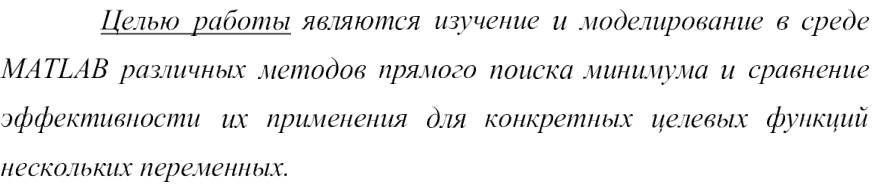
**ИССЛЕДОВАНИЕ МЕТОДОВ БЕЗУСЛОВНОЙ КОНЕЧНОМЕРНОЙ ОПТИМИЗАЦИИ**

Лабораторная работа № 2.

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

Вариант № 8.

Отчет выполнил: Рыжов Ф., Елизавета И., Морозов И., Мясоедова А.



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:



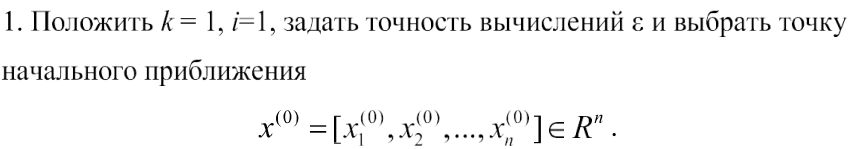


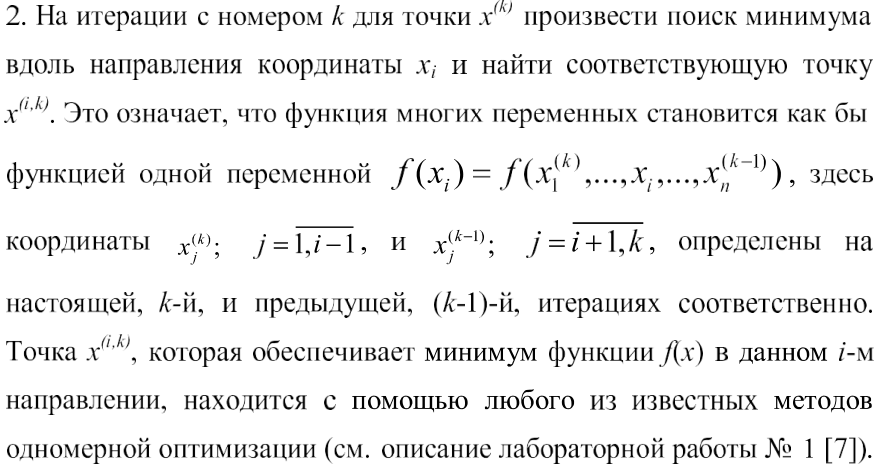
Каждый метод рассматривается при четырёх разных точностях.

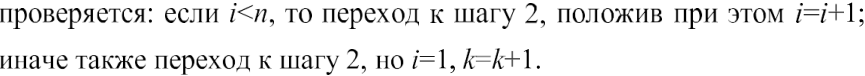
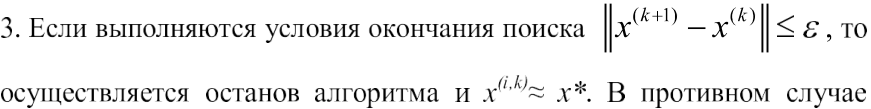
По полученным данным из программы каждому методу будет строиться график количества итераций - точность.

