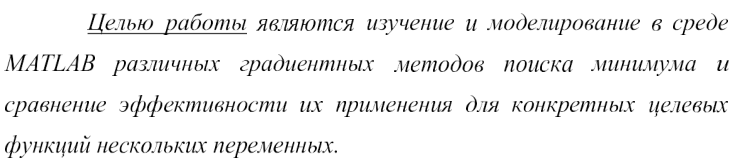
**ИССЛЕДОВАНИЕ МЕТОДОВ БЕЗУСЛОВНОЙ КОНЕЧНОМЕРНОЙ ОПТИМИЗАЦИИ**

Лабораторная работа № 3.

**ЧИСЛЕННЫЕ МЕТОДЫ ПОИСКА ЭКСТРЕМУМА ДЛЯ ДИФФЕРЕНЦИРУЕМЫХ ФУНКЦИЙ МНОГИХ ПЕРЕМЕННЫХ**

Вариант № 6.

Отчет выполнили: Асадов Н., Корнаухов Е., Кариев С.



Всего будет рассмотрено 7 метода, один из которых имеет 3 подвида:

1. Метод градиента с постоянным (заданным) шагом;
2. Метод наискорейшего спуска;
3. Метод сопряжённого градиента для квадратичных функций;
4. Метод Фленча-Ривза;
5. Метод Ньютона (модифицированный метод Ньютона, метод Марквардта);
6. Метод Давида-Флетчера-Пауэлла;
7. Метод Бройдена-Флетчера-Гольдфарба-Шанно.

Все рассмотренные методы реализовано на Mathlab 2022.

Начальные условия:



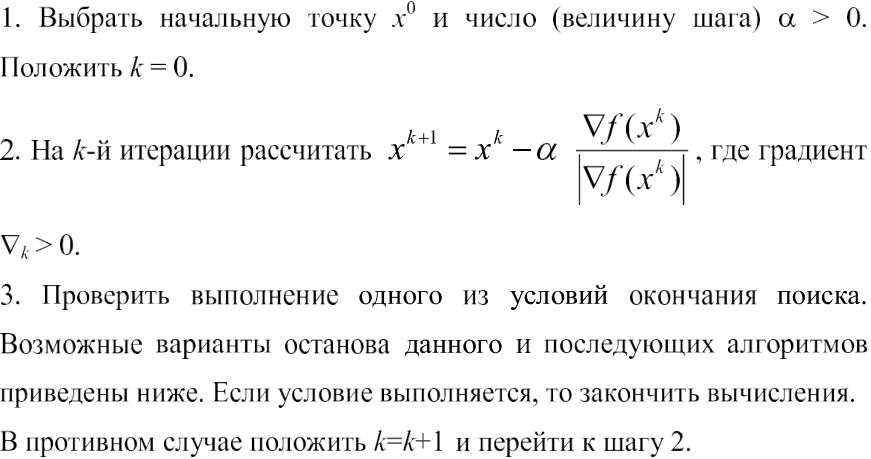


Каждый метод рассматривается при четырёх разных точностях.

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

Метод градиента с постоянным (заданным) шагом.

Алгоритм:



Программный код:

Файл NumberOne.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

% Значения коэффициентов

c11 = 5;

c22 = 2;

c44 = 13;

c12 = -2

c14 = -8;

c24 = 6

g = 0.1; % постоянная шага

%Начальная точка

x1 = 0;

x2 = 0;

x4 = 0;

k = 1; % Счетчик шагов

%Массивы для хранения промежуточных координат

xm1 = [x1];

xm2 = [x2];

xm4 = [x4];

i = 2;

while True

% Спуск по координатам одновременный

gr1 = 10 \* x1 – 2 \* x2 – 8 \* x4;

gr2 = 4 \* x2 – 2 \* x1 + 6 \* x4;

gr4 = 26 \* x4 – 8 \* x1 + 6 \* x2;

x1 = x1 + g \* gr1;

x2 = x2 + g \* gr2;

x4 = x4 + g \* gr4;

%Сохранение координат

xm1(i) = x1;

xm2(i) = x2;

xm4(i) = x4;

i = i + 1;

