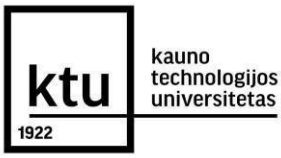


KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

COMPUTER DEPARTMENT

Skaitinių metodų ir algoritmų 3-a projektinė užduotis

Darbą atliko:

IFF 6/8 grupės
studentas
Tadas Laurinaitis

Darbą vertino:

Lekt. Dalia Čalnerytė

Užduotys

I užduotis. Interpoliavimas daugianariu.

1 lentelėje duota interpoliuojamos funkcijos analitinė išraiška. Pateikite interpoliacinės funkcijos išraišką naudodami ***1 lentelėje*** nurodytas bazines funkcijas, kai:

a. Taškai pasiskirstę tolygiai.
b. Taškai apskaičiuojami naudojant Čiobyševo abscises.

Interpoliavimo taškų skaičių parinkite laisvai, bet jis turėtų neviršyti 30. Pateikite du grafikus, kai interpoliacinės funkcijos apskaičiuojamos naudojant skirtingas abscises ir gautas interpoliuojančių funkcijų išraiškas. Tame pačiame grafike vaizduokite duotąją funkciją, interpoliacinę funkciją ir netiktį.

II užduotis. Interpoliavimas daugianariu ir splainu per duotus taškus

Pagal ***2 lentelėje*** pateiktą šalį ir metus, sudaryti interpoliuojančią kreivę 12 mėnesių temperatūroms atvaizduoti nurodytais metodais:

a. Daugianariu, sudarytu naudojant ***1 lentelėje*** nurodytas bazines funkcijas.
b. ***2 lentelėje*** nurodyto tipo splainu.

III užduotis. Parametrinis interpoliavimas.

Naudodami **parametrinio** interpoliavimo metodą ***2 lentelėje*** nurodytu splainu suformuokite ***2 lentelėje*** nurodytos šalies kontūrą. Pateikite pradinius duomenis ir rezultatus, gautus naudojant 10, 20, 50, 100 interpoliavimo taškų.

IV užduotis. Aproksimavimas

Pagal ***2 lentelėje*** nurodytą šalį ir metus mažiausių kvadratų metodu sudarykite aproksimuojančią kreivę 12 mėnesių temperatūroms atvaizduoti naudojant **antros**, **trečios**, **ketvirtos** ir **penktos** eilės daugianarius. Pateikite gautas daugianarių išraiškas.

Pav. #1 Užduočių sąrašas

30	$\frac{\ln(x)}{(\sin(2\cdot x)+1{,}5)} - \cos(\frac{x}{5}); 2 \le x \le 10$	Niutono

30	Kamerūnas	1999	Ermito (Akima)

24.4692	1999	1	CMR
26.1946	1999	2	CMR
27.7291	1999	3	CMR
26.7083	1999	4	CMR
25.3865	1999	5	CMR
24.4072	1999	6	CMR
23.3871	1999	7	CMR
22.9818	1999	8	CMR
23.555	1999	9	CMR
23.8082	1999	10	CMR
24.7706	1999	11	CMR
25.1817	1999	12	CMR

Pav. #2 Užduočių variantų sąrašas

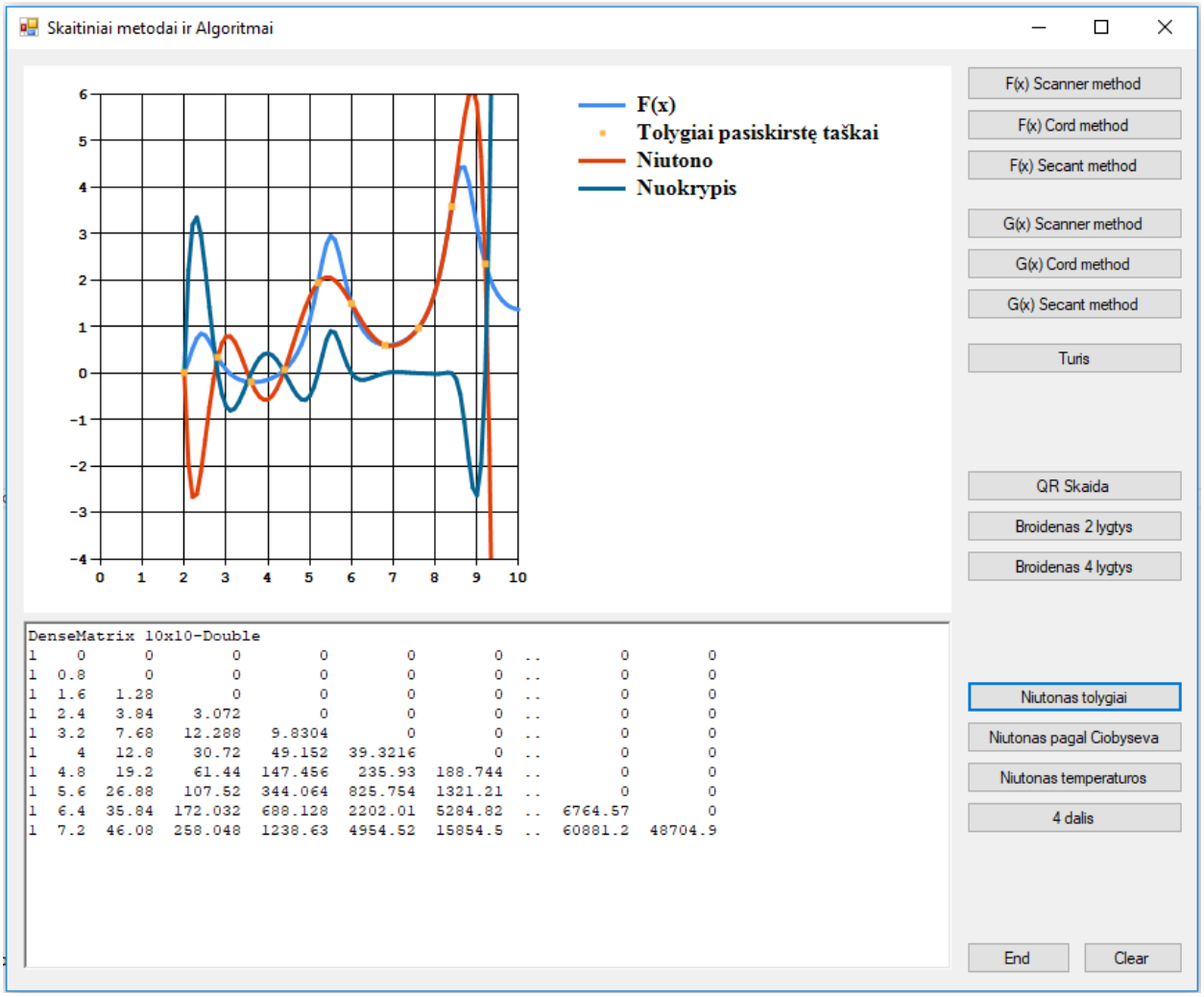
Užduočių sprendimai:

Spręstas variantas: 30 (užduočių funkcijos bei metodai kuriais jas spręsti matomos Pav. #2)

Niutono metodo realizacija, daryta pagal Moodle sistemoje įkeltame pdf'e esantį Matlab kodą:

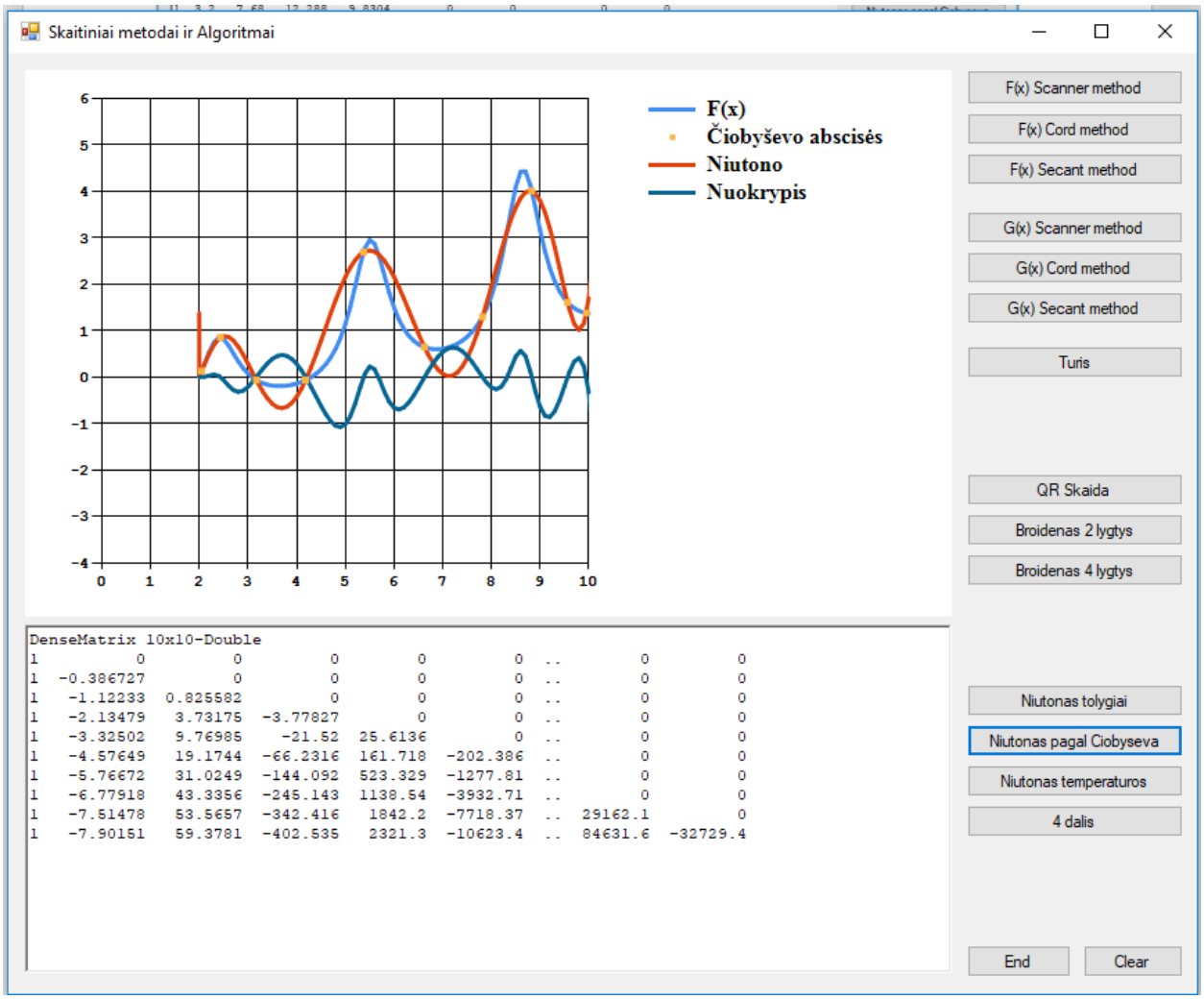
```
private double Newton(double[] X, double[] a, int n, double x)
{
    double rez = 0;
    for (int i = 0; i < n; i++)
    {
        double temp = 1;
        for (int j = 0; j < i; j++)
        {
            temp = temp * (x - X[j]);
        }
        rez += a[i] * temp;
    }
    return rez;
}

private double[] GetA(double[] X, double[] Y, int N)
{
    double[,] m = new double[N, N];
    double[] a = new double[N];
    for (int i = 0; i < N; i++)
    {
        m[i, 0] = 1;
    }
    for (int i = 1; i < N; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            m[i, j] = m[i, j - 1] * (X[i] - X[j - 1]);
        }
    }
    a[0] = Y[0];
    for (int i = 1; i < N; i++)
    {
        double temp = 0;
        for (int j = 0; j <= i; j++)
        {
            temp += m[i, j];
        }
        a[i] = Y[i] / temp;
    }
    Matrix<double> M = DenseMatrix.Build.Dense(N, N);
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            M[i, j] = m[i, j];
        }
    }
    richTextBox1.AppendText("" + M);
    Matrix<double> YY = DenseMatrix.Build.Dense(N, 1);
    for (int i = 0; i < N; i++) { YY[i, 0] = Y[i]; }
```

```
            Vector<double> YYY = YY.Column(0);
            Vector<double> A = (M.Inverse() * YYY);
            for (int i = 0; i < N; i++) {
                a[i] = A[i];
            }
            return a;
        }

  double[] chyobysev = new double[N];
  double[] Y = new double[N];
  for (int i = 0; i < N; i++)
  {
      chyobysev[i] = ((double)(xmax - xmin) / 2) * Math.Cos(Math.PI * ((double)(2 * i +
1)) / ((double)(2 * N))) + ((double)(xmax + xmin) / 2);
      Y[i] = Func(chyobysev[i]);
      task.Points.AddXY(chyobysev[i], Func(chyobysev[i]));
  }
```

