Міністерство освіти і науки України Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу

Методи оптимізації 2
Лабораторна робота №1
«Числові методи безумовної оптимізації першого порядку. Градієнтний метод та його варіації»
Бригада (варіант) №6

Виконали:
Студенти 3 курсу
Групи КА-03
Товкач Максим
Муравський Ігор
Страшук Віталій

Перевірили:
Яковлева Алла Петрівна
Спекторський Ігор Якович

Київ-2023

Мета роботи: реалізувати градієнтний метод використовуючи два різні методи вибору кроку.

Завдання лабораторної роботи

1. У таблиці варіантів завдань знайти цільову функцію згідно з номером варіанта.
2. Скласти програму для мінімізації цільової функції одним із градієнтних методів. Конкретний тип градієнтного методу обрати самостійно, ураховуючи особливості функції (наприклад, ярність).

Під час складання програми треба:

- обчислити цільову функцію в окремій підпрограмі;
- частинні похідні цільової функції обчислити числово.

3. Знайти мінімум заданої цільової функції.

Теоретичні відомості

Загальна оптимізаційна задача:

$f(x) \to min$, $x \in X$ (1)

де f(x)- цільова функція, X-допустима множина.

Градієнтний метод є методом першого порядку. Алгоритми, що використовують лише інформацію про значення функції, що мінімізується, називаються алгоритмами **нульового порядку**; алгоритми, що також використовують інформацію про значення перших похідних - алгоритмами **першого** порядку, про других похідних - **другого** порядку.

Правило отримання точки x^{k+1} з точки x^k називається ітерацією алгоритму або кроком.

Ітерацію будь-якого алгоритму для вирішення завдання (1) можна записати у вигляді
$x^{k+1}=x^k+\alpha_k h^k$, $\alpha_k \in R$, $k=0,1,2,3,...$ Де h^k – напрямок кроку, a_k – довжина кроку. Зазвичай назва методу мінімізації визначається способом вибору h^k, а його різні варіанти зв'язуються з різними способами вибору α_k. **Скінченнокровими** називаються методи, що гарантують відшукання розв’язку за скінченну кількість кроків. Скінченнокрові методи вдається побудувати лише для деяких спеціальних задач оптимізації - наприклад, для задач **лінійного** і **квадратичного** програмування.

Для **Нескінченнокрокових** методів досягнення розв’язку гарантується лише при граничному переході.

Будемо говорити, що метод (2) збігається, якщо:

$x^k \longrightarrow x^*$ при $k \to \infty$, **(3)** де x^*-розв’язокзадачі(1).

Якщо$f(x^k)\to f(x^*), k\to\infty$,тодітакожговорять,щометод(2) збігається **по функції**. Послідовність x^k при цьому називається **мінімізуючою**. Мінімізуюча послідовність може і не збігатися до точки мінімуму.

Ефективність методу характеризується **швидкістю збіжності**.

1) Говорять, що послідовність x^k збігається до x^* **лінійно**, якщо існують також константи $q\in(0;1)$ і k_0 , що :

$\|x^{k+1}-x^*\|\le q\|x^k-x^*\|$ при $k\ge k_0$
Така швидкість також називається швидкістю збіжності

геометричної прогресії.
2)Говорять , що x^k збігається до x^* **надлінійно** або із

надлінійною швидкістю збіжності, якщо
$\|x^{k+1}-x^*\| \le q_k \|x^k-x^*\|$, $q_k \to 0^+$ при $k\to\infty$. **(5)**

3) Швидкість збіжності зветься **квадратичною**, якщо
$\exists\, c, k_0 : \|x^{k+1}-x^*\| \le c\, \|x^k-x^*\|^2$ **,** $\forall k\ge k_0$ **(6)**

Більшість теорем про швидкість збіжності методів оптимізації доводиться через припущення опуклості цільової функції, оцінки швидкості збіжності виводяться при ще сильнішому припущенні - сильній опуклості цільової функції.

Для неопуклих задач чисельні методи дозволяють відшукати лише локальні розв'язки або стаціонарні точки. Задача пошуку глобального розв'язку в загальному випадку дуже складна навіть для функції однієї змінної. Отримання ж досить точного розв'язку багатовимірних задач глобальної оптимізації за допомогою існуючих методів в даний час неможливо.

На практиці зазвичай використовують такі **умови зупинки**:

$\|x^{k+1} - x^k\| \leq \varepsilon 1$

$\|f(x^{k+1}) - f(x^k)\| \leq \varepsilon 2$

$\|f'(x^{k+1})\| \leq \varepsilon$

Кажуть, що вектор h задає напрямок спадання функції f в точці x, якщо f(x + αh) < f(x) **(1)**

при всіх досить малих α> 0.