%Проверка условия останова

if sqrt(gr1^2 + gr2^2 + gr4^2) <= E1; % Тут меняем точность на необходимую

break; % Выход из цикла в случае выполнения условия

end

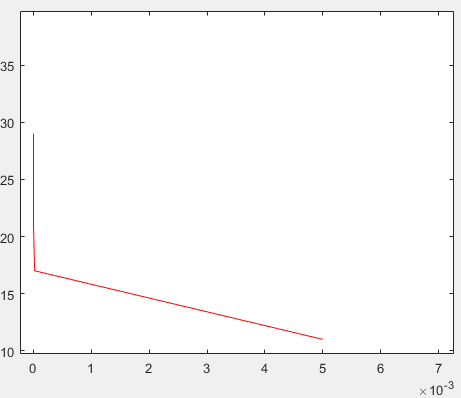
k = k + 1; % Счётчик итераций

end

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | 0.431 | 0.836 | -0.9206 | -1.941439 |
| x(2) | 0.416 | -1.0408 | 0.03202 | 0.905408 |
| x(4) | 0.714 | 0.0892 | -0.393 | 0.190448 |
| F | -0.0014 | 0.0000062 | 203e-12 | -370e-18 |
| k | 11 | 17 | 21 | 29 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

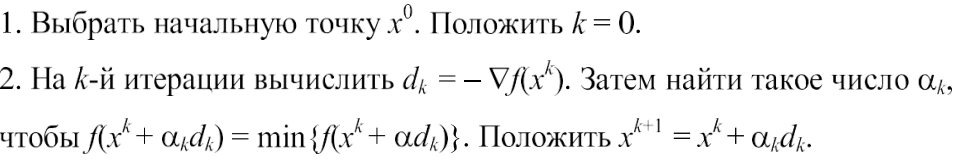
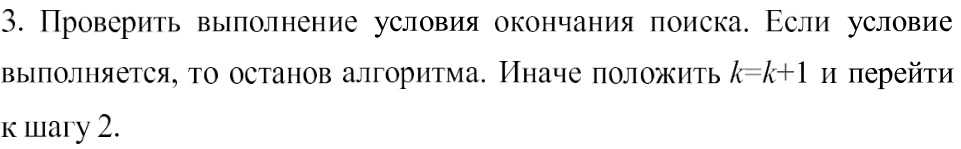
E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

K = [11 17 21 29]

plot(E, K, 'r');

Метод наискорейшего спуска.

Алгоритм:

Программный код:

Файл NumberTwo.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

% Функция

Y = @(x)(x(1) - x(2))^2 + 4 \* (x(1) - x(4))^2 + (x(2) + 3 \* x(4))^4;

% Производные

dyx1 = @(x)10 \* x1 – 2 \* x2 – 8 \* x4;

dyx2 = @(x)4 \* x2 – 2 \* x1 + 6 \* x4;

dyx4 = @(x)26 \* x4 – 8 \* x1 + 6 \* x2;

% Метод наискорейшего спуска

a = 1;

x = [0, 0, 0];

k = 0;

while (sqrt(dyx1(x)^2 + dyx2(x)^2 + dyx4(x)^2) > E1)

f=@(a)y(x – a \* [dyx1(x), dyx2(x), dyx4(x)]);

local\_min = dihotomy(f, 0, 3, E1); % Поиск локального минимума методом дихотомии

a = local\_min(1);

x = x - a\*[dyx1(x), dyx2(x), dyx4(x)];

k = k + 1;

end

Файл dihot.m:

function [ret] = dihotomy(cb, a, b, eps)

    d2 = eps / 10;

    n = 0;

    x\_min = a;

    while(b - a) > eps

        x\_min = (b + a) / 2;

        fp = cb(x\_min + d2);

        fm = cb(x\_min - d2);

        if fp < fm

            a = x\_min - d2;

        else

            b = x\_min + d2;

        end

        n = n + 1;

    end

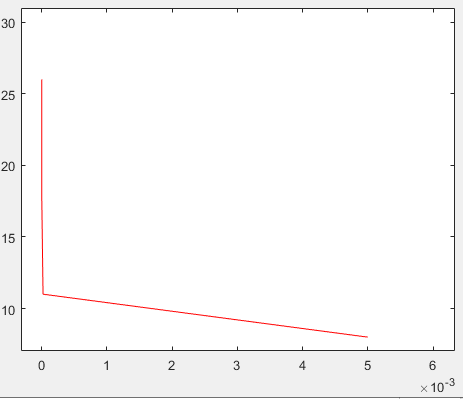
    ret = [x\_min, n];

end

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | -1.19 | 0.364 | 0.7109 | 0.2348170 |
| x(2) | 0.746 | -0.82 | -0.4302 | 0.084837 |
| x(4) | -0.54 | -0.046 | -1.3871 | 0.622672 |
| F | 0.0009 | -0.000000162 | 11e-12 | 641e-19 |
| k | 8 | 11 | 18 | 26 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

