Student name: Abao Zhang (张宝)

Student number: 12332459

Homework 8

Question 1

(a)

求导

$$f'(x) = 1 - \frac{6}{x^3}$$
$$f''(x) = 1 + \frac{18}{x^4}$$

$f''(x) > 0$恒成立，故$f'(x)$单调递增，$f'(x)$有唯一零点$p = \sqrt[3]{6}$，所以$f(x)$在$p = \sqrt[3]{6}$有最小值

(b)

$$f'(x) = -\cos(x) - \cos(3x)$$
$$f''(x) = \sin(x) + 3\sin(3x)$$

$f''(x) > 0$在$x \in [0, 2]$恒成立，故$f'(x)$单调递增，$f'(x)$有唯一零点$p = \frac{\pi}{4}$，所以$f(x)$在$p = \frac{\pi}{4}$有最小值

Question 2

首先求导$f'(x) = e^x + 2 + x$，根据黄金比例搜索算法

$$c = ra + (1 - r)b$$
$$d = (1 - r)a + rb$$

其中$r = \frac{\sqrt{5}-1}{2}$，简单比较$f(c)$ 和$f(d)$大小进而更新a, b即可，使用代码 `question-2.py` 进行计算，核心代码如下

```
def golden_ratio_search(f, df, a, b):
    gr = (np.sqrt(5) - 1) / 2
    for i in range(3):
        print(f"a_{i}={a}, b_{i}={b}")
        c_i = a * gr + (1 - gr) * b
        d_i = a * (1 - gr) + gr * b
        if f(c_i) < f(d_i):
            a, b = a, d_i
        else:
            a, b = c_i, b
```

结果如下:

$$a_0 = -2.4, b_0 = -1.6$$
$$a_1 = -2.4, b_1 \approx -1.9056$$
$$a_2 = -2.2111, b_2 = -1.9056$$

Question 3

$f'(x) = -\cos(x) + x - 1$, $f'(0.8) < 0$故在$p_0 = 0.8$的右边

首先初始化h, p_0, p_1, p_2 计算对应的函数值y_1, y_2, y_3，使用下面公式计算新的h

$$h_{\min} = \frac{h(4y_1 - 3y_0 - y_2)}{4y_1 - 2y_0 - 2y_2}$$

使用$h_{\min}$作为h更新p_0, p_1, p_2，下面给出核心代码

```python
def quadratic_approx(f, a, b):
    p_0, p_1, p_2, h = a, (a + b) / 2, b, (b - a) / 2
    for i in range(3):
        print(f"iteration i={i}, p_0={p_0}, p_1={p_1}, p_2={p_2}, h={h}")
        y_0, y_1, y_2 = f(p_0), f(p_1), f(p_2)
        h = h * (4 * y_1 - 3 * y_0 - y_2) / (4 * y_1 - 2 * y_0 - 2 * y_2)
        p_0 = p_min = p_0 + h
        p_1 = p_0 + h
        p_2 = p_0 + h * 2

quadratic_approx(lambda x: -np.sin(x) - x + x * x / 2, 0.8, 1.6)
```

结果如下

$i = 0$

$$p_0 = 0.8, p_1 = 1.2, p_2 = 1.6, h = 0.4$$

$i = 1$

$$p_0 \approx 1.2796, p_1 \approx 1.7592, p_2 \approx 2.2387, h \approx 0.47958$$

$i = 2$

$$p_0 \approx 1.2808, p_1 \approx 1.2820, p_2 \approx 1.2832, h \approx 0.0012205$$

Question 4

```python
i = 1
a, b = 0, 1
while True:
    a, b = b, a + b
    if 10**-8 > (3.99 - 3.33) / a:
        print(i, a)
        break
    i = i + 1
```

$$F_{40} = 102334155$$

Question 5

设O为原点，根据平行四边形原理有

$$\overrightarrow{OR} = \overrightarrow{OW} + \overrightarrow{WB} + \overrightarrow{WG} = (-2, -4)$$

$$\overrightarrow{OM} = \overrightarrow{OW} + \frac{\overrightarrow{WB} + \overrightarrow{WG}}{2} = \left(\frac{3}{2}, -1\right)$$

$$\overrightarrow{OE} = \overrightarrow{OW} + \left(\overrightarrow{WB} + \overrightarrow{WG}\right)\frac{3}{2} = \left(-\frac{11}{2}, -7\right)$$

Question 6

$$\nabla f(x, y) = (2x - 3, 3y^2 - 3)$$

$$S = -\frac{\nabla f(x, y)}{\|\nabla f(x, y)\|}$$

核心步骤为求解γ，使得$f(P_k + \gamma S_k)$最小化，这里使用在 Question 3 中编写的函数。即每一步朝下降的地方走尽可能大的一步（而不是乘以学习率）。核心代码如下：

```
def quadratic_approx(f, a, b, tol=1e-5, max_iter=10000):
    p_0, p_1, p_2, h = a, (a + b) / 2, b, (b - a) / 2
    for i in range(max_iter):
        y_0, y_1, y_2 = f(p_0), f(p_1), f(p_2)
        h = h * (4 * y_1 - 3 * y_0 - y_2) / (4 * y_1 - 2 * y_0 - 2 * y_2)
        p_0 = p_0 + h
        p_1 = p_0 + h
        p_2 = p_0 + h * 2
        if abs(h * 2) < tol:
            break
    return p_1


def gradient(f, pf, x, y, tol=1e-5, max_iter=30):
    z = f(x, y)
    X, Y, Z = [], [], []

    print(z)
    for i in range(max_iter):
        gx, gy = pf(x, y)
        norm = np.sqrt(gx**2 + gy**2)
        Sx, Sy = -gx / norm, -gy / norm

        def grid_fn(gamma):
            return f(x + gamma * Sx, y + gamma + Sy)

        gamma = quadratic_approx(grid_fn, 0, 10)
        new_x, new_y = x + gamma * Sx, y + gamma * Sy
        new_z = f(new_x, new_y)

        if np.abs(z - new_z) < tol:
            break
        x, y, z = new_x, new_y, new_z
        X.append(x)
        Y.append(y)
        Z.append(z)
        print(f"gamma={gamma}, P_1=({x, y})")
    return np.array(X), np.array(Y), np.array(Z)
```

使用上面的代码，即可获得每一步梯度下降的点，使用代码如下

```
def f(x, y):
    return x * x + y**3 - 3 * x - 3 * y + 5


def pf(x, y):
    return [2 * x - 3, 3 * y * y - 3]


step_X, step_Y, step_Z = gradient(f, pf, -1, 2)
```

下面为其前三步结果

$$\gamma = 0.20713770780647436, P_1 = (-0.8994050364232408, 1.8189290655618333)$$
$$\gamma = 0.35720714865053355, P_2 = (-0.6959580367401849, 1.5253200306752905)$$
$$\gamma = 0.5222617383382477, P_3 = (-0.3089526652064909, 1.1746290474205796)$$

可视化结果如下，详情请见 question-8.py

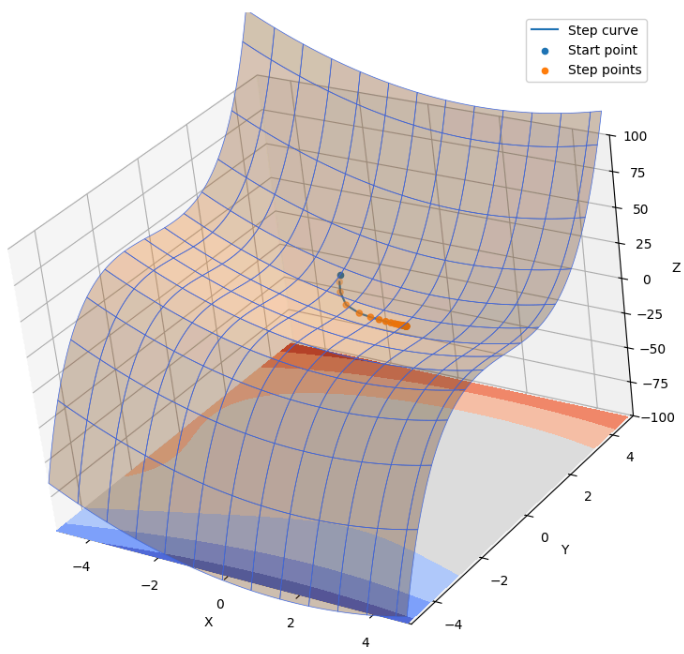

Figure 1: $f(x,y) = x^2 + y^3 - 3x - 3y + 5$ 从$(-1,2)$出发的下降过程

Question 7

代码如下

```
def modified_newtons(f, df, hf, p, tol=1e-5, max_iter=10):
    X, Y, Z = [p[0]], [p[1]], [f(p)]
    for i in range(max_iter):
        grad = df(p)
        hessian = hf(p)
        step = -np.matmul(grad, np.linalg.inv(hessian).T)
        print(hessian.shape)
        print(grad.shape)
        print(step.shape)

        def fn(g):
            return f(p + g * step)

        gamma = quadratic_approx(fn, 0, 100)
        new_p = p + gamma * step

        if np.abs(f(p) - f(new_p)) < tol:
            break
        p = new_p
        print(f"z={f(p)}, gamma = {gamma}")
        X.append(p[0])
        Y.append(p[1])
        Z.append(f(p))
    return np.array(X), np.array(Y), np.array(Z)
```