# 运用幂迭代法和 Rayleigh 商迭代法近似计算矩阵特征值的探索

**摘要：**矩阵的特征值是矩阵运用的一个重点，在学科研究方面具有重要的地位，进行矩阵特征值的讨论可以直接用来解决实际问题。本文简要介绍了近似计算矩阵特征值的幂迭代法和 Rayleigh 商迭代法，并对二者进行了实验和对比分析。在遇到具体问题时选择合适的计算方法以达到最优化解决的目的。

**关键词：**矩阵特征值；幂迭代法；Rayleigh 商迭代法

## 一　引言

用传统方法计算特征值在 $n$ 很小时是可以的，但当 $n$ 稍大时，计算工作量将以惊人的速度增大。由于计算带有误差，特征方程 $(\lambda E - A) = 0$ 的求解就很困难了，这时就需要一些其他方法求解矩阵特征值。

## 二　幂迭代法

### 2.1　理论基础

幂迭代法是一种计算矩阵主特征值 (矩阵按模最大的特征值) 以及对应特征向量的迭代法，设矩阵 $A = (a_{ij})_{m\times n}$ 的特征值为 $\lambda_1, \lambda_2, \cdots, \lambda_n$，其相应的特征向量为 $x_1, x_2, \cdots, x_n$，已知 $A$ 的主特征值都是实根，且满足条件

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

幂迭代法的基本思想是任意取一个非零的初始值向量 $v_0$，由矩阵 $A$ 构造一向量序列，称为迭代向量。由假设，$v_0$ 可表示为 $v_0 = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$(设 $\alpha_1 \neq 0$) 时

$$\begin{aligned} v_k &= Av_{k-1} = A^k v_0 \\ &= \alpha_1 \lambda_1^k x_1 + \alpha_2 \lambda_2^k x_2 + \cdots + \alpha_n \lambda_n^k x_n \\ &= \lambda_1^k \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k x_i \right] \\ &= \lambda_1^k (\alpha_1 x_1 + \varepsilon_k) \end{aligned}$$

其中 $\varepsilon_k = \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k x_i$，由假设 $\left| \frac{\lambda_i}{\lambda_1} \right| < 1 (i = 2, 3, \cdots, n)$，故 $\lim_{k\to\infty} \varepsilon_k = 0$ 从而 $\lim_{k\to\infty} \frac{v_k}{\lambda_1^k} = \alpha_1 x_1$。这说明序列 $\frac{v_k}{\lambda_1^k}$ 越来越接近 $A$ 的相对应 $\lambda_1$ 的特征向量，或者说当 $k$ 充分大时，$v_k \approx \alpha_1 \lambda_1^k x_1$，即迭代向量 $v_k$ 为 $\lambda_1$ 的特征向量的相似向量 (除了一个因子外)。

下面考虑主特征值 $\lambda_1$ 的计算，再用 $(v_k)_i$ 表示 $v_k$ 的第 $i$ 个分量

$$\frac{(v_{k+1})_i}{(v_k)_i} = \lambda_1 \left\{ \frac{\alpha_1 (x_1)_i + (\varepsilon_{k+1})_i}{\alpha_1 (x_1)_i + (\varepsilon_k)_i} \right\}$$

故

$$\lim_{k\to\infty} \frac{(v_{k+1})_i}{(v_k)_i} = \lambda_1$$

也就是说两相邻迭代向量的比值收敛到主特征值。这种由一非零向量 $v_0$ 以及矩阵 $A$ 的幂乘 $A^k$ 构造向量序列 $v_k$ 以计算 $A$ 的主特征值 $\lambda_1$ 以及相应特征向量的方法称为幂迭代法。

在上述同等条件下，幂迭代法可以这样进行：取一初始向量 $v_0(\alpha_1 \neq 0)$，构造的向量序列

$$\begin{cases} v_1 = Au_0 = Av_0 \\ v_2 = Au_1 = \dfrac{A^2 v_0}{\max\{Av_0\}} \\ v_k = \dfrac{A^k v_0}{\max\{A^{k-1} v_0\}} \end{cases}, \begin{cases} u_1 = \dfrac{v_1}{\max\{v_1\}} = \dfrac{Av_0}{\max\{Av_0\}} \\ u_2 = \dfrac{v_2}{\max\{v_2\}} = \dfrac{A^2 v_0}{\max\{A^2 v_0\}} \\ u_k = \dfrac{A^k v_0}{\max\{A^k v_0\}} \end{cases}$$

