APR 29TH, 2014

LAS 6-Matrix Decomposition

线性代数那些事 Things of Linear Algebra
逸夫图书馆, 2014/5/22

矩阵分解

[内容摘录自我去年写的Numeric Methods with Matlab总结的第一章 线性方程组求解和第三章 矩阵特征值和奇异值求解，进入上面的链接地址可以下载原始的PDF版本的内容，如果不想下载，那么也可以直接阅读下面的内容，源代码基本上都是使用或者修改自书籍《Numerical Computing with Matlab》，如果下面有不熟悉的术语也请阅读这本书或者喻文健老师的翻译版本《Matlab数值计算》]

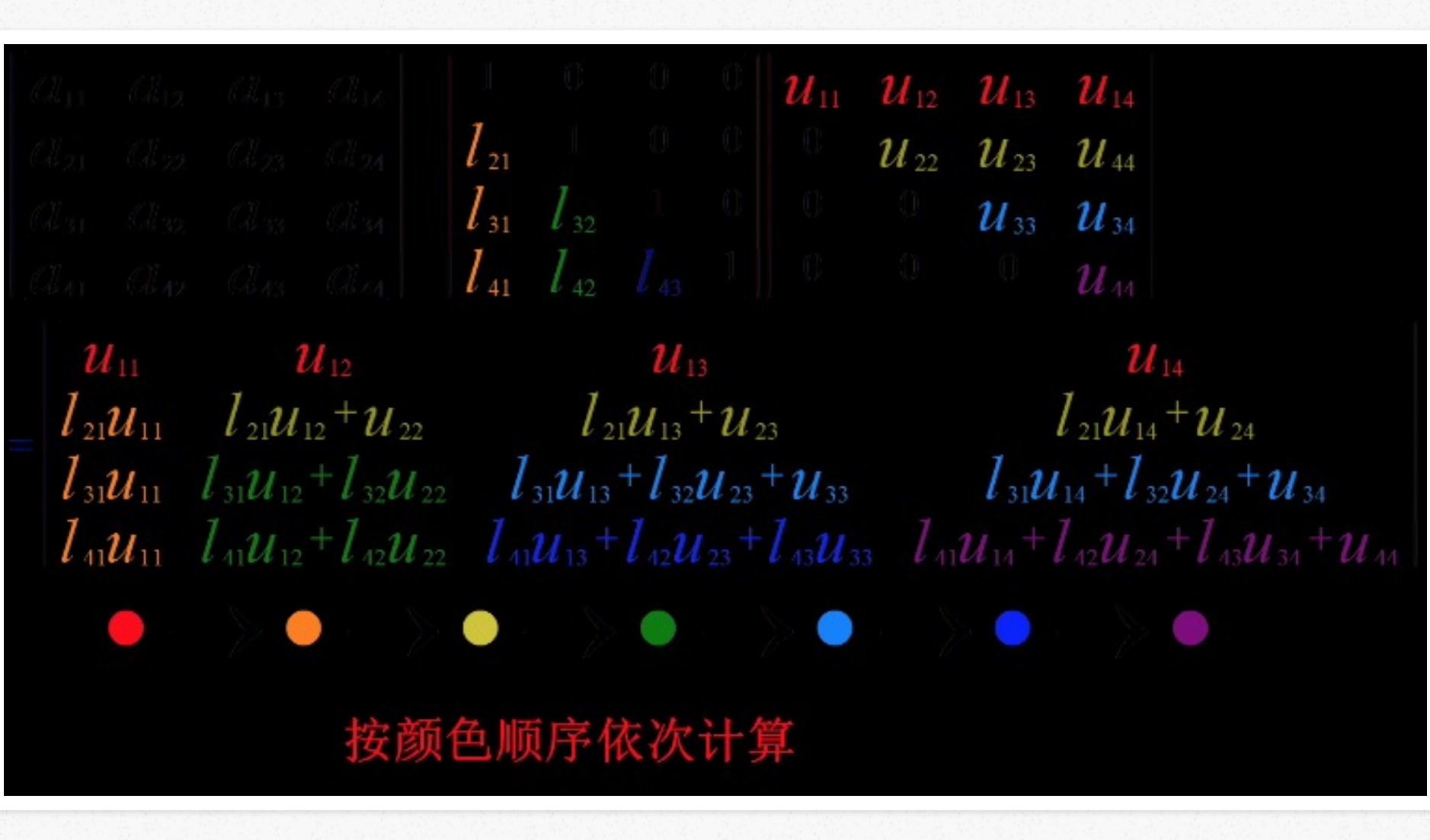
总结矩阵分解：由于实际中遇到的矩阵可能规模很大而且很复杂，为了更高效地处理矩阵于是就有了矩阵分解。分解一般是分解成规模更小的或者性质更好的矩阵(比如三角矩阵或者对角矩阵)，这个时候我们需要透过现象看本质，结合前面我们解释矩阵就是线性变换可知，矩阵分解实际上是把这个线性变换转换成几个其他的线性变换的组合，一般是旋转变换(对应一个正交矩阵)或者放缩变换(对应一个对角矩阵)的组合。

1.LU分解

线性方程组的实际应用中经常遇到系数矩阵不变，只是右端项发生变化的情况(多右端项问题)，这个时候如果还是使用高斯消去法的话，对于每个右端项都要进行重复的消去和回代的过程，显然计算量很大。为了解决这类问题，于是就有了LU分解算法。

关系式$LU = PA$即为矩阵A的LU分解(或者三角分解)，其中矩阵P为排列阵(单位阵经过行列交换得到的矩阵，有时候也叫置换阵)，矩阵L为单位下三角矩阵(对角线元素都为1)，矩阵U为上三角矩阵，这样的话，一般的线性方程组$Ax = b$可以等价为两个三角型线性方程组 $Ly = Pb$, $Ux = y$。

首先对系数矩阵A进行LU分解，然后对于每个右端项只要先计算出y然后再计算x即可，两个都只是解很简单的三角型线性方程组。计算过程如下，图片来源



LU分解有两种方法，一种是使用前面的高斯消去法，另一种是Crout方法(这里不介绍，详情请看参考《Numerical Methods: An introduction with Applications Using Matlab》)。

下面给出使用高斯消去法的LU分解算法源码[大部分内容和高斯消去法相同，只是它还要计算矩阵L, U, P]，它是LU分解常用的K-I-J版本。在Matlab中可以使用内置`lu`函数，调用示例：$[L, U, P] = lu(A)$

```
function [L,U,p] = lugs(A)
% 使用高斯消去法的LU分解 [ k-i-j 版本 ]
% 系数矩阵，单位下三角矩阵，上三角矩阵，排列阵
% A=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
% b=[12; -6.5; 16; 17];
% [L,U,P]=lugs(A)
% A(P,:)
% L*U

[n,n] = size(A);
p = (1:n)';
for k = 1:n-1
    %查找主元列对角线以下的绝对值最大的元素和索引
    [r,m] = max(abs(A(k:n,k)));
    m = m+k-1;
    %如果对应元素是0则跳过消去过程
    if (A(m,k) ~= 0)
        %满足条件的话就交换行
        if (m ~= k)
            A([k m],:) = A([m k],:);
            p([k m]) = p([m k]);%对应的排列阵也要跟着变化
        end
        %计算乘子
        i = k+1:n;
        A(i,k) = A(i,k)/A(k,k);
        %更新矩阵的行
        j = k+1:n;
        A(i,j) = A(i,j) - A(i,k)*A(k,j);%就地存储!
    end
end
%得到LU分解的结果
L = tril(A,-1) + eye(n,n);%下三角部分，但是对角线上都是1
U = triu(A);%上三角部分
```

wiki上关于LU分解的应用


求解线性方程 [编辑]

