

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**COMPUTER DEPARTMENT**

### Skaitinių metodų ir algoritmų 1-ma projektinė užduotis

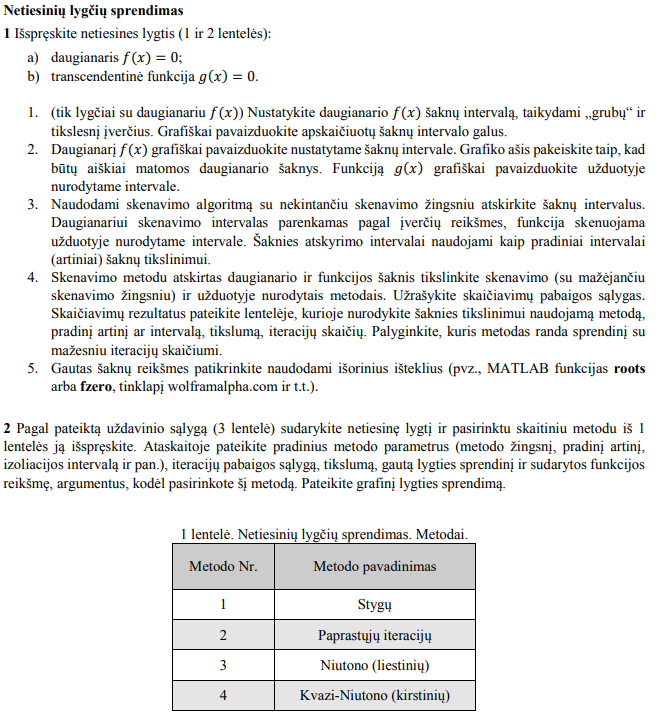
**Darbą atliko:**

IFF 6/8 grupės studentas

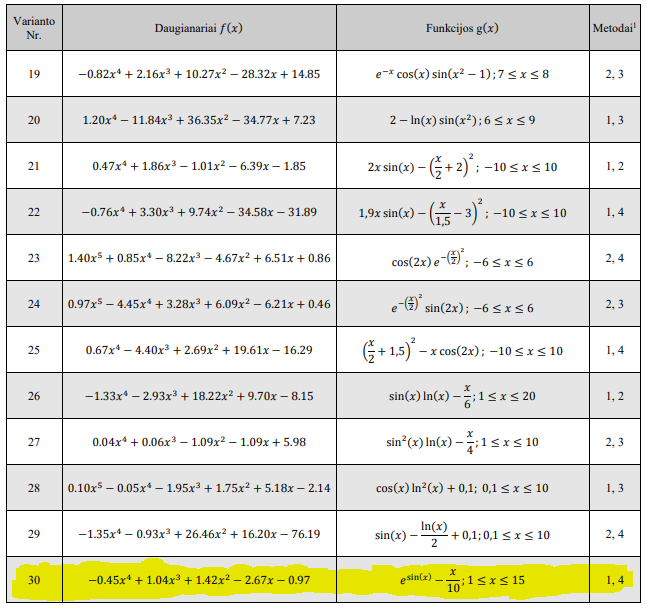
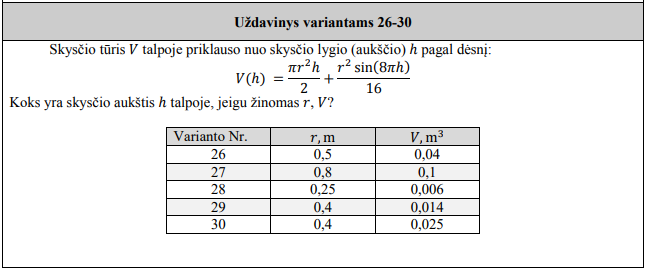
Tadas Laurinaitis