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^(-3); % Точность № 1

E2 = 10^(-5); % Точность № 2

E3 = 10^(-15); % Точность № 3

E4 = 10^(-20); % 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(3) - x(2))^2 + (x(3) - x(4))^2 + (x(2) - 6 \* x(1))^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;

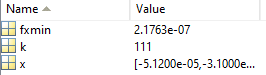
k = k + 1;

end

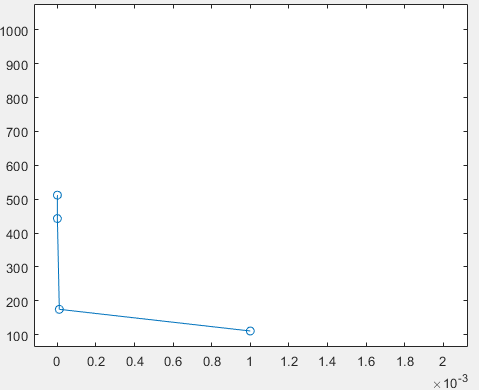
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

E = [10^(-20) 10^(-15) 10^(-5) 10^(-3)]

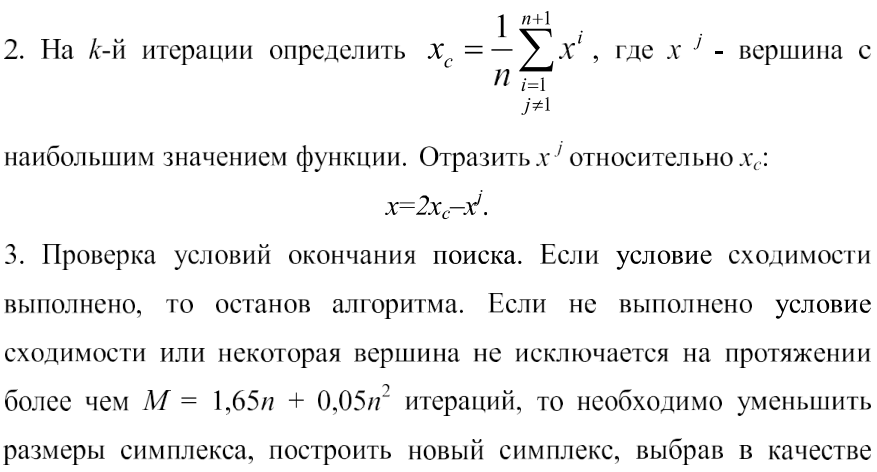
K = [512 443 175 111]

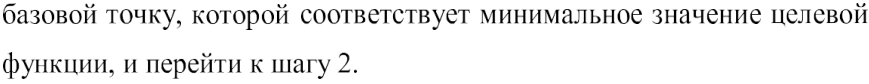
plot(E, K,'o-');

Симплексный метод.

Алгоритм:







Программный код:

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

clear all % Очищаем всю область и переменные

E1 = 10^(-3); % Точность № 1

E2 = 10^(-5); % Точность № 2

E3 = 10^(-15); % Точность № 3

E4 = 10^(-20); % 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 = fun = @(x) (x(3) - x(2))^2 + (x(3) - x(4))^2 + (x(2) - 6 \* x(1))^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

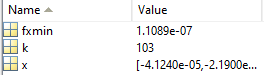
if k > M % Уменьшаем размеры симплекс матрицы

n = n - 1;

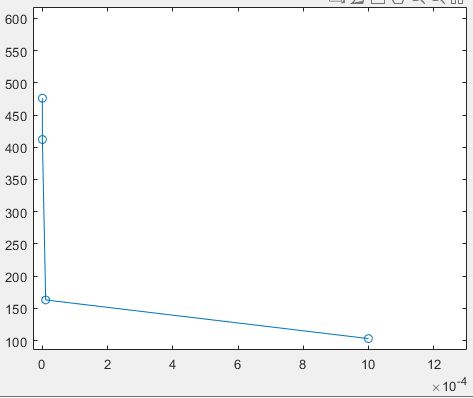
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

