III projektinė užduotis

1. Interpoliavimas daugianariu

1.1. Užduotis

Duota interpoliuojamos funkcijos analitinė išraiška. Pateikite interpoliacinės funkcijos išraišką naudodami nurodytą bazinę funkciją, kai:

- Taškai pasiskirstę tolygiai.
- 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.

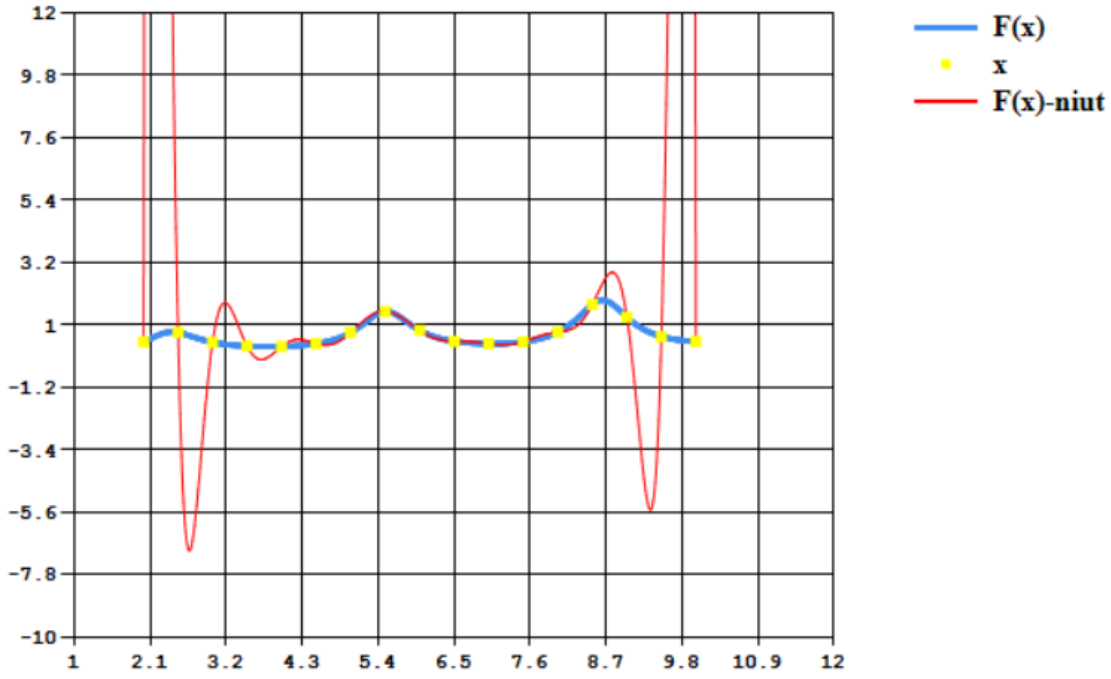
Funkcijos analitinė išraiška:

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

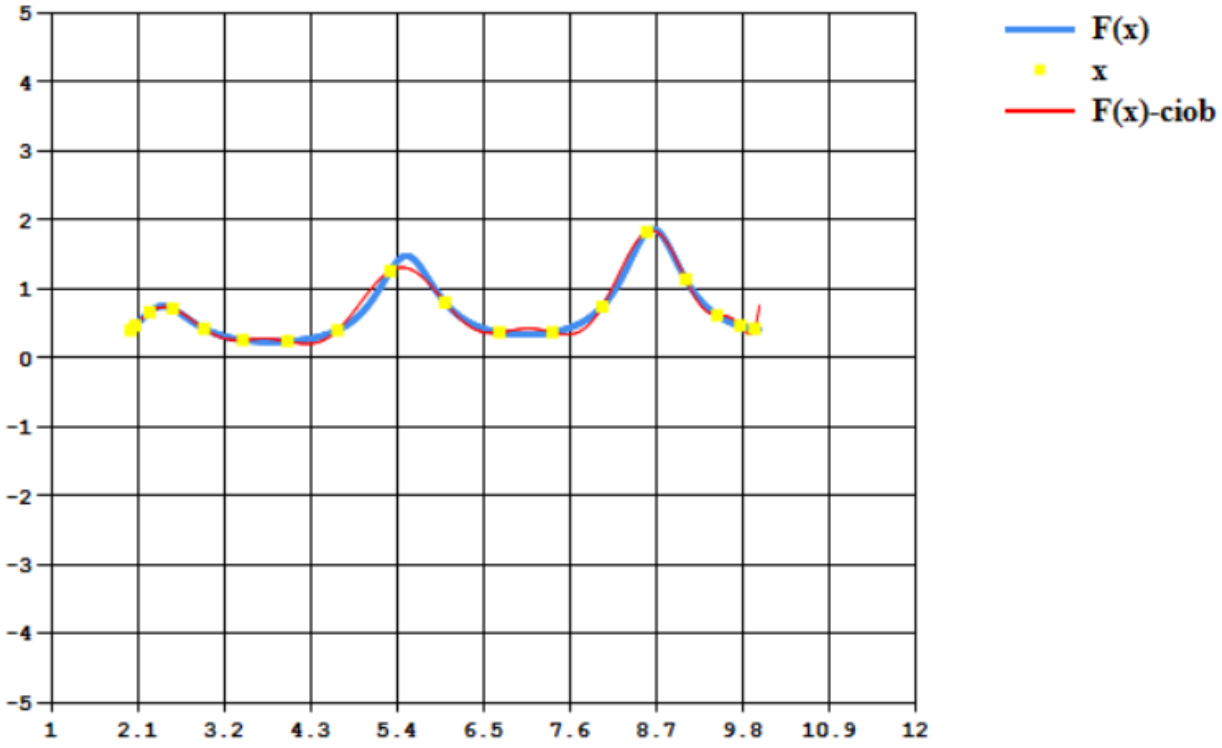
Bazinė funkcija: **Niutono.**

1.2. Rezultatai

Rezultatai kai taškai pasiskirstę tolygiai:



Rezultatai kai taškai apskaičiuojami naudojant Čiobyševo abscises:



Palyginus šiuos metodus matoma, kad iterpoliavimas naudojant Čiobyševo abscises yra tikslesnis, nei iterpoliavimas, kai taškai yra pasiskirstę tolygiai.