**Darbą vertino**:

Lekt. Dalia Čalnerytė

**Užduotys**

Pav. #1 uzduociu sarasas

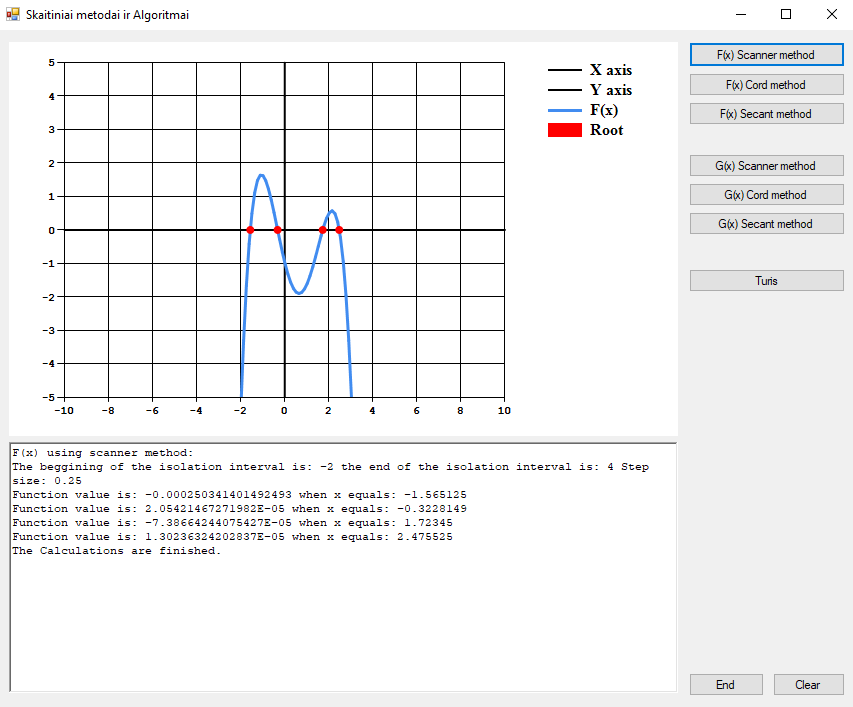
Pav. #2 uzduociu variantu sarasas 

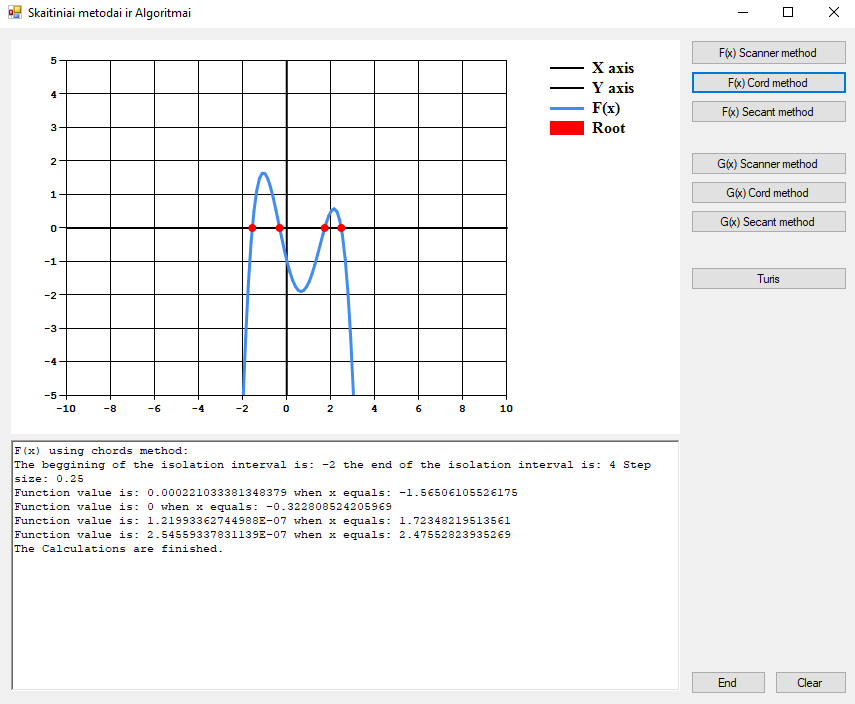
**Užduočių sprendimai:**

Spręstas variantas: 30 (f(x) ir g(x) funkcijos bei metodai kuriais jas spręsti pažymėtos geltonai)

