Федеральное агентство по образованию РФ
Новгородский государственный университет им. Ярослава Мудрого

Кафедра ИТИС

Однопараметрическая оптимизация методами поразрядного приближения и золотого сечения

Выполнил студент группы 9311
Лопатин А.С.

Великий Новгород, 2009

1. Цель работы

Реализовать алгоритмы поиска экстремума функции методами поразрядного приближения и золотого сечения на языке программирования C++.

2. Математическая модель решения

Дана функция $f(x)$, границы a и b, изначальное количество узлов оптимизации n и точность ε. Необходимо найти экстремум функции между границами a и b с учетом точности ε. Методы заключается в последовательном исключении частей отрезка.

Для метода поразрядного приближения выбирается шаг между узлами по формуле $h = \frac{b-a}{n}$. Начальное значение x_1 устанавливается в a, x_2 будет равным $x_1 + h$, т. е. на шаг дальше. Вычисляется 2 значения функции: $f_1 = f(x_1)$ и $f_2 = f(x_2)$. Если $f_1 < f_2$ (при поиске максимума; для минимума будет наоборот: $f_1 > f_2$) — экстремум еще не пропустили, поэтому увеличиваем x_1 на шаг h. Для определения чего мы ищем (минимум или максимум), можно умножить стороны неравенства на c, принимающего значения $\{-1, 1\}$, где –1 означает что мы ищем минимум, а 1 — максимум. Тут же можно убедиться что экстремум присутствует в заданном промежутке исходя из неравенства $x_1 \leq b$. Если же мы пропустили экстремум — отбрасываем отрезок (x_2, b), а a теперь будет равен $x_2 - 2 \cdot h$. Новый шаг определяется по формуле $h = \frac{h}{n}$, а начальное значение x снова задается на a. Точность можно определять исходя из неравенства $|f_1 - f_2| \leq \varepsilon$. Оптимальная точка x_{opt} выбирается средним арифметическим a и b.

Метод золотого сечения заключается в делении отрезка в пропорциях золотого сечения, которые определены следующим образом:

$$\varphi = \frac{\sqrt{5}+1}{2} \approx 1{,}6180339887 = \frac{b-a}{b-x_1} = \frac{b-a}{x_2-a}$$

где φ — пропорция золотого сечения, x_1 и x_2 — точки, которые делят отрезок (a, b) на отрезки (a, x_2) и (x_1, b) в отношении золотого сечения. Отсюда

можно выразить значения этих точек: $x_1 = b - \frac{(b-a)}{\varphi}$ и $x_2 = a + \frac{(b-a)}{\varphi}$. Тот из концов отрезка, к которому среди новых точек ближе оказалась та, значение в которой минимально (при поиске максимума; для минимума — наоборот) отбрасывают. Точность можно определять исходя из исходя из неравенства $|b - a| \leq \varepsilon$. Оптимальная точка опять же определяется средним арифметическим точек x_1 и x_2 (или a и b).

