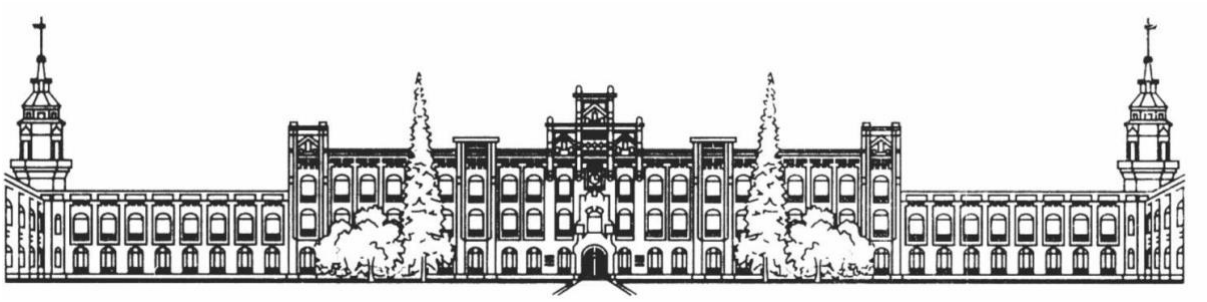


Міністерство освіти і науки України Національний технічний університет України

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

Методи оптимізації 2
Лабораторна робота №4
«Методи спряжених напрямів.
Метод спряжених градієнтів»
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: знайти мінімум заданих функцій за допомогою методів спряжених напрямів.

Завдання лабораторної роботи:

У таблиці варіантів завдань знайти цільову функцію згідно з номером варіанта, а також обрати тривимірну та не квадратичну функції.

Скласти програму мінімізації заданих функцій за допомгою методу спряжених градієнтів. Для не квадратичнох функції знайти точку мінімума двома методами – методом спряжених градієнтів з оновленням коефіцієнтів та без.

Під час складання програми треба:

- обчислити цільову функцію в окремій підпрогамі;
- частинні похідні цільової функції обчислити чисельно.

Знайти мінімум заданих функції.

4.1. Теоретичні відомості. Методи спряжених напрямів для квадратичної функції

Методи спряжених напрямів, якщо говорити «неформально», базуються на ідеї мінімізації квадратичної функції за скінченну кількість кроків. Визначимо метод формально.

Мінімізуємо квадратичну функцію

$$f(x) = \frac{1}{2}\langle Ax, x\rangle + \langle b, x\rangle,$$

де A – симетрична додатно визначена матриця розміру $n \times n$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^n$. Суть методу спряжених напрямів полягає в знаходженні таких напрямів h^0, h^1, …, h^{n-1}, що послідовність одновимірних мінімізацій уздовж h^j $(j = 1, 2 \ldots, n-1)$ мінімізує функцію f, тобто

$$f(x^n) = \min_{x \in \mathbb{R}^n} f(x),$$

де послідовність x^k, $0 \le k \le n$ визначають рекурентно:

$x^0 \in \mathbb{R}$ – довільне число;
$x^{k+1} = x^k + \alpha_k h^k$, $f(x^k + \alpha_k h^k) = \min\limits_{\alpha \in \mathbb{R}} f(x^k + \alpha h^k)$ $(0 \le k \le n)$.

Указану властивість має система напрямів $h^0, \ldots, h^{n-1}$, *взаємно (попарно) спряжених* відносно матриці A, тобто:

$$\left\langle Ah^i, h^j \right\rangle = 0 \quad \text{при } i \ne j.$$

Конкретні алгоритми мінімізації базуються на способах побудови системи взаємно спряжених напрямів.

У методі *спряжених напрямів першого порядку* система взаємно спряжених напрямів будується за правилом:

$$h^0 = -f'(x^0);$$
$$h^k = -f'(x^k) + \beta_{k-a} h^{k-1}, \quad \beta_{k-1} = \frac{\langle f'(x^k), Ah^{k-1} \rangle}{\langle h^{k-1}, Ah^{k-1} \rangle}. \qquad (4.1)$$

Відомо (див., наприклад, [3]), що побудовані за співвідношеннями (4.1) вектори $h^0, \ldots, h^{n-1}$ взаємно спряжені, а градієнти $f'(x^0), \ldots, f'(x^{n-1})$ попарно ортогональні. Метод спряжених градієнтів (як окремий випадок методу спряжених напрямів) забезпечує відшукання точки мінімуму функції f не пізніше ніж на n-му кроці.

