计算物理讲义

冯旭

1 数值计算的基础

2 线性方程组的直接解法

3 内插与函数的计算

4 数值积分

5 方程求根与函数求极值

6 本征值和本征矢量数值求解

这一章中，我们在一开始的时候将继续讨论和线性代数有关的内容，特别是本征值问题。我们也将对大型稀疏矩阵的线性代数问题进行简单的介绍。

6.1 QR 算法

本节将着重介绍著名的 QR 算法，它是 20 世纪十大算法之一，也是目前各界应用的最为广泛的数值求解本征值和本征矢量的算法之一。一般来说，如果待求的矩阵不是特别巨大 ($n \leq 3000$)，那么 QR 算法实际上是最为合适的算法。

在讲 QR 算法之前，我们不妨来考虑这样两个问题：

- 首先，什么样的矩阵易于求出全部本征值 $\Rightarrow$ 这个问题等价于，我们需要先对矩阵做哪些有用的约化，使得它容易求解本征值
- 第二个问题是，如果我们要对矩阵进行操作，那么什么样的操作变换能保持本征值不变

对于第一个问题，我们知道对角型、上 (下) 三角型的矩阵很容易求本征值，本征值就是他们的对角线元素。对于分块对角型，或者分块上 (下) 三角型的矩阵，如果对角块的维数较小，本征值也是容易求的。对于第二个问题，我们知道相似变换 $C^{-1}AC$ 能保持矩阵特征值不变，但在实际计算中，C 为正交矩阵是一个更好的选择，因为正交矩阵很容易求逆。而且正交矩阵因为要满足正交性，通常矩阵元素的数量级差别不大，有关的计算是数值稳定的。

所以 QR 算法的出发点就是基于以上两个考虑，它的想法是要寻找任意实矩阵 $A \in R^{n\times n}$ 的正交变换 $T = Q^T AQ$，使得 T 是个上三角矩阵，其中 $Q \in R^{n\times n}$ 是实正交矩阵。正交变换保证了 T 的本征值肯定与 A 的本征值一致。在实际做的过程当中，我们是先对矩阵 A 做 QR 分解，把 A 写成

正交矩阵和上三角矩阵的乘积，然后做迭代得到 $T = Q^T AQ$。它的步骤为：取 $T_0 = A$，然后对 T_k $(k = 0, 1, 2, \cdots)$ 做 QR 分解

$$T_k = Q_k R_k \tag{1}$$

得到 Q_k 和 R_k 以后，将它顺序相反乘在一起，得到

$$T_{k+1} = R_k Q_k \tag{2}$$

所以 $T_{k+1} = Q_k^T T_k Q_k$，每一步迭代都在做正交变换，我们希望迭代到最后，$\lim_{k\to\infty} T_k = T$。由于迭代之后新的矩阵都与原先的矩阵只差一个正交变换。因此新的矩阵的条件数一定不坏于原先的矩阵。这一点对于算法的稳定性是非常重要的。

QR 算法的初衷是利用正交变换将 $A \in R^{n\times n}$ 约化为一个上三角矩阵。不幸的是，这一点并不是总能做到的。因为一个任意的实矩阵有可能有成对的复本征值，但是实的正交矩阵变换后的矩阵元一定仍然保持是实的。如果上述变换总能成立就意味着变换后的三角矩阵的对角元 –也就是它的本征值 – 一定都是实数，与一般实矩阵可以有复本征值矛盾。但我们可以把它约化到一个近似的上三角矩阵:

$$Q^T AQ = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & R_{mm} \end{pmatrix} \tag{3}$$

其中位于对角线上的块矩阵 R_{ii} 要么就是一个实数要么就是一个 2×2 的、具有一对复共轭根的实矩阵。我们称这样形式的矩阵为实矩阵 $A \in R^{n\times n}$ 的实舒尔分解 (real Schur decomposition) 或者实舒尔形式。如果矩阵的本征值全为实数，且各不相等，那么实舒尔形式可以约化为完全上三角形式。

原则上只要我们不断进行 QR 迭代，最后矩阵就会被约化为实舒尔形式，但我们还需要知道这个迭代的收敛速度。我们有下面这个定理：给定 $A \in R^{n\times n}$ 并假设它的本征值的排序为

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| \tag{4}$$

那么我们有

$$\lim_{k\to\infty} T_k = \begin{pmatrix} \lambda_1 & t_{12} & \cdots & t_{1n} \\ 0 & \lambda_2 & \cdots & t_{2n} \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix} \tag{5}$$

其中非对角元的收敛速度为

$$|t_{i,j}^{(k)}| \sim O\left(\left|\frac{\lambda_i}{\lambda_j}\right|\right)^k, \quad i > j, \quad k \to \infty \tag{6}$$

此外，如果加上 A 是对称矩阵的假设，那么 T_k 可以直接收敛到对角矩阵。

这个定理告诉我们，如果矩阵的所有本征值的模都不相等，那么经过 QR 迭代，我们不仅仅会获得矩阵的实舒尔形式，我们甚至可以直接获得其舒尔形式 –即所有的本征值。但同时我们也看到，如果矩阵具有两个模十分接近的本征值，那么相应的非对角元可以收敛得很慢。当矩阵具有完全等模的本征值时，QR 迭代甚至可能不收敛。

我们可以看到 QR 算法执行的每一步迭代，都需要算一次 QR 分解，它的计算量大概是 $O(n^3)$。所以总的计算量是 $O(n^3)$ 再乘上迭代次数，这个迭代次数主要与矩阵的本征值分布相关，对于收敛很慢的矩阵，也许需要迭代的次数非常大。因此，有必要在我们进行迭代之前，先将矩阵化为简单的形式。常用的方法包括 Householder 变换和 Givens 变换。我们下面先讲一下 Householder 变换。