对于给定的线性方程组

$Ax = LUx = b$

要解出x，可以进行一下步骤

1. 首先，解方程$Ly = b$得到y;
2. 然后解方程$Ux = y$得到x。

在两次的求解中，我们遇到的都是三角矩阵，因此运用向前（向后）替代法就可以简洁地求解（参见三角矩阵），而不需要用到高斯消元法。然而，在将A进行LU分解时，仍然要用到高斯消元法。因此，这个方法适合在要对许多个不同的b求解时用。

求逆矩阵 [编辑]

求矩阵A的逆时，可以直接求L和U的逆矩阵，然后代入：$A^{-1}=U^{-1}L^{-1}$。也可以将单位矩阵分解成n个列向量，然后用上面求解线性方程的方法解出逆矩阵的列向量，然后拼起来。后者的复杂度在n^3级别，较高斯法为优。

计算行列式 [编辑]

矩阵L和U可以用来快速地计算矩阵A的行列式，因为det(A) = det(L) det(U)，而三角矩阵的行列式就是对角线元素的乘积。如果要求L是单位三角矩阵，那么

$\det(A)=\det(L)\det(U)=\prod_{i=1}^{n} u_{ii}$.

同样的方法也可以应用于LUP分解，只需乘上P的行列式，即相应置换的符号差。

2.Cholesky分解

对于对称矩阵A，如果它的各阶顺序主子式$\neq 0$，则它可以唯一分解为$A = LDL^T$，其中D是对角阵，L为单位下三角阵。若矩阵同时正定，那么存在实的非奇异的下三角矩阵L，满足$A = LL^T$，若限定L对角线元素> 0，那么此分解唯一。

考虑对称正定矩阵A，它的LU分解过程可以进一步简化，这也就是Cholesky分解！

事实证明，**对于对称正定矩阵，Cholesky分解是稳定的，不需要进行选主元操作。**另外，它的计算量和存储量都只是LU分解的一半。考虑到它的对称性，LU分解的结果如下：(注意矩阵L不再是单位下三角矩阵了)

$$A = \begin{bmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} & \dots & l_{1n} \\ 0 & l_{22} & l_{23} & \dots & l_{2n} \\ 0 & 0 & l_{33} & \dots & l_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & l_{nn} \end{bmatrix}$$

一种就地存储的Cholesky分解算法的实现步骤(对$j = 2, 3 \cdots, n$重复执行(3)(4)步即可)：

(1)求l_{11}。$l_{11} = \sqrt{a_{11}}$

(2)求l_{i1}。$l_{i1} = a_{i1}/l_{11}$

(3)假设矩阵L的前$j-1$列都已经求出了，即$l_{ik}(k \le j-1, i = 1, \cdots, n)$都已知，考虑计算$a_{j,j}$。

$$a_{jj} = \sum_{k=1}^{j} l_{ik}^2 \quad => \quad l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}$$

(4)求矩阵L中第j列剩余元素，即$l_{ij}(i > j)$，考虑计算a_{ij}。

$$a_{ij} = \sum_{k=1}^{n} l_{ik} l_{jk} = \sum_{k=1}^{j} l_{ik} l_{jk} = \sum_{k=1}^{j-1} l_{ik} l_{jk} + l_{ij} l_{jj} \quad => \quad l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}$$

简易版本的cholesky分解算法代码，在Matlab中可以调用内置`chol`函数，示例：$[R, p] = chol(A)$

```
function [L,U] = cholnm(A)
% 简易版本的Cholesky分解，就地存储，使用下三角
% p=pascal(5)
% [l,u]=cholnm(p)
% l*u

[n,n] = size(A);
for j=1:n %当前操作的列
    for k=1:j-1
        A(j,j)=A(j,j)-A(j,k)^2;
    end
    A(j,j)=sqrt(A(j,j));%求出对角线元素
    for i=j+1:n %求解对角线以下的元素
        for k=1:j-1
            A(i,j)=A(i,j)-A(i,k)*A(j,k);
        end
        A(i,j)=A(i,j)/A(j,j);%求出对角线以下的元素
    end
end
%求出上三角和下三角
L=tril(A);
U=L';
```

小总结：有了上面的LU分解和Cholesky分解，我们就可以写出一个简易版本的矩阵求逆算法。

```
function x = bslashnm(A,b)
% 解线性方程组
% A=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
% b=[12; -6.5; 16; 17];
% A\b
% x=bslashnm(A,b)

[n,n] = size(A);
if isequal(triu(A,1),zeros(n,n))
    %下三角直接前向回代即可
    x = forward(A,b);
    return
elseif isequal(tril(A,-1),zeros(n,n))
    %上三角直接后向回代即可
    x = backsubs(A,b);
    return
elseif isequal(A,A')
    [R,fail] = chol(A);%进行Cholesky分解
    if ~fail
        %如果分解成功，求解两个三角型线性方程组即可
        y = forward(R',b);
        x = backsubs(R,y);
        return
    end
end

%三角分解
[L,U,p] = lugs(A);
%对右端项进行排列然后解下三角线性方程组
y = forward(L,b(p));
%解上三角线性方程组
x = backsubs(U,y);

% --------前向消去 解下三角型线性方程组--------
% For lower triangular L, x = forward(L,b) solves L*x = b.
function x = forward(L,x)
[n,n] = size(L);
x(1) = x(1)/L(1,1);
for k = 2:n
    j = 1:k-1;
    x(k) = (x(k) - L(k,j)*x(j))/L(k,k);
end

% --------回代 解上三角型线性方程组------------
% For upper triangular U, x = backsubs(U,b) solves U*x = b.
function x = backsubs(U,x)
[n,n] = size(U);
x(n) = x(n)/U(n,n);
for k = n-1:-1:1
    j = k+1:n;
    x(k) = (x(k) - U(k,j)*x(j))/U(k,k);
end
```

3.QR分解

QR分解是求解中小规模矩阵所有特征值的有效方法，它的基本原理是利用相似变换(两个方阵相似的条件是存在可逆矩阵C,使得$A = C^{-1}BC$)不改变矩阵的特征值以及上三角矩阵的特征值就是它的对角线元素的性质对矩阵进行分解。

QR分解的实际计算有很多方法，例如Givens旋转、Householder变换，以及Gram-Schmidt正交化等等。每一种方法都有其优点和不足。

实Schur型矩阵：矩阵S为上三角阵，且对角块为1阶或者2阶矩阵。这类矩阵可以进行正交分解：$S = Q^T AQ$，称为实Schur分解，其中S的1阶对角块就是A的实特征值，2阶对角块的特征值是A的共轭复特征值。等式$A = QSQ^T$称为矩阵A的实Schur分解。

QR分解的目标是通过一系列**正交相似变换**(也就是说不单要求变换矩阵P可逆，还要求它是正交阵)将原矩阵转换成一个上三角型的相似矩阵(即实Schur型矩阵)，这样求解特征值就特别简单了。QR分解可以得到所有的特征值，但是不能得到特征向量。**如果所有的特征值都是实数的话，QR分解得到的是一个正交阵和一个上三角阵。如果有的特征值是复数的话，那么得到的是一个正交阵和一个对角线某些位置为(2x2)块矩阵的上三角阵。**

假设矩阵A_1是要求特征值的矩阵，QR分解求其特征值的迭代过程如下：

(1)对A_1进行QR分解得到$A_1 = Q_1 R_1$。其中Q_1是一个正交阵，R_1是一个上三角阵。

(2)用R_1乘以Q_1得到A_2，即$A_2 = R_1 Q_1$。由于$R_1 = Q_1^{-1} A_1$，则有$A_2 = Q_1^{-1} A_1 Q_1$，所以A_2和A_1是相似矩阵。