Варіант №6:

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

У першій та другій лабороторних роботах ми вже знайшли розв'язок для цієї задачі:

$f(-0.0000625; 0.125) = -0.0625$

Наша функція є квадратичною, отже метод має знаходити точний розв'язок для нашої функції за n кроків. В нашому випадку n (розмірність простору) рівна 2, тому метод має не більше 2 (максимум 3) ітерації.

Наша функція є квадратичною, тому можемо записати матрицю A для нашої неї:

$A = \begin{pmatrix} 1 & 0.0005 \\ 0.0005 & 4 \end{pmatrix}$

Будемо використовувати метод спряжених напрямків 1 порядку – метод спряжених градієнтів. На кожному кроці ми будемо мати інший напрямок, який будемо рахувати по формулі: $h_0 = -f'(x^0)$ та $h_k = f'(x^k) + \beta_{k-1} * h_{k-1}$, при $k > 0$

Коефіцієнт β_{k-1} знаходимо з формули: $\beta_{k-1} = \frac{(f'(x^k),\ A*h_{k-1})}{(h_{k-1},\ A*h_{k-1})}$

Наступне наближення ми знаходимо за формулою:

$$x^{k+1} = x^k + \alpha_k * h_k$$

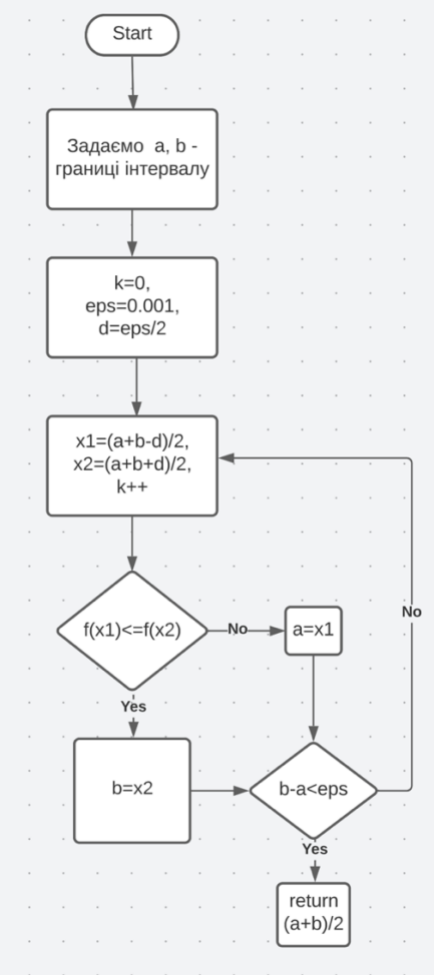
де α_k ми знаходимо, вирішуючи задачу одновимірної мінімізації:

$\min_{\alpha}(f\ (x^k + \alpha\ * h_k))$

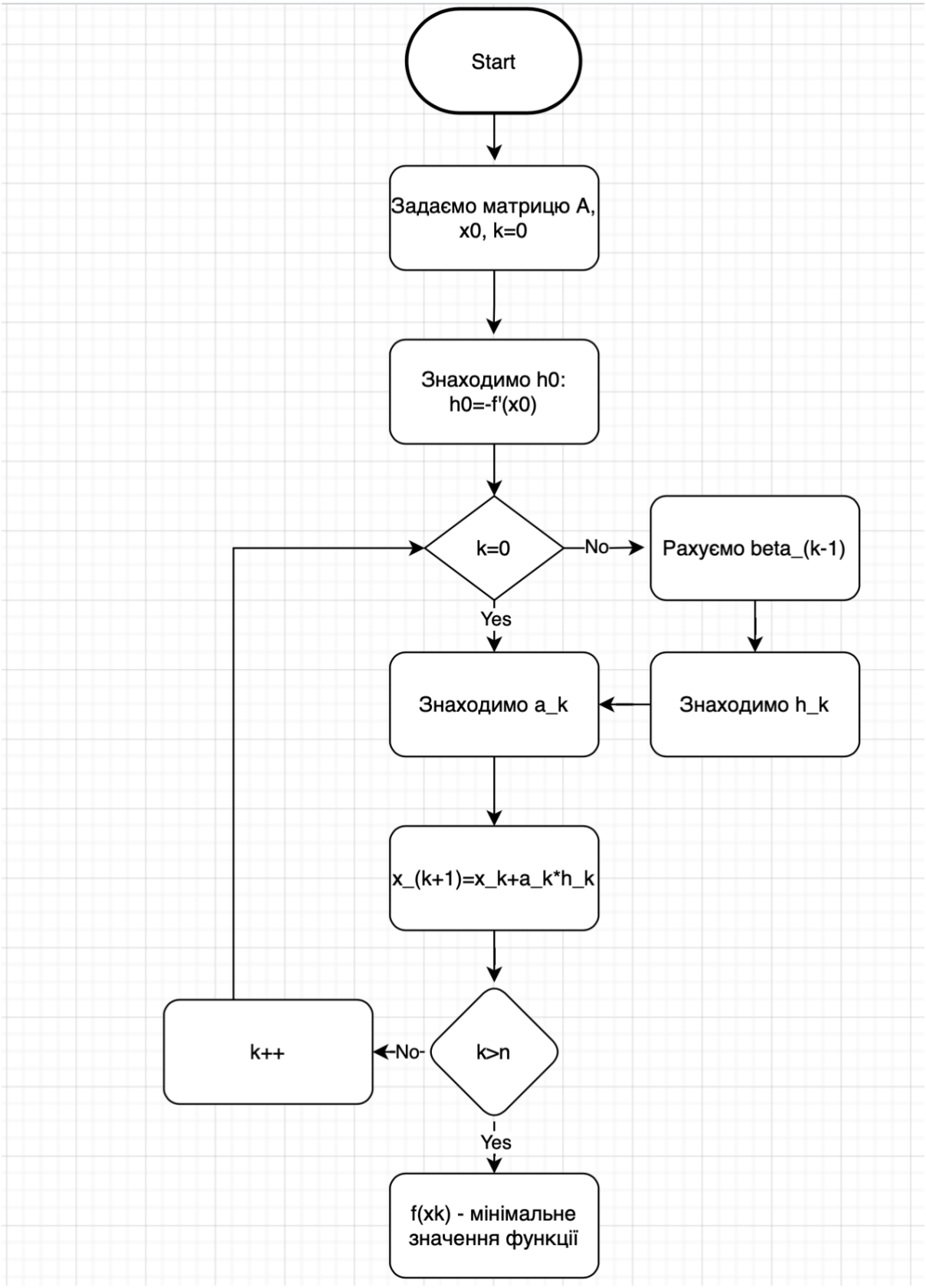
Знайдені напрямки будуть взаємноспряженими, тобто $(Ah_i, h_j) = 0$, при $i \neq j$

Вирішувати задачу одновимірної мінімізації (задля знаходження α_k) будемо за допомогою методу дихотомії:
Обираємо відрізок [a;b] та робимо припущення, що функція на цьому відрізку є унімодальною (має тільки один екстремум на цьому інтервалі). У цьому методі ми рахуємо середину інтервалу [a;b] (точка c) та знаходимо x1=c-d/2 та x2=c+d/2. Ми додаємо та віднімаємо похибку дельта (є(0;eps)) від середини відрізку, аби x1 та x2 залишалися на деякій відстані від середини та не було проблеми, що алгоритм застряг в точці мінімуму функції. Отже ми на кожній інтерації зменшуємо інтервал у 2 рази та відкидаємо той відрізок, де f буде більшим. Робимо такі дії, допоки двожина інтервалу не стане мешою за eps. Повертаємо середину відрізка.

Блок-схема алгоритму метода дихотомії:



Блок-схема алгоритму:



Для критерію зупинки можна взяти k>n, аби перевірити, чи дійсно метод сходиться за не більше, ніж n+1 ітерацію. Але для даної функції ми вже знаємо розв’язок, тому за умову зупинки можемр взяти if(|f(x[k+1]+0.0625|<eps) break;

Код програми:

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

   float T[1000][2];
   float A[2][2], B[2], h[10][2], bet[10];



   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 find_a(int k){//метод дихотомії

    float a=0,b=1,d,e=0.00001, Aa, AA,f1,f2;
    d=e/2;

    do{

        Aa=(a+b-d)/2;
        AA=(a+b+d)/2;

        f1=func(T[k][0] + Aa * h[k][0], T[k][1] + Aa * h[k][1]);
        f2=func(T[k][0] + AA * h[k][0], T[k][1] + AA * h[k][1]);


        if(f1 <=f2){
            b=AA;
        }

        else{
            a=Aa;
        }

        if(b-a<e)break;

    }while(1);

    return (a+b)/2;
}
```

```
int main() {
    float eps;
    eps=0.001;
    float a,x,y,aa,bb;
    int k=0;

    A[0][0]=1;
    A[1][1]=4;
    A[0][1]=0.0005;
    A[1][0]=0.0005;


