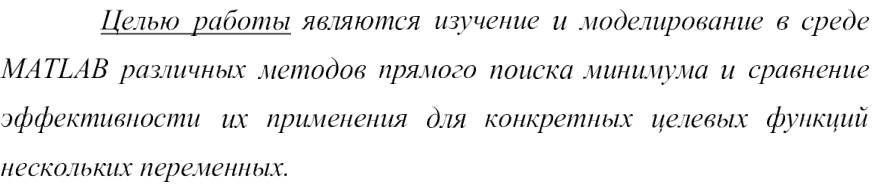
**ИССЛЕДОВАНИЕ МЕТОДОВ БЕЗУСЛОВНОЙ КОНЕЧНОМЕРНОЙ ОПТИМИЗАЦИИ**

Лабораторная работа № 2.

**МЕТОДЫ ПРЯМОГО ПОИСКА ЭКСТРЕМУМА ДЛЯ ФУНКЦИЙ МНОГИХ ПЕРЕМЕННЫХ**

Вариант № 9.

Отчет выполнили: Фролов Никита, Учаев Сергей, Булыгин Василий



Всего будет рассмотрено 4 метода:

1. Метод покоординатного спуска (метод Гаусса-Зайделя);
2. Симплексный метод;
3. Метод Нелдера-Мида;
4. Метод Хука-Дживса.

Все рассмотренные методы реализовано на Mathlab 2022.

Начальные условия:



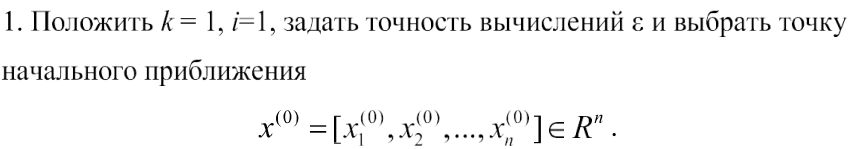


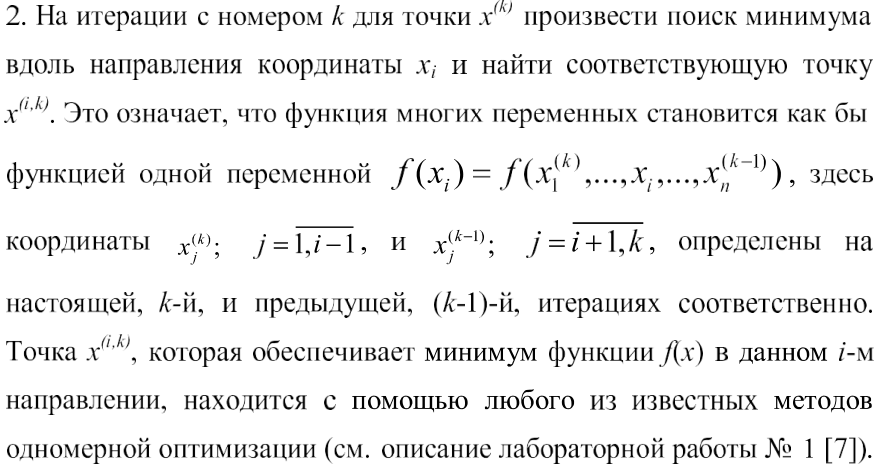
Каждый метод рассматривается при четырёх разных точностях.

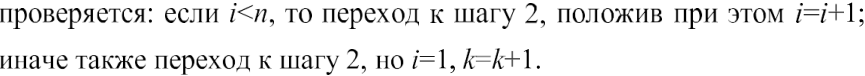
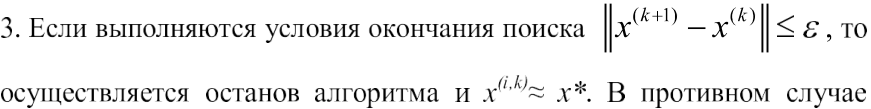
По полученным данным из программы каждому методу будет строиться график количества итераций - точность.

Метод покоординатного спуска (метод Гаусса-Зайделя).