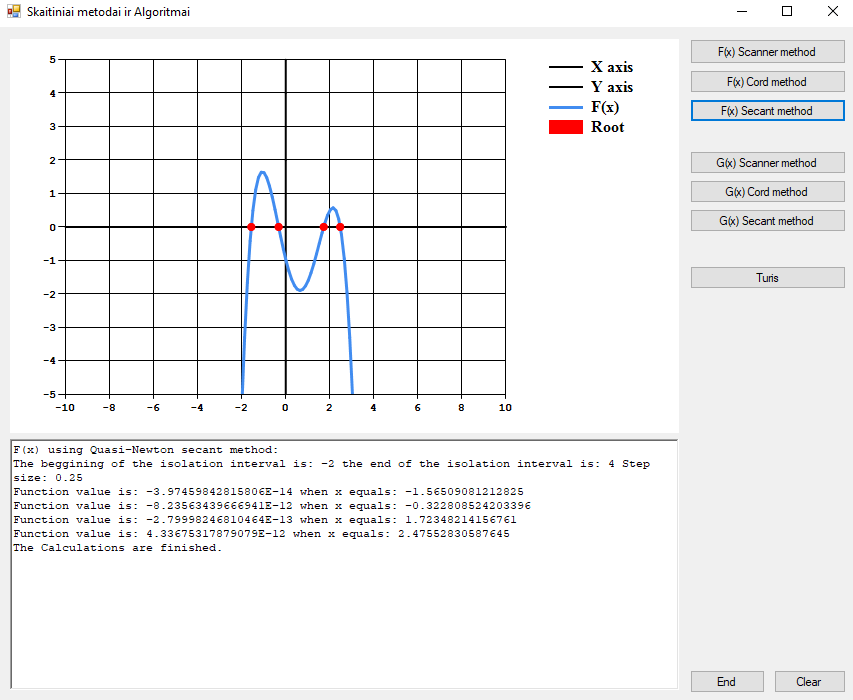
Grubus daugianario f(x) įvertis: R = 6,9(33)

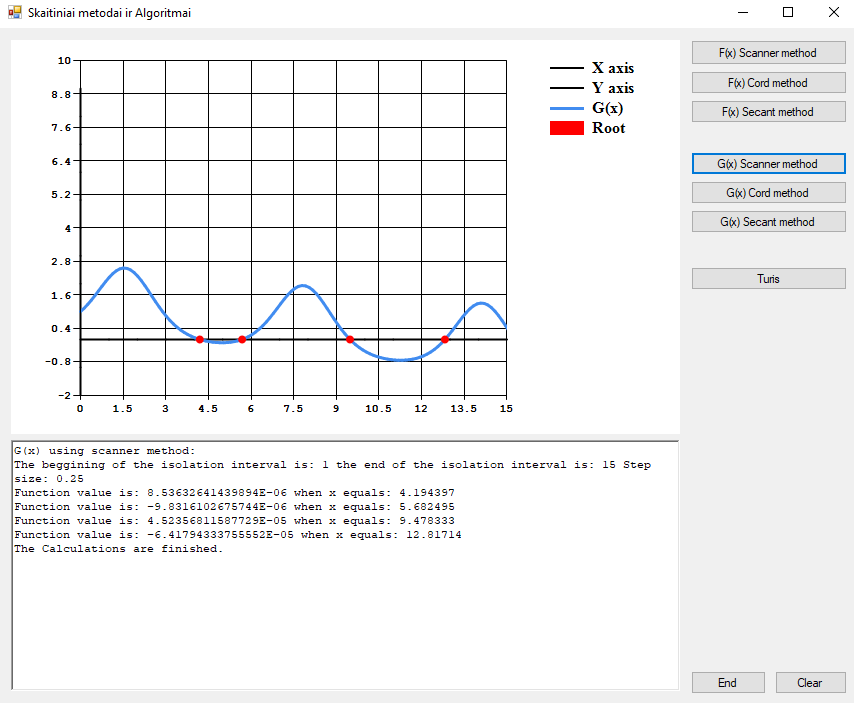
Tikslus daugianario f(x) įvertis: R = 4,1(55)