    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;

    h[0][0]=-1*det_x(T[0][0], T[0][1]);
    h[0][1]=-1*det_y(T[0][0], T[0][1]);

    while(1){

        if(k!=0){
```

```
            aa=A[0][0] * h[k-1][0] + A[0][1] * h[k-1][1] ;
            bb=A[1][0] * h[k-1][0] + A[1][1] * h[k-1][1] ;



   bet[k-1] = (det_x(T[k][0],T[k][1]) * aa + det_y(T[k][0],T[k][1]) * bb ) /(h[k-1][0] * aa + h[k-
1][1] * bb );



            h[k][0] = -1 * det_x(T[k][0], T[k][1]) + bet[k-1] * h[k-1][0];
            h[k][1] = -1 * det_y(T[k][0], T[k][1]) + bet[k-1] * h[k-1][1];



        }

        a=find_a(k);



        T[k+1][0]=T[k][0]+a*h[k][0];
        T[k+1][1]=T[k][1]+a*h[k][1];

   //
        cout<<"k="<<k<<" a="<<a<<" x="<<T[k][0]<<" y="<<T[k][1]<<"
f="<<func(T[k][0],T[k][1])<<endl;
   //       <<" h_x="<<h[k][0]<<" h_y="<<h[k][1]<<" h_z="<<h[k][2]<<" bet="<<bet[k-
1]<<endl;



        if(abs(func(T[k+1][0],T[k+1][1])+0.0625)<eps){
```

```
            cout<<"k="<<k+1<<" a="<<a<<" x="<<T[k+1][0]<<" y="<<T[k+1][1]<<"
f="<<func(T[k+1][0],T[k+1][1])<<endl;
            break;
        }
   //       if(k>2)break;


        k++;
      }


      cout<<fixed;
      cout.precision(4);
      cout<<endl<<endl<<"Кількість ітерацій:  "<<k+1<<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:

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


k=0 a=0.124947 x=0 y=0 f=0
k=1 a=0.124947 x=0 y=0.124947 f=-0.0625


Кількість ітерацій: 1
F(0.0000; 0.1249) = -0.0625
```

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


k=0 a=0.13063 x=10 y=-10 f=509.9
k=1 a=0.478426 x=7.38863 y=0.580036 f=55.3618
k=2 a=0.478426 x=-6.62804e-05 y=0.125025 f=-0.0625


Кількість ітерацій: 2
F(-0.0001; 0.1250) = -0.0625
```

```
Введіть початкову координату x: 100
Введіть початкову координату y: 50


k=0 a=0.147091 x=100 y=50 f=19955
k=1 a=0.424879 x=70.5675 y=-8.69266 f=5290.1
k=2 a=0.130375 x=0.0177917 y=0.160212 f=-0.0572212
k=3 a=0.130375 x=0.000188308 y=0.125108 f=-0.0624999


Кількість ітерацій: 3
F(0.0002; 0.1251) = -0.0625
```

```
Введіть початкову координату x: 1000
Введіть початкову координату y: 1000


k=0 a=0.130753 x=1000 y=1000 f=5e+06
k=1 a=0.477633 x=738.495 y=-46.02 f=553858
k=2 a=0.126297 x=0.0238037 y=0.225296 f=-0.0216909
k=3 a=0.126297 x=-0.00123382 y=0.125146 f=-0.0624985


Кількість ітерацій: 3
F(-0.0012; 0.1251) = -0.0625
```

Бачимо,що метод дійсна знаходить мінімальне значення функції (яке співпало для всіх початкових точок) за не більше ніж 2 кроки для малої початкової точки, та за 3 кроки для великої початкової точки. Точку мінімуму він знайшов з заданою точністю eps=0.001.

Зробимо тепер програму для пошуку точку мінімуму методом спряжених градієнтів, але для тривимірної функції.

Візьмемо 12 варіант:
$f(x,y,z) = 2x^2 + 8y^2 + 3z^2 + 0.01xz - x - y$

Запишемо матрицю А:

$$A = \begin{pmatrix} 2 & 0 & 0.005 \\ 0 & 8 & 0 \\ 0.005 & 0 & 3 \end{pmatrix}$$

Для даної задачі ми вже не знаємо розв'язку, тому за умову зупинки візьмемо k>n , де n=3

Код програми:

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


   float T[1000][3];
   float A[3][3], B[3], h[10][3], bet[10];



   float func(float x,float y, float z){
      float f;
      f=2*pow(x,2)+8*pow(y,2)+3*pow(z,2)+0.01*x*z-x-y
      return f;
   }


