САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Институт компьютерных наук и кибербезопасности

высшая школа компьютерных технологий и информационных систем

**РАСЧЁТНОЕ ЗАДАНИЕ**

**«Нелинейное программирование. Безусловная оптимизация целевой функции»**

по дисциплине «Системный анализ и принятие решений»

Выполнил:

студент гр. 5130901/20102

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вагнер А.А.

(подпись)

Преподаватель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сиднев А.Г.

(подпись)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Санкт-Петербург

2024

**Вариант 1.**

Исходные данные

Содержанием задания является поиск безусловного максимума целевой функции следующими методами:

– наискорейшего подъема;

– Ньютона;

– сопряженных градиентов;

– релаксационный;

– переменной метрики Бройдена;

Вычислим градиент целевой функции:

Матрица Гессе

Предусмотрим единое правило останова:

1.Метод наискорейшего подъёма

Пусть начальная точка будет

*=>* не является решением

Шаг 1.

Шаг 2,3,4,5,6

Последующие точки были найдены при помощи программы в MATLAB

## Листинг 1 – Код программы решения методом наискорейшего подъёма

clc;

clear all;

close all;

initialX = [2; 2];%Начальная точка

index = [-6,-9,4,20,60];%Значения всех аргументов

e = 0.1;

H = [index(1)\*2, index(3); index(3), index(2)\*2];

% открытие файла вывода для записи результатов

fileID = fopen('results.txt', 'wt');

if (fileID == -1)

error('Не удалось открыть файл вывода.');

return;

end

%Область построения

x\_1=0:.1:10.5;

x\_2=-0.3:.1:12;

[x\_1,x\_2]=meshgrid(x\_1,x\_2);

w=(index(1)\*x\_1.^2 +index(2)\*x\_2.^2 + index(3)\*x\_1.\*x\_2 + index(4)\*x\_1 + index(5)\*x\_2 );

%функция построение графика метода

function [] = PlotGraph ()

figure;

contour(x\_1,x\_2,w,50)

hold on;

plot(x, y, '.-k');

hold off;

end

%Вычисление функции и значение её производной

function [fX, dfX] = derivative(X)

% вычисление значения функции от Х

fX = index(1) \* X(1)^2 + index(2) \* X(2)^2 + index(3) \* X(1) \* X(2) + index(4) \* X(1) + index(5) \* X(2);

% вычисление частных производных по Х1 и Х2 соответственно

dfX = [index(1)\*2 \* X(1) + index(3) \* X(2) + index(4); index(2)\*2 \* X(2) + index(3) \* X(1) + index(5)];

end

%%

%метод наискорейшего подъема

X = initialX;

[fX, dfX] = derivative(X);

i = 1;

j = 1;

x(i) = X(1);

y(i) = X(2);

t=-(dfX'\*dfX)/(dfX'\*H\*dfX);

fprintf(fileID, 'Метод наискорейшего подъема\n\n');

fprintf(fileID, 'i x1 x2 gradf(X)1 gradf(X)2 t fX ||df(X)||\n');

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X, dfX, t, fX, norm(dfX));

while (norm(dfX) > e)

X = X+t\*dfX;

[fX, dfX] = derivative(X); %считаем значение функции и производной

i = i+1;

x(i) = X(1);

y(i) = X(2);

t=-(dfX'\*dfX)/(dfX'\*H\*dfX);

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X, dfX, t, fX, norm(dfX));

end

PlotGraph();

legend ('Линии равного уровня','Метод наискорейшего подъема');

fprintf(fileID, '\n\n');

end

Таблица с промежуточными точками

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0 |  |  |  | 116 | 32.249 |
| 1 |  |  | 0.0765 | 146.7273 |  |
| 2 |  |  | 0.0591 | 149.6850 | 3.1042 |
| 3 |  |  | 0.0765 | 149.9697 | 0.8466 |
| 4 |  |  | 0.0591 | 149.9971 | 0.2988 |
| 5 |  |  | 0.0765 | 149.9997 | 0.0815 |

## Рис. 1 – Траектория поиска решения методом наискорейшего подъёма

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

Поиск на начального интервала неопределённости из точки (2; 2)

Оптимальная длина шага t = 0,591. В соответствии с методом разбегающихся шагов примем

Следовательно, начальным интервалом неопределённости является интервал

Тогда количество итераций, требующихся на нахождение оптимальной длины шага по методу золотого сечения, учитывая, что точность :

Начальные данные:

Шаг 1: k = 2

Шаг 2: k = 3

Шаг 3: k = 4

Шаг 4: k = 5

Шаг 5: k = 6

Шаг 6: k = 7

Шаг 7: k = 8

Полученная оптимальная длина шага как , что примерно равно t, полученного из скрипта - 0.0765.