(3)此时，对A_2进行QR分解得到$A_2 = Q_2 R_2$，再将得到的Q_2和R_2反过来相乘，得到A_3，然后再对A_3进行QR分解，不断重复上面的操作直到得到上三角矩阵，最终的上三角矩阵对角线元素便是矩阵特征值。

上面的迭代过程中，每步都要进行一次QR分解，这是通过上节介绍的Householder变换实现的。

简易版本的QR分解的实现，Matlab中可以使用内置函数`qr`进行QR分解。

```
function [Q,R] = easyqrnm(R)
% 简易版本的QR分解
% A=[45 30 -25; 30 -24 68; -25 68 80];
% for i=1:40
%     [q,r]=easyqrnm(A);
%     A=r*q;
% end
% e=diag(A)
% eig(A)


n=size(R);
I=eye(n);
Q=I;
for j=1:n-1
    c=R(:,j);
    c(1:j-1)=0;
    e(1:n,1)=0;%
    if c(j) > 0
        e(j)=1;
    else
        e(j)=-1;
    end
    clen=sqrt(c'*c);
    v=c+clen*e;
    H=I-2/(v'*v) * v*v';%得到矩阵H
    Q=Q*H;%
    R=H*R;%
end
```

实际使用的QR分解要比上面的简易版本的QR分解复杂很多，但是基本思路是一样的，那就是通过一系列的正交相似变换$B = Q^T AQ$将矩阵A化为上三角或者对角块阶数很小的分块上三角阵，然后求特征值。实用的技术主要有两个：(1)将矩阵化简为上Hessenberg矩阵；(2)带原点位移的QR算法。

上Hessenberg矩阵和上三角矩阵的区别在于，其紧邻主对角线下方的副对角线上的元素不全为0。可以证明，**对上Hessenberg矩阵A_k进行QR算法的一步迭代，得到的A_{k+1}仍为上Hessenberg矩阵。**使用Householder变换可以将一般矩阵化为上Hessenberg矩阵，总共经过(n-2)步正交相似变换。**若原始矩阵是实对称阵，那么得到的上Hessenberg矩阵是一个对称的三对角阵。**对上Hessenberg矩阵进行QR分解时不需要使用Householder变换，只要使用一系列Givens旋转变换即可。Givens旋转变换是一类平面旋转变换，它能够消去给定向量的某一个分量(使其为0)，这不同于Householder变换消去向量中的多个分量。在处理稀疏向量或者稀疏矩阵时，Givens旋转更加高效。

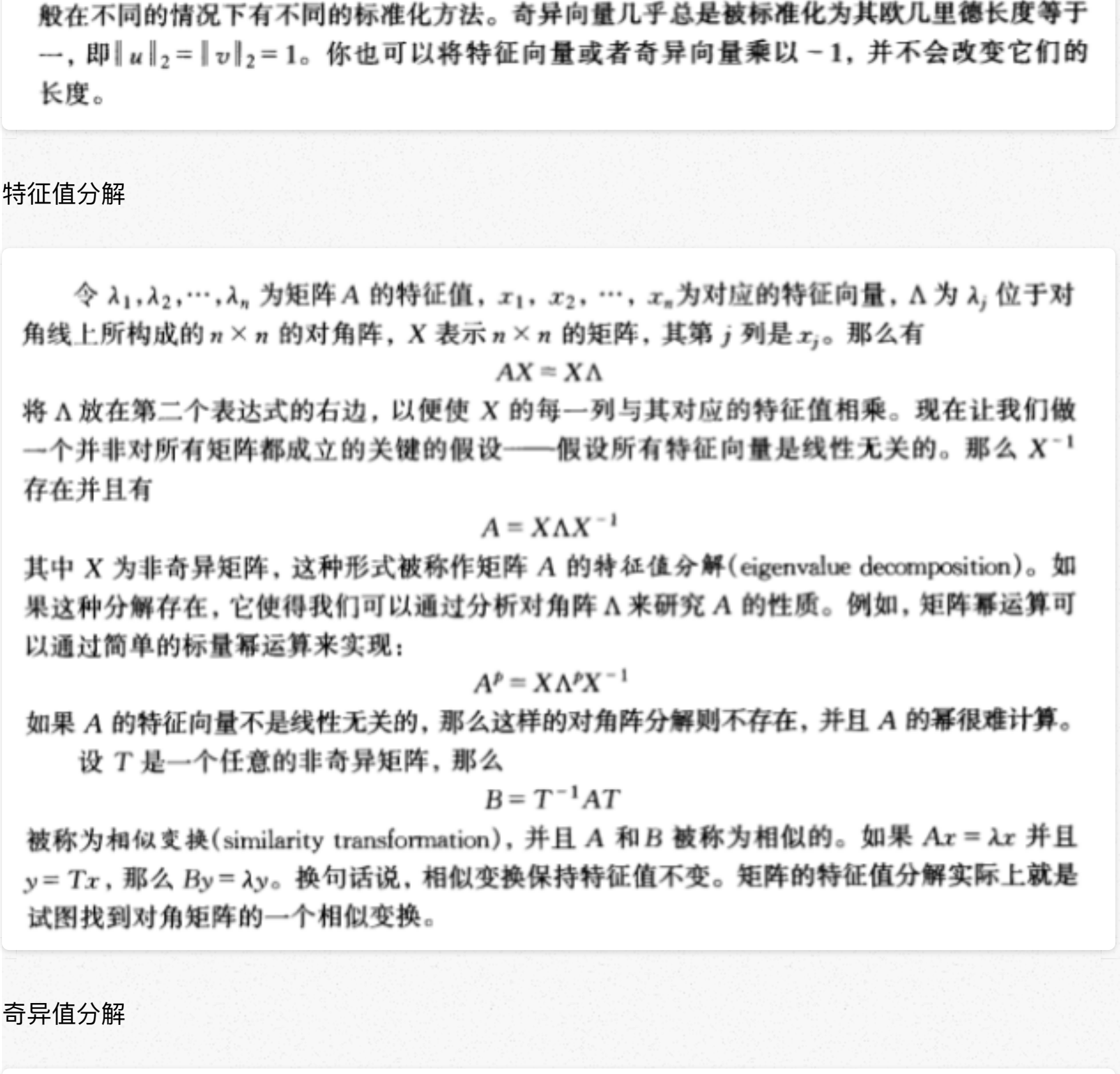
将原点位移技术与QR分解算法结合，通过原点位移可以改变做QR分解的矩阵，一方面它能够提高迭代收敛速度，另一方面也使QR分解对更一般的矩阵有效。