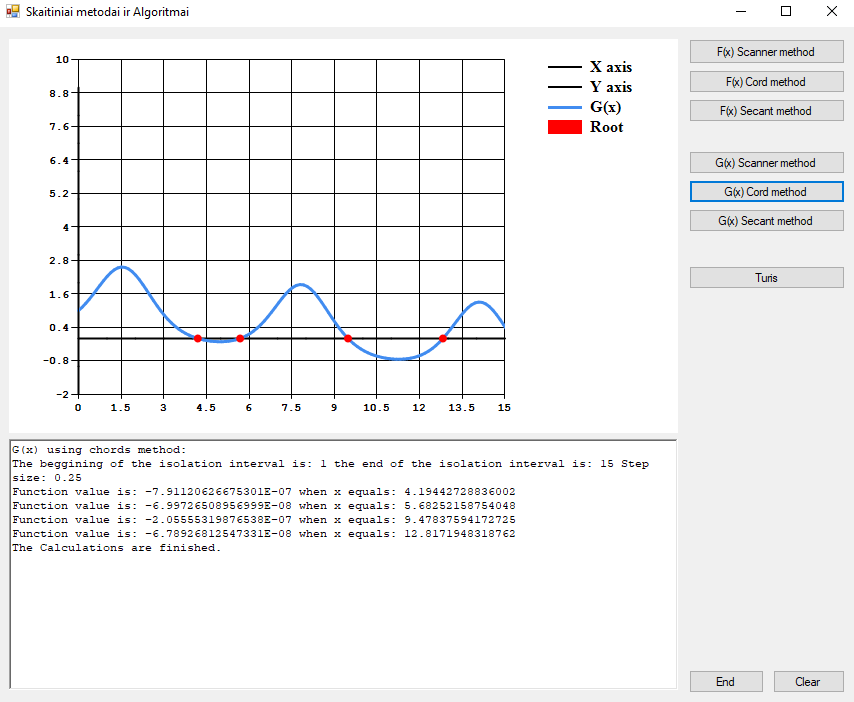
Pav. #3 f(x) skenavimo metodu

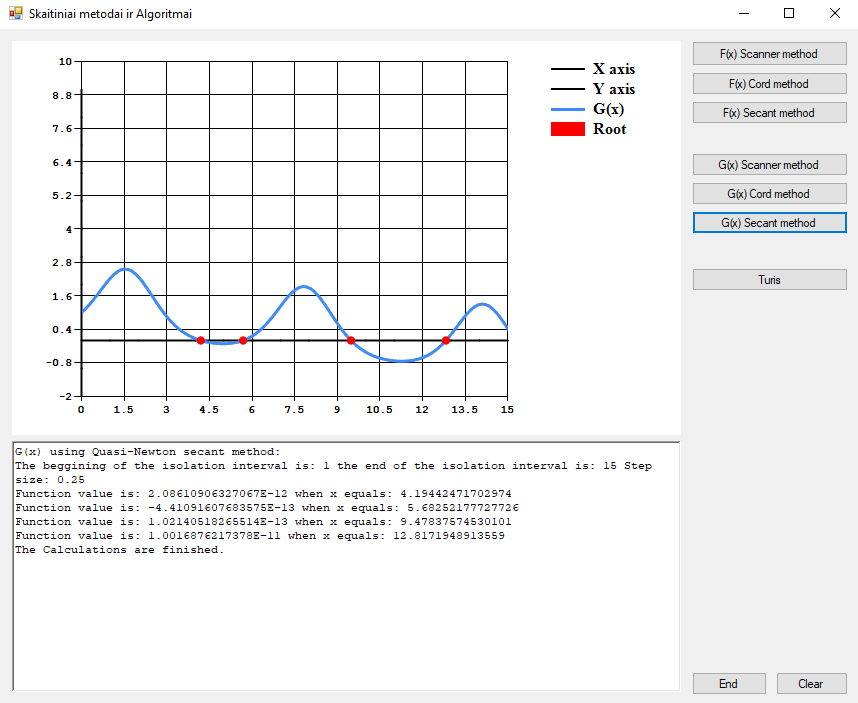


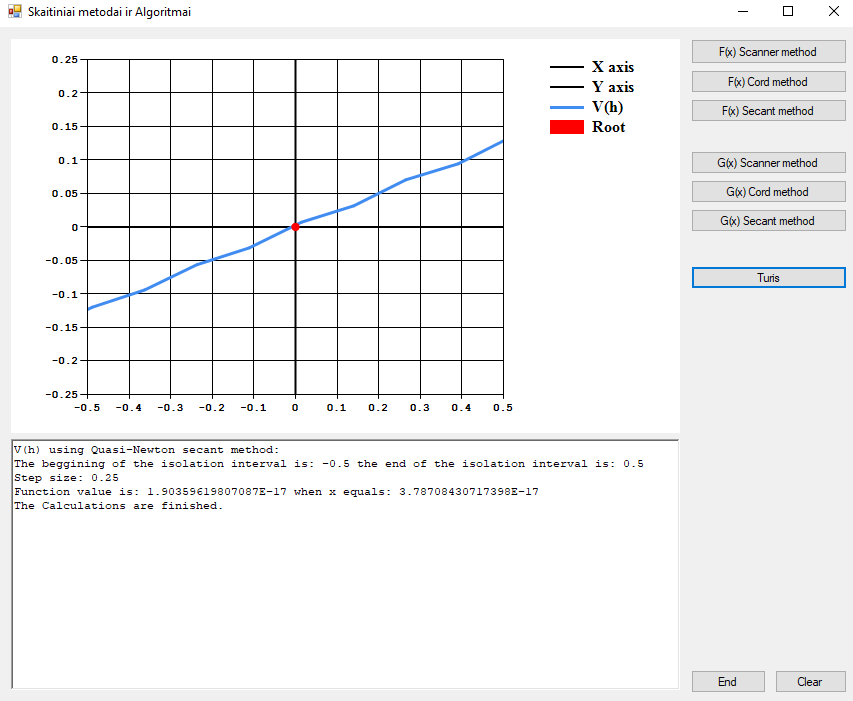
Pav. #4 f(x) stygu metodu

Pav. #5 f(x) Kvasi-Niutono kirstiniu metodu 

Pav. #6 g(x) skenavimo metodu 

Pav. #7 g(x) stygu metodu 

Pav. #8 g(x) Kvasi-Niutono kirstiniu metodu 

Pav. #9 V(h) Kvasi-Niutono kirstiniu metodu 

**Programos kodo fragmentai:**

/// <summary>

/// A function which iterates over the selected interval of values and checks if the sign of current value is the same as the sign of value which comes after the next step.

/// If the signs are different, then the step size is cut by half, and the iteration continues. After a fixed iteration count, the function prints out the value.

/// </summary>

/// <param name="start"></param>

/// <param name="end"></param>

/// <param name="step"></param>

/// <param name="function"></param>

private void ScannerMethod(double start, double end, double step, Func<double, double> function)

{

float x = (float)start; //starting point

iii = 0; //iteration number

//while x is in the selected interval

while (x < end)

{

//if signs are different then cut the step by half

if (Math.Sign(function(x + step)) != Math.Sign(function(x)))

{

step = step \* 0.5;

iii++;

}

//if sign stays the same, proceed to the next step

else

{

x += (float)step;

}

//after a number (iii) of iterations, print the result

if (iii == 13)

{

Root.Points.AddXY(x, 0);

richTextBox1.AppendText("Function value is: " + function(x) + " when x equals: " +x +"\n");

break;

}

}

}

/// <summary>

/// A function which finds roots of the function in the given interval by comparing starting point with the xMid point which is calculated using coefficient k, and start and end values.