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

```
}

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

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



float find_a(int k){//метод дихотомии

    float a=0,b=1,d,e=0.00001, Aa, AA,f1,f2;
    d=e/2;

    do{

        Aa=(a+b-d)/2;
        AA=(a+b+d)/2;

        f1=func(T[k][0] + Aa * h[k][0], T[k][1] + Aa * h[k][1], T[k][2] + Aa * h[k][2]);
        f2=func(T[k][0] + AA * h[k][0], T[k][1] + AA * h[k][1], T[k][2] + AA * h[k][2]);



        if(f1<=f2){
```

```
            b=AA;
        }

        else{
            a=Aa;
        }

        if(b-a<e)break;

    }while(1);

    return (a+b)/2;
}



int main() {
    float eps;
    eps=0.01;
    float a,x,y,z, aa,bb,cc;
    int k=0;

    A[0][0]=2;
    A[1][1]=8;
    A[2][2]=3;
    A[0][2]=0.005;
    A[2][0]=0.005;
    A[0][1]=0;
    A[1][0]=0;
    A[2][1]=0;
    A[1][2]=0;
```

```
//   B[0]=-1;
//   B[1]=-1;
//   B[2]=0;

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

    cout<<endl<<endl;

    T[0][0]=x;
    T[0][1]=y;
    T[0][2]=z;

    h[0][0]=-1*det_x(T[0][0], T[0][1], T[0][2]);
    h[0][1]=-1*det_y(T[0][0], T[0][1], T[0][2]);
    h[0][2]=-1*det_z(T[0][0], T[0][1], T[0][2]);

    while(1){

        if(k!=0){

            aa=A[0][0] * h[k-1][0] + A[0][1] * h[k-1][1] + A[0][2] * h[k-1][2];
            bb=A[1][0] * h[k-1][0] + A[1][1] * h[k-1][1] + A[1][2] * h[k-1][2];
            cc=A[2][0] * h[k-1][0] + A[2][1] * h[k-1][1] + A[2][2] * h[k-1][2];
```

```
    bet[k-1] = (det_x(T[k][0],T[k][1],T[k][2]) * aa + det_y(T[k][0],T[k][1],T[k][2]) * bb +
det_z(T[k][0],T[k][1],T[k][2]) * cc) /(h[k-1][0] * aa + h[k-1][1] * bb + h[k-1][2] * cc);



            h[k][0] = -1 * det_x(T[k][0], T[k][1], T[k][2]) + bet[k-1] * h[k-1][0];
            h[k][1] = -1 * det_y(T[k][0], T[k][1], T[k][2]) + bet[k-1] * h[k-1][1];
            h[k][2] = -1 * det_z(T[k][0], T[k][1], T[k][2]) + bet[k-1] * h[k-1][2];

        }

        a=find_a(k);



        T[k+1][0]=T[k][0]+a*h[k][0];
        T[k+1][1]=T[k][1]+a*h[k][1];
        T[k+1][2]=T[k][2]+a*h[k][2];
    //
        cout<<"k="<<k<<" a="<<a<<" x="<<T[k][0]<<" y="<<T[k][1]<<" z="<<T[k][2]<<"
f="<<func(T[k][0],T[k][1],T[k][2])<<endl;
    //      <<" h_x="<<h[k][0]<<" h_y="<<h[k][1]<<" h_z="<<h[k][2]<<" bet="<<bet[k-
1]<<endl;

    //      if((abs(T[k+1][0]-T[k][0])<eps)&&(abs(T[k+1][1]-T[k][1])<eps)&&(abs(T[k+1][2]-
T[k][2])<eps))break;
    //      if(abs(func(T[k+1][0],T[k+1][1],T[k+1][2])-func(T[k][0],T[k][1],T[k][2]))<eps)break;



        if(k>3)break;



        k++;
```

```
    }


    cout<<fixed;
    cout.precision(3);
    cout<<endl<<endl<<"Кількість ітерацій: "<<k<<endl<<"F("<<T[k+1][0]<<";
"<<T[k+1][1]<<"; "<<T[k+1][2]<<") = ";
    cout<<func(T[k+1][0], T[k+1][1], T[k+1][2])<<endl<<endl;


  }
```

Результати роботи програми:

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


k=0 a=0.0999452 x=0 y=0 z=0 f=0
k=1 a=0.15604 x=0.0999452 y=0.0999452 z=0 f=-0.1
k=2 a=4.40734e-06 x=0.249661 y=0.0625162 z=-0.000155961 f=-0.15625
k=3 a=4.40734e-06 x=0.249661 y=0.0625162 z=-0.000155968 f=-0.15625
k=4 a=4.40734e-06 x=0.249661 y=0.0625162 z=-0.000155968 f=-0.15625


