仅供参考。

hw3-task3

- P120，3.8习题，26,27,28,29,30,34

```
dec = 6 # 设置每一步计算保留小数点后位数（精度，可以自己调整）
import numpy as np
np.set_printoptions(formatter={'float': ('{: 0.' + str(dec) + 'f}').format})
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['text.usetex'] = True
import sympy as sp
```

习题3.26

```
J = sp.Matrix([
    [0.9, 0],
    [0.3, 0.8]
])
eig_vals = list(J.eigenvals().keys())
eig_vals
```

```
[0.900000000000000, 0.800000000000000]
```

```
print("收敛，λmax=")
max(eig_vals)
```

```
收敛，λmax=
```

0.9

习题3.27

Jacobi

```python
J = -sp.Matrix([
    [0, 1/2, 1/2],
    [1/2, 0, 1/2],
    [1/2, 1/2, 0]
])
eig_vals = list(J.eigenvals().keys())
eig_vals
```

```
[-1.00000000000000, 0.500000000000000]
```

```python
print("发散，λmin=")
min(eig_vals)
```

```
发散，λmin=
```

-1.0

G-S

```python
G = J.copy()
G[1, 0] = 0
G[1, :] += J[1, 0] * J[0, :]
G[2, 0] = 0
G[2, 1] = 0
G[2, :] += J[2, 0] * G[0, :]
G[2, :] += J[2, 1] * G[1, :]
G
```

$$\begin{bmatrix} 0 & -0.5 & -0.5 \\ 0 & 0.25 & -0.25 \\ 0 & 0.125 & 0.375 \end{bmatrix}$$

```python
print("收敛，ρ(G) =")
max(list((G.T @ G).eigenvals().keys()))
```

```
收敛，ρ(G) =
```

$$0.633190229629063$$

习题3.28

Jacobi

```python
J = sp.Matrix([
    [0, 0, -1],
    [1, 0, 0],
    [sp.Rational(1, 3), sp.Rational(2, 3), 0]
])
lamb = sp.Symbol("\lambda")
J_lamb = J - lamb * sp.eye(3)
char_poly = J_lamb.det()
char_poly_eq = sp.Eq(char_poly, 0)
print("特征方程：")
char_poly_eq
```

```
特征方程：
```

$$-\lambda^3 - \frac{\lambda}{3} - \frac{2}{3} = 0$$

```
lam = sp.solve(char_poly_eq, lamb)
print("特征值如下：")
lam[0].evalf(dec)
```

```
特征值如下：
```

$$0.373708 + 0.867355i$$

```
lam[1].evalf(dec)
```

$$0.373708 - 0.867355i$$

```
lam[2].evalf(dec)
```

$$-0.747415$$

```
print("收敛，λmax =")
a = lam[0].evalf(dec)
sp.Matrix(a.as_real_imag()).norm()
```

```
收敛，λmax =
```

$$0.944438$$

G-S

```
J
```

$$\begin{bmatrix}0 & 0 & -1\\1 & 0 & 0\\\frac{1}{3} & \frac{2}{3} & 0\end{bmatrix}$$

```
G = J.copy()
G[1, 0] = 0
G[1, :] += J[1, 0] * J[0, :]
G[2, 0] = 0
G[2, 1] = 0
G[2, :] += J[2, 0] * G[0, :]
G[2, :] += J[2, 1] * G[1, :]
G
```

$$\begin{bmatrix}0 & 0 & -1\\0 & 0 & -1\\0 & 0 & -1\end{bmatrix}$$

```
print("λ:")
list(G.eigenvals().keys())
```

```
λ:
```

```
[-1, 0]
```

显然这里是发散的。

习题3.29

```python
J = sp.Matrix([
    [0, 0, 2/3],
    [0, 0, -1/2],
    [1, -1/2, 0]
])
print("λ of J:")
list(J.eigenvals().keys())
```

```
λ of J:
```

```
[0, -0.957427107756338, 0.957427107756338]
```

```python
G = J.copy()
G[1, 0] = 0
G[1, :] += J[1, 0] * J[0, :]
G[2, 0] = 0
G[2, 1] = 0
G[2, :] += J[2, 0] * G[0, :]
G[2, :] += J[2, 1] * G[1, :]
G
```

$$\begin{bmatrix} 0 & 0 & 0.666666666666667 \\ 0 & 0 & -0.5 \\ 0 & 0 & 0.916666666666667 \end{bmatrix}$$

```python
print("λ of G:")
list(G.eigenvals().keys())
```

```
λ of G:
```

```
[0, 0.916666666666667]
```