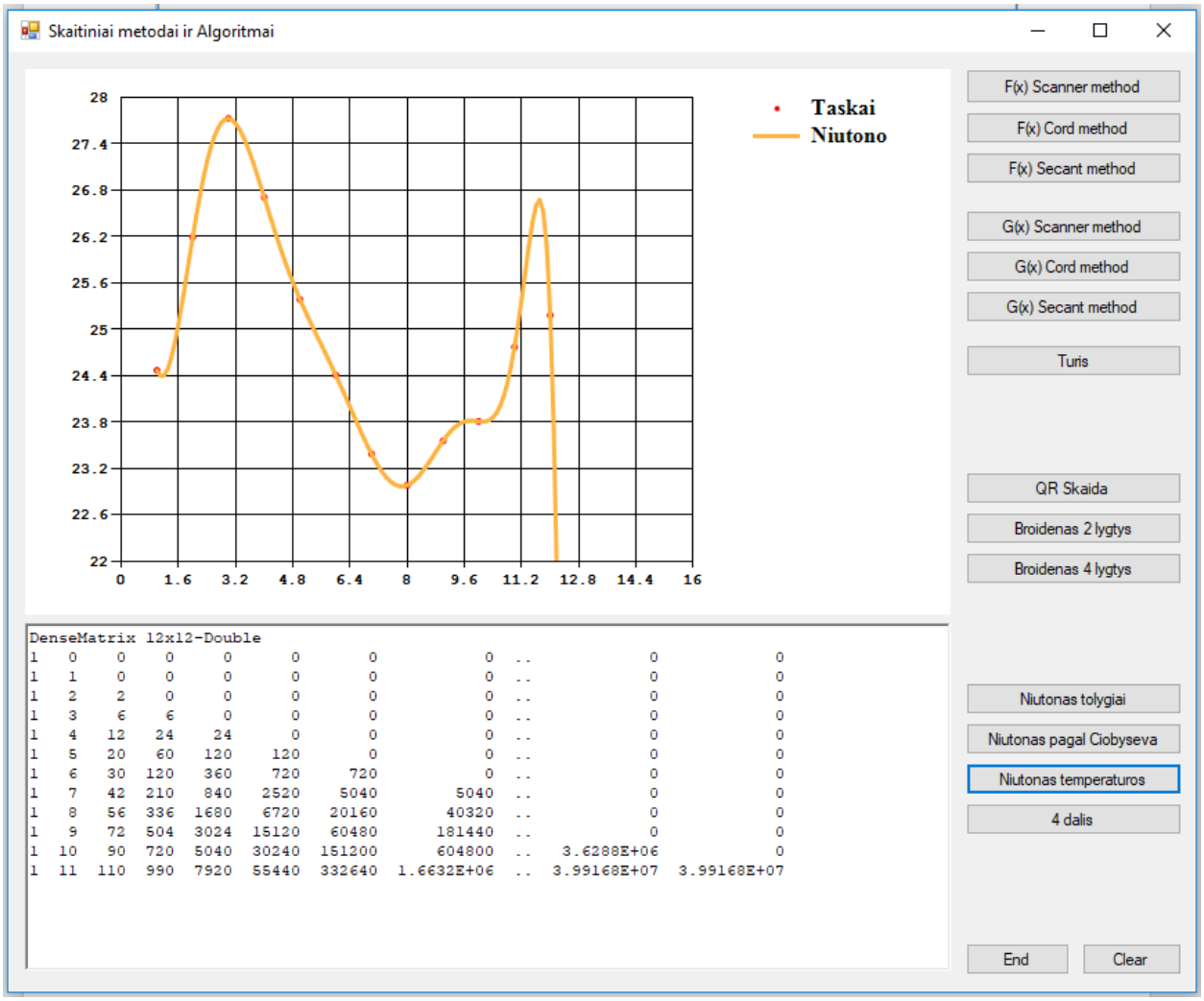



```
DenseMatrix 10x10-Double
1    0      0        0        0        0        0  ..        0        0
1  0.8      0        0        0        0        0  ..        0        0
1  1.6   1.28        0        0        0        0  ..        0        0
1  2.4   3.84    3.072        0        0        0  ..        0        0
1  3.2   7.68   12.288   9.8304        0        0  ..        0        0
1    4   12.8    30.72   49.152  39.3216        0  ..        0        0
1  4.8   19.2    61.44  147.456   235.93  188.744  ..        0        0
1  5.6  26.88   107.52  344.064  825.754  1321.21  ..        0        0
1  6.4  35.84  172.032  688.128  2202.01  5284.82  ..  6764.57        0
1  7.2  46.08  258.048  1238.63  4954.52  15854.5  ..  60881.2  48704.9
```

Pav. #3 1 Užduoties A dalis – Interpoliacinės funkcijos išraiška naudojant Niutono bazinį metodą bei tolygiai pasiskirsčiusius taškus



Pav. #4 1 Užduoties B dalis – Interpoliacinės funkcijos išraiška naudojant Niutono bazinį metodą bei taškus sudėtus pagal Čiobyševo abscises.