2. Метод Ньютона

Пусть начальная точка будет

*=>* не является решением

Шаг 1.

Следовательно решение является оптимальным, найдено за один шаг.

## Листинг 2. Код программы решения методом Ньютона

function [] = Main ()

clc;

clear all;

close all;

initialX = [2; 2];%Начальная точка

index = [-6,-9,4,20,60];%Значения всех аргументов

e = 0.1;

H = [index(1)\*2, index(3); index(3), index(2)\*2];

% открытие файла вывода для записи результатов

fileID = fopen('results.txt', 'wt');

if (fileID == -1)

error('Не удалось открыть файл вывода.');

return;

end

%Область построения

x\_1=0:.1:10.5;

x\_2=-0.3:.1:12;

[x\_1,x\_2]=meshgrid(x\_1,x\_2);

w=(index(1)\*x\_1.^2 +index(2)\*x\_2.^2 + index(3)\*x\_1.\*x\_2 + index(4)\*x\_1 + index(5)\*x\_2 );

%функция построение графика метода

function [] = PlotGraph ()

figure;

contour(x\_1,x\_2,w,50)

hold on;

plot(x, y, '.-k');

hold off;

end

%Вычисление функции и значение её производной

function [fX, dfX] = derivative(X)

% вычисление значения функции от Х

fX = index(1) \* X(1)^2 + index(2) \* X(2)^2 + index(3) \* X(1) \* X(2) + index(4) \* X(1) + index(5) \* X(2);

% вычисление частных производных по Х1 и Х2 соответственно

dfX = [index(1)\*2 \* X(1) + index(3) \* X(2) + index(4); index(2)\*2 \* X(2) + index(3) \* X(1) + index(5)];

end

%%

% метод Ньютона

X = initialX;

[fX, dfX] = derivative(X);

i = 1;

clear x y;

x(i) = X(1);

y(i) = X(2);

fprintf(fileID, 'Метод Ньютона\n\n');

fprintf(fileID, 'i x1 x2 f(X) ||df(X)||\n');

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f\n', i, X, fX, norm(dfX));

while (norm(dfX) > e)

X = X - H^(-1) \* dfX;

[fX, dfX] = derivative(X);

i = i + 1;

x(i) = X(1);

y(i) = X(2);

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f\n', i, X, fX, norm(dfX));

end

PlotGraph();

legend ('Линии равного уровня','Метод Ньютона');

fprintf(fileID, '\n\n');

end

## Рис. 2 – Траектория поиска решения методом Ньютона

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Автоматически созданное описание

Результат работы программы соответствует тому, что было получено в расчётах.

3. Метод сопряжённых градиентов

Пусть начальная точка будет

*=>* не является решением

Шаг 1.

Шаг 2.

Следовательно решение найдено

Решим задачу с помощью Matlab

## Листинг 3. Код программы решения методом сопряжённых градиентов

function [] = Main ()

clc;

clear all;

close all;

initialX = [2; 2];%Начальная точка

index = [-6,-9,4,20,60];%Значения всех аргументов

e = 0.1;

H = [index(1)\*2, index(3); index(3), index(2)\*2];

% открытие файла вывода для записи результатов

fileID = fopen('results.txt', 'wt');

if (fileID == -1)

error('Не удалось открыть файл вывода.');

return;

end

%Область построения

x\_1=0:.1:10.5;

x\_2=-0.3:.1:12;

[x\_1,x\_2]=meshgrid(x\_1,x\_2);

w=(index(1)\*x\_1.^2 +index(2)\*x\_2.^2 + index(3)\*x\_1.\*x\_2 + index(4)\*x\_1 + index(5)\*x\_2 );

%функция построение графика метода

function [] = PlotGraph ()

figure;

contour(x\_1,x\_2,w,50)

hold on;

plot(x, y, '.-k');

hold off;

end

%Вычисление функции и значение её производной

function [fX, dfX] = derivative(X)

% вычисление значения функции от Х

fX = index(1) \* X(1)^2 + index(2) \* X(2)^2 + index(3) \* X(1) \* X(2) + index(4) \* X(1) + index(5) \* X(2);