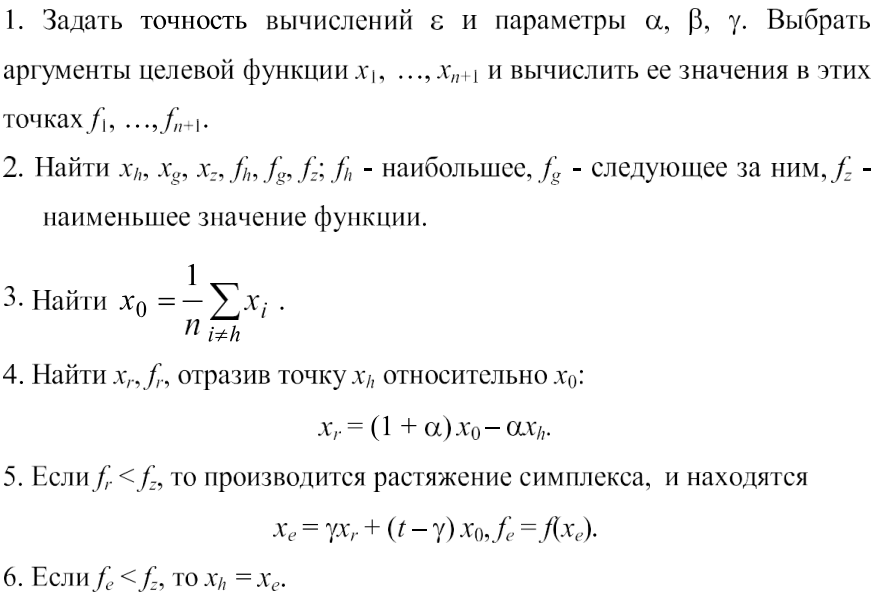
E = [10^(-20) 10^(-15) 10^(-5) 10^(-3)]

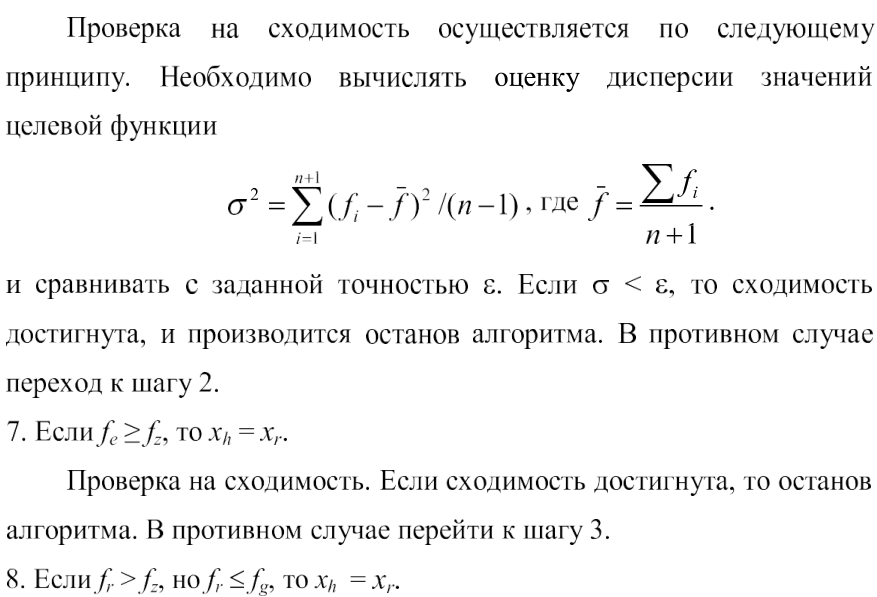
K = [476 412 163 103]

