

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**COMPUTER DEPARTMENT**

### Skaitinių metodų ir algoritmų 3-a projektinė užduotis

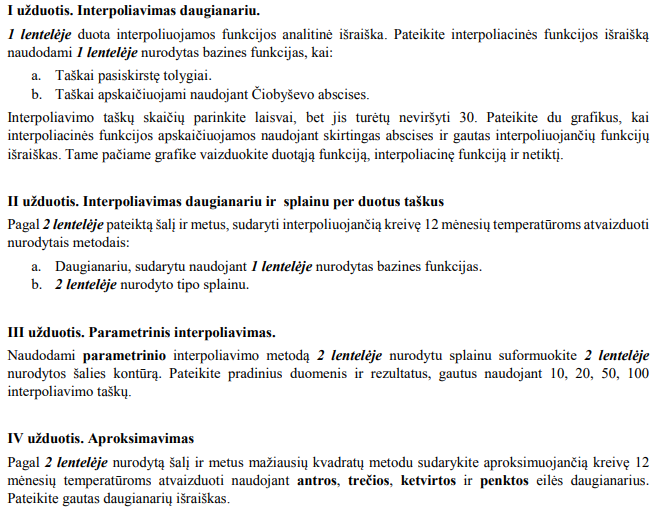
**Darbą atliko:**

IFF 6/8 grupės studentas

Tadas Laurinaitis

**Darbą vertino**:

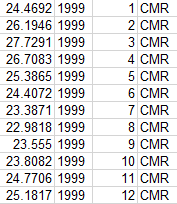
Lekt. Dalia Čalnerytė

**Užduotys**

Pav. #1 Užduočių sąrašas







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++)

{

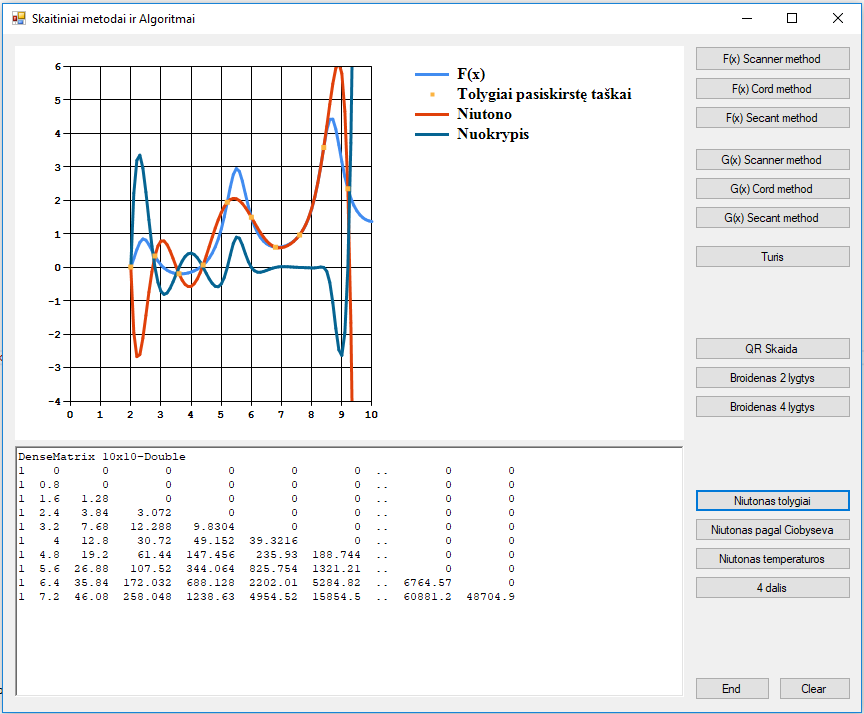
chyobysev[i] = ((double)(xmax - xmin) / 2) \* Math.Cos(Math.PI \* ((double)(2 \* i + 1)) / ((double)(2 \* N))) + ((double)(xmax + xmin) / 2);

Y[i] = Func(chyobysev[i]);