该部分的内容比较难，此处不过多介绍，详细内容参考书籍数值分析与算法,喻文健,清华大学出版社,2012,1

4.特征值分解和奇异值分解

关于特征值和奇异值及其分解可以参考书籍《Numerical Computing with Matlab》，下面附上几张重要内容截图：

特征值和奇异值


10.1 特征值与奇异值分解

方阵 A 的一组特征值(eigenvalue)和特征向量(eigenvector)是一个标量 λ 和一个非零向量 x，它们满足

$$Ax = \lambda x$$

矩阵 A 的一组奇异值(singular value)和奇异向量对(singular vector)是一个非负标量 σ 以及两个非零向量 u 和 v，它们满足

$$Av = \sigma u$$
$$A^H u = \sigma v$$

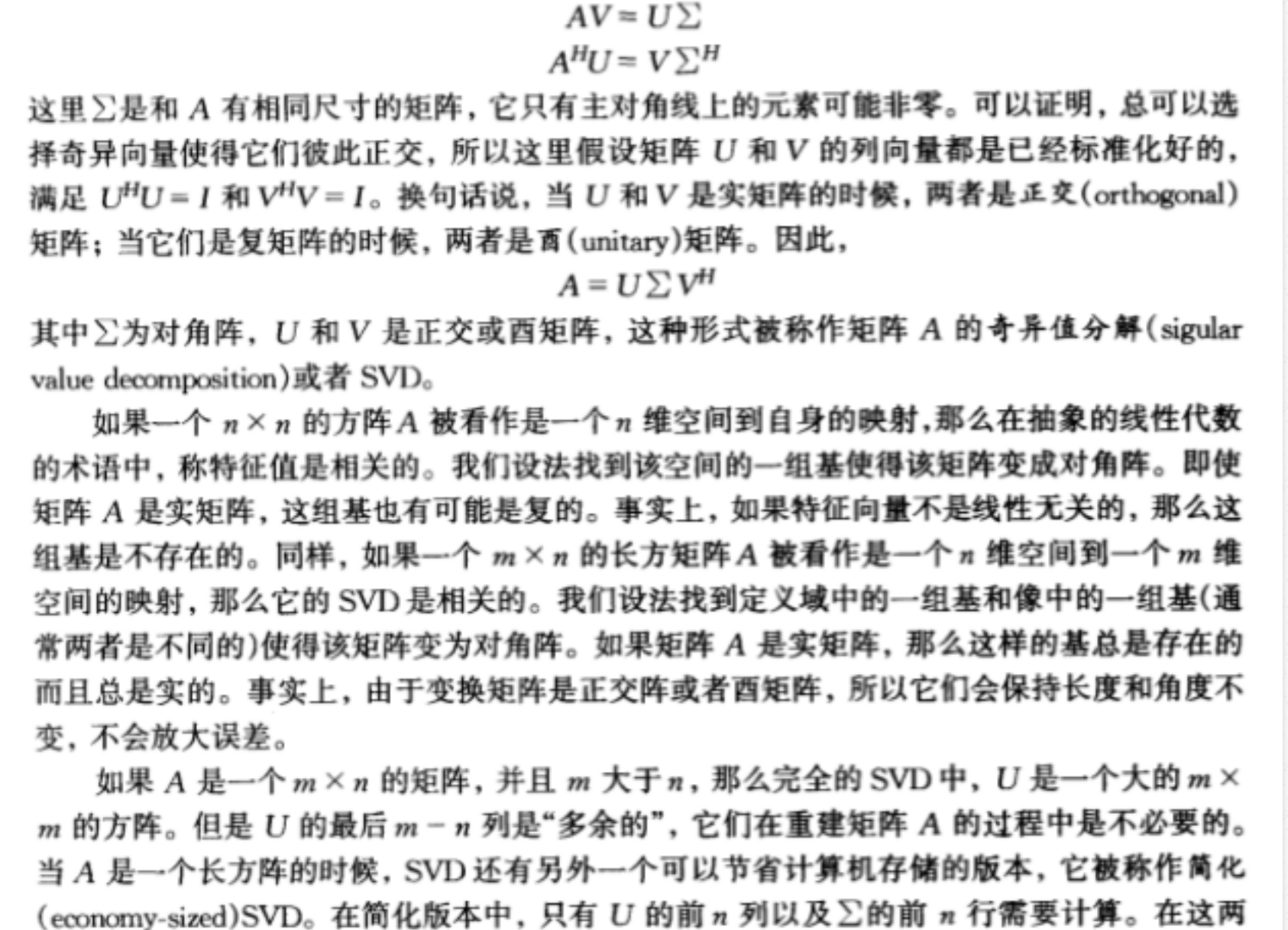
其中，A^H的上角标代表矩阵的厄密特转置(Hermitian transpose)，它表示对一个复矩阵的共轭转置。如果矩阵是实矩阵，那么它的 A^T 和 A^H 相同。在 MATLAB 中，矩阵的转置都用 A' 表示。

英文术语"eigenvalue"是德文"eigenvert"的局部翻译。它的完整翻译应该是"own value"(本征值)或者"characteristic value"(特征值)，但是这些说法并不常用。术语"singular value"与一个矩阵的奇异程度有关。

如果矩阵是由某个向量空间到自身的变换，那么在这种情况下，特征值起着重要的作用。线性常微分方程系统就是一个基本的例子。λ 的值可以对应于振动的频率，稳定性参数的临界值，或者原子的能量等级。如果矩阵是某个向量空间到另一个空间(很可能与原空间维数不同)的变换，那么在这种情况下，奇异值起着重要的作用。超定或者欠定代数方程组是一个基本的例子。

特征向量和奇异值向量的定义并没有规定它们的标准化。一个特征向量 x 或者一对奇异向量 u 和 v，可以通过任何非零因子换算而不改变其他的重要性质。对称矩阵的特征向量，一般被标准化为其欧几里德长度等于一，即$\|x\|_2 = 1$。另一方面，非对称矩阵的特征向量，一般在不同的情况下有不同的标准化方法。奇异向量几乎总是被标准化为其欧几里德长度等于一，即$\|u\|_2 = \|v\|_2 = 1$。你也可以将特征向量或者奇异向量乘以 -1，并不会改变它们的长度。

特征值分解


令 $\lambda_1, \lambda_2, \cdots, \lambda_n$ 为矩阵 A 的特征值，$x_1, x_2, \cdots, x_n$ 为对应的特征向量，Λ 为 λ_j 位于对角线上所构成的 $n \times n$ 的对角阵，X 表示 $n \times n$ 的矩阵，其第 j 列是 x_j。那么有

$$AX = X\Lambda$$

将 Λ 放在第二个表达式的右边，以便使 X 的每一列与其对应的特征值相乘。现在让我们做一个并非对所有矩阵都成立的关键的假设——假设所有特征向量是线性无关的。那么 X^{-1} 存在并且有

$$A = X\Lambda X^{-1}$$

其中 X 为非奇异矩阵，这种形式被称作矩阵 A 的特征值分解(eigenvalue decomposition)。如果这种分解存在，它使得我们可以通过分析对角阵 Λ 来研究 A 的性质。例如，矩阵幂运算可以通过简单的标量幂运算来实现：