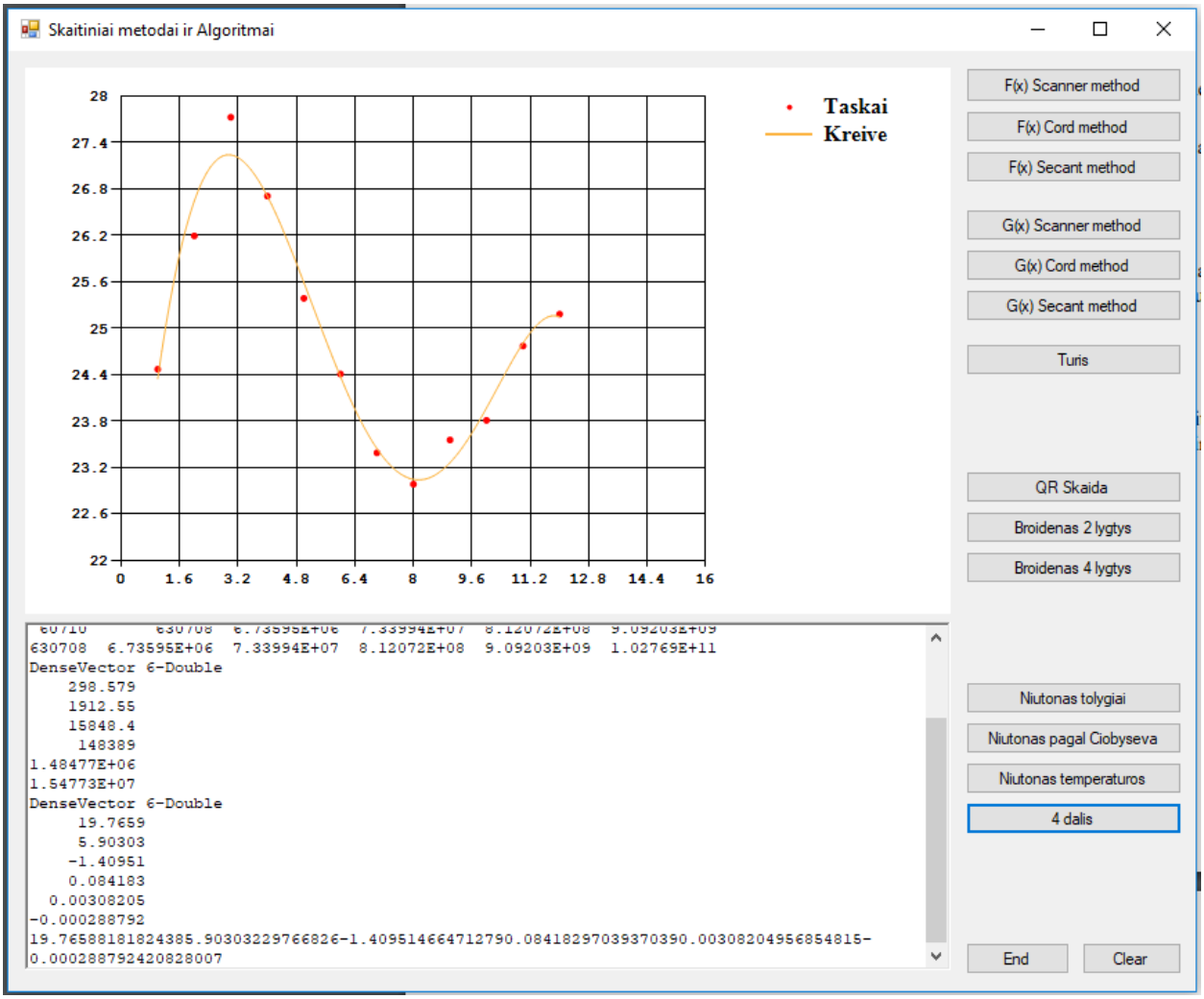


Pav. #5 2 Užduoties B dalis – Sudaryta interpoliuojanti temperatūrų kreivė naudojantis pirmos dalies daugianariu sudarytu pagal Niutono bazinį metodą.

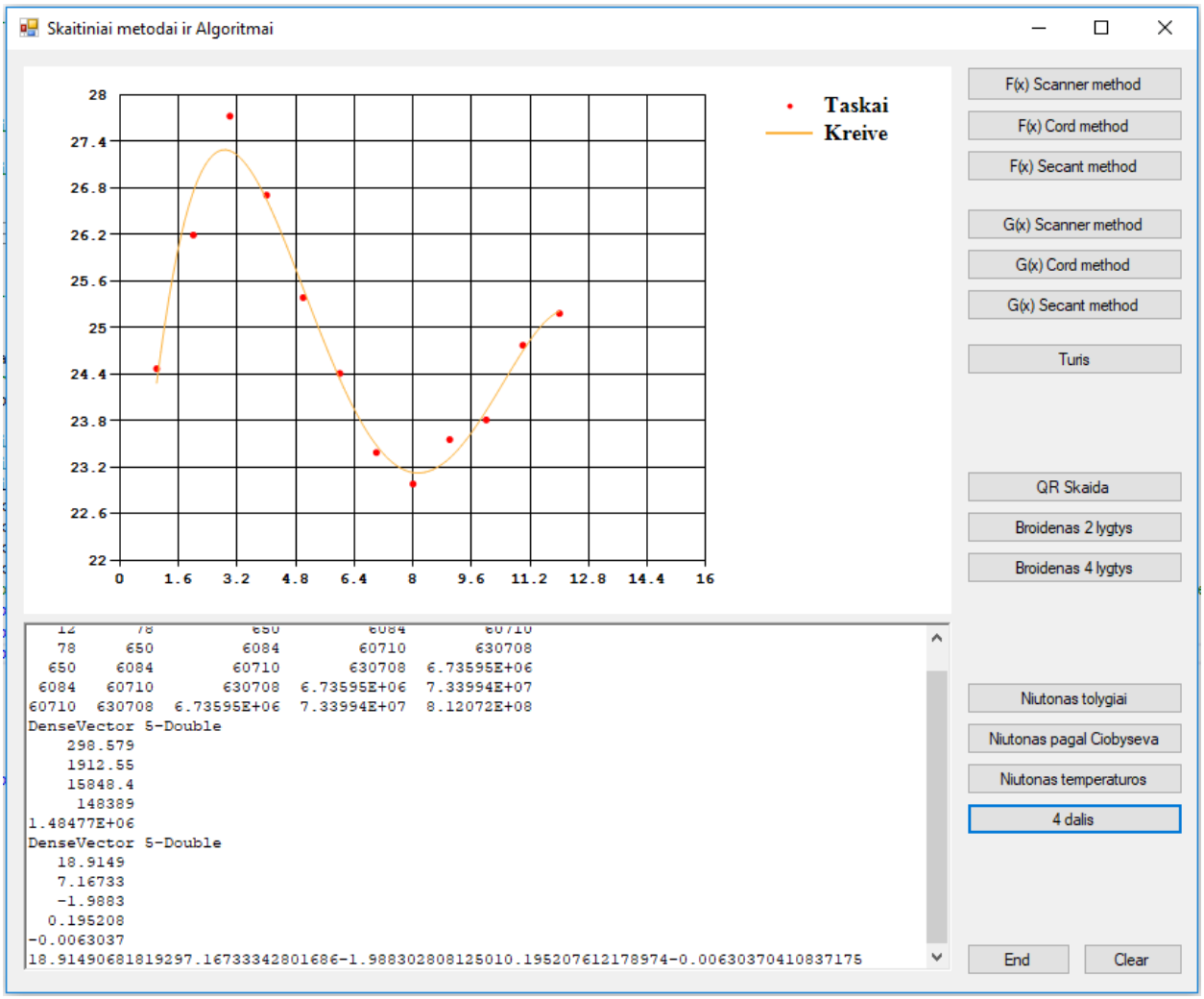
4-oje dalyje naudojamo mažiausių kvadratų metodo realizacija, padaryta pagal Moodle esančio pdf‘o Matlab kodą:

```
double[] mkm(int n, double[] x, double[] y)
{
    int m = x.Length;
    double[,] pa = new double[m, 2 * n + 1];
    double[,] pxy = new double[m, n + 1];
    double[] xk = new double[m];
    for (int i = 0; i < m; i++)
    {
        xk[i] = 1;
    }
    for (int j = 0; j < 2 * n + 1; j++)
    {
        for (int i = 0; i < m; i++)
        {
            pa[i, j] = xk[i];
        }
        if (j < n + 1)
        {
            for (int i = 0; i < m; i++)
            {
                pxy[i, j] = xk[i] * y[i];
            }
        }
        for (int i = 0; i < m; i++)
        {
            xk[i] = xk[i] * x[i];
        }
    }
    double[] s = new double[2 * n + 1];
    for (int i = 0; i < 2 * n + 1; i++)
    {
        double temp = 0;
        for (int j = 0; j < m; j++)
        {
            temp += pa[j, i];
        }
        s[i] = temp;
    }
    double[] b = new double[n + 1];
    for (int i = 0; i < n + 1; i++)
    {
        double temp = 0;
        for (int j = 0; j < m; j++)
        {
            temp += pxy[j, i];
        }
        b[i] = temp;
    }
    double[,] c = new double[n + 1, n + 1];
    for (int i = 0; i < n + 1; i++)
    {
        for (int j = 0; j < n + 1; j++)
        {
            c[i, j] = s[i + j];
```

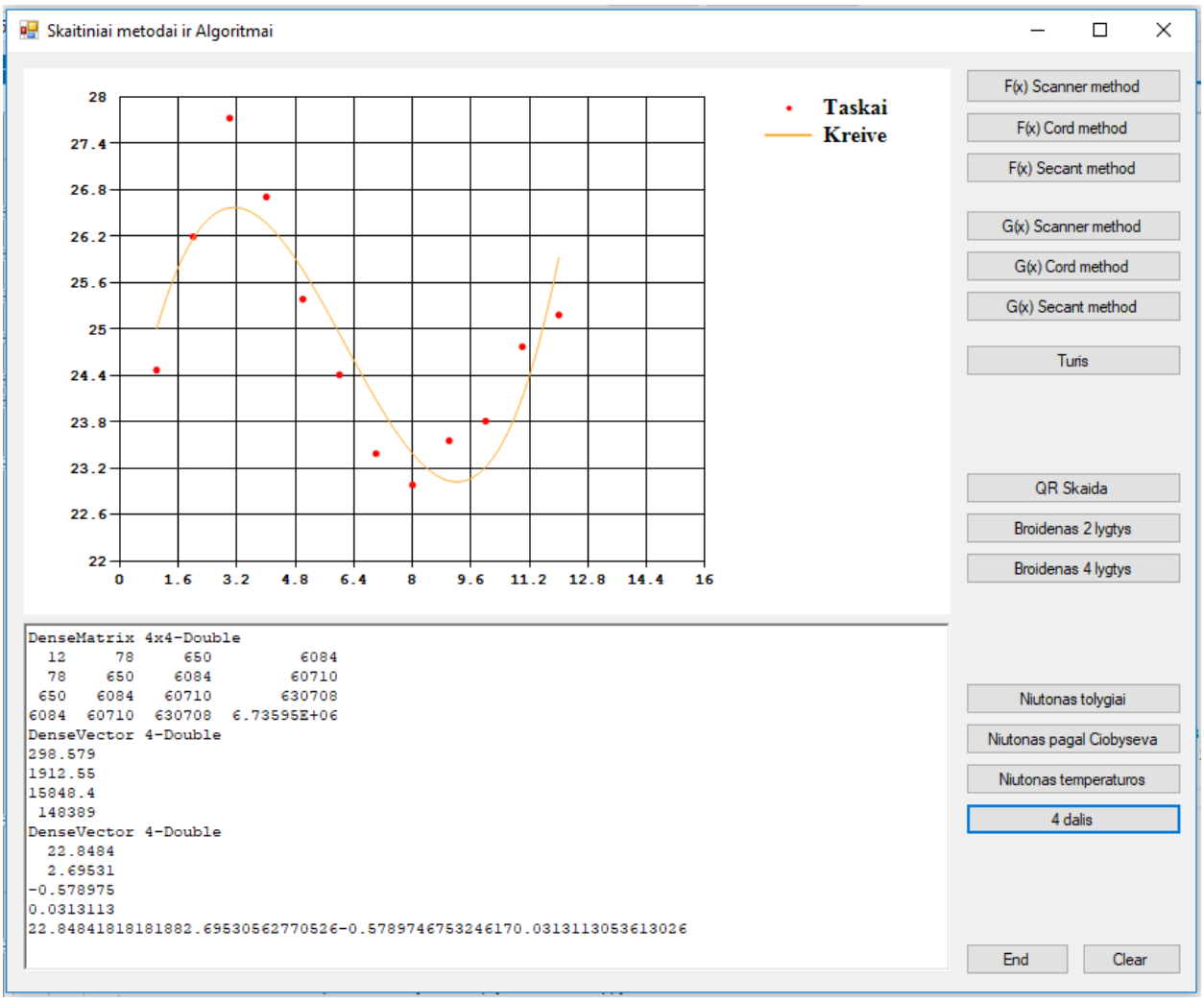
```
            }
        }
        Matrix<double> cm = DenseMatrix.OfArray(c);
        Vector<double> bm = Vector.Build.DenseOfArray(b);
        Vector<double> a = cm.Solve(bm);
        richTextBox1.AppendText(cm.ToString());
        richTextBox1.AppendText(bm.ToString());
        richTextBox1.AppendText(a.ToString());
        double[] array = new double[n + 1];
        for (int i = 0; i < n + 1; i++)
        {
            array[i] = a[i];
        }
        return array;
    }

    double func2(double[] array, double x)
    {
        double sum = 0;
        for (int i = 0; i < array.Length; i++)
        {
            sum += array[i] * Math.Pow(x, i);
        }
        return sum;
    }
```