Алгоритм:







Программный код:

Файл TaskOne.m (начальные условия + запуск метода):

clear all % Очищаем всю область и переменные

E1 = 10^(-5); % Точность № 1

E2 = 10^(-7); % Точность № 2

E3 = 10^(-9); % Точность № 3

E4 = 10^(-12); % Точность № 4

x = zeros(100, 4); % Массив для точек решений

x0 = [0, 0, 0, 0]; % Начальное приближение

n = 4; % Количество координат для нашей функции

fun = @(x) (x(1) - x(2))^2 + 4 \* (x(3) - x(4))^2 + (x(2) - 6 \* x(4))^4 + 2 \* (x(1) - x(3))^2 % Наша фунция по условиям

xmin = [0, 0, 0, 0]; % Точка минимума

fxmin = 0; % Значение функции в точки минимума

[xmin, fxmin, k] = MethodOne(fun, x0, x, E1, n, xmin, fxmin); % Вызов функции

Файл MethodOne.m (кода метода):

function [xmin, fxmin, k] = MethodOne(fun, x0, x, E1, n, xmin, fxmin)

k = 0; % Начальные значения (начинаем с первой итерации)

i = 1; % Начинаем с первой координаты

while True

[x(k, i), fxmin] = fibona(fun, x, i, k); % Поиск

% минимума по одной

% переменной происходит методом фибоначчи из первой Л. Р.

if abs(x(k) - x(k - 1)) <= E1 % Проверка останова

xmin = x(k);

break;

else

if i < n

i = i + 1;

else

i = 1;

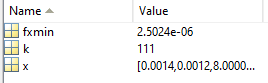
k = k + 1;

end

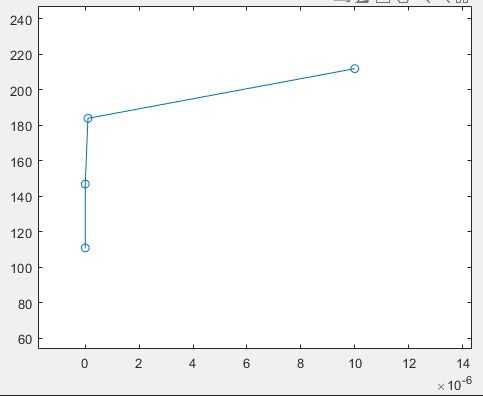
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

E = [10^(-12) 10^(-9) 10^(-7) 10^(-5)]

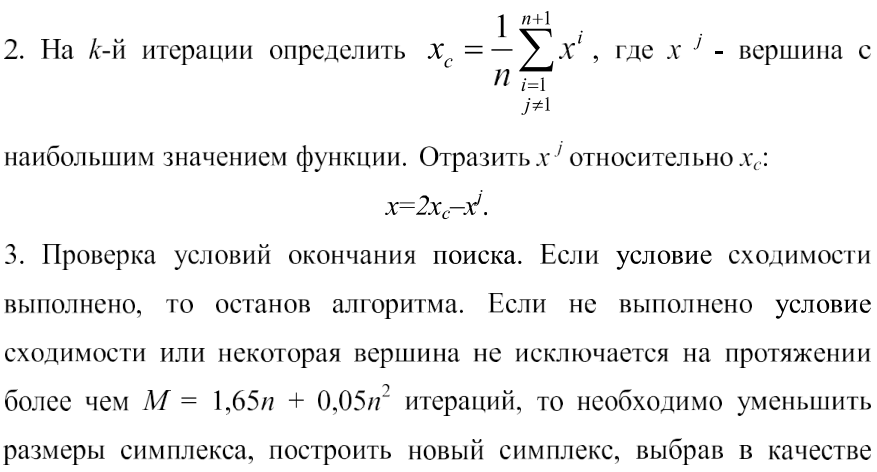
K = [111 147 184 212]

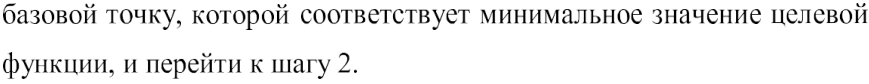
plot(E, K,'o-');

Симплексный метод.