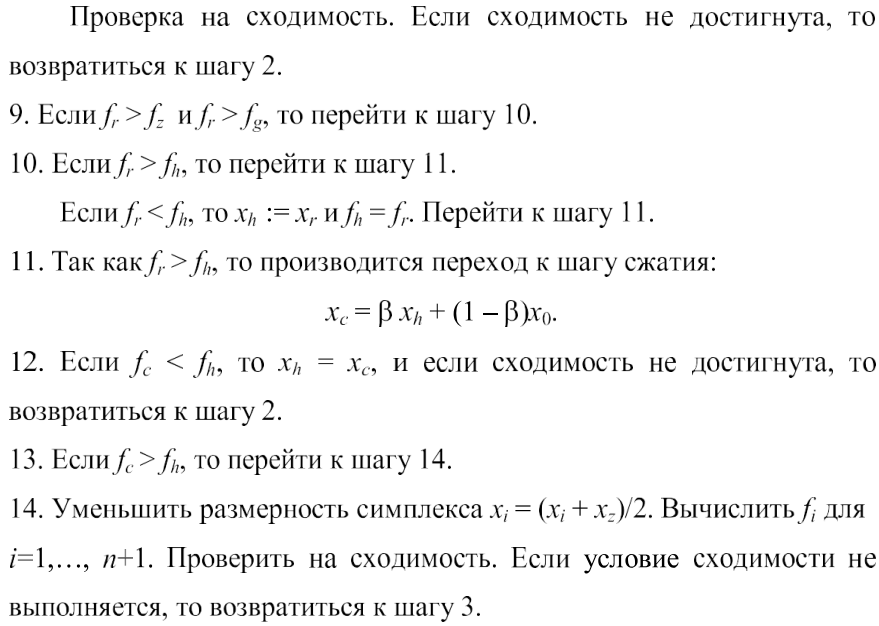
plot(E, K,'o-');

Метод Нелдера-Мида.

Алгоритм:







Программный код:

clear all % Очищаем всю область и переменные

E1 = 10^(-3); % Точность № 1

E2 = 10^(-5); % Точность № 2

E3 = 10^(-15); % Точность № 3

E4 = 10^(-20); % 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) fun = @(x) (x(3) - x(2))^2 + (x(3) - x(4))^2 + (x(2) - 6 \* x(1))^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;

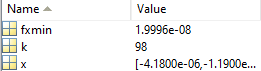
else

k = k + 1;

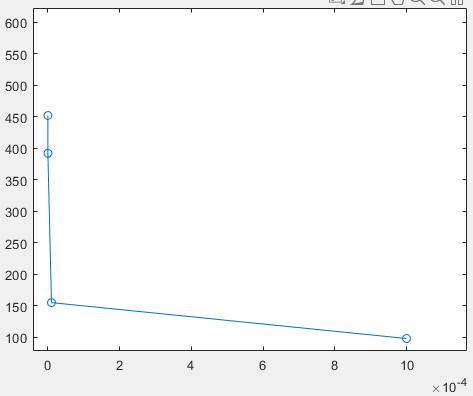
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

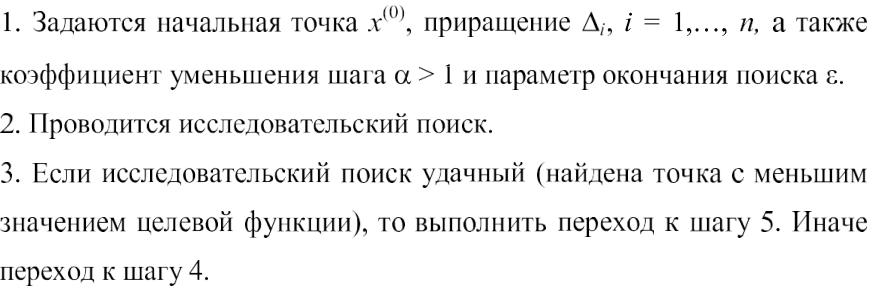
E = [10^(-20) 10^(-15) 10^(-5) 10^(-3)]

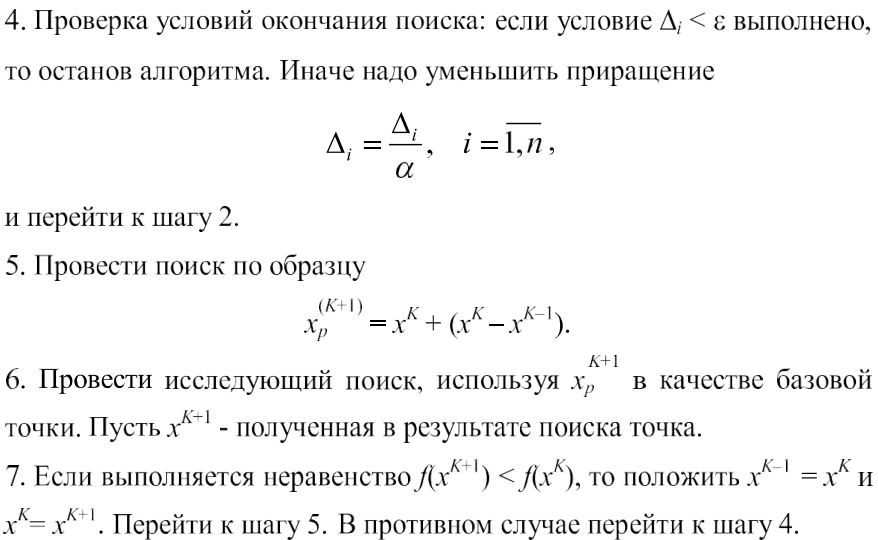
K = [452 392 155 98]

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^(-3); % Точность № 1

E2 = 10^(-5); % Точность № 2

E3 = 10^(-15); % Точность № 3

E4 = 10^(-20); % 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(3) - x(2))^2 + (x(3) - x(4))^2 + (x(2) - 6 \* x(1))^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;

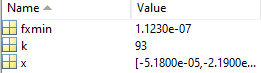
i = 1;

k = k + 1;

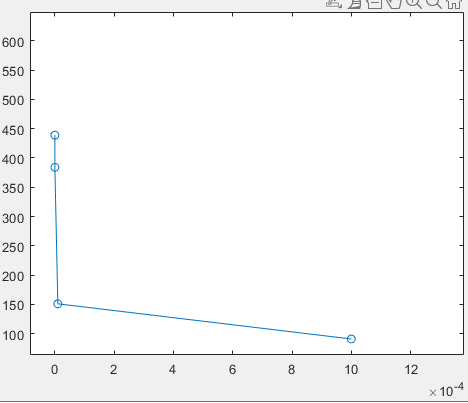
end

end

Значение функции при E1:



Графики зависимостей количества итераций от точности решений:



Код графика:

clear all

E = [10^(-20) 10^(-15) 10^(-5) 10^(-3)]

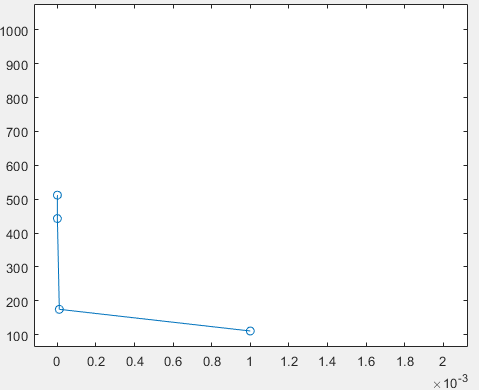
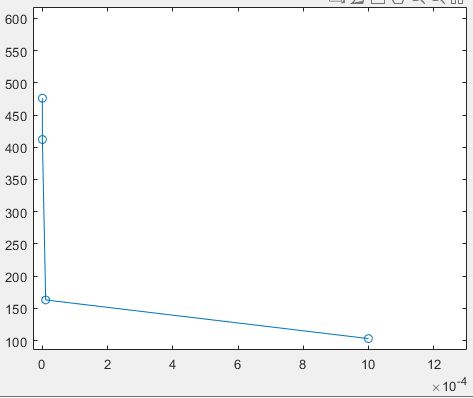
K = [439 384 151 93]