$$A^p = X\Lambda^p X^{-1}$$

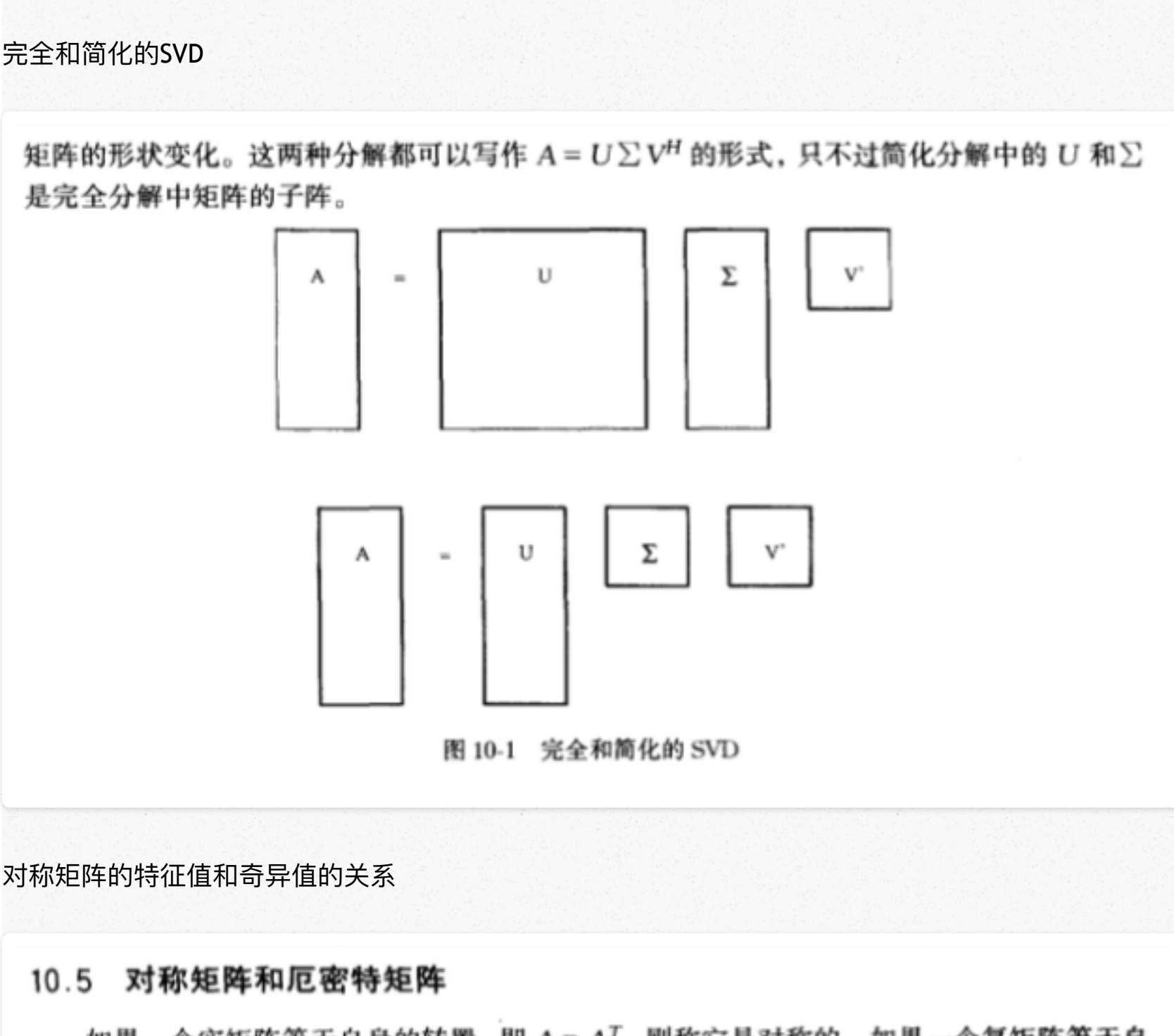
如果 A 的特征向量不是线性无关的，那么这样的对角分解则不存在，并且 A 的幂很难计算。

设 T 是一个任意的非奇异矩阵，那么

$$B = T^{-1}AT$$

被称为相似变换(similarity transformation)，并且 A 和 B 被称为相似的。如果 $Ax = \lambda x$ 并且 $y = Tx$，那么 $By = \lambda y$。换句话说，相似变换保持特征值不变。矩阵的特征值分解实际上就是试图找到对角矩阵的一个相似变换。

奇异值分解


把奇异值和奇异向量的定义方程写成矩阵的形式，如下

$$AV = U\Sigma$$
$$A^H U = V\Sigma^H$$

这里Σ是和 A 有相同尺寸的矩阵，它只有主对角线上的元素可能非零。可以证明，总可以选择奇异向量使得它们彼此正交，所以这里假设矩阵 U 和 V 的列向量都是已经标准化好的，满足 $U^H U = I$ 和 $V^H V = I$。换句话说，当 U 和 V 是实矩阵的时候，两者是正交(orthogonal)矩阵；当它们是复矩阵的时候，两者是酉(unitary)矩阵。因此，

$$A = U\Sigma V^H$$

其中Σ为对角阵，U 和 V 是正交或酉矩阵，这种形式被称作矩阵 A 的奇异值分解(sigular value decomposition)或者 SVD。

如果一个 $n \times n$ 的方阵 A 被看作是一个 n 维空间到自身的映射，那么在抽象的线性代数的术语中，称特征值是相关的。我们设法找到该空间的一组基使得该矩阵变成对角阵。即使矩阵 A 是实矩阵，这组基也有可能是复的。事实上，如果特征向量不是线性无关的，那么这组基是不存在的。同样，如果一个 $m \times n$ 的长方矩阵 A 被看作是一个 n 维空间到一个 m 维空间的映射，那么它的 SVD 是相关的。我们设法找到定义域中的一组基和像中的一组基(通常两者是不同的)使得该矩阵变为对角阵。如果矩阵 A 是实矩阵，那么这样的基总是存在的而且总是实的。事实上，由于变换矩阵是正交阵或者酉矩阵，所以它们会保持长度和角度不变，不会放大误差。

如果 A 是一个 $m \times n$ 的矩阵，并且 m 大于 n，那么完全的 SVD 中，U 是一个大的 $m \times m$ 的方阵。但是 U 的最后 $m-n$ 列是"多余的"，它们在重建矩阵 A 的过程中是不必要的。当 A 是一个长方阵的时候，SVD 还有另外一个可以节省计算机存储的版本，它被称作简化(economy-sized)SVD。在简化版本中，只有 U 的前 n 列以及Σ的前 n 行需要计算。在这两种分解中，矩阵 V 都是同样的 $n \times n$ 的矩阵，没有变化。图 10-1 显示了两种 SVD 版本中，

