數值 HW7

第一題:

[3]:
```python
import numpy as np
A = np.array([[ 4, -1,  0, -1,  0,  0],
              [-1,  4, -1,  0, -1,  0],
              [ 0, -1,  4,  0,  1, -1],
              [-1,  0,  0,  4, -1, -1],
              [ 0, -1,  0, -1,  4, -1],
              [ 0,  0, -1,  0, -1,  4]], dtype=float)
b = np.array([0, -1, 9, 4, 8, 6], dtype=float)

def jacobi(A, b, x0, tol=1e-10, max_iter=1000):
    D = np.diag(np.diag(A))
    R = A - D
    x = x0.copy()
    for i in range(max_iter):
        x_new = np.linalg.inv(D) @ (b - R @ x)
        if np.linalg.norm(x_new - x, np.inf) < tol:
            return x_new, i + 1
        x = x_new
    return x, max_iter

# 初始猜測
x0 = np.zeros_like(b)
# 執行
x_jacobi, it_jacobi = jacobi(A, b, x0)
# 顯示結果
print(f"Jacobi result in {it_jacobi} iterations:\n{x_jacobi}")
```

```
Jacobi result in 47 iterations:
[1.17478856 1.64317358 2.44824809 3.05598067 3.94965767 3.09947644]
```

第二題:

[5]:
```python
def gauss_seidel(A, b, x0, tol=1e-10, max_iter=1000):
    x = x0.copy()
    n = len(b)
    for it in range(max_iter):
        x_new = x.copy()
        for i in range(n):
            s1 = np.dot(A[i, :i], x_new[:i])
            s2 = np.dot(A[i, i+1:], x[i+1:])
            x_new[i] = (b[i] - s1 - s2) / A[i, i]
        if np.linalg.norm(x_new - x, np.inf) < tol:
            return x_new, it + 1
        x = x_new
    return x, max_iter

# 初始猜測
x0 = np.zeros_like(b)

x_gs, it_gs = gauss_seidel(A, b, x0)
print(f"Gauss-Seidel result in {it_gs} iterations:\n{x_gs}")
```

```
Gauss-Seidel result in 19 iterations:
[1.17478856 1.64317358 2.44824809 3.05598067 3.94965767 3.09947644]
```

第三題:

•[7]:
```
import numpy as np
A = np.array([[ 4, -1,  0, -1,  0,  0],
              [-1,  4, -1,  0, -1,  0],
              [ 0, -1,  4,  0,  1, -1],
              [-1,  0,  0,  4, -1, -1],
              [ 0, -1,  0, -1,  4, -1],
              [ 0,  0, -1,  0, -1,  4]], dtype=float)
b = np.array([0, -1, 9, 4, 8, 6], dtype=float)
def sor(A, b, x0, w=1.2, tol=1e-10, max_iter=1000):
    x = x0.copy()
    n = len(b)
    for it in range(max_iter):
        x_new = x.copy()
        for i in range(n):
            s1 = np.dot(A[i, :i], x_new[:i])
            s2 = np.dot(A[i, i+1:], x[i+1:])
            x_new[i] = (1 - w) * x[i] + (w / A[i, i]) * (b[i] - s1 - s2)
        if np.linalg.norm(x_new - x, np.inf) < tol:
            return x_new, it + 1
        x = x_new
    return x, max_iter

# 初始猜测
x0 = np.zeros_like(b)

x_sor, it_sor = sor(A, b, x0, w=1.25)
print(f"SOR result (w=1.25) in {it_sor} iterations:\n{x_sor}")
```

```
SOR result (w=1.25) in 25 iterations:
[1.17478856 1.64317358 2.44824809 3.05598067 3.94965767 3.09947644]
```

第四題:

•[9]:
```
import numpy as np
A = np.array([[ 4, -1,  0, -1,  0,  0],
              [-1,  4, -1,  0, -1,  0],
              [ 0, -1,  4,  0,  1, -1],
              [-1,  0,  0,  4, -1, -1],
              [ 0, -1,  0, -1,  4, -1],
              [ 0,  0, -1,  0, -1,  4]], dtype=float)
b = np.array([0, -1, 9, 4, 8, 6], dtype=float)

def conjugate_gradient(A, b, x0, tol=1e-10, max_iter=1000):
    x = x0.copy()
    r = b - A @ x
    p = r.copy()
    rs_old = np.dot(r, r)
    for i in range(max_iter):
        Ap = A @ p
        alpha = rs_old / np.dot(p, Ap)
        x += alpha * p
        r -= alpha * Ap
        rs_new = np.dot(r, r)
        if np.sqrt(rs_new) < tol:
            return x, i + 1
        p = r + (rs_new / rs_old) * p
        rs_old = rs_new
    return x, max_iter
# 初始猜测
x0 = np.zeros_like(b)
x_cg, it_cg = conjugate_gradient(A, b, x0)
print(f"Conjugate Gradient result in {it_cg} iterations:\n{x_cg}")
```

```
Conjugate Gradient result in 1000 iterations:
[1.17656665 1.64269366 2.44433267 3.06002082 3.95260785 3.09922059]
```