2. Interpoliavimas daugianariu ir splainu per duotus taškus

2.1. Užduotis

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

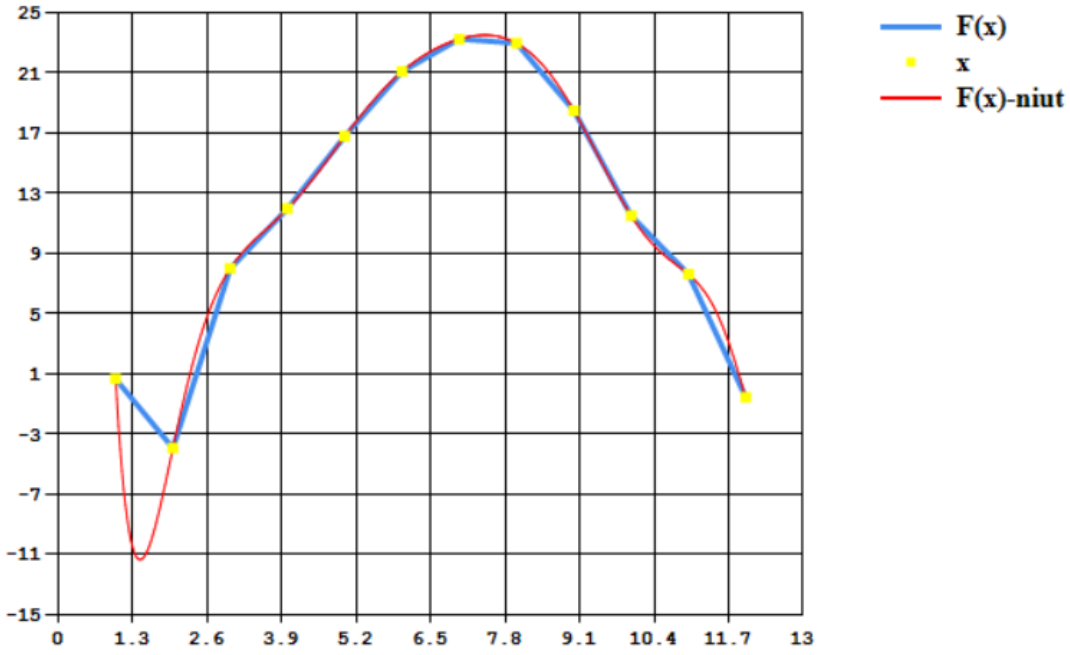
- Daugianariu, sudarytu naudojant nurodytą bazinę funkciją.
- Nurodyto tipo splainu.

Pateikta:

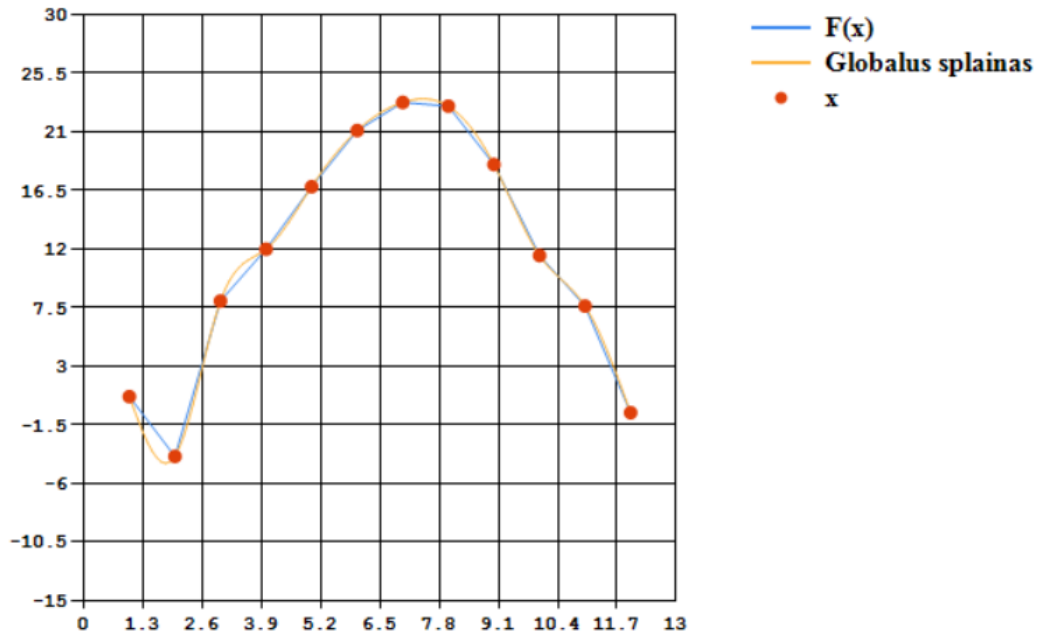
- Šalis – Vengrija
- Metai – 2012
- Bazinė funkcija – Niutono
- Globalus splainas

2.2. Rezultatai

Kreivė sudaryta daugianariu, naudojant Niutono bazinę funkciją:



Kreivė sudaryta globaliu splainu:



Palyginus šias kreives matoma, kad globalus splainas neturi tokių aštrių minimumų ir maximumų, tačiau atliekant interpoliavimą pagal daugianarį rezultatai atrodo tikslesni dauguma atvejų.

