Численные методы, ЛР-5

9.1.8

Задание

Задача 9.1 Методом Ньютона найти минимум и максимум унимодальной на отрезке $[a,b]$ функции $f(x)$ с точностью $\varepsilon = 10^{-6}$. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

Вариант

9.1.8	$\sin x - 2\cos x$	1	4

Теория

Формула, используемая для метода Ньютона

$$x^{(n+1)} = x^{(n)} - \frac{f'(x^{(n)})}{f''(x^{(n)})}, \quad n \geq 0.$$

Решение

```
def f(x):
    return np.sin(x) - 2 * np.cos(x)


def df(x):
    return np.cos(x) + 2 * np.sin(x)
```

```
def ddf(x):
    return -np.sin(x) + 2 * np.cos(x)
```

```
def newton_method(f, df, ddf, a, b, eps):
    x = (a + b) / 2
    iterations = 0

    while True:
        iterations += 1
        x_new = x - df(x) / ddf(x)
        if abs(x_new - x) < eps:
            break
        x = x_new
    return x, iterations



a, b = 1, 4
eps = 1e-6

x, iterations = newton_method(f, df, ddf, a, b, eps)
y = f(x)

print(f"Экстремум функции достигается в x = {x}, f(x) = {y}, ite
```

Результаты

9.3.3

Задание

Задача 9.3. Функция $x(t)$ задана неявно уравнением $F(x,t)=0$, $t_1 \le t \le t_2$, $x_1 \le x \le x_2$. Построить график зависимости функции $x(t)$ на заданном отрезке $[t_1, t_2]$ и найти ее минимум и максимум с точностью $\varepsilon = 10^{-6}$.

Вариант

9.3.3	$e^{2x+t} - e^{t^2} - 2\cos x$	0	2	0	2	Фибоначчи

Теория

Вычисление ошибки

В результате выполнения $N-1$ шагов отрезок локализации уменьшается в $F_{N+1}/2$ раз, а точка $x^{(N-1)}$ оказывается центральной для последнего отрезка локализации $[a^{(N-1)}, b^{(N-1)}]$. Поэтому для $x^{(N-1)}$ справедлива следующая оценка погрешности:

$$|\bar{x} - x^{(N-1)}| \leqslant \frac{1}{F_{N+1}} \Delta. \tag{9.11}$$

Решение

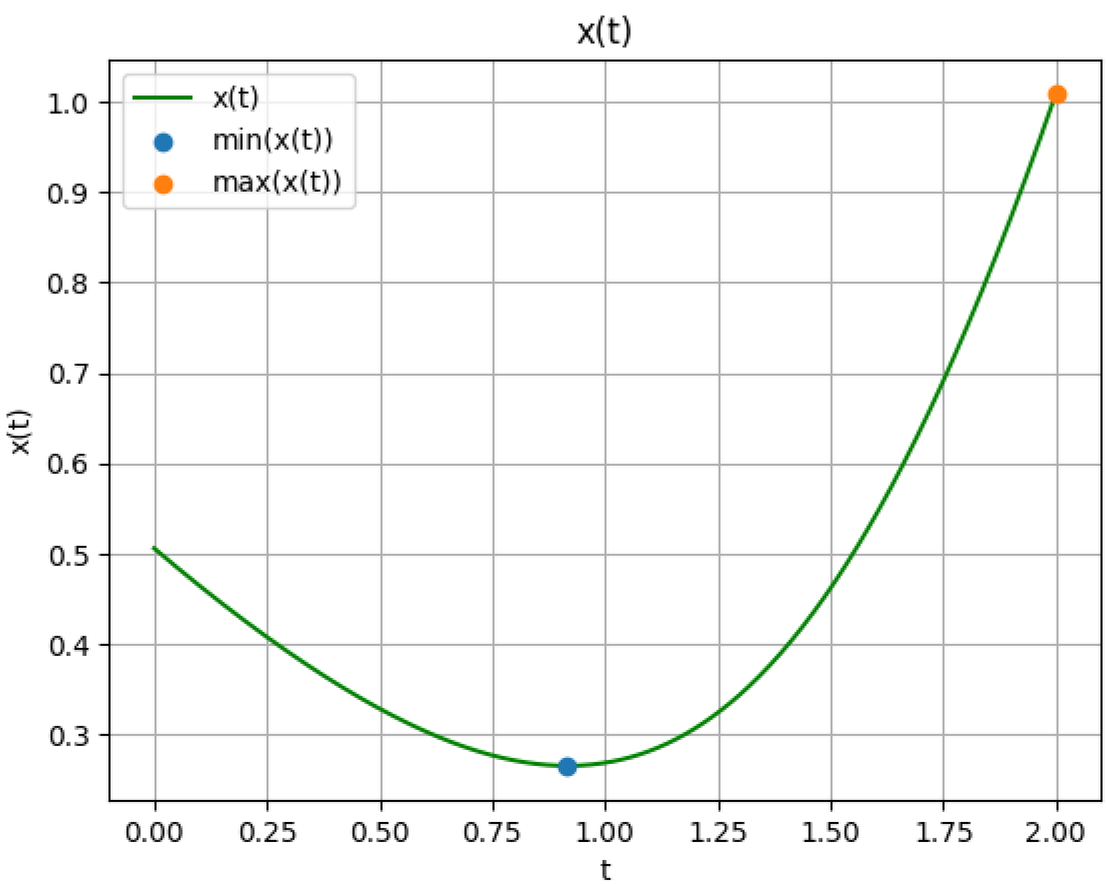
```
def F(x, t):
    return np.exp(2 * x + t) - np.exp(t ** 2) - 2 * np.cos(x)


def x_t(t):
    """
    Функция для получения функции зависимости x(t) из F(x(t), t)
    """
    x_initial_guess = 0
    x_solution = fsolve(F, x_initial_guess, args=(t))
    return x_solution[0]


@cache
```

```
def calc_fibonacci(n):
    """
    Функция для получения чисел Фибоначи
    """
    if n == 0:
        return 1
    if n == 1:
        return 1
    return calc_fibonacci(n - 1) + calc_fibonacci(n - 2)


def fibonacci_minimization(f, a, b, eps):
    """
    Функция для минимизации функции через метод Фибоначи
    """

    # ищем n по формуле
    n = 1
    while (b - a) / calc_fibonacci(n + 1) >= eps:
        n += 1

    # алгоритм для метода Фибоначи
    for k in range(n - 1):

        alpha = a + (calc_fibonacci(n - k - 1) / calc_fibonacci
        beta = a + (calc_fibonacci(n - k) / calc_fibonacci(n -

        if f(alpha) <= f(beta):
            a, b = a, beta
        else:
            a, b = alpha, b

    return b


epsilon = 1e-6
```

```
t1, t2 = 0, 2

t_min = fibonacci_minimization(lambda t: x_t(t), t1, t2, epsilon
t_max = fibonacci_minimization(lambda t: -x_t(t), t1, t2, epsilo

print(f"Минимум функции x(t) достигается при t = {t_min}, x(t) =
print(f"Максимум функции x(t) достигается при t = {t_max}, x(t)
```

Результаты

```
Минимум функции x(t) достигается при t = 0.9165990683599063, x(t) = 0.26480794519252865
Максимум функции x(t) достигается при t = 2, x(t) = 1.0096529685277487
```



9.5.8

Задание

Задача 9.5. Найти минимум функции 2-х переменных $f(x,y)$ с точностью $\varepsilon = 10^{-6}$ на прямоугольнике $[x_1, x_2] \times [y_1, y_2]$.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Задать указанную в варианте функцию $f(x,y)$.
2. Построить графики функции и поверхностей уровня $f(x,y)$.
3. По графикам найти точки начального приближения к точкам экстремума.
4. Используя встроенные функции, найти экстремумы функции с заданной точностью (см. *ПРИЛОЖЕНИЕ 9.В*).

Вариант

9.5.8	$x^2 + 2y^2 + e^x \cos(y-1)$	-2	1	-1	1

Решение

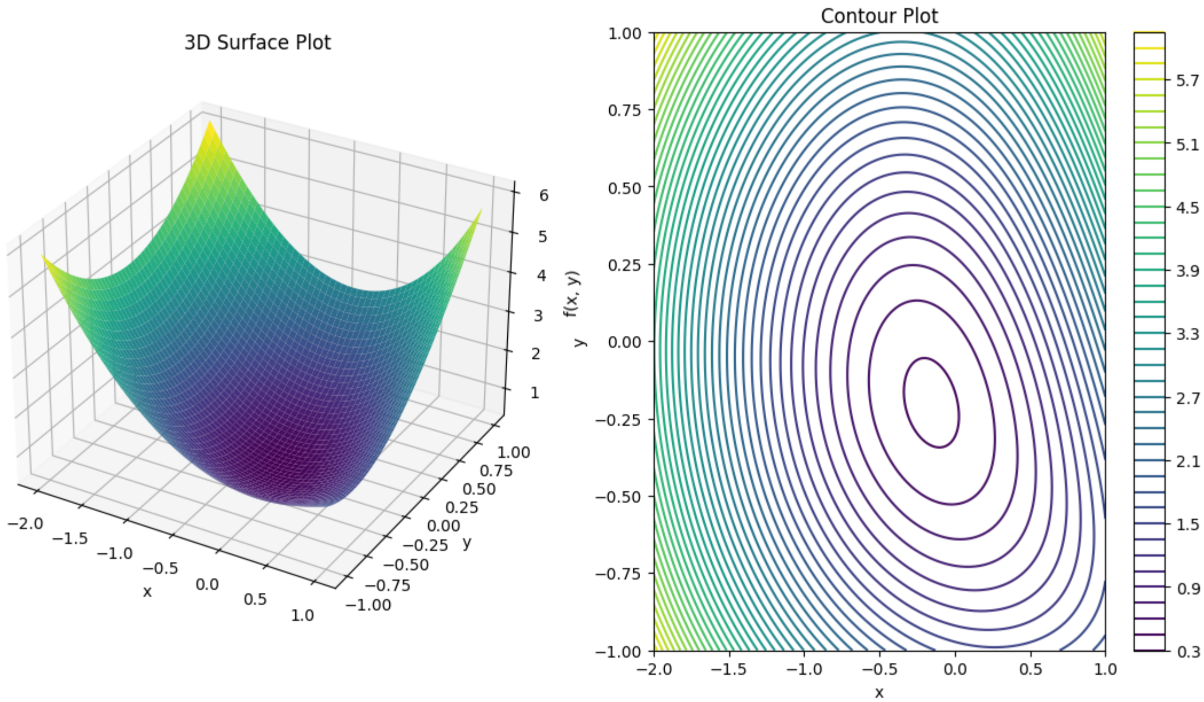
```
def f(x):
    return x[0] ** 2 + 2 * x[1] ** 2 + np.exp(x[0]) * np.cos(x[1

x1, x2 = -2, 1
y1, y2 = -1, 1
```

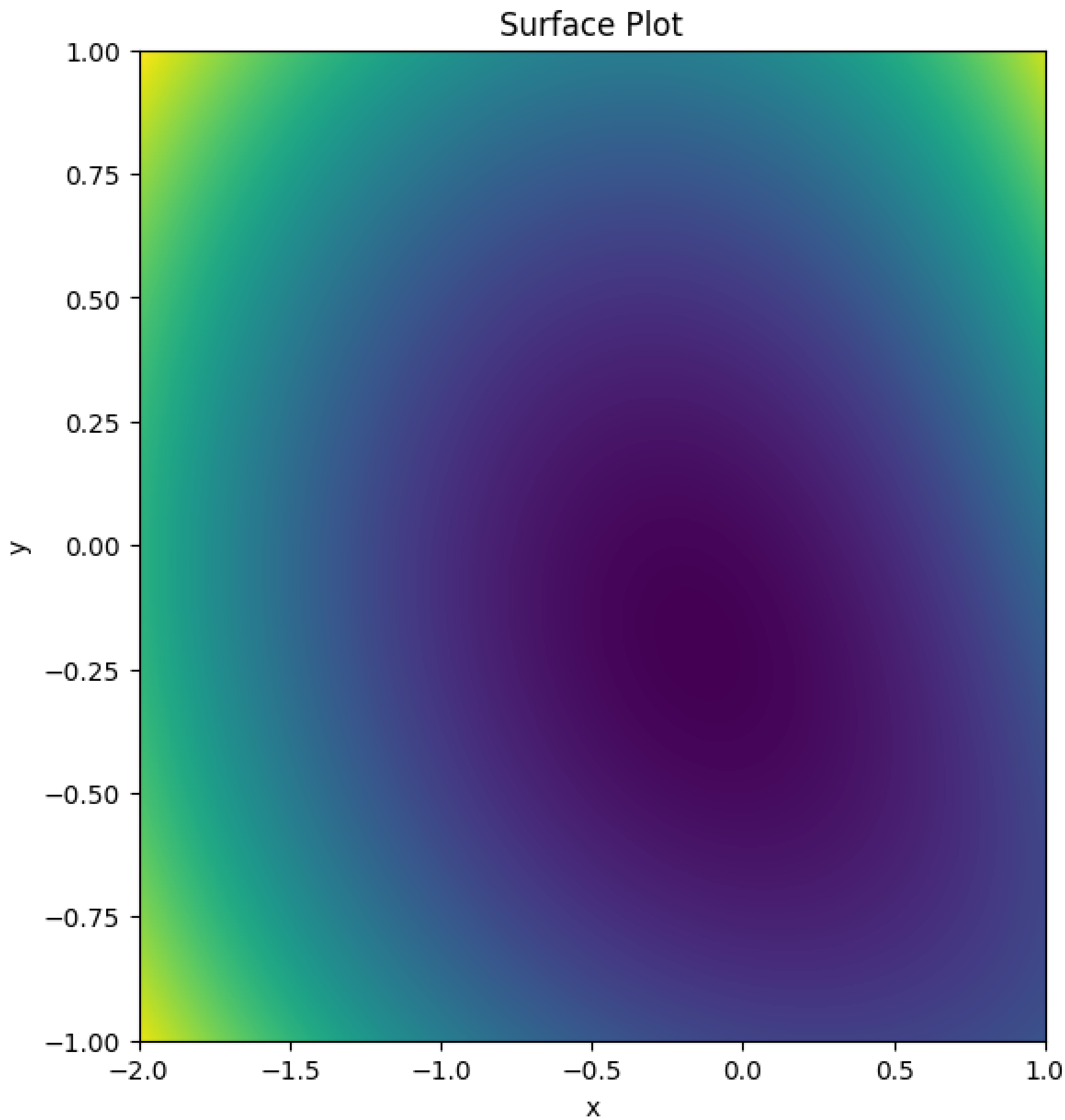
```
X = np.linspace(x1, x2, 400)
Y = np.linspace(y1, y2, 400)
X, Y = np.meshgrid(X, Y)
Z = f([X, Y])

fig = plt.figure(figsize=(14, 7))

ax1 = fig.add_subplot(121, projection='3d')
ax1.plot_surface(X, Y, Z, cmap='viridis')
ax1.set_title('3D Surface Plot')
ax1.set_xlabel('x')
```

```
ax1.set_ylabel('y')
ax1.set_zlabel('f(x, y)')

ax2 = fig.add_subplot(122)
contour = ax2.contour(X, Y, Z, 50)
ax2.set_title('Contour Plot')
ax2.set_xlabel('x')
ax2.set_ylabel('y')
plt.colorbar(contour, ax=ax2)

plt.show()
```

```
fig = plt.figure(figsize=(14, 7))
ax3 = fig.add_subplot(122)
ax3.imshow(Z, extent=[x1, x2, y1, y2], origin='lower', aspect='a
ax3.set_title('Surface Plot')
ax3.set_xlabel('x')
ax3.set_ylabel('y')
```

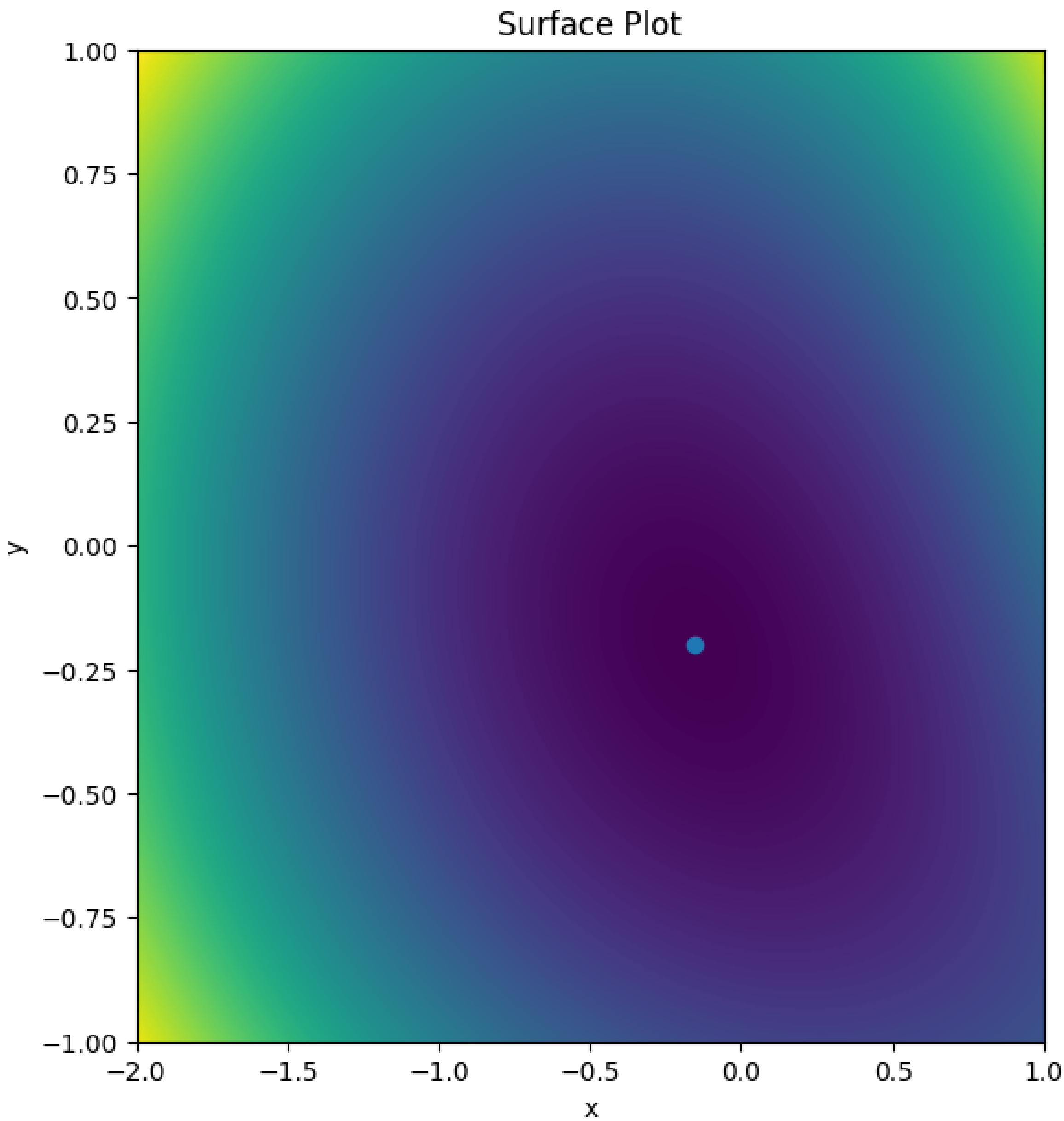
Результаты


3D Surface Plot
x
y
f(x, y)
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
1
2
3
4
5
6
Contour Plot
x
y
0.3
0.9
1.5
2.1
2.7
3.3
3.9
4.5
5.1
5.7




```
initial_guess = [0, 0]

result = minimize(f, initial_guess, tol=1e-6)
min_x, min_y = result.x
min_f = result.fun

print(f"Минимальное значение функции: {min_f} в точке ({min_x},
```

```
Минимальное значение функции: 0.41435312789245854 в точке (-0.1553346040458045, -0.19944312500711342)
```



9.6.8

Задание

Задача 9.6. Указанным в индивидуальном варианте методом найти минимум квадратичной функции $f(x,y)=a_{11}x^2+2a_{12}xy+a_{22}y^2+2a_{13}x+2a_{23}y$ с точностью $\varepsilon=10^{-6}$. Для решения задачи одномерной минимизации использовать метод Ньютона. Построить график функции f. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

Вариант

9.6.8	1	0.5	2.5	-3.5	-6.5	Сопряженных градиентов

Теория

§ 10.5. Метод сопряженных градиентов

Метод Ньютона и квазиньютоновские методы, обсуждавшиеся в предыдущем параграфе, весьма эффективны как средство решения задач безусловной минимизации. Однако они предъявляют довольно высокие требования к объему используемой памяти ЭВМ. Это связано с тем, что выбор направления поиска $\boldsymbol{p}^{(n)}$ требует решения систем линейных уравнений, а также с возникающей необходимостью хранения матриц типа $B^{(n)}$, $H^{(n)}$. Поэтому при больших m использование этих методов может оказаться невозможным. В существенной степени от этого недостатка избавлены методы сопряженных направлений.

1. Понятие о методах сопряженных направлений. Рассмотрим задачу минимизации квадратичной функции

$$F(\boldsymbol{x}) = \frac{1}{2}(A\boldsymbol{x}, \boldsymbol{x}) - (\boldsymbol{b}, \boldsymbol{x}) \qquad (10.33)$$

с симметричной положительно определенной матрицей A Напомним, что для ее решения требуется один шаг метода Ньютона и не более чем m шагов квазиньютоновского метода Методы сопряженных направлений также позволяют найти точку минимума функции (10.33) не более чем за m шагов. Добиться этого удается благодаря специальному выбору направлений поиска.

Будем говорить, что ненулевые векторы $\boldsymbol{p}^{(0)}$, $\boldsymbol{p}^{(1)}$, ..., $\boldsymbol{p}^{(m-1)}$ являются *взаимно сопряженными* (относительно матрицы A), если $(A\boldsymbol{p}^{(n)}, \boldsymbol{p}^{(l)}) = 0$ для всех $n \neq l$.

Под *методом сопряженных направлений* для минимизации квадратичной функции (10.33) будем понимать метод

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha_n \boldsymbol{p}^{(n)} \quad (n = 0, 1, 2, ..., m-1),$$

в котором направления $\boldsymbol{p}^{(0)}$, $\boldsymbol{p}^{(1)}$, ..., $\boldsymbol{p}^{(m-1)}$ взаимно сопряжены, а шаги

$$\alpha_n = -\frac{(\boldsymbol{g}^{(n)}, \boldsymbol{p}^{(n)})}{(A\boldsymbol{p}^{(n)}, \boldsymbol{p}^{(n)})}$$

получаются как решение задач одномерной минимизации:

$$\varphi_n(\alpha_n) = \min_{\alpha \geqslant 0} \varphi_n(\alpha), \quad \varphi_n(\alpha) = F(\boldsymbol{x}^{(n)} + \alpha \boldsymbol{p}^{(n)}).$$

Теорема 10.4. *Метод сопряженных направлений позволяет найти точку минимума квадратичной функции* (10 33) *не более чем за* m *шагов.*

Методы сопряженных направлений отличаются один от другого способом построения сопряженных направлений. Наиболее известным среди них является *метод сопряженных градиентов*

2. Метод сопряженных градиентов. В этом методе направления $\boldsymbol{p}^{(n)}$ строят по правилу

$$\boldsymbol{p}^{(0)} = -\boldsymbol{g}^{(0)}, \quad \boldsymbol{p}^{(n)} = -\boldsymbol{g}^{(n)} + \beta_{n-1}\boldsymbol{p}^{(n-1)}, \quad n \geqslant 1, \tag{10.34}$$

где

$$\beta_{n-1} = \frac{(A\boldsymbol{p}^{(n-1)}, \boldsymbol{g}^{(n)})}{(A\boldsymbol{p}^{(n-1)}, \boldsymbol{p}^{(n-1)})}. \tag{10.35}$$

4. Критерии окончания итераций. На практике часто используют следующие критерии окончания итераций:

$$|x^{(n+1)} - x^{(n)}| < \varepsilon_1, \qquad (10.16)$$

$$|f(x^{(n+1)}) - f(x^{(n)})| < \varepsilon_2, \qquad (10.17)$$

$$|f'(x^{(n)})| < \varepsilon_3, \qquad (10.18)$$

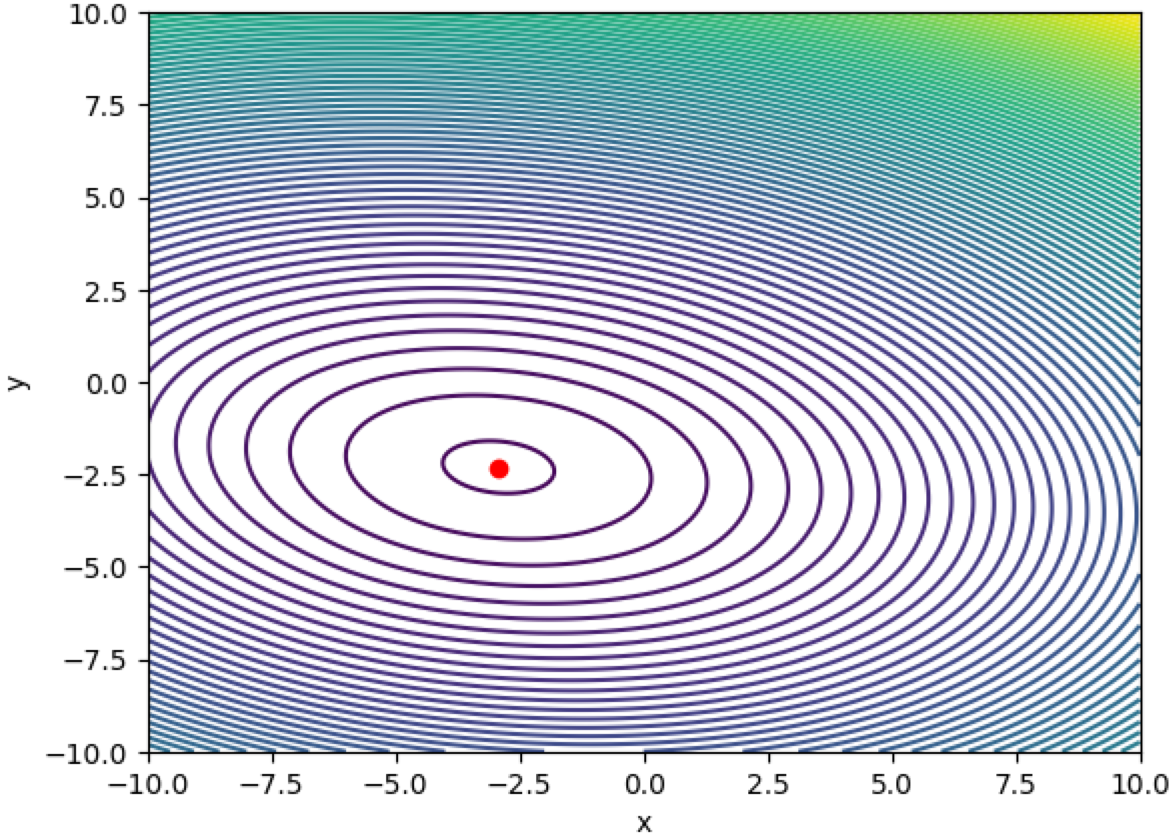
Решение

```
def conjugate_gradient(A, b, x0, epsilon):
    def grad_f(x):
        return A @ x - b

    x = x0
    g = grad_f(x)
    p_prev = None
    Ap_prev = None

    iterations = 0
    while np.linalg.norm(grad_f(x)) > epsilon:
        g = grad_f(x)
        beta_prev = np.dot(Ap_prev, g) / np.dot(Ap_prev, p_prev
        p = -g + beta_prev * p_prev if iterations > 0 else -g
        Ap = A @ p
        alpha = - np.dot(g, p) / np.dot(Ap, p) if np.linalg.norm
        x_new = x + alpha * p
        p_prev, Ap_prev = p, Ap
        x = x_new
        iterations += 1
        # print(f'it = {iterations}, x = {x}')
    return x, iterations
```

Результаты

```
A = 2 * np.array([[1, 0.25], [0.25, 2.5]])
b = 2 * np.array([-3.5, -6.5])
epsilon = 1e-6

def F(x):
    return 0.5 * ((A @ x) @ x) - (b @ x)

x0 = np.array([0, 0])
x_min, iterations = conjugate_gradient(A, b, x0, epsilon)

print("Точка минимума:", x_min)
print("Значение функции:", F(x_min))
print("Количество итераций:", iterations)
```

```
Точка минимума: [-2.92307692 -2.30769231]
Количество итераций: 2
```


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
y
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x