

Задание:

Реализовать алгоритмы одномерной минимизации функции:

- метод деления интервала пополам
- метод дихотомии
- метод золотого сечения
- метод Фибоначчи
- метод квадратичной интерполяции

Выполнение заданий

Построим график нашей функции

График исследуемой функции:

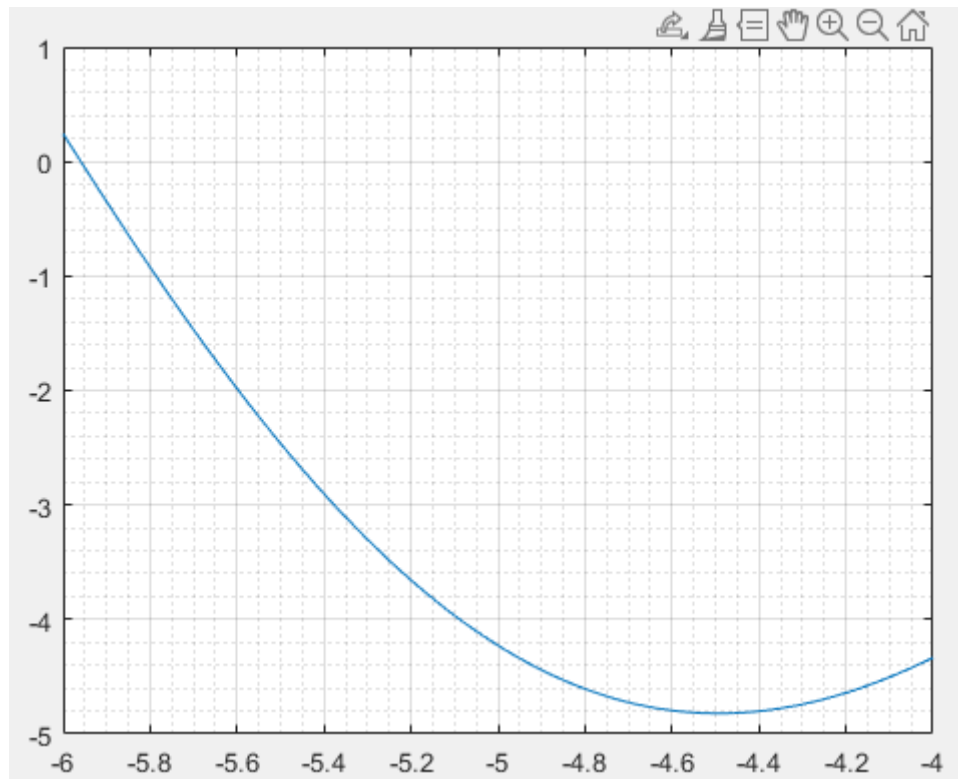


Рис. 1 График $x \sin x + 2 \cos x$

Найдем аналитическое решение нашей функции с помощью функции Matlab `min`:

ans =

-4.8206

Метод деления интервала пополам

Реализация кода в C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp5
{
    class Program
    {
        static double f(double X)
        {
            double fx;
            fx = X * Math.Sin(X) + 2 * Math.Cos(X);
            return fx;
        }
        static void Main(string[] args)
        {
            double x, y, z, i;
            i = 0;
            double a = Convert.ToDouble(Console.ReadLine());
            double b = Convert.ToDouble(Console.ReadLine());
            double eps = Convert.ToDouble(Console.ReadLine());
            while (true)
            {
                x = (a + b) / 2;
                y = (a + x) / 2;
                z = (x + b) / 2;
                if (f(y) < f(x))
                {
                    b = x;
                }
                if (f(z) < f(x))
                {
                    a = x;
                }
                if (f(y) >= f(x) && f(z) >= f(x))
                {
                    a = y;
                    b = z;
                }
                i++;
                if (Math.Round(Math.Abs(b - a), 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 * 3}");
                    break;
                }
            }
            Console.ReadKey(true);
            Console.ReadKey();
            Console.ReadLine();
        }
    }
}
```

Полученное решение имеет следующий вид:

```
Microsoft Visual Studio Debug Console
-6
-4
0,001
Значение x = -4,4931640625
Значение функции при = -4,8205723449045
Количество итераций при точности 0,001 = 11
Количество вычислений при точности 0,001 = 33
```

Метод дихотомии

Реализация кода в С#

