

计算物理

王延颢

2016 年 2 月 17 日

1. 本征问题的求解

此章比较全面地介绍数值求解本征问题的基本概念和方法。

1.1. 基本概念

一个 $N \times N$ 的矩阵 $\mathbf{A}$ 总会有本征矢量 (eigenvector) $\mathbf{x}$ 和对应的本征值 (eigenvalue) λ 满足

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{x} \quad (1.1)$$

显然一个本征矢量的倍数也满足上式，但是并不认为是不同的本征矢量，而 $\mathbf{0}$ 也不认为是一个本征矢量。显然上式等同于

$$\det|\mathbf{A} - \lambda \mathbf{I}| = 0 \quad (1.2)$$

其中 $\mathbf{I}$ 是单位矩阵。求解式(1.2)就可以得到 $\mathbf{A}$ 的 N 个本征值，其中相同的本征值被认为是简并 (degenerate) 的。一个本征值会有一个对应的本征矢量。直接求解该方程组的根的做法效率很低，需要专门用于求解本征问题的数值方法。

对称矩阵 (symmetric matrix) 满足

$$\mathbf{A} = \mathbf{A}^T \quad a_{ij} = a_{ji} \quad (1.3)$$

厄密矩阵 (Hermitian matrix) 满足

$$\mathbf{A} = \mathbf{A}^\dagger \quad a_{ij} = a_{ji}^* \quad (1.4)$$

正交矩阵 (orthogonal matrix) 满足

$$\mathbf{A}^T \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{A}^T = \mathbf{I} \quad (1.5)$$

单位矩阵 (unitary matrix) 满足

$$\mathbf{A} \cdot \mathbf{A}^\dagger = \mathbf{A}^\dagger \cdot \mathbf{A} = \mathbf{I} \quad (1.6)$$

正态矩阵 (normal matrix) 满足

$$\mathbf{A} \cdot \mathbf{A}^\dagger = \mathbf{A}^\dagger \cdot \mathbf{A} \quad (1.7)$$

对于实数矩阵，厄密等同于对称，单位等同于正交，二者都是正态矩阵。

复数矩阵的本征值一般都是复数，而厄密矩阵的本征值都是实数，而不对称的实数矩阵会有成对出现的复数本征值。非简并的正态矩阵的本征矢量是完备正交的，而简并的正态矩阵中的简并本征值所对应的本征矢量可以用其它本征矢量的线性叠加代替。

式(1.1)定义的本征矢量称为**右本征矢量**，相应地可以定义**左本征矢量**满足

$$\mathbf{x} \cdot \mathbf{A} = \lambda \mathbf{x} \quad (1.8)$$

显然每个左本征矢量都是 $\mathbf{A}$ 的转置矩阵的某个右本征矢量的转置矩阵，并且 $\mathbf{A}$ 的左本征值和右本征值相等。如果 $\mathbf{A}$ 是对称矩阵，则左右本征矢量互为转置。如果 $\mathbf{A}$ 是厄密矩阵，则左右本征矢量互为厄密共轭。

对于更一般的不对称情况，令矩阵 $\mathbf{X}_R$ 的每一列是右本征矢量， $\mathbf{X}_L$ 的每一行是左本征矢量，因为

$$\begin{aligned} \mathbf{A} \cdot \mathbf{X}_R &= \mathbf{X}_R \cdot \text{diag}(\lambda_1 \cdots \lambda_N) \\ \mathbf{X}_L \cdot \mathbf{A} &= \text{diag}(\lambda_1 \cdots \lambda_N) \cdot \mathbf{X}_L \end{aligned} \quad (1.9)$$

所以有

$$(\mathbf{X}_L \cdot \mathbf{X}_R) \cdot \text{diag}(\lambda_1 \cdots \lambda_N) = \text{diag}(\lambda_1 \cdots \lambda_N) \cdot (\mathbf{X}_L \cdot \mathbf{X}_R) \quad (1.10)$$

这说明 $\mathbf{X}_L \cdot \mathbf{X}_R$ 必须是对角矩阵。如果本征值不简并，则一个左本征矢量会和除了它对应的右本征矢量之外的所有右本征矢量正交，同时利用归一化条件使得它们正交归一。如果有简并的本征值，可以用线性组合的变换使得它们正交归一。正交归一化导致的结果是**以左本征矢量为行的矩阵是以右本征矢量为列的矩阵的逆矩阵**，即 $\mathbf{X}_L = \mathbf{X}_R^{-1}$ 。