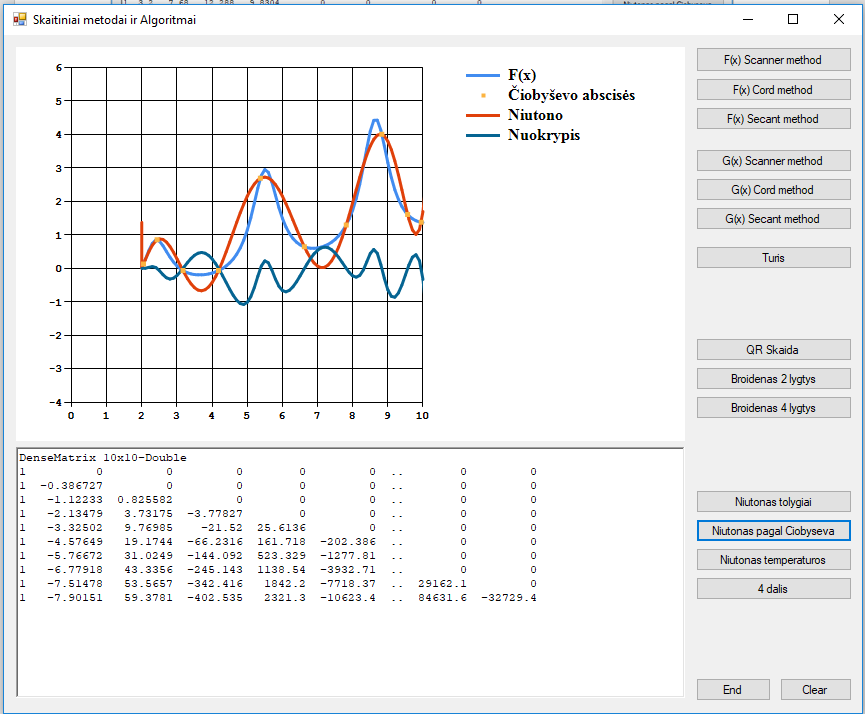
task.Points.AddXY(chyobysev[i], Func(chyobysev[i]));

}

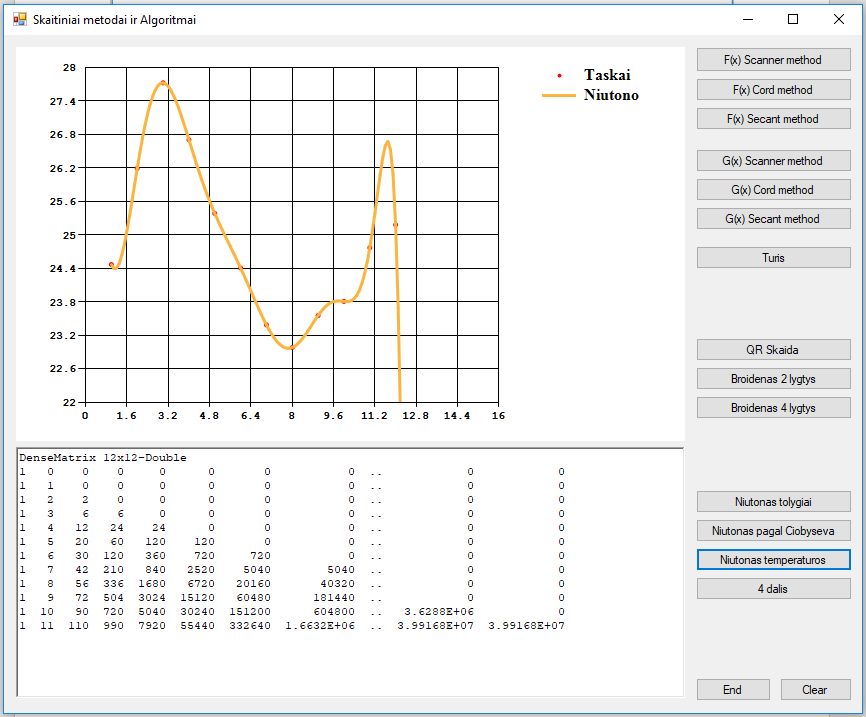
-------------------------------------------------------------------------------------------------------------------------------



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++)