Сам вектор h також називається напрямком спадання. Множина всіх напрямків спадання функції f в точці x позначається через U(x, f) (характеристика функції). Таким чином, якщо будь-який досить малий зсув з точки x в напрямку вектора h призводить до зменшення значення функції f, то h ∈ U(x, f).

Метод $x^{k+1} = x^k + \alpha k\, h^k$, $\alpha_k \in R$, $k=0,1,2,3,...$ **(4)**

називається методом спуску, якщо вектор h^k задає напрямок спадання функції f в точці x^k

А число a_k таке, що $f(x^{k+1}) < f(x^k)$, $k=0,1,2,3,...$. Найпростішим прикладом методу спуску є радієнтний метод, в якому $h^k = -f'(x^k)$.

Методи вибору довжини кроку

Ефективніше за все обирати α_k з наступної умови:

$f(x^k + \alpha_k h^k) = min f(x^k + \alpha h^k)$, **(5)**

α

де мінімум береться по α > 0. Такий метод найкращий, так як забезпечує досягнення найменшого значення функції уздовж заданого напрямку. Однак він вимагає на кожному кроці розв’язку одновимірної задачі мінімізації.

Є ще спосіб вибору коефіцієнтів α_k, що називається

дробленням кроку. Якщо h^k - напрямок спадання, то дроблення кроку виконується так:

вибираються деякі константи $\beta > 0$, $0 < \lambda < 1$ (часто $\beta = 1$, а $\lambda = 0.5$). Для коефіцієнту $\alpha = \beta$ перевіряється умова:

$$f(x^k + \alpha h^k) < f(x^k)$$

Якщо вона виконана, вважають $\alpha_k = \alpha$. Якщо ні, то проводиться дроблення кроку, тобто вважають $\alpha = \lambda\beta$ і знову перевіряють умову (*). Процес дроблення, тобто множення поточного значення α на λ, триває до тих пір, поки (*) не буде виконана. Цей процес не може бути нескінченним, оскільки h^k - напрямок спадання. Перше α, при якому умова виконана, приймається за α_k.

Чисельні методи безумовної оптимізації

Нехай дана задача безумовної оптимізації:

$$f(x) \longrightarrow min, \ x \in R^n$$

В градієнтному методі h^k береться рівним антиградієнту функції f в точці x^k, тобто

$$h^k = -f'(x^k).$$

Отже, в градієнтному методі

$$x^{k+1} = x^k - \alpha_k * f'(x^k), \quad \alpha_k > 0, \ k = 0, 1, 2, ...$$

Аналітичний розв'язок задачі:

$$f(x,y) = x^2 + 4y^2 + 0{,}001xy - y$$

1) Знаходимо стаціонарні точки:

$$\begin{cases} \dfrac{\partial f}{\partial x} = 2x + 0{,}001y = 0 \\ \dfrac{\partial f}{\partial y} = 8y + 0{,}001x - 1 = 0 \end{cases} \Rightarrow \begin{cases} x = -\dfrac{0{,}001y}{2} = -\dfrac{\frac{1}{1000}\cdot y}{2} = -\dfrac{y}{2000} \\ 8y + 0{,}001x - 1 = 0 \end{cases}$$

$$\Rightarrow 8y - \frac{y}{2000} \cdot \frac{1}{1000} - 1 = 0$$

$$\Rightarrow 8y - \frac{y}{2000000} - 1 = 0$$

$$\Rightarrow \frac{16000000y - y}{2000000} = 1$$

$$\Rightarrow \frac{15999999y}{2000000} = 1 \quad \Big| \cdot 2000000$$

$$\Rightarrow 15999999y = 2000000$$

$$\Rightarrow y = \frac{2000000}{15999999} \quad \Rightarrow x = -\frac{y}{2000} = -\frac{1000}{15999999}$$

Отже $x^*\left(-\dfrac{1000}{15999999}; \dfrac{2000000}{15999999}\right) \approx \boxed{\left(-0{,}0000625; 0{,}125\right)}$ – стаціонарна точка

2) Матриця других похідних (Гессіан)

$$f'' = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2 & 0{,}001 \\ 0{,}001 & 8 \end{pmatrix}$$

$$f''(x^*) = \begin{pmatrix} 2 & 0{,}001 \\ 0{,}001 & 8 \end{pmatrix} = A^*$$

3) Кутові мінори:

$$M_1^1 = 2 > 0$$

$$M_{12}^{12} = 16 - (0{,}001)^2 > 0$$

Отже, за критерієм Сильвестра, матриця A^* є додатно визначеною $\Rightarrow x^*$ - точка строгого min.