```
using System;

namespace ConsoleApp5
{
    class Program
    {
        static double f(double X)
        {
            double fx;
            fx = X * Math.Sin(X) + 2 * Math.Cos(X);
            return fx;
        }
        static void Main(string[] args)
        {
            double x, a, b, y, z;
            int i = 0;
            a = -6;
            b = -4;
            double e = Convert.ToDouble(Console.ReadLine());
            double eps = Convert.ToDouble(Console.ReadLine());
            while (true)
            {
                y = (a + b - e) / 2;
                z = (a + b + e) / 2;
                if (f(y) <= f(z))
                {
                    b = z;
                }
                if (f(y) > f(z))
                {
                    a = y;
                }
                x = (a + b) / 2;
                i++;
                if (Math.Round(Math.Abs(b - a), 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 * 2}");
                    break;
                }
            }
            Console.ReadKey(true);
            Console.ReadKey();
            Console.ReadLine();
        }
    }
}
```

Полученное решение имеет следующий вид:

```
C:\Users\User\source\repos\ConsoleApp6\ConsoleApp6\bin\Debug\C
0,001
0,001
Значение x = -4,49340943936312
Значение функции при = -4,82057247696292
Количество итераций при точности 0,001 = 36
Количество вычислений при точности 0,001 = 72
```

Метод золотого сечения

Реализация кода в С#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp5
{
    class Program
    {
        static double f(double X)
        {
            double fx;
            fx = X * Math.Sin(X) + 2 * Math.Cos(X);
            return fx;
        }
        static void Main(string[] args)
        {
            double x, a, b, y, z;
            int i = 0;
            a = -6;
            b = -4;
            double k = (3 - Math.Sqrt(5)) / 2;
            double eps = Convert.ToDouble(Console.ReadLine());
            while (true)
            {
                y = a + k*(b - a);
                z = a + b - y;
                if (f(y) <= f(z))
                {
                    b = z;
                    y = a + b - y;
                    z = y;
                }
                else if (f(y) > f(z))
                {
                    a = y;
                    y = z;
                    z = a + b - z;
                }
                x = (a + b) / 2;
                i++;
                if (Math.Round(Math.Abs(b - a), 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 * 2}");
                    break;
                }
            }
            Console.ReadKey(true);
            Console.ReadKey();
            Console.ReadLine();
        }
    }
}
```

Полученное решение имеет следующий вид:

C:\Users\User\source\repos\ConsoleApp7\ConsoleApp7\bin\Debu

```
0,001
Значение x = -4,49324912098007
Значение функции при = -4,82057242058542
Количество итераций при точности 0,001 = 16
Количество вычислений при точности 0,001 = 32
```

Метод Фибоначчи

Реализация кода в C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

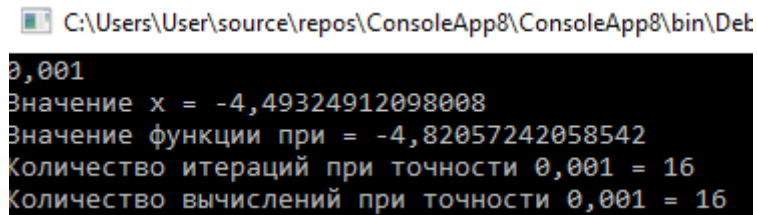
namespace ConsoleApp5
{
    class Program
    {
        static double f(double X)
        {
            double fx;
            fx = X * Math.Sin(X) + 2 * Math.Cos(X);
            return fx;
        }
        static void Main(string[] args)
        {
            double x, nye;
            double a = -6;
            double b = -4;
            double[] ch_fib = new double[5000];
            double lyambda;
            int i = 0;
            double eps = Convert.ToDouble(Console.ReadLine());
            double Fn = (Math.Abs(a - b)) / eps;
            ch_fib[0] = 1;
            ch_fib[1] = 1;
            long m;
            for(m = 2 ; m <= 400; m++)
            {
                ch_fib[m] = ch_fib[m-1] + ch_fib[m-2];
            }
            m = m - 1;
            lyambda = a + (ch_fib[m - 2] / ch_fib[m]) * (b - a);
            nye = a + (ch_fib[m - 1] / ch_fib[m]) * (b - a);
            while (true)
            {
                if (f(lyambda) > f(nye))
                {
                    a = lyambda;
                    lyambda = nye;
                    nye = a + ch_fib[m - i - 1] / ch_fib[m - i] * (b - a);
                }
                else if(f(lyambda) < f(nye))
                {
                    b = nye;
                    nye = lyambda;
                    lyambda = a + ch_fib[m - i - 2] / ch_fib[m - i] * (b - a);
                }
                x = (a + b) / 2;
                i++;
                if(Fn - 3 < i || Math.Round(Math.Abs(b - a), 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.WriteLine(ch_fib[i]);
    }
    Console.ReadKey(true);
    Console.ReadKey();
    Console.ReadLine();
}
}
}

```

Полученное решение имеет следующий вид:



```

C:\Users\User\source\repos\ConsoleApp8\ConsoleApp8\bin\De:
0,001
Значение x = -4,49324912098008
Значение функции при = -4,82057242058542
Количество итераций при точности 0,001 = 16
Количество вычислений при точности 0,001 = 16

```

Метод квадратичной интерполяции

Реализация кода в C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp5
{
    class Program
    {
        static double f(double X)
        {
            return X * Math.Sin(X) + 2 * Math.Cos(X);
        }
        static double Find(double xs, double h, double eps1, double eps2, int max_step)
        {
            double[] x = new double[3];
            double[] f_x = new double[3];
            int k;
            int i_min = 0, i_max = 0;
            double xn, f_xn, a1, a2;
            xn = 0;
            // Задаём начальную точку
            // Вычисляем точки x0, x1, x2
            x[0] = xs;
            x[1] = xs + h;
            if (f(x[0]) > f(x[1]))
            {
                x[2] = xs + 2 * h;
            }
            else
            {
                x[2] = xs - h;
            }
            // Вычисляем значения функции в этих точках
            f_x[0] = f(x[0]);
            f_x[1] = f(x[1]);
            f_x[2] = f(x[2]);
            /// находим F_min = min(f1, f2, f3)
            for (k = 0; k < max_step; k++)
            {
                if (f_x[0] < f_x[1])
                {
                    if (f_x[0] < f_x[2]) i_min = 0;
                    else i_min = 2;
                }
                else
                {
                    if (f_x[1] < f_x[2]) i_min = 1;
                    else i_min = 2;
                }
            }
        }
    }
}

```

```

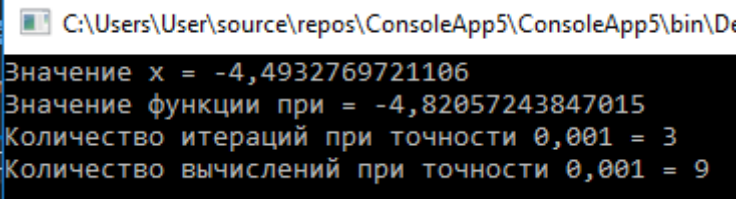
    }
    a1 = (f_x[1] - f_x[0]) / (x[1] - x[0]);
    a2 = 1.0 / (x[2] - x[1]) * ((f_x[2] - f_x[0]) / (x[2] - x[0]) - (f_x[1] - f_x[0]) / (x[1]
- x[0]));
    xn = (x[1] + x[0]) * 0.5 - a1 / (2 * a2);
    f_xn = f(xn);
    // Проверяем условия окончания. Если оба условия выполнены, то процедура окончена
    if ((Math.Abs((xn - x[i_min]) / xn) < eps1) && (Math.Abs((f_xn - f_x[i_min]) / f_xn) <
eps2))
    {
        Console.WriteLine($"Значение x = {xn}");
        Console.WriteLine($"Значение функции при = {f(xn)}");
        Console.WriteLine($"Количество итераций при точности {eps1} = {k}");
        Console.WriteLine($"Количество вычислений при точности {eps1} = {k * 3}");
        return xn;
    }
    // Выбираем наилучшую точку
    // и две точки по обе стороны от нее. Переобозначить эти точки в порядке возрастания
    if (f_x[0] >= f_x[1])
    {
        if (f_x[0] > f_x[2]) i_max = 0;
        else i_max = 2;

        if (f_x[1] > f_x[2]) i_max = 1;
        else i_max = 2;
    }
    if (f_xn < f_x[i_min])
    {
        x[i_max] = xn;
        f_x[i_max] = f_xn;
    }
    else
    {
        x[i_max] = 2 * x[i_min] - xn;
        f_x[i_max] = f(x[i_max]);
    }
}
Console.WriteLine($"Значение x = {xn}");
Console.WriteLine($"Значение функции при = {f(xn)}");
Console.WriteLine($"Количество итераций при точности {eps1} = {k}");
Console.WriteLine($"Количество вычислений при точности {eps1} = {k * 2}");
return xn;
}
static void Main(string[] args)
{
    double x;
    x = Find(-6, 0.5, 0.001, 0.001, 100);

    Console.ReadKey(true);
    Console.ReadKey();
    Console.ReadLine();
}
}
}

```

Полученное решение имеет следующий вид:



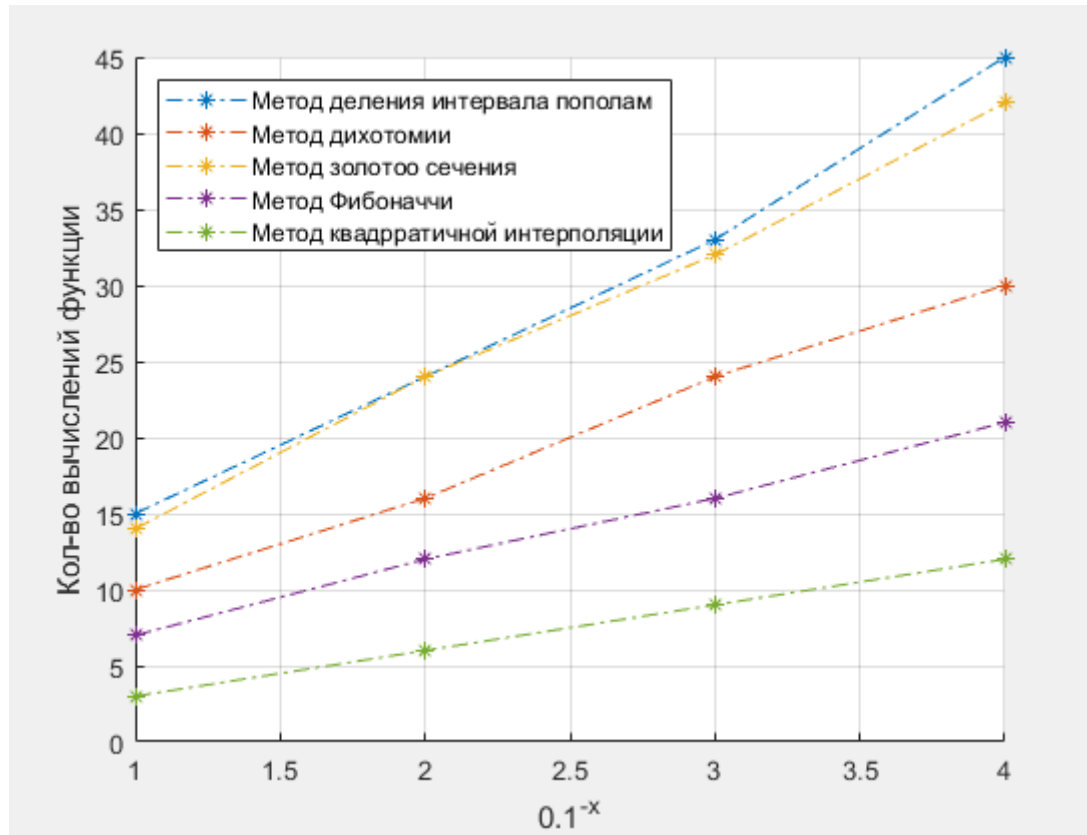
```

C:\Users\User\source\repos\ConsoleApp5\ConsoleApp5\bin\De
Значение x = -4,4932769721106
Значение функции при = -4,82057243847015
Количество итераций при точности 0,001 = 3
Количество вычислений при точности 0,001 = 9

```

Сравнение методов

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



Построим графики зависимостей

Из рисунка видно, что метод квадратичной интерполяции даёт отличный результат при меньшем количестве вычислений функции.

```
x = -6:0.00001:-4;
y = x.*sin(x)+2*cos(x);
min(y)
MDIP = [15, 24, 33, 45];
MD = [10, 16, 24, 30];
MZS = [14, 24, 32, 42];
MF = [7, 12, 16, 21];
MKI = [3, 6, 9, 12];
eps = [0.1, 0.01, 0.001, 0.0001];
hold on
grid on
plot(abs(log10(eps)), MDIP, '-.*');
plot(abs(log10(eps)), MD, '-.*');
plot(abs(log10(eps)), MZS, '-.*');
plot(abs(log10(eps)), MF, '-.*');
plot(abs(log10(eps)), MKI, '-.*');
xlabel('0.1^-x');
ylabel('Кол-во вычислений функции');
legend({'Метод деления интервала пополам', 'Метод дихотомии', 'Метод золотого сечения', 'Метод Фибоначчи', 'Метод квадратичной интерполяции'}, 'Location', 'northwest');
hold off
```