6.1.1 Householder 变换

事实上，Householder 变换是 Householder 形式化的矩阵计算的分解方法的一部分，也是属于 20 世纪十大算法之一。我们考虑 $v \in R^n$ 中的一个矢量。构造如下的矩阵:

$$P = 1_{n\times n} - 2vv^T/||v||_2^2 \tag{7}$$

其中 $||v||^2 = v^T v$ 是矢量 v 的欧氏模方。当我们将这个矩阵乘以 $x \in R^n$ 时，即 $y = Px$，它的作用将原来的矢量 x 相对于以 v 为法线方向的超平面进行镜面反射。或者说，x 中与 v 垂直的分量不变，但是与 v 平行的部分反了一个符号 (镜面反射的特点)。这个变换一般称为 Householder 变换; 相应的 v 称为 Householder 矢量，P 称为与 v 相对应的 Householder 矩阵。注意，Householder 矩阵并不改变矢量的长度，因此实际上是一个正交矩阵。而且它还是一个对称矩阵。另一点值得指出的是，当我们需要计算一个 Householder 矩阵乘以一个矢量 (例如 Px) 的时候，我们并不需要首先将 Householder 矩阵存在内存中，然后计算矩阵乘以矢量，我们需要的只是两个矢量的内积 $v^T x$，$v^T v$ 等等。因此仅仅是一个 $O(n)$ 量级的计算而不是通常的 $O(n^2)$ 的计算。

我们很容易证明下面这个定理，如果有两个矢量 x 和 y，他们的欧氏模方相同，那么我们可以通过 Householder 变换将 x 变成 y : $Px = y$。这里我们只需要将矢量 v 的方向取为 $v = x - y$ 即可。这个定理的一个推论是，我们一定可以把 x，通过 Householder 变换，变换成为只有第 m 个方向有非零分量的矢量

$$Px = (0, \cdots, 0, \pm||x||_2, 0, \cdots, 0)^T \tag{8}$$

这里只需要把 y 取为 $y = \mp||x||_2 e_m$，而矢量 v 取为 $v = x \pm ||x||_2 e_m$。

我们可以直接用 Householder 变换对矩阵 A 进行 QR 变换

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \to \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \to \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix} \tag{9}$$

但这并不是我们要讨论的重点。因为这只是对矩阵 A 左乘了一系列的正交矩阵，它做的并不是正交变换。我们希望做的一件事情，是利用 Householder 矩阵将任意的 $n \times n$ 矩阵通过正交变换化为上 Hessenberg 形式。

如果我们希望对矢量 x 的前 k 个分量保持不变，将从第 $k+2$ 个以及以后的所有分量都设为零。由于 Householder 为正交矩阵不改变矢量的模，因此新的矢量的第 $k+1$ 个分量一定大小为 $||x^{(n-k)}||_2$，其中 $x^{(n-k)}$ 为矢量 x 的后 $n-k$ 个分量构成的矢量。对于这个操作我们可以用如下的 Householder 矩阵，

$$P_{(k)} = \begin{pmatrix} 1_k & 0 \\ 0 & R_{n-k} \end{pmatrix}, \quad R_{n-k} = 1_{n-k} - 2\frac{\omega^{(k)}(\omega^{(k)})^T}{||\omega^{(k)}||_2^2} \tag{10}$$

其中 $\omega^{(k)} \in R^{n-k}$ 为 R^{n-k} 中的一个矢量。我们应当取

$$\omega^{(k)} = x^{(n-k)} \pm ||x^{(n-k)}||_2 e_1^{(n-k)} \tag{11}$$

其中 $e_1^{(n-k)}$ 是 R^{n-k} 中的第一个单位矢量。很明显 $P_{(k)}$ 满足 $P_{(k)}^T = P_{(k)}$ 和 $P_{(k)}^T P_{(k)} = 1$。

有了 $P_{(k)}$ 以后，我们把矩阵 A 看成 2×2 的分块矩阵 $A = \begin{pmatrix} a_{11} & r_1^T \\ c_1 & A_{22} \end{pmatrix}$，我们对它左乘 $P_{(1)}^T$，右乘 $P_{(1)}$，得到

$$P_{(1)}^T A = \begin{pmatrix} a_{11} & r_1^T \\ \gamma_1 e_1^{(n-1)} & R_{n-1} A_{22} \end{pmatrix}, \quad P_{(1)}^T A P_{(1)} = \begin{pmatrix} a_{11} & r_1^T R_{n-1} \\ \gamma_1 e_1^{(n-1)} & R_{n-1} A_{22} R_{n-1} \end{pmatrix} \to \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \tag{12}$$

对于任意的 $A \in R^{n \times n}$，我们可以利用一连串矩阵: $P_{(1)} \cdots P_{(n-2)}$ 试图将矩阵化为上 Hessenberg 形式。事实上，令 $A^{(0)} \equiv A$，对任意的 $k \geq 1$，我们有

$$A^{(k)} = P_{(k)}^T A^{(k-1)} P_{(k)} = (P_{(1)} \cdots P_{(k)})^T A (P_{(1)} \cdots P_{(k)}) \tag{13}$$

或者等价地写成 $A^{(k)} = Q_{(k)}^T A^{(0)} Q_{(k)}$，其中正交矩阵 $Q_{(k)} = P_{(1)} \cdots P_{(k)}$ 为一系列 Householder 矩阵的乘积。基本上经过第一次变换:$A^{(1)} = P_{(1)}^T A^{(0)} P_{(1)}$，原先矩阵的第一列中 a_{21} 以下的数都会被变换为零; 经过第二个变换，在保持第一列的结构的同时，将矩阵的第二列的 a_{32} 以下的矩阵元都变换为零，等等。利用 Householder 矩阵将矩阵变换为 Hessenberg 形式的操作一般称为 Hessenberg-Householder 约化 (Hessenberg-Householder reduction)。下式以 4×4 矩阵为例，演示了这种约化的过程:

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \xrightarrow{P_{(1)}} \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \xrightarrow{P_{(2)}} \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix} \tag{14}$$

值得提及的一点是，如果原先的矩阵 A 是一个对称矩阵，那么 Householder 变换后其对称性仍然保持。因此，在经过 Householder 约化之后，一个对称矩阵一定能够化为一个对称的、三对角矩阵。因为 Householder 变换要保证每个列矢量的模不变，所以我们没法直接得到对角矩阵，这当然也和我们之前提到过的并不是每个矩阵都可以对角化的有关。