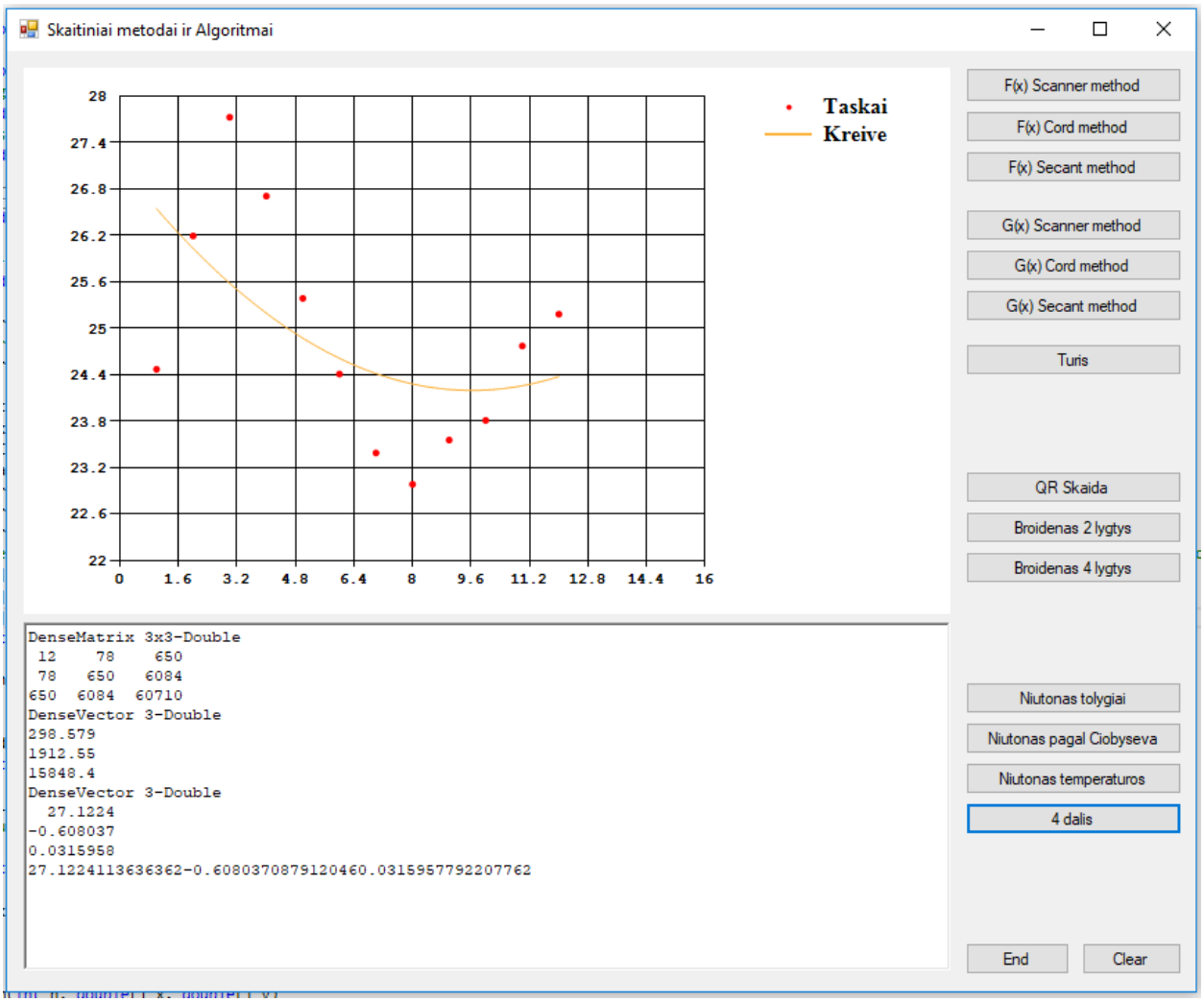
Pav. #6 4 Užduotis – Sudaryta aproksimuojanti kreivė 12 mėnesių temperatūroms atvaizduoti naudojant **penktos** eilės daugianarį. Išraiškų koeficientai matomi RichTextbox1 elemente.