完全和简化的SVD

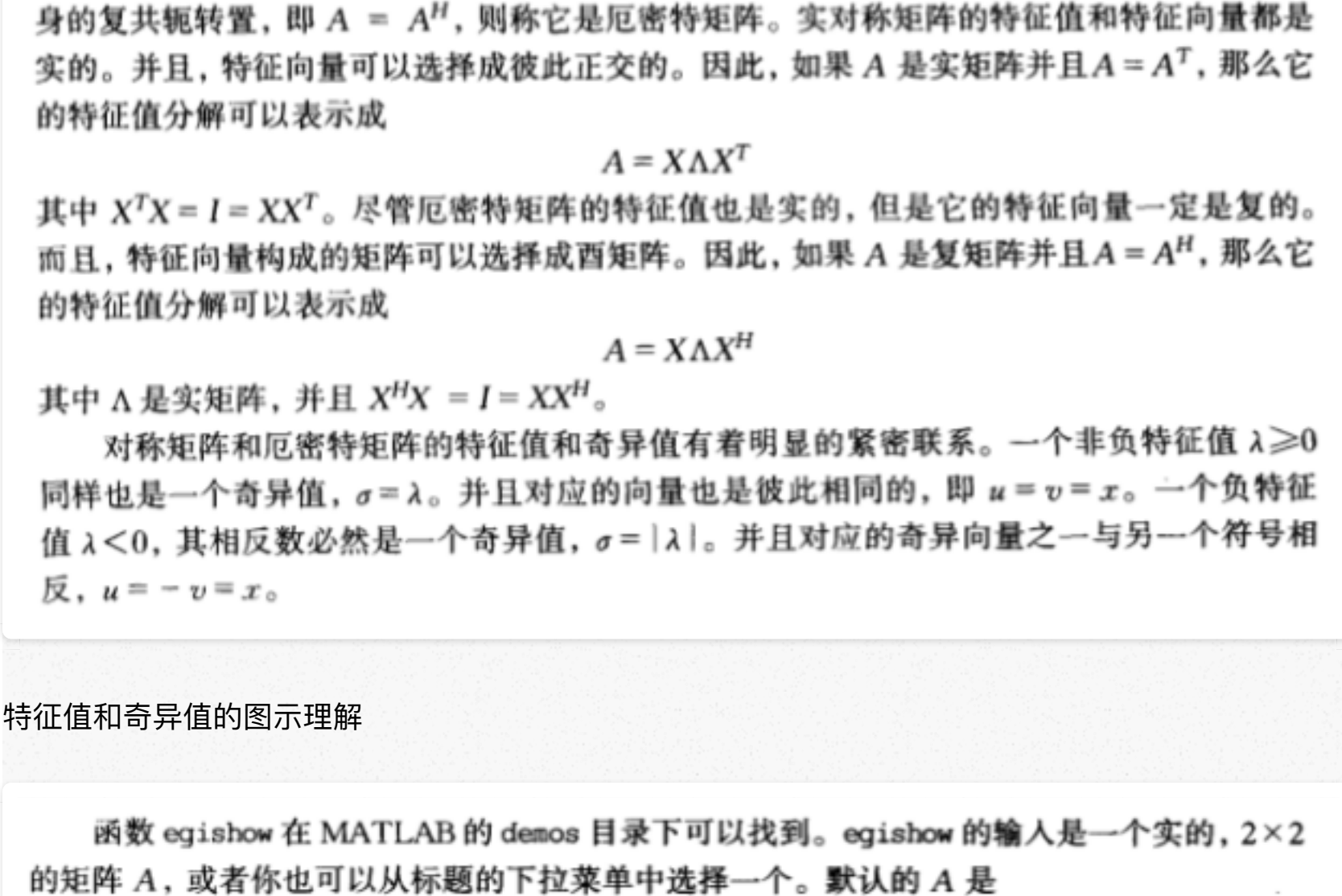
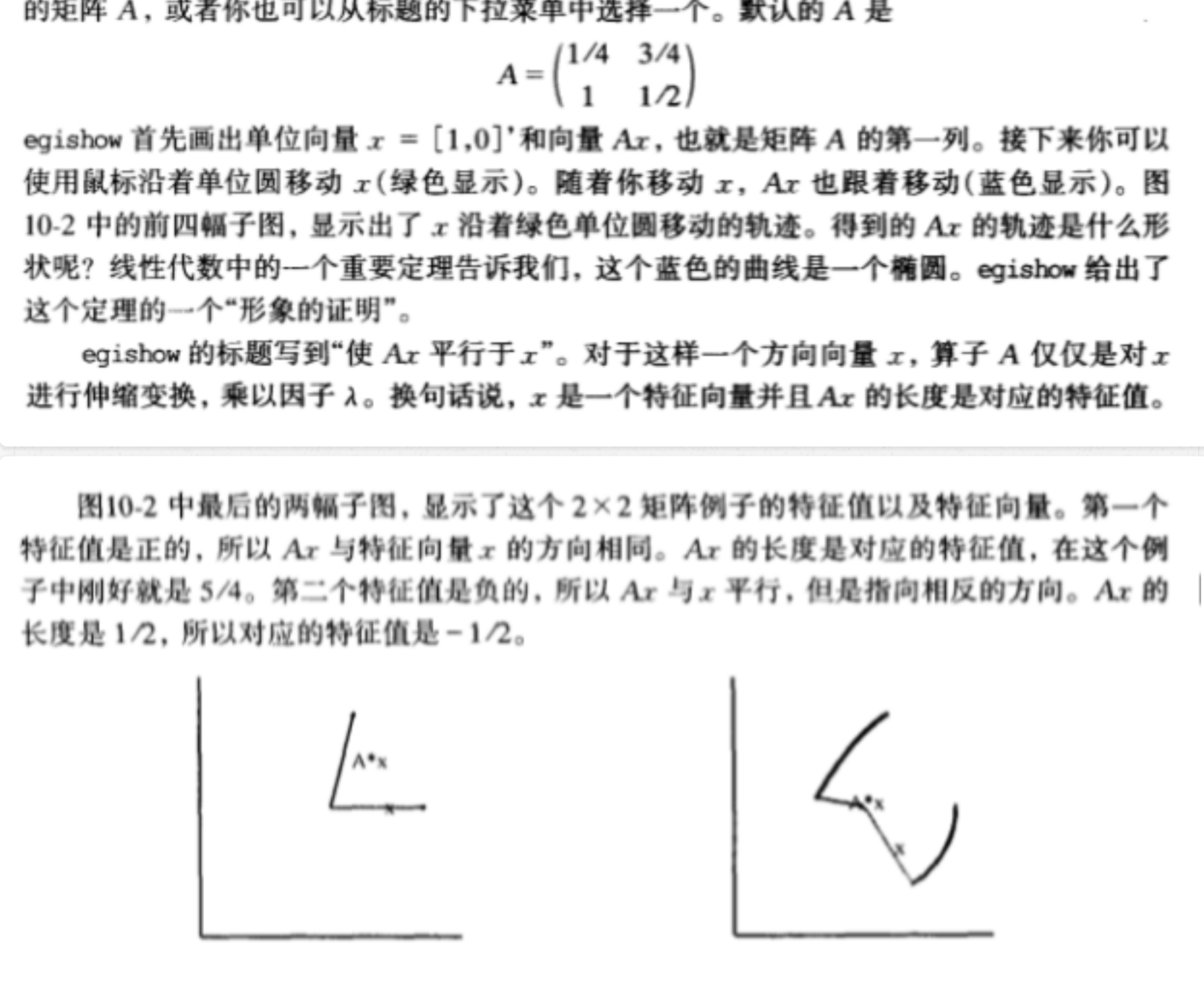

矩阵的形状变化。这两种分解都可以写作 $A = U\Sigma V^H$ 的形式，只不过简化分解中的 U 和Σ是完全分解中矩阵的子阵。

图 10-1 完全和简化的 SVD

对称矩阵的特征值和奇异值的关系


10.5 对称矩阵和厄密特矩阵

如果一个实矩阵等于自身的转置，即 $A = A^T$，则称它是对称的。如果一个复矩阵等于自身的复共轭转置，即 $A = A^H$，则称它是厄密特矩阵。实对称矩阵的特征值和特征向量都是实的。并且，特征向量可以选择成彼此正交的。因此，如果 A 是实矩阵并且$A = A^T$，那么它的特征值分解可以表示成

$$A = X\Lambda X^T$$

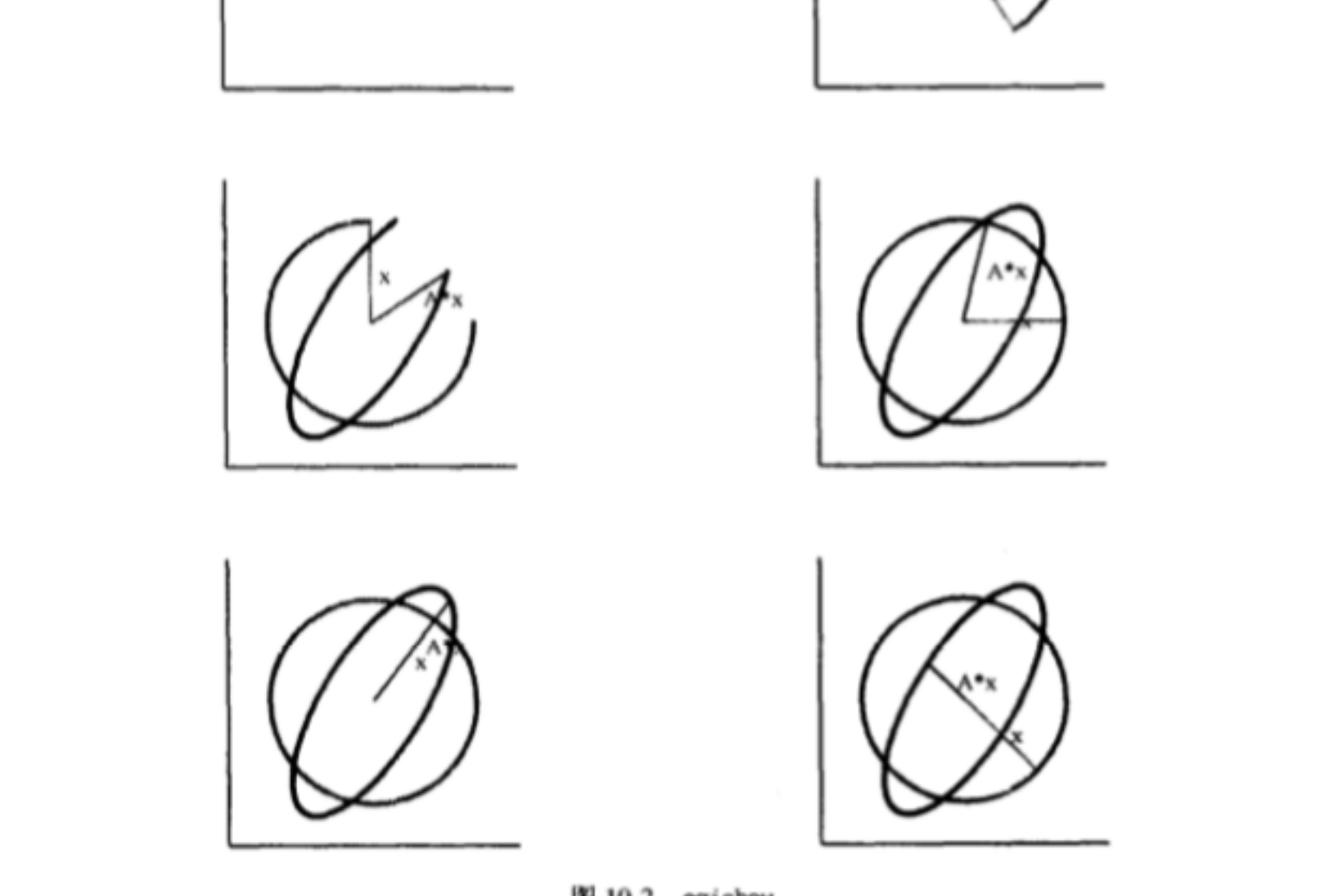
其中 $X^T X = I = XX^T$。尽管厄密特矩阵的特征值也是实的，但是它的特征向量一定是复的。而且，特征向量构成的矩阵可以选择成酉矩阵。因此，如果 A 是复矩阵并且$A = A^H$，那么它的特征值分解可以表示成

$$A = X\Lambda X^H$$

其中 Λ 是实矩阵，并且 $X^H X = I = XX^H$。

对称矩阵和厄密特矩阵的特征值和奇异值有着明显的紧密联系。一个非负特征值 $\lambda \geqslant 0$ 同样也是一个奇异值，$\sigma = \lambda$。并且对应的向量也是彼此相同的，即 $u = v = x$。一个负特征值 $\lambda < 0$，其相反数必然是一个奇异值，$\sigma = |\lambda|$。并且对应的奇异向量之一与另一个符号相反，$u = -v = x$。

特征值和奇异值的图示理解


函数 egishow 在 MATLAB 的 demos 目录下可以找到。egishow 的输入是一个实的，2×2 的矩阵 A，或者你也可以从标题的下拉菜单中选择一个。默认的 A 是

$$A = \begin{pmatrix} 1/4 & 3/4 \\ 1 & 1/2 \end{pmatrix}$$

egishow 首先画出单位向量 $x = [1,0]'$和向量 Ax，也就是矩阵 A 的第一列。接下来你可以使用鼠标沿着单位圆移动 x(绿色显示)。随着你移动 x，Ax 也跟着移动(蓝色显示)。图 10-2 中的前四幅子图，显示出了 x 沿着绿色单位圆移动的轨迹。得到的 Ax 的轨迹是什么形状呢？线性代数中的一个重要定理告诉我们，这个蓝色的曲线是一个椭圆。egishow 给出了这个定理的一个"形象的证明"。

egishow 的标题写到"使 Ax 平行于 x"。对于这样一个方向向量 x，算子 A 仅仅是对 x 进行伸缩变换，乘以因子 λ。换句话说，x 是一个特征向量并且 Ax 的长度是对应的特征值。

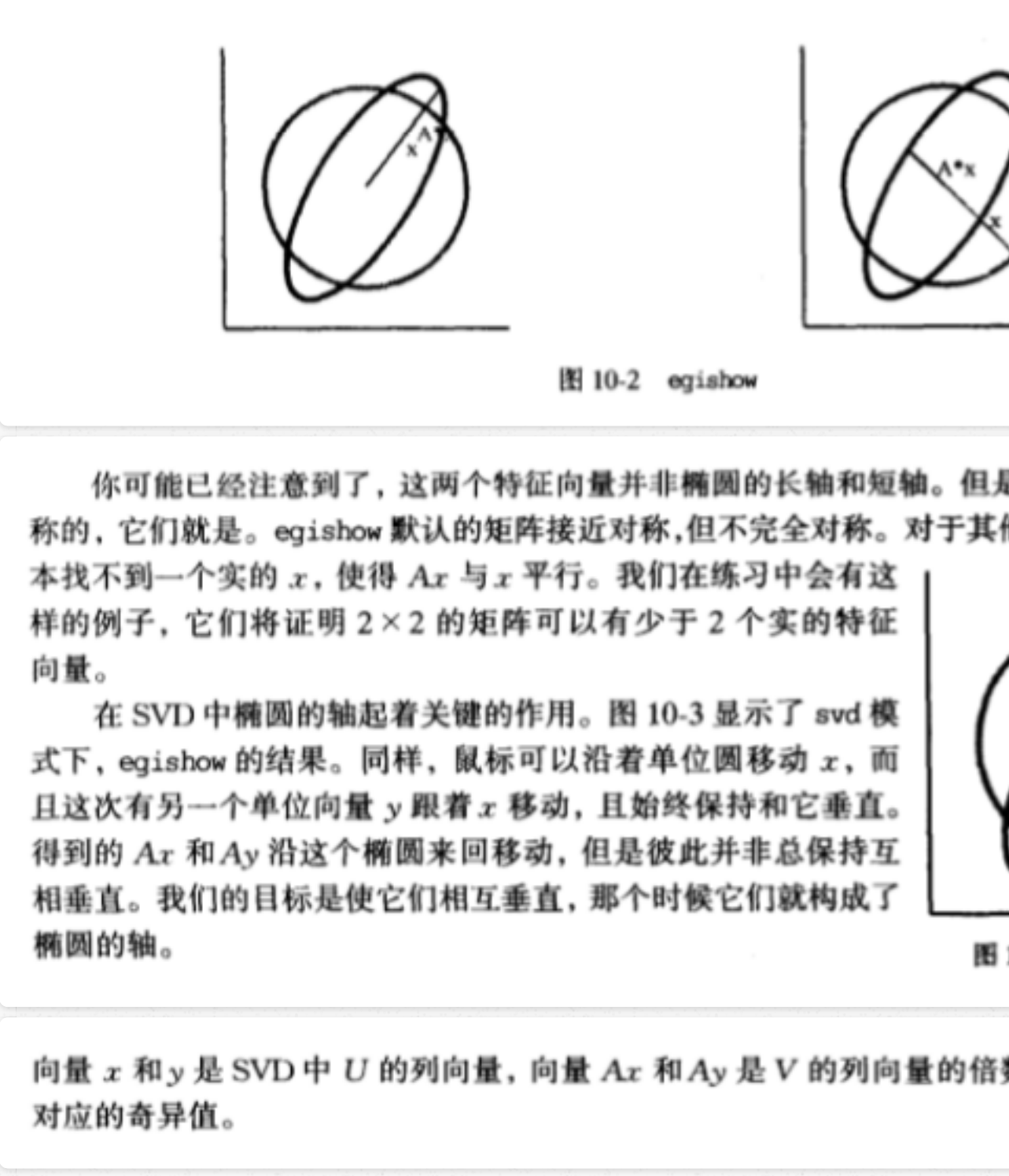
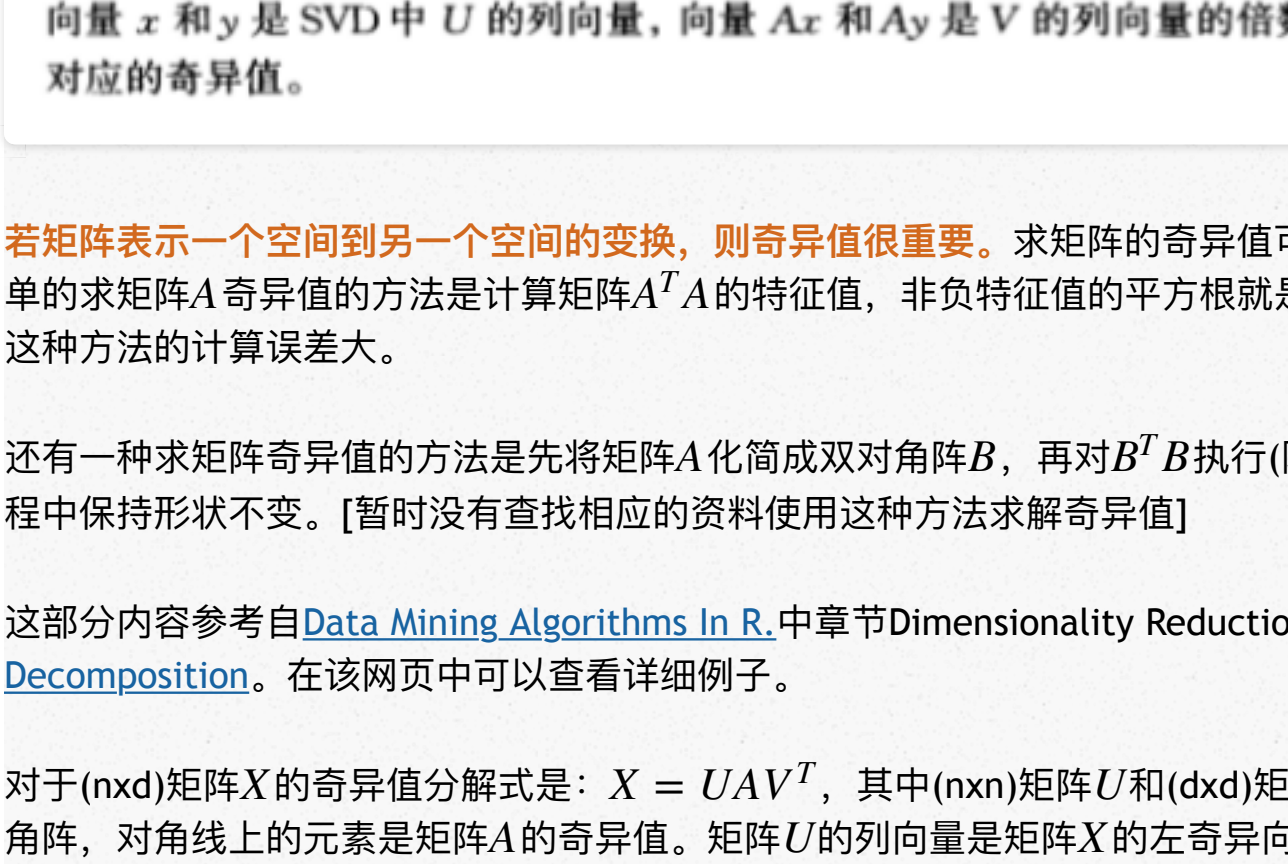


图10-2 中最后的两幅子图，显示了这个 2×2 矩阵例子的特征值以及特征向量。第一个特征值是正的，所以 Ax 与特征向量 x 的方向相同。Ax 的长度是对应的特征值，在这个例子中刚好就是 5/4。第二个特征值是负的，所以 Ax 与 x 平行，但是指向相反的方向。Ax 的长度是 1/2，所以对应的特征值是 −1/2。

图 10-2 egishow


你可能已经注意到了，这两个特征向量并非椭圆的长轴和短轴。但是如果这个矩阵是对称的，它们就是。egishow 默认的矩阵接近对称，但不完全对称。对于其他矩阵来说，可能根本找不到一个实的 x，使得 Ax 与 x 平行。我们在练习中会有这样的例子，它们将证明 2×2 的矩阵可以有少于 2 个实的特征向量。

在 SVD 中椭圆的轴起着关键的作用。图 10-3 显示了 svd 模式下，egishow 的结果。同样，鼠标可以沿着单位圆移动 x，而且这次有另一个单位向量 y 跟着 x 移动，且始终保持和它垂直。得到的 Ax 和 Ay 沿这个椭圆来回移动，但是彼此并非总保持互相垂直。我们的目标是使它们相互垂直，那个时候它们就构成了椭圆的轴。

图 10-3 egishow(svd)

向量 x 和 y 是 SVD 中 U 的列向量，向量 Ax 和 Ay 是 V 的列向量的倍数，而轴的长度就是对应的奇异值。

若矩阵表示一个空间到另一个空间的变换，则奇异值很重要。求矩阵的奇异值可以利用求矩阵的特征值的方法，最简单的求矩阵A奇异值的方法是计算矩阵$A^T A$的特征值，非负特征值的平方根就是特征值(对称矩阵的一个性质)，但是这种方法的计算误差大。

还有一种求矩阵奇异值的方法是先将矩阵A化简成双对角阵B，再对$B^T B$执行(隐式)QR迭代算法。$B_k^T B_k$在QR迭代过程中保持形状不变。[暂时没有查找相应的资料使用这种方法求解奇异值]

这部分内容参考自Data Mining Algorithms In R.中章节Dimensionality Reduction中的Singular Value Decomposition。在该网页中可以查看详细例子。

对于(nxd)矩阵X的奇异值分解式是：$X = UAV^T$，其中(nxn)矩阵U和(dxd)矩阵V都是正交阵，而(nxd)矩阵A是对角阵，对角线上的元素是矩阵A的奇异值。矩阵U的列向量是矩阵X的左奇异向量，矩阵V的列向量是矩阵X的右奇异向量。矩阵A对角线上的奇异值自上到下是从大到小分布的，并且所有的奇异值都大于0。

要计算矩阵X的奇异值分解式，就要计算$X^T X$和XX^T的特征值和特征向量。其中，**矩阵$X^T X$的特征向量就是矩阵V的列，而矩阵XX^T的特征向量就是矩阵U的列，矩阵X的奇异值就是矩阵$X^T X$和XX^T的共同非负特征值的平方根。矩阵X的奇异值的个数是矩阵X的秩，也就是线性无关的行(或者列)的数目。**

SVD分解的过程：(1)计算矩阵$X^T X$和矩阵XX^T的特征值和特征向量；(2)计算矩阵$X^T X$和矩阵XX^T的共同非负特征值的平方根得到奇异值；(3)通过计算的结果得到矩阵U，矩阵V和矩阵A。

[注：实际上不需要都进行上面的三个步骤，因为$X^T X$的特征值与特征向量和XX^T的特征值与特征向量是有联系的，容易证明，所以，其实只要计算两者中维度更小的那个即可，可以再查阅些资料学习]

在Matlab中直接使用内置函数`svd`即可，示例：

$[U, S, V] = svd(X)$

$[U, S, V] = svd(X, 0)$简化版本

Posted by hujiawei • Apr 29th, 2014 • math

« LAS 5-Orthogonal Matrix   Wouldn't it be nice »