如果我们把矩阵 A 化成了一个上 Hessenberg 矩阵，由于 Hessenberg 矩阵已经很接近上对角的形式，那么再对 Hessenberg 矩阵做 QR 分解，计算量就会大大下降。事实上每次 QR 迭代的计算量可以从 $O(n^3)$ 减少到 $O(n^2)$。如果 Hessenberg 矩阵是三对角的形式，那么 QR 迭代的计算量可以进一步减小。另外，Hessenberg 矩阵的大量矩阵元都已经为 0。我们知道，做 QR 迭代时候，实际上收敛的效果是让很多非对角元直接收敛到 0。对于 Hessenberg 矩阵来讲，我们只需使少量的非零元收敛到 0，QR 迭代所需的步数也大为减小。

6.1.2 Givens 旋转变换

我们利用 Hessenberg-Householder 约化可以将任意的矩阵 $A \in R^{n \times n}$ 通过正交变换化为上 Hessenberg 形式。我们可以进一步利用 Givens 变换矩阵 (又称为 Givens 转动矩阵) 把矩阵化成实舒尔形

式

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad G^{-1} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \tag{15}$$

其中 $c = \cos\theta$, $s = \sin\theta$, 这个 2×2 矩阵就是 Givens 矩阵。它的逆变换也很简单。适当选取 θ 的值, 我们可以用 Givens 旋转矩阵来消去向量的某个分量, 例如

$$Gx = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \end{pmatrix} \tag{16}$$

其中 $\alpha = \sqrt{x_1^2 + x_2^2}$。因为 G 是正交矩阵, 那么我们必有 $|\alpha| = ||x||_2$, 则

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \quad s = \frac{x_2}{\sqrt{x_1^2 + x_2^2}} \tag{17}$$

如果 x_1 或者 x_2 是很大的值, 有时候为了避免上溢, 会对公式进行调整, 如果 $|x_1| \geq |x_2|$, 可以计算

$$t = \frac{x_2}{x_1}, \quad c = \frac{1}{\sqrt{1 + t^2}}, \quad s = ct. \tag{18}$$

一个在第一、第二轴的平面内转动的 Givens 矩阵的形式为:

$$G(1, 2, \theta) = \begin{pmatrix} G(\theta) & 0 \\ 0 & 1_{(n-2)\times(n-2)} \end{pmatrix} \tag{19}$$

完成一次 Givens 变换以后 ($G_1^{(1)} = G(1, 2, \theta_1)$), 我们得到

$$H^{(0)} = \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix} \quad G_1 H^{(0)} = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix} \quad G_2 H^{(0)} = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \tag{20}$$

我们可以用一系列的 Givens 矩阵来实现 QR 分解。假设 T_k 为上 Hessenberg 矩阵, 我们有

$$G_{n-1} \cdots G_2 G_1 T_k = R_k \quad \Rightarrow \quad Q_k = (G_{n-1} \cdots G_2 G_1)^T \tag{21}$$

那么 $T_{k+1} = R_k Q_k$, 因为无论是左乘 Q_k^T 还是右乘 Q_k, 都是对同一个 2×2 矩阵进行操作, 所以 T_{k+1} 和 T_k 一样, 仍然具有上 Hessenberg 形式。然后我们可以用 Givens 矩阵对 T_{k+1} 进行 QR 分解, 然后一直迭代下去。因为是在执行 QR 算法, 上 Hessenberg 形式里的非对角元会随着迭代趋于 0。这里我们采用 Givens 变换的好处是我们并不需要算出 QR 分解中的矩阵 Q, 实际计算中只需要记录所用到的 Givens 旋转变换的参数即可。

Givens 旋转变换是一个转动操作, 对于矩阵有一对复共轭特征值的情形, 就无法用 Givens 旋转变换来实现 QR 算法。这个时候, 我们可以采用 Shifted QR 的方案。

6.1.3 Shifted QR

对于具有模非常接近的本征值的时候, QR 迭代很可能收敛得很慢甚至根本不收敛于矩阵的实舒尔形式。为了克服这一点, 我们可以引进具有单个 shift 的 QR 迭代。带位移的 QR 算法迭代公式如下

$$\begin{cases} Q_k R_k = T_k - \mu I & \text{做 QR 分解} \\ T_{k+1} = R_k Q_k + \mu I \end{cases} \tag{22}$$

我们很容易验证 T_{k+1} 和 T_k 是正交相似的：$T_{k+1} = R_k Q_k + \mu I = Q_k^T (T_k - \mu I) Q_k + \mu I = Q_k^T T_k Q_k$。我们选择 $\mu \in R$ 使得

$$|\lambda_1 - \mu| \geq |\lambda_2 - \mu| \geq \cdots |\lambda_n - \mu| \tag{23}$$

这样一来变换过程中第 k 次迭代时的非对角元 $t_{j,j-1}^{(k)}$ 大致会按照 $|(\lambda_j - \mu)/(\lambda_{j-1} - \mu)|^k$ 的方式趋于零。因此，一个比较自然的选择是令:

$$|\lambda_n - \mu| \ll |\lambda_i - \mu|, \quad i = 1, \cdots, n-1 \tag{24}$$

那么矩阵元 $t_{n,n-1}^{(k)}$ 会快速地趋于零，所以 $t_{n,n}^{(k)}$ 最先收敛到本征值。在实际的应用中，人们一般会选择 $\mu = t_{n,n}^{(k)}$。当我们得到这个本征值以后，我们可以删除第 n 行、第 n 列，然后从剩下的子矩阵中求其他的本征值。

对于实非对称矩阵，可能存在一对复共轭特征值，这个时候可以引入 double shift 的算法来改善 QR 迭代。我们考虑第 $n-1$ 和第 n 行、第 $n-1$ 和第 n 列的 4 个元素构成的一个 2×2 矩阵 G 包含了一对复共轭特征值，分别为 λ 和 $\bar{\lambda}$。我们引入一个复参数 μ，使得

$$|\lambda - \mu| \ll |\lambda_i - \mu| \tag{25}$$

构造 QR 分解

$$\begin{aligned} T - \mu I &= Q_1 R_1, \\ T_1 &= R_1 Q_1 + \mu I, \\ T_1 - \bar{\mu} I &= Q_2 R_2, \\ T_2 &= R_2 Q_2 + \bar{\mu} I \end{aligned} \tag{26}$$