对式(1.9)的第一个式子左乘上 $\mathbf{X}_L = \mathbf{X}_R^{-1}$ ，得到

$$\mathbf{X}_R^{-1} \cdot \mathbf{A} \cdot \mathbf{X}_R = \text{diag}(\lambda_1 \cdots \lambda_N) \quad (1.11)$$

这是**相似变换**（similarity transform）

$$\mathbf{A} \rightarrow \mathbf{Z}^{-1} \cdot \mathbf{A} \cdot \mathbf{Z} \quad (1.12)$$

的一个特例。因为

$$\begin{aligned} \det|\mathbf{Z}^{-1} \cdot \mathbf{A} \cdot \mathbf{Z} - \lambda \mathbf{I}| &= \det|\mathbf{Z}^{-1} \cdot (\mathbf{A} - \lambda \mathbf{I}) \cdot \mathbf{Z}| = \det|\mathbf{Z}| \det|\mathbf{A} - \lambda \mathbf{I}| \det|\mathbf{Z}^{-1}| \\ &= \det|\mathbf{A} - \lambda \mathbf{I}| \end{aligned} \quad (1.13)$$

所以相似变换不改变一个矩阵的本征值。式(1.11)说明任意具有完备本征矢量的矩阵都可以用相似变换进行对角化。

实数对称矩阵的本征矢量都是正交的实数矢量，所以变换矩阵也是正交的，从而相似变换退化成**正交变换**（orthogonal transformation）

$$\mathbf{A} \rightarrow \mathbf{Z}^T \cdot \mathbf{A} \cdot \mathbf{Z} \quad (1.14)$$

对于不对称的实数矩阵，可以用相似变换把它变成沿对角线有很多 2×2 的小块的矩阵，其它矩阵元都是 0，每个小块对应于一对共轭的复数本征值。

所有的本征方程求解的算法都是基于把给定矩阵 $\mathbf{A}$ 通过一系列相似变换进行对角化的想法：

$$\mathbf{A} \rightarrow \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \rightarrow \mathbf{P}_2^{-1} \cdot \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \cdot \mathbf{P}_2 \rightarrow \mathbf{P}_3^{-1} \cdot \mathbf{P}_2^{-1} \cdot \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \rightarrow \cdots \quad (1.15)$$

如果一系列变换最终得到了对角矩阵，则总的变换矩阵的每一列就是一个右本征矢量：

$$\mathbf{X}_R = \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \cdots \quad (1.16)$$

如果不需要知道本征矢量，只要求解本征值，则只要将原始矩阵变换成为**三角矩阵**（即对角元的上面或者下面的元素全部为 0）就可以了，此时对角元素就是该矩阵的本征值。还有些算法可以只求解到最大的几个本征值对应的本征矢量就停止，从而节约大量的计算时间。

1.2. 对称矩阵的 Jacobi 转换

对于对称矩阵，Jacobi 方法实现式(1.15)的步骤时，每一步转换把一个非对角元置为 0，虽然之前置为 0 的非对角元又会变成非 0，但是值越来越小，直到所有的非对角元小于某一设定的精度阈值。本征值就是对角元素，而本征矢量由式(1.16)给出。Jacobi 方法计算效率低，但是非常简单，因此适合于尺寸不太大的本征问题求解。

基本的 Jacobi 算法的转换矩阵是一个转动操作，形式为

$$\mathbf{P}_{pq} = \begin{pmatrix} 1 & & & & \\ & \cdots & & & \\ & & c & \cdots & s \\ & & \vdots & 1 & \vdots \\ & & -s & \cdots & c \\ & & & & \cdots & \\ & & & & & 1 \end{pmatrix} \quad (1.17)$$

其中 $c \equiv \cos(\phi)$ ， $s \equiv \sin(\phi)$ ， ϕ 是转角，从而转换后的矩阵

$$\mathbf{A}' = \mathbf{P}_{pq}^T \cdot \mathbf{A} \cdot \mathbf{P}_{pq} = \begin{pmatrix} \cdots & a'_{1p} & \cdots & a'_{1q} & \cdots \\ \vdots & \vdots & & \vdots & \\ a'_{p1} & \cdots & a'_{pp} & \cdots & a'_{pq} & \cdots & a'_{pn} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a'_{q1} & \cdots & a'_{qp} & \cdots & a'_{qq} & \cdots & a'_{qn} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \cdots & a'_{np} & \cdots & a'_{nq} & \cdots \end{pmatrix} \quad (1.18)$$