% вычисление частных производных по Х1 и Х2 соответственно

dfX = [index(1)\*2 \* X(1) + index(3) \* X(2) + index(4); index(2)\*2 \* X(2) + index(3) \* X(1) + index(5)];

end

%%

% метод сопряженных градиентов

X = initialX;

[fX, dfX] = derivative(X);

K = dfX;

i = 1;

clear x y;

x(i) = X(1);

y(i) = X(2);

t = -(dfX' \* K) / ((H \* K)' \* K);

fprintf(fileID, 'Метод сопряженных градиентов\n\n');

fprintf(fileID, 'i x1 x2 gradf(X)1 gradf(X)2 K1 K2 t fX ||df(X)||\n');

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X, dfX, K, t, fX, norm(dfX));

while (norm(dfX) > e)

X = X + t \* K;

[fX, dfX] = derivative(X);

K=dfX+(norm(dfX))^2 / (norm(K))^2 \*K;

i = i + 1;

x(i) = X(1);

y(i) = X(2);

t = -(dfX' \* K) / ((H \* K)' \* K);

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X, dfX, K, t, fX, norm(dfX));

end

PlotGraph();

legend ('Линии равного уровня','Метод сопряженных градиентов');

fprintf(fileID, '\n\n');

end

## Рис. 3 – Траектория поиска решения методом сопряжённых градиентов

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Результат работы программы соответствует полученным ранее

4. Релаксационный метод

Пусть начальная точка будет

*=>* не является решением

Шаг 1.

Следовательно решение пока не найдено.

Шаг 2,3,4,5,6,7.

Далее задача была решена при помощи алгоритма, написанного в Matlab.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0 |  |  | 0.0833 | 116 | 32.249 |
| 1 |  |  | 0.0556 | 116.6667 | 33.3333 |
| 2 |  |  | 0.0833 | 147.5309 | 7.4074 |
| 3 |  |  | 0.0556 | 149.8171 | 2.4691 |
| 4 |  |  | 0.0833 | 149.9865 | 0.5487 |
| 5 |  |  | 0.0556 | 149.9990 | 0.1829 |
| 6 |  |  | 0.0833 | 149.9999 | 0.0406 |

Следовательно оптимальное решение

## Листинг 4. Код программы решения релаксационным методом

function [] = Main ()

clc;

clear all;

close all;

initialX = [2; 2];%Начальная точка

index = [-6,-9,4,20,60];%Значения всех аргументов

e = 0.1;

H = [index(1)\*2, index(3); index(3), index(2)\*2];

% открытие файла вывода для записи результатов

fileID = fopen('results.txt', 'wt');

if (fileID == -1)

error('Не удалось открыть файл вывода.');

return;

end

%Область построения

x\_1=0:.1:10.5;

x\_2=-0.3:.1:12;

[x\_1,x\_2]=meshgrid(x\_1,x\_2);

w=(index(1)\*x\_1.^2 +index(2)\*x\_2.^2 + index(3)\*x\_1.\*x\_2 + index(4)\*x\_1 + index(5)\*x\_2 );

%функция построение графика метода

function [] = PlotGraph ()

figure;

contour(x\_1,x\_2,w,50)

hold on;

plot(x, y, '.-k');

hold off;

end

%Вычисление функции и значение её производной

function [fX, dfX] = derivative(X)

% вычисление значения функции от Х

fX = index(1) \* X(1)^2 + index(2) \* X(2)^2 + index(3) \* X(1) \* X(2) + index(4) \* X(1) + index(5) \* X(2);

% вычисление частных производных по Х1 и Х2 соответственно

dfX = [index(1)\*2 \* X(1) + index(3) \* X(2) + index(4); index(2)\*2 \* X(2) + index(3) \* X(1) + index(5)];

end

%%

%метод релаксационный

X=initialX;

[fX, dfX] = derivative(X);

i = 1;

j = 1;

clear x y;

x(i) = X(1);

y(i) = X(2);

K=[dfX(1);0];

t=-(dfX'\*K)/(K'\*H\*K);

fprintf(fileID, 'Релаксационный метод\n\n');

fprintf(fileID, 'i x1 x2 gradf(X)1 gradf(X)2 K1 K2 t fX ||df(X)||\n');

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X, dfX, K, t, fX, norm(dfX));

while (norm(dfX) > e)

X = X+t\*K;

[fX, dfX] = derivative(X);

i = i+1;

x(i) = X(1);

y(i) = X(2);

K=dfX;

t=-(dfX'\*K)/(K'\*H\*K);

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X, dfX, K, t, fX, norm(dfX));

end

PlotGraph();

legend ('Линии равного уровня','Релаксационный метод');

fprintf(fileID, '\n\n');

end

## Рис. 4 – Траектория поиска решения релаксационным методом

Изображение выглядит как текст, снимок экрана, диаграмма, круг

Автоматически созданное описание

5. Метод переменной метрики Бройдена

Пусть начальная точка будет

*=>* не является решением

Шаг 1.