这里 Q_1 和 Q_2 是幺正矩阵，我们有 $T_1 = Q_1^\dagger T Q_1$，$T_2 = Q_2^\dagger T_1 Q_2$。从第二个和第三个式子，可以得到

$$R_1 Q_1 + (\mu - \bar{\mu}) I = Q_2 R_2 \tag{27}$$

把这个式子左乘 Q_1、右乘 R_1，我们得到

$$Q_1 R_1 Q_1 R_1 + (\mu - \bar{\mu}) Q_1 R_1 = Q_1 R_1 (Q_1 R_1 + (\mu - \bar{\mu}) I) = (T - \mu I)(T - \bar{\mu} I) = Q_1 Q_2 R_2 R_1 \tag{28}$$

这里 $(T - \mu I)(T - \bar{\mu} I)$ 肯定是个实矩阵，因此 $Q_1 Q_2 R_2 R_1$ 是对实矩阵的 QR 分解。因此，我们一定可以通过选择适当的幺正矩阵 Q_1 和 Q_2，使得 $Z = Q_1 Q_2$ 是实正交矩阵。于是，我们有

$$T_2 = (Q_1 Q_2)^\dagger T (Q_1 Q_2) = Z^T T Z \tag{29}$$

这里, 我们可以看到, 虽然单个的变换用的是幺正变换, 但联合变换是正交变换。因为 $|\lambda - \mu| \ll |\lambda_i - \mu|$, double shift 能够帮助我们加速 QR 迭代的收敛。在实际操作中，我们选取

$$2\,\mathrm{Re}(\mu) = \mathrm{Tr}(G) = t_{n-1,n-1}^{(k)} + t_{n,n}^{(k)}, \quad |\mu|^2 = \det(G) = t_{n-1,n-1}^{(k)} t_{n,n}^{(k)} - t_{n-1,n}^{(k)} t_{n-1,n}^{(k)} \tag{30}$$

然后可以计算实矩阵的 QR 分解

$$(T - \mu I)(T - \bar{\mu} I) = T^2 - 2\,\mathrm{Re}(\mu) T + |\mu|^2 = ZR \tag{31}$$

得到 Z 以后，设定 $T_2 = Z^T T Z$。

小结：十大算法中的 QR 算法来解本征值。QR 算法的基本步骤是 QR 分解，它是要把一个矩阵分解成实正交矩阵和上三角矩阵相乘的形式：$T_k = Q_k R_k$。但做完分解，我们并不能马上从 R_k 得到矩阵的本征值，我们让 R_k 右乘 Q_k，试图构造变换 $T_{k+1} = R_k Q_k = Q_k^T T_k Q_k$，那么 T_{k+1} 和 T_k 是相似变换，所以本征值完全一样，但这里有个问题，虽然 $Q_k R_k$ 分解中得到的 R_k 是上三角矩阵，但 T_{k+1} 因为右乘了 Q_k，却并不是上三角矩阵。数学上可以证明，T_{k+1} 比 T_k 要更趋近于实舒尔型，于是我们经过 $T_k \to T_{k+1} \to T_{k+2} \to \cdots$ 的迭代，最终 $\lim_{k\to\infty} T_k$ 会给出我们想要的实舒尔形式。

当然，纯做 QR 迭代，每次迭代都要花费 $O(n^3)$ 的计算，是很耗时的。于是我们引进了 Householder 变换，将一般矩阵化成上 Hessenberg 矩阵的形式，这样有两个好处，一是对上 Hessenberg 矩阵做 QR 分解，每次迭代只需花费 $O(n^2)$ 的计算量，其次，QR 迭代的收敛速度也大为增加。这里 Householder 变换也是属于 20 世纪十大算法之一。有了上 Hessenberg 矩阵，我们可以引进 Givens 旋转矩阵来做 QR 分解。

除此之外，我们还介绍了 Shifted QR 的技术，包括单点 shift，和 double shift，其中 double shift 是用来处理复共轭本征对的情况，采用 shift 的技术，能使 QR 迭代的收敛性大为改善。

6.1.4 由矩阵的实舒尔形式计算其本征矢

假设通过前面讲述的 QR 算法我们已经得到了矩阵的实舒尔形式，$Q^T A Q = T$，如果我们不考虑出现 2×2 包含复共轭根的情况，(如果包含复共轭根，怎么从实舒尔形式得到本征矢量，留给同学们思考。) 那么 T 的对角元就是原先矩阵的本征值。假定对于某个本征值 λ，我们希望求解其本征矢 x。容易验明，若 $Ax = \lambda x$，则令 $y = Q^T x$，我们一定有 $Ty = \lambda y$。因此，为了求出 x 我们可以直接先求 y，然后再以 Q 左乘之即可。我们令 $\lambda = t_{kk} \in R$ 为 A 的某个单一本征值。那么矩阵 T 的形式一定为:

$$\begin{pmatrix} T_{11} & \nu & T_{13} \\ 0 & \lambda & \omega^T \\ 0 & 0 & T_{33} \end{pmatrix} \tag{32}$$

其中 $T_{11} \in R^{(k-1)\times(k-1)}$，$T_{33} \in R^{(n-k)\times(n-k)}$ 是两个上三角矩阵；$\nu \in R^{(k-1)}$，$w \in R^{(n-k)}$ 为两个矢量。按照假定，λ 是一个单一的本征值，因此它一定不会出现在 T_{11} 或者 T_{33} 的谱中。这意味着矩阵 $(T_{11} - \lambda I_{(k-1)\times(k-1)})$ 和 $[T_{33} - \lambda I_{(n-k)\times(n-k)}]$ 都是非奇异的上三角矩阵。我们假定 $y = (y_{k-1}^T, y', y_{n-k}^T)$，其中 $y_{k-1} \in C^{k-1}$，$y_{n-k} \in C^{n-k}$，于是本征方程 $(T - \lambda I)y = 0$ 可以明确地写为

$$\begin{cases} (T_{11} - \lambda I_{(k-1)\times(k-1)}) y_{k-1} + y'\nu + T_{13} y_{n-k} = 0 \\ \omega^T y_{n-k} = 0 \\ [T_{33} - \lambda I_{(n-k)\times(n-k)}] y_{n-k} = 0 \end{cases} \tag{33}$$