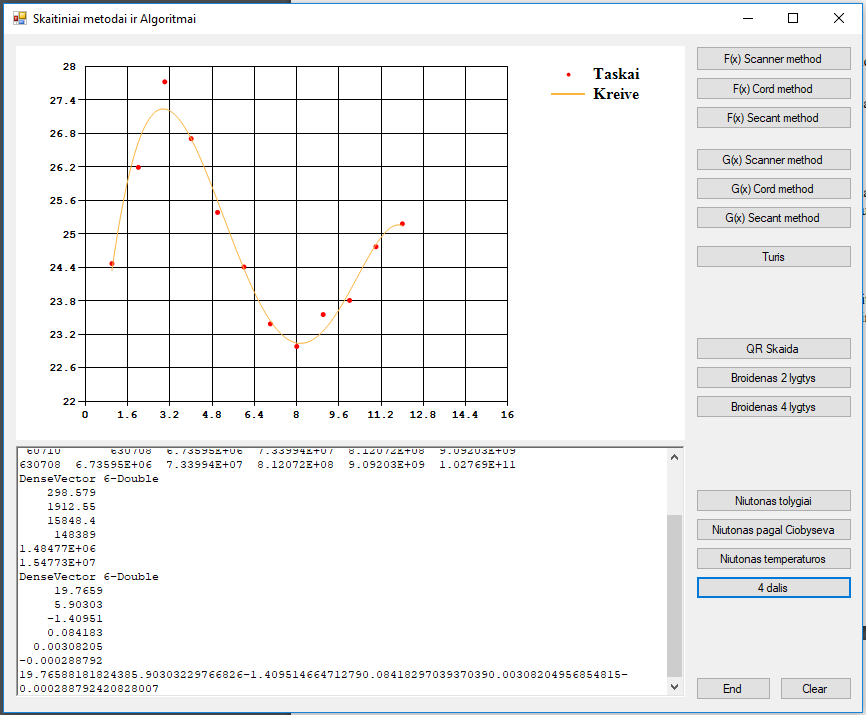
{

sum += array[i] \* Math.Pow(x, i);

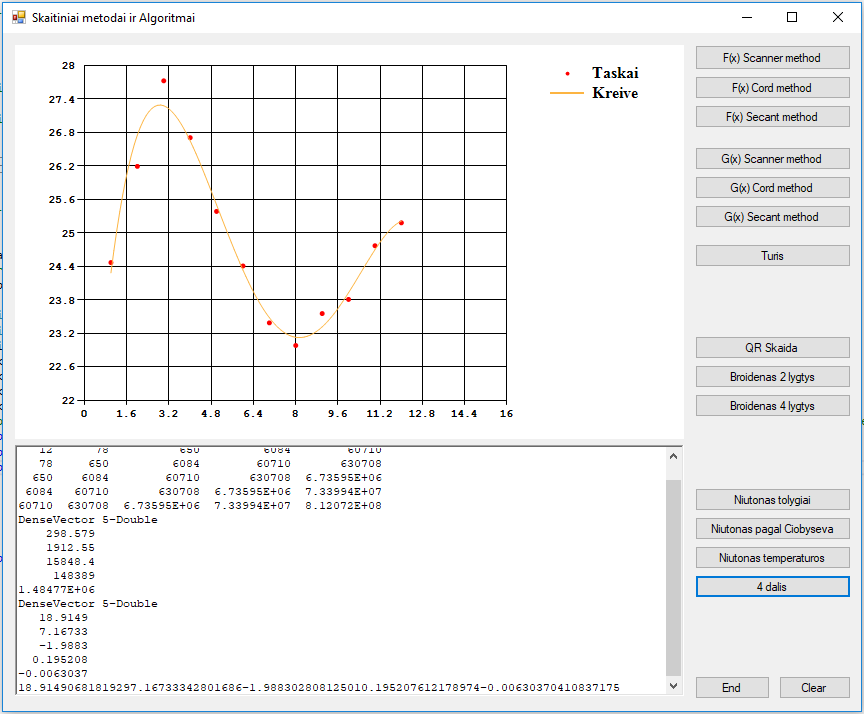
}

return sum;