Алгоритм:







Программный код:

Файл TaskTwo.m (начальные условия + запуск метода):

clear all % Очищаем всю область и переменные

E1 = 10^(-5); % Точность № 1

E2 = 10^(-7); % Точность № 2

E3 = 10^(-9); % Точность № 3

E4 = 10^(-12); % Точность № 4

x = zeros(100, 4); % Массив для точек решений

x0 = [0, 0, 0, 0]; % Начальное приближение

n = 4; % Количество координат для нашей функции

fun = @(x) (x(1) - x(2))^2 + 4 \* (x(3) - x(4))^2 + (x(2) - 6 \* x(4))^4 + 2 \* (x(1) - x(3))^2 % Наша фунция по условиям

xmin = [0, 0, 0, 0]; % Точка минимума

fxmin = 0; % Значение функции в точки минимума

[xmin, fxmin, k] = MethodTwo(fun, x0, x, E1, n, xmin, fxmin); % Вызов функции

Файл MethodTwo.m (кода метода):

function [xmin, fxmin, k] = MethodTwo(fun, x0, x, E1, n, xmin, fxmin)

x(0) = x0;

a = 1; % Масштабный множитель

d1 = (((n + 1)^0.5 + n - 1) / n \* sqrt(2)) \* a;

d2 = (((n + 1)^0.5 - 1) / n \* sqrt(2)) \* a;

k = 0; % Счётчик итераций

j = 0; % Временный индекс

M = 1,65 \* n + 0.05 \* n^2; % Ограничения по количеству итераций

while True

for i = 1: n

if (j == i)

x(i) = x(0, j) + d2;

else

x(i) = x(0, j) + d1;

end

end

for i = 1: n + 1

xc = xc + x(i) / n;

end

x (k) = 2\*xc - x(j);

j = j + 1;

if (x(k) - x(k -1 ) < E1)

xmin = x(k);

fxmin = fun(x(k));

break;

end

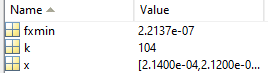
if k > M % Уменьшаем размеры симплекс матрицы

n = n - 1;

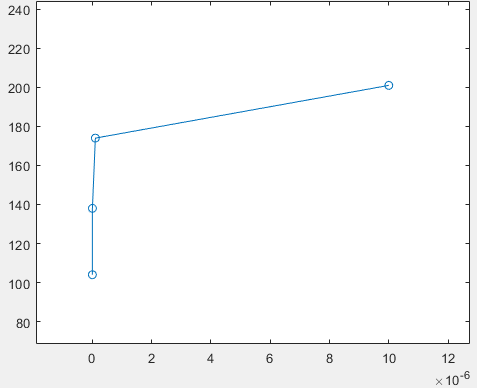
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

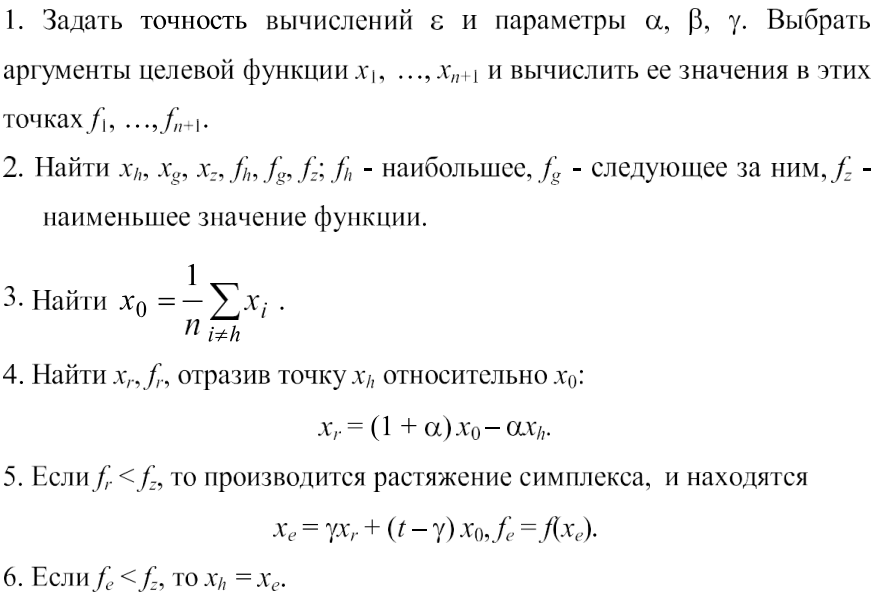
E = [10^(-12) 10^(-9) 10^(-7) 10^(-5)]