4) Функція є опуклою (оскільки всі мінори > 0)

5) ~~$f(x^*) \approx -0{,}0625$~~

Запишемо функцію у вигляді $f(\bar{x}) = (A\bar{x}, \bar{x}) + (b, \bar{x})$ (квадратична форма)

$$f(x,y) = \left(\begin{pmatrix} 1 & 0{,}001 \\ 0{,}001 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix} \right) + \left(\begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix} \right)$$

Отже $A = \begin{pmatrix} 1 & 0{,}001 \\ 0{,}001 & 4 \end{pmatrix}$

$$M_1^1 = 1 >$$

$$M_{12}^{12} = 4 - (0{,}001)^2 > 0$$

Всі кутові мінори матр. $A > 0$

$\Rightarrow$ За критерієм Сильвестра A - додатно визнач.

$\Rightarrow (Ax, x) > 0$ (при $x \neq 0$)

$\Rightarrow$ функція має т. глоб. min

6) Отже x^* - точка строгого глоб. min

7) $\boxed{f(x^*) \approx -0{,}0625}$

Вибір довжини кроку a:

У програмі реалізовано два методи вибору довжини кроку:

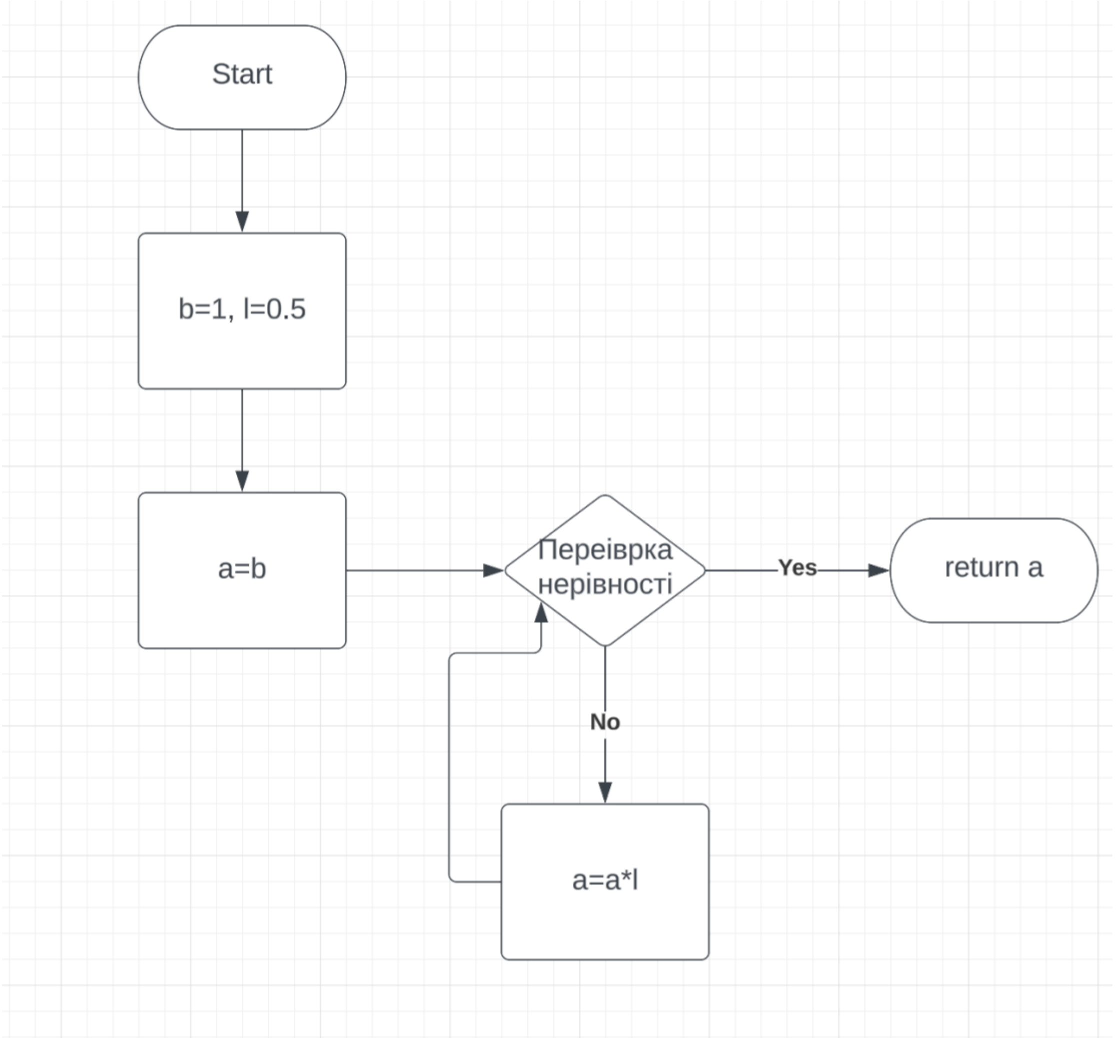
1)find_a(float k)