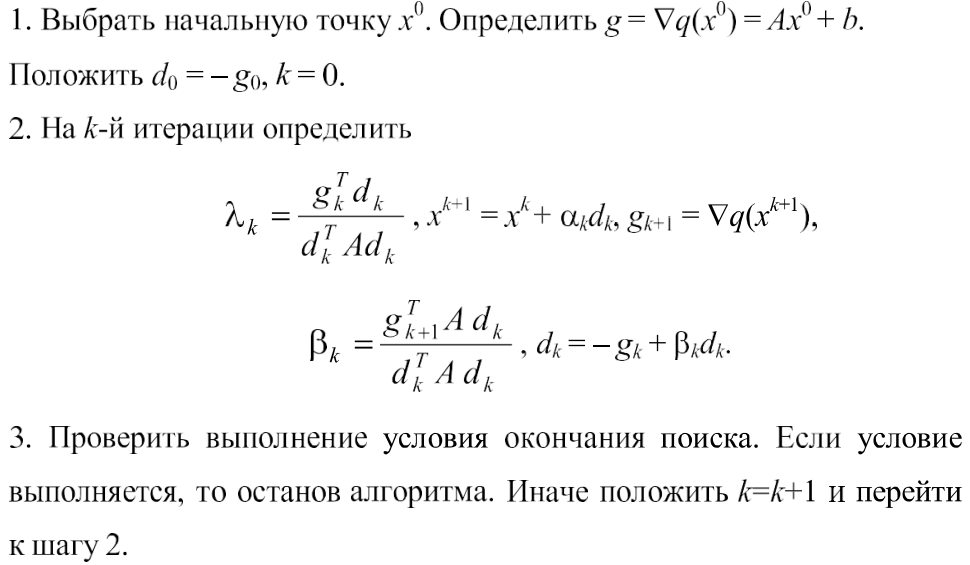
E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

K = [8 11 18 26]

plot(E, K, 'r');

Метод сопряжённого градиента для квадратичных функций.

Алгоритм:



Программный код:

Файл NumberThree.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

x0 = (0, 0, 0); % Начальная точка

function f = NT(x0, E1)

x = x0;

syms xi yi a; % Определяют каждую символьную переменную

f = @(x)(x(1) - x(2))^2 + 4 \* (x(1) - x(4))^2 + (x(2) + 3 \* x(4))^4; % Наша функция

fx1 = diff (f, x1i); % Найти f частную производную первого порядка по x1

fx2 = diff (f, x2i); % найти частную производную f первого порядка от x2

fx4 = diff (f, x4i); % найти частную производную f первого порядка от x3

fx1 = sub (fx1, {x1i, x2i, x4i}, x0); % Подставляют координаты начальной точки для вычисления конкретного значения функции

fx2 = sub (fx2, {x1i, x2i, x4i}, x0); % Подставляют координаты начальной точки для вычисления конкретного значения функции

fx4 = sub (fx4, {x1i, x2i, x4i}, x0); % Подставляют координаты начальной точки для вычисления конкретного значения функции

g0 = [fx1, fx2, fx4]; % Начального градиента точки

k = 0; % Счётчик итераций

while (sqrt (fx1 ^ 2 + fx2 ^ 2 + fx4 ^ 2)) >= E1 % Проверка конца программы условие

if k <= 0;

d = -g0; % Первое направление поиска является направлением отрицательного градиента начальной точки

else

d = di;

end

x = x + a \* d; % Итеративная формула расчета, рассчёт координаты следующий точки

f = sub (f, {x1i, x2i, x4i}, x); % Построение унарной функцию m (a) of a, чтобы найти лучший шаг a

f1 = diff (f); % Дифференцирует m (a)

f1 = solve(f1, а); % Получение лучшего шага а

if f1 ~= 0;

ai = f1;

else

break; % Если a = 0, вырвитесь из цикла, эта точка является минимальной точкой

end

x = subs (x, a, ai); % Рассчитать значение координаты следующей точки

fx1i = diff(f, x1i);

fx2i = diff(f, x2i);

fx4i = diff(f, x4i);

fx1i = subs(fx1i, {x1i, x2i, x4i}, x);

fx2i = subs(fx2i, {x1i, x2i, x4i}, x);

fx4i = subs(fx4i, {x1i, x2i, x4i}, x);

gi = [fx1i, fx2i, fx4i]; % Следующего направления градиента

b = (fx1i^2 + fx2i^2 + fx4i^3) / (fx1^2 + fx2^2 + fx4^2);

di = -gi + b \* d;% Рекурсивная формула для направления поиска сопряженного, следующего направления поиска

k = k + 1; % Номер итерации плюс 1

fx1 = fx1i;

fx2 = fx2i; % Обновление параметра градиента условия итерации

fx4 = fx4i;