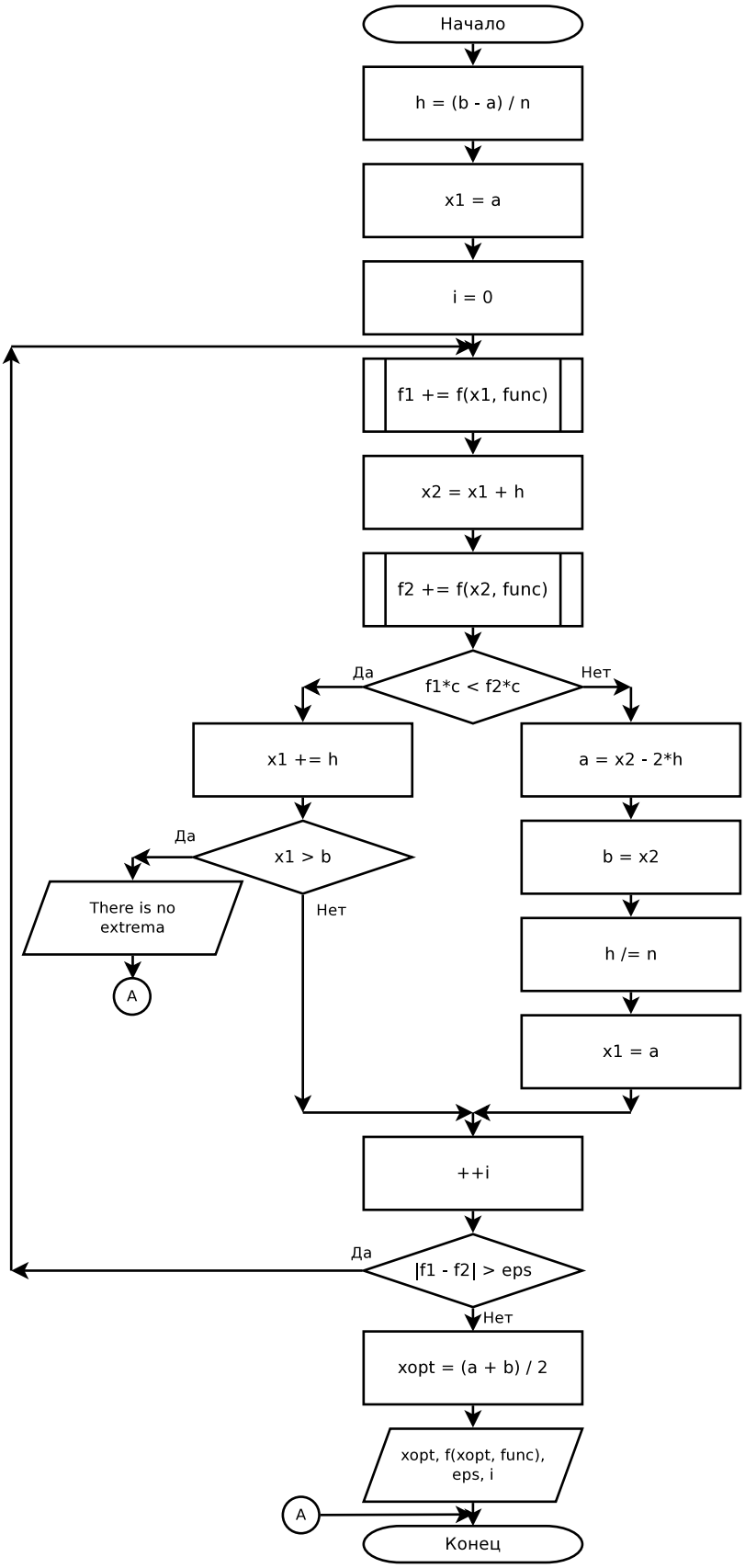
3. Спецификация

Таблица переменных:

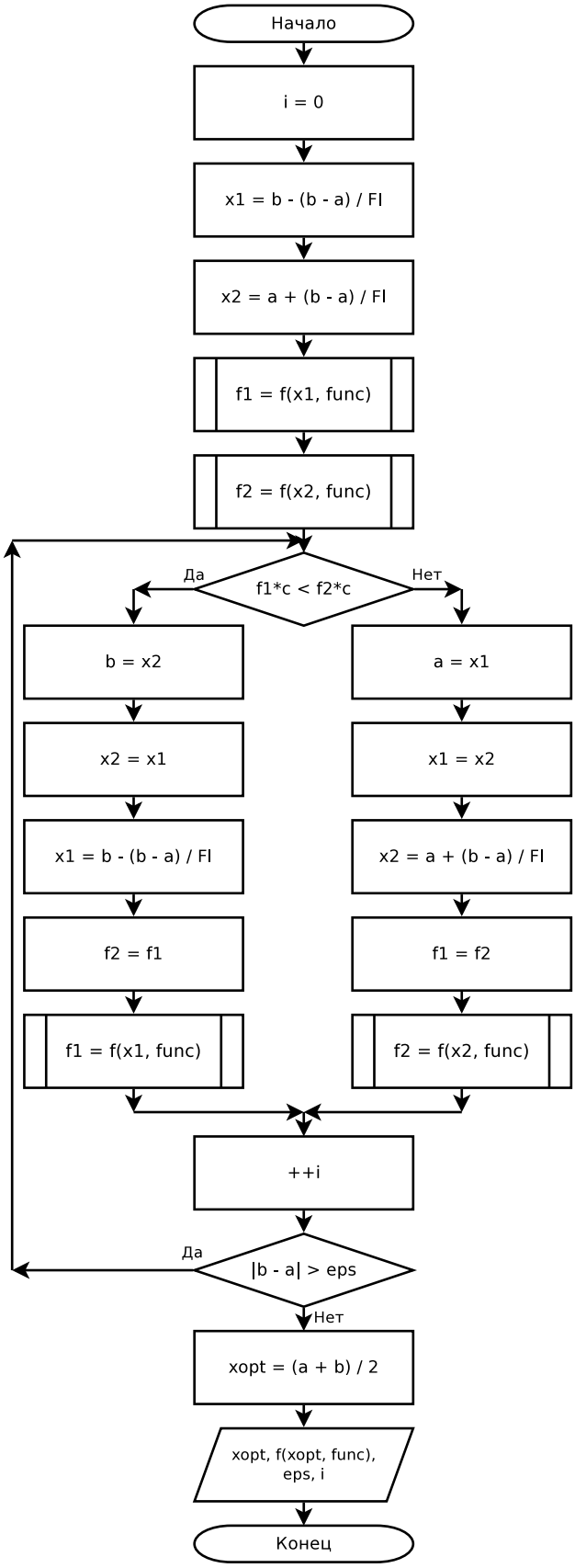
Исходное значение	Идентификатор	Тип	Вид	Размерность	Назначение
φ	FI	Вещественная константа	Простой	–	Пропорция золотого сечения
a	a	Вещественный	Простой	–	Левая граница
b	b	Вещественный	Простой	–	Правая граница
ε	eps	Вещественный	Простой	–	Точность
n	n	Целый	Простой	–	Количество узлов оптимизации
func	func	Целый	Простой	–	Номер функции
maxima	maxima	Логический	Простой	–	Вид экстремума
c	c	Целый	Простой	–	Множитель для поиска минимума или максимума
i	i	Целый	Простой	–	Счетчик итераций
h	h	Вещественный	Простой	–	Шаг между узлами оптимизации
x_1	x1	Вещественный	Простой	–	Аргумент функции
x_2	x2	Вещественный	Простой	–	Аргумент функции
f_1	f1	Вещественный	Простой	–	Значение функции
f_2	f2	Вещественный	Простой	–	Значение функции
x_{opt}	xopt	Вещественный	Простой	–	Оптимальный аргумент функции
y_{opt}	y	Вещественный	Простой	–	Оптимальное значение функции

4. Алгоритм

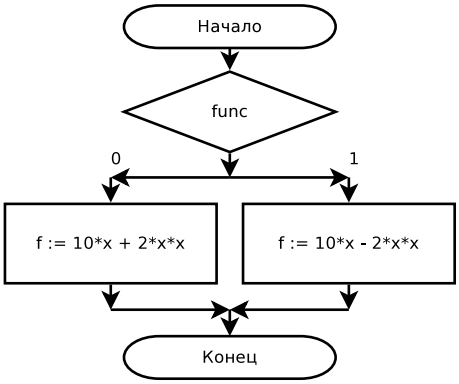
4.1. Процедура «bitwise_approx»



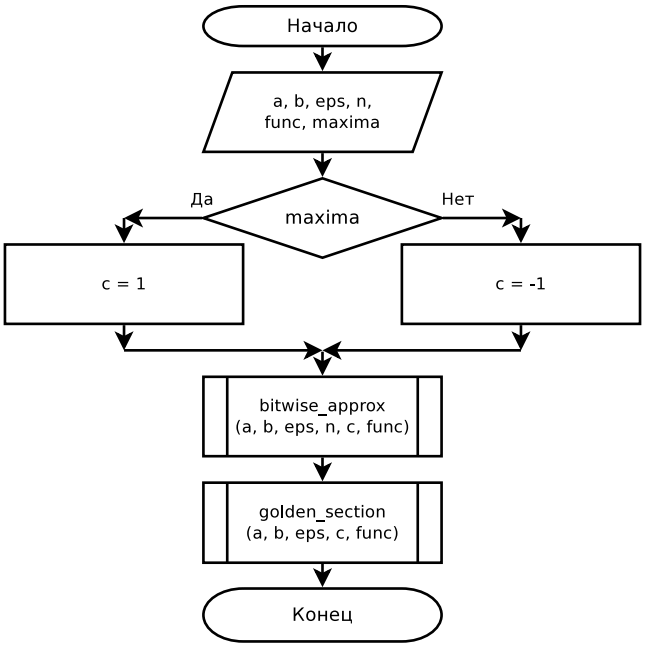
4.2. Процедура «golden_section»