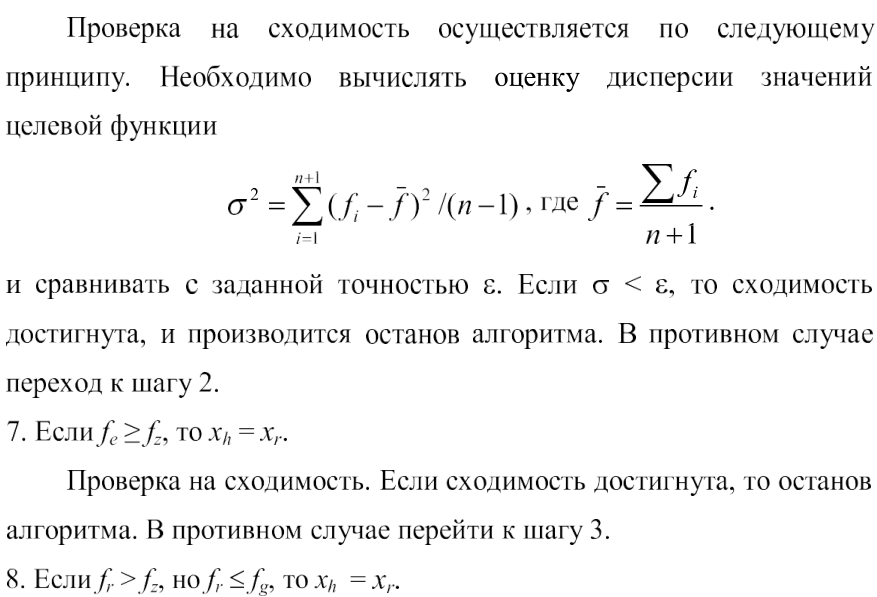
K = [104 138 174 201]

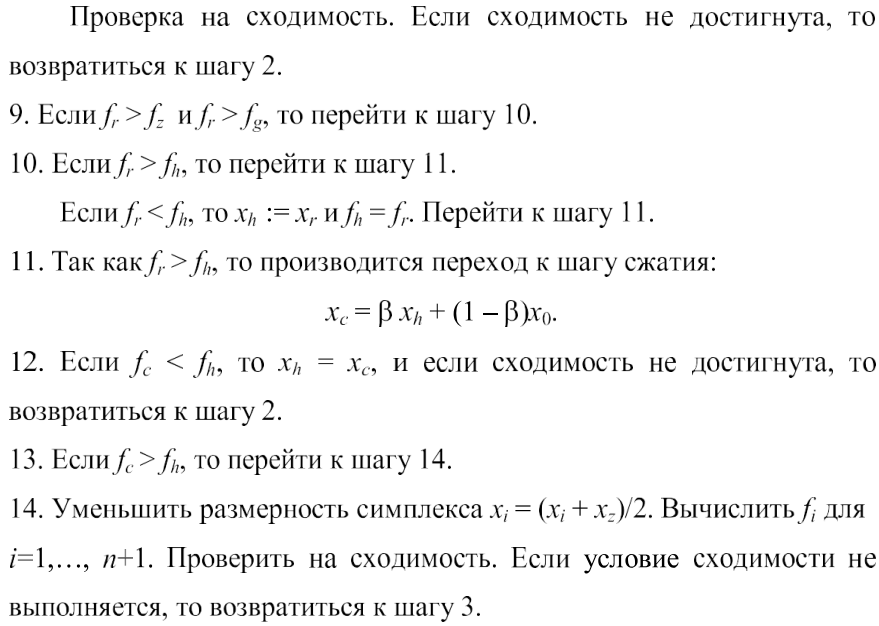
plot(E, K,'o-');

Метод Нелдера-Мида.