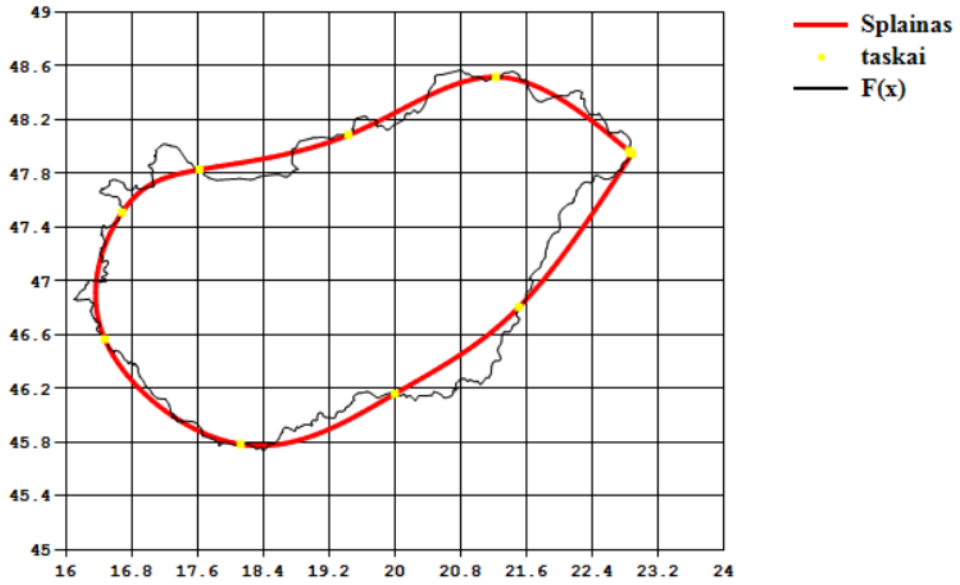
3. Parametrinis interpoliavimas

3.1. Užduotis

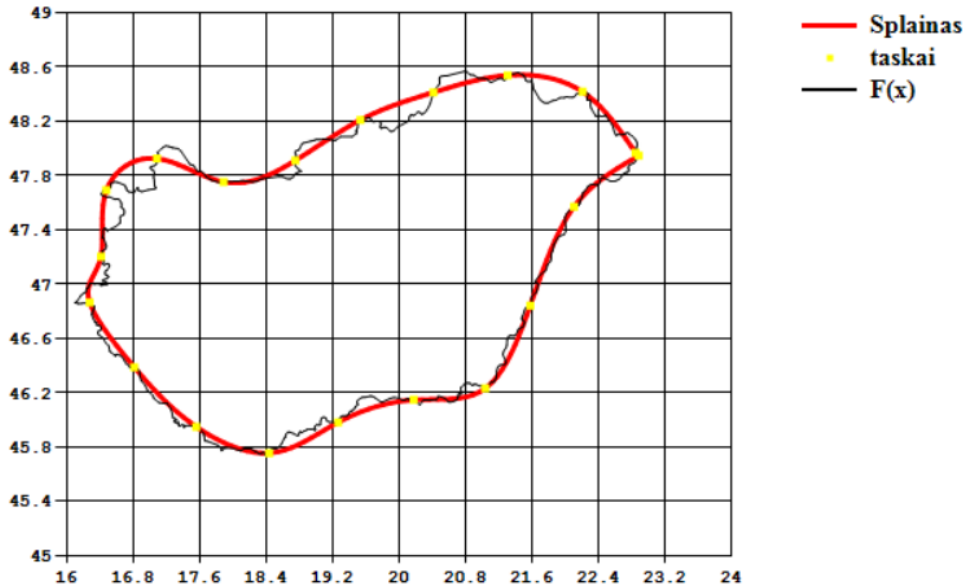
Naudodami **parametrinio** interpoliavimo metodą nurodytu splainu suformuokite nurodytos šalies kontūrą. Pateikite pradinius duomenis ir rezultatus, gautus naudojant 10, 20, 50, 100 interpoliavimo taškų.