Кількість ітерацій: 4
F(0.250; 0.063; -0.000) = -0.156
```

(Бачимо, що точне мінімальне значення f знайшлося ще на другій ітерації)

```
Введіть початкову координату x: 1
Введіть початкову координату y: 2
Введіть початкову координату z: 3


k=0 a=0.0745661 x=1 y=2 z=3 f=58.03
k=1 a=0.141472 x=0.774064 y=-0.31156 z=1.65707 f=9.76285
k=2 a=0.246024 x=0.429057 y=0.0634727 z=-0.0246162 f=-0.090407
k=3 a=4.40734e-06 x=0.250579 y=0.0621757 z=-0.000687752 f=-0.156249
k=4 a=4.40734e-06 x=0.250579 y=0.0621757 z=-0.000687746 f=-0.156249


Кількість ітерацій: 4
F(0.251; 0.062; -0.001) = -0.156
```

(Бачимо, що метод знайшов точний розв’язок на 3 ітерації)

```
Введіть початкову координату x: -500
Введіть початкову координату y: 1000
Введіть початкову координату z: 300


k=0 a=0.0637095 x=-500 y=1000 z=300 f=8.768e+06
k=1 a=0.208347 x=-372.581 y=-19.3527 z=185.323 f=383364
k=2 a=0.196117 x=-56.3899 y=0.438448 z=-50.548 f=14110.9
k=3 a=0.0622485 x=0.252708 y=0.0188442 z=-0.00279236 f=-0.140972
k=4 a=4.40734e-06 x=0.252388 y=0.0623217 z=-0.00159232 f=-0.156235


Кількість ітерацій: 4
F(0.252; 0.062; -0.002) = -0.156
```

Отже метод дійсно знаходить розв’язок не більше, ніж за n (=3) ітерацій, та за n+1 (=4) для поганих початкових точок. Значення функції співпали для кожної взятої початкової точки, а координати точки знайшлись з заданою точністю eps.

Тепер візьмемо не квадратичну функцію, наприклад:

Варіант 21:
$f(x,y) = (y - x^2)^2 + 100 * (1 - x)^2$

Це функція Розенброка.
Бачимо, що $(y - x^2)^2$ буде >0 для любих x,y ,та $100 * (1 - x)^2$ теж буде >0 при будь-яких x,y
Так, як ми сумуємо ці доданки, то можемо зробити висновок, що функція приймає лише не від'ємні значення, тому мінімум функції буде ≥ 0.
Значення функції буде рівне 0, якщо взяти точку (1;1):
$(1 - 1)^2 + 100 * (1 - 1)^2 = 0$
Отже робимо висновок, що функція приймає найменше значення – 0 у точці (1;1)

Ця функція погана тим, що при великих x програма не зможе знайти точку мінімума, адже якщо, наприклад, взяти x=1000,
то $f \approx (1000^2)^2 + 100 * (1000)^2 > 1000000000000$, що є занадто великим числом для компілятора.

Так, як функція не є квадратичною, то метод не буде скінченокроковим (не знайде точне значення за скінчену кількість кроків). Також в нас немає матриці A, тобто ми не можемо знайти коефіцієнт бета за минулою формулою. Тому ми будемо використовувати наступні формули:
1)Для Методу спряжених градієнтів без оновлення кроку:
Використовуємо ті самі формули задля знаходження x^k, h^0, h^k, a^k, але коєфіцієнт бета знаходимо за формулою:

$$\beta_{k-1} = \frac{(f'(x^k),\ \ f'(x^k) - f'(x^{k-1}))}{\left||f'(x^{k-1})|\right|^2}$$

В чисельнику маємо скалярний добуток векторів, а у знаменнику – норму в квадраті, яка буде рівна сумі квадратів координат.

2)Для Методу спряжених градієнтів з оновленням кроку використовуємо ті самі формули, але при k={0, n, 2n, 3n, 4n, …}, тобто при k={0, 2, 4, 6, 8, …}:
$\beta_{k-1} = 0$ (функція двовимірна, тому n=2)

Код програми:

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

float T[1000][2];
float A[2][2], B[2], h[1000][2], bet[1000];


float func(float x,float y){
    float f;
    f=pow(y-pow(x,2),2)+100*pow(1-x,2);
    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 find_a(int k){

    float a=0,b=1,d,e=0.00001, Aa, AA,f1,f2;
    d=e/2;
```

```
    do{

        Aa=(a+b-d)/2;
        AA=(a+b+d)/2;

        f1=func(T[k][0] + Aa * h[k][0], T[k][1] + Aa * h[k][1]);
        f2=func(T[k][0] + AA * h[k][0], T[k][1] + AA * h[k][1]);



        if(f1<=f2){
            b=AA;
        }

        else{
            a=Aa;
        }

        if(b-a<e)break;

    }while(1);

    return (a+b)/2;//*
}



int main() {
    float eps;
    eps=0.00001;
    float a,x,y,aa[2],bb;
```

```
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;

h[0][0]=-1*det_x(T[0][0], T[0][1]);
h[0][1]=-1*det_y(T[0][0], T[0][1]);

while(1){

    if(k!=0){

        if(k%2!=0){//*

            aa[0] = det_x(T[k][0],T[k][1]) - det_x(T[k-1][0],T[k-1][1]);
            aa[1] = det_y(T[k][0],T[k][1]) - det_y(T[k-1][0],T[k-1][1]);

            bb= pow(det_x(T[k-1][0],T[k-1][1]),2) + pow(det_y(T[k-1][0],T[k-1][1]),2);

            bet[k-1] = (det_x(T[k][0],T[k][1]) * aa[0] + det_y(T[k][0],T[k][1]) * aa[1]) /bb ;

        }else bet[k-1]=0;//*
```

```
            h[k][0] = -1 * det_x(T[k][0], T[k][1]) + bet[k-1] * h[k-1][0];
            h[k][1] = -1 * det_y(T[k][0], T[k][1]) + bet[k-1] * h[k-1][1];


        }

        a=find_a(k);


        T[k+1][0]=T[k][0]+a*h[k][0];
        T[k+1][1]=T[k][1]+a*h[k][1];

//
        cout<<"k="<<k<<" a="<<a<<" bet="<<bet[k-1]<<" x="<<T[k][0]<<" y="<<T[k][1]<<"
f="<<func(T[k][0],T[k][1])<<endl;
//         <<" h_x="<<h[k][0]<<" h_y="<<h[k][1]<<" h_z="<<h[k][2]<<" bet="<<bet[k-
1]<<endl;




        if(abs(func(T[k+1][0],T[k+1][1])-func(T[k][0],T[k][1]))<eps)break;

        k++;
    }


    cout<<fixed;
```

```
    cout.precision(3);
    cout<<endl<<endl<<"Кількість ітерацій: "<<k+1<<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:

Без оновлення кроку:

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


k=0 a=0.00490627 bet=0 x=0 y=0 f=100
k=1 a=0.514944 bet=0.000242668 x=0.981252 y=0 f=0.962239
k=2 a=0.00485286 bet=0.977024 x=0.990802 y=0.991633 f=0.00855878
k=3 a=0.0625079 bet=-0.000487636 x=1.00001 y=1.00067 f=4.3168e-07


Кількість ітерацій: 4
F(1.000; 1.001) = 0.000
```

