数值分析 实验5

2019011265 计93 丁韶峰

上机题1

实验内容

用幂法求特征值和特征向量。

实验过程

先 import numpy。

```python
import numpy as np
```

实现幂法。差值小于10^{-5}时停止迭代。

```python
def pow_method(A):
  n = A.shape[0]
  v = np.ones(n)
  old_lambda = 0
  while True:
    v = np.dot(A, v)
    new_lambda = v[np.argmax(np.abs(v))]
    v = v / new_lambda
    if (np.abs(new_lambda - old_lambda) < 1e-5):
      return new_lambda, v
    old_lambda = new_lambda
```

计算两个矩阵的特征值和对应特征向量如下。

```python
A = np.array([[5, -4, 1], [-4, 6, -4], [1, -4, 7]])
B = np.array([[25, -41, 10, -6], [-41, 68, -17, 10], [10, -17, 5, -3], [-6, 10, -3, 2]])
lambda_a, v_a = pow_method(A)
lambda_b, v_b = pow_method(B)
print("A lambda {} v {}".format(lambda_a, v_a))
print("B lambda {} v {}".format(lambda_b, v_b))
```

```
A lambda 12.254320584751564 v [-0.67401981  1.          -0.88955964]
B lambda 98.52169772379699 v [-0.60397234  1.          -0.25113513  0.14895345]
```

上机题3，4

实验内容

实现基本的QR算法，观察收敛情况。

实验过程

先定义 householder 变换。

```python
eps = 1e-6
def householder(x):
  sign = 1 if x[0] >= 0 else -1
  sigma = sign * np.linalg.norm(x)
  if np.abs(sigma - x[0]) < eps:
    return None
  y = np.copy(x)
  y[0] += sigma
  return y
```

根据伪代码，基于 Householder 变换实现基本的QR分解。

```python
def QR(A):
  n = A.shape[0]
  R = np.copy(A)
  Q = np.identity(n)
  for k in range(n - 1):
    R0 = R[k:, k:]
    v = householder(R0[:, 0])
    if v is None:
      continue
    w = v / np.linalg.norm(v)
    w = w.reshape((1, len(w)))
    H = np.identity(n)
    H[k:, k:] = np.identity(n - k) - (2 * np.matmul(w.transpose(), w))
    Q = np.matmul(Q, H)
    beta = np.dot(v.transpose(), v)
    for j in range(n - k):
      gamma = np.dot(v.transpose(), R0[:, j])
      R0[:, j] -= 2 * gamma * v / beta
  return Q, R
```

判断一个矩阵是不是伪上三角阵。需要考虑一阶和二阶分块的情况。

```python
def is_quasi_diag(A):
  n = A.shape[0]
  is_zero = A < eps
  i = 0
  while i < n:
    is_zero[i, i] = True
```

```
    if i < n - 1 and is_zero[i + 1, i] == False:
      is_zero[i + 1, i] = True
      i += 2
    else:
      i += 1
  for i in range(n):
    for j in range(i):
      if not is_zero[i, j]:
        return False
  return True
```

计算伪上三角阵的特征值，同样需要考虑一阶和二阶对角块的情况。

```
def get_eigenvalue(A):
  n = A.shape[0]
  eigenvalue = np.zeros(n, dtype=np.complex128)
  i = 0
  while i < n:
    if i < n - 1 and A[i + 1, i] > eps:
      eigenvalue[i : i + 2] = np.linalg.eig(A[i:i + 2, i:i + 2])[0]
      i += 2
    else:
      eigenvalue[i] = A[i, i]
      i += 1
  return eigenvalue
```

QR迭代。可能在成为伪三角阵前就收敛，也可能成为伪三角阵后才收敛。

```
def QR_iter(A):
  n = A.shape[0]
  cnt = 0
  while True:
    Q, R = QR(A)
    new_A = np.matmul(R, Q)
    cnt += 1
    if is_quasi_diag(new_A):
      break
    if np.max((np.abs(new_A - A))) < eps:
      print("QR coverges in {} steps, can't get eigenvalues".format(cnt))
      return new_A, None
    A = new_A
  print("QR coverges in {} steps".format(cnt))
  return A, get_eigenvalue(A)

```

移位 QR 迭代。根据伪代码实现即可。

```
def QR_shift_iter(A):
```

```
  n = A.shape[0]
  k = n
  cnt = 0
  while k > 1 and np.abs(A[k - 1, k - 2]) > eps:
    old_A = np.copy(A)
    s = A[k - 1, k - 1]
    Q, R = QR(A[:k, :k] - s * np.identity(k))
    A[:k, :k] = np.matmul(R, Q) + s * np.identity(k)
    cnt += 1
    if is_quasi_diag(A):
      break
    if np.max(np.abs(A - old_A) < eps):
      print("shifted QR coverges in {} steps, can't get eigenvalues".format(cnt))
  print("shifted QR coverges in {} steps".format(cnt))
  return A, get_eigenvalue(A)

```

基本的QR迭代，结果如下。无法计算出特征值。

```
A = np.matrix([[0.5, 0.5, 0.5, 0.5], [0.5, 0.5, -0.5 , -0.5], [0.5, -0.5, 0.5, -0.5], [0.5, -0.5, -0.5, 0.5]])
final_A, eig_A = QR_iter(A)
print(final_A)
print(eig_A)
```

```
[[-0.5 -0.5 -0.5 -0.5]
 [-0.5 -0.5  0.5  0.5]
 [-0.5  0.5 -0.5  0.5]
 [-0.5  0.5  0.5 -0.5]]
[[-1.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00 -1.00000000e+00  1.11022302e-16  1.11022302e-16]
 [ 0.00000000e+00  0.00000000e+00 -1.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -1.00000000e+00]]
[[ 0.5  0.5  0.5  0.5]
 [ 0.5  0.5 -0.5 -0.5]
 [ 0.5 -0.5  0.5 -0.5]
 [ 0.5 -0.5 -0.5  0.5]]
QR coverges in 1 steps, can't get eigenvalues
[[ 0.5  0.5  0.5  0.5]
 [ 0.5  0.5 -0.5 -0.5]
 [ 0.5 -0.5  0.5 -0.5]
 [ 0.5 -0.5 -0.5  0.5]]
None
```

移位QR迭代，结果如下。可以计算出特征值。

```python
shifted_final_A, shifted_eig_A = QR_shift_iter(A)
print(shifted_final_A)
print(shifted_eig_A)
```

```
QR coverges in 3 steps
[[-1.00000000e+00 -1.77392636e-06  1.02438497e-06 -7.24404755e-07]
 [-1.77392636e-06  1.00000000e+00  9.08451112e-13 -6.42446563e-13]
 [ 1.02438497e-06  9.08519774e-13  1.00000000e+00  3.71258381e-13]
 [-7.24404755e-07 -6.42476638e-13  3.71108492e-13  1.00000000e+00]]
[-1.+0.j  1.+0.j  1.+0.j  1.+0.j]
```

实验结论

幂法可以较快地计算出绝对值最大的特征值和对应的特征向量，且实现较简单。

对于本章上机题中的矩阵，基本的QR迭代无法计算出特征值，因为矩阵本就是正交的，经过迭代结果不会有变化。移位QR迭代破坏了矩阵的正交性，可以迭代计算出特征值。