其中 $u_1 = \dfrac{v_1}{\max\{v_1\}}$ 中 $\max\{v_1\}$ 表示向量 $v_1$ 的绝对值最大的分量。

由上面的式子可以得到

$$A^k v_0 = \sum_{i=1}^{n} \alpha_1 \lambda_i^k x_i = \lambda_1^k \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right]$$

$$u_k = \frac{A^k v_0}{\max\{A^k v_0\}} = \frac{\lambda_1^k \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right]}{\max\left\{ \lambda_1^k \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right] \right\}} = \frac{\left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right]}{\max\left\{ \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right] \right\}} \to \frac{x_1}{\max\{x_1\}}$$

$$v_k = \frac{\lambda_1^k \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right]}{\max\left\{ \lambda_1^{k-1} \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^{k-1} x_i \right) \right] \right\}}$$

$$\max\{v_k\} = \frac{\lambda_1 \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right]}{\max\left\{ \left[ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1}^k x_i \right) \right] \right\}} \to \lambda_1$$

所以求解矩阵的主特征值就只需要求解 $\max\{v_k\}$ 就行了。

扩展幂迭代法求实对称矩阵的所有特征值和特征向量。

扩展算法：

(1) 输入矩阵 $A = a_{ij} \in \mathbb{R}^{m\times n}$，给定误差界 $\varepsilon > 0$，置 $k = 0$ 个最大迭代次数 MAX；

(2) 给出一组线性无关且已正交规范化的初始向量 $v_1(0), v_2(0), \cdots, v_n(0)$；

(3) 分别将 $v_1(0), v_2(0), \cdots, v_n(0)$ 作为列向量，构成矩阵 $B$；

(4) 重复以下操作，逐步确定各近似特征向量，设 $i = 0$，内部迭代次数 iter；

i. 重复 iter 次操作：$k=k+1, u_j(k)=Av_j(k-1), v_j(k)=\frac{u_j(k)}{\max\{u_j(k)\}}(j=1,2,\cdots,n)$，即 $B=AB$；

ii. 计算 $\lambda_j(k)=\dfrac{(v_j(k),v_{j-1}(k))}{(v_{j-1}(k),v_{j-1}(k))}(j=1,2,\cdots,n)$；

iii. 对所有 $j=i,i+1,\cdots,n$ 将 $\lambda_j(k)$ 按从大到小排列，同时调整对应 $v_j(k)$ 在 $B$ 中的位置；

iv. 对 $B$ 做正交规范化；

v. 对所有 $j=i,i+1,\cdots,n$ 验证 $|\lambda_j(k)-\lambda_j(k-1)|<\varepsilon$ 是否成立，

若成立，则同时交换 $\lambda_j(k), v_j(k)$ 与 $\lambda_i(k), v_i(k)$ 的位置，$i=i+1$；

vi. 若 $i=n$，则转步骤 (5)；若 $k<$MAX，则继续重复 (4)；否则输出溢出，算法结束。

(5) 输出各 $\lambda_j(k)$ 为矩阵 $A$ 的特征值，$v_j(k)$ 为对应特征向量，算法结束。

## 2.2 数值实验

### 2.2.1 实例

求矩阵 $A=\begin{bmatrix}2&3&2\\10&3&4\\3&6&1\end{bmatrix}$ 的主特征值。

### 2.2.2 计算

实际计算时，为了避免计算过程中出现绝对值过大或过小的数参加运算，通常在每步迭代时，将向量“归一化”，即用模最大的分量。

修改后的迭代公式：

$$\begin{cases}y_k=Ax_{k-1}\\m_k=\max\{y_k\}\\x_k=\dfrac{y_k}{m_k}\end{cases}$$

依据以上迭代公式进行 Python 实现，本例选择 $x_0=(0,0,1)^{\mathrm{T}}$ 作初始向量，并设定迭代精度为 $10^{-10}$，最大迭代次数为 20 次，具体代码如下一页：

```python
import numpy as np
def Solve(mat, max_itrs, min_delta):
    '''
    mat 表示矩阵
    max_itrs 表示最大迭代次数
    min_delta 表示停止迭代阈值
    '''
    itrs_num = 0
    delta = float('inf')
    N = np.shape(mat)[0] #所有分量都为1的列向量
    x = np.ones(shape=(N, 1)) #x = np.array([[0],[0],[1]])
    while itrs_num < max_itrs and delta > min_delta:
        itrs_num += 1
        y = np.dot(mat, x)
        #print(y)
        m = y.max()
        #print("m={0}".format(m))
        x = y / m
        print("***********第{}次迭代*************".format(itrs_num))
        print("m={0}".format(m))
        print("====================================")
IOS = np.array([[2, 3, 2],[10, 3, 4],[3, 6, 1]], dtype=float)
Solve(IOS, 20, 1e-10)
```