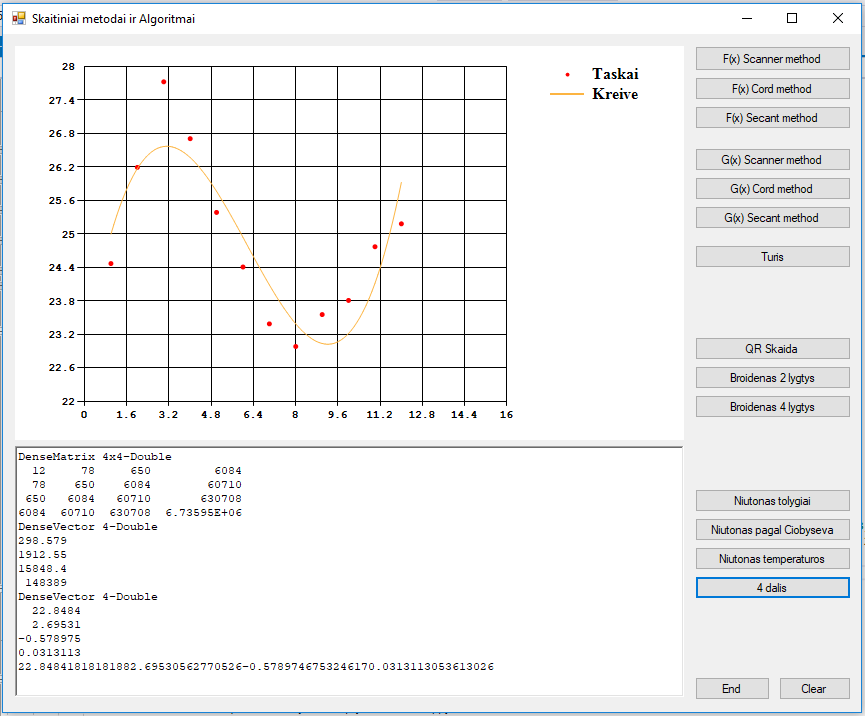
}



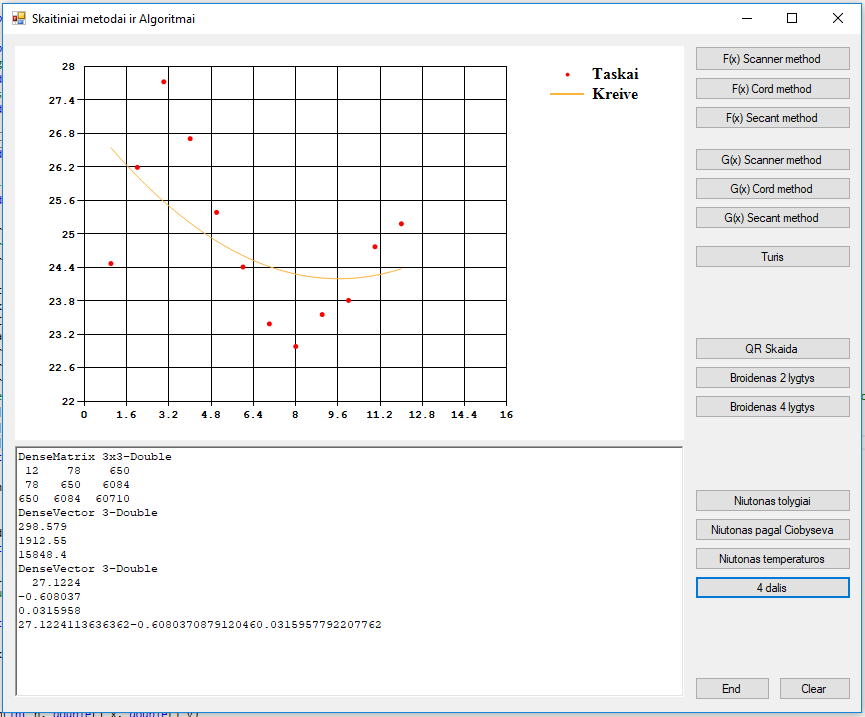
Pav. #6 4 Užduotis – Sudaryta aproksimuojanti kreivė 12 mėnesių temperatūroms atvaizduoti naudojant **penktos** eilės daugianarį. Išraiškų koeficientai matomi RichTextbox1 elemente.



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

PreparareForm(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

PreparareForm(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

//PreparareForm(0, 16, 18, 35);//deivio

PreparareForm(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