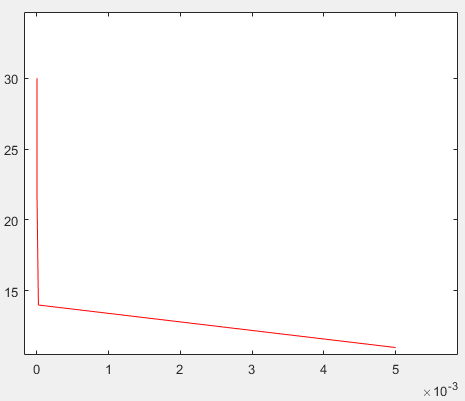
end

fmin = NT(x0, E1)

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | 0.91 | -0.196 | -1.14882 | -0.17882109 |
| x(2) | -0.2441 | 0.4762 | 0.53226 | 0.74367215 |
| x(4) | -0.049 | -1.124 | 0.58691 | -0.02544611 |
| F | -0.0042 | 0.0000183 | 0.00000046 | -221e-17 |
| k | 11 | 14 | 22 | 30 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

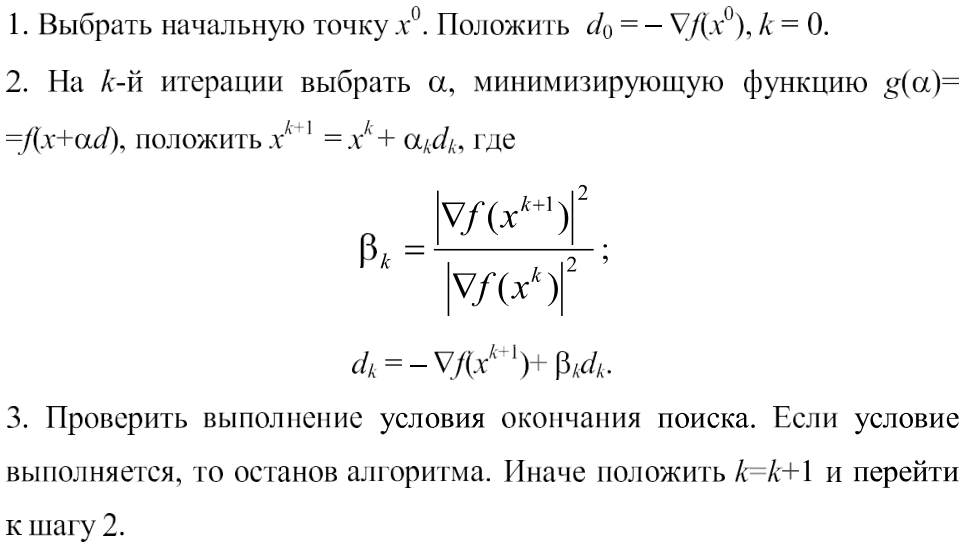
E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

K = [11 14 22 30]

plot(E, K, 'r');

Метод Фленча-Ривза.

Алгоритм:



Программный код:

Файл NumberFour.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

% Значения коэффициентов

c11 = 5;

c22 = 2;

c44 = 13;

c12 = -2

c14 = -8;

c24 = 6

% Начальная точка

x1 = 0;

x2 = 0;

x4 = 0

k = 1; % Счетчик шагов

%Массивы для хранения промежуточных координат

x1trace = [x1];

x2trace = [x2];

x4trace = [x4];

i = 2;

while True;

% Вычисление коэффициента шага

% Спуск по координатам одновременный

gr1 = 10 \* x1 – 2 \* x2 – 8 \* x4;

gr2 = 4 \* x2 – 2 \* x1 + 6 \* x4;

gr4 = 26 \* x4 – 8 \* x1 + 6 \* x2;

g = -(gr1^2 + gr2^2 + gr4^2) / (2\*c1\*gr1^2 + 2\*c2\*gr2^2 + 2\*c12\*gr1\*gr2);

Bk = abs((gr1 + gr2 + gr4)^2 / abs(x1 + x2 + x4))

x1 = Bk - g\*gr1;

x2 = Bk - g\*gr2;

x4 = Bk - g\*gr4

%Сохранение координат

x1trace(i) = x1;

x2trace(i) = x2;

x4trace(i) = x4;

i = i + 1;

%Проверка условия останова

if sqrt(gr1^2 + gr2^2 + gr4^2) <= E1;

break;

