -0,007 -0,002 -0,001 0,000 0,000 0,000 0,000 0,000 0,000 0,000
8,401 -3,345 0,872 -0,050 -0,021 -0,006 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
1,087 -0,267 0,681 -0,171 -0,054 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
0,305 1,927 -0,212 -0,465 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
3,452 1,153 -1,988 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
6,105 -3,770 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
-2,142 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000 0,000
Итерация № 0, X = -4,7091
Итерация № 1, X = -3,06458191224646
Итерация № 2, X = -2,59937958372716
Итерация № 3, X = -2,30874130952282
Итерация № 4, X = -2,09973053008284
Итерация № 5, X = -1,93793183463743
Итерация № 6, X = -1,80668831666187
Итерация № 7, X = -1,69668328749217
Итерация № 8, X = -1,60219549777738
Итерация № 9, X = -1,51947136211928
Итерация № 10, X = -1,44592283968449
Итерация № 11, X = -1,37969482139297
Итерация № 12, X = -1,31941522753598
Итерация № 13, X = -1,26404261023998
Итерация № 14, X = -1,21276903571633
Итерация № 15, X = -1,16495589195983
Итерация № 16, X = -1,12009012982953
C:\WINDOWS\system32\cmd.exe
Итерация № 276, X = 1,9999994084836
Итерация № 277, X = 1,99999943056092
Итерация № 278, X = 1,9999994509165
Итерация № 279, X = 1,9999994696846
Итерация № 280, X = 1,99999948698904
Итерация № 281, X = 1,99999950294396
Итерация № 282, X = 1,9999995176546
Итерация № 283, X = 1,99999953121801
Итерация № 284, X = 1,99999954372364
Итерация № 285, X = 1,999999555254
Итерация № 286, X = 1,99999956588515
Итерация № 287, X = 1,9999995756872
Было получено следующее решение: 2,0000
5,89701658507562E-14*x^14 + 1,9509920206327E-12*x^13 + 3,23476247844024E-11*x^12 + 3,90080108169016E-10*x^11 + 4,1266238
8284251E-09*x^10 + 4,06184867059716E-08*x^9 + 3,65580042031808E-07*x^8 + 2,93029669923914E-06*x^7 + 2,05241518201175E-05
*x^6 + 0,00012314273635326 1*x^5 + 0,000615677366827715*x^4 + -0,0131346729494084*x^3 + 0,00738803345290378*x^2 + 1,01477
609445775*x + 0,0303734485409262
Для продолжения нажмите любую клавишу . . .