Pav. #7 4 Užduotis – Sudaryta aproksimuojanti kreivė 12 mėnesių temperatūroms atvaizduoti naudojant **ketvirtos** eilės daugianarį. Išraiškų koeficientai matomi RichTextbox1 elemente.



Pav. #8 4 Užduotis – Sudaryta aproksimuojanti kreivė 12 mėnesių temperatūroms atvaizduoti naudojant **trečios** eilės daugianarį. Išraiškų koeficientai matomi RichTextbox1 elemente.



Pav. #8 4 Užduotis – Sudaryta aproksimuojanti kreivė 12 mėnesių temperatūroms atvaizduoti naudojant **antros** eilės daugianarį. Išraiškų koeficientai matomi RichTextbox1 elemente.

Programos kodo fragmentai:

```
private double Func(double x)
{
   return (Math.Log(x) / (Math.Sin(2 * x) + 1.5)) - Math.Cos(x/5);
}
// ----------------------- 1 Dalis Niutonas -------------------------------

private double Newton(double[] X, double[] a, int n, double x)
{
   double rez = 0;
   for (int i = 0; i < n; i++)
   {
      double temp = 1;
      for (int j = 0; j < i; j++)
      {
         temp = temp * (x - X[j]);
      }
      rez += a[i] * temp;
   }
   return rez;
}

private double[] GetA(double[] X, double[] Y, int N)
{
   double[,] m = new double[N, N];
   double[] a = new double[N];
```

```
for (int i = 0; i < N; i++)
{
   m[i, 0] = 1;
}
for (int i = 1; i < N; i++)
{
   for (int j = 1; j <= i; j++)
   {
      m[i, j] = m[i, j - 1] * (X[i] - X[j - 1]);
   }
}
a[0] = Y[0];
for (int i = 1; i < N; i++)
{
   double temp = 0;
   for (int j = 0; j <= i; j++)
   {
      temp += m[i, j];
   }
   a[i] = Y[i] / temp;
}
Matrix<double> M = DenseMatrix.Build.Dense(N, N);
for (int i = 0; i < N; i++)
{
   for (int j = 0; j < N; j++)
   {
      M[i, j] = m[i, j];
```

```
        }
    }
    richTextBox1.AppendText("" + M);
    Matrix<double> YY = DenseMatrix.Build.Dense(N, 1);
    for (int i = 0; i < N; i++) { YY[i, 0] = Y[i]; }
    Vector<double> YYY = YY.Column(0);
    Vector<double> A = (M.Inverse() * YYY);
    for (int i = 0; i < N; i++) {
        a[i] = A[i];
    }
    return a;
}
//Taskai pagal Ciobysevo abscises
private void Button13_click(object sender, EventArgs e)
{
    ClearForm(); // išvalomi programos duomenys
    PrepareareForm(0, 10, -4, 6);
    // Nubraižoma f-ja, kuriai ieskome saknies
    Fx = chart1.Series.Add("F(x)");
    Fx.ChartType = SeriesChartType.Line;

    int N = 10;
    double xmax = 10, xmin = 2;
    double temp = 2;
    for (int i = 0; i < N * 10; i++)
    {
        Fx.Points.AddXY(temp, Func(temp));
```

```
            temp = temp + 0.1;
        }
        Fx.BorderWidth = 3;

        Series task = chart1.Series.Add("Čiobyševo abscisės");
        task.ChartType = SeriesChartType.Point;

        double[] chyobysev = new double[N];
        double[] Y = new double[N];
        for (int i = 0; i < N; i++)
        {
            chyobysev[i] = ((double)(xmax - xmin) / 2) * Math.Cos(Math.PI * ((double)(2 * i + 1)) /
((double)(2 * N))) + ((double)(xmax + xmin) / 2);
            Y[i] = Func(chyobysev[i]);
            task.Points.AddXY(chyobysev[i], Func(chyobysev[i]));
        }

        Series newton = chart1.Series.Add("Niutono");
        newton.ChartType = SeriesChartType.Line;
        double tempo = 2;
        var a = GetA(chyobysev, Y, N);

        newton.Points.AddXY(2, Y[0]);
        for (int i = 1; i < N * 100; i++)
        {
            newton.Points.AddXY(tempo, Newton(chyobysev, a, N, tempo));
            tempo += 0.1;
```

```
    }
    newton.BorderWidth = 3;

    Series nuokrypis = chart1.Series.Add("Nuokrypis");
    nuokrypis.ChartType = SeriesChartType.Line;

    tempo = 2;
    for (int i = 1; i < N * 100; i++)
    {
        nuokrypis.Points.AddXY(tempo, Func(tempo) - Newton(chyobysev, a, N, tempo));
        tempo += 0.1;
    }
    nuokrypis.BorderWidth = 3;
}
//Taskai pasiskirste tolygiai
private void Button14_click(object sender, EventArgs e)
{
    ClearForm(); // išvalomi programos duomenys
    PreparareForm(0, 10, -4, 6);

    // Nubraižoma f-ja, kuriai ieskome saknies
    Fx = chart1.Series.Add("F(x)");

    Fx.ChartType = SeriesChartType.Line;
    double x = 0;

    int N = 10;
```

```
double xmax = 10, xmin = 2;
double temp = 2;
for (int i = 0; i < N * 10; i++)
{
   Fx.Points.AddXY(temp, Func(temp));
   temp = temp + 0.1;
}
Fx.BorderWidth = 3;

Series task = chart1.Series.Add("Tolygiai pasiskirstę taškai");
task.ChartType = SeriesChartType.Point;

double[] Y = new double[N];
double[] tolygus = new double[N];
double zingsnis = (xmax - xmin) / N;
double first = xmin;
for (int i = 0; i < N; i++)
{
   tolygus[i] = first + zingsnis * i;
   Y[i] = Func(tolygus[i]);
   task.Points.AddXY(tolygus[i], Func(tolygus[i]));
}

Series newton = chart1.Series.Add("Niutono");
newton.ChartType = SeriesChartType.Line;
double tempo = 2;
```

```
    var a = GetA(tolygus, Y, N);

    newton.Points.AddXY(2, Y[0]);
    for (int i = 1; i < N * 100; i++)
    {
        newton.Points.AddXY(tempo, Newton(tolygus, a, N, tempo));
        tempo += 0.1;
    }
    newton.BorderWidth = 3;