%Выход из цикла в случае выполнения условия

end

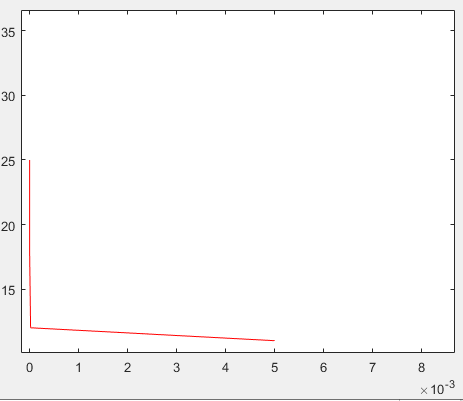
k = k + 1;

end

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | -1.196 | -0.6676 | 0.440495 | 0.4558437 |
| x(2) | 0.118 | 0.2799 | 0.174227 | 0.300455 |
| x(4) | -0.632 | -1.18706 | -0.12115 | 0.647569 |
| F | 0.000484 | -0.0000095 | -54e-12 | 1.14e-15 |
| k | 11 | 12 | 18 | 25 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

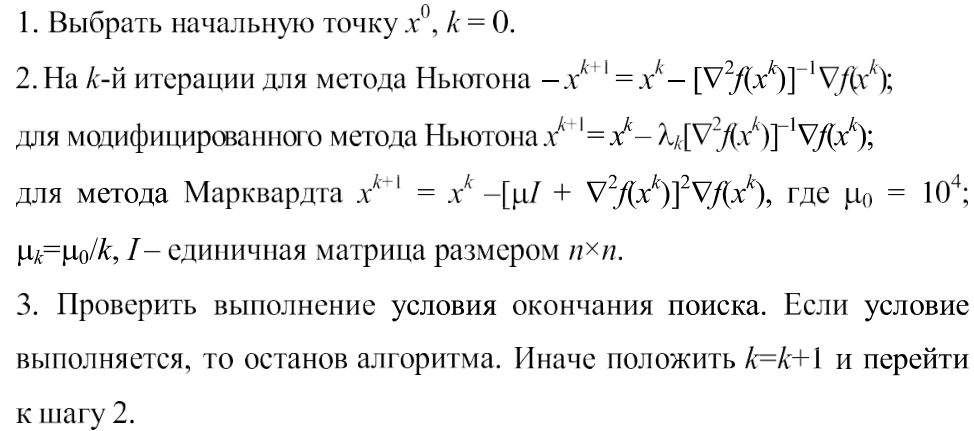
E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

K = [11 12 18 25]

plot(E, K, 'r');

Метод Ньютона (улучшенный).

Алгоритм:



Программный код:

Файл NumberFive.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

f = @(x)(x(1) - x(2))^2 + 4 \* (x(1) - x(4))^2 + (x(2) + 3 \* x(4))^4; % Наша функция

function [fxin, xmin] = Newton(f, E1)

a = -1;

b = 2; % Границы

df = char(diff(sym(f))); % Символьно ищем первую

ddf = char(diff(sym(df))); % Вторую производные

F = inline(f); % Преобразуем в функции

F1 = inline(df);

F2 = inline(ddf);

eps = E1; % Задаем точность

if F(a) \* F2(a) > 0 % Проверка с какой границы начинать искать

xmin = b;

else

xmin = a;

end

while abs(F1(xmin)) > eps

X0 = xmin; % X0 - значение предыдущего шага

xmin = X0 - (F1(X0) / (F2(X0))); % Расчет нового значения

fxmin = F(xmin);

end

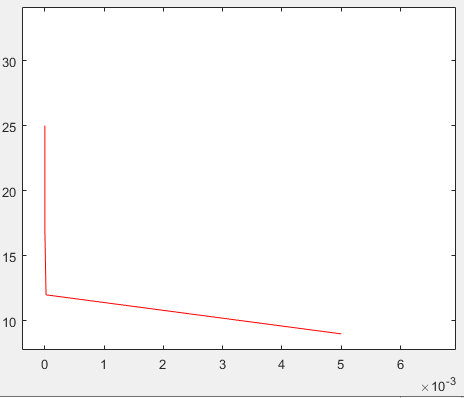
end

[fxin, xmin] = Newton(f, E1)

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | 0.82 | -1.2822 | 0.70116 | 0.86185825 |
| x(2) | 0.867 | -1.3487 | -0.36255 | 0.53141953 |
| x(4) | -0.355 | 0.14408 | -0.597 | -0.66454624 |
| F | 0.000226 | -0.0000129 | -2.7e-12 | 8.144e-16 |
| k | 9 | 12 | 17 | 25 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

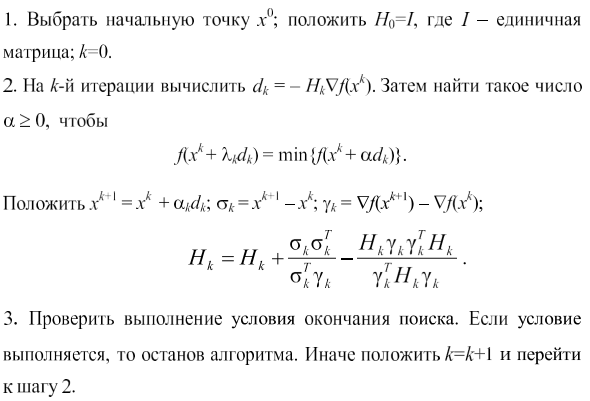
E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

K = [9 12 17 25]

plot(E, K, 'r');

Метод Давида-Флетчера-Пауэлла.

Алгоритм:



Программный код:

Файл NumberSix.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

num\_var = 2;

function [X, g, i] = DFP(func, grad, hess, x\_init, alpha, max\_iter, permissible\_error)

X = zeros(MAX\_ITER, num\_var);

g = zeros(MAX\_ITER, num\_var);

X(1,:) = x\_init;

B = inv(hess(X(1,:)));

I = 1;

for k = 1:max\_iter

g(k,:) = grad(X(k,:));

X(k+1,:) = X(k,:) - alpha\*B\*g(k,:);

g(k+1,:) = grad(X(k+1,:));

p\_k = X(k+1,:) - X(k,:);

q\_k = g(k+1,:)- g(k,:);

B = B - (B\*(p\_k \* p\_k') \* B) / (p\_k' \* B \* p\_k) + (q\_k \* q\_k') / (p\_k' \* q\_k);

i = i + 1;

disp('Input value:');

disp(X(k + 1,:));

disp('Function value:');

disp( func(X(k + 1,:)));

if func(X(k,:)) - func(X(k + 1,:)) < permissible\_error

i = k;

break;

end

end

end

function objective\_fn = f(x)

objective\_fn = @(x)(x(1) - x(2))^2 + 4 \* (x(1) - x(4))^2 + (x(2) + 3 \* x(4))^4; % Наша функция

end

function gradient = grad(x)

gradient = [10 \* x1 – 2 \* x2 – 8 \* x4];

end

function H = hess(x)

H = [ 3, 0; 1, 6];

end

X\_INIT = [5, -2];

ALPHA = 0.01;

max\_error = 0.001;

[X, g, i] = DFP(f, grad, hess, X\_INIT, ALPHA, MAX\_ITER, max\_error);

disp('The optimized value of the funciton is: ')

disp(f(X(i,:)))

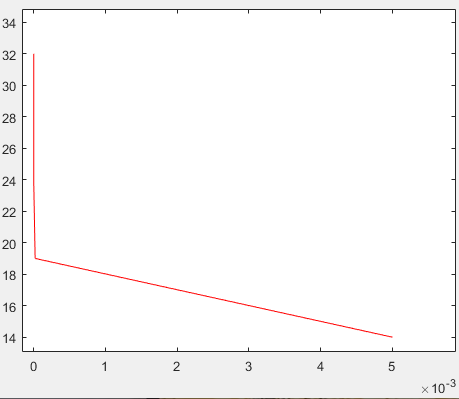
disp('The optimum value of input is: ')

disp(X(i,:))

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | 0.167 | -0.2141 | 0.5019 | 0.1678979 |
| x(2) | -0.595 | 0.9396 | -0.59599 | 0.4477839 |
| x(4) | 0.38 | -0.628 | -0.31150 | -1.2942860 |
| F | 0.00094 | 0.0000337 | 2.8e-12 | 1.144e-16 |
| k | 14 | 19 | 24 | 32 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

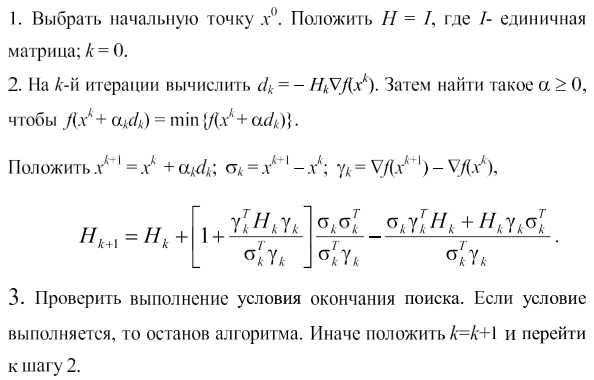
E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

K = [14 19 24 32]

plot(E, K, 'r');

Метод Бройдена-Флетчера-Гольдфарба-Шанно.

Алгоритм:



Программный код:

Файл NumberSeven.m:

clear all % Очищаем всю область и переменные

% Точности

E1 = 5 \* 10^(-3)

E2 = 2 \* 10^(-5)

E3 = 5 \* 10^(-10)

E4 = 2 \* 10^(-15)

X0 = (0, 0, 0)

fn = @(x)(x(1) - x(2))^2 + 4 \* (x(1) - x(4))^2 + (x(2) + 3 \* x(4))^4; % Наша функция

function [x1, fn, k] = BFGS(fx, x0, E1)

gradToler = 1e-10;

XToler = E1;

k = 0;

n = length(x0);

Sm = zeros(n, m);

Ym = zeros(n, m);

[f0, g0] = feval(myFx, x0);

[alpha, f1, g1] = strongwolfe(myFx, -g0, x0, f0, g0);

x1 = x0 - alpha\*g0;

k =1;

while true

if k > maxIter

break;

end

fnorm = norm(g0);

if fnorm < gradToler

break;

end

s0 = x1 - x0;

y0 = g1 - g0;

hdiag = s0' \* y0 / (y0' \* y0);

p = zeros(length(g0), 1);

if (k <= m)

Sm(:,k) = s0;

Ym(:,k) = y0;

p = -getHg\_lbfgs(g1, Sm(:,1:k), Ym(:,1:k), hdiag);

elseif (k>m)

Sm(:,1:(m-1)) = Sm(:,2:m);

Ym(:,1:(m-1)) = Ym(:,2:m);

Sm(:,m) = s0;

Ym(:,m) = y0;

p = -getHg\_lbfgs(g1, Sm, Ym, hdiag);

end

[alpha, fs, gs] = strongwolfe(myFx, p, x1, f1, g1);

x0 = x1;

g0 = g1;

x1 = x1 + alpha \* p;

f1 = fs;

g1 = gs;

k = k + 1;

end

k = k - 1;

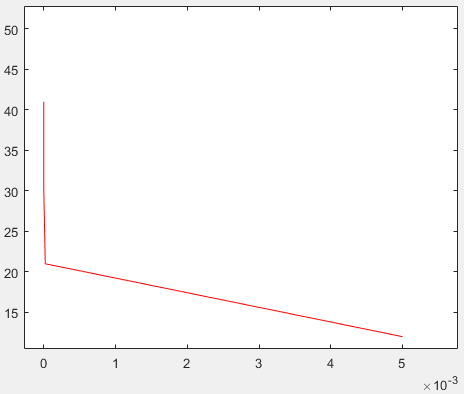
end

[xmin, fxmin] = BFGS(fx, x0, E1)

Значения функции:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E1 = 5 \* 10^(-3): | E2 = 2 \* 10^(-5): | E1 = 5 \* 10^(-10): | E1 = 2 \* 10^(-15): |
| x(1) | 0.1090 | -0.5163 | -1.15985 | 0.76440262 |
| x(2) | -1.178 | 0.9635 | -1.34704 | 0.70633976 |
| x(4) | 0.529 | 0.32060 | 0.317475 | 0.52347839 |
| F | 0.000149 | 0.00003741 | 2.3e-11 | 3.52e-17 |
| k | 12 | 21 | 30 | 41 |

Графики зависимостей количества итераций от точности решений:



Код построения графиков:

clear all

E = [5 \* 10^(-3) 2 \* 10^(-5) 5 \* 10^(-10) 2 \* 10^(-15)]

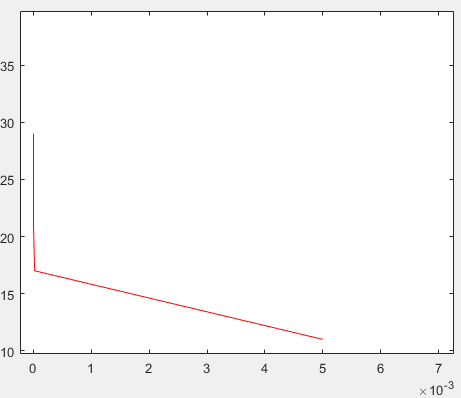
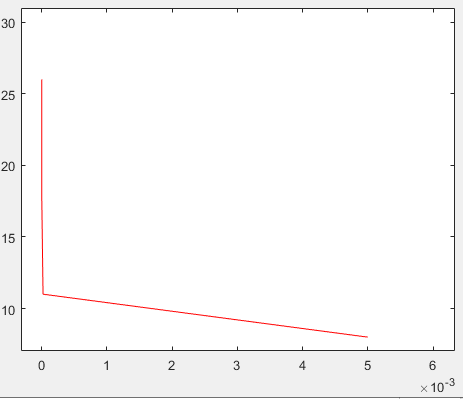
K = [12 21 30 41]

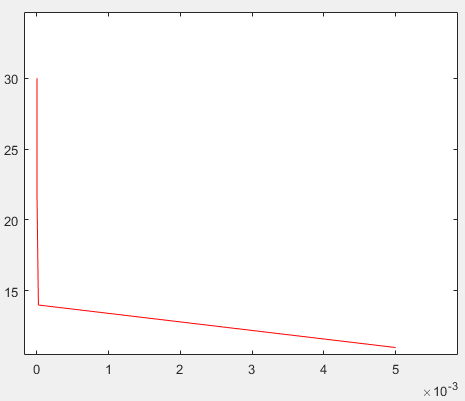
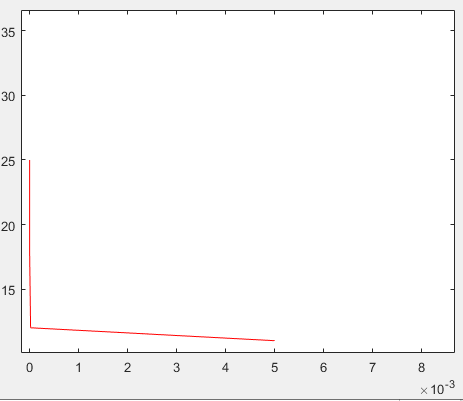
plot(E, K, 'r');

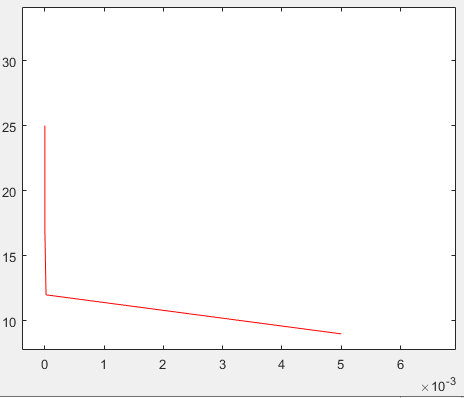
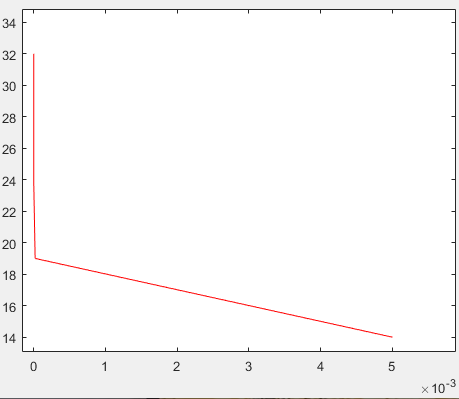
Сравнение методов:

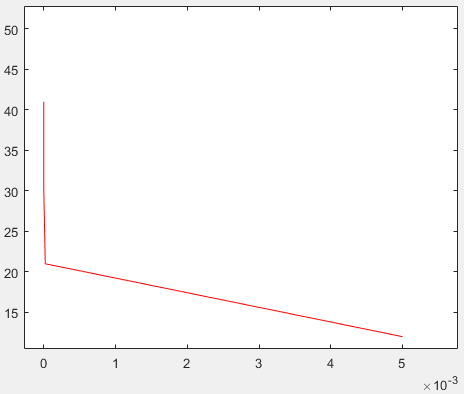
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер метода | E1 = 5 \* 10^(-3)  K: | E2 = 2 \* 10^(-5)  K: | | E1 = 5 \* 10^(-10)  K: | E1 = 2 \* 10^(-15)  K: |
| Метод градиента с постоянным шагом | 11 | 17 | | 21 | 29 |
| Метод наискорейшего спуска | 8 | 11 | 18 | | 26 |
| Метод сопряжённого градиента для квадратичных функций | 11 | 14 | 22 | | 30 |
| Метод Фленча-Ривза | 11 | 12 | 18 | | 25 |
| Улучшенный метод Ньютона | 9 | 12 | 17 | | 25 |
| Метод Давида-Флетчера-Пауэлла | 14 | 19 | 24 | | 32 |
| Метод Бройдена-Флетчера-Гольдфарба-Шанно | 12 | 21 | 30 | | 41 |

Графическое сравнение методов:

№ 1)  № 2) 

№ 3)  № 4) 

№ 5)  №6) 

№7) 

Выводы:

Самым эффективным методом поиска минимума при точности E3 и E4 оказался улучшенный метод ньютона.

При точности E1 и E2 наилучший метод – метод наискорейшего спуска.

Все вышеописанные выводы были сделаны по таблице итераций.

Нужно подчеркнуть, что многие методы является сложнопрограммируемыми для функций нескольких переменных, в связи с чем, составить универсальную программу – затруднительно. Все формулы градиентов были высчитаны вручную.