/// If signs of both start and xMid point values are the same, then the start point is equal to xMid point for the next iteration, otherwise the end point is equal to xMid point. This sequence

/// is carried out a selected number of times, more times meaning greater accuracy, until the xMid becomes the root.

/// </summary>

/// <param name="start"></param>

/// <param name="end"></param>

/// <param name="times"></param>

/// <param name="function"></param>

private void ChordsMethod(double start, double end, int times, Func<double, double> function)

{

double k = 0; //coefficient used in xMid calculation

double xMid = 0;

for (int i = 0; i < times; i++)

{

k = Math.Abs(function(start) / function(end));

xMid = ((start + k \* end) / (1 + k));

if (Math.Sign(function(start)) == Math.Sign(function(xMid)))

{

start = xMid;

}

else

{

end = xMid;

}

}

Root.Points.AddXY(xMid, 0);

richTextBox1.AppendText("Function value is: " + function(xMid) + " when x equals: " + xMid + "\n");

}

/// <summary>

///

/// </summary>

/// <param name="start"></param>

/// <param name="iterations"></param>

/// <param name="function"></param>

private void SecantMethod(ref double start, ref int iterations, Func<double, double> function)

{

double h = 0.001;

double prev;

while (Math.Abs(function(start)) > 1e-9 && iterations < 250)

{

prev = start;

start = start - Math.Pow(((function(start) - function(start - h)) / h), -1) \* function(start);

h = start - prev;

iterations++;

}

richTextBox1.AppendText("Function value is: " + function(start) + " when x equals: " + start + "\n");

Root.Points.AddXY(start, 0);

}