4.3. Функция «$f(x)$»



4.4. Функция «main»



5. Программа

```
#include <iostream>
#include <cmath>

using namespace std;

const double FI = 1.6180339887;  // (sqrt(5) + 1) / 2

void input(double & a, double & b, double & eps, int & n,
           int & func, bool & maxima)
{
```

```
    cout << "Input a = "; cin >> a;
    cout << "Input b = "; cin >> b;
    cout << "Input eps = "; cin >> eps;
    cout << "Input n = "; cin >> n;
    cout << "Function (0: y=10x+2x^2, 1: y=10x-2x^2) = "; cin >> func;
    cout << "Extremum type (0: minima, 1: maxima) = "; cin >> maxima;
}

void print(const char *method, const double & xopt, const double & y,
           const double & eps, int i)
{
    cout << method
         << "xopt = " << xopt
         << ", f(xopt) = " << y
         << ", eps = " << eps
         << ", i = " << i
         << endl;
}

double f(const double & x, int func)
{
    switch (func) {
        case 0:
            return 10*x + 2*x*x;
        case 1:
            return 10*x - 2*x*x;
    }
}

void bitwise_approx(double a, double b, const double & eps,
                    int n, int c, int func)
{
    double h = (b - a) / n, x1 = a, x2, f1, f2;
    int i = 0;

    do {
        f1 = f(x1, func);
        x2 = x1 + h;
        f2 = f(x2, func);
```

```
        if (f1*c < f2*c) {
            x1 += h;
            if (x1 > b) {
                cout << "There is no extrema" << endl;
                return;
            }
        } else {
            a = x2 - 2*h;
            b = x2;
            h /= n;
            x1 = a;
        }
        ++i;
    } while (abs(f1 - f2) > eps);

    double xopt = (a + b) / 2;
    print("Bitwise approximation: ", xopt, f(xopt, func), eps, i);
}

void golden_section(double a, double b, const double & eps, int c, int func)
{
    double x1, x2, f1, f2;
    int i = 0;

    x1 = b - (b - a) / FI;
    x2 = a + (b - a) / FI;
    f1 = f(x1, func); f2 = f(x2, func);
    do {
        if (f1*c > f2*c) {
            b = x2;
            x2 = x1;
            x1 = b - (b - a) / FI;
            f2 = f1;
            f1 = f(x1, func);
        } else {
            a = x1;
            x1 = x2;
            x2 = a + (b - a) / FI;
            f1 = f2;
```

```
            f2 = f(x2, func);
        }
        ++i;
    } while (abs(b - a) > eps);

    double xopt = (a + b) / 2;
    print("Golden section method: ", xopt, f(xopt, func), eps, i);
}

int main()
{
    double a, b, eps; int n, func; bool maxima;
    input(a, b, eps, n, func, maxima);
    int c = maxima ? 1 : -1;
    bitwise_approx(a, b, eps, n, c, func);
    golden_section(a, b, eps, c, func);
    return 0;
}
```

6. Шаблон ввода исходных данных

```
-100
100
0.0001
400
0
0
```

7. Шаблон вывода результата

```
Bitwise approximation: xopt = -2.5, f(xopt) = -12.5, eps = 0.0001, i = 581
Golden section method: xopt = -2.50001, f(xopt) = -12.5, eps = 0.0001, i = 31
```

8. Вывод

Исходя из проделанной работы можно сделать вывод, что метод золотого сечения превосходит по сходимости метод поразрядного приближения.