对比发现而这其实都是收敛的，但G-S似乎收敛的更快。

习题3.30

第一问

```
J = sp.Matrix([
    [0, 0],
    [-0.3, 0]
])
J.norm()
```

0.3

这里 G 和 J 敛散性都一样，都是收敛的。

第二问

```
A = sp.Matrix([
    [12, 0],
    [0.3, 1]
])
alpha = sp.Symbol(r"\alpha")
J = A * alpha + sp.eye(2)
J
```

$$\begin{bmatrix} 12\alpha + 1 & 0 \\ 0.3\alpha & \alpha + 1 \end{bmatrix}$$

```
lam_list = list(J.eigenvals().keys())
lam_list[0]
```

$12\alpha + 1$

```
lam_list[1]
```

$\alpha + 1$

这两个特征值应当都满足在范围 $(-1, 1)$ 内，因此自然有：

$$-\frac{1}{6} < \alpha < 0$$

习题3.31

```
A = sp.Matrix([
    [2, -1],
    [1, sp.Rational(3, 2)]
])
A
```

$$\begin{bmatrix} 2 & -1 \\ 1 & \frac{3}{2} \end{bmatrix}$$

```
J = A.copy()
J[0, :] /= -A[0, 0]
J[1, :] /= -A[1, 1]
J[0, 0] = 0
J[1, 1] = 0
J
```

$$\begin{bmatrix} 0 & \frac{1}{2} \\ -\frac{2}{3} & 0 \end{bmatrix}$$

```
G = J.copy()
G[1, 0] = 0
G[1, :] += J[1, 0] * J[0, :]
G
```

$$\begin{bmatrix} 0 & \frac{1}{2} \\ 0 & -\frac{1}{3} \end{bmatrix}$$

习题3.34

```python
J = sp.Matrix([
    [0, sp.Rational(1, 9), sp.Rational(1, 9)],
    [sp.Rational(1, 8), 0, 0],
    [sp.Rational(1, 9), 0, 0]
])
b = sp.Matrix([
    [sp.Rational(7, 9)],
    [sp.Rational(7, 8)],
    [sp.Rational(8, 9)],
])
J
```

$$\begin{bmatrix} 0 & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{8} & 0 & 0 \\ \frac{1}{9} & 0 & 0 \end{bmatrix}$$

```python
b.T
```

$$\begin{bmatrix} \frac{7}{9} & \frac{7}{8} & \frac{8}{9} \end{bmatrix}$$

```python
G = J.copy()
G[1, 0] = 0
G[1, :] += J[1, 0] * J[0, :]
G[2, 0] = 0
G[2, 1] = 0
G[2, :] += J[2, 0] * G[0, :]
G[2, :] += J[2, 1] * G[1, :]
G
```

$$\begin{bmatrix} 0 & \frac{1}{9} & \frac{1}{9} \\ 0 & \frac{1}{72} & \frac{1}{72} \\ 0 & \frac{1}{81} & \frac{1}{81} \end{bmatrix}$$

```python
bG = b.copy()
bG[1, 0] += J[1, 0] * bG[0, 0]
bG[2, 0] += J[2, 0] * bG[0, 0]
bG[2, 0] += J[2, 1] * bG[1, 0]
bG
```

$$\begin{bmatrix} \frac{7}{9} \\ \frac{35}{36} \\ \frac{79}{81} \end{bmatrix}$$

```python
x0 = sp.Matrix([
    [0], [0], [0]
])
x0
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```python
x = x0
for i in range(4):
    x = G @ x + bG
    pass
x.evalf(dec)
```

$$\begin{bmatrix} 0.999996 \\ 1.0 \\ 1.0 \end{bmatrix}$$

会发现经过四步迭代就已经收敛到解 $[1, 1, 1]$ 。