Алгоритм:







Программный код:

clear all % Очищаем всю область и переменные

E1 = 10^(-5); % Точность № 1

E2 = 10^(-7); % Точность № 2

E3 = 10^(-9); % Точность № 3

E4 = 10^(-12); % Точность № 4

x = zeros(100, 4); % Массив для точек решений

x0 = [0, 0, 0, 0]; % Начальное приближение

n = 4; % Количество координат для нашей функции

fun = @(x) (x(1) - x(2))^2 + 4 \* (x(3) - x(4))^2 + (x(2) - 6 \* x(4))^4 + 2 \* (x(1) - x(3))^2 % Наша фунция по условиям

xmin = [0, 0, 0, 0]; % Точка минимума

fxmin = 0; % Значение функции в точки минимума

[xmin, fxmin, k] = MethodThree(fun, x0, x, E1, n, xmin, fxmin); % Вызов функции

Файл MethodThree.m (кода метода):

function [xmin, fxmin, k] = MethodThree(fun, x0, x, E1, n, xmin, fxmin)

a = 1; % Необходимый параметр

b = 1; % Необходимый параметр

y = 1; % Необходимый параметр

fx = zeros(100, 4); % Массив для значений функций

while true

for i = 1: n + 1

fx(i) = fun(x(0));

end

xmax = max(x);

xmed = max(x - i);

xmin = min(x);

fmax = fun(xmax);

fmed = fun(xmed);

fmin = fun(xmin);

for i = 1: n

x0 = x0 + 1 / n \* (i);

end

xr = (1 + a) \* x0 - a \* xmax;

if fun(xr) < fun(xmed)

xe = y \* xr + (t - y) \* x0;

fe = fun(xe);

end

if fe < fun(xmed)

xmax = xe;

end

for i = 1: n + 1

f(x) = f(i) / (n + 1);

summ = ((f(i) - f(x))^2) / (n - 1);

end

q = sqrt(summ);

if fe >= fmid

xmax = xr;

end

if fun(xr) > fun(xmed)

xmax = xr;

end

if q < E1

xmin = x(i);

fxmin = fun(xmin);

break;

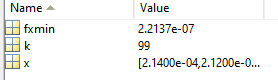
else

k = k + 1;

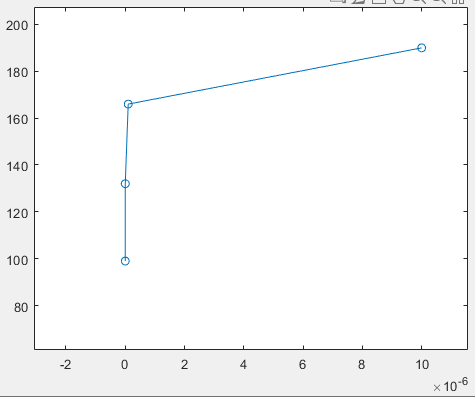
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

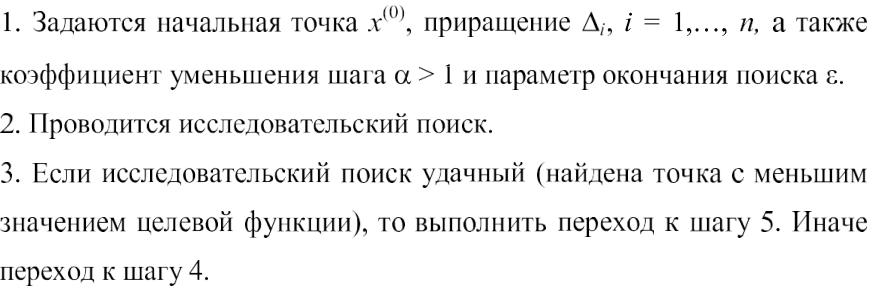
E = [10^(-12) 10^(-9) 10^(-7) 10^(-5)]