Це метод дроблення кроку. Ми обираємо деяке l з проміжку (0;1) (в нашому випадку l=0.5) та деяке b>0 (в нашому випадку b=1). Початкове значення a ми беремо, як значення змінної b (1). Потім ставимо це значення у нерівність:

$$f\left(x^k - a * f'(x^k)\right) < f(x^k)$$

Якщо нерівність справджується, то ми повертаємо значення a. А якщо ні, то домножаємо a на l та повертаємось до перевірки нерівності.

Блок схема алгоритму:



2)findA (float k)

Це метод найшвидшого спуску.

Нам треба знайти мінімальне а з умови:

$$f\left(x^k+\alpha_k h^k\right)=\min_{\alpha} f\left(x^k+\alpha h^k\right)$$

Де x^k та h^k=$-f'(x^k)$ – const.

Отже нам треба вирішувати задачу одновимірної мінімізації.

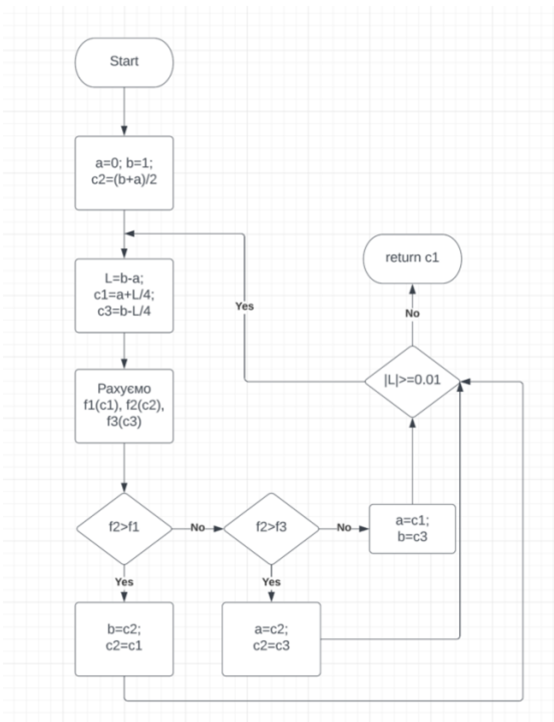
Для цього будемо використовувати метод половинного ділення.

Ми обираємо відрізок [a;b]=[0;1], вводимо змінну c2, яка дорівнює середині відрізку [a;b], змінну L=b-a(довжина відрізка), а також точку c1=a+L/4 та c2=b-L/4. Отже ми маємо три точки на нашому відрізку (c1, c2 та c3), які ділять його на 4 частини.