只有 p, q 行和 p, q 列的元素被改变为

$$\begin{aligned} a'_{rp} &= ca_{rp} - sa_{rq} \\ a'_{rq} &= ca_{rq} + sa_{rp} \\ a'_{pp} &= c^2 a_{pp} + s^2 a_{qq} - 2sca_{pq} \\ a'_{qq} &= s^2 a_{pp} + c^2 a_{qq} + 2sca_{pq} \\ a'_{pq} &= (c^2 - s^2)a_{pq} + sc(a_{pp} - a_{qq}) \end{aligned} \quad (1.19)$$

其中 $r=1, \dots, N$ 且 $r \neq p, q$ 。这一步操作的目的是实现 $a'_{pq} = 0$ ，由此可得

$$\theta \equiv \cot(2\phi) \equiv \frac{c^2 - s^2}{2sc} = \frac{a_{qq} - a_{pp}}{2a_{pq}} \quad (1.20)$$

令 $t \equiv s/c$ ，得到

$$t^2 + 2t\theta - 1 = 0 \quad (1.21)$$

上式的数值较小的根对应于较小的转动角 ϕ ，从而可以较好地保证转动操作的稳定性：

$$t = \frac{\text{sgn}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}} \quad (1.22)$$

当 θ^2 大到有溢出计算机存储位数的危险时，直接取极限值

$$t = \frac{1}{2\theta} \quad (1.23)$$

求出 t 以后，根据定义可以得到

$$c = \frac{1}{\sqrt{t^2 + 1}} \quad (1.24)$$

$$s = tc$$

在 $a'_{pq} = 0$ 的约束下，式(1.19)变为

$$\begin{aligned} a'_{rp} &= a_{rp} - s(a_{rq} + \tau a_{rp}) \\ a'_{rq} &= a_{rq} + s(a_{rp} - \tau a_{rq}) \\ a'_{pp} &= a_{pp} - ta_{pq} \\ a'_{qq} &= a_{qq} + ta_{pq} \\ a'_{pq} &= 0 \end{aligned} \quad (1.25)$$

其中 $\tau \equiv \tan(\phi/2) = \frac{s}{1+c}$ 。由式(1.25)就可以求得转换后的矩阵 $\mathbf{A}'$ 。

定义所有非对角元的平方和 $S = \sum_{r \neq s} |a_{rs}|^2$ ，一次转换后的平方和为

$$S' = S - 2|a_{pq}|^2 \quad (1.26)$$

因此 Jacobi 转换操作保证非对角元的绝对值是单调递减的，从而只要操作的步数足够多，

总可以使得所有非对角元趋近于 0。多次迭代后得到小于设定精度阈值意义上的对角矩阵

$$\mathbf{D} = \mathbf{V}_f^T \cdot \mathbf{A} \cdot \mathbf{V}_f \quad (1.27)$$

其中

$$\mathbf{V}_f = \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \cdots \quad (1.28)$$

$\mathbf{D}$ 的每一个对角元素就是一个本征值；因为

$$\mathbf{A} \cdot \mathbf{V}_f = \mathbf{V}_f \cdot \mathbf{D} \quad (1.29)$$

所以 $\mathbf{V}_f$ 的每一列就是一个本征矢量。在程序实现时， $\mathbf{V}$ 的初始值可以从单位矩阵开始，在

与 $\mathbf{A}$ 根据式(1.25)作旋转操作的同时进行变换 $\mathbf{V}' = \mathbf{V} \cdot \mathbf{P}_i$ ，对应的矩阵元操作为

$$\begin{aligned} v'_{rp} &= cv_{rp} - sv_{rq} \\ v'_{rq} &= sv_{rp} + cv_{rq} \end{aligned} \quad (1.30)$$

其它元素保持不变。

对于操作元素的顺序，原始的 Jacobi 方法每次对数值最大的元素进行清零操作，但是寻找最大值的计算复杂度是 $O(N^2)$ ，所以现在的方法都按照行的顺序，即先操作

$\mathbf{P}_{12}, \mathbf{P}_{13}, \cdots, \mathbf{P}_{1N}$ ，再操作 $\mathbf{P}_{23}, \mathbf{P}_{24}, \cdots, \mathbf{P}_{2N}$ ，等等。

1.3. 对称矩阵转换为三对角形式

Givens 方法和 Householder 方法可以在有限的步数内把对称矩阵转换为三对角 (tridiagonal) 矩阵，即只有对角线及其上下的三条斜线上的矩阵元不为 0。

Givens 方法在对式(1.19)操作时，并不把 a'_{pq} 设为 0，而是把 $a'_{p-1,q}$ 设为 0。因为 a'_{rp} 和 a'_{rq} 是 a_{rp} 和 a_{rq} 的线性叠加 ($r \neq p, q$)，所以一旦 a_{rp} 和 a_{rq} 被置为 0，之后的迭代中就一直保持为 0。Givens 算法简单，但是比 Householder 算法慢一倍，所以不太常用。

Householder 算法用 $N-2$ 个正交变换把一个 $N \times N$ 的对称矩阵变成三对角形式。每步变换把一整列和对应的一整行变为 0。Householder 的基本转换矩阵形式为

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{u} \cdot \mathbf{u}^T}{H} \quad (1.31)$$

其中 $\mathbf{u}$ 是一个任意的实矢量， $H = |\mathbf{u}|^2 / 2$ 。可以证明 $\mathbf{P}$ 是正交归一矩阵， $\mathbf{P}^2 = \mathbf{I}$ ，

$$\mathbf{P} = \mathbf{P}^{-1} = \mathbf{P}^T。$$

Householder 算法的基本原理如下。第一步操作时，取 $\mathbf{u} = \mathbf{x} \mp |\mathbf{x}| \mathbf{e}_1$ ，其中 $\mathbf{x}$ 是 $\mathbf{A}$ 的第一列， $\mathbf{e}_1 = (1, 0, \dots, 0)^T$ ，则

$$\begin{aligned} \mathbf{P} \cdot \mathbf{x} &= \mathbf{x} - \frac{\mathbf{u}}{H} \cdot (\mathbf{x} \mp |\mathbf{x}| \mathbf{e}_1)^T \cdot \mathbf{x} = \mathbf{x} - \frac{2\mathbf{u} \cdot (|\mathbf{x}|^2 \mp |\mathbf{x}| x_1)}{2|\mathbf{x}|^2 \mp 2|\mathbf{x}| x_1} \\ &= \pm |\mathbf{x}| \mathbf{e}_1 \end{aligned} \quad (1.32)$$

也就是说这样定义的 $\mathbf{P}$ 对 $\mathbf{x}$ 的操作会使得除了第一个元素外的所有元素都成为 0，从而对矩阵 $\mathbf{A}$ 的第一步转换操作结果

$$\begin{aligned} \mathbf{A}' &= \mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P} \\ &= \begin{pmatrix} a_{11} & k_1 & 0 & \cdots & 0 \\ k_1 & & & & \\ 0 & & & & \\ \vdots & & & & \\ 0 & & & & \end{pmatrix} \end{aligned} \quad (1.33)$$

其中空白处矩阵元的值无关紧要， $k = \pm \sum_{i=2}^N a_{i1}$ 。第二步操作时，取 $\mathbf{u}$ 的前两个矩阵元为 0，

后面的矩阵元是 $\mathbf{A}$ 的第二列的值，则第二步操作完毕时，本征矩阵的形式为

$$\mathbf{A}'' = \begin{pmatrix} a_{11} & k_1 & 0 & 0 & \cdots & 0 \\ k_1 & a_{22} & k_2 & 0 & \cdots & 0 \\ 0 & k_2 & & & & \\ 0 & 0 & & & & \\ \vdots & \vdots & & & & \\ 0 & 0 & & & & \end{pmatrix} \quad (1.34)$$

显然，按照这个算法操作 $N-2$ 次后，就得到了 $\mathbf{A}$ 对应的三对角矩阵。

Householder 的算法实际实现时还要做如下改进。首先令

$$\begin{aligned} \mathbf{p} &\equiv \frac{\mathbf{A} \cdot \mathbf{u}}{H} \\ \mathbf{q} &\equiv \mathbf{p} - \frac{\mathbf{u}^T \cdot \mathbf{p}}{2H} \mathbf{u} \end{aligned} \quad (1.35)$$

则

$$\mathbf{A}' = \mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{A} - \mathbf{q} \cdot \mathbf{u}^T - \mathbf{u} \cdot \mathbf{q}^T \quad (1.36)$$

上式是程序实现时真正使用的计算公式。操作顺序也不是从第一个矩阵元开始，而是在第 m 步（ $m=1, 2, \dots, N-2$ ）时令

$$\mathbf{u}^T = (a_{i1}, a_{i2}, \dots, a_{i,i-2}, a_{i,i-1} \pm \sqrt{\sigma}, 0, \dots, 0) \quad (1.37)$$

其中 $i \equiv N-m+1 = N, N-1, \dots, 3$, $\sigma = a_{i1}^2 + \dots + a_{i,i-1}^2$ 。其中 $\sqrt{\sigma}$ 前面的符号取为和 $a_{i,i-1}$ 的相同以减少数值截断误差。程序实现时的各个变量计算顺序为 $\sigma, \mathbf{u}, H, \mathbf{p}, K, \mathbf{q}, \mathbf{A}'$ 。在第 m 步时 $\mathbf{A}$ 的最后 $m-1$ 行和列的值为 0。如果得到的三对角矩阵对应的本征矢量矩阵为 $\mathbf{Q}_0$ ，则 $\mathbf{A}$ 对应的本征矢量矩阵为 $\mathbf{Q} = \mathbf{P}_1 \cdot \mathbf{P}_2 \cdots \mathbf{P}_{N-2} \mathbf{Q}_0$ 。

1.4. 三对角矩阵的本征值和本征矢量

得到三对角矩阵后，如果只需要知道少部分的本征值和本征矢量，可以用方程求根的方法进行求解，否则用 **QR/QL 算法** 会更有效。

QR/QL 算法 基于以下事实：任意一个实矩阵 $\mathbf{A}$ 可以分解成一个正交矩阵 $\mathbf{Q}$ 和一个上三角矩阵 $\mathbf{R}$ 的乘积

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R} \quad (1.38)$$

等价地，也可以是一个正交矩阵 $\mathbf{Q}$ 和一个下三角矩阵 $\mathbf{L}$ 的乘积

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{L} \quad (1.39)$$

以下只讨论 QL 算法， QR 算法同理。因为 $\mathbf{Q}$ 是正交矩阵，所以有

$$\mathbf{L} = \mathbf{Q}^T \cdot \mathbf{A} \quad (1.40)$$

从而有

$$\mathbf{A}' = \mathbf{Q}^T \cdot \mathbf{A} \cdot \mathbf{Q} \quad (1.41)$$

即可以用 $\mathbf{Q}$ 对 $\mathbf{A}$ 进行正交变换。 QL 算法循环执行以下操作直至 $\mathbf{A}$ 成为对角矩阵：

$$\begin{aligned} \mathbf{A}_s &= \mathbf{Q}_s \cdot \mathbf{L}_s \\ \mathbf{A}_{s+1} &= \mathbf{L}_s \cdot \mathbf{Q}_s \left(= \mathbf{Q}_s^T \cdot \mathbf{A}_s \cdot \mathbf{Q}_s \right) \end{aligned} \quad (1.42)$$

即在第 s 步对 $\mathbf{A}$ 矩阵进行 QL 分解（有专门的算法，对于三对角矩阵的复杂度为 $O(N)$ ），

再由 $\mathbf{L}$ 矩阵和 $\mathbf{Q}$ 矩阵得到新的 $\mathbf{A}$ 。可以证明循环操作的结果会使得 $\mathbf{A}$ 最终对角化。

算法实现时还有一些提高效率的改进方法，参见 W. H. Press et al., 477-481 页。

1.5. 厄密矩阵的求解

对复数矩阵元组成的厄密矩阵的求解，一种思路是把 Jacobi 算法或者 Householder+ QL 算法推广到复数形式。另一种思路是把复数矩阵转换成一个实对称矩阵，从而对实对称矩阵的算法可以直接用于对厄密矩阵的求解。如果厄密矩阵是 $\mathbf{C} = \mathbf{A} + i\mathbf{B}$ ，则 $N \times N$ 的复矩阵本征问题可以写成

$$(\mathbf{A} + i\mathbf{B}) \cdot (\mathbf{u} + i\mathbf{v}) = \lambda (\mathbf{u} + i\mathbf{v}) \quad (1.43)$$

等价于 $2N \times 2N$ 的实对称矩阵本征问题

$$\begin{pmatrix} \mathbf{A} & -\mathbf{B} \\ \mathbf{B} & \mathbf{A} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \quad (1.44)$$

因为 $\mathbf{C}$ 是厄密的，所以 $\mathbf{A}^T = \mathbf{A}, \mathbf{B}^T = -\mathbf{B}$ ，因此上式中的大矩阵是实对称阵。如果 $\mathbf{C}$ 的本征值是 $\lambda_1, \lambda_2, \dots, \lambda_N$ ，则上式的本征值是 $\lambda_1, \lambda_1, \lambda_2, \lambda_2, \dots, \lambda_N, \lambda_N$ ，对应的本征矢量是 $\mathbf{u} + i\mathbf{v}$

和 $i(\mathbf{u} + i\mathbf{v})$ ，所以求解式(1.44)得到本征值和本征矢量后，只要隔一个取一个值和矢量就可以得到 $\mathbf{C}$ 的本征解。这一方法比直接用复数形式求解要慢一倍和多一倍的存储空间。

非对称矩阵的解法不在此介绍，有兴趣可以参看 W. H. Press et al., 482-493 页。