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


k=0 a=0.00162565 bet=0 x=10 y=10 f=16200
k=1 a=0.51475 bet=-0.000114162 x=1.22128 y=10.2926 f=82.3551
k=2 a=0.00500164 bet=1.85029 x=0.889247 y=1.22111 f=1.41183
k=3 a=0.179378 bet=0.00594026 x=1.00172 y=1.05371 f=0.00282293
k=4 a=0.000248547 bet=0.00546096 x=0.999993 y=1.00002 f=6.39397e-09


Кількість ітерацій: 5
F(1.000; 1.000) = 0.000



Введіть початкову координату x: 1
Введіть початкову координату y: 100


k=0 a=0.0161939 bet=0 x=1 y=100 f=9801
k=1 a=0.455294 bet=0.0445775 x=7.41272 y=96.7928 f=5863.25
k=2 a=0.0331082 bet=13.0927 x=-3.59117 y=54.658 f=3851.91
k=3 a=0.00386105 bet=5.03701 x=-3.52708 y=11.7774 f=2049.88
k=4 a=0.0140005 bet=-0.163238 x=0.0425472 y=-13.4061 f=271.443
k=5 a=0.0050703 bet=-0.2413 x=0.578645 y=1.87585 f=20.1287
k=6 a=0.0518879 bet=-0.0313068 x=0.977159 y=0.524772 f=0.237129
k=7 a=0.00519237 bet=-0.0475006 x=0.99927 y=1.00227 f=6.71432e-05
k=8 a=4.40734e-06 bet=0.0012249 x=0.999998 y=0.999957 f=1.87028e-09