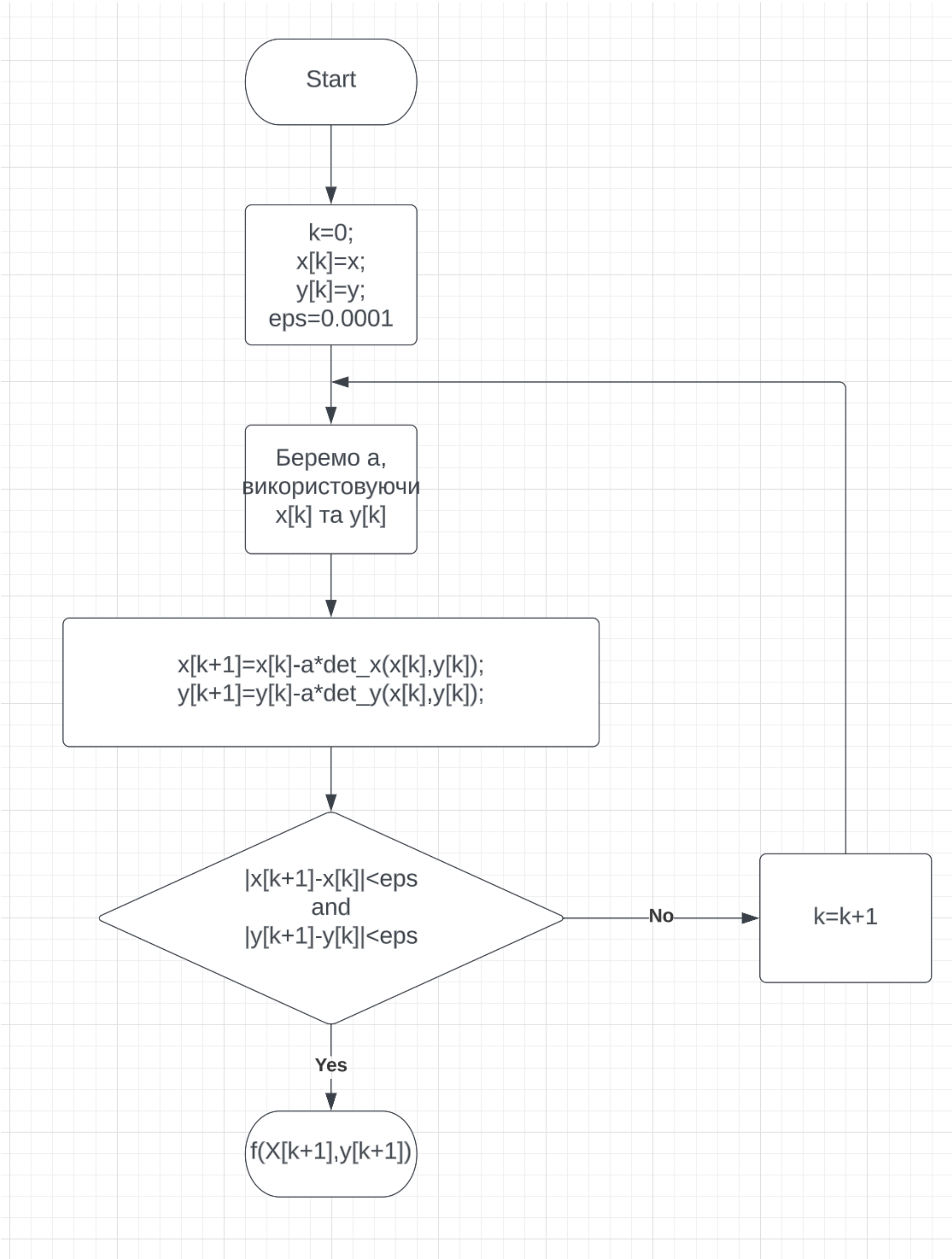
Тепер підставляємо у формулу замість змінної а змінну c1 (отримуємо f1), c2 (отримуємо f2) та c3 (отримуємо f3).

Тепер порівнюємо значення f1,f2 та f3. Якщо f1<f2, то ми відкидаємо другу половину відрузка (b=c2, c2=c1). Якщо f3<f2, то ми відкидаємо першу половину відрізка (a=c2, c2=c3). Інакше- звужуємо відрізок (a=c1, b=c3). Проводимо ці дії допоки |L|=|a-b| не стане меншим за 0.01. Повертаємо значення c1.

Блок схема алгоритму:



Блок схема всього алгоритму:



Код програми:

```
#include <iostream>
#include <math.h>
using namespace std;

float T[1000][2];

float func(float x,float y){
    float f;
    f=pow(x,2)+4*pow(y,2)+0.001*x*y-y;
    return f;
}

float det_x(float x,float y){
    float f,h=0.01;
    f=(func(x+h,y)-func(x-h,y))/(2*h);
    return f;
}

float det_y(float x,float y){
    float f,h=0.01;
    f=(func(x,y+h)-func(x,y-h))/(2*h);
    return f;
}
```

```
float grad_x(float x, float y){
    float f;
    f = 2*x+0.001*y;
    return f;
}

float grad_y(float x, float y){
    float f;
    f = 8*y+0.001*x-1;
    return f;
}


float find_a(int k){//метод дроблення
    int b=1;
    float l=0.5;
    float a=b*2;

    while(1){
        a=a*l;
        if(func(T[k][0]-a*grad_x(T[k][0],T[k][1]),T[k][1]-
a*grad_y(T[k][0],T[k][1]))<func(T[k][0],T[k][1]))break;
        if(a<0.005)break;
    }
```

```
    return a;
}

float findA(int k){//метод найшвидшого спуску + половинного ділення
    float a=0,b=1;
    float L,f1,f2,f3,A,c1,c2,c3;
    float x=T[k][0],y=T[k][1];
    float f_x=det_x(x,y), f_y=det_y(x,y);

    c2=(b+a)/2;

    do{
        L=b-a;
        c1=a+L/4;
        c3=b-L/4;

        f1=func(x - c1 * f_x, y - c1 * f_y);
        f2=func(x - c2 * f_x, y - c2 * f_y);
        f3=func(x - c3 * f_x, y - c3 * f_y);

        if(f1<f2){
            b=c2;
            c2=c1;
        }
```

```
        else if(f3<f2){
            a=c2;
            c2=c3;
        }
        else {
            a=c1;
            b=c3;
            }

    }while(abs(L)>=0.01);
    return A=c1;
}



int main() {
    float eps=0.00001;
    float a,x,y;
    int k=0;

    cout<<"Введіть початкову координату x: ";
    cin>>x;
    cout<<"Введіть початкову координату y: ";
    cin>>y;
    cout<<endl<<endl;
```

```
    T[0][0]=x;
    T[0][1]=y;

    while(1){
        a=findA(k);
        T[k+1][0]=T[k][0]-a*det_x(T[k][0],T[k][1]);
        T[k+1][1]=T[k][1]-a*det_y(T[k][0],T[k][1]);
        cout<<"k="<<k<<" a="<<a<<" x="<<T[k][0]<<" y="<<T[k][1]<<"
f="<<func(T[k][0],T[k][1])<<endl;
        if((abs(T[k+1][0]-T[k][0])<eps)&&(abs(T[k+1][1]-T[k][1])<eps))break;
        k++;
    }

    cout<<fixed;
    cout.precision(5);
    cout<<endl<<endl<<"Кількість кроків:
"<<k<<endl<<"F("<<T[k+1][0]<<"; "<<T[k+1][1]<<") = ";
    cout<<func(T[k+1][0],T[k+1][1])<<endl<<endl;

}
```