    Series nuokrypis = chart1.Series.Add("Nuokrypis");
    nuokrypis.ChartType = SeriesChartType.Line;

    tempo = 2;
    for (int i = 1; i < N * 100; i++)
    {
        nuokrypis.Points.AddXY(tempo, Func(tempo) - Newton(tolygus, a, N, tempo));
        tempo += 0.1;
    }
    nuokrypis.BorderWidth = 3;
}

// ----------------------------- 2 Dalis ---------------------------------------------

//Daugianaris pagal pirmos lenteles funkcija (Niutonas)
private void button15_Click(object sender, EventArgs e)
{
```

```
        ClearForm(); // išvalomi programos duomenys
        PrepareareForm(0, 16, 22, 28);

        Series task = chart1.Series.Add("Taskai");
        task.ChartType = SeriesChartType.Point;
        task.MarkerSize = 5;
        task.MarkerStyle = MarkerStyle.Circle;
        task.MarkerColor = Color.Red;

        double[] temperature = { 24.4692, 26.1946, 27.7291, 26.7083, 25.3865, 24.4072,
23.3871, 22.9818, 23.555, 23.8082, 24.7706, 25.1817};
        double[] month = new double[12] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
        int N = month.Length;
        for (int i = 0; i < month.Length; i++)
        {
           task.Points.AddXY(month[i], temperature[i]);
        }
        Series newton = chart1.Series.Add("Niutono");
        newton.ChartType = SeriesChartType.Line;
        double tempo = 1;
        var a = GetA(month, temperature, N);

        // newton.Points.AddXY(month[0], temperature[0]);
        for (int i = 1; i < N * 100; i++)
        {
           newton.Points.AddXY(tempo, Newton(month, a, N, tempo));
           newton.Points.AddXY(tempo, Newton(month, a, N, tempo));
```

```
            newton.Points.AddXY(tempo, Newton(month, a, N, tempo));
            tempo += 0.1;
        }
        newton.BorderWidth = 3;
    }

    // ---------------------------- 4 Dalis ------------------------------------------
    private void button16_Click(object sender, EventArgs e)
    {
        ClearForm(); // išvalomi programos duomenys
        //PrepareareForm(0, 16, 18, 35);//deivio
        PrepareareForm(0, 16, 22, 28); //mano

        Series task = chart1.Series.Add("Taskai");
        Series kreive = chart1.Series.Add("Kreive");
        kreive.ChartType = SeriesChartType.Line;
        task.ChartType = SeriesChartType.Point;
        task.MarkerSize = 5;
        task.MarkerStyle = MarkerStyle.Circle;
        task.MarkerColor = Color.Red;
        //double[] temperature = { 19.8794, 23.7751, 27.883, 31.0976, 33.2133, 33.8218,
31.8081, 29.8617, 29.1279, 28.8316, 25.9363, 21.4918 }; //deivio
        double[] temperature = { 24.4692, 26.1946, 27.7291, 26.7083, 25.3865, 24.4072,
23.3871, 22.9818, 23.555, 23.8082, 24.7706, 25.1817 };//mano
        double[] month = new double[12] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
        double[] a = mkm(2, month, temperature); //pirmas parametras - polinomo eile
        for (int i = 0; i < a.Length; i++)
        {
```

```
        richTextBox1.AppendText(a[i].ToString());
    }
    int N = 100;
    double dx = (12.0 - 1) / (N - 1);
    for (int i = 0; i < N; i++)
    {
        kreive.Points.AddXY(1 + i * dx, func2(a, 1 + i * dx));
        //func2(a, i * dx);
    }
    for (int i = 0; i < month.Length; i++)
    {
        task.Points.AddXY(month[i], temperature[i]);
    }
}

double[] mkm(int n, double[] x, double[] y)
{
    int m = x.Length;
    double[,] pa = new double[m, 2 * n + 1];
    double[,] pxy = new double[m, n + 1];
    double[] xk = new double[m];
    for (int i = 0; i < m; i++)
    {
        xk[i] = 1;
    }
    for (int j = 0; j < 2 * n + 1; j++)
    {
```

```
    for (int i = 0; i < m; i++)
    {
        pa[i, j] = xk[i];
    }
    if (j < n + 1)
    {
        for (int i = 0; i < m; i++)
        {
            pxy[i, j] = xk[i] * y[i];
        }
    }
    for (int i = 0; i < m; i++)
    {
        xk[i] = xk[i] * x[i];
    }
}
double[] s = new double[2 * n + 1];
for (int i = 0; i < 2 * n + 1; i++)
{
    double temp = 0;
    for (int j = 0; j < m; j++)
    {
        temp += pa[j, i];
    }
    s[i] = temp;
}
double[] b = new double[n + 1];
```

```
for (int i = 0; i < n + 1; i++)
{
   double temp = 0;
   for (int j = 0; j < m; j++)
   {
      temp += pxy[j, i];
   }
   b[i] = temp;
}
double[,] c = new double[n + 1, n + 1];
for (int i = 0; i < n + 1; i++)
{
   for (int j = 0; j < n + 1; j++)
   {
      c[i, j] = s[i + j];
   }
}
Matrix<double> cm = DenseMatrix.OfArray(c);
Vector<double> bm = Vector.Build.DenseOfArray(b);
Vector<double> a = cm.Solve(bm);
richTextBox1.AppendText(cm.ToString());
richTextBox1.AppendText(bm.ToString());
richTextBox1.AppendText(a.ToString());
double[] array = new double[n + 1];
for (int i = 0; i < n + 1; i++)
{
   array[i] = a[i];
```

```
    }
    return array;
}

double func2(double[] array, double x)
{
    double sum = 0;
    for (int i = 0; i < array.Length; i++)
    {
        sum += array[i] * Math.Pow(x, i);
    }
    return sum;
}
#endregion
```