Лабораторна робота 7
Чисельні методи розв'язання задач одномірної оптимізації та задач безумовної оптимізації

7.1. Мета роботи

Вивчення чисельних методів одновимірної оптимізації та методів безумовної оптимізації для практичного розв'язання задачі пошуку точки оптимуму функції однієї змінної та задачі пошуку точки оптимуму функції декількох змінних, придбання навичок використання цих методів для розв'язання задач одномірної оптимізації та задач безумовної оптимізації із застосуванням комп'ютера.

7.2. Методичні вказівки по організації самостійної роботи

По темі лабораторної роботи студент повинен: *знати* загальне формулювання задачі одномірної оптимізації та задачі безумовної оптимізації; *уміти* розв'язувати ці задачі з використанням чисельних методів одновимірної чи безумовної оптимізації [4, 9].

7.2.1. Чисельні методи розв'язання задач одномірної оптимізації

Розглянемо нелінійну неперервну функцію $f(x)$ однієї змінної $x \in R^1$. Задача одновимірної оптимізації полягає в знаходженні точки $x^* \in R^1$, у якій функція $f(x)$ приймає мінімальне значення. У загальному випадку функція $f(x)$ може мати кілька точок мінімуму. Розглянуті нижче чисельні методи розв'язання задачі одновимірної оптимізації дозволяють знаходити одну точку мінімуму на заданому відрізку $[a,b]$. При цьому на відрізку повинна існувати тільки одна точка мінімуму.

Розглянемо кілька чисельних методів пошуку точки мінімуму функції $f(x)$ на відрізку $[a,b]$.

Метод половинного ділення (дихотомії). При пошуку точки мінімуму функції $\boldsymbol{f(x)}$ методом половинного ділення задаються відрізок $[a,b]$, на якому існує тільки одна точка мінімуму, і бажана точність $\varepsilon > 0$. На 0-му кроці методу $[a_0,b_0] = [a,b]$, на k-му кроці методу маємо поточний відрізок $[a_k,b_k]$. Далі визначаються дві точки $x_{k1} = \frac{a_k + b_k}{2} - \delta$ й $x_{k2} = \frac{a_k + b_k}{2} + \delta$ (відступають в обидва боки від середини відрізка $x_k = \frac{a_k + b_k}{2}$ на величину $\delta \le \frac{\varepsilon}{4}$) і перевіряється

умова $f(x_{k1}) < f(x_{k2})$. Якщо зазначена умова виконується, то $b_{k+1} = x_{k2}$, $a_{k+1} = a_k$. Якщо умова не виконується, то $a_{k+1} = x_{k1}$, $b_{k+1} = b_k$. Ділення відрізка навпіл триває доти, поки $|b_k - a_k| > \varepsilon$. При цьому справедлива наступна оцінка швидкості збіжності:

$$|b_k - a_k| \le \frac{1}{2^{\frac{k}{2}}} + \left(1 - \frac{b-a}{2^{\frac{k}{2}}}\right)\delta,$$

тобто метод збігається приблизно з геометричною швидкістю з коефіцієнтом $q = \frac{1}{\sqrt{2}}$.

Недоліком методу половинного ділення є те, що на кожній ітерації методу значення функції $f(x)$ потрібно обчислювати у двох точках x_{k1}, x_{k2}, а це може виявитися істотним, коли кожне обчислення значення функції $f(x)$ вимагає великих ресурсних витрат. Цей недолік усувається в методі золотого перетину.