3.2. Rezultatai

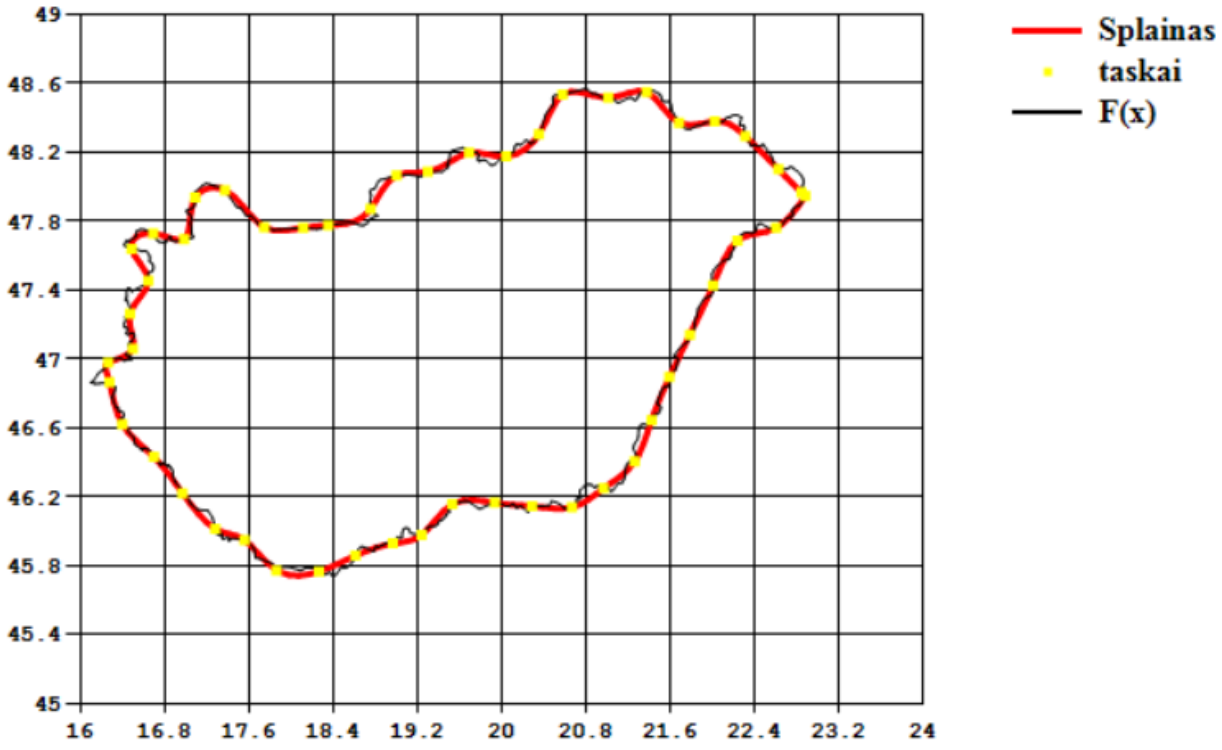
Gautas rezultatas naudojant 10 taškų:



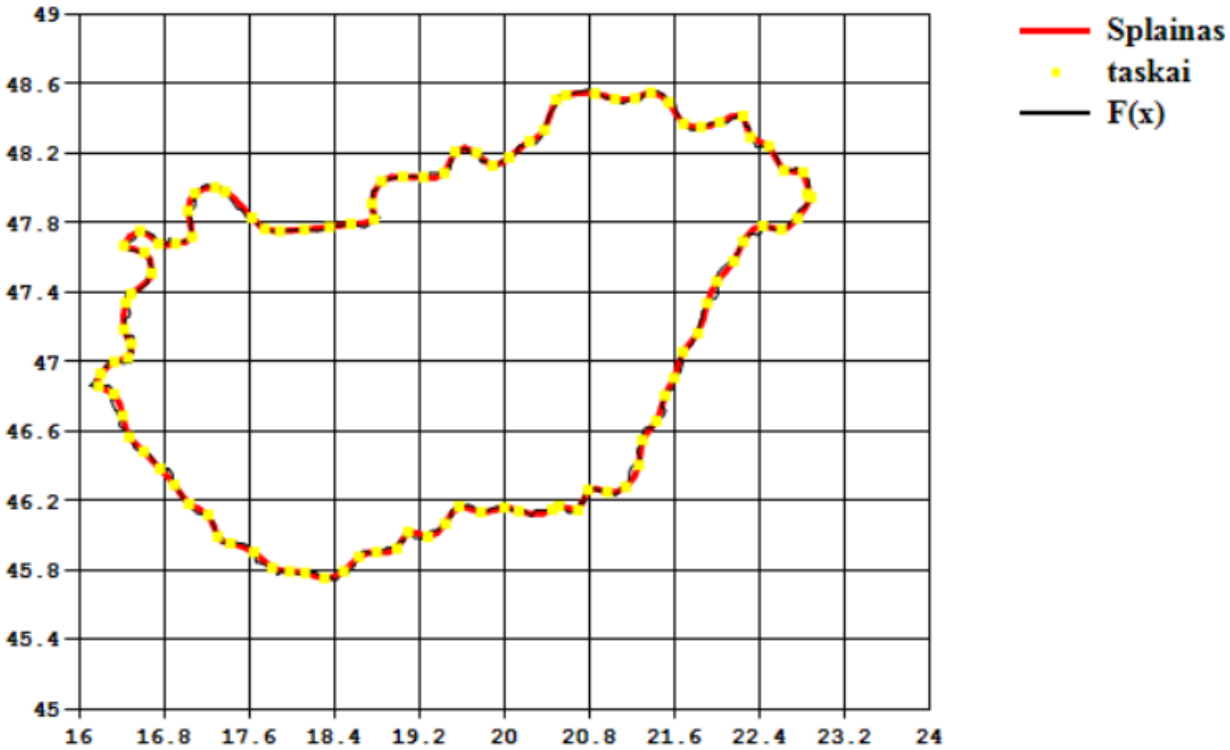
Gautas rezultatas naudojant 20 taškų:



Gautas rezultatas naudojant 50 taškų:



Gautas rezultatas naudojant 100 taškų:

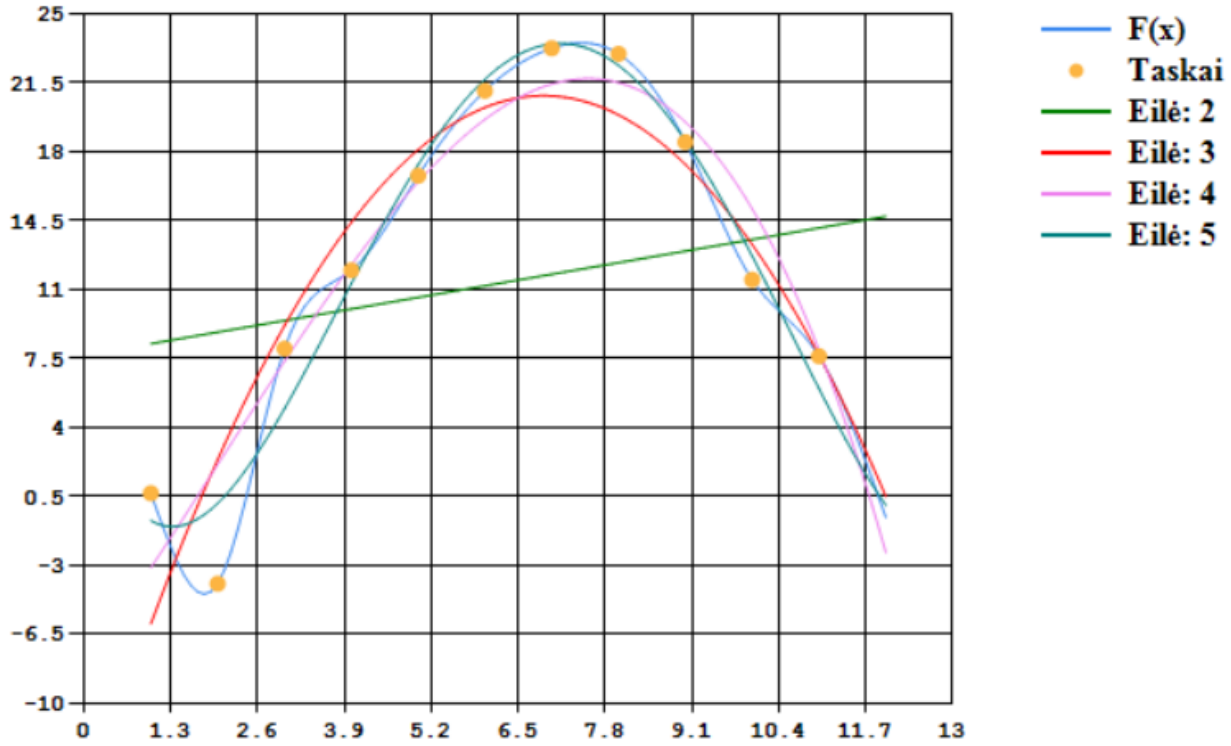


4. Aproksimavimas

4.1. Užduotis

Pagal 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.

