**Цель работы:** знакомство с задачей минимизации функций одной переменной **м**етодами, требующими вычисления производной.

**Задание:**

Реализовать алгоритмы одномерной минимизации функции:

* метод средней точки
* метод хорд
* метод Ньютона

**Индивидуальный вариант:**



**Выполнение заданий**

Построим график нашей функции

График исследуемой функции:

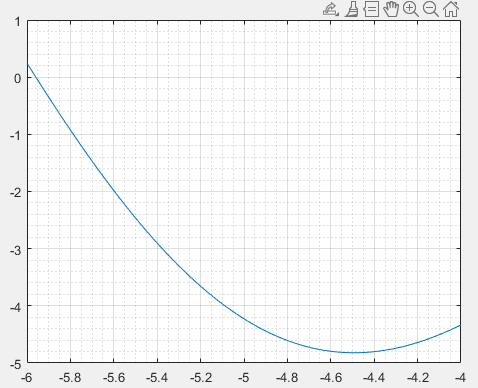


Рис. 1 График

Найдем аналитическое решение нашей функции c помощью функции Matlab min:

ans =

-4.8206

**Метод средней точки**

Реализация кода в C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp10

{

class Program

{

static double F(double X)

{

double fx;

fx = X \* Math.Sin(X) + 2 \* Math.Cos(X);

return fx;

}

static double f(double X)

{

double fx;

fx = X \* Math.Cos(X) - Math.Sin(X);

return fx;

}

static void Main(string[] args)

{

double xk;

double a = -6;

double b = -4;

double eps = Convert.ToDouble(Console.ReadLine());

while(true)

{

xk = (a + b) / 2;

if(f(xk)<0)

{

a = xk;

}

else

{

b = xk;

}

if ((Math.Abs(b - a) < eps))

{

Console.WriteLine($"Значение x = {xI}");

Console.WriteLine($"Значение функции при = {F(xI)}");

Console.WriteLine($"Количество итераций при точности {eps} = {i}");

Console.WriteLine($"Количество вычислений при точности {eps} = {i}");

break;

}

Console.ReadKey(true);

Console.ReadKey();

Console.ReadLine();

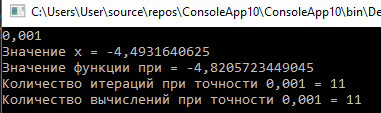
}

}

}

}

Полученное решение имеет следующий вид:



**Метод хорд**

Реализация кода в C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp12

{

class Program

{

static double F(double X)

{

double fx;

fx = X \* Math.Sin(X) + 2 \* Math.Cos(X);

return fx;

}

static double f(double X)

{

double fx;

fx = X \* Math.Cos(X) - Math.Sin(X);

return fx;

}

static void Main(string[] args)

{

double a = -6;

double b = -4;

double xI, Xzv, i;

i = 0;

xI = a - f(a)/(f(a)-f(b))\*(a-b);

double eps = Convert.ToDouble(Console.ReadLine());

while (true)

{

xI = a - f(a) / (f(a) - f(b)) \* (a - b);

if (f(xI) > 0)

{

b = xI;

}

else

{

a = xI;

}

i++;

if ((Math.Abs(b - a) < eps))

{

Console.WriteLine($"Значение x = {xI}");

Console.WriteLine($"Значение функции при = {F(xI)}");

Console.WriteLine($"Количество итераций при точности {eps} = {i}");

Console.WriteLine($"Количество вычислений при точности {eps} = {i\*3}");

break;

}

}

Console.ReadKey(true);

Console.ReadKey();

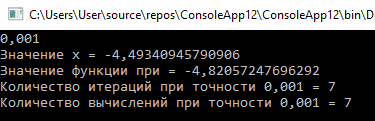
Console.ReadLine();

}

}

}

Полученное решение имеет следующий вид:



**Метод Ньютона**

Реализация кода в C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp11

{

class Program

{

static double F(double X)

{

double fx;

fx = X \* Math.Sin(X) + 2 \* Math.Cos(X);

return fx;

}

static double f(double X)

{

double fx;

fx = X \* Math.Cos(X) - Math.Sin(X);

return fx;

}

static double ff(double X)

{

double fx;

fx = - X\*Math.Sin(X);

return fx;

}

static void Main(string[] args)

{

double x, i, dx;

x = -5;

dx = F(x);

i = 0;

double eps = Convert.ToDouble(Console.ReadLine());

while((Math.Abs(F(x)) > eps))

{

x = x - dx / ff(x);

i++;

dx = f(x);

if (Math.Round(Math.Abs(f(x)), 10, MidpointRounding.ToEven) <= eps)

{

Console.WriteLine($"Значение x = {x}");

Console.WriteLine($"Значение функции при = {F(x)}");

Console.WriteLine($"Количество итераций при точности {eps} = {i}");

Console.WriteLine($"Количество вычислений при точности {eps} = {i}");

break;

}

}

Console.ReadKey(true);

Console.ReadKey();

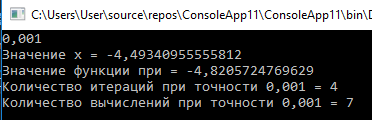
Console.ReadLine();

}

}

}

Полученное решение имеет следующий вид:



**Сравнение методов**

Проведём сравнение методов. Для этого найдём зависимость количества вычислений функции от точности решения.

x = -6:0.00001:-4;

y = x.\*sin(x)+2\*cos(x);

min(y)

MDP = [5, 8, 11, 15];

MH = [3, 6, 9, 18];

MNEWT = [5, 5, 7, 7];

eps = [0.1, 0.01, 0.001, 0.0001];

hold on

grid on

plot(abs(log10(eps)), MDP, '-.\*');

plot(abs(log10(eps)), MH, '-.\*');

plot(abs(log10(eps)), MNEWT, '-.\*');

xlabel('0.1^-^x');

ylabel('Кол-во вычислений функции');

legend({'Метод хорд', 'Метод средней точки', 'Метод Ньютона'}, 'Location', 'northwest');

hold off

Постоим графики зависимостей



Из рисунка видно, что метод ньютона даёт наилучший результат при меньшем количестве вычислений функции.