由于 $(T_{11} - \lambda I_{(k-1)\times(k-1)})$ 和 $[T_{33} - \lambda I_{(n-k)\times(n-k)}]$ 都是非奇异地，因此上述方程可以“解出”(不失一般性，我们可以令 $y' = 1$) 如下的解:

$$y = \begin{pmatrix} -(T_{11} - \lambda I_{(k-1)\times(k-1)})^{-1}\nu \\ 1 \\ 0 \end{pmatrix} \tag{34}$$

最终，原来矩阵 A 的本征矢 x 可以由 $x = Qy$ 给出。

6.2 奇异值分解

与本征值有关的，是所谓矩阵的奇异值。奇异值分解是一项应用十分广泛的计算技术。首先让我们回忆这个线性代数中十分重要的定理。

对于一个 $m \times n$ 的复矩阵 $A \in C^{m\times n}$，一定存在两个幺正矩阵 $U \in C^{m\times m}$ 和 $V \in C^{n\times n}$，它们能够将 A“对角化”:

$$U^{\dagger}AV = \Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_p) \in C^{m\times n}, \quad p = \min(m, n) \tag{35}$$

其中 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ 称为矩阵 A 的奇异值 (singular values) 而上式则称为矩阵 A 的奇异值分解 (singular value decomposition, SVD)。

矩阵的那些非零的奇异值实际上是矩阵 $A^{\dagger}A$ 的本征值的根号:

$$\sigma_i(A) = \sqrt{\lambda_i(A^{\dagger}A)} \tag{36}$$

一个矩阵的奇异值分解的计算一般运用 Golub-Kahan-Reinsch 算法。不失一般性，我们假定 $A \in R^{m\times n}$ 且 $m \geq n$，否则我们就计算 A^T 的奇异值分解。奇异值分解分为以下步骤

- 首先，矩阵 A 先被变换为如下形式

$$\bar{U}^T A \bar{V} = \begin{pmatrix} B \\ 0 \end{pmatrix} \tag{37}$$

其中 $\bar{U}$ 和 $\bar{V}$ 是两个正交矩阵，$B \in R^{n\times n}$ 是一个上双对角 (upper bidiagonal) 矩阵。矩阵 $\bar{U}$ 和 $\bar{V}$ 由以下步骤，通过 $n + m - 3$ 个 Householder 矩阵 $\bar{U}_1, \cdots, \bar{U}_{m-1}$ 和 $\bar{V}_1, \cdots, \bar{V}_{n-2}$ 依次产生。首先我们将 $(\bar{U}_1)^T$ 左乘到 A 上：$A^{(1)} = (\bar{U}_1)^T A$，使 $A^{(1)}$ 的第一列的第二个矩阵元以下都为零; 然后将 $\bar{V}_1$ 右乘到 $A^{(1)}$ 上得到 $A^{(2)} = A^{(1)}\bar{V}_1$，使得 $A^{(2)}$ 的第一行的第三个矩阵元右边的全部为零，同时不破坏第一列的那些已经化为零的矩阵元

$$A^{(1)} = \bar{U}_1^T A = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \Rightarrow A^{(2)} = A^{(1)}\bar{V}_1 = \begin{pmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \tag{38}$$

我们很容易验证，B 的奇异值与 A 的奇异值相同，因为

$$B^T B = (\bar{V}^T A^T \bar{U})(\bar{U}^T A \bar{V}) = \bar{V}^T A^T A \bar{V} \tag{39}$$

$B^T B$ 是 $A^T A$ 的相似矩阵

- 第二步，我们可以利用 QR 迭代将上双对角矩阵 B 对角化。即

$$B = B^{(0)} \to B^{(1)} \to \cdots \to \Sigma \tag{40}$$

其中 QR 迭代满足

$$B^{(i+1)} = [S^{(i)}]^T B^{(i)} T^{(i)} \tag{41}$$

这里 $S^{(i)}$ 和 $T^{(i)}$ 都是正交矩阵。$T^{(i)}$ 的作用是让 $M^{(i)} = [B^{(i)}]^T B^{(i)}$ 矩阵收敛到一个对角矩阵，而 $S^{(i)}$ 的作用是让 $B^{(i)}$ 保留上双对角的形式。具体的，对于每一步迭代，为了方便起见，我们去掉上标 (i)，而把符号记为

$$\hat{B} \equiv B^{(i)}, \quad \bar{B} \equiv B^{(i+1)}, \quad S \equiv S^{(i)}, \quad T \equiv T^{(i)}, \quad M \equiv \hat{B}^T \hat{B}, \quad \bar{M} = \bar{B}^T \bar{B} \tag{42}$$

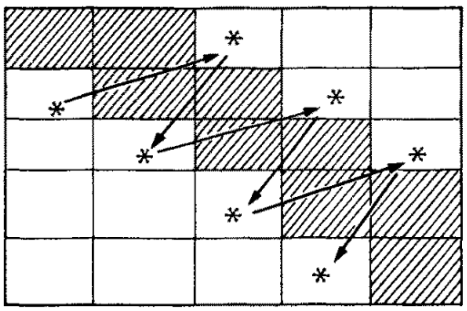
$\hat{B} \to \bar{B}$ 的变换可以由 Givens 转动得到

$$\bar{B} = \underbrace{S_n^T S_{n-1}^T \cdots S_2^T}_{S^T} \hat{B} \underbrace{T_2 T_3 \cdots T_n}_{T} \tag{43}$$

$$S_k = \begin{bmatrix} 1 & 0 & & & & & \\ 0 & \ddots & & & & 0 & \\ & & 1 & & & & \\ & & & \cos\theta_k & -\sin\theta_k & & \\ & & & \sin\theta_k & \cos\theta_k & & \\ & & & & & 1 & \\ & 0 & & & & \ddots & 0 \\ & & & & & 0 & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ (k-1) \\ (k) \\ \\ \\ \end{matrix}$$

$(k-1)$ (k)

T_k 的形式取成和 S_k 一样，只是参数换成了 φ_2。选取一个任意的 φ_2，$\hat{B}T_2$ 的效果是产生了一个 B_{21} 的矩阵元，再乘上 S_2 的效果是去掉 B_{21} 的矩阵元，但是又产生了 B_{13} 的矩阵元，这样乘下去，到最后一步，S_n^T 负责去掉 $B_{n,n-1}$，但不产生任何新的矩阵元



我们把这个过程叫做“追逐”(chasing)。由于 $\bar{B} = S^T \hat{B} T$,

$$\bar{M} = \bar{B}^T \bar{B} = T^T M T \tag{44}$$

其中 $\bar{M}$ 和 M 都是三对角矩阵。到目前为止，我们还有一个自由的参量 φ_2 可供选择。如果我们能够通过选择适当的 φ_2，使得 T 矩阵刚好是 M 的 QR 分解里的 Q 矩阵，那么从 $M \to \bar{M}$ 就是 QR 迭代了。事实上，这件事情是可以做到的，只需要把 T_2 的第一列矩阵元取成正比 M 的第一列矩阵元的形式即可。这里，我们不做具体的证明，只是借助一个 2×2 的 $\hat{B}$ 矩阵为例，做一下说明

$$\hat{B} = \begin{pmatrix} x & z \\ 0 & y \end{pmatrix}, \quad M = \begin{pmatrix} x^2 & xz \\ xz & y^2 + z^2 \end{pmatrix}, \quad T = \frac{1}{\sqrt{x^2 + z^2}} \begin{pmatrix} x & -z \\ z & x \end{pmatrix} \tag{45}$$

M 可以分解成

$$M = \frac{1}{\sqrt{x^2+z^2}} \begin{pmatrix} x & -z \\ z & x \end{pmatrix} \cdot \sqrt{x^2+z^2} \begin{pmatrix} x & \alpha \\ 0 & \beta \end{pmatrix} \tag{46}$$

这里 α、β 满足方程

$$\begin{cases} x\alpha - z\beta = xz \\ z\alpha + x\beta = y^2 + z^2 \end{cases} \tag{47}$$

在这种情况下，T 矩阵确实是 M 的 QR 分解里的 Q 矩阵。

把 W 和 Z 记为

$$W = S^{(0)}S^{(1)}\cdots, \quad Z = T^{(0)}T^{(1)}\cdots \tag{48}$$

则我们获得了两个正交矩阵 W 和 Z 使得

$$W^T B Z = \Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_n) \tag{49}$$

其中 $\sigma_i, i = 1, \cdots, n$ 就是矩阵 A 的奇异值。于是矩阵 A 的奇异值分解可以表达为,

$$U^T A V = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} \tag{50}$$

其中的矩阵 $U = \bar{U}\,\mathrm{diag}(W, I_{(m-n)\times(m-n)})$，$V = \bar{V}Z$

如果 A 是复矩阵，那么在推导过程中的 U，V 矩阵需要替换为相应的幺正矩阵。于是我们就得到

$$U^\dagger A V = \Sigma \tag{51}$$

这个结论反过来写，就是

$$A = U\Sigma V^\dagger = \sum_{k=1}^{p} \sigma_k U_k \otimes (V^\dagger)_k \tag{52}$$

其中 U_k 是 U 矩阵的第 k 列，$(V^\dagger)_k$ 是 $V^\dagger$ 矩阵得到第 k 行 (V 矩阵的第 k 列)

或者写成矩阵元的形式

$$A_{ij} = \sum_{k=1}^{p} \sigma_k (u_i)_k (v_j)_k^* \tag{53}$$

其中 u_i 表示矩阵 U 的第 i 行矢量，而 v_j 则表示矩阵 V 的第 j 行矢量。

事实上奇异分解具有很广泛的应用，其中一种应用就是图片的压缩存储。假如说我们有一张像素为 1024×512 的照片，作为数据储存的时候实际上对应于一个 1024×512 的矩阵。完整存下来需要保存 $1024 \times 512 = 5 \times 10^5$ 的数据，如果我们仅保存按奇异值从大到小排序的前 50 项，那么总共需要存储的数据为 $(1 + 1024 + 512) \times 50 = 7.5 \times 10^4$，总存储量大概为原矩阵的 15%，但由于大奇异值的信息都已包括在新的矩阵中，实际上我们能还原出和原图差别不大的图片。事实上正因为大奇异值往往对应着矩阵中隐含的重要信息，我们不仅可以把它应用在数据压缩上，也可以对图片去噪声。我们有理由相信那些较小的奇异值是由噪声引起的，我们可以强行令它们为 0。

6.3 对称矩阵的算法

6.3.1 Jacobi 算法

如果矩阵是实对称的方矩阵，那么我们可以利用下面的一些算法来计算其本征值和本征矢量。最为直接的方法就是 Jabobi 算法，它直接利用 Givens 转动将矩阵的非对角元变换为零。

设 $A^{(0)} = A \in R^{n\times n}$ 为一实对称矩阵，对一对不相等的指标 p 和 q 满足 $1 \le p < q \le n$，我们将 Givens 矩阵 $G(p,q,\theta) = G_{pq}$ 作用于 $A^{(k-1)}$ 得到新的 $A^{(k)}$：

$$A^{(k)} = G_{pq}^T A^{(k-1)} G_{pq} \tag{54}$$

我们可以选择 θ 使得转动以后的 $A^{(k)}$ 矩阵的矩阵元满足

$$a_{ij} = 0, \quad \text{如果}：(i,j) = (p,q) \tag{55}$$

由于 Givens 矩阵的结构，上述关系等价于

$$A^{(k)} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \tag{56}$$

我们发现要使非对角矩阵元为 0，$t \equiv s/c$ 需满足一个二次方程:

$$t^2 + 2\eta t - 1 = 0, \quad \eta = \frac{a_{qq} - a_{pp}}{2a_{pq}} \tag{57}$$

如果 $\eta \ge 0$，我们取 $t = 1/(\eta + \sqrt{1+\eta^2})$；如果 $\eta < 0$，我们取 $t = -1/(-\eta + \sqrt{1+\eta^2})$。这两个根写成分母的形式是为了避免两个相近的数相消，然后令

$$c = \frac{1}{\sqrt{1+\eta^2}}, \quad s = ct \tag{58}$$

在 Givens 变换的作用下，除了

$$a_{pp}, a_{pq}, a_{qq} \quad \to \quad a'_{pp}, a'_{pq} = 0, a'_{qq} \tag{59}$$