### 2.2.3 输出

| 第 1 次迭代 | 第 2 次迭代 | 第 3 次迭代 | 第 4 次迭代 |
|---|---|---|---|
| $m = 17.0$ | $m = 9.470588235294118$ | $m = 11.58385093167702$ | $m = 10.831635388739945$ |
| 第 5 次迭代 | 第 6 次迭代 | 第 7 次迭代 | 第 8 次迭代 |
| $m = 11.049799514875502$ | $m = 10.985906091381928$ | $m = 11.003953118800313$ | $m = 10.998903290037243$ |
| 第 9 次迭代 | 第 10 次迭代 | 第 11 次迭代 | 第 12 次迭代 |
| $m = 11.000302582696586$ | $m = 10.999916852432536$ | $m = 11.000022790833176$ | $m = 10.999993763594336$ |
| 第 13 次迭代 | 第 14 次迭代 | 第 15 次迭代 | 第 16 次迭代 |
| $m = 11.000001704609195$ | $m = 10.999999534421143$ | $m = 11.000000127100696$ | $m = 10.999999965313513$ |
| 第 17 次迭代 | 第 18 次迭代 | 第 19 次迭代 | 第 20 次迭代 |
| $m = 11.00000000946407$ | $m = 10.999999997418142$ | $m = 11.000000000704278$ | $m = 10.999999999807901$ |

### 2.3 结果分析

矩阵 $A$ 的主特征值 $\lambda = 11$，幂迭代法适用条件是只需要求出矩阵的按模最大的特征值和相应的特征向量。其优点是算法简单，代码逻辑顺畅，易编写，容易在计算机上实现；缺点是收敛速度较慢。

## 三 Rayleigh 商迭代法

### 3.1 理论基础

对于特征值问题 $Ax = \lambda x$，记 $A$ 的 $n$ 个特征值由小到大为 $\lambda_1, \lambda_2, \cdots, \lambda_n$，它们对应的单位特征向量为 $z_1, z_2, \cdots, z_n$。

Rayleigh 商迭代法求 $A$ 的特征对 $(\lambda, z)$ 的算法如下：

取一个单位初始向量 $x_0$，对于 $kk = 0, 1, 2, \cdots$ 重复如下步骤，称为 RQI 算法 。

(1) 计算 $\rho_k = (Ax_k, x_k)$。

(2) 若 $A - \rho_k I$ 奇异，那么解 $(A - \rho_k I)x_{k+1} = 0$ 得单位向量，停止计算，此时 $(\rho_k, x_{k+1})$ 即为特征对。否则解方程：$(A - \rho_k I)y_{k+1} = x_k$。

(3) $x_{k+1} = \dfrac{y_{k+1}}{||y_{k+1}||}$.

(4) 如果 $||y_{k+1}||$ 足够大，那么计算停止，此时 $(\rho_{k+1}, x_{k+1})$ 即为 $A$ 的近似特征对。这里用到的内积和范数，全是指通常意义下的。

定理：设 $\{x_k\}$ 是由初始向量 $x_0$，通过上述 RQI 算法 ，产生的单位向量序列，那么当 $k \to \infty$ 时，$\{\rho_k\}$ 总有极限，并且下列二者必有其一：

(1) $(\rho_k, x_k) \to (\lambda, z)$，这里 $Az = \lambda z$，且收敛速度是三次的。

(2) $\rho_k \to \alpha$，$\alpha$ 是两个特征值 $\lambda_{i1}, \lambda_{i2}$ 平均值即 $\alpha = \dfrac{\lambda_{i1} + \lambda_{i2}}{2}$，此时 $x_{2k} \to x_+, x_{2k+1} \to x_-$。

这里 $x_+ = \dfrac{z_{i1} + z_{i2}}{\sqrt{2}}, x_- = \dfrac{z_{i1} - z_{i2}}{\sqrt{2}}$，这时的收敛速度是线性的。并且情况 (2) 在 $x_k$ 的扰动下是不稳定的。

这个定理是 Rayleigh 商迭代法的理论基础，按照 RQI 算法 计算，当 $\rho_k \to \lambda$ 时，还有两种可能，即

(a) $x_n \to z$ 是对应特征值 $\lambda$ 的单位特征向量。

(b) $x_{2k} \to z, x_{2k+1} \to -z$。

在后一种情况下，就不全有 $x_k \to z$ 的结论，此时 $\{x_k\}$ 根本不收敛。

为了避免情况 (b) 的发生，我们提出了如下的 Rayleigh 商迭代法，称为 RQI 算法 ：

(1) 取一个单位初始向量 $x_0, 0 \to k$。

(2) 计算 $\rho_k = (Ax_k, x_k)$。

(3) 计算 $r_k = Ax_k - \rho_k x_k$。若 $||r_k|| < \varepsilon$，则 $(\rho_k, x_k)$ 为近似特征对，计算结束，否则转 (2)。