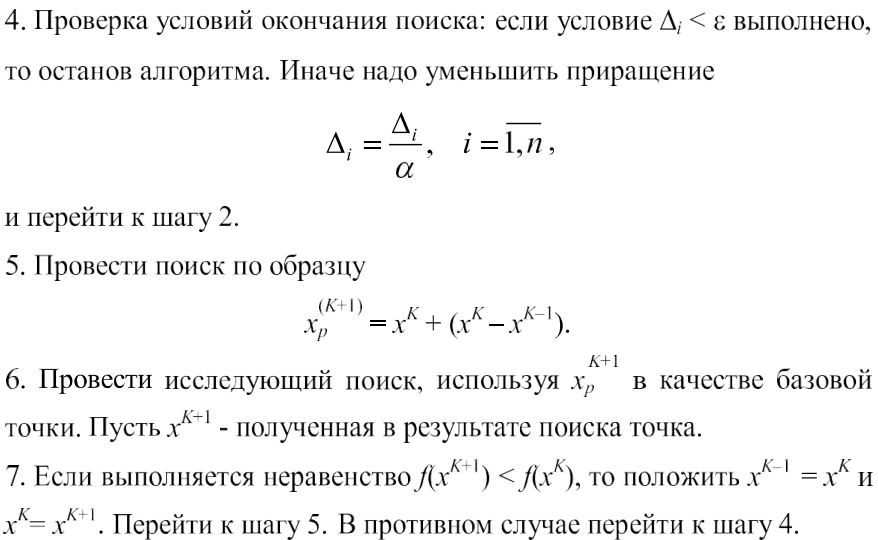
K = [99 132 166 190]

plot(E, K,'o-');

Метод Хука-Дживса.

Алгоритм:





Код программы:

Файл TaskFour.m (начальные условия + запуск метода):

clear all % Очищаем всю область и переменные

E1 = 10^(-5); % Точность № 1

E2 = 10^(-7); % Точность № 2

E3 = 10^(-9); % Точность № 3

E4 = 10^(-12); % Точность № 4

x = zeros(100, 4); % Массив для точек решений

x0 = [0, 0, 0, 0]; % Начальное приближение

n = 4; % Количество координат для нашей функции

fun = @(x) (x(1) - x(2))^2 + 4 \* (x(3) - x(4))^2 + (x(2) - 6 \* x(4))^4 + 2 \* (x(1) - x(3))^2; % Наша фунция по условиям

xmin = [0, 0, 0, 0]; % Точка минимума

fxmin = 0; % Значение функции в точки минимума

[xmin, fxmin, k] = MethodFour(fun, x0, x, E1, n, xmin, fxmin); % Вызов функции

Файл MethodFour.m (кода метода):

function [xmin, fxmin, k] = MethodFour(fun, x0, x, E1, n, xmin, fxmin)

i = 1;

k = 1;

delta(i) = 1; % Приращение

a = 2; % Коэффициент уменьшения шага

while True

x(k) = fminsearch(fun, x(k - 1));

i = i + 1;

x(k + 1) = x(k)+ (x(k) - x(k - 1));

while fun(x(k + 1)) < fun(x(k))

x(k + 2) = fminsearch(fun, x(k + 1));

end

if delta < E1

xmin = x(k - 1);

fxmin = fun(x(k - 1));

break;

else

delta = delta / a;

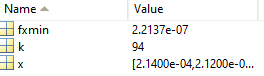
i = 1;

k = k + 1;

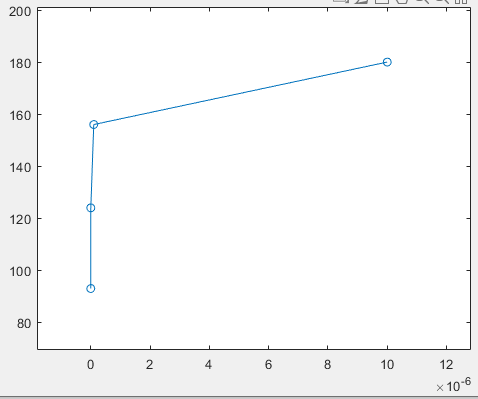
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код графика:

clear all

E = [10^(-12) 10^(-9) 10^(-7) 10^(-5)]

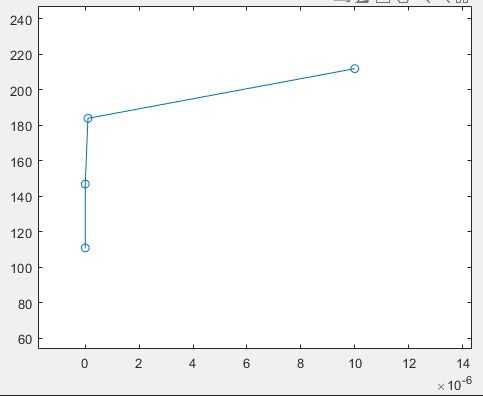
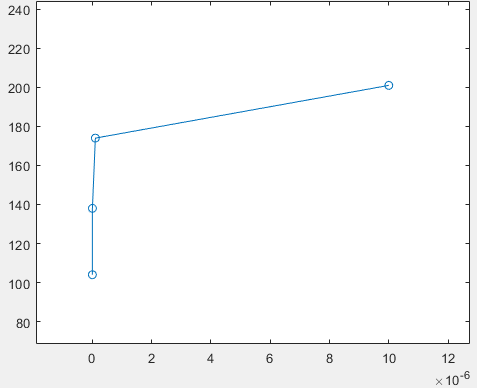
K = [93 124 156 180]

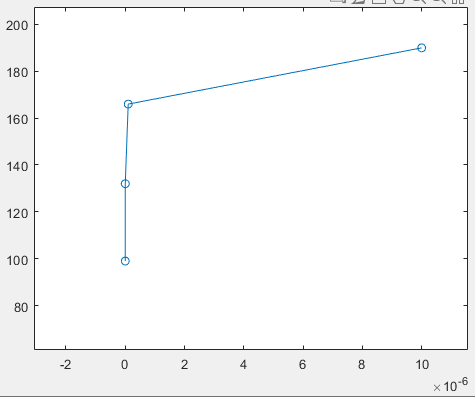
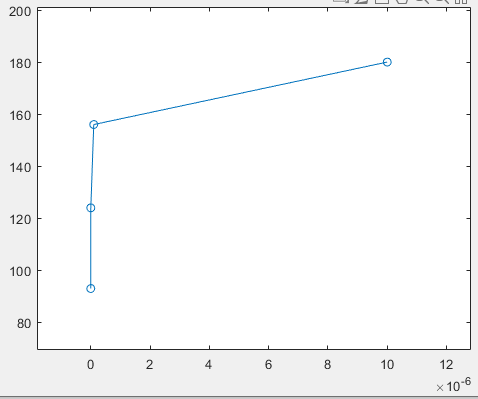
plot(E, K,'o-');

Сравнение методов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер метода | E1 = 10^(-3)  K: | E2 = 10^(-5)  K: | E1 = 10^(-9)  K: | E1 = 10^(-12)  K: |
| 1 | 111 | 147 | 184 | 212 |
| 2 | 104 | 138 | 174 | 201 |
| 3 | 99 | 132 | 166 | 190 |
| 4 | 93 | 124 | 156 | 180 |

Графическое сравнение:

№ 1) № 2) 

№ 3) № 4) 

Выводы:

В ходе лабораторной работы были освоены 4 метода прямого поиска экстремума для функций многих переменных: метод покоординатного спуска (метод Гаусса-Зайделя), cимплексный метод, метод Нелдера-Мида, метод Хука-Дживса.

В результате проведения лабораторной работы метод Хука-Дживса оказался самым эффективным, о чём свидетельствует таблица итераций.