Результати роботи програми:

Початкова точка (1;1)

Використовуючи метод найшвидшого спуску:

```
Введіть початкову координату x: 1
Введіть початкову координату y: 1


k=0 a=0.130859 x=1 y=1 f=4.001
k=1 a=0.435547 x=0.73815 y=0.0838557 f=0.489199
k=2 a=0.126953 x=0.0951164 y=0.226897 f=-0.0118995
k=3 a=0.486328 x=0.0709369 y=0.123396 f=-0.0574489
k=4 a=0.123047 x=0.00187967 y=0.129603 f=-0.0624115
k=5 a=0.451172 x=0.00140116 y=0.125072 f=-0.0624978
k=6 a=0.123047 x=8.04309e-05 y=0.124812 f=-0.0624998
k=7 a=0.498047 x=4.52727e-05 y=0.124997 f=-0.0625
k=8 a=0.498047 x=-6.20604e-05 y=0.125009 f=-0.0625
k=9 a=0.123047 x=-6.25242e-05 y=0.124974 f=-0.0625
k=10 a=0.498047 x=-6.25242e-05 y=0.125 f=-0.0625


Кількість кроків: 10
F(-0.00006; 0.12500) = -0.06250
```

Використовуючи метод дроблення кроку:

```
Введіть початкову координату x: 1
Введіть початкову координату y: 1


k=0 a=0.25 x=1 y=1 f=4.001
k=1 a=0.25 x=0.49975 y=-0.750246 f=3.2511
k=2 a=0.25 x=0.250063 y=1.00012 f=3.06362
k=3 a=0.25 x=0.124783 y=-0.75018 f=3.01674
k=4 a=0.25 x=0.0625799 y=1.00015 f=3.005
k=5 a=0.25 x=0.0310401 y=-0.750162 f=3.00208
k=6 a=0.25 x=0.0157068 y=1.00015 f=3.00133
k=7 a=0.25 x=0.00760353 y=-0.750153 f=3.00113
k=8 a=0.25 x=0.0039885 y=1.00015 f=3.00106
k=9 a=0.25 x=0.00174439 y=-0.75015 f=3.00105
k=10 a=0.25 x=0.00106192 y=1.00015 f=3.00102
k=11 a=0.25 x=0.000278115 y=-0.750144 f=3.00101
k=12 a=0.125 x=0.000325799 y=1.00014 f=3.00098
k=13 a=0.5 x=0.000120163 y=0.125001 f=-0.0625
k=14 a=0.00390625 x=-6.24694e-05 y=0.124997 f=-0.0625


Кількість кроків: 14
F(-0.00006; 0.12500) = -0.06250
```

Початкова точка (0;5)

Використовуючи метод найшвидшого спуску:

```
Введіть початкову координату x: 0
Введіть початкову координату y: 5


k=0 a=0.123047 x=0 y=5 f=95
k=1 a=0.123047 x=-0.000610203 y=0.201038 f=-0.0393724
k=2 a=0.123047 x=-0.000484765 y=0.126188 f=-0.0624942
k=3 a=0.498047 x=-0.000380987 y=0.125019 f=-0.0624999
k=4 a=0.123047 x=-6.37188e-05 y=0.124945 f=-0.0625
k=5 a=0.498047 x=-6.34209e-05 y=0.124999 f=-0.0625


Кількість кроків: 5
F(-0.00006; 0.12500) = -0.06250
```