4.2. Rezultatai



5. Programos kodas

```
public class Lab3
  {
    private readonly Form1 form;


    public Lab3(Form1 form)
    {
      this.form = form;
    }
```

```
double F(double x) => Math.Log10(x) / (Math.Sin(2 * x) + 1.5);

public void Uzd11()
{
   var arrayOfX = new List<double>();

   form.ClearForm();
   form.PrepareareForm(1,12,-10,12);
   var Fx = form.chart1.Series.Add("F(x)");
   Fx.ChartType = SeriesChartType.Line;
   var x = 2d;
   while (x <= 10)
   {
      Fx.Points.AddXY(x, F(x));
      x = x + 0.01;
   }
   Fx.BorderWidth = 3;

   var X1X2 = form.chart1.Series.Add("x");
   X1X2.MarkerSize = 6;
   X1X2.ChartType = SeriesChartType.Point;
   X1X2.Color = Color.Yellow;

   var intervals = new List<double>();
   for (var i = 2d; i <= 10; i = i + 0.5)
   {
      arrayOfX.Add(F(i));
```

```
    intervals.Add(i);
    X1X2.Points.AddXY(i, F(i));
}
form.richTextBox1.AppendText(intervals.Count.ToString());

var busima = new double[intervals.Count, intervals.Count];
for (int i = 0; i < intervals.Count; i++)
{
    busima[i, 0] = 1;
    for (int j = 1; j < intervals.Count; j++)
    {
        var temp = 1d;

        for (var k = 0; k < j; k++)
        {
            temp *= intervals[i] - intervals[k];
        }

        busima[i, j] = temp;
    }
}

var m = Matrix<double>.Build.DenseOfArray(busima);
var y = Vector<double>.Build.DenseOfArray(arrayOfX.ToArray());
var a = m.Inverse() * y;

// Niutonas
```

```
var Fx_niut = form.chart1.Series.Add("F(x)-niut");
Fx_niut.ChartType = SeriesChartType.Line;
Fx_niut.Color = Color.Red;
Fx_niut.MarkerSize = 7;

x = 2;
while (x <= 10)
{
   var temp = 0d;
   for (int j = 0; j < intervals.Count; j++)
   {
      var temp1 = 1d;

      for (int i = 0; i < j; i++)
      {
         temp1 = temp1 * (x - intervals[i]);
      }
      temp = temp + a[j] * temp1;
   }

   Fx_niut.Points.AddXY(x, temp);
   x = x + 0.01;
}

form.richTextBox1.AppendText("\nMatrica m:\n");
form.richTextBox1.AppendText(m.ToString());
```

```
    form.richTextBox1.AppendText("\nVektorius y:\n");
    form.richTextBox1.AppendText(y.ToString());

    form.richTextBox1.AppendText("\nVektorius a:\n");
    form.richTextBox1.AppendText(a.ToString());
}

public void Uzd12()
{
    var arrayOfX = new List<double>();

    form.ClearForm();
    form.PrepareareForm(1, 12, -5, 5);
    var Fx = form.chart1.Series.Add("F(x)");
    Fx.ChartType = SeriesChartType.Line;
    var x = 2d;
    while (x <= 10)
    {
        Fx.Points.AddXY(x, F(x));
        x = x + 0.01;
    }
    Fx.BorderWidth = 3;

    var X1X2 = form.chart1.Series.Add("x");
    X1X2.MarkerSize = 6;
    X1X2.ChartType = SeriesChartType.Point;
    X1X2.Color = Color.Yellow;
```

```
var aa = 2d;
var b = 10d;
var intervals = new List<double>();
for (var i = 0d; i < 9; i = i + 0.5)
{
   var temp = (b - aa) / 2.0 * Math.Cos(Math.PI * (2 * i + 1) / (2.0 * 9.0)) + (b + aa) / 2;

   intervals.Add(temp);
   arrayOfX.Add(F(temp));
   X1X2.Points.AddXY(temp, F(temp));
}

var busima = new double[intervals.Count, intervals.Count];
for (int i = 0; i < intervals.Count; i++)
{
   busima[i, 0] = 1;
   for (int j = 1; j < intervals.Count; j++)
   {
      var temp = 1d;

      for (var k = 0; k < j; k++)
      {
         temp *= intervals[i] - intervals[k];
      }

      busima[i, j] = temp;
```

```
    }
}

var m = Matrix<double>.Build.DenseOfArray(busima);
var y = Vector<double>.Build.DenseOfArray(arrayOfX.ToArray());
var a = m.Inverse() * y;

// Ciobysevo
var Fx_ciob = form.chart1.Series.Add("F(x)-ciob");
Fx_ciob.ChartType = SeriesChartType.Line;
Fx_ciob.Color = Color.Red;
Fx_ciob.MarkerSize = 7;

x = 2;
while (x <= 10)
{
    var temp = 0d;
    for (int j = 0; j < intervals.Count; j++)
    {
        var temp1 = 1d;

        for (int i = 0; i < j; i++)
        {
            temp1 *= x - intervals[i];
        }
        temp = temp + a[j] * temp1;
    }
```

```
            Fx_ciob.Points.AddXY(x, temp);
            x = x + 0.01;
        }

        form.richTextBox1.AppendText("\nMatrica m:\n");
        form.richTextBox1.AppendText(m.ToString());

        form.richTextBox1.AppendText("\nVektorius y:\n");
        form.richTextBox1.AppendText(y.ToString());

        form.richTextBox1.AppendText("\nVektorius a:\n");
        form.richTextBox1.AppendText(a.ToString());
    }

    readonly double[] temperature = { 0.64822, -3.9455, 7.99805, 11.983, 16.7858, 21.1055,
23.2656, 22.9801, 18.4952, 11.5009, 7.61598, -0.5816 };

    public void Uzd21()
    {
        form.ClearForm();
        form.PrepareareForm(0, 13, -15, 25);
        var Fx = form.chart1.Series.Add("F(x)");
        Fx.ChartType = SeriesChartType.Line;

        var x = 1d;
        var id = 0;
```

```
while (x <= 12)
{
   Fx.Points.AddXY(x, temperature[id]);
   x++;
   id++;
}

Fx.BorderWidth = 3;

var X1X2 = form.chart1.Series.Add("x");
X1X2.MarkerSize = 6;
X1X2.ChartType = SeriesChartType.Point;
X1X2.Color = Color.Yellow;

id = 0;

List<double> intervals = new List<double>();
for (int i = 1; i <= 12; i++)
{
   intervals.Add(i);
   X1X2.Points.AddXY(i, temperature[id]);
   id++;
}

double[,] busima = new double[intervals.Count, intervals.Count];

for (int i = 0; i < intervals.Count; i++)
```

```
{
   busima[i, 0] = 1;
   for (int j = 1; j < intervals.Count; j++)
   {
      var temp = 1d;

      for (var k = 0; k < j; k++)
      {
         temp *= intervals[i] - intervals[k];
      }

      busima[i, j] = temp;
   }
}

Matrix<double> m = Matrix<double>.Build.DenseOfArray(busima);
Vector<double> y = Vector<double>.Build.DenseOfArray(temperature.ToArray());
Vector<double> a = m.Inverse() * y;

var Fx_niut = form.chart1.Series.Add("F(x)-niut");
Fx_niut.ChartType = SeriesChartType.Line;
Fx_niut.Color = Color.Red;
Fx_niut.MarkerSize = 7;

x = 1;
while (x <= 12)
{
```

```
        var temp = 0d;
        for (int j = 0; j < intervals.Count; j++)
        {
            var temp1 = 1d;

            for (int i = 0; i < j; i++)
            {
                temp1 = temp1 * (x - intervals[i]);
            }
            temp = temp + a[j] * temp1;
        }

        Fx_niut.Points.AddXY(x, temp);
        x = x + 0.01;
    }

    form.richTextBox1.AppendText("\nMatrica m:\n");
    form.richTextBox1.AppendText(m.ToString());

    form.richTextBox1.AppendText("\nVektorius y:\n");
    form.richTextBox1.AppendText(y.ToString());

    form.richTextBox1.AppendText("\nVektorius a:\n");
    form.richTextBox1.AppendText(a.ToString());
}

public void Uzd22()
```

```
{
    form.ClearForm();
    form.PreparareForm(-0, 13, -15, 30);
    var Fx = form.chart1.Series.Add("F(x)");
    Fx.ChartType = SeriesChartType.Line;

    var x1 = 1d;
    var id = 0;
    while (x1 <= 12)
    {
        Fx.Points.AddXY(x1, temperature[id]);
        x1++;
        id++;
    }

    var GS = form.chart1.Series.Add("Globalus splainas");
    GS.ChartType = SeriesChartType.Line;
    GS.Points.Clear();
    var x = new double[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };

    var dF = Isvestines(x, temperature);
    var ve = new double[12];
    ve[0] = 0;
    ve[11] = 0;
    for (var i = 0; i < 12 - 2; i++)
    {
        ve[i + 1] = dF[i];
```

```
}

dF = Vector<double>.Build.DenseOfArray(ve);
for (var i = 0; i < 11; i++)
{
   for (var j = x[i]; j <= x[i + 1]; j = j + 0.1)
   {
      var s = j - x[i];
      var d = x[i + 1] - x[i];

      var y = dF[i] * (Math.Pow(s, 2) / 2) -
            dF[i] * (Math.Pow(s, 3) / (6 * d)) +
            dF[i + 1] * (Math.Pow(s, 3) / (6 * d)) + s *
            ((temperature[i + 1] - temperature[i]) / d) - s *
            (dF[i] * (d / 3)) - s *
            (dF[i + 1] * (d / 6)) +
            temperature[i];
      GS.Points.AddXY(j, y);
   }
}

var X1X2 = form.chart1.Series.Add("x");
X1X2.MarkerStyle = MarkerStyle.Circle;
X1X2.MarkerSize = 8;
X1X2.ChartType = SeriesChartType.Point;
X1X2.Points.Clear();
for (int i = 0; i < 12; i++)
```

```
    {
        X1X2.Points.AddXY(x[i], temperature[i]);
    }
}

private Vector<double> Isvestines(double[] x, double[] y)
{
    var n = x.Length;
    var vekt = new double[n - 2, n - 2];
    var X = Matrix<double>.Build.DenseOfArray(vekt);
    var d = new double[n - 1];

    for (var i = 0; i < n - 1; i++)
    {
        d[i] = x[i + 1] - x[i];
    }

    for (var i = 0; i < n - 2; i++)
    {
        for (var j = 0; j < n - 2; j++)
        {
            if (i == 0 && j == 0 ||
                i == n - 2 && j == n - 2 ||
                j == (i) && i != 0 && i != n - 2)
            {
                X[i, j] = (d[i] + d[i + 1]) / 3;
            }
```

```
            else if (i == 0 && j == 1 ||
                    j == (i + 1) && i != 0 && i != n - 2)
            {
                X[i, j] = d[i + 1] / 6;
            }
            else if (i == n - 2 && j == n - 3 ||
                    j == i - 1 && i != 0 && i != n - 2)
            {
                X[i, j] = d[i] / 6;
            }
            else
            {
                X[i, j] = 0;
            }
        }
    }
    var v = new double[n - 2];
    for (var i = 0; i < n - 2; i++)
    {
        v[i] = (y[i + 2] - y[i + 1]) / d[i + 1] - ((y[i + 1] - y[i]) / d[i]);
    }
    var Y = Vector<double>.Build.DenseOfArray(v);

    return X.Solve(Y);
}

readonly double[] countryX = { ... };
```

```
readonly double[] countryY = { ... };

public void Uzd3()
{
    var taskuSk = 100;

    var atstumai = new double[taskuSk];
    var ve = new double[taskuSk];
    var x = new double[taskuSk];
    var y = new double[taskuSk];
    var t = new double[countryX.Length];
    atstumai[0] = 0;
    x[0] = countryX[0];
    y[0] = countryY[0];
    t[0] = 0;

    form.ClearForm(); // išvalomi programos duomenys
    form.PreparareForm(16, 24, 45, 49);
    var S1 = form.chart1.Series.Add("Splainas");
    S1.Color = Color.Red;
    S1.BorderWidth = 3;
    S1.Points.Clear();
    S1.ChartType = SeriesChartType.Line;
    var taskai = form.chart1.Series.Add("taskai");
    taskai.Color = Color.Yellow;
    taskai.BorderWidth = 5;
    taskai.Points.Clear();
```

```
taskai.ChartType = SeriesChartType.Point;

var Fx = form.chart1.Series.Add("F(x)");
Fx.ChartType = SeriesChartType.Line;
Fx.Points.Clear();
Fx.Color = Color.Black;