Метод золотого перетину. При пошуку точки мінімуму функції $f(x)$ методом золотого перетину задаються відрізок $[a,b]$, на якому існує тільки одна точка мінімуму, і бажана точність $\varepsilon > 0$. На 0-му кроці методу $[a_0,b_0] = [a,b]$, на k-му кроці методу маємо поточний відрізок $[a_k,b_k]$. Далі на першій ітерації визначаються дві точки $x_{k1} = a_k + (1-\lambda)(b_k - a_k)$ й $x_{k1} = a_k + \lambda(b_k - a_k)$, де $\lambda = \frac{-1+\sqrt{5}}{2}$, і перевіряється умова $f(x_{k1}) < f(x_{k2})$. Якщо зазначена умова виконується, то $b_{k+1} = x_{k2}$, $a_{k+1} = a_k$, $x_{k2} = x_{k1}$, $x_{k1} = a_{k+1} + (1-\lambda)(b_{k+1} - a_{k+1})$. Якщо умова не виконується, то $a_{k+1} = x_{k1}$, $b_{k+1} = b_k$, $x_{k1} = x_{k2}$, $x_{k2} = a_{k+1} + \lambda(b_{k+1} - a_{k+1})$. Таким чином, на всіх наступних ітераціях (починаючи із другої) значення функції $f(x)$ додатково обчислюється тільки в одній точці, друга ж точка (а значить і значення функції в ній) просто перевизначається. Золотий перетин відрізка триває доти, поки $|b_k - a_k| > \varepsilon$. При цьому справедлива наступна оцінка швидкості збіжності:

$$|b_k - a_k| \le \frac{1}{(1+\lambda)^{k-1}}(b-a),$$

тобто метод збігається з геометричною швидкістю з коефіцієнтом $q = \frac{1}{1+\lambda} = \lambda$.

7.2.2. Чисельні методи розв'язання задач безумовної оптимізації

Розглянемо нелінійну неперервну функцію $f(x)$ декількох змінних $x \in R^n$. Задача безумовної оптимізації полягає в знаходженні точки $x^* \in R^n$, у якій функція $f(x)$ приймає мінімальне значення. Коротко задача безумовної оптимізації записується у вигляді:

$$f(x) \to \min,\ x \in R^n \qquad (7.1)$$

У загальному випадку функція $f(x)$ може мати кілька локальних точок мінімуму. Розглянуті нижче чисельні методи розв'язання задачі багатовимірної оптимізації дозволяють знаходити одну з локальних точок мінімуму.

Розглянемо кілька чисельних методів пошуку точки мінімуму функції $f(x)$, які в принципі мають одну загальну схему.

Узагальнена схема чисельних методів розв'язання задачі безумовної оптимізації

При пошуку точки мінімуму функції $f(x)$ чисельним методом задаються початкове наближення розв'язку $x^{(0)} \in R^n$ й бажана точність $\varepsilon > 0$ розв'язку по градієнту. На k-му кроці методу маємо поточне наближення розв'язку $x^{(k)} \in R^n$. Наступне наближення розв'язку $x^{(k+1)} \in R^n$ визначається за формулою:

$$x^{(k+1)} = x^{(k)} + \alpha_k h^{(k)}, \qquad (7.2)$$

де напрямок пошуку $h^{(k)} \in R^n$ визначається **конкретним** чисельним методом, а кроковий множник $\alpha_k > 0$ зазвичай визначається одним із способів:

а) як точка мінімуму функції однієї змінної $\varphi_k(\alpha) = f(x^{(k)} + \alpha h^{(k)})$, $\alpha > 0$ (повний пошук за напрямком);

б) за алгоритмом дроблення кроку (описаний нижче).

Обчислення тривають доти, поки не виконається умова $\left\| f'(x^k) \right\| \le \varepsilon$ (хоча можливі й інші критерії «останову»).

Алгоритм дроблення кроку для визначення крокового множника в чисельних методах розв'язання задачі безумовної оптимізації

В алгоритмі дроблення кроку при визначенні крокового множника α_k в чисельних методах розв'язання задачі безумовної оптимізації, що підпадають під схему (7.2), задаються параметри:

ρ – максимальне значення кроку, $\rho > 0$,

γ – коефіцієнт рівня спадання функції, $\gamma \in (0, \frac{1}{2})$,

λ – коефіцієнт дроблення кроку, $\lambda \in (0,\ 1)$.

При цьому вектор $h^{(k)}$ повинен бути напрямком спадання функції $f(x)$, тобто задовольняти умові $\langle f'(x^{(k)}), h^{(k)} \rangle < 0$.

На початку вибирається $\alpha = \rho$ й перевіряється нерівність:

$$f(x^{(k)} + \alpha h^{(k)}) \leq f(x^{(k)}) + \alpha\gamma \langle f'(x^{(k)}), h^{(k)} \rangle. \qquad (7.3)$$

Якщо нерівність (7.3) не виконується, то α зменшується шляхом множення на λ, тобто $\alpha = \lambda\alpha$, і знову перевіряється (7.3). Якщо нерівність (7.3) виконується, то кроковий множник α_k приймається рівним поточному значенню α.

Якщо вектор $h^{(k)}$ є напрямком спадання функції $f(x)$, то описаний процес дроблення кроку буде скінченим, тобто нерівність (7.3) виконається через скінчене число кроків.

Чисельні методи розв'язання задачі безумовної оптимізації

Метод найшвидшого спуску. Метод будує ітераційну послідовність $\{x^{(k)}\}$, $k = 0,\ 1,\ 2, \ldots$, наближень розв'язку за схемою (7.2), де напрямок пошуку $h^{(k)}$ визначається за формулою:

$$h^{(k)} = -f'(x^{(k)}), \qquad (7.4)$$

кроковий множник $\alpha_k > 0$ визначається, як точка мінімуму функції однієї змінної $\varphi_k(\alpha) = f(x^{(k)} + \alpha h^{(k)})$. Обчислення триває доти, поки не виконається умова $\|f'(x^k)\| \leq \varepsilon$.

Недоліком методу найшвидшого спуску є його досить низька швидкість збіжності для «яружних» (погано обумовлених) функцій.

Метод сполучених градієнтів. Метод будує ітераційну послідовність $\{x^{(k)}\}$, $k = 0,\ 1,\ 2,\ldots$, наближень розв'язку за схемою (7.2), де напрямок пошуку $h^{(k)}$ визначається за формулою:

$$\begin{cases} h^{(k)} = -f'(x^{(k)}), \quad k \in \{0, n,\ 2n,\ 3n, \ldots\} \\ h^{(k)} = -f'(x^{(k)}) + \beta_{k-1} h^{(k-1)}, \quad k \notin \{0, n,\ 2n,\ 3n, \ldots\} \end{cases} \tag{7.5}$$

коефіцієнт β_{k-1} визначається за однією з формул:

$$\textit{а)} \quad \beta_{k-1} = \frac{\langle f'(x^{(k)}), f'(x^{(k)}) - f'(x^{(k-1)}) \rangle}{\|f'(x^{k-1})\|^2},$$

$$\textit{б)} \quad \beta_{k-1} = \frac{\|f'(x^{(k)})\|^2}{\|f'(x^{(k-1)})\|^2} \tag{7.6}$$

Кроковий множник $\alpha_k > 0$ в (7.2) може бути визначений одним із способів:

а) як точка мінімуму функції однієї змінної $\varphi_k(\alpha) = f(x^{(k)} + \alpha h^{(k)})$;

б) за алгоритмом дроблення кроку (7.3).

Обчислення тривають доти, поки не виконається умова $\|f'(x^k)\| \le \varepsilon$.

Перевагою методу сполучених градієнтів є його висока швидкість збіжності для нормальних (не «яружних») функцій.

Недоліком методу сполучених градієнтів є його низька швидкість збіжності для «яружних» (погано обумовлених) функцій.

Метод Ньютона з регулюванням кроку. Метод будує ітераційну послідовність $\{x^{(k)}\}$, $k = 0,\ 1,\ 2,\ldots$, наближень розв'язку за схемою (7.2), де напрямок пошуку $\boldsymbol{h^{(k)}}$ визначається за формулою:

$$h^{(k)} = -\left[f''(x^{(k)})\right]^{-1} f'(x^{(k)}), \tag{7.7}$$

кроковий множник $\alpha_k > 0$ в (7.2) зазвичай визначається за алгоритмом дроблення кроку (7.3), у якому $\rho = 1$. У класичному методі Ньютона кроковий множник α_k завжди приймається рівним 1.

Перевагою методу Ньютона з регулюванням кроку є його висока (квадратична) швидкість збіжності навіть для «яружних» функцій.

Недоліком методу Ньютона є необхідність на кожній ітерації обчислення й обернення матриці других похідних цільової функції (метод другого порядку).

7.3. Контрольні приклади

Приклад 1. Знайти точку мінімуму функції $f(x)=x^2+10\cos(x)$ методом: половинного ділення з точністю 10^4. Відрізок $[a,b]$, що містить точку мінімуму знайти самостійно.

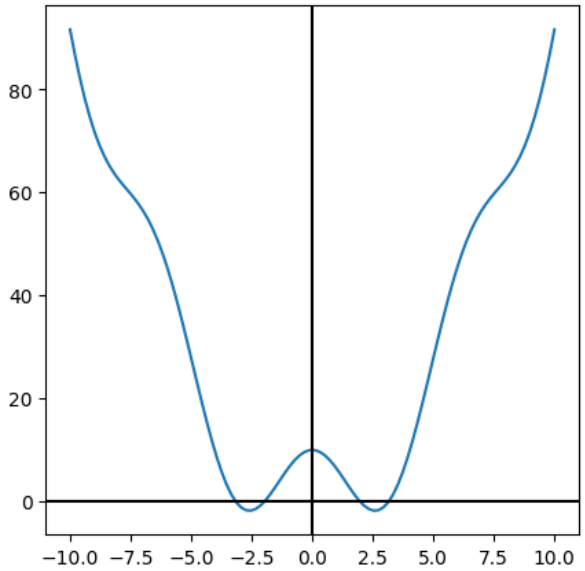
Розв'язання. Спочатку визначимо відрізок, що містить розв'язок. Зробимо це графічно.

```
import numpy as np
import math
import matplotlib.pyplot as plt

def Fun(x):
  y = x**2 + 10*math.cos(x)
  return y

a = -10
b = 10
n = 100
x = np.linspace(a, b, n)
y = [0]*n
for i in range(n):
  y[i] = Fun(x[i])

plt.figure(figsize=(5,5))
plt.plot(x, y)
plt.axhline(0, color='k') #x-axis line
plt.axvline(0, color='k') #y-axis line
plt.show()
```



Судячи із графіку і виду функції, вона має дві точки мінімуму, причому вони симетричні відносно точки початку координат. Знайдемо праву точку мінімуму, тому відрізок $[a,b]$ візьмемо рівним $[0, 5]$.

Реалізуємо метод половинного ділення у вигляді процедури *BiSection*, параметр методу δ візьмемо рівним $\frac{\varepsilon}{4}$:

```
def BiSection(f, a, b, eps):
  delta = eps/4
  while ((b-a) > eps):
    x = (a+b)/2
    x1 = x - delta
    x2 = x + delta
    if f(x1) < f(x2):
      b = x2
    else:
      a = x1
  return (a+b)/2
```

Знайдемо першу (праву) точку мінімуму.

```
a = 0
b = 5
eps = 0.0001
X1 = BiSection(Fun, a, b, eps)
print(X1)
```

Output:

```
2.5957288705825805
```

Друга (ліва) точка мінімуму може бути знайдена за допомогою тієї ж процедури.

```
 = -5
b = 0
eps = 0.0001
X2 = BiSection(Fun, a, b, eps)
print(X2)
```

Output:

```
-2.5957288705825805
```

Приклад 2. Знайти точку мінімуму функції двох змінних $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ (функція Розенброка) методом Ньютона з регулюванням кроку з точністю 10^{-4}. Початкове наближення $x^{(0)} = (-1{,}2,\ 1)^T$.

Розв'язання. Спочатку визначимо всі необхідні функції для реалізації методу Ньютона:

Fun – функція обчислення значень цільової функції;
FunX – функція обчислення градієнта цільової функції;
FunXX – функція обчислення матриці Гессе цільової функції.

```
import numpy as np

def Fun(x):
  y = 100*(x[1]-x[0]**2)**2 + (1 - x[0])**2
  return y

def FunX(x):
  z = [0]*2
  z[0] = 200*(x[1]-x[0]**2)*(-2*x[0]) - 2*(1 - x[0])
  z[1] = 200*(x[1]-x[0]**2)
  return z

def FunXX(x):
  Z = np.zeros((2, 2)) # Z is matrix (nrow=2, ncol=2)
  Z[0,0] = 1200*x[0]**2 - 400*x[1] + 2
  Z[1,0] = -400*x[0]
  Z[0,1] = -400*x[0]
  Z[1,1] = 200
  return Z
```

Реалізуємо алгоритм дроблення кроку (7.3) у вигляді процедури *AlDrStep*:

```
def AlDrStep(Fun, xk, hk, fk, scphp, ro, kvF):
  Lambda = 0.8
  Gamma = 0.4

  alfa = ro
  x1 = xk + alfa*hk
  f1 = Fun(x1)
  kvF = kvF + 1
  while (f1 > (fk + alfa*Gamma*scphp)):
    alfa = alfa*Gamma
    x1 = xk + alfa*hk
    f1 = Fun(x1)
    kvF = kvF + 1

  return [x1, f1, kvF]
```

Реалізуємо метод Ньютона з регулюванням кроку у вигляді процедури *MetodNewton:*

```
from numpy import linalg as LA
```

```
def MetodNewton(Fun, FunX, FunXX, x0, eps, kmax):
  xk = x0
  fk = Fun(xk)
  gk = FunX(xk)
```

```
  kvF = 1

  # Цикл ітераційного процесу
  k = 0
  while ((LA.norm(gk) > eps) and (k < kmax)):
    Hk = FunXX(xk)
    hk = -np.dot(LA.inv(Hk), gk)
    scphp = np.dot(gk, hk)
    [xk, fk, kvFk] = AlDrStep(Fun, xk, hk, fk, scphp, 1, kvF)
    kvF = kvFk
    gk = FunX(xk)
    k = k + 1

  return [xk, k, kvF]
```

Застосуємо цю процедуру для вирішення задачі:

```
x0 = [-1.2, 1]
eps = 0.0001
kmax = 50
[x1, k, kvF] = MetodNewton(Fun, FunX, FunXX, x0, eps, kmax)

print('x=', x1)
print('fun=', Fun(x1))
print('jac=', FunX(x1))
print('nit=', k)
print('nfev=', kvF)
print('njev=', k)
```

Output:

```
x= [0.9999995  0.99999898]
fun= 2.8774032595104205e-13
jac= [6.5113876416974795e-06, -3.7581214584747613e-06]
nit= 21
nfev= 28
njev= 21
```

Для порівняння застосуємо для вирішення цієї ж задачі процедуру ***minimize*** пакета *scipy.optimize* з квазі-ньтоновським методом 'BFGS':

```
from scipy.optimize import minimize

res = minimize(Fun, x0, method='BFGS')
print(res)
```

Output:

```
message: Optimization terminated successfully.
  success: True
   status: 0
      fun: 2.0243313190213974e-11
        x: [ 1.000e+00  1.000e+00]
      nit: 32
      jac: [ 2.710e-07 -1.562e-07]
```

```
 hess_inv: [[ 5.021e-01  1.005e+00]
            [ 1.005e+00  2.015e+00]]
      nfev: 117
      njev: 39
```

Як видно з результатів, функція ***MetodNewton*** знашла точку мінімуму точніше і швидче, ніж функція ***minimize***, але це тому, що наша программа обчислювала градієнт і гесіан цільової функції аналітично, а функція ***minimize*** – градієнт чисельно, а гесіан апроксимувала.

Для візуальної інтерпретації розв'язку задачі побудуємо графік функції в околиці точки мінімуму (1,1) з застосуванням бібліотеки **rgl**:

```
import matplotlib.pyplot as plt

x1left = 0.8
x1rigth = 1.1
x2left = 0.8
x2rigth = 1.1
N = 30

x1 = np.linspace(X1left, X1rigth, N)
x2 = np.linspace(X2left, X2rigth, N)

Y = np.zeros((N, N)) # Y is matrix (nrow=N, ncol=N)
xv = [0]*2
for i in range(N):
  xv[0] = x1[i]
  for j in range(N):
    xv[1] = x2[j]
    Y[i,j] = Fun(xv)


plt.figure(figsize=(10,10))
ax = plt.axes(projection='3d')
ax.contour3D(x1, x2, Y, 50, cmap='plasma')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y');
```

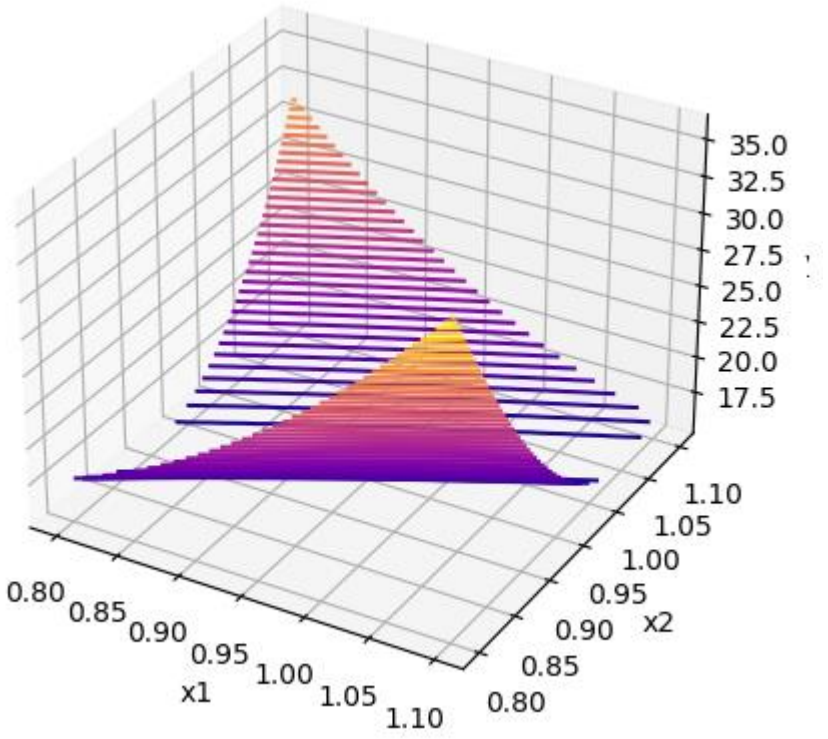


Рис. 7.1. Графік функції Розенброка в околиці точки мінімуму (1,1)

З графіку (рис. 7.1.) видно, що в околиці точки мінімуму (1,1) функції Розенброка має «яружний» вигляд (анімація Lab_07.mp4), тобто дуже полога вдовж дна та крута в поперек.

7.4. Порядок виконання роботи і варіанти завдань

7.4.1. Зміст звіту

У теоретичній частині роботи необхідно коротко описати:

- постановку задачі одновимірної оптимізації;
- методи розв'язання задачі одновимірної оптимізації;
- постановку задачі безумовної оптимізації;
- методи розв'язання задачі безумовної оптимізації.

У практичній частині роботи необхідно:

- запрограмувати у вигляді окремих модулів методи половинного ділення і золотого перетину;
- запрограмувати у вигляді окремих модулів методи найшвидшого спуску, сполучених градієнтів і Ньютона;
- привести тексти складених програм;
- знайти точку мінімуму функції однієї змінної 2-ма методами: половинного ділення (дихотомії) і золотого перетину. Точність

розв'язку – 10^{-4}, відрізок $[a,b]$, що містить точку мінімуму знайти самостійно.

- знайти точку мінімуму функції декількох змінних 3-м методами: найшвидшого спуску, сполучених градієнтів і Ньютона (точність розв'язку 10^{-4});
- порівняти трудомісткість і швидкість збіжності методів.

7.4.2. Варіанти індивідуальних завдань

Варіант	Функція
1	$f(x)=(x-\exp(-x))^2$
2	$f(x)=(x-\cos(x))^2$
3	$f(x)=(x-x^2-1)^2$
4	$f(x)=(x-2\exp(-x))^2$
5	$f(x)=(x-\exp(-3x))^2$
6	$f(x)=(x-3\cos(x))^2$
7	$f(x)=(x-\exp(-3x^2))^2$
8	$f(x)=(x-tg(x))^2$
9	$f(x)=(x-\cos(2x))^2$
10	$f(x)=(x-tg(2x)-1)^2$
11	$f(x)=(x-\exp(-3x)+1)^2$
12	$f(x)=(x-\exp(-x^2))^2$
13	$f(x)=(x-\ln(x)+2)^2$
14	$f(x)=(x-\exp(-3x)-2)^2$
15	$f(x)=(x^2-\exp(-x^2))^2$

Варі-ант	Функція $f(x)$	Початкове наближен-ня	Точка мінімуму x^*	Значення $f(x^*)$
1	$f(x)=6x_1+2x_1^2-2x_1x_2+2x_2^2$	(-1,-1)	(-2,-1)	-6
2	$f(x)=x_1+x_2^2+\left(\frac{x_1+x_2-10}{3}\right)^2$	(-1,-1)	(5,0.5)	7.5
3	$f(x)=(x_1-1)^2+100(x_1-x_2)^2$	(3,4)	(1,1)	0

4	$f(x)=5(x_1-3)^2+(x_2-5)^2$	(0,0)	(3,5)	0
5	$f(x)=x_1^2-x_1x_2+x_2^2$	(1,2)	(0,0)	0
6	$f(x)=9x_1^2+16x_2^2-90x_1-128x_2$	(0,3)	(5,4)	-481
7	$f(x)=2x_1^2+2x_2^2+2x_1x_2-4x_1-6x_2$	(1,1)	(1/3,4/3)	-14/3
8	$f(x)=x_1^2-x_1x_2+x_2^2-2x_1+x_2$	(3,5)	(1,0)	-1
9	$f(x)=5x_1^2+4x_1x_2+x_2^2-16x_1-12x_2$	(1,1)	(-4,14)	-152
10	$f(x)=2x_1^2+2x_2^2+x_1x_2-11x_1-8x_2$	(-3,-5)	(2,3)	-23
11	$f(x)=x_1-x_2+2x_1^2+2x_1x_2+x_2^2$	(1,1)	(-1,1.5)	-1.25
12	$f(x)=x_1^2+x_2^2+x_1x_2$	(1,1)	(0,0)	0
13	$f(x)=x_1^2+16x_2^2$	(2,2)	(0,0)	0
14	$f(x)=(1-x_1)^2+(x_1-x_2)^2$	(-5,-8)	(1,1)	0
15	$f(x)=x_1^2+4x_2^2+1$	(3,5)	(0,0)	1

7.5. Контрольні запитання

1. Сформулюйте постановку задачі одновимірної оптимізації.
2. Які є чисельні методи для розв'язання задачі одновимірної оптимізації? Чим вони відрізняються?.
3. Сформулюйте постановку задачі безумовної оптимізації.
4. Які є чисельні методи для розв'язання задачі безумовної оптимізації? Чим вони відрізняються?