Кількість ітерацій: 9
F(1.000; 1.000) = 0.000



Введіть початкову координату x: -10
Введіть початкову координату y: -1000


k=0 a=0.000218029 bet=0 x=-10 y=-1000 f=1.2221e+06
k=1 a=0.50559 bet=0.00211551 x=0.0729551 y=-999.52 f=999137
k=2 a=0.00516948 bet=0.319137 x=-4.23129 y=15.5936 f=2741.98
k=3 a=0.285621 bet=-0.0170543 x=1.36555 y=18.9299 f=304.584
k=4 a=0.00690516 bet=15.5648 x=1.83332 y=6.03774 f=76.6072
k=5 a=0.0166898 bet=0.0125418 x=0.994034 y=1.14952 f=0.0296139
k=6 a=0.00627192 bet=-0.0711216 x=0.999212 y=0.995947 f=6.8183e-05
k=7 a=0.000492686 bet=-0.0032284 x=0.999997 y=1.00008 f=8.52233e-09


Кількість ітерацій: 8
F(1.000; 1.000) = 0.000
```

З оновленням кроку:

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


k=0 a=0.00490627 bet=0 x=0 y=0 f=100
k=1 a=0.514944 bet=0.000242668 x=0.981252 y=0 f=0.962239
k=2 a=0.0048109 bet=0 x=0.990802 y=0.991633 f=0.00855878
k=3 a=0.520346 bet=0.000667149 x=0.99984 y=0.991537 f=6.88653e-05
k=4 a=4.40734e-06 bet=0 x=1 y=1 f=1.0411e-09


Кількість ітерацій: 5
F(1.000; 1.000) = 0.000



Введіть початкову координату x: 10
Введіть початкову координату y: 10


k=0 a=0.00162565 bet=0 x=10 y=10 f=16200
k=1 a=0.51475 bet=-0.000114162 x=1.22128 y=10.2926 f=82.3551
k=2 a=0.00485668 bet=0 x=0.889247 y=1.22111 f=1.41183
k=3 a=0.51919 bet=0.000296483 x=1.00426 y=1.21693 f=0.0452412
k=4 a=0.00476131 bet=0 x=1.0002 y=1.0004 f=4.05876e-06


Кількість ітерацій: 5
F(1.000; 1.000) = 0.000



Введіть початкову координату x: 1
Введіть початкову координату y: 100


k=0 a=0.0161939 bet=0 x=1 y=100 f=9801
k=1 a=0.455294 bet=0.0445775 x=7.41272 y=96.7928 f=5863.25
k=2 a=0.0239377 bet=0 x=-3.59117 y=54.658 f=3851.91
k=3 a=0.676263 bet=0.0517219 x=4.02991 y=52.659 f=2244.37
k=4 a=0.00427685 bet=0 x=2.37027 y=0.496052 f=214.001
k=5 a=0.515631 bet=0.00040115 x=0.990483 y=0.539867 f=0.203706
k=6 a=0.00480327 bet=0 x=1.00372 y=0.996967 f=0.00149001
k=7 a=0.259864 bet=-0.000270747 x=0.999942 y=0.997068 f=8.2683e-06


Кількість ітерацій: 8
F(1.000; 0.999) = 0.000
```

```
Введіть початкову координату x: -10
Введіть початкову координату y: -1000


k=0 a=0.000218029 bet=0 x=-10 y=-1000 f=1.2221e+06
k=1 a=0.50559 bet=0.00211551 x=0.0729551 y=-999.52 f=999137
k=2 a=0.00516567 bet=0 x=-4.23129 y=15.5936 f=2741.98
k=3 a=0.395506 bet=0.0003568 x=1.37546 y=15.6175 f=202.49
k=4 a=0.00490245 bet=0 x=1.6964 y=4.761 f=52.0436
k=5 a=0.518064 bet=0.00297291 x=1.07623 y=4.74254 f=13.4281
k=6 a=0.00481472 bet=0 x=0.976383 y=1.0229 f=0.0606168
k=7 a=0.519972 bet=6.37007e-05 x=1.00043 y=1.02223 f=0.000475278
k=8 a=0.000492686 bet=0 x=1 y=1 f=5.13026e-10


Кількість ітерацій: 9
F(1.000; 1.000) = 0.000
```

Висновок:

Отже бачимо, що значення функції співпали обома варіаціями методу,а також співпали з нашим аналітичним розв’язком. Бачимо, що метод без оновлення кроку для більшості взятих початкових точок виявився трішки швидшим, проте лише на 1 ітерацію.

Отже ми розглянули метод спряжених градієнтів для функцій з різними вимірами, а також для не квадратичної функції, та знайшли мінімальне значення для цієї функцій.