Цель работы: знакомство с оптимизационными задачами, изучение различных

методов одномерной оптимизации и сравнение эффективности их применения

для конкретных целевых функций.

**Задача №22 (Золотое сечение)**

Цель: Какой из прямоугольников с периметром 80 см. имеет наибольшую площадь? Вычислите площадь этого прямоугольника.

Методами: Золотое сечение;

Для решения задачи методом золотого сечения мы будем искать максимальное значение функции площади S = xy при условии, что периметр P = 2x+2y равен 80 и сумма длин сторон x+y равна 40.

Для начала определим, какую из переменных мы будем изменять при поиске максимального значения функции S. Поскольку y зависит от x по формуле y=40-x, мы будем изменять только переменную x.

Используя формулу периметра P=2x+2y, выразим y через x:  
2x + 2y = 80  
2y = 80 - 2x  
y = 40 - x

Теперь выразим площадь S через x:  
S = xy = x(40 - x) = 40x - x^2

Зададим начальный интервал поиска значения x. Так как сумма сторон x+y равна 40, то x не может превышать 20. Будем искать максимальное значение функции S на интервале от 0 до 20.

#include <iostream>

#include <cmath>

using namespace std;

const double phi = (1 + sqrt(5)) / 2; // Золотое сечение ~1.618

const double eps = 1e-4; // Точность вычислений

double S(double x) {

return 40 \* x - x \* x; // Функция площади

}

int main() {

double a = 0, b = 20; // Начальный интервал поиска

double x1 = b - (b - a) / phi; // Вычисляем первую точку деления

double x2 = a + (b - a) / phi; // Вычисляем вторую точку деления

double y1 = S(x1);

double y2 = S(x2);

// Пока не достигнута нужная точность, продолжаем делить интервал

while (abs(b - a) > eps) {

if (y1 > y2) { // Если значение функции в первой точке больше, то новый интервал [a, x2]

b = x2;

x2 = x1;

y2 = y1;

x1 = b - (b - a) / phi;

y1 = S(x1);

}

else { // Иначе новый интервал [x1, b]

a = x1;

x1 = x2;

y1 = y2;

x2 = a + (b - a) / phi;

y2 = S(x2);

}

}

// Выводим максимальное значение функции и соответствующее значение x

cout << "x = " << (a + b) / 2 << endl;

cout << "S = " << S((a + b) / 2) << endl;

return 0;

}  
  
После запуска программы мы получим ответ:

x = 20

S = 400

Таким образом, прямоугольник со сторонами 20 см и 20 см имеет наибольшую площадь, равную 400 кв. см.

С обьяснением:

#include <iostream>

#include <cmath>

using namespace std;

const double phi = (1 + sqrt(5)) / 2; // Золотое сечение, значение примерно равно 1.618

const double eps = 1e-4; // Это значение точности вычислений, которую мы стремимся достичь

// Функция для вычисления площади. Значение x будет варьироваться при каждом вызове функции.

double S(double x) {

return 40 \* x - x \* x;

}

int main() {

double a = 0, b = 20; // Это наши начальные границы интервала поиска - [a,b]

// Мы вычисляем точки деления интервала в соответствии с принципом золотого сечения.

double x1 = b - (b - a) / phi; // Вычисляем первую точку деления

double x2 = a + (b - a) / phi; // Вычисляем вторую точку деления

// Вычисляем значения функции в этих точках

double y1 = S(x1);

double y2 = S(x2);

// Пока абсолютное значение разности концов интервала больше требуемой точности,

// продолжаем наш процесс разделения интервала, по сути выполняя алгоритм золотого сечения

while (abs(b - a) > eps) {

// Если значение функции в первой точке больше, то уменьшаем интервал до [a, x2]

if (y1 > y2) {

b = x2;

x2 = x1;

y2 = y1;

x1 = b - (b - a) / phi;

y1 = S(x1);

}

else { // В противном случае уменьшаем интервал до [x1, b]

a = x1;

x1 = x2;

y1 = y2;

x2 = a + (b - a) / phi;

y2 = S(x2);

}

}

// В конце мы выводим среднее значение поискового интервала (точку в которой достигается макс./мин.),

// а также значение функции в этой точке (т.е. максимальное/минимальное значение функции).

cout << "x = " << (a + b) / 2 << endl;

cout << "S = " << S((a + b) / 2) << endl;

return 0;

}

**Задача №22 (Метод деление интервала пополам)**

Для решения этой задачи методом деления интервала пополам в C++, нам нужно сначала понять, как связаны периметр и площадь прямоугольника. Затем мы будем итеративно приближаться к прямоугольнику с наибольшей площадью, изменяя его стороны.

1. Определите формулу для площади прямоугольника в зависимости от его сторон: Пусть длина прямоугольника будет x см, а ширина будет (80 - 2x) см (поскольку периметр равен 80 см и состоит из двух длин и двух ширин). Площадь S прямоугольника равна S = x \* (80 - 2x).
2. Найдите производную S по x и приравняйте ее к нулю, чтобы найти точку экстремума. В данном случае, это будет точка максимума.
3. Решите уравнение для x. Найденное значение x будет являться длиной прямоугольника, а ширина будет (80 - 2x).
4. Рассчитайте площадь S для найденных значений x и (80 - 2x).
5. Сравните площади всех найденных прямоугольников и выберите прямоугольник с наибольшей площадью.

Вот C++ код, который реализует этот алгоритм:

#include <iostream>

#include <cmath>

double findMaxRectangleArea() {

double left = 0.0;

double right = 40.0; // Максимальная длина прямоугольника, так как периметр равен 80

while (fabs(right - left) > 1e-6) {

double x1 = (left \* 2 + right) / 3.0;

double x2 = (left + right \* 2) / 3.0;

double area1 = x1 \* (80 - 2 \* x1);

double area2 = x2 \* (80 - 2 \* x2);

if (area1 < area2) {

left = x1;

} else {

right = x2;

}

}

// Найденная длина x будет длиной прямоугольника с наибольшей площадью

double length = (left + right) / 2.0;

double width = 80 - 2 \* length;

double maxArea = length \* width;

return maxArea;

}

int main() {

double maxArea = findMaxRectangleArea();

std::cout << "Наибольшая площадь прямоугольника: " << maxArea << " кв. см." << std::endl;

return 0;

}

Этот код использует метод деления интервала пополам для поиска длины прямоугольника, который имеет наибольшую площадь среди всех прямоугольников с периметром 80 см. Результат будет выведен на экран.