(4) 求解方程得 $A-\rho_k I x_{k+1}=\tau_k x_k$，得 $x_{k+1}$，其中 $\tau_k$ 是一个实数，使得 $||x_{k+1}||=1$ 并且 $(x_{k+1},x_k)\geq 0, k+1\to k$。

两个算法的主要差别是在 RQI 算法　中相当于上述的 $\tau_k$ 始终取正的，即 $\tau_k=||y_{k+1}||^{-1}$，而在算法　中 $\tau_k$ 取决于 $(x_{k+1},x_k)\geq 0$ 可以是正也可以是负的，这样不会产生情况 (b)。

## 3.2 数值实验

### 3.2.1 实例

以幂迭代法所用的矩阵 $A$ 为例，求主特征值。

### 3.2.2 计算

规范化向量 $u^{(k)}$ 的瑞利商可以给出主特征值 $\lambda_1$ 更好的近似：

$$\frac{\left(Au^{(k)},u^{(k)}\right)}{\left(u^{(k)},u^{(k)}\right)}=\lambda_1+O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^{2k}\right)$$

依据理论基础和改进算法进行 Python 实现，选择 $x_0=(1,1,1)^{\mathrm{T}}$ 作初始向量，并设定迭代精度为 $10^{-10}$，最大迭代次数为 20 次，具体代码如下：

```
import numpy as np
def eig_power(A,v0,eps):
    uk = v0
    flag = 1
    val_old = 0
    n = 0
    while flag:
        n = n+1
        vk = A*uk
        t = ((A*uk).T*uk)/(uk.T*uk)
        val = t[0,0]
        uk = vk/val
        if (np.abs(val-val_old)<eps):
            flag = 0
        val_old = val
        print(np.asarray(uk).flatten(),val)
    print('max eigenvalue:',val)
return val, uk
if __name__ == '__main__':
    A = np.matrix([[2, 3, 2],[10, 3, 4],[3, 6, 1]], dtype='float')
    v0 = np.matrix([[1],[1],[1]], dtype='float')
    val,uk = eig_power(A,v0,1e-10)
```

#### 3.2.3 输出

| 第 1 次迭代 | 第 2 次迭代 | 第 3 次迭代 | 第 4 次迭代 |
|---|---|---|---|
| $m = 11.333333333333334$ | $m = 10.643835616438356$ | $m = 10.985777515491293$ | $m = 10.999167655116823$ |
| 第 5 次迭代 | 第 6 次迭代 | 第 7 次迭代 | 第 8 次迭代 |
| $m = 10.998043732856468$ | $m = 11.000632908726239$ | $m = 10.9997831414603$ | $m = 11.00006502382655$ |
| 第 9 次迭代 | 第 10 次迭代 | 第 11 次迭代 | 第 12 次迭代 |
| $m = 10.999981024770307$ | $m = 11.000005387391742$ | $m = 10.999998491080834$ | $m = 11.000000418651231$ |
| 第 13 次迭代 | 第 14 次迭代 | 第 15 次迭代 | 第 16 次迭代 |
| $m = 10.999999884520566$ | $m = 11.000000031730645$ | $m = 10.999999991303198$ | $m = 11.000000002379668$ |
| 第 17 次迭代 | 第 18 次迭代 | 第 19 次迭代 | 第 20 次迭代 |
| $m = 10.99999999934958$ | $m = 11.000000000177648$ | $m = 10.999999999951504$ | $m = 11.000000000013234$ |

### 3.3 结果分析

经实验发现，此实例中的初始值 $(1,1,1)^{\mathrm{T}}$ 与幂迭代法中的 $(0,0,1)^{\mathrm{T}}$ 对结果的影响很小，可以认为两实验在同等的初始条件下进行。对比幂迭代法的实验结果可以发现，Rayleigh 商迭代法可以很快速地收敛到精确值附近，甚至在第一次迭代时就已经把误差控制在比较小的范围内，因此，其优点是收敛速度快。但是，Rayleigh 商迭代法的代码不如幂迭代法那样清楚易懂，它有很强的逻辑性和专业性，上手较难，不易实现。再比较二者精度，幂迭代法的精度相较 Rayleigh 商迭代法略低，但差别很小，对一般数值实验的影响可以忽略。