Використовуючи метод дроблення кроку:

```
Введіть початкову координату x: 0
Введіть початкову координату y: 5


k=0 a=0.125 x=0 y=5 f=95
k=1 a=0.25 x=-0.000619888 y=0.124865 f=-0.0624996
k=2 a=0.25 x=-0.00034119 y=0.125136 f=-0.0624999
k=3 a=0.25 x=-0.000201864 y=0.124865 f=-0.0624999
k=4 a=0.25 x=-0.000132155 y=0.125136 f=-0.0624999
k=5 a=0.125 x=-9.73698e-05 y=0.124864 f=-0.0624999
k=6 a=0.5 x=-8.86386e-05 y=0.125 f=-0.0625
k=7 a=0.00390625 x=-6.24685e-05 y=0.125 f=-0.0625


Кількість кроків: 7
F(-0.00006; 0.12500) = -0.06250

Program ended with exit code: 0
```

Початкова точка (-3.42;10)

Використовуючи метод найшвидшого спуску:

```
Введіть початкову координату x: -3.42
Введіть початкову координату y: 10


k=0 a=0.123047 x=-3.42 y=10 f=401.662
k=1 a=0.427734 x=-2.57961 y=0.279568 f=6.68673
k=2 a=0.126953 x=-0.372961 y=-0.248249 f=0.633952
k=3 a=0.486328 x=-0.278232 y=0.130879 f=0.015015
k=4 a=0.123047 x=-0.0076718 y=0.108141 f=-0.061305
k=5 a=0.451172 x=-0.00579712 y=0.124738 f=-0.0624668
k=6 a=0.123047 x=-0.000622356 y=0.125687 f=-0.0624978
k=7 a=0.498047 x=-0.000484657 y=0.125011 f=-0.0624998
k=8 a=0.123047 x=-6.41379e-05 y=0.124968 f=-0.0625
k=9 a=0.498047 x=-6.37483e-05 y=0.125 f=-0.0625


Кількість кроків: 9
F(-0.00006; 0.12500) = -0.06250
```

Використовуючи метод дроблення кроку:

```
Введіть початкову координату x: -3.42
Введіть початкову координату y: 10


k=0 a=0.25 x=-3.42 y=10 f=401.662
k=1 a=0.25 x=-1.71254 y=-9.74945 f=392.906
k=2 a=0.25 x=-0.853853 y=10.0004 f=390.751
k=3 a=0.25 x=-0.429659 y=-9.7506 f=390.236
k=4 a=0.25 x=-0.212603 y=10.0011 f=390.133
k=5 a=0.25 x=-0.108843 y=-9.75136 f=390.12
k=6 a=0.25 x=-0.0516224 y=10.0019 f=390.153
k=7 a=0.25 x=-0.0283527 y=-9.7525 f=390.199
k=8 a=0.25 x=-0.0119495 y=10.0031 f=390.241
k=9 a=0.125 x=-0.00851631 y=-9.75327 f=390.258
k=10 a=0.5 x=-0.00508308 y=0.125084 f=-0.0624748
k=11 a=0.125 x=-6.25104e-05 y=0.124751 f=-0.0624998
k=12 a=0.00390625 x=-6.24871e-05 y=0.125 f=-0.0625


Кількість кроків: 12
F(-0.00006; 0.12500) = -0.06250
```

Початкова точка (100;100)

Використовуючи метод найшвидшого спуску:

```
Введіть початкову координату x: 100
Введіть початкову координату y: 100


k=0 a=0.126953 x=100 y=100 f=49910
k=1 a=0.486328 x=74.6094 y=-1.46332 f=5576.48
k=2 a=0.123047 x=2.02823 y=4.68702 f=87.3089
k=3 a=0.451172 x=1.52852 y=0.195932 f=2.2943
k=4 a=0.126953 x=0.149179 y=-0.0607781 f=0.0977991
k=5 a=0.482422 x=0.111309 y=0.127884 f=-0.0500628
k=6 a=0.123047 x=0.00385154 y=0.1167 f=-0.0622092
k=7 a=0.451172 x=0.00288934 y=0.12487 f=-0.0624912
k=8 a=0.123047 x=0.000225781 y=0.125338 f=-0.0624995
k=9 a=0.498047 x=0.0001548 y=0.125005 f=-0.0625
k=10 a=0.123047 x=-6.16286e-05 y=0.124984 f=-0.0625
k=11 a=0.498047 x=-6.18348e-05 y=0.125 f=-0.0625


Кількість кроків: 11
F(-0.00006; 0.12500) = -0.06250
```