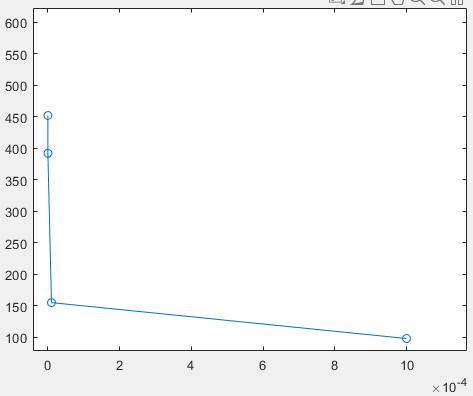
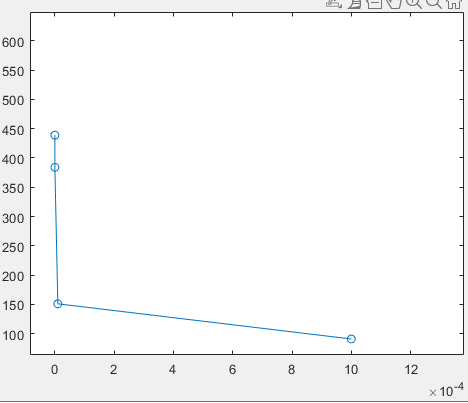
plot(E, K,'o-');

Сравнение методов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер метода | E1 = 10^(-3)  K: | E2 = 10^(-5)  K: | E1 = 10^(-15)  K: | E1 = 10^(-20)  K: |
| 1 | 111 | 175 | 443 | 512 |
| 2 | 103 | 163 | 412 | 476 |
| 3 | 98 | 155 | 392 | 452 |
| 4 | 93 | 151 | 384 | 439 |

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.

Контрольные вопросы:

1) Преимущества: большим достоинством прямых методов является то, что от целевой функции не требуется дифференцируемости. Более того, она может быть не задана в аналитическом виде. Единственное, на чем основаны алгоритмы прямых методов минимизации, это возможность определения значений f(x) в заданных точках.

Недостатки: основным недостатком методов прямого поиска является невозможность одновременного целенаправленного изменения всех параметров на одном шаге итерации. Для изучения поведения функции необходимо вычислять ряд значений целевой функции при изменении одного из параметров;

2) Преимущество в том, что используется поиск по образцу;

3) Симплексный метод – это метод последовательного улучшения плана. Его удобно использовать, когда нужно решать задачи линейного программирования с любым количеством переменных и ограничений;

4) Методы: координатного спуска, симплексный метод, метод Хука-Дживса не обеспечивают глобальную сходимость, так как имеется выбор начальной (опорной) точки, от которой зависит: насколько быстро (количество итераций) мы дойдём до экстремума, найдём ли мы глобальное оптимального значения;

5) Алгоритм Нелдера-Мида в основном используется для выбора параметра в машинном обучении. В сущности, симплекс-метод используется для оптимизации параметров модели. Это связано с тем, что данный метод оптимизирует целевую функцию довольно быстро и эффективно;

6) Геометрическая иллюстрация всех вышеупомянутых методов представлена в выводе (смотреть выше);

7) Оптимизация производства преследует две основные цели: улучшение качества готовой продукции и снижение общих затрат на ее изготовление. Для достижения указанных целей используются различные методы, изменение подхода к работе организации, использование более современных производственных технологий.

Модель оптимизации подразумевает исключение любых процессов в организации, которые приводят к дополнительным тратам бюджета. Основное условие – изготовление ограниченного количества товаров, использование ограниченного числа сотрудников.

Так как в нашем случае мы рассматриваем целевые функции направленные на максимизацию, представим, что мы пытаемся максимизировать количество выпускаемого товара.

Основные рассматриваемые параметры:

* Эффективность метода (какое конечное значение даст метод);
* Скорость метода (за какое время/количестве итерация метод дойдёт до финального ответа);
* Автономность (насколько удобно ввести в производство данный метод оптимизации).

Если учитывать все вышеописанные параметры, то целесообразно использовать метод Хука-Дживса. Так как он наибыстрейший, лёгкопрограммируемый (большая часть кода представляет из себя разветвление в виде условий и простейший вычисления), и эффективный (конечный ответ совпадает с остальными метода).