其他非对角元也发生了一些改变，比方说

$$\begin{aligned} a'_{pk} &= a'_{kp} = ca_{pk} - sa_{qk}, \quad k \ne p,q \\ a'_{qk} &= a'_{kq} = sa_{pk} + ca_{qk}, \quad k \ne p,q \end{aligned} \tag{60}$$

但这些矩阵元的总平方和没有发生改变。

为了考察 Jacobi 迭代收敛的速度我们定义矩阵的一个特征函数

$$\Psi(M) = \left(\sum_{i \ne j} m_{ij}^2 \right)^{1/2} \tag{61}$$

前面的 Jacobi 迭代给出：

$$\Psi(A^{(k)})^2 = \Psi(A^{(k-1)})^2 - 2\left(a_{pq}^{(k-1)}\right)^2 < \Psi(A^{(k-1)})^2 \tag{62}$$

因此，在进行 Jacobi 迭代时，最有效的是从其模最大的非对角元开始。但是实际上也可以就对每一个非对角元都做一遍。

Jacobi 算法仅仅适用于实对称矩阵以及它的复推广厄米矩阵。而且它的计算代价还是比较大的，基本上是 $O(n^3)$。但是它有一个优点就是超级稳定。原因就在于上面讨论的严格成立的估计。因此它特别适合于体量比较小的 (典型的 $n \leq O(100)$) 实对称矩阵或者复厄米矩阵。如果我们不仅仅要求本征值还要求本征矢，这往往会增加大约 50% 的计算量。

6.3.2 Sturm 序列

所谓的 Sturm 序列适用于计算一个实的三对角对称矩阵的本征值问题。我们假设三对角矩阵 T 的对角元为 d_i，$i = 1, 2, \cdots, n$；两个副对角线上的元素为 b_i，$i = 1, 2, \cdots, n-1$。不失一般性我们假设所有的 $b_i \neq 0$，否则经过重新排列问题可以化为一个更小的三对角矩阵的本征值问题。一个重要的观察是，这样一个三对角矩阵的特征多项式可以通过迭代的方法给出。令 $p_0(x) = 1$ 和 $p_1(x) = d_1 - x$，

$$p_i(x) = (d_i - x)p_{i-1}(x) - b_{i-1}^2 p_{i-2}(x), \quad i = 2, \cdots, n \tag{63}$$

我们在讨论正交多项式的章节里面曾经讨论过相关的迭代。容易验证 $p_n(x)$ 恰好就是 T 的特征多项式。不仅如此，事实上 $p_i(x) = \det(T_i - xI_{i\times i})$ 是 T_i 的特征多项式，其中 T_i 就是由矩阵 T 的前 i 行和 i 列构成的矩阵 (也称为主子矩阵)。上面公式给出的多项式序列称为 Sturm 序列。

Sturm 序列满足一系列重要的性质。例如 $p_i(x)$ 的根 (也就是 T_i 的本征值) 与 $p_{i-1}(x)$ 的根进行排序，可以证明，这两组根一定会交错地出现。另外一个是我们很容易确定小于某个数值的实根的个数，这就是下面的结论：对于上述的 Sturm 序列 $\{p_i(x) : i = 0, 1, \cdots, n\}$，我们选取任意的 $\mu \in R$ 并且构造如下的序列:

$$S_\mu = \{p_0(\mu), p_1(\mu), \cdots, p_n(\mu)\}, \tag{64}$$

那么上述实数序列中从前往后数的过程中，其数值变号的次数 $s(\mu)$ 就是 $p_n(x) = 0$ 所具有的严格小于 μ 的实根的数目。这里我们约定: 如果 $p_i(\mu) = 0$，我们认为它是与前面的邻居 $p_{i-1}(\mu)$ 不同号的 (即算做一次变号)。

作为一个例子，考虑 4×4 的三对角矩阵 $\begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{pmatrix}$。这个矩阵的四个本征值由小到大依次是 $\{0.38, 1.38, 2.62, 3.62\}$。如果我们计算 $\mu = 3$ 时各个多项式的值我们有: $p_0(3), p_1(3), p_2(3), p_3(3), p_4(3) = 1, -1, 0, 1, -1$，从左到右数过去一共变号三次 (按照定理中的约定，其中的 0 也算一次)，因此 $p_4(x) = 0$ 的根中有 3 个小于 3。

由于所有对称矩阵 (或者厄米矩阵) 的根都是实根，因此如果我们仅仅希望寻找它的本征值的话，我们完全可以利用前一章中所提及的各种求根的方法，其中最为稳定的当属对分法。当然，这需要首先选定一个寻找的区间，这时我们可以利用 Gershgorin 圆盘定理。定理描述如下：对任意的 $A \in C^{n\times n}$，它的谱记为 $\sigma(A)$。我们构建复平面的 n 个圆 (盘):

$$R_i = \left\{ z \in C : |z - a_{ii}| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}| \right\} \tag{65}$$

其中 $i = 1, 2, \cdots, n$。它们称为 Gershgorin 圆盘。那么矩阵 A 的本征值一定落在这些圆盘的并集之中

$$\sigma(A) \subseteqq S_R = \cup_{i=1}^{n} R_i \tag{66}$$

这个称为 Gershgorin 圆盘定理。这个定理对于实对称矩阵 (或者厄米矩阵) 特别有效，因为它们的本征值一定都位于实轴上。

根据这个圆盘定理，我们发现，对于实对称矩阵来说，我们发现搜寻其实根区间的下限 α 和上限 β 可以分别取为,

$$\alpha = \min_{1\le i\le n}[d_i - (|b_{i-1}| + |b_i|)], \quad \beta = \max_{1\le i\le n}[d_i + (|b_{i-1}| + |b_i|)] \tag{67}$$

其中我们约定了 $b_0 = b_n = 0$。

这个算法的设计如下：我们假定 $a^{(0)} = \alpha$，$b^{(0)} = \beta$，然后取 $c^{(0)}$ 为中间值 $c^{(0)} = (\alpha + \beta)/2$，然后把 $c^{(0)}$ 代入 Sturm 序列，计算符号变化的次数 $s(c^{(0)})$。如果我们希望计算第 i 个本征值 λ_i，那么我们将 $s(c^{(0)})$ 和 $n - i$ 比较，如果 $s(c^{(0)}) > n - i$，就意味着 $c^{(0)} > \lambda_i$，这个时候，我们让 $b^{(1)} = c^{(0)}$，减少搜索区间的上限；反之，则让 $a^{(1)} = c^{(0)}$。经过 r 次迭代以后，$c^{(r)} = \frac{a^{(r)}+b^{(r)}}{2}$ 就给出了 λ_i 的近似值，与真实值相差不超过 $(|\alpha| + |\beta|) \cdot 2^{-(r+1)}$。

6.4 稀疏矩阵的本征值问题：Lanczos 方法

如果我们希望计算的矩阵是一个庞大的稀疏矩阵，那么前面所列举的方法并不能有效地进行操作。这时，如果我们仅仅需要求解矩阵的部分而不是全部本征对，那么我们可以利用 Lanczos 方法来进行计算。

我们假设矩阵 $A \in R^{n\times n}$ 是对称的实稀疏矩阵。我们给它的本征值排一下序

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \tag{68}$$

当 n 很大的时候，我们用 Lanczos 方法去近似得到它的最大和最小本征值。

它的基本思路是这样的，我们可以构造 Rayleigh 比值

$$r(x) = \frac{x^T A x}{x^T x}, \quad \text{for } x \in R^n \tag{69}$$

对于矩阵 A 的最大和最小本征值，我们有

$$\lambda_1(A) = \max_{x\in R^n, x\neq 0} r(x), \quad \lambda_n(A) = \min_{x\in R^n, x\neq 0} r(x) \tag{70}$$

也就是说，我们要搜索适当的 x，使得 $r(x)$ 取得全局最大值和全局最小值。我们首先想到的是搜索方向沿着梯度 $\nabla r(x)$ 来得到最大值，和 $-\nabla r(x)$ 来得到最小值。之前我们在共轭梯度法的章节里面曾经讲过，这样的搜索其实并不是最佳搜索，我们应该采用的是共轭梯度法，在 Krylov 子空间里面进行搜索，是更优化的搜索方式，随着 Krylov 子空间维度的增加，我们会越来越接近我们想要的全局最大和最小值。

假设我们从任意矢量 v 出发，考虑矢量空间:

$$K_m(A; v) = \text{span}\{v, Av, \cdots, A^{m-1}v\}, \tag{71}$$

我们称之为矢量 v 和矩阵 A 的 m 阶的 Krylov 子空间。

对于一个 $n\times n$ 阶的稀疏矩阵 A，由于它的非零矩阵元只有 $O(n)$ 个，因此对于给定的矢量 v，计算矢量 Av 仅仅需要 $O(n)$ 的浮点数计算量而不是稠密矩阵的 $O(n^2)$。同时储存 Av 也仅仅需要 $O(n)$ 的内存。这就是为什么我们对于大型稀疏矩阵需要利用迭代的方法，以及为何我们对它的 Krylov 子空间感兴趣。我们一般需要寻找最大和最小本征值的近似解，因此在这类计算中我们一般假设 $m\ll n$。

具体构造 Krylov 子空间的方法，就是所谓的 Lanczos 迭代。算法如下

$$
\begin{aligned}
&v_0=0,\quad v_1=v,\quad \beta_1=0\\
&\text{For } k=1,2,\cdots,m-1,\ \text{do}\\
&\qquad \omega_k=Av_k\\
&\qquad \alpha_k=\omega_k^T v_k\quad \text{计算内积}\\
&\qquad \omega_k=\omega_k-\alpha_k v_k-\beta_k v_{k-1}\quad \text{更新 } \omega_k\\
&\qquad \beta_{k+1}=||\omega_k||_2,\quad v_{k+1}=\frac{\omega_k}{||\omega_k||_2}\quad \text{归一化 } \omega_k\\
&\text{End}
\end{aligned}
\tag{72}
$$

我们不难看出，如果 $v_1,\cdots,v_k$ 张成了 k 阶 Krylov 子空间，那么再加上 v_{k+1}，就张成了 $k+1$ 阶 Krylov 子空间，另外，如果有 $v_k^T v_{k-1}=0$，我们马上能得到 $v_{k+1}^T v_k=0$，因此经过上述的 Lanczos 迭代，我们获得了一组 $K_m(A;v)$ 中的正交归一的基矢，它们构成了一个长方正交矩阵 (因为一般来说 $m\ll n$):

$$
V_m=(v_1,v_2,\cdots,v_m) \tag{73}
$$

我们还可以证明 $H_m=V_m^T A V_m$ 是个 $m\times m$ 阶三对角矩阵。这是因为

$$
(H_m)_{\alpha\beta}=v_\alpha^T A v_\beta \tag{74}
$$

如果 $\alpha>\beta+1$，那么 Av_β 由 $\beta+1$ 阶的 Krylov 子空间中的基矢线性组合构成，根据 v_k 矢量的构造方式，v_α 肯定与 Av_β 正交，因此当 $\alpha>\beta+1$ 时，我们有 $(H_m)_{\alpha\beta}=0$。反之亦然。

由于 H_m 是一个比原矩阵 A 小很多的矩阵，因此处理它的本征值问题要简单的多。而且由于 H_m 的三对角形式，我们可以运用前一小节提及的 Sturm 序列求解。需要注意的是，由于舍入误差的传递，一般的 Lanczos 算法当 m 变得比较大的时候往往会遇到稳定性的问题。也就是说，原先做好的正交归一化步骤有可能由于舍入误差的影响而不再成立。这时可能在需要重新进行 Gram-Schmidt 的操作。

尽管存在一定的局限性，Lanczos 方法有两个显著的优点：一是它在变换的过程中保证了一直保持了稀疏矩阵的形式不变，这对于大型矩阵 n 很大的时候，尤为重要；另外，它能够比较快地趋于矩阵 A 本征值的极值。我们之前提到过可以用这个方法来计算最小的若干个本征值和本征矢量，它能够帮助我们加速 CG 算法解线性方程组。