Використовуючи метод дроблення кроку:

```
Введіть початкову координату x: 100
Введіть початкову координату y: 100


k=0 a=0.25 x=100 y=100 f=49910
k=1 a=0.25 x=50 y=-99.8047 f=42438.7
k=2 a=0.25 x=25.0488 y=100.098 f=40608
k=3 a=0.25 x=12.5 y=-99.9023 f=40176.8
k=4 a=0.25 x=6.25 y=100.195 f=40095.9
k=5 a=0.25 x=3.07617 y=-99.9512 f=40070.1
k=6 a=0.25 x=1.5625 y=100.244 f=40097.9
k=7 a=0.25 x=0.732422 y=-100.049 f=40139.6
k=8 a=0.25 x=0.390625 y=100.342 f=40173.8
k=9 a=0.25 x=0.146484 y=-100.146 f=40217.4
k=10 a=0.125 x=0.0976562 y=100.439 f=40251.9
k=11 a=0.25 x=0.0488281 y=0.0976562 f=-0.0571203
k=12 a=0.25 x=0.0243896 y=0.152332 f=-0.0589134
k=13 a=0.25 x=0.0121567 y=0.0976624 f=-0.0593616
k=14 a=0.25 x=0.00605392 y=0.152335 f=-0.0594737
k=15 a=0.25 x=0.00298889 y=0.0976639 f=-0.0595017
k=16 a=0.25 x=0.00147005 y=0.152335 f=-0.0595087
k=17 a=0.25 x=0.000696955 y=0.0976642 f=-0.0595105
k=18 a=0.25 x=0.000324054 y=0.152336 f=-0.0595109
k=19 a=0.25 x=0.000123959 y=0.0976643 f=-0.059511
k=20 a=0.25 x=3.75789e-05 y=0.152336 f=-0.059511
k=21 a=0.25 x=-1.92784e-05 y=0.0976643 f=-0.059511
k=22 a=0.125 x=-3.40864e-05 y=0.152336 f=-0.0595111
k=23 a=0.5 x=-4.46104e-05 y=0.125 f=-0.0625
k=24 a=0.00390625 x=-6.24917e-05 y=0.125 f=-0.0625


Кількість кроків: 24
F(-0.00006; 0.12500) = -0.06250
```

Висновок:

Ми склали програму для мінімізації заданої цільової функції f(x,y). Для цього ми використовували градієнтні методи з вибором довжини кроку за допомогою методів дроблення кроку та найшвидшого спуску. Напрямок h^k у градієнтному методі є рівним $-grad(x^k)$ (отже напрямок спадання). Великою перевагою градієнтного методу є те, що початкову точку можна брати будь яку. Градієнтний метож має лінійну швидкість (яка є медленішою та менш ефективнішою за квадратичну та надлінійну). Ми бачимо з результатів роботи програми, що біля точки мінімума метод сповільнюється, а далеко від точки – летить дуже швидко (адже біля точки min антиградієнт стає дуже малим (бо в точці min grad=0)). За точність було взято $eps=10^{(-5)}$, а умовою зупинки - $||x^{(k+1)}-x^k||<=eps$. Також, можна побачити, що в кожній наступній точці значення функції стає меншим (реалізовується метод спуску). З наведеної нижче таблиці результатів для 4 різних точок ми бачимо, що метод найшвидшого спуску потребує менше ітерацій для знаходження точки мінімума, аніж метод дроблення кроку. Реузльтати (координати точки та значення функції у цій точці) співпали для всіх початкових точок (з точністю eps=0.00001) та співпали з розв’язком, який ми знайшли аналітично.

Початкова точка	Метод вибору довжини кроку	Кількість ітерацій	Координати точки мінінмума	Значення функції у цій точці
(1; 1)	Дроблення кроку	14	(-0.00006; 0.125)	-0.0625
	Найшвидшого спуску	10	(-0.00006; 0.125)	-0.0625
(0;5)	Дроблення кроку	7	(-0.00006; 0.125)	-0.0625
	Найшвидшого спуску	5	(-0.00006; 0.125)	-0.0625

(-3.42; 10)	Дроблення кроку	12	(-0.00006; 0.125)	-0.0625
	Найшвидшого спуску	9	(-0.00006; 0.125)	-0.0625
(100;100)	Дроблення кроку	24	(-0.00006; 0.125)	-0.0625
	Найшвидшого спуску	11	(-0.00006; 0.125)	-0.0625