for (int i = 1; i < countryX.Length; i++)
{
  t[i] = t[i - 1] + Math.Sqrt(Math.Pow(countryX[i] - countryX[i - 1], 2) + Math.Pow(countryY[i] - countryY[i - 1], 2));
}
for (int i = 0; i < countryX.Length; i++)
{
  Fx.Points.AddXY(countryX[i], countryY[i]);
}

var deltaT = t[countryX.Length - 1] / (taskuSk - 1);

for (int i = 1; i < taskuSk; i++)
{
  int j = 0;
  while (j != countryX.Length && t[j] < i * deltaT)
  {
    j++;
  }
  atstumai[i] = t[j - 1];
```

```
    x[i] = countryX[j - 1];
    y[i] = countryY[j - 1];
}
for (int i = 0; i < taskuSk; i++)
{
    taskai.Points.AddXY(x[i], y[i]);
}

var x_isvestines = Isvestines(atstumai, x);
var y_isvestines = Isvestines(atstumai, y);
ve[0] = 0;
ve[taskuSk - 1] = 0;
for (int i = 0; i < taskuSk - 2; i++)
{
    ve[i + 1] = x_isvestines[i];
}
x_isvestines = Vector<double>.Build.DenseOfArray(ve);
ve[0] = 0;
ve[taskuSk - 1] = 0;
for (int i = 0; i < taskuSk - 2; i++)
{
    ve[i + 1] = y_isvestines[i];
}
y_isvestines = Vector<double>.Build.DenseOfArray(ve);

for (int i = 0; i < taskuSk - 1; i++)
{
```

```
    for (double j = atstumai[i]; j < atstumai[i + 1]; j = j + 0.1)
    {
        var s = j - atstumai[i];
        var d = atstumai[i + 1] - atstumai[i];

        var nx = x_isvestines[i] * (Math.Pow(s, 2) / 2) -
                x_isvestines[i] * (Math.Pow(s, 3) / (6 * d)) +
                x_isvestines[i + 1] * (Math.Pow(s, 3) / (6 * d)) + s *
                ((x[i + 1] - x[i]) / d) - s *
                (x_isvestines[i] * (d / 3)) - s *
                (x_isvestines[i + 1] * (d / 6)) +
                x[i];

        var ny = y_isvestines[i] * (Math.Pow(s, 2) / 2) -
                y_isvestines[i] * (Math.Pow(s, 3) / (6 * d)) +
                y_isvestines[i + 1] * (Math.Pow(s, 3) / (6 * d)) + s *
                ((y[i + 1] - y[i]) / d) - s *
                (y_isvestines[i] * (d / 3)) - s *
                (y_isvestines[i + 1] * (d / 6)) +
                y[i];

        S1.Points.AddXY(nx, ny);
    }
  }
  S1.Points.AddXY(x[0], y[0]);
}
```

```
public void Uzd4()
{
    var series = new Series[4];
    form.ClearForm();
    form.PrepareareForm(-0, 13, -10, 25);

    var Fx = form.chart1.Series.Add("F(x)");
    Fx.ChartType = SeriesChartType.Line;
    Fx.Points.Clear();

    var x = new double[]{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
    var dF = Isvestines(x, temperature);
    double[] ve = new double[12];
    ve[0] = 0;
    ve[11] = 0;
    for (int i = 0; i < 12 - 2; i++)
    {
        ve[i + 1] = dF[i];
    }
    dF = Vector<double>.Build.DenseOfArray(ve);

    for (int i = 0; i < 11; i++)
    {
        for (double j = x[i]; j <= x[i + 1]; j = j + 0.1)
        {
            var s = j - x[i];
            var d = x[i + 1] - x[i];
```

```
        var y = dF[i] * (Math.Pow(s, 2) / 2) -
                dF[i] * (Math.Pow(s, 3) / (6 * d)) +
                dF[i + 1] * (Math.Pow(s, 3) / (6 * d)) + s *
                ((temperature[i + 1] - temperature[i]) / d) - s *
                (dF[i] * (d / 3)) - s *
                (dF[i + 1] * (d / 6)) +
                temperature[i];

        Fx.Points.AddXY(j, y);
    }
}

var X1X2 = form.chart1.Series.Add("Taskai");
X1X2.MarkerStyle = MarkerStyle.Circle;
X1X2.MarkerSize = 8;
X1X2.ChartType = SeriesChartType.Point;
X1X2.Points.Clear();

for (int i = 0; i < 11; i++)
{
    X1X2.Points.AddXY(x[i], temperature[i]);
}

var a = Vector<double>.Build.DenseOfArray(temperature);
var v = Vector<double>.Build.DenseOfArray(temperature);
var colors = new[]{ Color.Green, Color.Red, Color.Violet, Color.Teal, Color.Purple };
```

```
double F2(Vector<double> _a, double _x)
{
    var sum = 0d;
    for (int i = 0; i < _a.Count; i++)
    {
        sum = sum + _a[i] * Math.Pow(_x, i);
    }
    return sum;
}

for (int i = 0; i < 4; i++)
{
    series[i] = form.chart1.Series.Add("Eilė: " + (i + 2));
    series[i].ChartType = SeriesChartType.Line;
    series[i].Color = colors[i];

    series[i].Points.Clear();
    a = Aproksimavimas(x, 12, i + 2, v);
    for (var j = 1d; j < 12; j = j + 0.1)
    {
        series[i].Points.AddXY(j, F2(a, j));
    }
}
}

private Vector<double> Aproksimavimas(double[] taskai, int taskuSk, int eile, Vector<double> v)
```

```
    {
        var vekt = new double[taskuSk, eile];
        for (var i = 0; i < taskuSk; i++)
        {
            for (var j = 0; j < eile; j++)
            {
                vekt[i, j] = Math.Pow(taskai[i], j);
            }
        }
        // iš masyvo sugeneruoja matricą, is matricos išskiria eilutę - suformuoja vektorių
        var m = Matrix<double>.Build.DenseOfArray(vekt);
        var a = Vector<double>.Build.DenseOfArray(taskai);
        var mt = m.Transpose();
        a = (mt * m).Solve(mt * v);
        form.richTextBox1.AppendText(m.ToString());
        form.richTextBox1.AppendText(v.ToString());
        form.richTextBox1.AppendText(a.ToString());
        for (var i = 0; i < eile; i++)
        {
            form.richTextBox1.AppendText("(" + a[i] + ")" + "x" + "^" + i + "+");
        }
        form.richTextBox1.AppendText("\n");
        return a;
    }
}
```