Шаг 2.

Также рассмотрим решение этой задачи при помощи скрипты в Matlab

## Листинг 5. Код программы решения методом переменной метрики Бройдена

function [] = Main ()

clc;

clear all;

close all;

initialX = [2; 2];%Начальная точка

index = [-6,-9,4,20,60];%Значения всех аргументов

e = 0.1;

H = [index(1)\*2, index(3); index(3), index(2)\*2];

% открытие файла вывода для записи результатов

fileID = fopen('results.txt', 'wt');

if (fileID == -1)

error('Не удалось открыть файл вывода.');

return;

end

%Область построения

x\_1=0:.1:10.5;

x\_2=-0.3:.1:12;

[x\_1,x\_2]=meshgrid(x\_1,x\_2);

w=(index(1)\*x\_1.^2 +index(2)\*x\_2.^2 + index(3)\*x\_1.\*x\_2 + index(4)\*x\_1 + index(5)\*x\_2 );

%функция построение графика метода

function [] = PlotGraph ()

figure;

contour(x\_1,x\_2,w,50)

hold on;

plot(x, y, '.-k');

hold off;

end

%Вычисление функции и значение её производной

function [fX, dfX] = derivative(X)

% вычисление значения функции от Х

fX = index(1) \* X(1)^2 + index(2) \* X(2)^2 + index(3) \* X(1) \* X(2) + index(4) \* X(1) + index(5) \* X(2);

% вычисление частных производных по Х1 и Х2 соответственно

dfX = [index(1)\*2 \* X(1) + index(3) \* X(2) + index(4); index(2)\*2 \* X(2) + index(3) \* X(1) + index(5)];

end

%%

% метод Бройдена

X=initialX;

[fX,dfX] = derivative(X);

i=1;

clear x y;

x(i) = X(1);

y(i) = X(2);

n = -eye(2);

K = dfX;

t = - dfX'\*K/(K'\*H\*K)

fprintf(fileID, 'метод Бройдена\n\n');

fprintf(fileID, 'i x gradf(X) K t fX n ||df(X)||\n');

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X(1), dfX(1), K(1), t, fX, n(1,1), n(1,2),norm(dfX));

fprintf(fileID, ' %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f \n',X(2), dfX(2), K(2), n(2,1), n(2,2));

bX=X;

[fbX, dfbX] = derivative(bX);

X = X + t \* K;

[fX, dfX] = derivative(X); % i-тая

i=2;

x(i) = X(1);

y(i) = X(2);

dg=dfX - dfbX

dx=X-bX

z=dx-n\*dg

dn=(z\*z')/(z'\*dg)

n = n +dn

K=-n\*dfX

t= -(dfX'\*K)/(K'\*H\*K)

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X(1), dfX(1), K(1), t, fX, n(1,1), n(1,2),norm(dfX));

fprintf(fileID, ' %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f \n',X(2), dfX(2), K(2), n(2,1), n(2,2));

while (norm(dfX) > e)

bX=X;

X=X+t\*K;

i = i + 1;

x(i) = X(1);

y(i) = X(2);

[fbX, dfbX] = derivative(bX);

[fX, dfX] = derivative(X); % i-тая

dg=dfX - dfbX;

dx=X-bX;

z=dx-n\*dg;

dn=(z\*z')/(z'\*dg);

n = n +dn;

K=-n\*dfX;

t= -(dfX'\*K)/(K'\*H\*K);

fprintf(fileID, '%-4d %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n',i, X(1), dfX(1), K(1), t, fX, n(1,1), n(1,2),norm(dfX));

fprintf(fileID, ' %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f \n',X(2), dfX(2), K(2), n(2,1), n(2,2));

end

PlotGraph();

legend ('Линии равного уровня','Метод Бройдена');

fprintf(fileID, '\n\n');

end

## Рис. 5 – Траектория поиска решения методом переменной метрики Бройдена

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание