第四版

数值线性代数

张　强　编著

2016 年 8 月

南京大学数学系
中国南京

前　言

作为南京大学数学系计信专业《数值计算方法 II》的配套讲义，其主要内容包括线性方程组、矩阵特征值问题、以及非线性方程（组）的数值解法。主要参考书目：

- ✉ 林成森, 数值计算方法（第二版）, 科学出版社, 2005
- ✉ 李大明, 数值线性代数, 清华大学出版社, 2010
- ✉ 蔡大用, 数值代数, 清华大学出版社, 2005
- ✉ 徐树方, 高立, 张平文, 数值线性代数, 北京大学出版社, 2010
- ✉ 威尔金森, 代数特征值问题, 石钟慈, 邓建新译, 科学出版社, 2001
- ✉ 李庆扬，王能超，易大义，数值分析，清华大学出版社，2010
- ✉ 李庆扬，关治，白峰杉，数值计算原理，清华大学出版社，2009

目 录

第 1 章
线性方程组的直接解法

直接解法是有限步完成的准确算法。换言之，在没有任何舍入误差的情况下，直接解法经过有限次的四则运算（可能包括少量的开方运算），可以给出准确的计算结果。本章重点介绍线性方程组的高斯消元方法。尽管各种实现方法在理论上是等价的，但是它们在计算机上的具体数值表现却并不相同。在学习的过程中，我们要关注各种实现方法的具体操作过程，包括：(a) 数据操作和编程技巧; (b) 数据存储量和计算复杂度；以及 (c) 舍入误差的控制。

1.1 高斯消元方法

对于中小规模（矩阵阶数是 1000 ∼ 10000）的线性代数方程组

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.1.1}$$

高斯消元方法是最常用的方法。特别地，当系数矩阵 $\mathbb{A}$ 是随机稠密（几乎处处非零）的时候，高斯消元方法堪称是最佳的选择。

消元思想最早出现在中国秦汉时代的《九章算术》；直至 19 世纪初，西方国家才基于消元思想，提出了高斯消元方法[i]。在 20 世纪中叶以后，由于科技的飞速发展，线性方程组的求解规模（或矩阵阶数）也变得越来越大。数字计算机作为重要的计算工具，相应的高斯消去方法也需要细致的研究。

[i] 1809 年，发表于 Theoria Motus

1.1.1 基本过程

事实上, 在线性代数课程中, 大家都学过高斯消元方法。其基本思想是：通过消元过程, 不断缩减待解的未知量个数, 使得线性方程组 (1.1.1) 转化为同解的三角形线性方程组。若用矩阵语言来描述, 消元过程等价于增广矩阵 $[\mathbb{A}|\boldsymbol{b}]$ 的上梯形化过程, 即通过一系列的初等行变换（或者初等变换矩阵的左乘）过程, 将其转化为上梯形矩阵。

在 Matlab 中, 高斯消元方法的对应命令是 A\b, 其中 A = $\mathbb{A}$ 是 (1.1.1) 的系数矩阵, 而 b = $\boldsymbol{b}$ 是右端向量。

✎ **论题 1.1.** 顺序高斯消元算法是简单的处理过程。

作为数值方法, 我们还要关注它在计算机上的自动实现过程, 特别是关于数据的存储和操作问题。基于实用的角度, 我们希望数据存储量要尽可能的精简, 数值计算结果要尽可能的可靠。

```
1.  For k = 1, 2, ..., n, Do
2.    For i = k + 1, ..., n, Do
3.      a_ik := a_ik / a_kk;
4.    Enddo
5.    For j = k + 1, ..., n, Do
6.      For i = k + 1, ..., n, Do
7.        a_ij := a_ij − a_ik a_kj;
8.      Enddo
9.    Enddo
10. Enddo
```

在左侧的图文框中,我们给出顺序高斯消元方法的伪代码, 其中 a_{ij} 是系数矩阵对应的二维数组元素, n 是矩阵的阶数或二维数组的维数。请注意, 赋值符号":="蕴含着简单而重要的数据覆盖技术。换言之, 在对应系数矩阵 $\mathbb{A}$ 的二维数组中, 位于对角线下方位置的数据将依次被相应的消元乘子所覆盖, 而位于对角线及其上方位置的数据将被高斯消元方法最终给出的上三角矩阵元素所覆盖。**数据存储单元的元素变化是数值方法实现的一个重点**。

在实际的数值计算中, 我们常常把右端项 $\boldsymbol{b}$ 放在矩阵 $\mathbb{A}$ 的右侧, 形成相应的增广矩阵。它依旧可以用一个二维数组来存储。此时, 我们仅

需修改图文框中的第 5 步，将关于 j 的取值范围扩充到 $n+1$。

算法的计算复杂度，通常可以利用乘除法运算的次数来衡量。在系数矩阵的高斯消元过程中，乘除法运算的总次数为

$$\sum_{k=1}^{n-1}(n-k)(n-k+1)=\mathcal{O}(n^3/3).$$

相应右端项的消元操作，仅仅需要 $\mathcal{O}(n^2)$ 次的乘除法运算。在消元之后，我们只需求解上梯形部分所对应的三角形方程组，就能得到最终的答案。相应的求解过程称为回代过程，它只需要共 $\mathcal{O}(n^2)$ 次的乘除法运算。

注意到顺序高斯消元方法含有除法运算，我们需要考虑除法运算的合理性。或者说，我们需要讨论这个算法是否可以顺利地执行到底。若消元过程中，后续的对角元素为零，则顺序高斯消元方法将意外终止。

暂时假设所有的四则运算都是准确的，我们有如下的理论结果。

定理 1.1. *若顺序高斯消元过程中的对角线元素*

$$a_{kk}^{(k)}\neq 0,\quad k=1,2,\ldots,n-1, \tag{1.1.2}$$

则消元过程可执行到底。(1.1.2) *成立的充分必要条件是矩阵* $\mathbb{A}$ *的前* $n-1$ *阶顺序主子阵都是非奇异的。若所有对角线元素均不为零，则回代过程也可执行。*

在顺序高斯消元过程中，我们需要不断地检验 (1.1.2) 是否成立。事实上，我们应当在除法运算前，都需要事先判断是否除零。但是，若系数矩阵具有某些特殊的结构，我们不必担心此事。因为，我们可以事先得到非常明确的结论，例如

定理 1.2. *若系数矩阵* $\mathbb{A}$ *是对称正定的，则顺序高斯消元法可执行到底，且每一步的中间矩阵的元素绝对值不超过原矩阵元素的最大值。*

证明：见后面的矩阵 Cholesky 分解部分。 □

代码编写是数值方法研究中非常重要的工作之一，因为它对数值算法的实现效率有重要的影响。一个高效的计算机代码应该强烈地依赖于所用的计算机语言，以及所用的计算机硬件结构。事实上，一个比较关键的因素是数据的读写效率。

✯ **说明 1.1.** 在图文框中的伪代码中，三重循环次序是 k–j–i。这种方法特别适用于 *Fortran* 语言，因为此时的二维数组是按列连续存放的，数据寻址所付出的代价较低。

然而，*C++* 语言中的二维数组是按行连续存放的。若依旧使用 k–j–i 三重循环次序，数据的读取是跳跃的，使得数据指针的移动过于频繁，从而消耗过多的计算时间，影响算法的性能。此时，我们有必要交换最内两层的循环次序。

✯ **说明 1.2.** 代码编写和执行效率还同计算机的硬件结构有关，特别是数据的读写加速设备（例如二级缓存等）的使用。

在图文框中的伪代码中，所有的数据操作都是数和数之间的单一四则运算。每个四则运算都要进行一次数据的读写，而数据的读写要比四则运算消耗更多的 *CPU* 时间。这样的代码仅仅处于 *BLAS-1* 的代码级别[ii]，处理大规模数据的能力和效率都很低。

实际上，数值计算还可以采用更高的 *BLAS* 代码级别来实现。譬如，在 *BLAS-2* 代码级别中，数据操作基于向量与向量乘积（含矩阵与向量乘积）的块数据操作。借用 *Matlab* 中的向量语言，顺序高斯消元方法的 *BLAS-2* 版本为：

[ii] BLAS=Basic Linear Algebraic Subroutine.

1. *For* $k = 1, 2, \ldots, n$, *Do*
2. $\quad \mathbb{A}(k+1:n, k) := \mathbb{A}(k+1:n, k)/\mathbb{A}(k, k)$;
3. $\quad \mathbb{A}(k+1:n, k+1:n) :=$
$$\mathbb{A}(k+1:n, k+1:n) - \mathbb{A}(k+1:n, k) \star \mathbb{A}(k, k+1:n);$$
4. *Enddo*

这种代码可以充分减少数据的读写时间。最高的代码级别是 *BLAS-3*, 它主要基于矩阵与矩阵乘积的块数据操作。这需要涉及很多关于并行计算的实现过程。因其超出课程设置，详略。

事实上, 即使我们可以在计算机上顺利地执行顺序高斯消元过程, 得到的数值结果并没有像理论上那样完美。在上面的讨论中，我们均基于精确计算，忽略了近似计算的影响。我们要指出：由于庞大的四则运算, 近似计算带来的舍入误差积累是不可轻视的。有时候，它们会造成计算结果的严重偏离。

构造计算机上可行的算法，建立数值稳定性分析，划定算法的适用范围，指出算法的缺陷，完善算法的实现效果等等问题，均是计算数学的重要特点和主要任务。在学习、使用和构造计算方法时，我们要随时注意上述问题。

1.1.2 主元技巧

由于数字计算机无法回避舍入误差的问题，顺序高斯消元过程给出的计算结果可能是极其不准确的，甚至可能是毫无价值的。计算结果的偏差程度通常同计算机的位长（或者机器精度）有关。

为说明这个现象，让我们在仅具有三位有效数字的十进制虚拟计算机上，采用顺序高斯消元方法，求解如下的线性方程组。数据流的变化

是

$$\begin{bmatrix} 0.001 & 1.00 & 1.00 \\ 1.000 & 2.00 & 3.00 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.001 & 1.00 & 1.00 \\ 1000 & -1000 & -1000 \end{bmatrix}.$$

回代过程给出数值解 $x_{\text{num}} = (0.00, 1.00)^\top$，它与精确解

$$x_\star = (1.002\cdots, 0.998\cdots)^\top$$

相去甚远。若我们增加计算机的位长，顺序高斯消元方法可以给出更好的数值解。

因此说，如果舍入误差的影响非常严重，理论上准确的算法也不能在计算机上直接应用。在现有的计算环境下，如何构造可行的算法，精细地控制舍入误差的产生和积累，是数值方法中一个非常重要和必要的研究工作。

为控制高斯消元方法中的舍入误差，最简单易行的方法是引进高斯主元策略，将高斯顺序消元方法修正为高斯主元消元方法。所谓的高斯主元，是指位于当前对角线位置右下方，按绝对值最大的那个矩阵元素。常用的主元策略有列主元/全主元（Wilkinson, 1961）策略和⊠型主元（Neal 和 Poole, 1992）策略，其中列主元策略在搜索时间上占有优势，故常作为首选策略。

✎ **论题 1.2.** 列主元高斯消元方法的代码实现是非常简单的。对于顺序高斯消元方法的任意两个版本，我们都仅需在相应的第 *2* 行代码前，填入如下一小段的补丁代码

> - 选择 l 使得 $|a_{lk}| = \max_{k\le i\le n} |a_{ik}|$，交换第 l 行和第 k 行数据；

这样的处理，可以使所有的消元乘子按绝对值均不超过 *1*，消元过程中的乘法运算所带来的舍入误差被有效地控制。

✯ **说明 1.3.** 事实上，行交换过程要花费很多毫无价值的数据读写时间。为提高代码的执行效率，我们需要引进一个 n 维指标向量 $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$，并重写代码，避免数据的移动。

实际上，指标向量 $\boldsymbol{p}$ 记录整个消元过程中的行交换信息。它的初始状态是自然序列，即 $p_i = i$。若第 k 步消元需要进行第 k 行和第 r 行的交换，我们只需轮换指标向量 $\boldsymbol{p}$ 中的第 k 分量和第 r 个分量，并调整对应循环中的行指标。

❀ **思考 1.1.** 重写代码，实现上述思想。

计算经验表明：同全主元策略相比，列主元策略在数值稳定性方面不相上下，但主元搜索时间却得到明显的减少。因此，在中小型稠密线性方程组的数值计算方法中，列主元高斯消元法已成为最受欢迎的算法之一。

✯ **说明 1.4.** 教科书还给出了另外一种列主元选取策略，即按比例选取列主元。在每次寻求高斯消元主元之前，我们首先将右下角矩阵的所有行向量按最大模分量进行单位化。这样的处理工作等价于一个所谓的预处理过程。

✯ **说明 1.5.** 行列式的计算是简单的，我们可以利用顺序高斯消元过程得到的对角线元素乘积得到。若采用列主元策略，请注意行交换的次数。详略。

1.1.3 高斯消元变换阵

我们常常使用矩阵（或行变换）语言来描述高斯消元过程，其本质就是系数矩阵的三角化过程，或增广矩阵的上梯形化过程。相应的核心问题是：

已知 m 维非零向量 $\boldsymbol{a}=(a_1,a_2,\ldots,a_m)$，其中 $a_1\neq 0$。我们能否构造一个简单矩阵 $\mathbb{H}$，左乘向量 $\boldsymbol{a}$ 后，可将 $\mathbb{H}\boldsymbol{a}$ 转化为仅首个位置非零的向量？

事实上，这个问题是数值代数中的基本问题。

相应的实现方法有很多，例如高斯消元阵、Householder 镜像变换矩阵、和 Givens 平面旋转矩阵。它们都是所谓的简单矩阵，即它们具有某些特殊的结构，在计算复杂度方面具有一定的优势。本章暂且考虑高斯消元阵；后两个矩阵将在第三章中介绍。

定义 1.1. 记 $\hat{\boldsymbol{e}}_1=(1,0,0,\ldots,0)^\top$，是首个位置为 *1* 的 m 维单位向量。对于向量 $\boldsymbol{a}$，相应的 m 阶 *Gauss* 消元阵为单位矩阵的秩一修正，即

$$\mathbb{S}_{m\times m}=\mathbb{I}_{m\times m}-\hat{\boldsymbol{p}}\hat{\boldsymbol{e}}_1^\top, \tag{1.1.3}$$

其中 $\hat{\boldsymbol{p}}=(0,a_2/a_1,a_3/a_1,\ldots,a_m/a_1)^\top$ 是由所有高斯消元乘子构成的 m 维列向量。

通常，我们将上述数值目标扩展到 n 维向量

$$\boldsymbol{a}=(a_{1k},a_{2k},\ldots,a_{kk},\ldots,a_{nk})^\top, \tag{1.1.4}$$

它可理解为第 k 步高斯消元过程中的第 k 列向量。设 $a_{kk}\neq 0$，我们可以利用非零的对角分量 a_{kk}，将其下方的所有元素清零。

定义 1.2. 令 $\ell_{ik}=a_{ik}/a_{kk}$ 为相应的消元乘子，定义

$$\boldsymbol{\ell}_k=(0,0,\ldots,0,\ell_{k+1,k},\ell_{k+2,k},\ldots,\ell_{n,k})^\top.$$

对应的高斯消元矩阵为

$$\mathbb{L}_k^{-1}=\mathbb{I}-\boldsymbol{\ell}_k\boldsymbol{e}_k^\top, \tag{1.1.5}$$

其中 $\boldsymbol{e}_k$ 是第 k 个分量为 1 的 n 维单位向量。事实上，这个矩阵可以理解为 $n-k+1$ 阶的高斯消元阵的单位矩阵扩张[iii]，即

$$\mathbb{L}_k^{-1} = \begin{bmatrix} \mathbb{I}_{(k-1)\times(k-1)} & \mathbb{O} \\ \mathbb{O} & \mathbb{S}_{(n-k+1)\times(n-k+1)} \end{bmatrix}. \tag{1.1.6}$$

✍ **论题 1.3.** 注意到向量 $\boldsymbol{\ell}_k$ 的构成方式，顺序高斯消元的第 k 步操作可描述为高斯消元阵 (1.1.6) 的左乘。因此，顺序高斯消元方法可描述为

$$\mathbb{L}_{n-1}^{-1}\cdots\mathbb{L}_2^{-1}\mathbb{L}_1^{-1}\mathbb{A}^{(1)} = \mathbb{A}^{(n)}, \quad \mathbb{L}_{n-1}^{-1}\cdots\mathbb{L}_2^{-1}\mathbb{L}_1^{-1}\boldsymbol{b}^{(1)} = \boldsymbol{b}^{(n)},$$

其中 $\mathbb{A}^{(1)} = \mathbb{A}$ 和 $\boldsymbol{b}^{(1)} = \boldsymbol{b}$ 是相应的代数方程组信息。

可证高斯消元矩阵具有如下的基本性质：

$$\mathbb{L}_k = \mathbb{I} + \boldsymbol{\ell}_k \boldsymbol{e}_k^\top; \qquad \mathbb{L}_i\mathbb{L}_j = \mathbb{L}_i + \mathbb{L}_j - \mathbb{I}, \quad (i<j).$$

由高斯顺序消元过程，我们可以衍生出一个重要的矩阵分解，即

$$\mathbb{A} = \mathbb{L}\mathbb{U},$$

其中 $\mathbb{U} = \mathbb{A}^{(n)}$ 是上三角矩阵，而

$$\mathbb{L} = \mathbb{L}_1\cdots\mathbb{L}_{n-1} = \begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{n,n-1} & 1 \end{bmatrix}$$

[iii] 这种默认的扩张方式将在本课程中多次使用，以后不再赘述。

是单位下三角矩阵，其中

$$\ell_{ij} = a_{ij}^{(j)}/a_{jj}^{(j)}$$

是高斯消元乘子。这种方式通常称为矩阵的三角分解，在矩阵理论中占有非常重要的地位。类似的工作及其应用将在下节内容中做深入的介绍。

✍ **论题 1.4.** 列主元的选取可描述为初等排列阵的左乘运算。从而，列主元高斯消元法可矩阵描述为：

$$\mathbb{U} = \mathbb{A}^{(n)} = \mathbb{L}_{n-1}^{-1}\mathbb{I}_{n-1,r_{n-1}}\cdots\mathbb{L}_2^{-1}\mathbb{I}_{2,r_2}\mathbb{L}_1^{-1}\mathbb{I}_{1,r_1}\mathbb{A}^{(1)},$$

其中 $\mathbb{U}$ 是消元之后的上三角矩阵，$\mathbb{L}_k^{-1}$ 是对应第 k 步列主元交换后的高斯消元矩阵。利用数学归纳法，我们可证

$$\mathbb{U} = \underbrace{\mathbb{L}_{n-1}^{-1}\widetilde{\mathbb{L}}_{n-2}^{-1}\cdots\widetilde{\mathbb{L}}_2^{-1}\widetilde{\mathbb{L}}_1^{-1}}_{\widetilde{\mathbb{L}}^{-1}}\underbrace{\mathbb{I}_{n-1,r_{n-1}}\cdots\mathbb{I}_{2,r_2}\mathbb{I}_{1,r_1}}_{\mathbb{P}}\ \mathbb{A},$$

其中 $\mathbb{P}$ 是由单位矩阵经过相应的行交换而形成的置换阵，而

$$\widetilde{\mathbb{L}}_k^{-1} = \mathbb{I}_{n-1,r_{n-1}}\cdots\mathbb{I}_{k+1,r_{k+1}}\mathbb{L}_k^{-1}\mathbb{I}_{k+1,r_{k+1}}\cdots\mathbb{I}_{n-1,r_{n-1}} \tag{1.1.7}$$

的非零元素分布和 $\mathbb{L}_k^{-1}$ 相同，仅仅是高斯消元乘子的所在位置略有不同。如前面的讨论，可知矩阵 $\widetilde{\mathbb{L}}$ 具有单位下三角结构。

1.1.4 逆矩阵的计算

利用高斯消元方法，求解一系列的同型线性代数方程组

$$\mathbb{A}\boldsymbol{x}_j = \boldsymbol{e}_j, \quad j = 1, 2, \ldots, n.$$

我们可以给出矩阵 $\mathbb{A}$ 的逆矩阵 $\mathbb{A}^{-1} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$。但是，这个算法的计算复杂度较高，整个过程共需 $\mathcal{O}(4n^3/3)$ 次乘除法运算。

为此, 我们通常采用更为快速的 Gauss-Jordan (G-J) 消元算法, 进行逆矩阵的计算。G-J 消元方法也是一个古老的算法，它是由测量技师 Wilhelm Jordan (1842-1899) 和 B.I. Clasen (1887) 分别独立提出。

论题 1.5. *Gauss-Jordan 消元的基本思想：利用对角元素，直接消去同列的其他所有元素。设第 k 列元素是 (1.1.4)，其中 $a_{kk} \neq 0$。相应的 G-J 消元过程可描述为 G-J 消元矩阵*

$$\mathbb{M}_k = \begin{bmatrix} 1 & & & m_{1k} & & \\ & \ddots & & \vdots & & \\ & & 1 & m_{kk} & & \\ & & & m_{k+1,k} & & \\ & & & m_{k+2,k} & 1 & \\ & & & \vdots & & \ddots \\ & & & m_{n,k} & & 1 \end{bmatrix}$$

的左乘，其中的 G-J 消元因子是

$$m_{ik} = \begin{cases} 1/a_{kk}, & i = k; \\ -a_{ik}/a_{kk}, & i \neq k. \end{cases}$$

因数据存储方式不同，它的程序实现可以有两种方式。一种方法是存储增广矩阵 $[\,\mathbb{A}|\,\mathbb{I}\,]$，然后执行 G-J 消元过程。另一种方法是借助数据覆盖技术，仅仅存储矩阵 $\mathbb{A}$，可节省一半的数据存储空间。相应的伪代码如下：

```
1.  For k = 1, 2, ..., n, Do
2.      交换 A 的第 k 行和第 p_k 行，其中 p_k 为相应的列主元；
3.      a_kk = 1/a_kk;
4.      For i = 1, ..., n 且 i ≠ k, Do a_ik := −a_ik a_kk; Enddo
5.      For i = 1, ..., n 且 i ≠ k, Do
6.          For j = 1, ..., n 且 j ≠ k,
7.            Do a_ij := a_ij + a_ik a_kj;
8.          Enndo
9.      Enddo
10.     For j = 1, ..., n 且 j ≠ k, Do a_kj := a_kk a_kj; Enndo
11. Enddo
12. For k = n, n − 1, ..., 1, Do
13.     交换 A 的第 k 列和第 p_k 列；
14. Enddo
```

显然，这个算法总共需 $\mathcal{O}(n^3)$ 次乘除法运算。在 Matlab 中，逆矩阵的相应命令是 inv()。

请注意上述伪代码三个关键操作的次序：首先计算同列的单位化因子和消元乘子（第 3-4 行），然后执行其余各列的 G-J 消元（第 5-9 行），最后进行相应行的单位化（第 10 行）。

算法中的第 12-14 行是恢复计算数据的真正存储位置。因为在执行第 k 步 G-J 消元的时候，实际操作对应 $\mathbb{M}_k\mathbb{I}_{k,i_k}$ 的左乘，其中 $\mathbb{M}_k$ 为相应的 G-J 消元阵。相应的 G-J 消元因子应出现在第 i_k 列。但是，在计算机上我们未执行相应的列交换，它们依旧存储在第 k 列。

✯ **说明 1.6.** *若将 G-J 消元算法应用于单个线性代数方程组的求解，它的计算效率比高斯消元方法要差，共需 $\mathcal{O}(n^3/2)$ 次乘除法运算。*

✯ **说明 1.7.** *逆矩阵的求解方法还有很多种。例如，我们可采用 Newton 迭代法 $\mathbb{X}_{k+1} = 2\mathbb{X}_k(\mathbb{I} - \mathbb{A}\mathbb{X}_k)$，求解矩阵 $\mathbb{A}$ 的逆矩阵。因课时*

限制，此处不再赘述。

1.2 直接三角解法

本节考虑高斯消元方法的其他实现方式。为简单起见，我们略去回代过程，重点关注系数矩阵的处理过程。因为设计出发点是系数矩阵的各种三角分解，相应的算法称为直接三角解法。若所有计算都是准确的，直接三角解法同顺序高斯消元方法是完全等价的。但是，它们的数据走向和运算次序是完全不同的。

1.2.1 矩阵三角分解

定义 1.3. 若存在上三角矩阵 $\mathbb{U}$ 和下三角矩阵 $\mathbb{L}$，使得

$$\mathbb{A} = \mathbb{L}\mathbb{U}, \tag{1.2.8}$$

则称矩阵 $\mathbb{A}$ 具有三角分解。若 $\mathbb{L}$ 为单位下三角阵，称其为 *Doolittle* 分解；若 $\mathbb{U}$ 为单位上三角阵，称其为 *Crout* 分解。

请注意：按照这个定义，并不是所有的矩阵都具有相应的三角分解，例如

$$\mathbb{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

那么，我们能否找到矩阵存在 LU 三角分解的条件呢？

定理 1.3. 若 n 阶矩阵 $\mathbb{A}$ 的前 $n-1$ 个顺序主子式

$$\mathbb{A}_k = \det \mathbb{A}(1:k, 1:k), \quad k = 1:n-1,$$

均非奇异，则顺序高斯消元法可以给出矩阵 $\mathbb{A}$ 的 *Doolittle* 分解。

✯ **说明 1.8.** 定理 *1.3* 中的条件仅仅是充分的。若关于顺序主子式的条件不成立，矩阵也可以有 LU 分解。譬如，

$$\begin{bmatrix} 0 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{1.2.9}$$

请问这个矩阵的 LU 分解是唯一的吗？

✯ **说明 1.9.** 由论题 *1.4* 可知，即使可逆矩阵 $\mathbb{A}$ 不存在 LU 三角分解，它也会有如下的三角分解

$$\mathbb{PA} = \mathbb{LU},$$

其中 $\mathbb{P}$ 是某个置换阵，$\mathbb{L}$ 和 $\mathbb{U}$ 分别是下三角阵和上三角阵。

矩阵的 LU 三角分解是不唯一的。为保证三角分解具有唯一性，我们需要考虑它的一个标准形式，即两个三角形矩阵都是单位的（对角线元素恒为一）。

定义 1.4. 称 $\mathbb{A} = \mathbb{LDR}$ 为一个 LDR 三角分解，若 $\mathbb{D}$ 为对角阵，$\mathbb{R}$ 和 $\mathbb{L}$ 分别为上下单位三角矩阵。

定理 1.4. n 阶矩阵 $\mathbb{A}$ 具有唯一的 LDR 三角分解，当且仅当顺序主子式 $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_{n-1}$ 均非奇异。

证明：数学归纳法与矩阵分块。见教科书。 □

1.2.2 矩阵分解的应用

基于矩阵的某种三角分解，我们可以给出相应的直接三角解法。设计思想比较简单，就是所谓的矩阵乘法公式。对于 LU 三角分解，有

$$\boxed{a_{ij} = \sum_{r=1}^{\min(i,j)} l_{ir} u_{rj},}$$

其中某个上（下）三角阵的对角线要求恒为一。若 $\mathbb{L}$（或 $\mathbb{U}$）是单位三角阵，则相应的方法称为 Doolittle（或 Crout）方法。

Crout 方法

为说明这种算法的实现过程，我们以 Crout 方法为例[iv]。它同顺序高斯消元法的主要区别是计算次序的不同。系数矩阵中的数据呈瀑布型方式，由左上角向右下角，先列后行地依次更新一次。

在下面的图文框中，我们给出了 Crout 方法的伪代码。其中，数据覆盖技术被采用，分解后的两个三角型矩阵数据依旧存储在原有数据的位置上。若引进列主元策略，我们只需在伪代码的第 5 行前，添加相应的补丁代码。请注意：在选取主元之前，我们要先完成相应列的计算。详略。

1. For $k = 1, 2, \ldots, n$, Do
2. For $i = k, k+1, \ldots, n$, Do
3. $a_{ik} := a_{ik} - \sum_{r=1}^{k-1} a_{ir} a_{rk}$;
4. Enddo
5. For $j = k+1, k+2, \ldots, n$, Do
6. $a_{kj} := (a_{kj} - \sum_{r=1}^{k-1} a_{kr} a_{rj}) / a_{kk}$;
7. Enddo
8. Enddo

请注意：若求和符号中的上标小于下标，则对应的求和操作（对应代码的第 3 行和第 6 行）为一个空操作，相应的返回值为零。这个默认准则将在以后的讨论中一直使用，不再赘述。

直接三角解法在本质上也是一种高斯消元方法，它可视为不同的执行过程而已。但是，这样的算法在某些方面具有自己的优势，例如它不必计算和存储消元过程中的中间结果。

1. 在每次高斯消元过程中，我们都要更新右下角的整块矩阵。换言之，

[iv] Doolittle 方法是类似的。事实上，Doolittle 算法就是顺序（或列主元）高斯消元方法的不同实现过程而已。若计算过程是精确的，相应的计算结果是完全一致的，仅仅是计算流程和数据控制略有不同。在 Matlab 中，相应的命令是 lu().

右下角的元素将被多次地更新。计算所涉及到的数据范围会非常大，使得数据指针的移动距离非常大，需要耗费大量的数据读写时间。

2. 而在直接三角解法中，为更新某个位置，我们仅仅需要读写同这个位置处于同行或同列的相关数据。换言之，直接三角解法是一种“需求驱动”的算法，在数据读写方面的代价将有明显的下降。这点完全不同于高斯消元法。

请注意：它们的计算复杂度是一样的。效率的提升来自数据读写方面。

Cholesky 方法

若系数矩阵 $\mathbb{A}$ 是一个实对称正定矩阵，三角分解的计算复杂度可以进一步降低。我们可采用著名的 Cholesky 方法[v] 或 $\mathbb{L}\mathbb{L}^\top$ 算法。在 Matlab 中，相应的命令是 chol().

定理 1.5. *设 $\mathbb{A}$ 是实对称的正定矩阵，则它有三角分解*

$$\mathbb{A} = \mathbb{L}\mathbb{L}^\top, \tag{1.2.10}$$

其中 $\mathbb{L}$ 是一个下三角阵。若要求 $\mathbb{L}$ 的对角线元素均为正数，则这种分解是唯一的。

Cholesky 方法具有重要的理论价值。例如，我们可以利用它的计算过程，判断一个给定的对称矩阵是否正定。

✍ **论题 1.6.** *基于三角分解 (1.2.10) 的方法称为 Cholesky 方法。它也常被称为平方根法，因为它的实现基于矩阵元素计算公式*

[v] André-Louis Cholesky 是一个法国军官，潜心于测地学研究，勘测过希腊克里特岛和北非。

$$a_{ij} = \sum_{k=1}^{j} l_{ik} l_{jk}, \quad i \geq j,$$

我们需要开根号运算，才能得到对角线元素的取值。同乘除法运算相比，开根号运算将消耗大量的 CPU 时间。

注意到矩阵分解的对称性，我们只需计算矩阵 $\mathbb{L}$ 的下三角部分。相应数据，可以按照逐列或者逐行的方式进行依次更新。因为逐行更新的方式难以实现并行化，故我们大多采用逐列更新的方式。在左侧的图文框中，我们给出了平方根方法的伪代码，其中的 l_{ij} 覆盖存放在原有数据 a_{ij} 的位置。

1. For $j = 1, 2, \ldots, n$, Do
2. $\quad a_{jj} := \left(a_{jj} - \sum_{k=1}^{j-1} a_{jk}^2\right)^{1/2}$;
3. $\quad$ For $i = j+1, j+2, \ldots, n$, Do
4. $\quad\quad a_{ij} := (a_{ij} - \sum_{k=1}^{j-1} a_{ik} a_{jk})/a_{jj}$;
5. $\quad$ Enddo
6. Enddo

这种逐列更新的方式也称为“向左看”算法。又因为直到第 j 步外围循环时，矩阵的第 j 列数据才会被更新，故而这种算法也称为“需求驱动的算法”或者“延迟更新的算法”。

定理 1.6. *若矩阵 $\mathbb{A}$ 是对称正定的，则 $l_{ij} \leq \sqrt{a_{ii}}$，其中 $j \leq i$。*

因此说，$\mathbb{L}$ 中的元素均是可以控制的，相应的算法是数值稳定的。此时，我们不用进行主元的选取。

论题 1.7. *为避免平方根算法中的开根号运算（它比乘除法慢很多），我们可采用修正的平方根算法。它基于所谓的标准三角分解*

$$\mathbb{A} = \mathbb{L}\mathbb{D}\mathbb{L}^{\top},$$

其中 $\mathbb{L}$ 是单位下三角阵，$\mathbb{D}$ 是正数构成的对角阵，因为 $\mathbb{A}$ 是正定的。

在 Matlab 中，修正的平方根算法的相应命令是 ldl(). 在修正的平

方根算方法中，基本计算公式为

$$a_{ij} = \sum_{k=1}^{j} l_{ik} d_k l_{jk}, \quad i \geq j.$$

此时，数据的逐行处理是便捷的。在下面的两个图文框中，我们分别给出了修正平方根算法的两个伪代码。

1. For $i = 1, 2, \ldots, n$, Do
2. For $j = 1, 2, \ldots, i-1$, Do
3. $a_{ij} := (a_{ij} - \sum_{k=1}^{j-1} a_{ik} a_{kk} a_{jk}) / a_{jj}$;
4. Enddo
5. $a_{ii} := a_{ii} - \sum_{k=1}^{i-1} a_{ik} a_{kk} a_{ik}$.
6.
7.
8.
9. Enddo

1. For $i = 1, 2, \ldots, n$, Do
2. For $j = 1, 2, \ldots, i-1$, Do
3. $a_{ij} := a_{ij} - \sum_{k=1}^{j-1} a_{ik} a_{jk}$;
4. Enddo
5. For $j = 1, 2, \ldots, i-1$, Do
6. $c := a_{ij}$; $a_{ij} := a_{ij} / a_{jj}$;
7. $a_{ii} := a_{ii} - c a_{ij}$;
8. Enddo
9. Enddo

虽然左侧代码有效地避免了开根号运算，但是乘除法的总次数却比前面的平方根算法增加了一倍。为减少左侧代码（第 3 行和第 5 行）中的重复运算，我们引进中间辅助变量

$$g_{ij} = l_{ij} d_j,$$

它对应右侧代码第 3 行中的 a_{ij} 和第 6 行的局部变量 c。右侧代码第 6 行的另一个代码，对应 g_{ij} 到 l_{ij} 的过程，相应数据存储在系数矩阵的原油位置（见第 3 行代码中的 a_{jk}）。这个例子是有效缩减计算复杂度的一个典型代表。

✯ **说明 1.10.** 我们要指出矩阵的对称正定性是一个非常重要的条件，它可以保证 $LDL^\top$ 算法具有良好的数值稳定性。

为说明这个断言，我们考虑一个简单的反例，即

$$\mathbb{A} = \begin{bmatrix} \varepsilon & 1 \\ 1 & \varepsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon - \varepsilon^{-1} \end{bmatrix} \begin{bmatrix} 1 & \varepsilon \\ 0 & 1 \end{bmatrix},$$

当 $|\varepsilon| \ll 1$ 很小时，矩阵 $\mathbb{A}$ 是不定的，上述 $LDL^\top$ 分解势必引起很大的舍入误差。但是，我们直接计算可知相应的逆矩阵为

$$\mathbb{A}^{-1} = \frac{1}{\varepsilon^2 - 1} \begin{bmatrix} \varepsilon & 1 \\ 1 & \varepsilon \end{bmatrix}.$$

直接按照这个公式计算，舍入误差的影响将非常小。

因此，对于不定的对称矩阵，我们常把列主元高斯消元法和二阶块高斯消去法相结合，来保证数值上的稳定性。

带状矩阵的分解

在多数的大规模科学计算时，线性方程组的系数矩阵会含有大量的零元素。它们将虚耗巨额的数据存储空间和大量的四则运算时间，降低高斯消元算法的处理能力和计算效率。为此，我们应当充分发掘矩阵的稀疏特性。例如，我们可以利用非零元素分布的结构特性，从数据存储和算法优化两个方面，对高斯消元方法进行改进。

✯ **说明 1.11.** 稀疏矩阵的数据存储是门相对繁杂的计算机技术。因篇幅有限，我们仅仅简要介绍一些基本技术，譬如变形存储，或者利用三元数结构体 (i, j, a_{ij}) 记录非零元素。为解决数据关联和便于快速搜索，我们需要引进单向链表或者双向链表等复杂数据结构。在 *Matlab* 中，相关的基本命令有 *sparse(), speye(), spones(), spdiag(), full()* 等等。详细内容可参阅 *Matlab* 的帮助文件。

✯ **说明 1.12.** 当利用高斯消元方法法求解稀疏的线性方程组时，零元素在消元之后可能会变成非零元素。这使得稀疏矩阵变成稠密矩阵，造

成数据存储的最大困难。换言之，我们需要适当控制新增的非零元素总数。这是数值处理的关键之处，最著名的数值策略有不完全三角分解。深入的讨论需要涉及图论的内容，详略。

在本讲义中，我们以最简单的带状矩阵为例。所谓的带状矩阵，是指远离对角线一定距离的矩阵元素均为零。利用数学归纳法，可以证明：对于带状矩阵，相应 LU 分解中的两个三角型矩阵依旧具有相同的带状结构。事实上，在这个带宽内，矩阵可能依旧存在巨额的零元素；相关的深入处理非常复杂，详略。

✍ **论题 1.8.** 设矩阵的半带宽为 d，我们可按斜线（或按行）存储技术存储矩阵，即仅仅用 $(2d-1)n$ 个存储单位替代普通的 n^2 个存储单位。请针对你的数据存储方式重写高斯消去法省略无用的零操作，并估算最终所需的乘除法次数是多少？

✯ **说明 1.13.** 我们需强调指出：带状矩阵的逆矩阵通常不再是带状矩阵。因此，若无特别要求，我们很少主动取计算一个矩阵的逆。

❀ **思考 1.2.** 带状矩阵的逆矩阵可能具有漂亮的结构。设 $\mathbb{H}$ 是不可约的上 *Hessenberg* 矩阵，即它比上三角矩阵多一条所有元素非零的副对角线。*Ikebe (1979)* 指出：存在两个列向量 $\boldsymbol{p} = (p_i)_{i=1}^n$ 和 $\boldsymbol{q} = (q_j)_{j=1}^n$，使得位于逆矩阵 $\mathbb{H}^{-1}$ 下三角区域的元素可表示为

$$(\mathbb{H}^{-1})_{ij} = p_i q_j, \quad i \geq j.$$

利用 *Ikebe* 的结果，我们可以给出一个直接算法，计算对称三对角矩阵的逆矩阵。具体内容留作练习[vi]。

[vi] 通常，$q_1 = 1$ 被预先地设定。

追赶法

作为稀疏矩阵的代表，三对角结构堪称最简单的等带宽矩阵。为表示简单，我们记它为

$$\mathbb{A} = \text{tridiag}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}),$$

其中 $\boldsymbol{a} = (a_i), \boldsymbol{b} = (b_i)$ 和 $\boldsymbol{c} = (c_i)$ 分别为从下至上的三条对角线向量；参见下面论题中的左侧图文框。

对于这样的线性方程组，相应的求解算法通常称为追赶法，或者 Thomas 算法。它具有非常简单的结构。

论题 1.9. 所谓的追赶法就是 *Crout* 算法及其两个三角型线性方程组的求解过程，此时的三角形矩阵仅仅有两个斜对角线元素非零。在下面右侧的图文框中，我们给出了相应伪代码中的 *Crout* 分解片段。矩阵分解所需的乘除法次数共计 $\mathcal{O}(2n)$，追和赶的过程需要 $\mathcal{O}(3n)$ 次乘除法。

$$\mathbb{A} = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \cdots & \cdots & \cdots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix}$$

1. $c_1 := c_1/b_1$;
2. *For* $i = 2, 3, \ldots, n$, *Do*
3. $\quad b_i := b_i - a_i c_{i-1}$;
4. $\quad c_i := c_i/b_i$;
5. *Enddo*

定理 1.7. 若三对角矩阵是严格对角占优的，即

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad i = 1 : n,$$

则追赶法可以顺利进行到底，我们无需进行主元的选取，相应的数值结果是稳定的。

证明：数学归纳法。见教科书。 □

对于三对角的线性方程组，求解方法有很多种，例如变参数追赶法和线性插值法。前者基于矩阵分解 $\mathbb{A} = \mathbb{D}\mathbb{L}\mathbb{R}$，其中 $\mathbb{D}$ 是对角阵，而 $\mathbb{L}$ 和 $\mathbb{R}$ 是相应的三角型矩阵。因篇幅有限，此处不再赘述。后者基于非齐次线性方程组解的线性结构；下面给予简要介绍。

我们线考虑前 $n-1$ 个方程所形成的不定线性方程组，它的通解可表示为两个特解的线性组合

$$\theta(0, \xi_2, \ldots, \xi_n)^\top + (1-\theta)(1, \eta_2, \ldots, \eta_n)^\top,$$

其中 θ 是待定系数。若所有的 c_i 均不等于零[vii]，则右上角的 $n-1$ 阶下三角矩阵块是可逆的，两个特解可以轻松地确定。待定系数 θ 可利用线性方程组的最后一个方程来确定。

❀ **思考 1.3.** 循环三对角方程组具有重要的应用价值。所谓的循环三对角阵就是，在原有三对角阵的右上角和左下角分别补充上相应的丢失元素。请利用矩阵分解技术，给出这个循环三对角方程组的追赶法实现过程。

1.3 向量范数和矩阵范数

为深入探讨上述数值方法的性质，我们需要建立数值代数分析领域中的基本工具，即所谓的向量范数和矩阵范数[viii]。范数的概念在泛函分析中具有重要地位，但是在数值分析（或矩阵论）中，真正有效使用这些概念是始于上世纪 40-50 年代。

[vii] 若某个 $c_i = 0$，则线性方程组可分割为两个小规模的问题。这样的问题称为可约的。

[viii] 本课程将更多地关注实数域上的向量和矩阵，虽然相应概念和结论可以非常容易地从实数域推广到复数域。

1.3.1 向量范数和矩阵范数的定义

定义 1.5. 向量范数的定义有三条规则：*(a)* 非负性；*(b)* 齐次性；*(c)* 三角不等式。具有某个范数度量的线性空间称为赋范空间。

设 $\boldsymbol{x} = (x_i)_{i=1}^n \in \mathbb{R}^n$，相应的 l_p 向量范数为

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^n |x_i|, \quad \|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^n |x_i|^2\right)^{1/2}, \quad \|\boldsymbol{x}\|_\infty = \max_{1\le i\le n} |x_i|,$$

其中 l_2 范数也称为 Euclid（欧几里得）范数。

在 $\mathbb{R}^n$ 空间，我们可以定义两个向量的内积

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \boldsymbol{x}^\top \boldsymbol{y}, \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

它满足著名的 Hölder 不等式：

$$|\boldsymbol{x}^\top \boldsymbol{y}| \le \|\boldsymbol{x}\|_p \|\boldsymbol{y}\|_q, \quad 1/p + 1/q = 1.$$

定义 1.6. 矩阵范数的定义有四条规则：*(a)* 非负性；*(b)* 齐次性；*(c)* 三角不等式；*(d)* 相容性。

说明 1.14. 在上述定义中，关于相容性的规则是非常重要的。前三条可视为向量范数的自然推广。

思考 1.4. 设 $\mathbb{A} = (a_{ij})$ 是一个 n 阶矩阵，请问 $\max_{ij} |a_{ij}|$ 是否是一个矩阵范数？那么，$n \max_{ij} |a_{ij}|$ 呢？

在 Matlab 中，向量范数和矩阵范数的相应命令都是 norm().

定理 1.8. 任意的两个向量（或矩阵）范数均是彼此等价的，且它们关于其元素的变化都是（一致）连续的。

证明：信息量略大。见教科书。 □

1.3.2 向量范数和矩阵范数的联系

定义 1.7. 设 $\|\cdot\|_\alpha$ 为矩阵范数，而 $\|\cdot\|_\beta$ 为向量范数。若成立

$$\|\mathbb{A}\boldsymbol{x}\|_\alpha \leq \|\mathbb{A}\|_\beta \|\boldsymbol{x}\|_\alpha, \quad \forall \boldsymbol{x} \in \mathbb{R}^n,$$

则称矩阵范数 $\|\cdot\|_\beta$ **相容**于向量范数 $\|\cdot\|_\alpha$。特别地，若存在某个非零向量使上述不等式中的等号成立，则称矩阵范数 $\|\cdot\|_\beta$ **从属**于向量范数 $\|\cdot\|_\alpha$.

定理 1.9. 对任意的矩阵范数 $\|\cdot\|_\beta$，均存在某个向量范数 $\|\cdot\|_\alpha$，使得两者是相容的；但是，它们不一定具有从属关系，因为

$$\|\mathbb{I}_{n\times n}\|_\beta = 1$$

是矩阵范数 $\|\cdot\|_\beta$ 从属于某个向量范数 $\|\cdot\|_\alpha$ 的必要条件。

Frobenius 范数（又称为 Suchur 范数）

$$\|\mathbb{A}\|_F = \Big(\sum_{i,j=1}^{n} |a_{ij}|^2\Big)^{1/2} = \Big(\mathrm{trac}(\mathbb{A}^\top \mathbb{A})\Big)^{1/2},$$

它与 l_2 向量范数相容，但不从属于任何向量范数。

思考 1.5. 证明 $\|\mathbb{AB}\|_F \leq \|\mathbb{A}\|_F \|\mathbb{B}\|_F$，进而说明 $\|\cdot\|_F$ 确实是一个矩阵范数。

定理 1.10. 对任意的向量范数 $\|\cdot\|_\alpha$，我们均可导出算子范数

$$\|\mathbb{A}\|_\alpha = \sup_{\boldsymbol{x}\neq 0} \frac{\|\mathbb{A}\boldsymbol{x}\|_\alpha}{\|\boldsymbol{x}\|_\alpha} = \max_{\|\boldsymbol{x}\|_\alpha = 1} \|\mathbb{A}\boldsymbol{x}\|_\alpha,$$

它是同向量范数 $\|\cdot\|_\alpha$ 从属（显然相容）的矩阵范数。这个定义可推广到任意形状的矩阵。

✍ **论题 1.10.** 分别对应三个常用的向量范数（$1,2,\infty$ 范数），我们可得如下三个（相容且从属的）矩阵范数:

1. 列范数: $\|\mathbb{A}\|_1 = \max\limits_{1\leq j\leq n}\sum\limits_{i=1}^{n}|a_{ij}|$;
2. 谱范数: $\|\mathbb{A}\|_2 = \left[\varrho(\mathbb{A}^\top\mathbb{A})\right]^{1/2}$, 其中 $\varrho(\mathbb{A}^\top\mathbb{A})$ 是矩阵 $\mathbb{A}^\top\mathbb{A}$ 的谱半径。
3. 行范数: $\|\mathbb{A}\|_\infty = \max\limits_{1\leq i\leq n}\sum\limits_{j=1}^{n}|a_{ij}|$.

显然，$\|\mathbb{A}\|_1 = \|\mathbb{A}^\top\|_\infty$ 和 $\|\mathbb{A}^\top\|_2 = \|\mathbb{A}\|_2$.

定理 1.11. 任意（相容）矩阵范数均满足 $\varrho(\mathbb{A}) \leq \|\mathbb{A}\|$.

定理 1.12. 对于任意正常数 ε，至少存在一个矩阵范数 $\|\cdot\|_\star$，使得

$$\|\mathbb{A}\|_\star \leq \varrho(\mathbb{A}) + \varepsilon. \tag{1.3.11}$$

定理 1.13. *(Banach 引理)* 设某个范数 $\|\cdot\|$，使得单位矩阵 $\mathbb{I}$ 满足 $\|\mathbb{I}\| = 1$。若 $\|\mathbb{A}\| < 1$（或者 $\varrho(\mathbb{A}) < 1$）时，则 $\mathbb{I}\pm\mathbb{A}$ 可逆，且

$$\frac{1}{1+\|\mathbb{A}\|} \leq \|(\mathbb{I}\pm\mathbb{A})^{-1}\| \leq \frac{1}{1-\|\mathbb{A}\|}. \tag{1.3.12}$$

定理 1.14. 矩阵的谱范数和 *Frobenius* 范数，在（左右）酉变换下均保持不变。

1.4 线性方程组的摄动理论

若线性方程组的已知数据发生变化，相应的解向量也会发生改变。那么，解向量的改变量受到怎样的限制呢？如果改变量没有受到很好的限制，数值计算是很难得到良好的结果。

1.4.1 条件数

线性方程组解的敏感程度同系数矩阵的条件数密切有关。条件数可以衡量一个线性方程组的固有不可靠性，描述已知数据的改变对于解向量的影响程度。要奢求一个算法的数值可靠性超过问题的固有可靠性，都是无望的。

定义 1.8. 可逆矩阵 $\mathbb{A}$ 的条件数定义为

$$\kappa(\mathbb{A}) = \|\mathbb{A}\|\|\mathbb{A}^{-1}\|,$$

其中 $\|\cdot\|$ 是某个矩阵范数。若条件数非常大[ix]，则称矩阵 $\mathbb{A}$ 是病态的；否则，称矩阵 $\mathbb{A}$ 是良态的。

若矩阵 $\mathbb{A}$ 是实对称正定的，则谱条件数为

$$\kappa_2(\mathbb{A}) = \frac{\lambda_{\max}}{\lambda_{\min}},$$

其中 $\lambda_{\max}$ 和 $\lambda_{\min}$ 为最大特征值和最小特征值。这个量的数值计算是比较困难的，但是它却比较适宜于理论分析。

论题 1.11. 由定义可知，条件数具有如下的基本性质：

1. $\kappa(\mathbb{A}) \geq 1$;
2. $\kappa(c\mathbb{A}) = \kappa(\mathbb{A}), 0 \neq c = const$;
3. $\kappa(\mathbb{A}^{-1}) = \kappa(\mathbb{A})$;
4. $\kappa(\mathbb{A}\mathbb{B}) \leq \kappa(\mathbb{A})\kappa(\mathbb{B})$.
5. 任意的两个条件数都是等价的。

说明 1.15. *Hilbert* 矩阵 $\mathbb{H}_n = (h_{ij})$ 是著名的病态矩阵，其中

$$h_{ij} = \frac{1}{i+j-1}.$$

[ix]这是一个相对概念下的陈述，通常同计算环境（或机器精度）有关。

相应的逆矩阵为 $\mathbb{H}_n^{-1}=(b_{ij})$，其中

$$b_{ij}=\frac{(-1)^{i+j}(n+i-1)!(n+j-1)!}{(i+j-1)!\Big[(i-1)!(j-1)!\Big]^2(n-i)!(n-j)!}.$$

在 *Matlab* 中，相应的命令是 *hilb()* 和 *invhilb()*.

其它的著名病态矩阵还有范得蒙矩阵等等。

定理 1.15. （*Kahan, 1996*）可逆矩阵 $\mathbb{A}$ 的病态程度描述了它同奇异矩阵集合的接近程度，因为

$$\min_{\delta\mathbb{A}}\left\{\frac{\|\delta\mathbb{A}\|_2}{\|\mathbb{A}\|_2}:\mathbb{A}+\delta\mathbb{A}\text{奇异}\right\}=\kappa_2^{-1}(\mathbb{A}).$$

换言之，条件数越大，矩阵越接近奇异。

证明：显然，利用 Banach 引理可知结论的左端必然大于或等于右端。下面，我们只需证明等号是可以取到的。令

$$\boldsymbol{y}=\frac{\mathbb{A}^{-1}\boldsymbol{x}}{\|\mathbb{A}^{-1}\boldsymbol{x}\|_2},\quad \delta\mathbb{A}=-\frac{\boldsymbol{x}\boldsymbol{y}^\top}{\|\mathbb{A}^{-1}\|_2},$$

其中 $\boldsymbol{x}$ 为单位长度向量，使得 $\|\mathbb{A}^{-1}\boldsymbol{x}\|_2=\|\mathbb{A}^{-1}\|_2$。至此，定理结论是不难验证的，因为 $(\mathbb{A}+\delta\mathbb{A})\boldsymbol{y}=0$。 □

✯ **说明 1.16.** 考虑行列式大小度量矩阵是否病态是很自然的。但是，不幸的是，它们几乎没有任何的关系。例如，矩阵

$$\mathbb{A}_n=\begin{bmatrix}1&-1&\cdots&-1\\0&1&\cdots&-1\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&1\end{bmatrix}$$

的行列式为 1，但 $\kappa(\mathbb{A}_n)=n2^{n-1}$。另一方面，一个非常良态的矩阵行列式可能非常小，例如对角线元素均为 10^{-1} 的 n 阶对角阵。

下面，让我们考虑一个简单的 2 阶矩阵 $\mathbb{A}$ 及其逆矩阵

$$\mathbb{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{bmatrix}, \quad \mathbb{A}^{-1} = \varepsilon^{-1} \begin{bmatrix} 1+\varepsilon & -1 \\ -1 & 1 \end{bmatrix}.$$

其中的 ε 是很小的正数。显然，矩阵 $\mathbb{A}$ 的谱条件数为

$$\kappa_2(\mathbb{A}) = (2+\varepsilon+\sqrt{4+\varepsilon^2})^2/(4\varepsilon) \approx 4/\varepsilon.$$

当 ε 靠近零时，矩阵 $\mathbb{A}$ 变得很病态；此时的逆矩阵 $\mathbb{A}^{-1}$ 关于 ε 的变化非常敏感。在高斯列主元消去过程中，相应的 LU 三角分解为

$$\mathbb{L}_\varepsilon = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad \mathbb{U}_\varepsilon = \begin{bmatrix} 1 & 1 \\ 0 & \varepsilon \end{bmatrix},$$

其中 $\mathbb{L}_\varepsilon$ 是良态的，而 $\mathbb{U}_\varepsilon$ 是病态的。我们不难证明（留为作业）：以 ε 为参数，$\mathbb{U}$ 的条件数同 $\mathbb{A}$ 的条件数处于同样的量级。这是一个普遍的现象吗？

1.4.2 摄动分析

下面，我们建立本章最重要的理论结果。

设线性方程组的系数矩阵 $\mathbb{A}$ 和右端项 $\boldsymbol{b}$ 分别发生了 $\delta\mathbb{A}$ 和 $\delta\boldsymbol{b}$ 的扰动，相应的解向量 $\boldsymbol{x}$ 将产生 $\delta\boldsymbol{x}$ 的扰动。经过相对简单的分析，我们有重要的扰动估计：设 $\|\mathbb{A}^{-1}\|\|\delta\mathbb{A}\| < 1$，则

1. 右端项有扰动: $\dfrac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \kappa(\mathbb{A})\dfrac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$.
2. 系数矩阵有扰动：$\dfrac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \dfrac{\kappa(\mathbb{A})\frac{\|\delta\mathbb{A}\|}{\|\mathbb{A}\|}}{1-\kappa(\mathbb{A})\frac{\|\delta\mathbb{A}\|}{\|\mathbb{A}\|}}$.

粗略地讲，解向量的相对改变量 $\delta\boldsymbol{x}$ 可被有效地控制，它几乎同系数矩阵的条件数 $\kappa(\mathbb{A})$ 成正比例。

在一般的理论框架下，我们很难改进上述扰动估计，因为不等式中的等号是可以取到的（请给出实例）。但是，大量的数值经验却表明，这个结论是非常保守的，因为真正遇到等号成立的概率并不高。譬如，让我们考虑一个简单的二阶线性方程组

$$\begin{bmatrix} \gamma & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \gamma \\ 1 \end{bmatrix}.$$

显见，当对角线元素发生改变时，真实扰动的放大率应该为 1。但是，问题的矩阵条件数显然同 γ 有关。若 $\gamma \gg 1$，上述扰动估计的缺陷是非常明显的，因为它给出的上界同真实误差相距甚远。

因此，摄动估计的改善是数值工作者一直努力的目标。目前，对于某些具有特定性质的现象方程组，许多可用的但略带风险的上界估计被相继提出。详细的处理方式，略。

1.4.3 精度分析

在数值实践中，如何判定计算结果的可靠性是一个非常重要的问题。常用的方法之一是利用同型的线性方程组，进行所谓的试算，即借用已知问题的真解，通过数值解和真解的比较，测算出结算结果中的可靠位数。另一种常用的方法是基于所谓的事后误差估计

$$\frac{\|\boldsymbol{x}_\star - \boldsymbol{x}_{\text{num}}\|}{\|\boldsymbol{x}_\star\|} \leq \kappa(\mathbb{A}) \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}, \tag{1.4.13}$$

其中的 $\boldsymbol{r} = \mathbb{A}\boldsymbol{x}_{\text{num}} - \boldsymbol{b}$ 是一个可信任[x]的可计算数值残量，$\boldsymbol{x}_{\text{num}}$ 是计算得到的数值解。

[x]通常，这个残量的计算结果是较为可信的，因为它的舍入误差积累要远远小于高斯消元过程中的误差积累。

为计算方便, 在事后误差估计 (1.4.13) 中, 我们通常取无穷范数。此时的关键问题是给出条件数 $\kappa_\infty(\mathbb{A})$ 的一个合理估计。为此, 我们需要分别估算 $\|\mathbb{A}\|_\infty$ 和 $\|\mathbb{A}^{-1}\|_\infty$. 前者的计算很简单, 而后者[xi]的计算却略为困难, 原因有二。首先, 我们不会真正计算出具体的逆矩阵 $\mathbb{A}^{-1}$; 其次, 逆矩阵的计算结果同样面临可靠性的问题。在 Matlab 中, 矩阵条件数可用命令 rcond() 给出。

✯ **说明 1.17.** 数值解的精度可通过迭代进行改进, 其基本思想就是对残量进行同类型的线性方程组修正。略。

1.5 列主元高斯消元法的数值稳定性分析

由于计算机位长总是有限的, 所以在数据存储和计算的过程中, 舍入误差是不可避免的, 对于理论上精确的高斯消元方法产生影响。本节简略介绍列主元高斯消元法的数值稳定性, 即它关于舍入误差的敏感程度。数值结果的相对误差不仅同系数矩阵的条件数有关, 它还严重依赖计算机的精度。详细内容可请见教科书。

[xi]通常, 我们会借用任意列范数 $\|\mathbb{B}\|_1$ 的一个估算方法。它是著名的最优化方法"盲人下山法"的一个应用, 已被 LAPACK 中所采用。因篇幅有限, 我们略过相关的理论解释, 仅给出其算法描述。

1. 取任意初始向量 $\boldsymbol{x}$, 满足 $\|\boldsymbol{x}\|_1 = 1$;
2. $\boldsymbol{w} = \mathbb{B}\boldsymbol{x}; \boldsymbol{v} = sign(\boldsymbol{w}); z = \mathbb{B}^T\boldsymbol{v}$;
3. 若 $\|\boldsymbol{z}\|_\infty \le \boldsymbol{z}^\top\boldsymbol{x}$, 则输出估算值 $\|\boldsymbol{w}\|_1$;
4. 否则, 令 $\boldsymbol{x} = \boldsymbol{e}_j$ 为 j 个标准单位向量, 其中的 j 由 $|z_j| = \|\boldsymbol{z}\|_\infty$ 确定。返回到第 2 步 ;

令 $\mathbb{B} = \mathbb{A}^{-\top}$, 我们可得 $\|\mathbb{B}\|_1 = \|\mathbb{A}^{-1}\|_\infty$。在调用上述估算方法时, 我们无需在第 2 步计算付出过高的代价。因为, 我们可以利用 $\mathbb{A}$ 的高斯消元过程中得到的 $\mathbb{L}\mathbb{U}$, 进行两个三角型线性方程组的快速求解。

1.5.1 浮点运算

让我们快速回顾计算机上的浮点数表示及其运算。在当前的大多数计算机上，浮点数[xii]通常表示为

$$f = \pm 0.d_1 d_2 \cdots d_t \times 2^J, \quad d_1 \neq 0,$$

其中整数 J 的取值范围决定了计算机的浮点数范围，t 是计算机的位长。位长决定了机器精度，即计算机器所能表示的最小正数 2^{1-t}。对于双精度浮点数（$t \approx 64$），最小的正数约为 10^{-16} 或 10^{-17} 量级。

请注意：所有浮点数仅仅构成实数域上的一个离散子集，对于四则运算（加减乘除）不再是封闭的。令 $\star$ 表示任意的一种四则运算，简单的分析可知浮点数运算满足估计

$$fl(a \star b) = (a \star b)(1+\delta), \quad \text{其中 } |\delta| \leq \vartheta = \begin{cases} \frac{1}{2} \times 2^{1-t}, & \text{舍入法}, \\ 2^{1-t}, & \text{截断法}. \end{cases}$$

简单应用这个估计，当 $n\vartheta$ 较小的时候，可知向量的内积满足

$$|fl(\boldsymbol{x}^\top \boldsymbol{y}) - \boldsymbol{x}^\top \boldsymbol{y}| \leq 1.01 n\vartheta |\boldsymbol{x}|^\top |\boldsymbol{y}|, \tag{1.5.14}$$

其中的 n 为向量的维数。类似地，矩阵的数乘、加法和乘法分别满足

$$fl(\alpha \mathbb{A}) = \alpha \mathbb{A} + \mathbb{E}, \qquad |\mathbb{E}| \leq \vartheta |\alpha \mathbb{A}|, \tag{1.5.15a}$$

$$fl(\mathbb{A} + \mathbb{B}) = \mathbb{A} + \mathbb{B} + \mathbb{E}, \quad |\mathbb{E}| \leq \vartheta |\mathbb{A} + \mathbb{B}|, \tag{1.5.15b}$$

$$fl(\mathbb{A}\mathbb{B}) = \mathbb{A}\mathbb{B} + \mathbb{E}, \qquad |\mathbb{E}| \leq 1.01 n\vartheta |\mathbb{A}||\mathbb{B}|, \tag{1.5.15c}$$

其中的 n 为矩阵的阶数，$\mathbb{E}$ 为最终的误差积累。这里的绝对值运算是指向量或矩阵的所有元素分别取绝对值，相应的不等式为对应元素的比较。

[xii] 实际上，$d_1 = 1$ 是不需要存储的。

上述分析工作称为一个向前误差分析过程。通常，相应的分析过程较为繁琐，依赖于具体的计算环境。

在数值分析工作中，我们更多地采用向后误差分析技术。换言之，我们认为所有的四则运算均是精确的，而将计算误差归结为初始数据的扰动误差所导致的。例如，(1.5.15a) 可表示为

$$fl(\alpha\mathbb{A}) = \alpha(\mathbb{A} + \mathbb{E}), \quad |\mathbb{E}| \leq \vartheta|\mathbb{A}|,$$

其中的 $\mathbb{E}$ 为初始矩阵的扰动。这种方法的优点是，它将浮点数的运算完全转化为实数域上的精确运算，便于后续的理论分析。

1.5.2 算法的舍入误差分析

考虑线性方程组 $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$，其中 $\mathbb{A} = (a_{ij})$ 是系数矩阵。利用向后误差分析技术，我们可将列主元高斯消元法的（受舍入误差影响的）数值解 $\boldsymbol{x} + \delta\boldsymbol{x}$ 视为扰动问题

$$(\mathbb{A} + \delta\mathbb{A})(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{b} \tag{1.5.16}$$

的精确解，其中 $\delta\boldsymbol{x}$ 是因舍入误差而产生的偏差。利用线性方程组的摄动理论，为估计 $\delta\boldsymbol{x}$，我们只需给出摄动矩阵 $\delta\mathbb{A}$ 的一个合理估计。

结合列主元高斯消元算法的具体实现过程，相应的矩阵分解（或者消元）过程可以表示为

$$\mathbb{P}\mathbb{A} + \mathbb{E} = \mathbb{L}\mathbb{U},$$

其中 $\mathbb{E} = (e_{ij})$ 是扰动矩阵，$\mathbb{P}$ 是置换矩阵，$\mathbb{L} = (\ell_{ij})$ 和 $\mathbb{U} = (u_{ij})$ 是计算机上真正存储的两个三角型矩阵。后续的计算过程可描述为两个三角形方程组

$$(\mathbb{L} + \mathbb{F})\boldsymbol{y} = \mathbb{P}\boldsymbol{b}, \quad (\mathbb{U} + \mathbb{G})(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{y}$$

的精确计算，其中 $\mathbb{F}=(f_{ij})$ 和 $\mathbb{G}=(g_{ij})$ 是相应的两个扰动矩阵。综上所述，可知 (1.5.16) 的扰动矩阵为

$$\delta\mathbb{A}=\mathbb{E}+\mathbb{P}(\mathbb{F}\mathbb{U}+\mathbb{L}\mathbb{G}+\mathbb{F}\mathbb{G}). \tag{1.5.17}$$

利用数学归纳法和向前误差分析技巧，可知事后误差估计中的三个扰动矩阵元素分别满足

$$|e_{ij}|\leq 2n\vartheta\max_{ijk}|a_{ij}^{(k)}|,$$
$$|f_{ij}|\leq\frac{6}{5}(n+1)\vartheta|\ell_{ij}|,\quad |g_{ij}|\leq\frac{6}{5}(n+1)\vartheta|u_{ij}|,$$

其中 ϑ 是机器精度，n 是矩阵的阶数，$a_{ij}^{(k)}$ 是在第 k 步高斯消元后计算机所保存的具体数据。

在列主元高斯消元过程中，$\mathbb{L}$ 的元素均以 1 为上界，相应的行范数满足 $\|\mathbb{L}\|_\infty\leq n$。为给出 $\|\mathbb{U}\|_\infty$ 的估计，我们定义主元增长因子

$$\boxed{\eta(\mathbb{A})=\frac{\max_{ijk}|a_{ij}^{(k)}|}{\max_{ij}|a_{ij}|}.}$$

因此，有 $\|\mathbb{U}\|_\infty\leq n\eta(\mathbb{A})\|\mathbb{A}\|_\infty$。从而，当 $n\vartheta$ 较小的时候，由 (1.5.17) 可知[xiii]

$$\|\delta\mathbb{A}\|_\infty\leq Cn^3\vartheta\eta(\mathbb{A})\|\mathbb{A}\|_\infty,$$

其中 $C\approx 10$ 为绝对常数。利用线性方程组的摄动理论可知：若相对扰动很小，最终的舍入误差满足

$$\frac{\|\delta\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty}\leq Cn^3\vartheta\eta(\mathbb{A})\kappa_\infty(\mathbb{A}), \tag{1.5.18}$$

[xiii]在下面的两个估计中，我们略去了关于 n 的低阶项。

其中 C 也是一个绝对常数。

结果 (1.5.18) 表明：对于大多数的线性方程组，当执行列主元高斯消元方法时，舍入误差所引起的相对误差可得到有效的控制。换言之，列主元高斯消元方法是一个可行的数值稳定的算法。

✯ **说明 1.18.** 结果 (1.5.18) 含有主元增长因子 $\eta(\mathbb{A})$。 在每次执行列主元消元之后，对角线右上方的元素绝对值至多放大 *2* 倍。因此，$\eta(\mathbb{A})$ 不会超过 2^{n-1}。在某些个例中，这个上限是可以取到的。但是，大量的数值经验表明，$\eta(\mathbb{A})$ 通常处于 $n^{2/3}$ 或 $n^{1/2}$ 的量级，并没有理论上的那么可怕。

第 2 章
线性方程组的迭代解法

当线性方程组的计算规模变得越来越庞大的时候，直接解法的数值应用将受到诸多限制。由于内存空间有限，相应的数据存储遇到严峻的挑战；与此同时，直接法的计算复杂度也是无法承受的。为此，迭代算法脱颖而出。相应的计算目标不再是经过有限步运算给出精确解，而是构造一个简单快捷的计算过程，自动生成一个快速收敛到精确解的向量序列。通常，迭代方法都具有如下的优点：其一，它无需改变线性方程组的任何（非零）数据，适用于大规模稀疏矩阵的数据存储方式。其二，它的数值编程实现非常容易。由于大多数的迭代算法均基于内积运算（包含矩阵同向量的乘法，向量同向量的内积），在本质上就具有所谓较高的 BLAS-2 机制。

2.1 迭代法的基本理论

迭代方法的基本框架是，通过简单的计算规则

$$\boldsymbol{x}_k = \boldsymbol{f}_k(\boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}, \ldots, \boldsymbol{x}_{k-r}), \quad k \geq r, \tag{2.1.1}$$

自动生成一个向量序列 $\{\boldsymbol{x}_k\}_{k=0}^{\infty}$，其中 $\{\boldsymbol{x}_k\}_{k=0}^{r-1}$ 是格式启动的初值，需要由用户给出相应的猜测值。我们的目标是 $\boldsymbol{x}_k$ 能否快速地收敛到线性方程组的精确解。

通常，称 (2.1.1) 是一个 r 阶迭代方法，称 $\boldsymbol{f}_k$ 为相应的 r 阶迭代函数。若 $\boldsymbol{f}_k$ 同迭代步数 k 无关，则称这个迭代方法是定常的；否则，称其为非定常的。

定义 2.1. 若线性方程组 $\mathbb{A}\boldsymbol{x}=\boldsymbol{b}$ 的精确解 $x_\star$ 是迭代公式 (2.1.1) 的稳态解，则称相应的迭代方法是完全相容的。

在本讲义中，默认所有的迭代方法是完全相容的。

2.1.1 一阶迭代方法

为简单起见，我们重点关注线性方程组 $\mathbb{A}\boldsymbol{x}=\boldsymbol{b}$ 的一阶迭代方法。通常，它具有如下两种表示形式：

$$\boldsymbol{x}_k=\boldsymbol{x}_{k-1}+\mathbb{H}_k(\boldsymbol{b}-\mathbb{A}\boldsymbol{x}_{k-1})=\boldsymbol{x}_{k-1}-\mathbb{H}_k\boldsymbol{r}_{k-1}, \tag{2.1.2a}$$

$$\boldsymbol{x}_k=\mathbb{G}_k\boldsymbol{x}_{k-1}+\boldsymbol{g}_k, \tag{2.1.2b}$$

其中 $\mathbb{H}_k$ 称为预处理矩阵，$\mathbb{G}_k$ 称为迭代矩阵。

称 $\boldsymbol{r}_k=\mathbb{A}\boldsymbol{x}_k-\boldsymbol{b}$ 为第 k 步迭代的残量；若 $\boldsymbol{r}_k=0$，则相应的数值解 $\boldsymbol{x}_k$ 就是精确解 $\boldsymbol{x}_\star=\mathbb{A}^{-1}\boldsymbol{x}$。

论题 2.1. 上述两个迭代公式是等价的。若 $\mathbb{G}_k=\mathbb{I}-\mathbb{H}_k\mathbb{A}$ 和 $\boldsymbol{g}_k=\mathbb{H}_k\boldsymbol{b}$，则我们可由第二种表述得到第一种表述。

因此，迭代算法的研究重点就是，构造漂亮的迭代矩阵或预处理矩阵，使得向量序列 $\boldsymbol{x}_k$ 快速地收敛到问题的精确解。

2.1.2 收敛性分析

准备知识

我们需要介绍向量序列和矩阵序列的基础概念和常用结论。其中，相应的收敛定义是非常直接的，即所有的对应分量都是收敛的。若利用范数作为度量工具，收敛的定义更为简单。

定义 2.2. $\lim_{k\to\infty}\boldsymbol{x}_k=\boldsymbol{x}\Leftrightarrow\lim_{k\to\infty}\|\boldsymbol{x}_k-\boldsymbol{x}\|=0.$

定义 2.3. $\lim_{k\to\infty}\mathbb{A}_k=\mathbb{A}\Leftrightarrow\lim_{k\to\infty}\|\mathbb{A}_k-\mathbb{A}\|=0.$

对于一个矩阵序列（或者矩阵级数）是否收敛，矩阵的范数和谱半径是重要的分析工具。常用的主要结论有

定理 2.1. $\lim_{k\to\infty}\mathbb{A}^k=\mathbb{O}\Leftrightarrow\varrho(\mathbb{A})<1.$

定理 2.2. 矩阵级数 $\sum_{k=0}^{\infty}\mathbb{B}^k$ 收敛的充要条件是 $\varrho(\mathbb{B})<1$，且

$$\sum_{k=0}^{\infty}\mathbb{B}^k=(\mathbb{I}-\mathbb{B})^{-1}.$$

若存在某个范数使得 $\|\mathbb{B}\|<1$，则矩阵级数 $\sum_{k=0}^{\infty}\mathbb{B}^k$ 也是收敛的。相应的余项满足

$$\left\|\sum_{k=m+1}^{\infty}\mathbb{B}^k\right\|\leq\sum_{k=m+1}^{\infty}\|\mathbb{B}\|^k\leq\frac{\|\mathbb{B}\|^{m+1}}{1-\|\mathbb{B}\|}.\tag{2.1.3}$$

这个性质同绝对收敛的幂级数性质保持形式上的一致，故而我们可将其视为有限项的三角不等式的推广。

收敛性判定

记 $\boldsymbol{e}_k=\boldsymbol{x}_k-\boldsymbol{x}_\star$ 为迭代方法 (2.1.2) 的第 k 步（迭代）误差，其中 $\boldsymbol{x}_\star$ 是线性方程组 $\mathbb{A}\boldsymbol{x}=\boldsymbol{b}$ 的精确解。由于迭代方法是完全相容的，我们可知相应的误差方程

$$\boldsymbol{e}_k=\mathbb{G}_k\boldsymbol{e}_{k-1},\quad\text{或}\quad\boldsymbol{e}_k=(\mathbb{I}-\mathbb{H}_k\mathbb{A})\boldsymbol{e}_{k-1}.\tag{2.1.4}$$

若对**任意**的初始向量 $\boldsymbol{e}_0$，均有 $\lim_{k\to\infty}\boldsymbol{e}_k=\boldsymbol{0}$，则称这个迭代方法是收敛的。否则，称其是发散的。

定理 2.3. 迭代方法 (2.1.2) 收敛等价于迭代矩阵的无穷乘积是零矩阵，即

$$\lim_{k\to\infty}\Pi_{m=1}^{k}\mathbb{G}_m=\lim_{k\to\infty}\Pi_{m=1}^{k}(\mathbb{I}-\mathbb{H}_m\mathbb{A})=\mathbb{O}.$$

若迭代矩阵 $\mathbb{G}_k\equiv\mathbb{G}$ 或预处理矩阵 $\mathbb{H}_k\equiv\mathbb{H}$（即算法是定常的），则结果可以简化:

$$\varrho(\mathbb{G})<1$$

是一阶定常迭代方法收敛的充要条件，而 $\|\mathbb{G}\|<1$ 只是一阶定常迭代方法收敛的充分条件。

收敛速度的刻画与估计

通常，误差向量 $\boldsymbol{e}_k$ 趋于零的快慢程度，决定一个迭代算法的优劣。下面，我们以一阶定常迭代方法 $\boldsymbol{x}_k=\mathbb{G}\boldsymbol{x}_{k-1}+\boldsymbol{g}$ 为例。此时，我们有误差估计：

$$\|\boldsymbol{e}_k\|\leq\|\mathbb{G}^k\|\|\boldsymbol{e}_0\|. \tag{2.1.5}$$

这是一个非常保守的估计。但是，这个结果通常是不可改善的，因为它的上限是可以实现的。

✍ **论题 2.2.** 利用保守估计 (2.1.5)，我们通常按误差下降的平均效应，来刻画迭代误差趋零的快慢程度。

1. 平均收敛速度 $R_k(\mathbb{G})=-\frac{1}{k}\ln\|\mathbb{G}^k\|$.
2. 渐近收敛速度 $R_\infty(\mathbb{G})=\lim_{k\to\infty}R_k(\mathbb{G})=-\ln\varrho(\mathbb{G})$.

这些概念是上世纪五六十年代提出的，譬如渐近收敛速度是由 *Young* 在 *1954* 年给出的。

通常，我们以渐近收敛速度做为迭代算法基本评判指标。**要达到指定的误差要求，所需的最小迭代次数同渐近收敛速度的倒数成正比**。

因此说，关于迭代算法的收敛快慢，通常的评判标准是基于其最差的数值表现。请注意：真实迭代误差的下降速度，会因为初始向量的选取，而产生明显的差异。

❀ **思考 2.1.** 事实上，上述两个关于收敛速度的概念均是基于所谓的平均意义，它们同迭代误差在每一步的真实变化速度，还是具有一定的细节差距。为深入理解上述概念，请比较两个矩阵谱范数 $\|\mathbb{A}^m\|_2$ 和 $\|\mathbb{B}^m\|_2$ 关于 m 的发展过程，其中

$$\mathbb{A}=\begin{bmatrix}\alpha & 4\\ 0 & \alpha\end{bmatrix},\quad \mathbb{B}=\begin{bmatrix}\alpha & 0\\ 0 & \beta\end{bmatrix},\quad 0<\alpha<\beta<1.$$

显然，$\mathbb{A}$ 的谱半径要更小一点。请观察：若 α 靠近 *1*，当 m 较小的初始阶段，是否会发生 $\|\mathbb{A}^m\|_2>\|\mathbb{B}^m\|_2$ 的现象呢？换言之，初始阶段的误差衰减同渐近表现可以截然不同。

❀ **思考 2.2.** 证明：对于任意范数，均有

$$\lim_{k\to\infty}\|\mathbb{G}^k\|^{1/k}=\varrho(\mathbb{G}).$$

这个证明可在任何一本教科书中找到。

由于 $\|\mathbb{G}^k\|$ 和 $\varrho(\mathbb{G})$ 两个量难以直接计算，上述两个收敛速度概念仅仅具有理论价值，无法用于当前迭代误差的估算。但是，迭代矩阵的某些范数 $\|\mathbb{G}\|$ 是可直接计算的，具有重要的应用价值。

定理 2.4. 若 $\|\mathbb{G}\|<1$，则一阶定常迭代方法 $\boldsymbol{x}_k=\mathbb{G}\boldsymbol{x}_{k-1}+\boldsymbol{g}$ 是收敛的。相应的迭代误差满足

1. 先验误差估计: $\|\boldsymbol{e}_k\| \leq \|\mathbb{G}\|^k \|\boldsymbol{e}_0\|$;
2. 后验误差估计 (I): $\|\boldsymbol{e}_k\| \leq \frac{\|\mathbb{G}\|}{1-\|\mathbb{G}\|} \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$;
3. 后验误差估计 (II): $\|\boldsymbol{e}_k\| \leq \frac{\|\mathbb{G}\|^k}{1-\|\mathbb{G}\|} \|\boldsymbol{x}_1 - \boldsymbol{x}_0\|$,

其中 $\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$ 称为相邻误差，是一个可计算的量。

✯ **说明 2.1.** 在先验误差估计中，右端的上界是不可计算的；而在后验误差估计中，右端的上界是可计算的。显然，这些估计都是相当保守的估计，实际误差可能远远小于这些上界估计。

2.1.3 停机标准

迭代过程何时停止是个重要的问题。我们当然希望迭代停止的时候，数值误差能够满足用户要求，例如[i]

$$\|\boldsymbol{e}_k\| \leq \mathcal{E}, \tag{2.1.6}$$

其中 $\mathcal{E}$ 是用户给出的停机指标。但是，这中策略通常仅仅作为理论研究 (或数值实验) 中的一种停机标准，因为问题的真解是未知的，使得迭代误差是一个无法真正计算的量。

因此，在实际的数值计算中，我们更多地利用下面三种简便的近似停机准则，尽管它们不保证 (2.1.6) 一定成立。它们分别是：

1. 残量准则：$\|\boldsymbol{r}_k\| \leq \mathcal{E}$;
2. 相邻误差准则：$\delta_k \equiv \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\| \leq \mathcal{E}$;
3. 后验误差停机准则：$\delta_k^2/(\delta_{k-1} - \delta_k) \leq \mathcal{E}$.

[i] 这是绝对误差。当然，我们也可使用相对误差。

其中, 第三种停机标准来源于后验误差估计 (I) 和关于 $\|\mathbb{G}\|$ 的一个估算。我们通常取范数为最大模范数或者欧几里得范数。

✯ **说明 2.2.** 我们要强调: 舍入误差也会对迭代格式的收敛性, 或者停机判断产生影响, 特别是待解的线性方程组非常病态的时候。

当我们应用后两个停机准则的时候, 用户还要小心出现所谓的"假停机"现象。例如, 当利用具有 *6* 位有效数字的 *10* 进制计算机求解一个简单的实例

$$\boldsymbol{x}_k = \begin{bmatrix} 0 & 1-10^{-6} \\ 1-10^{-6} & 0 \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} 10^{-6} \\ 10^{-6} \end{bmatrix},$$

其迭代矩阵的范数接近 1。设 $\mathcal{E} = 10^{-5}$。若初值为 $\boldsymbol{x}_0 = (0.1, 0.1)^\top$, 则每次迭代仅对每个分量产生 10^{-6} 的增量, 使得 $\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_\infty = 10^{-6} < \mathcal{E}$, 出现所谓的假停机。

在本课程中, 我们重点关注迭代方法的方法误差。因篇幅有限, 我们略去关于舍入误差的分析。

2.2 古典迭代算法

Jacobi 方法和 Gauss-Seidel 方法是重要的古典迭代算法, 具有非常简单的实现方式。它们都是不含参数的一阶定常迭代方法。

2.2.1 基本算法

Jacobi 方法和 Gauss-Seidel 方法分别基于同步更新策略和异步更新策略。它们的实现过程非常类似。算法在迭代过程中, 每个分量的更新方式如下：

1. J 方法： $\boldsymbol{x}_k^{(i)} = \dfrac{1}{a_{ii}}\Big[\boldsymbol{b}^{(i)} - \sum\limits_{j\neq i} a_{ij}\boldsymbol{x}_{k-1}^{(j)}\Big]$;

2. GS 方法： $\boldsymbol{x}_k^{(i)} = \dfrac{1}{a_{ii}}\Big[\boldsymbol{b}^{(i)} - \sum\limits_{j<i} a_{ij}\boldsymbol{x}_{k}^{(j)} - \sum\limits_{j>i} a_{ij}\boldsymbol{x}_{k-1}^{(j)}\Big]$;

两者相比，Jacobi 方法的并行性更好，但它需要两组工作单元，记录解向量。

1. Jacobi 方法是由 Jacobi (1845 年) 提出的。而后的相关工作还包括 Geiringer (1945) 的同步位移法，以及 Killer (1958) 的 Richardson 方法。从物理的角度来讲，Jacobi 方法也称为阻尼法。

2. 著名的 Gauss-Seidel 方法最初由 Gauss (1822) 提出, 用于由测地问题而产生的最小二乘问题法方程组的求解。而后，此方法由 Seidel (1874) 再次提出, 却遭到弃用。Von. Misers 和 Pollaczek-Geiringer (1949) 给出了最早的理论分析，进而使得 Gauss-Seidel 方法重新得到重视和发展。

2.2.2 矩阵分裂方式

通常，一阶定常迭代方法可基于所谓的“矩阵分裂”方式

$$\mathbb{A} = \mathbb{Q} - \mathbb{R},$$

其中 $\mathbb{Q}$ 是逼近 $\mathbb{A}$ 的一个容易求逆的预处理矩阵。利用由此导出的等价不动点方程 $\boldsymbol{x} = \mathbb{Q}^{-1}(\mathbb{R}\boldsymbol{x} + \boldsymbol{b})$，我们可以给出相应的迭代方法

$$\boldsymbol{x}_k = \mathbb{Q}^{-1}(\mathbb{R}\boldsymbol{x}_{k-1} + \boldsymbol{b}). \tag{2.2.7}$$

因此，这种迭代方法也成为不动点迭代方法。

设系数矩阵 $\mathbb{A}$ 按对角线部分，严格[ii]下三角部分和严格上三角部分，进行划分，可得如下的形式：

$$\mathbb{A} = \mathbb{D} - \mathbb{DL} - \mathbb{DU}. \tag{2.2.8}$$

若选取对角线部分 $\mathbb{D}$，或下三角部分 $\mathbb{D} - \mathbb{DL}$ 作为预处理矩阵，相应的矩阵分裂可形成上述两个古典算法。Jacobi 迭代矩阵和 GS 迭代矩阵分别是

$$\mathbb{B} = \mathbb{I} - \mathbb{D}^{-1}\mathbb{A}, \quad \mathbb{T}_1 = (\mathbb{I} - \mathbb{L})^{-1}\mathbb{U}. \tag{2.2.9}$$

算法收敛的充要条件是相应的迭代矩阵谱半径小于一。

✯ **说明 2.3.** 迭代矩阵的谱与线性方程组的行排序有关，譬如，我们用 J 方法求解如下两个同解的线性方程组

$$\begin{bmatrix} 3 & -10 \\ 9 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -7 \\ 5 \end{bmatrix}, \qquad \begin{bmatrix} 9 & -4 \\ 3 & -10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ -7 \end{bmatrix}.$$

请计算 J 迭代矩阵的谱半径，并指出它们的 J 迭代是否收敛？这隐含地说明预处理技术的必要性。详细的预处理思想容后介绍。

2.2.3 收敛性分析

通常，Jacobi 方法和 Gauss-Seidel 方法的收敛性没有任何的关系。但是，若系数矩阵具有某些特殊的结构，GS 方法将比 J 方法收敛得更快。

以 Jacobi 方法的迭代矩阵为起点，有

定理 2.5. *若 $\|\mathbb{B}\|_\infty < 1$，则 GS 方法也收敛，并比 J 方法更快。*

定理 2.6. *若 $\|\mathbb{B}\|_1 < 1$，则 GS 方法也收敛。*

[ii] 这里的严格是指对角线元素也等于零。

以系数矩阵本身的特点出发，我们有

定理 2.7. *若系数矩阵 $\mathbb{A}$ 是对角占优的，即它是*

1. *严格对角占优矩阵：* $\forall i, |a_{ii}| > \sum_{j\neq i} |a_{ij}|$;
2. *或者弱对角占优不可约矩阵：* $\forall i, |a_{ii}| \geq \sum_{j\neq i} |a_{ij}|$，*且至少有一个不等式是严格成立的；所谓的不可约是指问题不能分解出一个独立的小规模问题。*

则 J 方法和 GS 方法均收敛。

定理 2.8. *设系数矩阵 $\mathbb{A}$ 是对称正定的，则 GS 方法必然收敛。此时，J 方法收敛的充分必要条件是 $2\mathbb{D} - \mathbb{A}$ 也要对称正定。显然，它比 GS 方法的要求更高。*

✯ **说明 2.4.** 上面四个定理的证明过程充分展示了迭代法收敛性分析的基本技巧：**范数估计方法**或**特征值分析方法**。

✯ **说明 2.5.** 通常，矩阵 $\mathbb{A}$ 对角占优的强度可用

$$\min_i \left[|a_{ii}| - \sum_{j\neq i} |a_{ij}| \right]$$

来衡量。但是，这个量的大小同算法的收敛速度并无实质的联系。例如，让我们观察两个系数矩阵

$$\mathbb{A}_1 = \begin{bmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{bmatrix}, \quad \mathbb{A}_2 = \begin{bmatrix} 1 & -\frac{3}{4} \\ -\frac{1}{4} & 1 \end{bmatrix}.$$

虽然 $\mathbb{A}_1$ 的对角占优性更强一些，但是它的 *J* 方法在收敛速度方面却略慢一些。因为，它们的 *Jacobi* 迭代矩阵满足 $\varrho(\mathbb{B}_1) > \varrho(\mathbb{B}_2)$.

2.3 逐次超松弛方法

在迭代算法的发展历史上，逐次超松弛[iii]方法具有非常重要的地位。因为它开辟了迭代算法研究的新视野：适当加权平均相邻的迭代解，可以使迭代算法的收敛速度，获得本质上的突破。

SOR 方法是以古典的 GS 方法为蓝本。理论分析和数值实践均表明权重的选取非常重要。下面，我们给出相关的基本内容。

✍ **论题 2.3.** *每个分量的更新方式是*

$$\boxed{\boldsymbol{x}_k^{(i)} = (1-\omega)\boldsymbol{x}_{k-1}^{(i)} + \frac{\omega}{a_{ii}}\Big[\boldsymbol{b}^{(i)} - \sum_{j<i} a_{ij}\boldsymbol{x}_k^{(j)} - \sum_{j>i} a_{ij}\boldsymbol{x}_{k-1}^{(j)}\Big],}$$

其中 ω 称为松弛因子。显然，当 $\omega = 1$，SOR 方法就是 GS 算法。相应的迭代矩阵是

$$\mathbb{T}_\omega = (\mathbb{I} - \omega\mathbb{L})^{-1}[(1-\omega)\mathbb{I} + \omega\mathbb{U}]. \tag{2.3.10}$$

❀ **思考 2.3.** *请指出 SOR 迭代方法所对应的矩阵分裂形式。*

为保证 SOR 方法的收敛性，参数 ω 需满足适当的条件。

定理 2.9. *SOR 方法收敛的必要条件是 $0 < \omega < 2$.*

定理 2.10. *若系数矩阵 $\mathbb{A}$ 对称正定，则 $0 < \omega < 2$ 是 SOR 方法收敛的充分必要条件。*

证明：这是迭代矩阵特征值分析方法的典型例子。见教科书。 □

数值实验表明：松弛因子 ω 影响 SOR 方法的收敛速度。在某些情况，我们可以找到最佳的松弛因子 ω_{opt}，使得 SOR 方法的收敛速度获

[iii] Successive over-relax method=SOR.

得显著的提升。关于最佳松弛因子的研究，曾是 SOR 算法的一个研究亮点；相关的研究工作极大地促进了迭代方法的发展，特别是上世纪 80 年代之前的研究。

✎ **论题 2.4.** 关于最佳松弛因子的研究，强烈依赖于线性方程组系数矩阵的非零元素分布结构，相关的概念有“相容次序”和“性质 $\boldsymbol{A}$”等。详细的内容请见教科书。常用的结论有

1. 三对角阵或块三对角阵都是具有相容次序的。
2. 若矩阵具有相容次序，则它必具有性质 $\boldsymbol{A}$。
3. 若矩阵具有性质 $\boldsymbol{A}$，它不一定具有相容次序，但经过适当的行列重排后便可具有相容次序。

为避免过分纠结于这些概念之间的详细讨论，我们不妨假设稀疏矩阵具有典型的矩阵结构，即所谓的性质 $\boldsymbol{A}$。

例如，在椭圆方程的差分离散过程中，最终的线性代数方程组常常满足如下的情形：未知量归属于两个互不相交的集合，使得每个未知量同其所属集合中的其他未知量不发生任何关联[iv]。这样的结构称为性质 A。换言之，系数矩阵满足

$$\mathbb{P}\mathbb{A}\mathbb{P}^{\top} = \begin{bmatrix} \mathbb{D}_1 & \mathbb{H} \\ \mathbb{K} & \mathbb{D}_2 \end{bmatrix}, \tag{2.3.11}$$

其中 $\mathbb{P}$ 是某个置换阵，$\mathbb{D}_1$ 和 $\mathbb{D}_2$ 是对角阵。若系数矩阵具有 (2.3.11) 右侧的结构，相应的未知量编号方式称为红黑（或棋盘）编号原则。当然，每个代数方程在线性方程组中的排列次序，也按分组方式进行了相应的调整。

[iv] 所谓两个变量具有关联，是指它们同时出现在一个线性方程中。

若系数矩阵 $\mathbb{A}$ 具有性质 A（甚至更一般的相容次序），J 方法同 SOR 方法的迭代矩阵特征值具有特殊的关系。为简化理论分析的复杂程度[v]，我们直接假设系数矩阵 $\mathbb{A}$ 恰好具有 (2.3.11) 右侧的结构，相应的 J 迭代矩阵[vi]是

$$\mathbb{B} = \mathbb{L} + \mathbb{U} = \begin{bmatrix} \mathbb{O} & \mathbb{H} \\ \mathbb{K} & \mathbb{O} \end{bmatrix}.$$

定理 2.11. *设 λ 是 SOR 迭代矩阵 $\mathbb{T}_\omega$ 的特征值，而 μ 是 J 迭代矩阵 $\mathbb{B}$ 的特征值。它们具有如下的对应关系：*

$$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2. \tag{2.3.12}$$

请注意：此时的 μ 和 $-\mu$ 是成对出现的。

证明：利用相似变换的分块矩阵表述

$$\begin{bmatrix} \mathbb{I} & \\ & \alpha^{-1}\mathbb{I} \end{bmatrix} \begin{bmatrix} \mathbb{D}_1 & \alpha^{-1}\mathbb{H} \\ \alpha\mathbb{K} & \mathbb{D}_2 \end{bmatrix} \begin{bmatrix} \mathbb{I} & \\ & \alpha\mathbb{I} \end{bmatrix} = \begin{bmatrix} \mathbb{D}_1 & \mathbb{H} \\ \mathbb{K} & \mathbb{D}_2 \end{bmatrix}, \tag{2.3.13}$$

对于 J 方法和 SOR 方法，我们分别计算相应的迭代矩阵特征值，即可得到定理结论。参见教科书。 □

定理 2.12. *设 n 阶矩阵 $\mathbb{B}$ 的所有特征值均可表达为*

$$\mu_j = \alpha_j + \sqrt{-1}\beta_j,$$

其中 α_j 和 β_j 均为实数。若存在正数 D，使得

$$\alpha_j^2 + \beta_j^2/D < 1, \quad j = 1, 2, \ldots, n,$$

[v] 繁琐的证明可参阅教课书

[vi] 为简单起见，非对角块部分仍采用了原有符号。

则当 $0 < \omega < 2/(1+D)$ 时，*SOR* 迭代方法是收敛的。

因此，若矩阵 $\mathbb{B}$ 的特征值 μ_j 均为实数，则 *SOR* 迭代方法收敛的充分必要条件是

$$\boxed{0 < \omega < 2, \text{ 且 } \varrho(\mathbb{B}) < 1.}$$

证明：注意到 (2.3.12)，利用实系数二次方程的根与系数的关系。□

考虑一个简单的情形：J 迭代矩阵 $\mathbb{B}$ 的特征值均为实数，且按模小于 1。此时，SOR 方法最佳松弛因子的设置问题可以容易地解决。

✍ **论题 2.5.** 首先，让我们固定 $\mathbb{B}$ 的某个特征值 μ。考虑直线

$$y = \frac{\lambda + \omega - 1}{\omega},$$

同抛物线

$$y^2 = \lambda |\mu|^2,$$

在 (λ, y) 平面上的交点，其中 ω 为参数。不妨设它们的交点存在[vii]，即相应的横坐标 $\lambda_1(\omega)$ 和 $\lambda_2(\omega)$ 均为实数。它们对应二次方程 (2.3.12) 的根，是 *SOR* 迭代矩阵 $\mathbb{T}_\omega$ 的特征值。当 ω 变化时，要使 $\max_{i=1,2} |\lambda_i(\omega)|$ 达到最小，直线同抛物线必须相切，即二次方程 (2.3.12) 的判别式为零，对应的

$$\omega = \frac{2}{1 + \sqrt{1 - \mu^2}}, \quad \max_{i=1,2} |\lambda_i(\omega)| = |\omega - 1|. \tag{2.3.14}$$

显然，当 $|\mu|$ 从小变大时，相应的 ω 随之增加。此时，抛物线的开口变宽，相应的切点位置会随之右移，对应的切线以 $(1,1)$ 为中心逆时针旋转。在这个过程中，切线同原有的窄口抛物线是不相交的，*SOR* 迭代矩阵的谱半径对应最后的切点位置。

[vii]若交点不存在，由定理 2.11 可知，(2.3.12) 存在共轭复根，满足 $|\lambda_i(\omega)| \equiv |\omega - 1|$。因此，我们只需考虑切点位置同 μ 的关系。

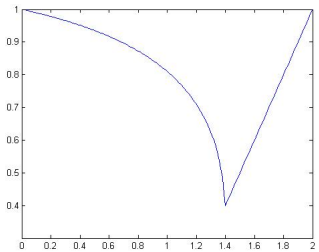
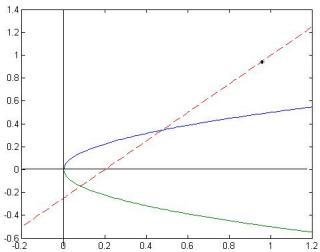

图 2.3.1: 左：谱半径函数 $\varrho(\mathbb{T}_\omega)$. 右：直线与抛物线的交点。

因此，最佳松弛因子对应 $\mu = \rho(\mathbb{B})$ 时的切线参数，即

$$\boxed{\omega_{\text{opt}} = \frac{2}{1+\sqrt{1-\varrho^2(\mathbb{B})}},}$$

相应的 *SOR* 迭代矩阵谱半径为

$$|\omega_{\text{opt}} - 1| = \frac{1-\sqrt{1-\varrho(\mathbb{B})^2}}{1+\sqrt{1-\varrho(\mathbb{B})^2}}. \tag{2.3.15}$$

事实上，ω_{opt} 对应谱半径函数 $f(\omega) \equiv \varrho(\mathbb{T}_\omega)$ 的不可微点；参见下面左侧的插图。关于谱半径函数的具体表达式，请参见教科书。

✯ **说明 2.6.** 注意到谱半径函数在不可微点的左导数为无穷大，实际计算时通常会偏大选取松弛因子。

✎ **论题 2.6.** 最佳松弛因子的选取方法: 先取一个偏大的 $\omega \in (0,2)$, 然后执行几步 *SOR* 迭代过程。由幂法可知，在适当的条件下，迭代矩阵 $\rho(\mathbb{T}_w)$ 的绝对值最大特征值可近似表示为

$$\rho(\mathbb{T}_w) \approx \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_\infty}{\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_\infty}.$$

然后，我们利用 (2.3.12) 确定 *Jacobi* 迭代矩阵 $\mathbb{B}$ 的谱半径，即

$$\rho(\mathbb{B}) = \frac{\rho(\mathbb{T}_w) + \omega - 1}{\omega[\rho(\mathbb{T}_w)]^{1/2}}.$$

由这个较为精确的估计 $\rho(\mathbb{B})$，我们可以给出最佳松弛因子的近似。上述过程重复多次，我们可以逐步地改进 ω_{opt} 的精确度。

✎ **论题 2.7.** 对应这个最佳松弛因子 ω_{opt}，逐次超松弛算法的迭代速度可获得本质上的提升。

教科书给出了抽象的分析过程和普适结果。下面，我们仅仅给出一个具体的实例，直接展现这个事实。让我们考虑 (6.0.3) 给出的线性方程组，其系数矩阵 $\mathbb{A}_n$ 是一个块三对角矩阵。经过简单的行列重排，我们不难发现 $\mathbb{A}_n$ 满足性质 A。对于这个特殊矩阵，我们可以计算出三对角矩阵 $\mathbb{T}_n$ 的特征值和特征向量分别为

$$\begin{aligned}
\lambda_\kappa &= 2\left(1 - \cos\frac{\kappa\pi}{n+1}\right), \\
\mathbf{v}_\kappa &= \sqrt{\frac{2}{n+1}}\left(\sin\frac{\kappa\pi}{n+1}, \sin\frac{2\kappa\pi}{n+1}, \cdots, \sin\frac{n\kappa\pi}{n+1}\right)^\top,
\end{aligned}$$

其中 $\kappa = 1, 2, \ldots, n.$ 由矩阵的 Kronecker 分裂形式[viii]

$$\mathbb{A}_n = \mathbb{T}_n \otimes \mathbb{I}_n + \mathbb{I}_n \otimes \mathbb{T}_n,$$

可知 $\mathbb{A}_n$ 的特征值为

$$\lambda_{pq} = \lambda_p + \lambda_q,$$

其中的指标 p 和 q 均遍历 1 到 n 的所有自然数。注意到

$$\mu_{pq} = \frac{1}{4}(\lambda_p + \lambda_q - 4)$$

[viii] 设 $\mathbb{A} = (a_{ij})$ 为 m 阶方阵，$\mathbb{B} = (b_{ij})$ 为 n 阶方阵，则 $\mathbb{A} \otimes \mathbb{B} = (a_{ij}\mathbb{B})$ 为 mn 阶方阵。设 $\mathbb{A}$ 的特征值为 $\{\nu_i\}_{i=1:m}$，$\mathbb{B}$ 的特征值为 $\{\sigma_j\}_{j=1:n}$，则 $\mathbb{A} \otimes \mathbb{B}$ 的特征值为 $\{\nu_i\sigma_j\}_{i=1:m}^{j=1:n}$.

这蕴含三种迭代方法的渐近收敛速度分别为

$$R_\infty(\mathbb{B}) = -\ln \varrho(\mathbb{B}) = -\ln\cos h\pi \sim \frac{1}{2}\pi^2 h^2, \tag{2.3.16a}$$

$$R_\infty(\mathbb{L}_1) = -2\ln \varrho(\mathbb{B}) \sim \pi^2 h^2, \tag{2.3.16b}$$

$$R_\infty(\mathbb{L}_{\text{opt}}) = -\ln \frac{1-\sin(h\pi)}{1+\sin(h\pi)} \sim 2h\pi, \tag{2.3.16c}$$

其中 $h = 1/(n+1)$。上面结果是由 Franke 最早给出的，而后由 Young 推广到更一般的情形，例如具有性质 A 的系数矩阵。详细的内容可参见教科书。

✯ **说明 2.7.** *类似于逐次超松弛方法，我们还可以构造出所谓的块 SOR 方法和对称 SOR 方法。详略。*

2.4 迭代加速方法

关于 SOR 方法的收敛研究表明：我们可以挖掘和利用已有的计算信息，提高迭代序列的收敛速度。本节给出两种常用的迭代加速方法，特别是半迭代方法。

2.4.1 外推方法

外推方法是 SOR 方法的直接推广。设

$$\boldsymbol{x}_k = \mathbb{G}\boldsymbol{x}_{k-1} + \boldsymbol{g} \tag{2.4.17}$$

为基础迭代算法，对于相邻的两个数值解进行适当的加权平均，我们可以建立新的迭代方法

$$\boxed{\boldsymbol{x}_k = \gamma(\mathbb{G}\boldsymbol{x}_{k-1} + \boldsymbol{g}) + (1-\gamma)\boldsymbol{x}_{k-1},}$$

其中 γ 是某个固定的权重。

❀ **思考 2.4.** 设迭代矩阵 $\mathbb{G}$ 是实对称的，请找到最优的参数 γ，使得外推方法所对应的迭代矩阵

$$\gamma\mathbb{G} + (1-\gamma)\mathbb{I}$$

具有尽可能小的谱半径。换言之，使外推方法的收敛速度达到最快。

2.4.2 半迭代方法

半迭代方法可视为外推思想的极致推广。此时，我们想要充分利用手中的全部计算资源，希望通过一个恰当的加权平均处理，使得

$$\boldsymbol{y}_m = \sum_{k=0}^{m} \alpha_{m,k}\boldsymbol{x}_k \tag{2.4.18}$$

是一个比 $\boldsymbol{x}_m$"更加接近精确解"的迭代解，其中 $\{\boldsymbol{x}_k\}_{k=0}^{\infty}$ 是由基础迭代算法 (2.4.17) 给出的迭代序列。这样的算法通常称为半迭代方法。

若 $\boldsymbol{x}_k \equiv \boldsymbol{x}_\star$ 就是精确解，我们自然希望 $\boldsymbol{y}_m \equiv \boldsymbol{x}_\star$ 也是精确解。因此，我们通常要求 (2.4.18) 中的参数组 $\{\alpha_{m,k}\}_{k=0}^{m}$ 满足相容性条件

$$\sum_{k=0}^{m} \alpha_{m,k} = 1. \tag{2.4.19}$$

记 $\boldsymbol{\eta}_m = \boldsymbol{y}_m - \boldsymbol{x}_\star$ 是半迭代方法 (2.4.18) 的第 m 步误差。简单计算可知误差方程为

$$\boldsymbol{\eta}_m = \sum_{k=0}^{m} \alpha_{m,k}\mathbb{G}^k\boldsymbol{e}_0 = \mathbb{P}_m(\mathbb{G})\boldsymbol{e}_0, \tag{2.4.20}$$

其中 $\boldsymbol{e}_0 = \boldsymbol{\eta}_0 = \boldsymbol{x}_0 - \boldsymbol{x}_\star$ 为初始误差。这里的 $\mathbb{P}_m(\lambda)$ 是一个 m 次多项式，相应的系数由参数组 $\{\alpha_{m,k}\}_{k=0}^{m}$ 给出。基于这个误差方程，我们自

然希望 $\mathbb{P}_m(\mathbb{G})$ 的谱半径能够远远小于 $\mathbb{G}$ 的谱半径，进而显著提升迭代序列的收敛速度。

至此，一个新的概念诞生了，迭代方法的研究思路也从“单项式算法”拓展到“多项式算法”。

变系数 Richardson 方法

事实上，半迭代方法的基本思想已经在某些传统算法中隐式地实现。例如简单的变系数 Richardson（1910）迭代法

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \tau_k(\boldsymbol{b} - \mathbb{A}\boldsymbol{x}_{k-1}), \tag{2.4.21}$$

其中的迭代参数 τ_k 是决定算法优劣的关键。Richardson 迭代法可以理解为一种简单的残量松弛方法。

若迭代参数 τ_k 保持不变，则称 (2.4.21) 为定常 Richardson 迭代法；否则，称其为变系数 Richardson 迭代法。

✍ **论题 2.8.** *变系数 Richardson 迭代法可视为定常 Richardson 迭代法的一种半迭代加速。*

为清楚展现变系数迭代的数值效果，让我们考虑一个特殊的情形，即系数矩阵 $\mathbb{A}$ 是实对称的正定矩阵。此时，迭代矩阵的谱范数就是它的谱半径

$$\max_{\lambda_i \in \lambda(\mathbb{A})} \left| \prod_{k=1}^{m} (1 - \tau_k \lambda_i) \right|.$$

给定总迭代步数 m，我们能否找到某个参数组 $\{\tau_k\}_{k=1}^m$，使得变系数 Richardson 迭代法获得最佳的收敛效果？为此,让我们考虑如下的 Cheybeshev 极大极小问题

$$\{\tau_k\}_{k=1}^m \ \operatorname{argmin} \max_{\lambda_{\min} \le \lambda \le \lambda_{\max}} \left| \prod_{k=1}^{m} (1 - \tau_k \lambda) \right|,$$

其中 $\lambda_{\max} > 0$ 和 $\lambda_{\min} > 0$ 分别是系数矩阵 $\mathbb{A}$ 的最大特征值和最小特征值。由 Cheybeshev 理论可知，最佳多项式是

$$P_m^\star(\lambda) = \frac{T_m\left(\dfrac{\lambda_{\max}+\lambda_{\min}-2\lambda}{\lambda_{\max}-\lambda_{\min}}\right)}{T_m\left(\dfrac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right)} \tag{2.4.22}$$

其中 $T_m(z)$ 是标准的 Cheybeshev 多项式[ix]。最佳参数组 $\{\tau_k\}_{k=1}^m$ 就是 $P_m^\star(\lambda)$ 的根，即

$$\boxed{\tau_k = \left[\frac{\lambda_{\max}-\lambda_{\min}}{2}\cos\left(\tfrac{2k-1}{2m}\pi\right) + \frac{\lambda_{\max}+\lambda_{\min}}{2}\right]^{-1},}$$

此时，变系数 Richardson 方法满足

$$\frac{\|\boldsymbol{e}_m\|_2}{\|\boldsymbol{e}_0\|_2} \leq 2\left(\frac{\sqrt{\kappa(\mathbb{A})}-1}{\sqrt{\kappa(\mathbb{A})}+1}\right)^m, \tag{2.4.23}$$

其中 $\kappa(\mathbb{A}) = \lambda_{\max}/\lambda_{\min}$ 为系数矩阵 $\mathbb{A}$ 的谱条件数。

✍ **论题 2.9.** *在定常 Richardson 迭代法中，最佳迭代参数 τ 该如何选取呢？相应的收敛速度满足怎样的估计？*

[ix] Cheybeshev 多项式

$$T_m(z) = \begin{cases} \frac{1}{2}[(z+\sqrt{z^2-1})^m + (z-\sqrt{z^2-1})^m], & |z| \geq 1; \\ \cos(m\arccos z), & |z| \leq 1. \end{cases}$$

具有非常完美的性质，例如恒等式

$$T_m\left(\frac{1+r^2}{1-r^2}\right) = \frac{1}{2}\left[\left(\frac{1+r}{1-r}\right)^2 + \left(\frac{1-r}{1+r}\right)^2\right], \quad r \in (-1,1).$$

证明留作练习。

设迭代的总步数 m 足够大，使得算法均达到用户的要求。上述结果表明：为使误差满足用户要求，在定常 Richardson 迭代方法中，实际所需的步数与 $\kappa(\mathbb{A})$ 成正比；而在变系数 Richardson 迭代方法中，实际所需的步数与 $\sqrt{\kappa(\mathbb{A})}$ 成正比。换言之，变系数策略使 Richardson 方法的收敛速度有了显著的提升。

✯ **说明 2.8.** 请注意：最优参数组的计算与 m 相关。在执行所谓的最佳变系数 *Richardson* 方法之前，我们需要事先确定总迭代步数 m 的值，然后才能确定相应的最优参数组。但是，若 m 取值很大，参数组的计算因舍入误差而产生偏差，进而导致实际的迭代收敛速度受到严重的破坏。为克服这个困难，常用的解决方法是采用较小的 m，并引入所谓的循环算法（或重新启动）策略。

Cheybeshev 加速方法

让我们再回到半迭代方法 (2.4.18)。虽然其思想非常朴素，但它还不是一个可以真正应用的数值方法。其一是数据存储的困境，我们不可能保留所有的历史数据和历史参数；其二是舍入误差的控制，我们需要给出尽可能准确的参数组信息。

✎ **论题 2.10.** 设基础迭代方法 (2.4.17) 的迭代矩阵 $\mathbb{G}$ 是实对称的。理论上最佳的参数组是什么？

设 $\mathbb{G}$ 具有完备的特征向量系 $\{\boldsymbol{\xi}_i\}_{k=1}^n$，相应的特征值 $\{\lambda_i\}_{i=1}^n$ 均为实数。设初始误差可以线性展开为

$$\boldsymbol{e}_0 = \sum_{1\le i\le n} \beta_i \boldsymbol{\xi}_i,$$

由误差方程 (2.4.20) 可知，半迭代方法 (2.4.18) 的误差满足

$$\boldsymbol{e}_k = \sum_{i=1}^{n} \Big[\sum_{\ell=0}^{k} \alpha_{k,\ell} \lambda_i^{\ell} \Big] \beta_i \boldsymbol{\xi}_i = \sum_{i=1}^{n} Q_k(\lambda_i) \beta_i \boldsymbol{\xi}_i.$$

我们希望找到一组最佳参数 $\{\alpha_{k,\ell}\}_{\ell=0}^{k}$，使得迭代误差 (在 2-范数度量下) 收敛向零的速度达到最快？如前面的讨论，这个目标也可转化为一个 Chebyshev 极大极小问题：

$$Q_k^{\star}(\lambda) = \arg \min_{Q_k \in \mathbb{P}_k^{\sharp}} \max_{\lambda \in [\lambda_{\min}, \lambda_{\max}]} |Q_k(\lambda)|,$$

其中 $\lambda_{\max}$ 和 $\lambda_{\min}$ 是迭代矩阵 $\mathbb{G}$ 的最大特征值和最小特征值[x]，$\mathbb{P}_k^{\sharp}$ 是由次数不超过 k 的系数和为一的多项式全体而构成的集合。

答案就是归一化[xi]的 Chebyshev 多项式

$$Q_k^{\star}(\lambda) = \frac{T_k\left(\ell(\lambda)\right)}{T_k\left(\ell(1)\right)}, \quad \text{其中 } \ell(\lambda) = \frac{2\lambda - \lambda_{\max} - \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}},$$

其中 $T_k(z)$ 为标准的 Chebyshev 多项式。显见，半迭代方法 (2.4.18) 的最佳参数组 $\{\alpha_{k,\ell}\}_{\ell=0}^{k}$，就是这个最佳多项式 $Q_k^{\star}(\lambda)$ 的相应系数。

✯ **说明 2.9.** 上述讨论也可以推广到基础迭代矩阵 $\mathbb{G}$ 具有复特征值的情形，其参数组的设定与复特征值所属的椭圆区域有关。具体内容超出本课程的要求，略。

❀ **思考 2.5.** 请尝试度量一下这个算法的平均收敛速度，并回答算法收敛速度是否具有本质上的提升？

[x] 请注意：不是 $\mathbb{A}$ 的最大特征值和最小特征值。

[xi] 此处，我们假设了 $\mathbb{G}$ 的特征值均小于 1；其它情形也可处理，但过程较繁，略。

✎ **论题 2.11.** 当迭代步数 k 非常大的时候，我们不可能真正计算和存储所有的迭代参数和历史数据。为此，我们必须给出一个等价的且实用的计算公式。

此时，Chebyshev 多项式的正交性，特别是所谓的三项递推关系式

$$T_{n+1}(z) = 2xT_n(z) - T_{n-1}(z) \tag{2.4.24}$$

将起到至关重要的作用，其中 $T_0(z) = 1$ 和 $T_1(z) = z$. 利用这个三项递推关系式，由误差方程 (2.4.20)，我们可以反推出如下的变系数二步迭代公式

$$\begin{aligned}\boldsymbol{x}_{k+1} &= 2\ell(\mathbb{G})\frac{T_k(\xi)}{T_{k+1}(\xi)}\boldsymbol{x}_k - \frac{T_{k-1}(\xi)}{T_{k+1}(\xi)}\boldsymbol{x}_{k-1} + \frac{4}{\lambda_{\max}-\lambda_{\min}}\frac{T_k(\xi)}{T_{k+1}(\xi)}\boldsymbol{g}\\ &= \rho_{k+1}\{\nu(\mathbb{G}\boldsymbol{x}_k + \boldsymbol{g}) + (1-\nu)\boldsymbol{x}_k\} + (1-\rho_{k+1})\boldsymbol{x}_{k-1},\end{aligned}$$

其中的固定参数为

$$\nu = \frac{2}{2-\lambda_{\max}-\lambda_{\min}},\ \xi = \frac{2-\lambda_{\max}-\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}},$$

变化的参数为

$$\rho_{k+1} = \frac{2\xi T_k(\xi)}{T_{k+1}(\xi)}.$$

在上述过程中，我们还用到了 $(\mathbb{I}-\mathbb{G})\boldsymbol{x}_\star = \boldsymbol{g}$，即基础迭代方法是相容的性质。

为启动这个半迭代算法，我们仅需任意选取一个初始值 $\boldsymbol{x}_0$，然后由基础迭代方法给出 $\boldsymbol{x}_1 = \mathbb{G}\boldsymbol{x}_0 + \boldsymbol{g}$。

事实上，利用三项递推关系式 (2.4.24)，我们可按公式

$$\rho_{k+1} = \Big[1 - \frac{1}{4\xi^2}\rho_k\Big]^{-1},\ \text{其中 } \rho_1 = 2.$$

计算变化的参数 ρ_k。换言之，在真正的半迭代方法中，我们无需存储关于 Chebyshev 多项式的任何信息，以及大量的历史数据。

❀ **思考 2.6.** 设线性方程组的系数矩阵 $\mathbb{A}$ 具有性质 $\boldsymbol{A}$，或者直接假设它就是由 (2.3.12) 所定义。请考虑 *Jacobi* 迭代方法的半迭代加速，并观察 ρ_k 的极限到底是什么？它与 *SOR* 方法的最佳松弛因子有何关联？

在半迭代方法中，最佳参数组的设定均同基础迭代矩阵（或者间接地同系数矩阵）的谱分布有关。最大和最小特征值的估算不准，将会影响到最佳参数组的设置，进而严重破坏算法的收敛速度。但是，特征值的计算是一个更加困难的代数问题，这个要求难以实现。在某种程度上，这个缺陷是致命的，它严重限制了半迭代方法的广泛应用。我们期待能够找到一组无参数的快速方法。

2.5 共轭斜量法

共轭斜量法是由 Hestenes 和 Stiefel 在上世纪 50 年代首先提出的，是求解对称正定线性方程组的首选数值方法。它无需事先估计系数矩阵的特征值，具有无参数、收敛快等优势。尽管它是基于变分形式的直接法，却常常被视为一种迭代算法。

2.5.1 等价的极值问题与求解

✎ **论题 2.12.** 考虑线性方程组 $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$，其中的系数矩阵 $\mathbb{A}$ 是对称正定的。在共轭斜量法的多种引进方式之中，较为直观的是将线性方程组问题转化为二次方程（椭圆抛物面）的最优化问题：

$$\boldsymbol{x}_\star = \arg\min_{\forall \boldsymbol{x}} f(\boldsymbol{x}), \text{ 其中 } f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^\top \mathbb{A}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}.$$

这里的 $f(\boldsymbol{x})$ 称为离散系统的总能量，$\boldsymbol{x}_\star$ 是问题的真解。

定理 2.13. 两种表述是彼此等价的。

针对二次方程的优化问题，最优化理论给出了很多快速方法。最常用的核心技术是一维搜索策略，即由当前位置 $\boldsymbol{x}_k$ 出发，沿着某个搜索方向 $\boldsymbol{p}_k$，到达一个新的最优位置 $\boldsymbol{x}_{k+1}$。简单的计算给出相应的答案

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, \quad \text{其中 } \alpha_k = -\frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top \mathbb{A} \boldsymbol{p}_k}, \quad \boldsymbol{r}_k = \mathbb{A}\boldsymbol{x}_k - \boldsymbol{b}.}$$

依次执行一维搜索过程，所有的搜索方向将形成一个历史搜索空间

$$\mathcal{L}_k = \operatorname{span}\{\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k-1}\}, \tag{2.5.25}$$

对应一个逐维扩张的过程。显然，新的一维搜索位置

$$\boldsymbol{x}_k \in \pi_k \equiv \boldsymbol{x}_0 + \mathcal{L}_k. \tag{2.5.26}$$

下面，我们考虑一个重要的问题。

定义 2.4. 若 $\boldsymbol{x}_k = \arg\min_{x \in \pi_k} f(\boldsymbol{x})$，则称 $\boldsymbol{x}_k$ 关于搜索空间 $\mathcal{L}_k$ 是最优的。

引理 2.1. 当前搜索位置 $\boldsymbol{x}_k \in \pi_k$ 关于历史搜索空间 $\mathcal{L}_k$ 最优的充要条件是当前位置的残量 $\boldsymbol{r}_k = \mathbb{A}\boldsymbol{x}_k - \boldsymbol{b}$ 满足 $\boldsymbol{r}_k \perp \mathcal{L}_k$，即

$$\boxed{\boldsymbol{r}_k^\top \boldsymbol{p}_\ell = 0, \quad \ell = 0, 1, \ldots, k-1.}$$

假设搜索位置关于历史搜索空间的最优性质一直保持。若最终的搜索空间 $\mathcal{L}_k$ 可以扩张到 $\mathbb{R}^n$，则最终的最优搜索位置就是二次方程的最优位置，也就是线性方程组的精确解。

那么，上述假设能够真正地实现吗？为此，我们需要探讨历史搜索空间 $\mathcal{L}_k$ 的构造方式，使得引理 2.1 的充要条件成立。

2.5.2 共轭斜量方法的框架

一维搜索算法给出的新位置，关于当前搜索方向是最优的。关于以前的搜索方向，它是否也是最优的呢？

✎ **论题 2.13.** *在当前位置 $\boldsymbol{x}_k$ 处，最自然的搜索方向 $\boldsymbol{p}_k$ 是最速下降方向，即*

$$\boldsymbol{p}_k = -grad f(\boldsymbol{x}_k) = \boldsymbol{b} - \mathbb{A}\boldsymbol{x}_k = -\boldsymbol{r}_k. \tag{2.5.27}$$

基于这种搜索方向，相应的一维搜索过程称为最速下降法。

在执行两步最速下降法之后，有 $\boldsymbol{x}_{k+2} \in \boldsymbol{x}_k + span(\boldsymbol{r}_k, \boldsymbol{r}_{k+1})$。显然，它关于搜索方向 $\boldsymbol{p}_{k+1} = -\boldsymbol{r}_{k+1}$ 是最优的，即 $\boldsymbol{r}_{k+2}^\top \boldsymbol{r}_{k+1} = 0$。但是简单的分析表明:

$$\begin{aligned}\boldsymbol{r}_{k+2}^\top \boldsymbol{r}_k &= (\boldsymbol{r}_{k+1} + \alpha_{k+1}\mathbb{A}\boldsymbol{r}_{k+1})^\top \boldsymbol{r}_k = \alpha_{k+1}\boldsymbol{r}_{k+1}^\top \mathbb{A}\boldsymbol{r}_k \\ &= \frac{\alpha_{k+1}}{\alpha_k}\boldsymbol{r}_{k+1}^\top(\boldsymbol{r}_{k+1} - \boldsymbol{r}_k) = \frac{\alpha_{k+1}}{\alpha_k}\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_{k+1} \neq 0.\end{aligned}$$

换言之，$\boldsymbol{x}_{k+2}$ 关于前一个搜索方向 $\boldsymbol{p}_k = -\boldsymbol{r}_k$ 不是最优的。

因此，引理 2.1 的充要条件是不成立的，最速下降法无法实现我们的假设目标。

❀ **思考 2.7.** *上述事实造成最速下降方法的收敛速度越来越慢，产生所谓的"盘旋收敛"现象。尽管如此，可以证明：最速下降法是收敛的。请读者给出相应的（按 2-范数）收敛估计。*

要使当前位置关于所有的历史搜索方向都是最优的，这是一个难以直接回答的问题。让我们考虑一个简单的前提条件：

如何确保当前位置关于最近的两个搜索方向都是最优的？换言之，当前位置关于局部的二维搜索空间是最优的。

设 $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{q}$ 是从位置 $\boldsymbol{x}_k$ 出发，沿搜索方向 $\boldsymbol{q}$ 给出的最优位置。若还想要它关于搜索方向 $\boldsymbol{p}$ 也是最优的，我们应该有

$$0 = \boldsymbol{r}_{k+1}^{\top}\boldsymbol{p} = (\boldsymbol{r}_k - \mathbb{A}\boldsymbol{q})^{\top}\boldsymbol{p} = \boldsymbol{r}_k^{\top}\boldsymbol{p} - \boldsymbol{q}^{\top}\mathbb{A}\boldsymbol{p} = -\boldsymbol{q}^{\top}\mathbb{A}\boldsymbol{p},$$

即两个搜索方向 $\boldsymbol{p}$ 和 $\boldsymbol{q}$ 应该是 $\mathbb{A}$-共轭的。这导出如下概念。

定义 2.5. 对任意两个指标 i 和 j，均有

$$\boldsymbol{p}_i^{\top}\mathbb{A}\boldsymbol{p}_j = 0, \quad i \neq j,$$

则称非零的向量系 $\{\boldsymbol{p}_\kappa\}_{\kappa=0}^{m}$ 构成一个共轭向量系。以此共轭向量系为搜索方向的一维搜索算法，称为共轭斜量法。

利用数学归纳法，我们可以证明

定理 2.14. 在共轭斜量法中，当前位置 $\boldsymbol{x}_{k+1}$ 关于搜索空间 $\mathcal{L}_k$ 是最优的。

在一个共轭向量系中，所有的搜索方向都是线性无关的。若所有计算都是精确的，由定理 2.14 可知，共轭斜量法至多进行 n 步，利用有限次的四则运算，便可给出问题的真解。因此说，共轭斜量法是直接算法。

2.5.3 共轭斜量系的构造过程

论题 2.14. 共轭斜量法获得应用的前提是共轭斜量系的构造。事实上，共轭斜量方向可以局部地递推构造。

既然 $\boldsymbol{r}_{k+1}$ 与 $\boldsymbol{p}_k$ 互相垂直，它们可以张成一个二维平面。在这个平面上，我们希望找到一个新的搜索方向 $\boldsymbol{p}_{k+1}$，使其同旧的搜索方向 $\boldsymbol{p}_k$ 是 $\mathbb{A}$-共轭的。相应的计算流程如下：

取初始方向为 $\boldsymbol{p}_0 = -\boldsymbol{r}_0 = \boldsymbol{b} - \mathbb{A}\boldsymbol{x}_0$，依次执行

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, \quad \alpha_k = -\frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top \mathbb{A}\boldsymbol{p}_k}, \tag{2.5.28a}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \alpha_k \mathbb{A}\boldsymbol{p}_k, \tag{2.5.28b}$$

$$\boldsymbol{p}_{k+1} = -\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k, \quad \beta_k = \frac{\boldsymbol{r}_{k+1}^\top \mathbb{A}\boldsymbol{p}_k}{\boldsymbol{p}_k^\top \mathbb{A}\boldsymbol{p}_k}. \tag{2.5.28c}$$

迭代过程包含了大量的内积运算，具有很好的并行特征。

论题 2.15. 上述共轭斜量方向的构造过程是局部的。但是，它可以给出整体的共轭斜量系，直至迭代终止。

在共轭斜量法中，三套向量组扮演着非常重要的角色。在迭代终止之前，即 $\boldsymbol{r}_k \neq 0$，我们有：

$$\begin{aligned}\operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_k\} &= \operatorname{span}\{\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_k\} \\ &= \operatorname{span}\{\boldsymbol{r}_0, \mathbb{A}\boldsymbol{r}_0, \ldots, \mathbb{A}^k \boldsymbol{r}_0\}.\end{aligned}$$

这三个空间分别称为残量空间，搜索空间和 Krylov 子空间。

基于不同的向量组，共轭斜量法可以由对称正定情形推广到非对称非正定的情形。

论题 2.16. （非零）共轭方向与（非零）残量方向之间的关系：

$$\boxed{\boldsymbol{r}_i^\top \boldsymbol{r}_j = 0, \forall i \neq j, \quad \boldsymbol{p}_k^\top \boldsymbol{r}_i = \begin{cases} -\boldsymbol{r}_k^\top \boldsymbol{r}_k, & i \leq k; \\ 0 & i \geq k+1. \end{cases}}$$

换言之，残量 $\boldsymbol{r}_i$ 是椭圆抛物面在当前位置的法方向，而共轭方向 $\boldsymbol{p}_i$ 是更快指向椭圆抛物面顶点的斜方向。

利用上述性质，共轭斜量法的参数计算可以进一步的简化，即

$$\boxed{\alpha_k = \frac{\boldsymbol{r}_k^\top \boldsymbol{r}_k}{\boldsymbol{p}_k^\top \mathbb{A} \boldsymbol{p}_k}, \quad \beta_k = \frac{\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^\top \boldsymbol{r}_k}.}$$

因为重复运算的出现，相应的内积运算次数下降了。

❀ **思考 2.8.** 若我们忽视 α_k 和 β_k 的计算过程，直接将它们看作事先设定好的迭代参数。请问：共轭斜量法是一个二阶非定常迭代方法吗？

2.5.4 收敛性分析

通常，共轭斜量法可以视为一种迭代算法。

定理 2.15. 共轭斜量法的迭代误差按 l_2 模是单调下降的。

证明：基于共轭斜量法有限步到达精确解的性质。 □

✍ **论题 2.17.** 利用定理 (2.14) 可知，共轭斜量法的每步误差能量都对应某个二次泛函的极小，即

$$\|\boldsymbol{e}_{k+1}\|_{\mathbb{A}}^2 \equiv \boldsymbol{e}_{k+1}^\top \mathbb{A} \boldsymbol{e}_{k+1} = \min_{Q_k \in \mathbb{P}_k} \boldsymbol{e}_0^\top \left\{ \mathbb{A} \Big[\mathbb{I} + \mathbb{A} Q_k(\mathbb{A}) \Big]^2 \right\} \boldsymbol{e}_0, \quad (2.5.29)$$

其中 $\mathbb{P}_k$ 是由次数不超过 k 的多项式全体形成的集合。

它蕴含两个重要的推论。

定理 2.16. 若矩阵 $\mathbb{A}$ 仅仅具有 m 个相异的特征值，则共轭斜量法至多 m 步便可得到线性方程组的真解。

定理 2.17. 共轭斜量法满足如下的估计:

$$\frac{\|\boldsymbol{e}_m\|_{\mathbb{A}}}{\|\boldsymbol{e}_0\|_{\mathbb{A}}} \leq 2\left(\frac{\sqrt{\kappa(\mathbb{A})}-1}{\sqrt{\kappa(\mathbb{A})}+1}\right)^m,$$

其中 $\kappa(\mathbb{A})$ 是矩阵 $\mathbb{A}$ 的谱条件数。

✯ **说明 2.10.** 定理 *2.17* 表明：共轭斜量法和（最优参数的）半迭代方法具有相同的收敛速度。但是，共轭斜量法的实现过程不需要矩阵 $\mathbb{A}$ 的特征值信息，实际应用起来更加便捷。

✯ **说明 2.11.** 当系数矩阵严重病态的时候，共轭斜量法的收敛速度将会变慢[xii]。

综合上述两个定理的推导过程，我们可以发现：共轭斜量法的收敛速度不仅依赖系数矩阵的条件数，还依赖系数矩阵的特征值聚集状态。通常，共轭斜量法的实际迭代次数远远少于矩阵的阶数，并且具有所谓的"超线性收敛"现象。

2.5.5 预处理共轭斜量方法

预处理技术是数值代数的一个基本技术，应用范围极广。它可以改善同解方程组的条件数，降低数值敏感程度，提高算法的计算效率。下面，我们基于对称正定线性方程组的共轭斜量方法，简要阐述预处理技术的基本思想和实现过程。

[xii] 即便如此，它常常或也给出更加可信的数值结果。

✍ **论题 2.18.** 令 $\mathbb{Q}=\mathbb{C}\mathbb{C}^{\top}$, 考虑等价的线性方程组

$$\boxed{\mathbb{C}^{-1}\mathbb{A}\mathbb{C}^{-\top}\mathbb{C}^{\top}\boldsymbol{x}=\mathbb{C}^{-1}\boldsymbol{b},}$$

并采用 *CG* 算法求解这个等价问题。相应的计算流程如下:

取初始方向 $\boldsymbol{r}_0=\mathbb{A}\boldsymbol{x}_0-\boldsymbol{b}$, 令 $\boldsymbol{z}_0=\mathbb{Q}^{-1}\boldsymbol{r}_0, \boldsymbol{p}_0=-\boldsymbol{z}_0$, 做迭代

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_k\boldsymbol{p}_k,\quad \alpha_k=-\frac{\boldsymbol{r}_k^{\top}\boldsymbol{z}_k}{\boldsymbol{p}_k^{\top}\mathbb{A}\boldsymbol{p}_k}, \tag{2.5.30a}$$

$$\boldsymbol{r}_{k+1}=\boldsymbol{r}_k+\alpha_k\mathbb{A}\boldsymbol{p}_k, \tag{2.5.30b}$$

$$\boldsymbol{z}_{k+1}=\mathbb{Q}^{-1}\boldsymbol{r}_{k+1}, \tag{2.5.30c}$$

$$\boldsymbol{p}_{k+1}=-\boldsymbol{z}_{k+1}+\beta_k\boldsymbol{p}_k,\quad \beta_k=\frac{\boldsymbol{r}_{k+1}^{\top}\boldsymbol{z}_{k+1}}{\boldsymbol{r}_k^{\top}\boldsymbol{z}_k}. \tag{2.5.30d}$$

在每步迭代, 我们仅需增加少量的额外工作, 即一个关于预处理方程 $\mathbb{Q}\boldsymbol{z}=\boldsymbol{g}$ 的求解过程。在 *Matlab* 中, 它所对应的命令是 *pcg()*.

此时, $\mathbb{Q}\boldsymbol{z}=\boldsymbol{g}$ 的求解过程是否简单易行, 成为预处理策略能否成功的关键。通常, 预处理矩阵可利用矩阵分裂思想给出, 譬如 SSOR 方法中的

$$\mathbb{Q}=(\mathbb{D}+\omega\mathbb{L})\mathbb{D}^{-1}(\mathbb{D}+\omega\mathbb{L})^{\top}. \tag{2.5.31}$$

我们还可以采用不完全 LU 分解, 或者逆矩阵 $\mathbb{A}^{-1}$ 的多项式近似等方法, 给出相应的预处理矩阵。若同解方程组的条件数得到明显的下降, CG 算法的收敛速度将会得到很大的改善。因篇幅有限, 此处不再展开。

✯ **说明 2.12.** 到目前为止, 共轭斜量方法已经成功地推广到任意类型的线性方程组, 形成更一般的 *Galerkin* 方法或者 *Krylov* 子空间投

影方法。例如 *GMRES* 方法，双稳定化的 *CG* 方法，或者平方 *CG* 方法等等。这些方法都已经被收录在 *Matlab* 中。

第 3 章
线性最小二乘问题

在数据分析和线性回归等实际问题中，我们常常遇到如下的线性方程组

$$\mathbb{A}_{m\times n}\boldsymbol{x}_n = \boldsymbol{b}_m, \tag{3.0.1}$$

相应的未知数和约束条件个数是不匹配的。通常，它是一个矛盾方程组[i]，任何向量都无法使其等号成立。

定义 3.1. *若 $f(\boldsymbol{x}) = \|\mathbb{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ 在 $\tilde{\boldsymbol{x}}$ 处达到极小（或者最小）[ii]，则称 $\tilde{\boldsymbol{x}}$ 为线性方程组 (3.0.1) 的一个最小二乘解。*

这个概念的提出、应用及其计算方法，都是值得深入研究的课题。

3.1 基本理论和数值难度

3.1.1 最小二乘解

定理 3.1. *最小二乘解是必然存在的。*

证明：证明的方法有很多。下面，我们用代数的方法证明。若 $\boldsymbol{b} \in \mathrm{Range}(\mathbb{A})$，由线性方程组的基本理论可知，解在传统意义下是存在的。显然，这个解就是最小二乘解。若 $\boldsymbol{b} \notin \mathrm{Range}(\mathbb{A})$，利用 $\boldsymbol{b}$ 在 $\mathrm{Range}(\mathbb{A})$ 及其正交补空间上的直和分解

$$\boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2, \quad \text{其中} \quad \boldsymbol{b}_1 \in \mathrm{Range}(\mathbb{A}), \boldsymbol{b}_2 \in [\mathrm{Range}(\mathbb{A})]^{\perp},$$

[i] 若存在无穷多解，称其为不定方程组。

[ii] 当然，我们可采用其他度量方式，例如残量的最大模、1-模以及加权范数等等。本课程不讨论这些问题。

可知最小二乘解可由 $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}_1$ 所决定，其存在性是显然的。 □

❀ **思考 3.1.** 利用数学分析的方法，证明最小二乘解是必然存在的。

定理 3.2. 最小二乘问题与法方程组

$$\mathbb{A}^\top \mathbb{A}\boldsymbol{x} = \mathbb{A}^\top \boldsymbol{b}$$

同解，即最小二乘解的对应残量 $\boldsymbol{r} = \mathbb{A}\boldsymbol{x} - \boldsymbol{b}$ 同 $\mathbb{A}$ 的每个列向量都是正交的。

证明：利用极值同导数的关系，简单的计算即可。 □

定理 3.3. 若矩阵 $\mathbb{A}_{m\times n}$ 是列满秩的，即 $rank(\mathbb{A}) = n$，则最小二乘解是唯一的，并具有表达式

$$\boldsymbol{x} = (\mathbb{A}^\top \mathbb{A})^{-1}\mathbb{A}^\top \boldsymbol{b}.$$

否则，最小二乘解是不唯一的，有无穷多个。

3.1.2 满秩分解表示

最小二乘解的公式表达一直是相关领域的重要目标。下面，我们将基于系数矩阵的某个满秩分解，显式地给出一个重要的最小二乘解。

定理 3.4. 若 $r = rank(\mathbb{A}_{m\times n}) > 0$，则矩阵存在满秩分解

$$\mathbb{A}_{m\times n} = \mathbb{G}_{m\times r}\mathbb{F}_{r\times n}$$

其中 $\mathbb{G}$ 是列满秩的，$\mathbb{F}$ 是行满秩的。

定理 3.5. 若矩阵 $\mathbb{A}$ 具有满秩分解 $\mathbb{A} = \mathbb{G}\mathbb{F}$，则线性方程组 (3.0.1) 有一个最小二乘解

$$\boxed{\boldsymbol{x}_{\mathrm{LS}} = \mathbb{G}^{\top}(\mathbb{G}\mathbb{G}^{\top})^{-1}(\mathbb{F}^{\top}\mathbb{F})^{-1}\mathbb{F}^{\top}\boldsymbol{b}.}$$

这个解具有重要的意义，称为**极小最小二乘解**，因为它是最小二乘解集合中向量长度最小的那个唯一解。

3.1.3 矩阵广义逆

众所周知，若系数矩阵 $\mathbb{A}$ 是可逆的，线性方程组的解可以利用逆矩阵直接表示。那么，最小二乘问题也可以吗？早在 1920 年，E. H. Moore 就提出了广义逆矩阵的概念，但是在其后的 30 多年中，由于不知这个概念的用途，他的理论几乎几乎被遗忘。直到 1955 年，R. Penrose 更加明确地给出等价定义，广义逆矩阵的研究才真正进入崭新的阶段。

定义 3.2. 设 $\mathbb{A}$ 是已知的 $m \times n$ 阶实矩阵[iii]。若 $n \times m$ 阶实矩阵 $\mathbb{X}$ 满足

$$\boxed{\mathbb{A}\mathbb{X}\mathbb{A} = \mathbb{A}, \quad \mathbb{X}\mathbb{A}\mathbb{X} = \mathbb{X}, \quad (\mathbb{A}\mathbb{X})^{\top} = \mathbb{A}\mathbb{X}, \quad (\mathbb{X}\mathbb{A})^{\top} = \mathbb{X}\mathbb{A},}$$

则称 $\mathbb{X}$ 是 $\mathbb{A}$ 的（*Moore-Penrose*）广义逆，记为 $\mathbb{X} = \mathbb{A}^{\dagger}$.

定理 3.6. 广义逆矩阵是存在唯一的。

证明：存在性可由满秩分解所给出，唯一性可由定义给出。 □

定理 3.7. 若有满秩分解 $\mathbb{A} = \mathbb{F}\mathbb{G}$，则

$$\mathbb{A}^{\dagger} = \mathbb{G}^{\top}(\mathbb{G}\mathbb{G}^{\top})^{-1}(\mathbb{F}^{\top}\mathbb{F})^{-1}\mathbb{F}^{\top}.$$

换言之，极小最小二乘解可表示为 $\boldsymbol{x}_{LS} = \mathbb{A}^{\dagger}\boldsymbol{b}$.

[iii] 对于复矩阵，这个概念的推广是简单的，即定义中的转置变为共轭转置。

证明：简单验证广义逆矩阵的四条性质即可。 □

广义逆矩阵的计算较为复杂。但是，对于某些具有特殊结构的矩阵，相应的广义逆矩阵计算是简单的。例如，

$$\begin{bmatrix} \mathbb{X}_{r,r} & \mathbb{O}_{r,n-r} \\ \mathbb{O}_{m-r,r} & \mathbb{O}_{m-r,n-r} \end{bmatrix}^{\dagger} = \begin{bmatrix} \mathbb{X}_{r,r}^{-1} & \mathbb{O}_{r,m-r} \\ \mathbb{O}_{n-r,r} & \mathbb{O}_{n-r,m-r} \end{bmatrix},$$

其中 $\mathbb{X}$ 是可逆的方阵。

✎ **论题 3.1.** *若矩阵本身就是可逆的，广义逆矩阵与古典逆矩阵是相同的。但是，广义逆矩阵与古典逆矩阵两个概念有着明显的区别。*

为说明这个问题，让我们考虑奇异的方阵。此时，很多适用于古典逆矩阵的运算规则和理论性质，可能不再恒成立，例如：

1. $(\mathbb{AB})^{\dagger} \neq \mathbb{B}^{\dagger}\mathbb{A}^{\dagger}$，　$\mathbb{AA}^{\dagger} \neq \mathbb{A}^{\dagger}\mathbb{A}$，　$(\mathbb{A}^{k})^{\dagger} \neq (\mathbb{A}^{\dagger})^{k}$；
2. $\mathbb{A}$ 与 $\mathbb{A}^{\dagger}$ 的非零特征值不是互为倒数。
3. 广义逆矩阵可能不再连续地依赖于原矩阵元素的变化而变化。

最后的性质特别重要，它表明最小二乘问题的数值计算可能是非常困难的。

事实上，在广义逆矩阵的扰动理论中，扰动矩阵的秩能否保持不变是特别重要的。在扰动过程中，若矩阵的秩发生变化，微小的元素变化可能会引起广义逆矩阵的元素相距甚远。但是，若矩阵的秩保持不变，则广义逆矩阵的元素变化也是连续的，相应的敏感程度可利用矩阵的条件数来描述。为理解上述结论，请考虑如下两个矩阵

$$\mathbb{A}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \varepsilon & 0 \end{bmatrix}, \quad \mathbb{A}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \varepsilon & 1 \end{bmatrix},$$

的两个广义逆矩阵在 $\varepsilon \to 0$ 时的表现。

❀ **思考 3.2.** 请给出相应的反例。

3.1.4 数值算法

最小二乘问题的系数矩阵是否（列）亏秩，将严重影响最小二乘解的数值计算结果。当矩阵是亏秩的时候，相应列向量的线性相关表现较为复杂[iv]，使得某些算法无法顺利进行到底。即便算法可以顺利地进行，最终的计算结果可能含义不同。此外，由于舍入误差的数值影响表现截然不同，不同算法给出的计算结果可能区别明显。

因篇幅有限，若未做特殊声明，均假设最小二乘问题是列满秩的。对于列亏秩的情形，我们仅仅会给出适当的说明。

最小二乘问题的数值计算方法有直接法和迭代法两种，其中直接法主要有正规化和直交化两种方法。至于其它的方法（例如最优化方法），略。

正规化方法

当系数矩阵 $\mathbb{A}$ 是列满秩的，此时的最小二乘解是唯一的。我们可采用法方程组 $\mathbb{A}^\top \mathbb{A}\boldsymbol{x} = \mathbb{A}^\top \boldsymbol{b}$，或者所谓的扩展方程组

$$\begin{bmatrix} \mathbb{A}_1 & 0 & \mathbb{I}_n \\ \mathbb{A}_2 & \mathbb{I}_{m-n} & 0 \\ 0 & \mathbb{A}_2^\top & \mathbb{A}_1^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{0} \end{bmatrix},$$

借用直接法或迭代法进行相应的计算。其中，$\boldsymbol{r}$ 为待解残量，$\mathbb{A}_1$ 是系数矩阵 $\mathbb{A}$ 中的 $r = \mathrm{rank}(\mathbb{A}) = n$ 阶可逆方阵。相应数据对应如下的矩阵分

[iv] 数值确定一个矩阵的秩和最大线性无关组，是非常困难的，特别当矩阵本身就是亏秩的时候

块

$$\begin{bmatrix} \mathbb{A} & \boldsymbol{r} & \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \mathbb{A}_1 & \boldsymbol{r}_1 & \boldsymbol{b}_1 \\ \mathbb{A}_2 & \boldsymbol{r}_2 & \boldsymbol{b}_2 \end{bmatrix}.$$

虽然这种方法在表面上回避了 $\mathbb{A}^\top\mathbb{A}$ 的直接计算，并同时计算出相应的残量，扩展方程组方法与法方程组方法具有相同的数值困难。

正规化方法具有很大的数值困难，特别是对于那些病态的最小二乘问题。主要的原因主要有二个。

1. 法方程组系数矩阵条件数 $\kappa_2(\mathbb{A}^\top\mathbb{A})$ 是矩阵 $\mathbb{A}$ 条件数

   $$\kappa_2(\mathbb{A}) = \|\mathbb{A}\|_2\|\mathbb{A}^\dagger\|_2$$

   的平方[v]，矩阵条件数的膨胀是非常严重的。换言之，法方程组的病态程度将更加可怕，舍入误差将会严重影响最终结果。

2. 浮点运算可能出现上下溢出，使得法方程组系数矩阵亏秩。设 ϑ 是机器精度。当 $\varepsilon < \sqrt{\vartheta}$ 时，列满秩矩阵 $\mathbb{A}_\varepsilon$ 满足

   $$\mathbb{A}_\varepsilon = \begin{bmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix}, \quad \mathbb{A}_\varepsilon^\top\mathbb{A}_\varepsilon = \begin{bmatrix} 1+\varepsilon^2 & 1 \\ 1 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

   换言之，计算机给出的法方程组已经奇异了。

尽管如此，正规化方法堪称最快的计算方法。当系数矩阵是良态的时候，它常常是首选的方法。

[v] 任意矩阵的谱范数均可定义为 $\|\mathbb{A}\|_2 = [\varrho(\mathbb{A}^\top\mathbb{A})]^{1/2}$，或者为矩阵 $\mathbb{A}$ 的最大奇异值。事实上，列满秩的最小二乘问题关于解的灵敏度将主要由 $\kappa_2(\mathbb{A}) + \|\boldsymbol{r}\|_2\kappa_2^2(\mathbb{A})$ 来度量，而关于残量的敏感度只是线性的依赖于 $\kappa_2(\mathbb{A})$。

✯ **说明 3.1.** 共轭斜量方法适用于最小二乘问题的计算。即使系数矩阵不是列满秩的，共轭斜量方法也可以给出最小二乘解，但是它不一定是极小最小二乘解。

直交化方法

为克服正规化方法中的舍入误差影响，我们通常会采用更为健壮的直交化方法。它等同于系数矩阵（或增广矩阵）的直交分解过程，具体实现过程将在下一节中给出。

3.2 矩阵的直交分解

下面, 我们介绍两类三种方法。第一类方法是 Gram-Schmidt 正交化过程，将系数矩阵的线性无关列向量组转化为一个直交列向量组；第二类方法是正交矩阵变换方法，利用 Householder 镜像矩阵，或者 Givens 平面旋转矩阵，将系数矩阵逐步转化为一个上梯形矩阵。

用矩阵语言来描述，第一类方法是一系列三角形（或梯形）矩阵的右乘过程，而第二类方法是一系列直交阵的左乘过程。

3.2.1 Gram-Schmidt 直交化方法

Gram-Schmidt（GS）直交化方法是一个重要的代数计算工具。它几乎出现在所有的高等代数教材中；此处不再赘述。但是，传统的实现方法具有较差的数值稳定性，不太适用于病态问题的数值计算。

✎ **论题 3.2.** 利用 *Gram-Schmidt* 直交化方法，我们可得矩阵 $\mathbb{A}$ 的一个直交分解:

(A): $\mathbb{A}_{m\times n}\mathbb{P}_{n\times n} = \mathbb{Q}_{m\times r}\mathbb{U}_{r\times n}$, 其中 $r = rank(\mathbb{A})$, $\mathbb{P}$ 为 n 阶置换阵, $\mathbb{Q}$ 是列直交阵, $\mathbb{U}$ 是对角线元素为正的上梯形矩阵。

这种表述也称为矩阵的 QR 分解。

✯ **说明 3.2.** 我们可以采用数据覆盖技术，将列直交阵 $\mathbb{Q}_{m\times r}$ 依旧存储在系数矩阵 $\mathbb{A}$ 的原有位置；我们只需额外开辟一个数据空间，存储上梯形矩阵 $\mathbb{U}$ 的信息。

✯ **说明 3.3.** 事实上，*GS* 直交化方法有两种不同的执行次序，在数学上它们是等价的:

1. 传统的 *CGS* 方法是利用当前列向量与历史列向量组的正交性，逐列计算矩阵 $\mathbb{U}$ 的信息;
2. 而修正的 *MGS* 方法是利用当前列向量与未来列向量组的正交性，逐行计算矩阵 $\mathbb{U}$ 的信息。我们不断剔除待处理列向量在历史列向量组空间的投影，可视为计算的问题规模越来越小，舍入误差的积累影响较弱。

不同的执行次序造成舍入误差具有不同的表现。这种根据数学上等价的方法而产生的算法具有不同的稳定性，是计算数学的一个特点，应引起重视。

✯ **说明 3.4.** 同传统的 *CGS* 方法相比，修正的 *MGS* 方法数值健壮性更好。设 $\mathbb{A}_{m\times n}$ 是一个列满秩矩阵，相应的舍入误差具有如下的理论结果和数值算例。

1. 基于向后误差分析理论，修正的 *MGS* 直交化过程可等价描述为扰动矩阵的精确 QR 分解，即 $\mathbb{A} + \delta\mathbb{A}_{MGS} = \mathbb{Q}_{MGS}\mathbb{R}_{MGS}$，其中

$\delta\mathbb{A}_{MGS}$ 是扰动矩阵。舍入误差分析理论表明:

$$\|\delta\mathbb{A}_{MGS}\|_2 \le c_{m,n}\vartheta\|\mathbb{A}\|_2,$$
$$\|\mathbb{Q}_{MGS}^{\top}\mathbb{Q}_{MGS} - \mathbb{I}\|_2 \le c_{m,n}\vartheta\kappa_2(\mathbb{A}) + O((\vartheta\kappa_2(\mathbb{A}))^2),$$

其中 ϑ 是机器精度，$c_{m,n}$ 是绝对常数。

2. 类似地，传统的 *CGS* 直交化过程也可等价描述为 $\mathbb{A} + \delta\mathbb{A}_{CGS} = \mathbb{Q}_{CGS}\mathbb{R}_{CGS}$，其中 $\delta\mathbb{A}_{CGS}$ 是扰动矩阵。关于扰动矩阵的舍入误差分析结果依旧成立，即

$$\|\delta\mathbb{A}_{CGS}\|_2 \le c_{m,n}\vartheta\|\mathbb{A}\|_2$$

但是，所得的直交阵在直交方面的表现不再满足上面的第二条性质。

3. 为说明上述结论，我们考虑 25×15 阶范德蒙矩阵

$$\mathbb{A} = (p_i^{j-1}), \quad \text{其中} \quad p_i = i/25.$$

此实验摘录于 *N.J.Higham* 的*"Accuracy and Stability of Numerical Algorithms"* 第二版的第 *373* 页。我们有

$$\|\delta\mathbb{A}_{CGS}\|_2 = \|\mathbb{A} - \mathbb{Q}_{CGS}\mathbb{R}_{CGS}\|_2 = 5.0 \times 10^{-16},$$
$$\|\delta\mathbb{A}_{MGS}\|_2 = \|\mathbb{A} - \mathbb{Q}_{MGS}\mathbb{R}_{MGS}\|_2 = 1.0 \times 10^{-15}.$$

换言之，它们给出的 *QR* 乘积均很好地近似原始矩阵 $\mathbb{A}$；因此，它们是向后稳定的算法。换言之，即使数值得到的 $\mathbb{Q}_{\text{num}}$ 和 $\mathbb{R}_{\text{num}}$ 同真实结果的误差可能很大，但它们的乘积 $\mathbb{Q}_{\text{num}}\mathbb{R}_{\text{num}}$ 却神奇地非常准确地接近 $\mathbb{A}$。这是直交分解的一个优势之一。但是，计算结果

表明

$$\|\mathbb{Q}_{CGS}^{\top}\mathbb{Q}_{CGS}-\mathbb{I}\|_2=5.2,$$
$$\|\mathbb{Q}_{MGS}^{\top}\mathbb{Q}_{MGS}-\mathbb{I}\|_2=9.5\times10^{-9}.$$

换言之，两种方法在列向量的直交性方面有明显的差异。

4. 当矩阵的条件数非常恶劣的时候，上述两种 *GS* 直交化方法给出的对角线元素也存在明显的差异。设目标矩阵 $\mathbb{C}$ 是由一个元素快速变化的对角阵 $diag\{2^{-k}\}_{k=1}^{80}$ 相继左乘和右乘两个任意的直交阵而得到的。我们将两种方法给出的对角线元素绘制在图 *3.2.1* 中，其中方框表示 *CGS* 方法，圆圈表示 *MGS* 方法，星号表示真实的对角元素。

因此，在实际的数值计算中，所谓的 *GS* 直交化过程均使用修正的 *MGS* 方法。这个默认规则将不再赘述。

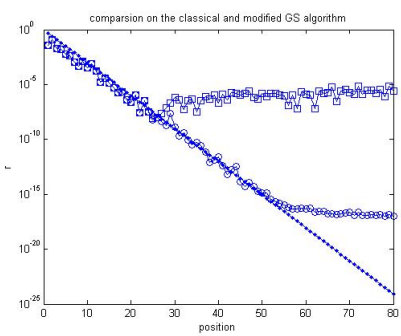

图 3.2.1: 矩阵 $\mathbb{U}diag\{2^{-k}\}_{k=1}^{80}\mathbb{V}^{\top}$ 的对角线元素比较。

✯ **说明 3.5.** 为避免 *GS* 直交化过程中因为除零而造成异常停机，矩阵 $\mathbb{A}$ 的前 r 列向量必须是线性无关的。若矩阵的前 r 列向量线性相

关，*GS* 直交化过程需要执行所谓的列交换，找到一个最大线性无关列向量组。对于一个列亏秩矩阵，我们可以在理论上找到相应的置换阵。但是，在数值实现上，这却是一个非常困难的任务，因为舍入误差的积累常常会造成"数值秩"的改变[vi]。相关的数值处理方法超出课程范围，此处略。因此，*GS* 直交化非常适于处理列满秩矩阵，对于亏秩矩阵的数值效果较差。

✎ **论题 3.3.** 利用两次 *Gram-Schmidt* 直交化过程，我们可以建立矩阵的直交分解。例如，矩阵具有**不完全直交分解**:

> *(B):* $\mathbb{A}_{m\times n} = \mathbb{Q}_{m\times r}\mathbb{R}_{r\times r}\mathbb{V}^{\top}_{n\times r}$, 其中 $\mathbb{R}$ 为 r 阶上三角阵，$\mathbb{Q}_{m\times r}$ 和 $\mathbb{V}_{n\times r}$ 均是列直交阵。

经过简单的列向量正交扩充，我们可给出矩阵的**完全直交分解**:

> *(C):* $\mathbb{A}_{m\times n} = \mathbb{H}_{m\times m}\tilde{\mathbb{R}}_{m\times n}\mathbb{K}^{\top}_{n\times n}$, 其中 $\tilde{\mathbb{R}}$ 是 r 阶上三角阵 $\mathbb{R}$ 的零扩充阵, $\mathbb{H}_{m\times m}$ 和 $\mathbb{K}_{n\times n}$ 均是直交方阵。

请注意：矩阵的完全直交分解是一个非常有用的分析工具。

尽管 MGS 直交化过程的数值稳定性有所加强，直交化后的列向量在正交性方面的数值表现，依旧令人不甚满意[vii]。在数值计算中，我们更多地会采用正交矩阵（如 Householder 镜像变换阵或 Givens 平面旋转阵）技术，来实现矩阵的 QR 分解。

[vi] 数值秩的概念容后给出。

[vii] 事实上，仅仅在这个指标上，修正的 MGS 方法的表现明显弱于 Householder 方法。

3.2.2 Householder 镜像变换

Householder 镜像变换阵非常著名。它最初出现在 Turnbull 和 Aitken (1932) 的书中，用于证明 Schur 矩阵分解的存在性。而后，它被 Householder 应用于矩阵的特征值计算 (1958)，并因此而得名。

定义 3.3. 设 $\boldsymbol{u}_n$ 是 n 维非零实向量，则称

$$\mathbb{H}_{n\times n} = \mathbb{I}_{n\times n} - b^{-1}\boldsymbol{u}_n\boldsymbol{u}_n^{\top}, \quad \text{其中 } b = \frac{1}{2}\|\boldsymbol{u}_n\|_2^2, \tag{3.2.2}$$

为 *Householder* 镜像变换阵。它是单位矩阵的秩一修正。

思考 3.3. 为计算一个高维向量 $\boldsymbol{u}$ 的长度，我们要小心“上溢”与“下溢”现象，需采用如下代码

```
1. m =max(abs(u));
2. y = u/m;
3. return  m * norm(y);
```

来增强算法的健壮性。

论题 3.4. 通常，一个 n 阶矩阵同向量的乘法共需 n^2 次乘除法运算。利用秩一修正矩阵在计算复杂度方面的优势，我们有

$$\mathbb{H}_{n\times n}\boldsymbol{g}_n = \boldsymbol{g}_n - b^{-1}(\boldsymbol{u}_n^{\top}\boldsymbol{g}_n)\boldsymbol{u}_n.$$

换言之，*Householder* 镜像变换阵同向量的乘积，只需 $2n+1$ 次乘除法运算。

Householder 镜像变换阵是对称的正交矩阵。简单计算可知，它具有重要的“镜像”效应：

$$\mathbb{H}\boldsymbol{u} = -\boldsymbol{u}, \quad \mathbb{H}\boldsymbol{g} = 0, \forall \boldsymbol{g} \perp \boldsymbol{u}.$$

这使得它成为重要的数值工具。Householder 镜像变换阵的核心价值主要体现在如下基本代数问题的实现过程中。

✍ **论题 3.5.** 设 $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)^\top$ 为一个已知的 n 维向量。我们能否左乘一个 *Householder* 镜像变换阵 $\mathbb{H}$，使得

$$\mathbb{H}_{n\times n}\boldsymbol{a} = (\alpha, 0, 0, \ldots, 0)^\top ?$$

记相应的算法为 $[\alpha, b] = \mathsf{householder}(\boldsymbol{a})$。参见下面的图文框。

在这个伪代码中，我们将镜面法向量 $\boldsymbol{u}$ 覆盖保存在原有向量 $\boldsymbol{a}$ 的位置上。不难发现，我们仅需改变 $\boldsymbol{a}$ 的第一个分量。在输出列表中，我们新增了两个浮点数空间：其一是带符号的 α，对应向量 $\boldsymbol{a}$ 的长度。其二是 b，对应 Householder 镜像变换矩阵的参数。事实上，b 也可以不保留；我们之所以保留它，是为了后续计算的方便。这就是所谓的空间换取速度的策略。我们无需直接保存这个 Householder 镜像变换阵 $\mathbb{H}$，因为它可以由记录的 b 和 $\boldsymbol{u}$ 快速给出。事实上，在后续的计算中，矩阵 $\mathbb{H}$ 某个向量的乘法可通过单位矩阵的秩一修正特点来实现。

1. $\alpha := -\mathrm{sgn}(a_1)\|\boldsymbol{a}\|_2$;
2. $b := \alpha^2 - \alpha a_1$;
3. $a_1 := a_1 - \alpha$.

我们要指出：上述 α 的选取策略，可以保证镜像变换阵中的 b 具有更大的绝对值，使得相应的舍入误差可以得到有效的控制。

✯ **说明 3.6.** *Wilkinson* 指出：这种算法具有非常好的数值稳定性

$$\|\mathbb{H}_{num} - \mathbb{H}\|_2 \leq C\vartheta,$$

其中 C 是绝对常数，ϑ 是机器精度。

❀ **思考 3.4.** 若将论题 *3.5* 中的目标推广到复数域，我们需要解决哪些问题？

✍ **论题 3.6.** 利用 *Householder* 镜像变换阵[viii]的左乘，我们可以将矩阵 $\mathbb{A}_{m\times n}$ 逐步变换到上三角（梯形）矩阵 $\mathbb{R}_{m\times n}$，即

$$\mathbb{H}_{\min(m,n)}\cdots\mathbb{H}_1\mathbb{A}=\mathbb{R}.$$

对于列满秩矩阵，基本实现方法如下:

1. *For* $k=1,2,\ldots,n$, *Do*
2. 计算 $m-k+1$ 阶矩阵 $\mathbb{H}_k$ 的主要信息，即 $[\alpha,b]=$*householder*$(\mathbb{A}(k:m,k))$;
3. 计算矩阵乘积: $\mathbb{H}_k\mathbb{A}(k:m,k+1:n)$;
4. *Enddo*

请注意: 我们使用了数据覆盖技术。换言之，*Householder* 镜像矩阵的镜面法向量 $\boldsymbol{u}$，被保存在原有的相应位置。变换后的对角线元素记录在 α 中，我们需要开辟一个数组来保存它们。若还要记录参数 b，我们还需开辟一个数组。

第 *3* 步的矩阵乘积运算，可以利用论题 *3.4* 的公式进行。

❀ **思考 3.5.** 如何得到最终的直交阵？

✯ **说明 3.7.** 基于 *Householder* 镜像变换阵的直交化过程,对于舍入误差的表现是非常完美和稳定的。它也是一种向后稳定的算法。对于列满秩矩阵, *Householder* 镜像变换方法比 *MGS* 方法更好, 因为 *Householder* 镜像变换方法给出的 $\mathbb{Q}_{\rm num}$ 具有更好的列正交性。

[viii] 为叙述方便，我们也将单位阵视为一个 Householder 变换阵。

✯ **说明 3.8.** 为更好地控制舍入误差表现，我们通常以余下列向量中具有最大长度的列向量，作为 *Householder* 镜像变换的主列向量。事实上，借助这种策略，*Householder* 镜像变换也可以较好地应用于列亏秩矩阵的直交化过程。

3.2.3 Givens 平面旋转变换

若 $\mathbb{A}$ 是一个稀疏矩阵, Givens 平面旋转阵将会更加便捷。同 Householder 镜像变换阵方法相比，整体的计算复杂度会明显下降。

Givens 平面旋转阵同 Householder 镜像变换阵，具有显著的差别。它通常是非对称的，而且是单位矩阵的秩二修正。

定义 3.4. 记 $c = \cos\theta$ 和 $s = \sin\theta$. 称正交矩阵

$$\mathbb{G}(i,j;\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \tag{3.2.3}$$

为 (i,j) 平面上的 *Givens* 平面旋转阵。

相应的基本代数操作是如下的问题。

论题 3.7. 如何构造一个 *Givens* 平面旋转阵 $\mathbb{G}(i,j;\theta)$，将

$$\boldsymbol{a} = (\cdots, x_i, \cdots, x_j, \cdots)^\top$$

的第 j 个分量变换为零？

数值实现是非常简单的，记相应的算法为 $[c,s]=\mathsf{givens}(i,j,\boldsymbol{a})$。在下面的图文框中，我们给出算法的伪代码。

1. 若 $x_j=0$，则 $c=1,s=0$；
2. 若 $|x_j|\geq|x_i|$，通常取 $s>0$，即

$$t=\frac{x_i}{x_j},s=\frac{1}{\sqrt{1+t^2}},c=st;$$

3. 若 $|x_j|<|x_i|$，通常取 $c>0$，即

$$t=\frac{x_j}{x_i},c=\frac{1}{\sqrt{1+t^2}},s=ct;$$

其中，最后两步的分类处理是为了保证 $|t|\leq 1$，使得舍入误差表现更好。Wilkinson 指出上述算法有很好的数值稳定性，即

$$|c_{num}-c|+|s_{num}-s|\leq C\vartheta,$$

其中 C 是绝对常数，ϑ 是机器精度。

✯ **说明 3.9.** 为记录平面旋转阵的信息，除位置信息 i 和 j 之外，我们还要记录角度信息，即 c 和 s 两个数。事实上，利用 *Stewart (1976)* 提出的技术，我们可以只存储一个浮点数 ρ，并将它存储在原有数据 x_j 的位置。相应的伪代码是

1. 若 $c=0$，令 $\rho=1$；
2. 若 $|s|<|c|$，令 $\rho=sgn(c)s/2$；
3. 若 $|s|\geq|c|$，令 $\rho=2sgn(s)/c$.

事实上，第 *2* 步中的 $|\rho|\leq 1/2$，第 *3* 步中的 $|\rho|\geq 2$。因此，我们可以采用如下的互逆操作过程：

1. 若 $\rho = 1$，令 $c = 0, s = 1$;
2. 若 $|\rho| < 1$，令 $s = 2\rho, c = \sqrt{1 - s^2}$;
3. 若 $|\rho| > 1$，令 $c = \rho/2, s = \sqrt{1 - c^2}$.

由 ρ 恢复 c 和 s 的值。

✍ **论题 3.8.** 如何利用 *Givens* 平面旋转技术，实现论题 *3.5* 中的数值目标？请给出相应的实现策略。

❀ **思考 3.6.** 至此，我们可以构造一系列 *Givens* 平面旋转阵

$$\mathbb{G}(i_1, j_1), \ldots, \mathbb{G}(i_r, j_r),$$

或者一个 *Householder* 镜像变换阵，将一个 n 维向量 $\boldsymbol{a}$ 变换到仅首个分量非零。它们之间应存在一定的关系。但是，*Householder* 镜像变换阵不可能由若干个 *Givens* 平面旋转阵的乘积来表示，因为

$$\det \mathbb{G} = 1, \quad \det \mathbb{H} = -1.$$

请注意：任意的 *Givens* 平面旋转阵可分解为两个 *Householder* 镜像变换阵的乘积。

3.2.4 小结

✯ **说明 3.10.** 让我们比较一下上述三种直交化方法的计算复杂度。若矩阵是稠密的，*Hoseholder* 镜像变换法所需的乘除法次数是 *Givens* 平面旋转变换方法的一半。事实上，前者就是为了减少后者的计算量而提出的。

假设矩阵是列满秩（$m \geq n = r$）的，相应的乘除法运算次数为

$$N_{opt}^{\text{House}} \approx \sum_{k=1}^{n} 2(n+1-k)(m+1-k) \approx mn^2 - \frac{1}{3}n^3, \tag{3.2.4a}$$

$$N_{opt}^{\text{GS}} \approx \sum_{k=1}^{n} 2(k-1)m \approx mn^2. \tag{3.2.4b}$$

因此，*Hoseholder* 镜像变换法的计算复杂度稍微低于 *MGS* 法。因此，在大多数情况下，无论是在计算复杂度，还是在舍入误差方面，*Householder* 镜像变换方法都是首选方法。只有当矩阵 $\mathbb{A}$ 足够稀疏的时候，*Givens* 平面旋转变换才会在计算效率上面占有优势。

✯ **说明 3.11.** 对于列满秩矩阵，若锁定上三角阵 $\mathbb{U}$ 的对角线元素符号，则它的 *QR* 分解是唯一的。这是一个非常重要的结论，详见教科书习题 *7.11.* 因此，上述三种直交化方法给出的直交列向量要么相同，要么相反而已。

✯ **说明 3.12.** 当 $m = n$ 时，*Householder* 镜像变换方法也可用于线性方程组的求解。但是，我们很少使用这种方法，因为它的乘除法运算次数是高斯消元法的两倍。同时，基于列主元策略的高斯消元方法已经可以给出很好的数值结果。

3.3 最小二乘解的各种表示

利用前面的各种直交化实现过程，我们可以显式建立最小二乘解的多种表达方式，并由此给出相应的数值求解方法。在计算复杂度和舍入误差表现方面，它们各具特色。

3.3.1 最小二乘解的基本结构

利用正交（或者酉）变换不改变向量长度的基本性质，我们可以将复杂的最小二乘问题，等价地转化为简单的最小二乘问题。

论题 3.9. 设 $r = rank(\mathbb{A}_{m\times n})$。利用完全直交分解 *(C)*，我们可以建立最小二乘解的一般结构

$$\boxed{\boldsymbol{x}_{LS} = \mathbb{K}\begin{bmatrix}\mathbb{R}^{-1}\boldsymbol{g}\\ \boldsymbol{y}\end{bmatrix}, \quad \text{其中 } \mathbb{H}^{\top}\boldsymbol{b} = \begin{bmatrix}\boldsymbol{g}\\ \boldsymbol{h}\end{bmatrix},}$$

其中 $\boldsymbol{y}$ 是任意的 $n-r$ 维向量。当 $\boldsymbol{y}=0$ 时，$\boldsymbol{x}_{LS}$ 就是极小最小二乘解。最小二乘解的残量大小为 $\|\boldsymbol{h}\|_2$。

这个结果具有重要的理论意义，它给出了最小二乘解的完整结构。在实际的数值计算中，这个表达式并不理想，因为完全直交分解 (C) 包含一个列向量直交扩张过程，相应的数值实现是较为困难的。事实上，直交扩张的结果对于最小二乘解的计算，也没有任何的实质贡献。因此，我们希望给出最小二乘解的简化计算公式。

3.3.2 Gram-Schmidt 直交化方法

假设系数矩阵 $\mathbb{A}_{m\times n}$ 的前 r 个列向量是线性无关的，其中 $r = \text{rank}(\mathbb{A}) > 0$。

论题 3.10. 利用不完全直交分解 *(B)*，可以给出极小最小二乘解

$$\boxed{\boldsymbol{x}_{LS} = \mathbb{V}_{r\times n}\mathbb{R}_{r\times r}^{-1}\mathbb{Q}_{m\times r}^{\top}\boldsymbol{b}.}$$

不完全直交分解需要两次 *GS* 正交化过程。直接利用 *Gram-SchmildtS* 直交化过程 *(A)*，可以给出极小最小二乘解

$$\boxed{\boldsymbol{x}_{LS} = \mathbb{U}_{r\times n}^{\top}(\mathbb{U}_{r\times n}\mathbb{U}_{r\times n}^{\top})^{-1}\mathbb{Q}_{m\times r}^{\top}\boldsymbol{b}.}$$

若矩阵 $\mathbb{A}$ 是列满秩的，我们有更简洁的答案

$$\boxed{\boldsymbol{x}_{LS} = \mathbb{U}_{n\times n}^{-1}\mathbb{Q}_{m\times n}^{\top}\boldsymbol{b}.}$$

事实上，上述三个公式中的矩阵都是 $\mathbb{A}^{\dagger}$。

✯ **说明 3.13.** 在上述最小二乘解的三个表达式中，我们都需要计算 $\mathbb{Q}^{\top}\boldsymbol{b}$。常用的处理方法是将 $\boldsymbol{b}$ 融入到增广矩阵 $[\mathbb{A}|\boldsymbol{b}]$ 中，对增广矩阵执行相应的直交化操作，我们可以在增广矩阵的最后一列提取相关的信息。若利用得到的直交阵进行矩阵计算，舍入误差方面的表现要差一些。这可解释为计算次序对舍入误差的影响。

✯ **说明 3.14.** 在上述表达式中，我们没有强调矩阵 $\mathbb{A}$ 的列满秩。因为它们对于亏秩矩阵也是理论上可行的。但是，对于亏秩矩阵，相应的直交化过程是不稳定的。数值结果可能产生巨大偏差，甚至计算过程意外停机。

3.3.3 直交矩阵变换

✎ **论题 3.11.** 采用 *Householder* 镜像变换方法，对增广矩阵 $[\mathbb{A}|\boldsymbol{b}]$ 执行逐次消元，我们可得如下结果

$$\underbrace{\mathbb{H}_r\cdots\mathbb{H}_2\mathbb{H}_1}_{\mathbb{Q}^\top}[\,\mathbb{A}\mid\boldsymbol{b}\,]=\begin{bmatrix}\mathbb{R} & \mathbb{R}_1 & \mathbb{Q}_1^\top\boldsymbol{b}\\ \mathbb{O} & \mathbb{O} & \mathbb{Q}_2^\top\boldsymbol{b}\end{bmatrix}=\begin{bmatrix}\mathbb{U}_{r\times n} & \mathbb{Q}_1^\top\boldsymbol{b}\\ \mathbb{O} & \mathbb{Q}_2^\top\boldsymbol{b}\end{bmatrix},$$

其中 $\mathbb{R}$ 是一个 r 阶的可逆上三角矩阵，$\mathbb{U}_{r\times n}$ 是一个行满秩的上梯形矩阵，$\mathbb{Q}=[\mathbb{Q}_1\ \mathbb{Q}_2]$ 为 m 阶的正交阵。它可以给出一个最小二乘解

$$\boldsymbol{x}_{LS}=\mathbb{R}^{-1}\mathbb{Q}_1^\top\boldsymbol{b}, \tag{3.3.5}$$

右下角 $\mathbb{Q}_2^\top\boldsymbol{b}$ 的长度为最小二乘解的对应残量大小。

1. 若矩阵 $\mathbb{A}$ 是列满秩的，则 $\mathbb{R}_1$ 是空的，此时的 (3.3.5) 是唯一的极小最小二乘解。
2. 若矩阵 $\mathbb{A}$ 是列亏秩的，则 $\mathbb{R}_1$ 是非空的，此时的 (3.3.5) 不一定是极小最小二乘解。若要得到极小最小二乘解，我们还需施行右侧的 *Householder* 变换。详略。

✯ **说明 3.15.** 引进主列向量策略，我们可以增强 *Householder* 直交变换方法的数值稳定性。这个策略可以使 *Householder* 方法处理亏秩矩阵的最小二乘问题求解。

✯ **说明 3.16.** *Givens* 平面旋转方法和 *Householder* 镜像变换方法的实现是类似的。

3.4 奇异值分解

奇异值分解是 Schur 分解的推广，在理论分析和实际应用（例如信息处理、多元统计分析等工程技术领域）中都具有非常重要的作用。在 Matlab 中，相应的命令是 svd()。

定理 3.8. 令 $p = \min(m,n)$。设 $\mathbb{A} \in \mathbb{R}^{m\times n}$，则存在两个直交阵 $\mathbb{U} \in \mathbb{R}^{m\times m}$ 和 $\mathbb{V} \in \mathbb{R}^{n\times n}$，使得

$$\boxed{\mathbb{A} = \mathbb{U}_{m\times m}\mathbb{D}_{m\times n}\mathbb{V}^{\top}_{n\times n},}$$

其中 $\mathbb{D} = generaldiag(\sigma_1, \sigma_2, \ldots, \sigma_p)$ 是广义的对角阵。

称 σ_i 为奇异值，称 $\mathbb{U}$ 中的第 i 列向量 $\boldsymbol{u}_i$ 为左奇异向量，称 $\mathbb{V}$ 中第 i 列 $\boldsymbol{v}_i$ 为右奇异向量，因为

$$\boxed{\boldsymbol{u}_i^{\top}\mathbb{A} = \sigma_i\boldsymbol{v}_i^{\top}, \quad \mathbb{A}\boldsymbol{v}_i = \sigma\boldsymbol{u}_i.}$$

通常，奇异值是按降序排列的，即 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$，其中 $r = \mathrm{rank}(\mathbb{A})$；余下的奇异值均为零，即 $\sigma_{r+1} = \sigma_{r+2} = \cdots \sigma_p = 0$。

论题 3.12. 几何含义：任意矩阵均可视为一个线性变换。奇异值分解理论表明，我们可以建立两个正交坐标系，将相应的线性变换描述为沿着每个坐标轴的变化。它反应了刚体线性变形仅仅对应着所谓的旋转和拉伸。

论题 3.13. 矩阵秩，值域、核空间，以及秩一展开等。

$$\boxed{\begin{array}{l} 1.\ range(\mathbb{A}) = span\{\boldsymbol{u}_i\}_{i=1}^{r}, \\ 2.\ ker(\mathbb{A}) = span\{\boldsymbol{v}_i\}_{i=r+1}^{n}, \\ 3.\ \mathbb{A} = \sum_{i=1}^{r}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^{\top}. \end{array}}$$

论题 3.14. 奇异值刻画了一个矩阵到低秩矩阵集合之间的距离。设 $\mathbb{A}$ 的秩为 r，有

$$\min_{\mathrm{rank}(\mathbb{B})=k} \|\mathbb{A} - \mathbb{B}\|_2 = \|\mathbb{A} - \sum_{i=1}^{k}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^{T}\|_2 = \sigma_{k+1}, \quad 若\ k < r.$$

若数值计算的奇异值 σ_k 和 σ_{k+1} 分别处于机器精度的两侧，则称 k 是这个矩阵的**数值秩**。我们希望数值算法给出的数值秩接近真实秩，相应的最小二乘解计算过程具有足够的可靠性。

✎ **论题 3.15.** 设 $\mathbb{A} \in \mathbb{R}^{m\times n}$ 有奇异值分解 $\mathbb{A} = \mathbb{U}\mathbb{D}\mathbb{V}^\top$，则 (3.0.1) 的极小最小二乘解可表示为

$$\boxed{\boldsymbol{x}_{LS} = \mathbb{V}\mathbb{D}^\dagger\mathbb{U}^\top\boldsymbol{b},}$$

其中的矩阵就是 $\mathbb{A}^\dagger$。这种计算方法具有数值稳定性强、适用于亏秩矩阵等优点。但是，矩阵的奇异值分解过程需要耗费更多的机时。因此，只有当问题的病态非常严重时，我们才会采用奇异值分解方法求解最小二乘解。

✯ **说明 3.17.** 奇异值分解理论在理论上非常漂亮，但是相应的数值计算却是相对非常困难。常用的方法是：首先采用 *Householder* 变换法，将矩阵变换为双对角线上三角阵，然后再通过一个迭代过程（类似于求解特征值的 *QR* 方法），将其近似转化为一个对角矩阵。即使没有任何舍入误差，我们也不能在有限步数内得到精确的奇异值分解。具体内容可参阅 *Golub* 和 *Kahan* 在 *20* 世纪 *60* 年代的工作；详略。

✯ **说明 3.18.** 奇异值分解还是一个重要的矩阵分析工具。例如，对于 $m \times n$ 阶矩阵 $\mathbb{A}$，我们可以建立

$$\mathbb{A}^\dagger = \lim_{a\to 0}\left[(\mathbb{A}^\top\mathbb{A} + a^2\mathbb{I})^{-1}\mathbb{A}^\top\right] = \lim_{a\to 0}\left[\mathbb{A}^\top(\mathbb{A}\mathbb{A}^\top + a^2\mathbb{I})^{-1}\right].$$

及其各种结论。

✯ **说明 3.19.** 奇异值分解在图像压缩中也有很好的应用价值。通常，一个图像可以用矩阵来表示。事实上，一个图像的主要结构通常是

有限的，相应的数学解读是：具有重要价值的主奇异值个数远远小于矩阵的阶数。利用前 k 个主奇异值，对矩阵 $\mathbb{A}$ 的秩一展开做截断，我们就可以给出矩阵 $\mathbb{A}$ 的漂亮近似。此时，我们只需记录前 k 个主奇异值 σ_i，及其相应的奇异向量 $\boldsymbol{u}_i$ 和 $\boldsymbol{v}_i$，从而将图像的数据存储量从 mn 下降到 $(m+n+1)k$。

在图 *3.4.2* 中，我们应用上述策略对 *Matlab* 系统自带的小丑图，进行了压缩与恢复。在 *Matlab* 中，相应的实现过程非常简单，即

```
1. load clown.mat;
2. [U,S,V]=svd(X);
3. colormap('gray'); image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

当 k 适当大（右下角的子图）时，恢复后的小丑图像已经同原始图像没有明显的差别了。

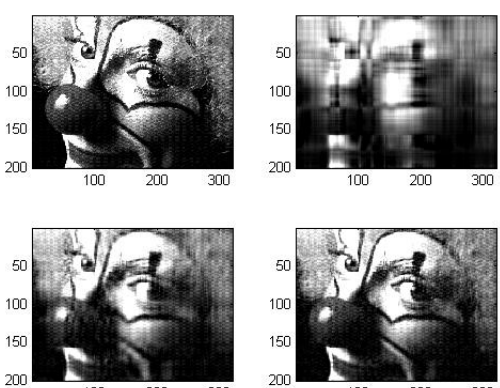

图 3.4.2: 小丑图的压缩与恢复：左上角为原图，余下三个为 $k = 5, 10, 15$.

3.5 离散数据拟合

在数据拟合或者线性回归等实际问题中，某个真实函数 $\phi(x)$ 常常近似为如下的简单模型或者经验公式

$$y(x) = \sum_{j=0}^{n} \alpha_j \phi_j(x), \tag{3.5.6}$$

其中 $\{\phi_j(x)\}_{j=0}^n$ 是已知的线性无关基函数，$\{\alpha_j\}_{j=0}^n$ 是待定的参数，需要从大量的离散数据 $\{(x_i, y_i)\}_{i=1:m}$ 中挖掘。上述目标可以表示为一个线性的最小二乘问题

$$y_i = \sum_{j=0}^{n} \alpha_j \phi_j(x_i), \quad i = 1:m.$$

相应的最小二乘解可利用前面的各种方法给出。

论题 3.16. 线性回归。

当利用法方程组求解这个最小二乘问题时，我们希望法方程组的系数矩阵是非奇异的，或者最小二乘问题中的系数矩阵是列满秩的。

论题 3.17. 函数的最佳平方逼近问题与离散数据的最小二乘问题具有密切的联系。由 $\{\phi_j(x)\}_{j=0}^n$ 的线性无关性，可知最佳平方逼近问题所对应的法方程组是对称正定的，有唯一解。那么，针对同一个模型，离散数据的最小二乘问题对应的法方程组，是否也一定可逆，具有唯一解呢？这个目标不是永远成立的。为保障最小二乘解的唯一性，我们需要所谓的 *Haar* 条件:

对不全为零的参数组 $\{\beta_j\}_{j=0}^n$，方程

$$g(x) = \sum_{j=0}^{n} \beta_j \phi_j(x)$$

的根不超过 n 个。

由 *Haar* 条件可知，多项式数据拟合总是唯一存在的。

第 4 章
矩阵特征值问题

在结构力学、电力网络、量子化学、以及理论物理学中，矩阵特征值问题应用广泛。设 $\mathbb{A}$ 是已知的 n 阶矩阵，相应的特征值问题是

$$\mathbb{A}\boldsymbol{x} = \lambda\boldsymbol{x}, \quad \boldsymbol{x} \neq 0,$$

其中 $(\lambda, \boldsymbol{x})$ 称为特征值和（右）特征向量。矩阵特征值问题是一个包含非线性现象的简单线性代数问题，相应的数值计算存在更多的困难。

4.1 特征值问题的敏感度分析

4.1.1 特征值问题的相关知识

论题 4.1. 设 $\mathbb{A}$ 是一个 n 阶方阵，其特征多项式

$$f(\lambda) \equiv \det(\lambda\mathbb{I} - \mathbb{A}) = (\lambda - \lambda_1)^{n_1}(\lambda - \lambda_2)^{n_2}\cdots(\lambda - \lambda_r)^{n_r} \qquad (4.1.1)$$

是 $n_1 + n_2 + \cdots + n_r = n$ 次多项式，其中 λ_i 是特征值。即使矩阵 $\mathbb{A}$ 是实的，它的特征值和特征向量也可能是复数的。对于特征值 λ_i，n_i 为相应的代数重数，而

$$\gamma_i = n - rank(\lambda_i\mathbb{I} - \mathbb{A})$$

描述了特征子空间的维数，称为相应的几何重数。

论题 4.2. 首项系数为 1 且使 $p(\mathbb{A}) = 0$ 的最低次数多项式

$$p(\lambda) = (\lambda - \lambda_1)^{\ell_1}(\lambda - \lambda_2)^{\ell_2}\cdots(\lambda - \lambda_r)^{\ell_r}, \quad 1 \leq \ell_i \leq n_i, \qquad (4.1.2)$$

称为 $\mathbb{A}$ 的最小多项式，其中 $V_i = \ker((\mathbb{A} - \lambda_i \mathbb{I})^{\ell_i})$ 是维数为 n_i 的 $\mathbb{A}$-不变子空间。

论题 4.3. 若存在可逆矩阵 $\mathbb{X}$，使得 $\mathbb{B} = \mathbb{X}^{-1}\mathbb{A}\mathbb{X}$，则称 $\mathbb{A}$ 和 $\mathbb{B}$ 是相似的。相似矩阵具有相同的特征多项式。

论题 4.4. 矩阵 $\mathbb{A}$ 同 $\mathbb{A}^{\top}$ 的特征值是相同的。

论题 4.5. 矩阵 $\mathbb{A}\mathbb{B}$ 和 $\mathbb{B}\mathbb{A}$ 的非零特征值是相同的。

论题 4.6. 作为重要的矩阵分析工具之一，*Jordan* 分解可以给出所有的特征信息，并清楚地判定特征向量的亏损情况。非亏损矩阵就是可对角化矩阵，相应的代数重数与几何重数都是相等的。但是，基于 *Jordan* 分解的数值方法，大多都是不稳定的。

论题 4.7. *Schur* 分解理论表明：任意的矩阵都可通过酉相似变换到上三角阵。更常用的是实数域的 *Schur* 分解定理，即任意的实矩阵都可通过正交矩阵相似变换到块上三角矩阵，其中对角线的矩阵块阶数至多为 *2*。这种分解方式在数值上更易实现。

特征值的近似程度可以直接用两者的距离来刻画。众所周知，特征向量的非零数乘依旧是特征向量。因此，关于特征向量的近似程度，应该用相应子空间的距离来刻画。

定义 4.1. 设 $\mathcal{P}$ 和 $\mathcal{Q}$ 是维数相同的两个子空间，其相应的子空间正交投影可分别表示为幂等矩阵 $\mathbb{P}$ 和 $\mathbb{Q}$，则两个子空间 $\mathcal{P}$ 与 $\mathcal{Q}$ 的距离可定义为

$$\operatorname{dist}(\mathcal{P}, \mathcal{Q}) = \|\mathbb{P} - \mathbb{Q}\|_2 = \left\{1 - [\sigma_{\min}(\mathbb{P}^{\mathrm{H}}\mathbb{Q})]^2\right\}^{1/2}, \tag{4.1.3}$$

其中 $\sigma_{\min}(\mathbb{P}^{\mathrm{H}}\mathbb{Q})$ 是矩阵 $\mathbb{P}^{\mathrm{H}}\mathbb{Q}$ 的最小正奇异值。

考虑一个简单的实例。设 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 是两个单位向量，对应的两个子

空间和正交投影矩阵分别是

$$\mathcal{P} = \mathrm{span}(\boldsymbol{x}),\ \mathbb{P} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{H}};\quad \mathcal{Q} = \mathrm{span}(\boldsymbol{y}),\ \mathbb{Q} = \boldsymbol{y}\boldsymbol{y}^{\mathrm{H}}.$$

由上述定义可知，两个空间的距离就是

$$\mathrm{dist}(\mathcal{P}, \mathcal{Q}) = (1 - \cos^2\theta)^{1/2} = |\sin\theta|,$$

其中 θ 是两个向量的夹角。换言之，若称两个一维子空间非常靠近，是指相应的基向量夹角非常接近零。

❀ **思考 4.1.** 请问：若 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 等长，$\|\boldsymbol{x} - \boldsymbol{y}\|_2$ 同它们的夹角 θ 有何关系？

4.1.2 特征值的简单估计

矩阵元素的分布信息，可以给出特征值的基本分布情况。这是矩阵理论的主要内容。例如，所有的特征值都落在一个圆盘内，其圆心为原点，半径是矩阵的谱半径

$$\varrho(\mathbb{A}) \leq \|\mathbb{A}\|,$$

其中 $\|\mathbb{A}\|$ 是任意的矩阵范数。更精细的结果还有著名的圆盘定理。

定理 4.1. 【*Gerschgorin* 第一圆盘定理】矩阵 $\mathbb{A} = (a_{ij})_{n\times n}$ 的任意特征值必落在某个圆盘 S_i 内，其中

$$S_i = \{\lambda\colon |\lambda - a_{ii}| \leq \sum_{j\neq i} |a_{ij}|\},\quad i = 1:n.$$

证明：略。 □

定理 4.2. 【*Gerschgorin* 第二圆盘定理】设 $\mathbb{A} = (a_{ij})_{n\times n}$ 的 n 个圆盘有 m 个构成了一个联通域，并与其它 $n-m$ 个圆盘严格分离，则这个联通域中恰好有 m 个特征值。

证明：略。 □

若引进适当的对角阵 $\mathbb{D}$, 再使用圆盘定理估计 $\mathbb{D}^{-1}\mathbb{A}\mathbb{D}$ 的特征值, 我们可以更加准确地估计 $\mathbb{A}$ 特征值。

4.1.3 敏感度分析

矩阵特征值关于矩阵元素的变化[i]是连续依赖的, 但其敏感程度却常常呈现出巨大的区别。例如，让我们考虑一个简单的 n 阶矩阵

$$\begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & \cdots & \cdots & \cdots & \\ & & & 0 & 1 \\ 0 & & & & 0 \end{bmatrix},$$

其特征值为零。若左下角位置发生一个扰动 $\varepsilon > 0$, 则特征值将变成 $\sqrt[n]{\varepsilon}$, 数值扰动的影响很大。随着扰动位置向上方和右侧漂移，数值扰动的影响越来越弱。甚至，当 ε 出现在对角线上方的任意位置，矩阵的特征值保持不变。

对于一个矩阵特征值问题，相应的特征敏感程度主要有两种刻画方式。下面，我们给出常见的定义方式，具体的理论证明过程略。

[i]这可以利用复变函数中的留数定理证明，此处略。

整体条件数

定理 4.3. [*Bauer-Fike* 定理] 设 $\mathbb{A}$ 和 $\mathbb{B}$ 为两个已知的复矩阵，其中 $\mathbb{A}$ 可通过矩阵 $\mathbb{Q}$ 相似变换为一个对角阵。对于 $\mathbb{B}$ 的任意特征值 $\mu \in \lambda(\mathbb{B})$，均存在 $\mathbb{A}$ 的一个特征值 $\lambda \in \lambda(\mathbb{A})$，使得

$$|\lambda - \mu| \leq \|\mathbb{Q}^{-1}\|\|\mathbb{Q}\|\|\mathbb{A} - \mathbb{B}\|,$$

其中 $\|\cdot\|$ 是任意的从属矩阵范数。

证明：简单的特征信息描述，略。 □

定义 4.2. 关于矩阵 $\mathbb{A}$ 的特征值整体条件数是

$$\nu(\mathbb{A}) = \inf_{\mathbb{Q}\in\mathcal{D}_{\mathbb{A}}} \|\mathbb{Q}\|\|\mathbb{Q}^{-1}\|, \tag{4.1.4}$$

其中集合 $\mathcal{D}_{\mathbb{A}}$ 包含可使 $\mathbb{A}$ 对角化的所有相似矩阵。

定理 4.4. 设 $(\lambda, \boldsymbol{x})$ 是非亏损矩阵 $\mathbb{A}$ 的一个近似特征信息对，记

$$\boldsymbol{r} = \mathbb{A}\boldsymbol{x} - \lambda\boldsymbol{x},$$

则存在一个特征值 $\lambda_i \in \lambda(\mathbb{A})$，使得

$$|\lambda - \lambda_i| \leq \nu(\mathbb{A})\frac{\|\boldsymbol{r}\|_2}{\|\boldsymbol{x}\|_2}.$$

证明：由 $\boldsymbol{r}$ 的定义可知

$$\Big[\mathbb{A} - \frac{\boldsymbol{r}\boldsymbol{x}^H}{\|\boldsymbol{x}\|_2^2} - \lambda\mathbb{I}\Big]\boldsymbol{x} = 0,$$

即 $(\lambda, \boldsymbol{x})$ 是扰动矩阵的特征信息对。由 Bauer-Fike 定理，即证。 □

说明 4.1. *Bauer-Fike* 定理的结果可推广到亏损矩阵，具体结果同 *Jordan* 分解中的 *Jordan* 块阶数有关。尽管如此，上述概念依旧是有效的。

Wilinson 条件数

矩阵特征值的扰动敏感程度，还同特征信息的局部情况有关。为说明这个现象，我们考虑简单的单特征值情况，即相应的特征子空间是一维的。此时的特征信息关于矩阵元素是连续可导的。对方程

$$(\mathbb{A}+\varepsilon\mathbb{E})\boldsymbol{x}(\varepsilon)=\lambda(\varepsilon)\boldsymbol{x}(\varepsilon)$$

关于变量 ε 求导数，其中 $\mathbb{E}$ 是任意给定的矩阵。显然，有 $\lambda(0)=\lambda$ 和 $\boldsymbol{x}(0)=\boldsymbol{x}$，对应 $\mathbb{A}$ 的单特征值信息。简单推导 $\varepsilon=0$ 的导数信息，我们可得如下定义。

定义 4.3. 设 λ 是 $\mathbb{A}$ 的单特征值，相应的局部特征值条件数是

$$W(\lambda;\mathbb{A})=\frac{1}{|\boldsymbol{x}^{\mathrm{H}}\boldsymbol{y}|},\tag{4.1.5}$$

其中 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 分别是相应的单位左特征向量和单位右特征向量。

补充说明

上述两种条件数均为酉相似变换下的不变量。因此，在计算矩阵的特征值时，我们可以放心地将矩阵进行酉相似变换，将其约化为某些简单的结构。

论题 4.8. 无论是整体刻画还是局部刻画，上述两种条件数均表明：对称矩阵的特征值问题都是良态的。矩阵特征值问题的病态与线性方程组问题的病态是两个完全不同的概念。

说明 4.2. 若两个单特征值非常靠近，或者重特征值的特征子空间是亏损的，相应的特征问题通常是病态的。

说明 4.3. 特征向量的敏感性比较复杂。它不仅同相应的特征值条件数有关，也和这个特征值是否重根，或与其他特征值的分离程度有

关。但是，将敏感的特征向量放在一起，形成的特征子空间却可以是不敏感的。具体讨论超出课程设置，可参见 *Wilkinson* 的专著。

4.2 幂法

无论是设计思想，还是数值实现方法，幂法都非常简单。因此，它被广泛地应用于大型稀疏矩阵的主特征值计算。事实上，它也是很多特征值计算方法的设计起点。

4.2.1 正幂法

✍ **论题 4.9.** 幂法的设计基于如下的简单事实：在算子 $\mathbb{A}$（或矩阵左乘）的不断作用下，初始向量所包含的不同特征成分将呈现出明显不同的几何增长速度。利用这种差异，我们可以将（对应最大模特征值的）主特征信息从其他特征信息中筛选和分离出来。

基本公式表达如下：设矩阵 $\mathbb{A}$ 具有完备的特征向量系，或者仅仅具有线性初等因子。对任意的非零向量 $\boldsymbol{v}_0$，我们有

$$\boxed{\mathbb{A}^k\boldsymbol{v}_0 = \sum_{1\le j\le n}\alpha_j\lambda_j^n\boldsymbol{x}_j = \lambda_1^n\sum_{1\le j\le n}\alpha_j\left(\frac{\lambda_j}{\lambda_1}\right)^n\boldsymbol{x}_j,}$$

其中 λ_1 是可完全分离的主（实）特征值，即

$$|\lambda_1| > |\lambda_2| \ge |\lambda_3| \ge \cdots \ge |\lambda_n|.$$

这个公式表明 $\mathbb{A}^k\boldsymbol{v}_0/\lambda_1^k$ 可以用来近似矩阵 $\mathbb{A}$ 的主特征信息。

然而，这个非常直接的想法在实际计算中是行不通的。主要原因如下：其一，待解的主特征信息是未知的；其二，直接计算高幂次矩阵，将

造成庞大的工作量, 并破坏矩阵的稀疏性, 引入严重的舍入误差积累。出于数值操作的考量, 我们需要将上述公式进行适当的变形。

✎ **论题 4.10.** 主要的想法是引进递推公式, 并在每步迭代中进行适当的单位化, 以避免数值计算出现上下溢出。幂法的基本结构为

$$\boxed{\boldsymbol{u}_k = \mathbb{A}\boldsymbol{v}_{k-1}, \quad m_k = \max(\boldsymbol{u}_k), \quad \boldsymbol{v}_k = \boldsymbol{u}_k/m_k,}$$

其中 $\max(\boldsymbol{u}_k)$ 表示向量 $\boldsymbol{u}_k$ 按模最大的首个分量值。

定理 4.5. 幂法中的向量 $\boldsymbol{v}_k$ 收敛[ii]到主特征向量 $\boldsymbol{x}_1$, 而 m_k 收敛到主特征值 λ_1, 即

$$m_k = \lambda_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right),$$

其中 λ_2 是按模第二大的主特征值。前两个主特征值的比值决定了幂法的收敛速度。

✯ **说明 4.4.** 要在幂法的迭代过程有效收集到主特征信息, 理论上要求初始向量在待解的主特征子空间 $span(\boldsymbol{x}_1)$ 上投影分量非零。若投影分量非常接近零时, m_k 趋于主特征值将呈现出一个非常缓慢的假收敛过程。但是, 随着数值计算的进行, 舍入误差的积累可能会起到积极的作用, 将收敛过程加快, 或将收敛目标纠正到真正的主特征值。

事实上, 我们不用过分担心初始向量的选取问题。因为, 在数值计算中, 我们通常会选取若干个初始向量。或者, 若发现收敛过程非常缓慢的时候, 我们可终止幂法的计算, 然后更换初始向量, 重新进行新的计算。

[ii]其真正含义应是收敛到特征子空间, 即 $\mathrm{span}(\boldsymbol{v}_k) \to \mathrm{span}(\boldsymbol{x}_1)$, 或者等价于这两个空间的距离或者两个向量的夹角趋于零。

✍ **论题 4.11.** 幂法也可以应用到更加一般的矩阵主特征值计算。但是，幂法的数值表现同主特征信息的分布情况具有密切的相关性。

1. 特征向量系的完备很非常重要的要求。此时，主特征值的分离是个重要的条件。等模的主特征值主要有如下三种情形。

   (a) 主特征值是半单的实重根: 即特征值是重根，但特征向量系无亏失 (或几何重数等于代数重数)。此时，幂法依旧有效，但是得到的特征向量同初始向量的选取密切相关。

   (b) 主特征值互为相反数; 此时幂法本身是不收敛的，它含有两个收敛的子列。此时，我们可以考虑连续执行两步幂法，并由此得到相应的主特征信息。

   (c) 主特征值为等模的共轭复数: 幂法虽然可以工作，但其推广过程过于复杂，且计算效率不高。在数值计算过程中，我们要用到最小二乘技术，确定对应共轭复数的实二次方程的两个系数。当特征值的虚部非常小的时候，相应的数值稳定性很差。

2. 若主特征值是重根并造成特征向量系的亏失，幂法将不再按几何方式收敛，而是按极其缓慢的调和方式收敛。

❀ **思考 4.2.** 设 $\mathbb{J}$ 是具有单一特征值 λ 的 n 阶 *Jordan* 阵，$\boldsymbol{v}_0$ 是任意的非零初始向量。请观察 $\mathbb{J}^k\boldsymbol{v}_0$ 当 $k \to \infty$ 时的具体表现。

在本节结束之前，我们要重点指出：幂法的计算公式和收敛情形，强烈依赖于最大模特征值在圆周 $r = \rho(\mathbb{A})$ 上的分布情况。因此，幂法的实际应用具有某种不便性，特别是特征值的分布情形无法事先明确的时候。只有在矩阵的阶数非常高，无法利用其它高效算法的时候，幂法才会成为首选的方法。

4.2.2 加速技巧

✍ **论题 4.12.** 由定理 *4.5* 可知，单位化信息 m_k 按几何方式收敛到主特征值，相应的收敛速度是线性（或者一阶）的。此时，*Aitken* 加速技巧是一种非常有效的加速方法，即

$$\tilde{m}_k = m_k - \frac{(\Delta m_k)^2}{\Delta^2 m_k},$$

将比 m_k 更快地趋近主特征值，其中 $\Delta m_k = m_{k+1} - m_k$ 为向前差分。故而，该方法也称为 Δ^2 方法。

✍ **论题 4.13.** 原点平移方法是一种最简单易行的加速方法，即我们转而考虑 $\mathbb{A} - \mu\mathbb{I}$ 的主特征值求解，其中 μ 是平移量。理论上，我们希望平移之后的前两个主特征值的比值非常小，而 $\lambda_1 - \mu$ 依旧是主特征值。但是，在幂法中确定高效的平移量是个很难的问题。平移技巧更多地用于其他方法，例如反幂法。

✍ **论题 4.14.** 若 $\mathbb{A}$ 是对称矩阵，我们通常采用更有效的 *Rayleigh* 商加速法。换言之，m_k 被替换为 $\boldsymbol{v}_k$ 的 *Rayleigh* 商，即

$$R(\boldsymbol{v}_k) = \frac{\boldsymbol{v}_k^\top \mathbb{A}\boldsymbol{v}_k}{\boldsymbol{v}_k^\top \boldsymbol{v}_k} = \lambda_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right).$$

相应的向量单位化可直接采用欧几里得范数进行，m_k 无需计算。同定理 *4.5* 相比，线性收敛的速度提高至原有速度的平方倍。

✯ **说明 4.5.** 事实上，*Rayleigh* 商 $R(\boldsymbol{x}) = \boldsymbol{x}^\top \mathbb{A}\boldsymbol{x}/\boldsymbol{x}^\top \boldsymbol{x}$ 是一个非常重要的量，它是关于 μ 的矛盾方程组 $\mathbb{A}\boldsymbol{x} - \mu\boldsymbol{x} = 0$ 的最小二乘解。它表示同 $\boldsymbol{x}$ 夹角最小的特征子空间所对应的特征值。

设 $\mathbb{A}$ 是任意的矩阵，不要求它是对称的。当 $\boldsymbol{x}$ 变化时 $R(\boldsymbol{x})$ 的集合称为 $\mathbb{A}$ 的值域，记为 $V(\mathbb{A})$。它可以看作复平面上的一个点集，具有如下性质：

1. $V(\mathbb{A})$ 包含 $\mathbb{A}$ 的所有特征值；
2. $V(\mathbb{A})$ 是酉不变的；
3. $V(\mathbb{A})$ 是一个有界闭凸集。若 $\mathbb{A}$ 是规范矩阵，则 $V(\mathbb{A})$ 是复平面上以所有特征值为顶点的单纯形。特别地，当 $\mathbb{A}$ 是 *Hermite* 矩阵时，$V(\mathbb{A})$ 是以最大特征值和最小特征值为端点的闭区间。

更多的讨论可参阅相关文献。

4.2.3 反幂法

反幂法可以计算可逆矩阵按模最小的特征值信息。它的基本思想非常简单，就是对逆矩阵 $\mathbb{A}^{-1}$ 施行相应的正幂法，即

$$\mathbb{A}\boldsymbol{v}_k = \boldsymbol{u}_{k-1}, \quad m_k = \max(\boldsymbol{v}_k), \quad \boldsymbol{u}_k = \boldsymbol{v}_k/m_k.$$

主要的计算量来自一个同型线性方程组的数值求解。显然，m_k^{-1} 趋于模最小的特征值，而 $\boldsymbol{u}_k$ 趋于相应的特征向量。

✯ **说明 4.6.** 半次迭代：同型线性方程组的数值求解可基于系数矩阵 $\mathbb{A}$ 的 *LU* 分解。通常，我们可以先给出 $\mathbb{A}$ 的相应三角分解，然后在每步迭代中求解两个三角形线性方程组。甚至，因初始向量可以任意的选取，第一个半步计算可以略去，相应的算法称为半次迭代方法。

事实上，反幂法更多地应用于特征信息的精度提升。设 q 是某个特征值的近似，利用原点平移方法，执行如下的反幂法

$$(\mathbb{A}-q\mathbb{I})\boldsymbol{v}_k=\boldsymbol{u}_{k-1},\quad m_k=\max(\boldsymbol{v}_k),\quad \boldsymbol{u}_k=\boldsymbol{v}_k/m_k.$$

我们可以利用 $q+m_k^{-1}$ 给出最接近 q 的特征值，利用 $\boldsymbol{u}_k$ 给出相应的特征向量。当 q 非常接近矩阵 $\mathbb{A}$ 的某个特征值 λ 时，上述反幂法的收敛速度是非常快的。

✯ **说明 4.7.** 由于 q 非常接近矩阵 $\mathbb{A}$ 的某个特征值 λ，相应的矩阵 $\mathbb{A}-q\mathbb{I}$ 非常接近奇异，相应的线性方程组是一个非常病态的问题。此时，我们通常只需迭代一次就可得到足够好的近似特征向量。第二次迭代不会明显地改善计算效果，甚至产生破坏作用。

这个现象可以给出相应的理论分析。事实上，在第一步迭代中，舍入误差的坏影响却起到了正面的效果。简单地说，线性方程组的求解误差造成其数值解在特征子空间上的投影长度改变。误差越大，在特征子空间上的投影越大。这对于近似特征向量的计算而言，是非常有利的，因为此时的计算核心是特征方向，而不是特征向量的长度。

✯ **说明 4.8.** 反幂法的实现是以列主元高斯消元（或 LU 分解）方法为基础。若算法执行失败，则 $\mathbb{A}-q\mathbb{I}$ 的绝对值最小的特征值可近似为零。若需要求解相应的特征向量，我们可对平移量 q 进行一个非常小的扰动，以使带有新位移量的反幂法顺利地进行。

若在反幂法的执行过程中，利用 Rayleigh 商作为平移量，进行算法加速，我们可以得到著名的 Rayleigh 商算法：

$$(\mathbb{A}-\mu_{k-1}\mathbb{I})\boldsymbol{v}_k=\boldsymbol{q}_{k-1},\quad \boldsymbol{q}_k=\boldsymbol{v}_k/\|\boldsymbol{v}_k\|_2,\quad \mu_k=\boldsymbol{q}_k^H\mathbb{A}\boldsymbol{q}_k,$$

其中 $\mu_0=\boldsymbol{q}_0^H\mathbb{A}\boldsymbol{q}_0$，而 $\boldsymbol{q}_0$ 是任给的单位向量。可以证明：μ_k 和 $\boldsymbol{q}_k$ 均非常快速地趋于矩阵的某个特征信息。

✯ **说明 4.9.** 在理论分析中，我们通常利用

$$\rho_k = \|(\mathbb{A} - \mu_k \mathbb{I})\boldsymbol{q}_k\|_2$$

刻画 *Rayleigh* 商 μ_k 靠近某个特征值的程度。当 *Rayleigh* 商迭代收敛于单特征值和对应的特征向量时，可以证明它具有平方收敛速度，即 ρ_{k+1} 可被 ρ_k^2 的某个倍数所控制。当矩阵 $\mathbb{A}$ 是对称的，我们可证明其具有三次方收敛速度。具体的分析过程略。

4.2.4 其他特征值的求解

下面，我们探讨其他特征值（例如第二个主特征信息）的计算问题。我们将主要介绍三种方法。

降维法

降维方法是数值计算的一种常用方法。就此处而言，就是应用幂法求解某个低阶矩阵的主特征值。其核心内容称之为矩阵的收缩技术。

✎ **论题 4.15.** 矩阵收缩的关键步骤是找到一个可逆矩阵 $\mathbb{S}$，将已得的主特征向量 $\boldsymbol{x}_1$ 转化为仅首个位置非零的向量，即

$$\mathbb{S}\boldsymbol{x}_1 = t\boldsymbol{e}_1.$$

我们可采用 *Gauss* 消元阵、*Householder* 镜像变换阵或者 *Givens* 平面旋转阵，实现这个目标。

要求解 $\mathbb{A}$ 的第二个主特征信息，我们只需对 $\mathbb{S}^{-1}\mathbb{A}\mathbb{S}$ 的右下角 $n-1$ 阶矩阵执行幂法，得到相应的主特征信息即可。请注意两个矩阵特征信息之间的联系。降维方法的主要缺陷是矩阵稀疏结构遭到破坏。

❀ **思考 4.3.** 请思考 $\mathbb{S}^{-1}\mathbb{A}\mathbb{S}$ 的计算方法。

Wieldant 收缩法

要求解 $\mathbb{A}$ 的第二主特征信息，我们也可以考虑秩一修正矩阵

$$\mathbb{A}_1 = \mathbb{A} - \sigma \boldsymbol{x}_1 \boldsymbol{v}^H,$$

其中 σ 和 $\boldsymbol{v}$ 是待定的信息。通常 $\sigma = \lambda_1$ 和 $\boldsymbol{v} = \boldsymbol{x}_1$ 为已解出的主特征信息。

这种方法称为 Wieldant 收缩法。它的优势是 $\mathbb{A}_1$ 可以不必存储和真正计算出来。换言之，我们只需额外存储两个向量 σ 和 $\boldsymbol{v}$ 即可。此外，它特别适用于对称矩阵[iii]，因为第一主特征值可以彻底转化为零。

子空间同时迭代法

在上述两种计算方法中，第二个主特征信息的计算精度都要受到第一个主特征信息的计算精度影响。因为舍入误差的积累，特征信息的逐次求解过程，势必造成计算精度会越来越差。因此，我们希望建立一个高效的算法，同时求出矩阵的前面若干个主特征信息。

子空间同时迭代法就可以实现这个目标。下面，我们以对称矩阵为例，给出它内蕴的两个基本算法。

✎ **论题 4.16.** 基本的子空间同时迭代方法：任取 m 个列直交向量，组成初始的列直交矩阵 $\mathbb{V}_0$；然后，循环执行

$$\boxed{\mathbb{U}_k = \mathbb{A}\mathbb{V}_{k-1}, \quad \mathbb{U}_k = \mathbb{V}_k \mathbb{R}_k.}$$

第一步是基本的正幂法过程，而第二步是对（斜）像空间重构新的正交基底，以避免特征空间的数值坍塌。

[iii] 当然，这种方法也可用于非对称矩阵，数值效果也不错。

若 $\mathbb{A}$ 是实对称矩阵，我们可以在上述算法中引入广义的 Rayleigh 商加速技术，即所谓的子空间投影算法。

✍ **论题 4.17.** 子空间投影算法是一种常用的数值技术，即我们如何在一个低维的子空间中，求解出原有高维问题的某个近似解。

假设所谓的低维子空间是由 $\mathbb{V}_{k-1}$ 的 m 个列直交向量张成的。设想要求解的特征信息被局限在这个子空间（或最小二乘思想）中，我们可以导出一个小规模的 m 阶特征值问题

$$\mathbb{V}_{k-1}^{\top}\mathbb{A}\mathbb{V}_{k-1}\boldsymbol{y}_{k-1} = \tilde{\lambda}_{k-1}\boldsymbol{y}_{k-1}.$$

显而易见，这个小规模特征问题要比原有的 n 阶问题容易很多。在某种程度上，我们可以认为 $\tilde{\lambda}_{k-1}$ 是矩阵 $\mathbb{A}$ 的一个近似特征值，对应的近似特征向量是 $\mathbb{V}_{k-1}\boldsymbol{y}_{k-1}$。

教科书给出的子空间同时迭代法就是上述两个算法的合并版本。它具有如下的收敛性分析结果。

定理 4.6. 设实对称矩阵 $\mathbb{A}$ 的特征值彼此互异，则子空间同时迭代法关于特征值和特征向量均具有很好的收敛性质。

证明：该算法的收敛性证明是典型的。见教科书。 □

✯ **说明 4.10.** 初始的正交列向量组可如下选取：其一是由一个随机向量构造 *Householder* 矩阵，然后再任取一些列向量；其二是随机构造一组列向量，然后进行 *Gram-Schmildt* 正交化。大量的数值经验表明，后者的数值表现要更好一些。

✯ **说明 4.11.** 子空间同时迭代算法中低阶方阵的特征值和特征向量，可以采用后面介绍的 *Jacobi* 方法给出结果。当迭代步数 k 适当大的时候，对称的低阶矩阵将非常接近于一个对角阵，相应的 *Jacobi* 方法可以特别有效和快捷地求解特征值。

4.3 实对称矩阵的 Jacobi 方法

熟知的代数结论：对于实对称矩阵 $\mathbb{A}$，我们可以利用一个正交矩阵，将其相似变换到一个对角阵，从而得到所有的特征值。但是，我们很难通过简单机械的方式，获得这个正交阵。退而求其次，我们希望找到一系列简单易行的直交阵序列，实现这个正交矩阵的作用。

注意到平面旋转阵的乘积也是直交的，Jacobi (1846) 提出了著名的 Jacobi 方法。这个古老的算法具有编程简单、并行效率高等优点。但请注意：Jacobi 方法没有实现最初的目标，通常无法在有限步完成特征值的计算。

4.3.1 基本思想

对于二阶实对称矩阵，Jacobi 方法的基本思想得到完美的体现。换言之，我们可以构造一个 Givens 平面旋转阵 $\mathbb{G}(p,q) \equiv \mathbb{G}(p,q;\theta)$，执行相似变换

$$\begin{bmatrix} b_{pp} & b_{pq} \\ b_{pq} & b_{qq} \end{bmatrix} = \mathbb{G}(p,q) \begin{bmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{bmatrix} \mathbb{G}(p,q)^{\top}$$

将矩阵的非对角线元素变换为零，即 $b_{pq} = 0$。这里的 p 和 q 是二阶子矩阵的两个行号，θ 为待定的旋转角度。这个操作过程称为一个 Jacobi 旋转变换，相应的非对角元素 a_{pq} 称为旋转主元。

旋转角度 θ 可按如下公式确定：

$$\cot 2\theta = \xi := \frac{a_{pp} - a_{qq}}{2a_{pq}}, \quad \theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]. \tag{4.3.6}$$

限定 θ 的取值范围，是为了保证 $t = \tan\theta$ 的绝对值不超过 1.

为确定平面旋转矩阵 $\mathbb{G}(p,q;\theta)$ 中的两个具体元素，我们通常使用如下的计算公式

$$\begin{aligned} t &= \tan\theta = \operatorname{sgn}(\xi)\left[|\xi| + \sqrt{1+\xi^2}\right]^{-1}, \\ c &= \cos\theta = \frac{1}{\sqrt{1+t^2}}, \quad s = \sin\theta = ct, \end{aligned} \tag{4.3.7}$$

其中 ξ 的值由 (4.3.6) 给出。

✎ **论题 4.18.** 若 $|\xi|$ 非常大的时候，我们需要小心处理舍入误差的影响。此时，旋转角度 θ 接近于零，上述计算公式将导致有效位数的大量损失，从而计算结果几乎总是 $c=1$ 和 $s=0$，*Jacobi* 迭代没有任何的效果。为此，我们需要将 (4.3.7) 中的第一个公式修正为

$$t = 1/(2\xi),$$

或者采用新的计算公式

$$\begin{aligned} T &= \tan\frac{\phi}{2} = \frac{a_{pq}}{2(a_{pp}-a_{qq})}, \\ \cos\phi &= \frac{1-T^2}{1+T^2}, \quad \sin\phi = \frac{2T}{1+T^2}. \end{aligned} \tag{4.3.8}$$

最后两个三角函数值关于 c 和 s 具有很好的逼近结果，因为：当 $\theta\to 0$ 时，有

$$\phi - \theta \approx \frac{5}{4}\theta^3.$$

证明留作练习。

✯ **说明 4.12.** 当 $a_{pq}=0$ 时，我们不需要执行 *Jacobi* 旋转。在实际计算中，若

$$|a_{pq}| < \mathcal{E}\sqrt{a_{pp}a_{qq}},$$

我们就在数值上认定这个非对角元素 a_{pq} 为零，其中 $\mathcal{E}$ 是事先给定的一个关值，如 $\mathcal{E}=10^{-14}$。

4.3.2 古典 Jacobi 方法

将 Jacobi 思想应用到高阶矩阵时, 数据的零化过程表现将变得略有不同, 对角化过程通常无法在有限步数内结束。主要原因是：旋转矩阵的相似变换会影响到整个 (p,q) 井字线上的数据, 已经被零化的非对角元素可能会重新变成非零。

论题 4.19. *古典 Jacobi 方法*: 令 $\mathbb{A}_0=\mathbb{A}$。*在第 k 步中*, 设

$$a_{pq}^{(k)}=\arg\max_{i\neq j}|a_{ij}|.$$

将其作为旋转主元, *执行一个 Jacobi 旋转变换过程*

$$\mathbb{A}_{k+1}=\mathbb{G}_k\mathbb{A}_k\mathbb{G}_k^{\top},$$

其中 $\mathbb{G}_k=\mathbb{G}_k(p,q;\theta)$ 是在 (p,q) 平面的 Jacobi 旋转阵。

定理 4.7. *古典 Jacobi 方法的矩阵序列 $\mathbb{A}_k$, 本质收敛到某个对角矩阵。所谓的本质收敛是指在集合意义下的收敛。*

证明：观察 Jacobi 旋转变换过程前后的非对角元素平方和。可证相应的 Frobenius 范数趋于零。 □

说明 4.13. *在 Jacobi 方法中限定 $|t|\leq 1$ 是有意义的, 它可以保证算法的真正收敛, 即对角线元素目标一致地趋向某个特征值, 不会发生收敛位置的跳转。换言之, Jacobi 方法的矩阵序列真正收敛到某个固定的对角阵*[iv]。

说明 4.14. *若 $\mathbb{A}$ 的特征值互异, 可以证明 Jacobi 方法给出的特征向量也是收敛的。若 $\mathbb{A}$ 有重特征值, 虽然我们不再保证正交矩阵的列向量关于特征向量的收敛性, 但可确保多维特征子空间的收敛性。*

[iv]这要用到如下引理：设 $\{\boldsymbol{u}_k\}_{k=0}^{\infty}$ 是有限维赋范空间的有界序列。若它最多有有限个聚点且 $\lim_{k\to\infty}\|\boldsymbol{u}_{k+1}-\boldsymbol{u}_k\|=0$, 则 $\boldsymbol{u}_k$ 收敛到某个聚点。

✯ **说明 4.15.** *Demmel* 和 *Veselić* 指出：在 *Jacobi* 方法中，特征值计算的相对误差可以被 $\vartheta\kappa_2(\mathbb{D}^{-1/2}\mathbb{A}\mathbb{D}^{-1/2})$ 所控制，其中 $\mathbb{D} = diag(\mathbb{A})$，$\vartheta$ 为机器精度，$\kappa_2(\cdot)$ 是谱条件数。换言之，*Jacobi* 方法是一个数值稳定的算法。

✎ **论题 4.20.** *Givens* 平面旋转相似变换，仅仅影响位于 (p,q) 并字线上的元素。利用旋转角度的计算公式，位于对角线交叉点上的元素变化可进一步的简化，即

$$a_{pp} := a_{pp} + ta_{pq}, \quad a_{qq} := a_{qq} - ta_{pq}.$$

因为一个 *Jacobi* 旋转可分解为 *Givens* 平面旋转阵的一次左乘运算和一次右乘运算，所以我们仅需两次乘除法运算，就可以得到那些位于并字线上的非交叉点上的所有矩阵元素。考虑到矩阵的对称性，每次迭代所需的乘除法次数仅为 $\mathcal{O}(4n)$。

4.3.3 循环 Jacobi 方法

✎ **论题 4.21.** 在循环 *Jacobi* 方法中，我们放弃旋转主元的全局搜索，而是直接对矩阵的所有非对角线元素，依次进行相应的 *Jacobi* 旋转。通常，这样的计算过程（即共计 $n(n-1)/2$ 次旋转）称为一次 *Jacobi* "扫描"过程。

✯ **说明 4.16.** 在 *Schonhage (1964)* 和 *Van Kempen (1966)* 的工作中，他们指出：若循环 *Jacobi* 方法收敛，则该方法将具有渐进平方收敛速度。此外，在 *Brent* 和 *Luk (1985)* 的工作中，他们指出算法的扫描总次数是同 $\log n$ 成正比例的，其中 n 为矩阵的阶数。

✎ **论题 4.22.** 在循环 *Jacobi* 方法中，若非对角线上的元素 a_{pq} 已

经非常接近零时，相应的 *Jacobi* 旋转对算法的收敛性没有太多的有益贡献。因此，在实际计算中，我们通常引进**阈值扫描策略**：设 $\sigma \geq n$ 是一个固定的常数，逐步设置阈值 δ_k，直至达到机器精度或用户要求为止，即

$$\boxed{\delta_k = \delta_{k-1}/\sigma, \quad k = 1, 2, \ldots.}$$

其中 δ_0 为矩阵 $\mathbb{A}$ 的所用非对角元素平方和的开根号。当 $|a_{pq}|$ 小于设定的阈值时，我们跳过相应的 *Jacobi* 旋转操作；直至所有的非对角线元素均小于给定的阈值时，我们按上述规则，重新设置新的阈值。

❀ **思考 4.4.** 证明：阈值扫描策略下的 *Jacobi* 方法给出的矩阵序列也是收敛的。

✯ **说明 4.17.** 虽然 *Jacobi* 方法的收敛速度不如后面介绍的 *QR* 方法，*Jacobi* 方法依旧被广泛使用，因为它还具有所谓的并行化优势。我们可以将行列指标集 $\{(p,q)\}$ 进行轮换分组，分别在一个 *CPU* 上，执行 *Givens* 平面旋转矩阵左乘和右乘运算，便可实现循环 *Jacobi* 方法的并行计算。

4.3.4 特征向量的计算

✎ **论题 4.23.** 在上述 *Jacobi* 方法中，我们可利用存储的旋转矩阵信息 $\{(p,q;c,s)\}$，快速恢复相应的特征向量。每个 *Jacobi* 旋转矩阵的信息存储，仅仅需要两个整数和一个或两个浮点数，记录旋转位置 (p,q) 和相应的角度信息。

更好的计算方法是利用带有偏移量的反幂法，确定某个特征值所对应的特征向量。

4.4 实对称矩阵的 Givens-Householder 方法

实对称矩阵的特征值计算还有另外一种处理方式。它是两种数值策略（有限步三对角化过程和二分法求根法）的有机结合。

4.4.1 三对角化策略

在有限步数内难以达到对角阵，那么能够达到的最简单矩阵是什么呢？它就是所谓的三对角矩阵。相应的三对角化过程主要有两种实现方法，虽然它们的基本思想是一样的。

✎ **论题 4.24.** 我们可采用 *Givens* 平面旋转阵，实现对称矩阵的三对角化。主要思路是利用每个**副对角线元素**及其下方的元素，依次构造相应的平面旋转阵，将副对角线下方的所有元素清零。相应的旋转信息可存储在原有的位置。详细处理可参见说明 *3.9*。

这个数值算法的实现目标同 *Jacobi* 方法截然不同，所采用的计算公式也是完全不同的。

✎ **论题 4.25.** 当然，我们也可采用 *Householder* 镜像变换阵，实现对称矩阵的三对角化。假设左上角矩阵 $\mathbb{A}_k$ 已经被成功地三对角化，我们可执行如下操作：

$$\begin{bmatrix} \mathbb{I}_k & 0 \\ 0 & \mathbb{H}_{n-k} \end{bmatrix} \left[\begin{array}{c|c} \mathbb{A}_k & \begin{matrix} 0 \\ \boldsymbol{a}_k^\top \end{matrix} \\ \hline 0 \quad \boldsymbol{a}_k & \mathbb{A}_{n-k} \end{array}\right] = \left[\begin{array}{c|c} \mathbb{A}_k & \begin{matrix} 0 \\ (\mathbb{H}_{n-k}\boldsymbol{a}_k)^\top \end{matrix} \\ \hline 0 \quad \mathbb{H}_{n-k}\boldsymbol{a}_k & \mathbb{H}_{n-k}\mathbb{A}_{n-k} \end{array}\right],$$

其中 $\mathbb{H}_{n-k}$ 是 *Householder* 镜像变换阵，可使**对角线下方**的 $n-k$ 维列

向量

$$\boldsymbol{a}_k = (a_{k,k+1}, a_{k,k+2}, \ldots, a_{kn})^\top$$

转化为仅首个位置非零的向量。在数值计算过程中，我们仍然可以采用相应的数据覆盖技术。*Householder* 镜像变换矩阵的信息可保存在相应的位置。

✯ **说明 4.18.** 上述两种三对角化过程的优劣之处，主要在于它们的计算复杂度。若矩阵是稠密的，*Hoseholder* 镜像变幻方法比 *Givens* 平面旋转方法更有优势，乘除法的总次数可由 $\mathcal{O}(4n^3/3)$ 下降到 $\mathcal{O}(2n^3/3)$，开根号的总次数可由 $n^2/2$ 下降到 $n-2$。但是，当矩阵是稀疏的时候，*Givens* 平面旋转方法的优势才会显现出来。

❀ **思考 4.5.** 我们能否使用 *Gauss* 消元阵，在有限步完成任意对称矩阵的三对角化呢？请给出具体的实现过程，并评价算法的优缺点。

4.4.2 二分法

对于实对称三对角矩阵

$$\mathbb{T}_n = \text{symtridiag}(\{\alpha_i\}_{i=1}^n, \{\beta_i\}_{i=2}^n)$$

我们可以采用二分法计算特征值，其中 α_i 位于对角线，而 β_i 位于非对角线。

不妨设 $\mathbb{T}_n$ 是不可约的，即非对角线元素都是非零的。若某个非对角元素为零，则这个特征值问题可分割为两个小规模的特征值问题。

特征值的计算

对应三对角对称矩阵 $\mathbb{T}_n$，定义多项式序列

$$p_i(\lambda) = \det[(\mathbb{T}_n - \lambda\mathbb{I}_n)(1:i, 1:i)], \quad i = 1, 2, \ldots, n.$$

显然, $p_n(\lambda)$ 就是矩阵 $\mathbb{T}_n$ 的特征多项式。利用行列式的性质, 不难知道这些多项式的次数递增, 且满足如下的递推关系式 :

$$p_i(\lambda) = (\alpha_i - \lambda)p_{i-1}(\lambda) - \beta_i^2 p_{i-2}(\lambda), \quad i = 2, \ldots, n,$$

其中 $p_0(\lambda) = 1$。可以证明, 上述多项式还满足如下性质 :

1. $\operatorname{sgn} p_i(-\infty) = 1, \operatorname{sgn} p_i(+\infty) = (-1)^i$;
2. 相邻的两个多项式没有公共根 ;
3. 若 $p_i(\mu) = 0$, 则 $p_{i-1}(\mu)p_{i+1}(\mu) < 0$;
4. $p_i(\lambda)$ 的根全是单根, 并且 $p_i(\lambda)$ 的根可以严格地分隔开 $p_{i+1}(\lambda)$ 的根。

相应的证明是简单的 ; 可参阅教科书。

定义 4.4. 给定 μ, 称 $\{p_i(\mu)\}_{i=0}^k$ 构成一个 *Sturm* 序列。记 *Sturm* 序列中相邻两个数符号相同的总数目为 $s_k(\mu)$, 称为符号相同数。

补充说明: 若 $p_i(\mu) = 0$, 称 $p_i(\mu)$ 和 $p_{i-1}(\mu)$ 的符号相反, 而称 $p_{i+1}(\mu)$ 和 $p_i(\mu)$ 的符号相同。

符号相同数是一个可计算量, 它具有如下的重要性质。

定理 4.8. 对任意给定实数 μ, $p_r(\lambda)$ 恰有 $s_r(\mu)$ 个根严格大于 μ.

证明 : 数学归纳法。 □

论题 4.26. 这个定理具有重要的应用价值。换言之, 在 $(a, b]$ 区间内, 三对角对称矩阵 $\mathbb{T}_n$ 的特征值共有 $s_n(a) - s_n(b)$ 个。

利用这个性质 (或者定理 4.8), 我们可以快速地确定对称三对角矩阵 $\mathbb{T}_n$ 的第 k (按大小排序) 个特征值。基本计算过程如下 :

1. 特征值的隔离：

   (a) 界定特征值范围 $a \le \lambda \le b$.

   (b) 取中点位置 $c = (a+b)/2$，计算符号一致数 $s_n(c)$。折半缩减区间 $[a,b]$ 到 $[a,c]$ 或 $[c,b]$，使其左端点的同号数为 k，而右端点的同号数为 $k+1$。

2. 特征值的确定：继续区间的等分操作，直到区间长度达到用户的要求。在每次区间二分之后，我们放弃符号相同数相等的那个子区间。

显然，上述二分法操作过程在理论上是无条件收敛的。

✯ **说明 4.19.** 尽管舍入误差会限制二分法的计算精度，但是，利用标准的浮点运算误差分析，我们可以证明二分法是数值稳定的。

✯ **说明 4.20.** 对于高阶矩阵，*Sturm* 序列 $\{p_i(\mu)\}_{i=0}^n$ 的计算，可能存在严重的舍入误差积累和上下数值溢出的风险。为此，我们经常采用如下算法，来统计符号相同数：令 $q_1(\mu) = p_1(\mu)$, 递归计算

$$q_i(\mu) = \alpha_i - \mu - \frac{\beta_i^2}{q_{i-1}(\mu)}, i = 2, 3, \ldots.$$

补充说明：最后一项的比值应该按照极限的概念来理解。特别地，若 $q_{i-1}(\mu) = 0$，则直接定义 $q_i(\mu) = -\infty$，它对符号相同数目没有任何贡献。因此，符号相同数 $s_k(\mu)$ 正好是序列 $q_1(\mu), q_2(\mu), \ldots, q_k(\mu)$ 中非负数的总数。

特征向量的计算

确定三对角对称矩阵 $\mathbb{T}_n$ 的特征值之后，我们可以利用直接法（例如令 $x_1 = 1$），快速求解相应的特征向量。但是，这种方法的数值稳定性比较差。我们可采用带原点平移的反幂法，改善特征值的近似程度，并同时求出相应的特征向量。

最后，利用相似约化过程中的信息，我们可以由三对角矩阵的特征值，恢复重构原始矩阵的特征向量。

4.5 QR 方法

QR 方法在 1961-1962 年所提出，其前身是基于矩阵三角分解的 LR 方法 (1958)。它同矩阵的 Schur 分解和幂法有着密切的关系，是目前中小规模稠密矩阵全部特征信息的最有效计算方法之一。

4.5.1 基本思想

✎ **论题 4.27.** 记 $\mathbb{A}_0 = \mathbb{A}$ 为原始矩阵。我们依次进行矩阵的直交分解和交换相乘，可得 *QR* 算法:

$$\boxed{\mathbb{A}_k = \mathbb{Q}_k\mathbb{R}_k, \quad \mathbb{A}_{k+1} = \mathbb{R}_k\mathbb{Q}_k,}$$

其中 $\mathbb{Q}_k$ 是正交阵，$\mathbb{R}_k$ 是上三角阵。显然，迭代序列 $\{\mathbb{A}_k\}_{k=0}^{\infty}$ 中的任意两个矩阵是正交相似的。

QR 方法的收敛性比较复杂。

定理 4.9. 设矩阵 $\mathbb{A}$ 的所有特征值均为实数，且按绝对值是严格分离的。若以 $\mathbb{A}$ 的左特征向量为行向量组成的矩阵 $\mathbb{X}$ 具有 *LU* 分解，则

QR 方法给出的迭代序列 $\mathbb{A}_k$ 本质上收敛到一个上三角阵，相应的对角线元素趋于矩阵 $\mathbb{A}$ 的特征值。

这个定理的证明, 非常类似于子空间同时迭代法的讨论, 此处略。下面，我们给出如下的简要说明。

✍ **论题 4.28.** 事实上，*QR* 方法同幂法有密切的联系，因为

$$\mathbb{A}^k = \mathbb{Q}_1\mathbb{Q}_2\cdots\mathbb{Q}_k\mathbb{R}_k\cdots\mathbb{R}_2\mathbb{R}_1 = \widetilde{\mathbb{Q}}_k\widetilde{\mathbb{R}}_k.$$

我们可以利用正（或反）幂法的收敛性，阐述 *QR* 方法的最左侧（或者最右侧）第一个列向量的收敛情形。通常，迭代矩阵最右下角的元素具有最快的收敛速度。

✯ **说明 4.21.** 事实上，*QR* 方法同 *Rayleight* 商迭代有更为密切的联系。通常，*QR* 方法具有平方收敛速度。若矩阵是实对称的，则它可达到三次方收敛速度。

4.5.2 数值实现

若直接对原始矩阵执行 QR 方法，相应的计算复杂度将会非常高，这使得 QR 方法在特征值问题的计算中缺乏足够的竞争力。为此，我们需要在 QR 方法中引入更多的数值加速技巧。

上 Hessenberg 化过程

✍ **论题 4.29.** 首先，为降低 *QR* 方法的计算复杂度，我们需要将矩阵 $\mathbb{A}$ 相似变换到一个上 *Hessenberg* 矩阵。这部分的操作内容同对称矩阵的三对角化过程几乎是一模一样的。

譬如，我们可以采用 *Householder* 镜像变换阵，或者 *Givens* 平面旋转阵，来完成上述目标。若矩阵是非对称的，上三角部分也需要花费额外的计算来完成。整个过程共需要 $\mathcal{O}(5n^3/3)$ 次乘除法运算。

错位相乘方法

✍ **论题 4.30.** 对于上 *Hessenberg* 矩阵，相应的 QR 分解可采用一系列的 *Givens* 平面旋转阵来实现。换言之，我们可依次用对角线元素，将其下方的副对角线元素旋转为零。因此，一次完整的 QR 迭代过程需要次 $\mathcal{O}(2n^2)$ 乘除法运算。

但是，中间计算数据的存储是需要讨论的问题，即关于

$$\mathbb{A}_{k+1} = \mathbb{G}_{k-1}^{\top} \cdots \mathbb{G}_1^{\top} \mathbb{A}_k \mathbb{G}_1 \cdots \mathbb{G}_{k-1}$$

的操作过程要细致分析。若从中心的 $\mathbb{A}_k$ 出发，利用旋转阵左右相乘后形成的相似变换阵，例如 $\mathbb{G}_1^{\top}\mathbb{A}_k\mathbb{G}_1$，不再是一个上 *Hessenberg* 矩阵。为保持计算过程中，迭代矩阵一直具有所谓的上 *Hessenberg* 结构，我们可采用错位相乘方式。例如，我们先计算 $\mathbb{G}_1^{\top}\mathbb{A}_k$，将 $(2,1)$ 的元素清零，然后在两次分别左乘和右乘不同的 *Givens* 平面旋转阵。然后，我们再分别左乘和右乘不同的 *Givens* 平面旋转阵。

原点平移技术

✍ **论题 4.31.** 为提高 QR 方法的收敛速度，我们可采用原点平移技术。例如单步位移的 QR 方法

$$\boxed{\mathbb{A}_k - t_k\mathbb{I} = \mathbb{Q}_k\mathbb{R}_k, \quad \mathbb{A}_{k+1} = \mathbb{R}_k\mathbb{Q}_k + t_k\mathbb{I},}$$

其中 t_k 为位移量。此时，成功的平移量有如下两种策略:

1. 第一种是直接以 $\mathbb{A}_k$ 右下角的元素为位移量，即 $t_k = a_{nn}^{(k)}$.

2. 第二种是 *Wilkinson* 位移量，它是右下角二阶矩阵

$$\begin{bmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{n,n}^{(k)} \end{bmatrix} \tag{4.5.9}$$

靠近 $a_{n,n}^{(k)}$ 的一个特征值。它特别适合于对称三对角矩阵，此时的位移量为

$$\boxed{t_k = a_{n,n}^{(k)} + \delta - sgn(\delta)\sqrt{\delta^2 + \beta_{n-1}^2},}$$

其中 $\delta = (a_{n-1,n-1}^{(k)} - a_{n,n}^{(k)})/2$，$\beta_{n-1} = a_{n-1,n}^{(k)} = a_{n,n-1}^{(k)}$。

✯ **说明 4.22.** 位移策略不保证迭代一定收敛。譬如，二阶置换阵

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

具有等模的两个特征值，带第一种位移量的 QR 迭代序列会出现所谓的循环，矩阵序列是不收敛的。

降阶处理

当 $a_{n,n-1}^{(k)} \approx 0$ 时，我们有理由取 $a_{nn}^{(k)}$ 作为 $\mathbb{A}$ 的一个近似特征值。当 $a_{n-1,n-2}^{(k)} \approx 0$ 时，我们有理由取 (4.5.9) 的两个特征值作为 $\mathbb{A}$ 的近似特征值。此时，我们可将右下角矩阵剔除，换言之，待解的矩阵阶数缩短。

✎ **论题 4.32.** 副对角线元素为零的数值判断准则：

$$|a_{n,n-1}^{(k)}| \leq \mathcal{E} \min(|a_{n,n}^{(k)}|, |a_{n-1,n-1}^{(k)}|),$$

其中 $\mathcal{E}$ 为根据精度要求预先给定的一个小的正数。

✯ **说明 4.23.** 为求解多项式 $p_n(x) = a_0 + a_1x + a_2x^2 + \cdots a_nx^n$ 的全部根，我们可考虑其对应友矩阵

$$\begin{bmatrix} 0 & & & & -a_0 \\ 1 & 0 & & & -a_1 \\ & 1 & \ddots & & \vdots \\ & & \ddots & 0 & -a_{n-2} \\ & & & 1 & -a_{n-1} \end{bmatrix}$$

的特征值问题。在 *Matlab* 中，命令 *roots()* 就是对这个友矩阵调用著名的 *QR* 方法，所需的计算量约为 $\mathcal{O}(n^3)$。最近的研究表明，若充分利用友矩阵的特点，这个计算量可以下降到 $\mathcal{O}(n^2)$；详略。

4.5.3 隐式对称 QR 方法

前面的讨论说明：从上 Hessneberg 矩阵相似变换到下一个上 Hessneberg 矩阵的过程，是 QR 方法的核心问题。除错位相乘技术之外，我们还可用隐含的方式来实现，即所谓的"驱逐出境"策略。相应的理论基础是任意矩阵上 Hessenberg 化的唯一性：

定理 4.10. *设存在 $\mathbb{U}$ 和 $\mathbb{V}$ 两个正交阵，使得*

$$\mathbb{U}^\top \mathbb{A}\mathbb{U} = \mathbb{H}, \quad \mathbb{V}^\top \mathbb{A}\mathbb{V} = \mathbb{G},$$

其中 $\mathbb{H}$ 和 $\mathbb{G}$ 都是不可约的上 Hessenberg 阵。若 $\mathbb{U}$ 和 $\mathbb{V}$ 的第一列相等，则整个过程可视为唯一的。换言之，存在一个对角阵 $\mathbb{D} = diag\{\pm 1\}$，使得 $\mathbb{U} = \mathbb{V}\mathbb{D}$ 和 $\mathbb{H} = \mathbb{D}\mathbb{G}\mathbb{D}$.

对于实对称上 Hessneberg（即实对称三对角）矩阵，我们可用单步位移方法解出全部的（实）特征值。我们假设矩阵一直具有不可约[v]结

[v]若矩阵的不可约结构被破坏，则这个矩阵特征值问题可以分解为多个小规模的特征值问题。

构（即副对角线元素非零），我们可以引进 Wilkinson 位移策略和“驱逐出境”策略，建立所谓的隐式对称 QR 算法。相应的伪代码如下：

1. 计算 Wilkinson 位移 μ; 令 $x = \mathbb{T}(1,1) - \mu, y = \mathbb{T}(2,1)$;
2. For $k = 1 : n-1$, Do
3. 计算 $[c,s] = \text{GIVENS}(x,y)$，将列向量 $(x,y)^\top$ 中的 y 旋转消灭; 相应的旋转阵记为 $\mathbb{G}(k,k+1)$;
4. 计算 $\mathbb{T} = \mathbb{G}(k,k+1)\mathbb{T}\mathbb{G}(k,k+1)^\top$;
5. 若 $k < n-1$，令 $x = \mathbb{T}(k+1,k), y = \mathbb{T}(k+2,k)$.
6. Enddo

若采用斜对角线方式存储对称三对角矩阵，图文框中的第 4 步和第 5 步伪代码应做相应的修改。

4.5.4 双重位移的 QR 方法

✍ **论题 4.33.** 若实矩阵的特征值是复数，上述算法不可能给出相应的共轭复特征值，我们需要采用复数的位移量，并在复数域中运算。注意到共轭复特征值含于 *Schur* 阵的二阶实对角矩阵块中，连续两步的不同（共轭复数）位移可以耦合在一起，使得运算依旧可以在实数域中完成。相应的方法称为双重位移的 *QR* 方法。

若直接使用连续两步的 *QR* 方法，我们需要进行矩阵的平方运算。这会导致每步迭代需要 $\mathcal{O}(n^3)$ 次乘除法运算。这可用隐式 *QR* 算法来优化。它非常类似于对称情形的隐式操作，详略。

4.6 奇异值分解

略。

第 5 章
非线性方程（组）的数值方法

非线性方程（组）是更为普遍的数学问题，相应的根计算具有重要的意义。此时的数值方法研究非常富有挑战性，其基本思想同线性方程组的迭代解法非常类似，但是相应的数值表现和理论分析却明显不同。

5.1 基本概念

为求非线性方程（组）$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ 的根，我们通常采用迭代方法。换言之，我们将按给定的 r 阶迭代公式

$$\boldsymbol{x}_k = \boldsymbol{g}(\boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}, \ldots, \boldsymbol{x}_{k-r})$$

给出迭代序列 $\{\boldsymbol{x}_k\}$，其中 $\boldsymbol{g}$ 是迭代函数。同线性方程组的迭代方法类似，我们要重新明确如下概念：

1. 迭代序列是确定的：即迭代公式的所有运算是有效，或者迭代序列位于 $\boldsymbol{g}$ 的定义域。

2. 收敛情况的刻画：非线性问题的迭代方法收敛性很复杂，它包括所谓的全局收敛（或大范围收敛）以及局部收敛。

   (a) 传统的局部分析通常假定了问题真解的信息。

   (b) 若收敛性分析不依赖问题真解的信息，这种分析称为半局部收敛分析。

对于线性问题，迭代方法仅有收敛或不收敛两种状态，因为若其收敛必是全局收敛。

3. 收敛速度的刻画：记 $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}_\star$ 为第 k 步迭代误差，其中 $\boldsymbol{x}_\star$ 是方程的某个真解。设存在非负常数 p 和 C，使得[i]

$$\|\boldsymbol{e}_{k+1}\| \le C\|\boldsymbol{e}_k\|^p, \quad \forall k \ge k_0. \tag{5.1.1}$$

这里的 $\|\cdot\|$ 是向量范数。对单个标量方程而言，$\|\cdot\|$ 就是绝对值；为分析简便，我们常常将 (5.1.1) 简化为极限形式，例如

$$\lim_{k\to\infty} \frac{\|\boldsymbol{e}_{k+1}\|}{\|\boldsymbol{e}_k\|^p} = C. \tag{5.1.2}$$

此时，我们有如下结论：

(a) 若 $C \neq 0$，则称算法是 p 阶收敛的。若 $C = 0$，则称算法是至少 p 阶收敛的。

(b) 当 $p = 1$ 时，界定常数 C 还需要有额外限制，即 $0 \le C < 1$。若 $C \neq 0$，则称算法是线性收敛的；若 $C = 0$，则称算法是超线性收敛的。

4. 算法效率的刻画：算法的实用效率（即数值解收敛到同样精度所需的计算时间）是一个更为重要的问题。在求解非线性方程组时，算法的实用效率是瓶颈之一。

设 W 是每步迭代的计算复杂度指标，例如所需的乘除法总次数，或者对应的 CPU 时间等。通常，计算效率定义为

$$\eta = \frac{\ln p}{W}, \text{ 若 } p > 1; \quad \eta = \frac{\ln C}{W}, \text{ 若 } p = 1.$$

[i] 从实用的角度出发，通常要求 $p \ge 1$。

换言之，为提高计算效率，我们可以提高算法的收敛阶，或者降低每步迭代所需的计算复杂度。

5. 数值稳定性：舍入误差不应严重破坏数值算法理论上的收敛效果。这是一个非常繁琐的分析内容，详略。我们仅仅通过数值实验来体会这方面的概念。

✯ **说明 5.1.** 停机标准的设置于线性方程组的迭代方法中的设置是类似的。最主要的度量方式是残量和相邻解差距，即

$$\|\boldsymbol{f}(\boldsymbol{x}_k)\| \leq \mathcal{E}, \quad \text{或者} \quad \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\| \leq \mathcal{E}, \tag{5.1.3}$$

其中 $\mathcal{E}$ 是用户给定的要求。

5.2 标量方程的数值求解

作为本章的重点内容，我们主要讨论单个非线性方程的数值求解技术。在 Matlab 中，我们可用 fzero() 计算猜测值附近的一个根。

5.2.1 区间分半法

✎ **论题 5.1.** 它是闭区间上连续函数介值定理的一个直接应用。若我们能够确定初始区间内仅有一个实根，我们可以将这个区间不断的折半，得到一个区间套序列。在这个区间套序列中，相应两个端点处的函数值保存符号相反。

区间分半算法非常简单，但是它给出的近似程度不是很高，很难推广到非线性方程组。

✎ **论题 5.2.** 作为一个简单实例，我们可以简要探讨舍入误差对算法的影响。例如假停机现象，以及用户要求不能过高等等。

论题 5.3. 类似的方法还有所谓的试位法，即中间位置不是 $[a,b]$ 的中点，而是在两个端点处的线性插值函数同 x 轴的交点

$$c = b - \frac{f(b)(b-a)}{f(b)-f(a)}.$$

其余的处理是类似的。

5.2.2 不动点迭代

更常用的迭代算法是基于不动点理论，即我们找到一个同解于非线性方程 $f(x)=0$ 的不动点方程

$$x = g(x),$$

其中 $g(x)$ 称为迭代函数或者不动点函数。相应的一阶不动点迭代算法可表述为

$$x_{k+1} = g(x_k). \tag{5.2.4}$$

不动点迭代也称为 Picard 迭代。

对于不动点迭代方法 (5.2.4)，迭代函数决定算法的主要性质。常用的收敛性结论有：

定理 5.1. 【压缩映像】若定义在 $[a,b]$ 上的迭代函数 $g(x)$ 满足

1. $g(x) \in [a,b], \forall x \in [a,b]$;
2. 存在 *Lip* 常数 $0 \le L < 1$，使得
   $|g(x)-g(y)| \le L|x-y|, \forall x,y \in [a,b]$;

任取初值 $x_0 \in [a,b]$，不动点迭代序列均收敛到问题的真解 $x_\star$，且满足线性收敛误差估计

$$|e_k| \le \frac{L^k}{1-L}|x_1 - x_0|.$$

证明：略。 □

定理 5.2. 若迭代函数 $g(x)$ 在 $[a,b]$ 上连续可微，且满足

$$g^{(j)}(x_\star)=0,\quad j=0,1,\ldots,m-1;\quad g^{(m)}(x_\star)\neq 0,$$

则称不动点迭代 (5.2.4) 是局部收敛的，且具有 m 阶收敛。

5.2.3 加速迭代收敛

✎ **论题 5.4.** 若迭代序列 $\{x_k\}_{k=0}^{\infty}$ 线性收敛到某个真解 $x_\star$，我们可采用 *Aitken* 加速技术，定义新序列 $\{\tilde{x}_k\}_{k=1}^{\infty}$，其中

$$\tilde{x}_k = x_k - \frac{(x_{k+1}-x_k)^2}{x_{k+2}-2x_{k+1}+x_k}.$$

若 x_k 一直不等于真解 $x_\star$，我们有结论

$$\lim_{k\to\infty}\frac{x_{k+1}-x_\star}{x_k-x_\star}=C<1 \quad\Rightarrow\quad \lim_{k\to\infty}\frac{\tilde{x}_{k+1}-x_\star}{x_k-x_\star}=0.$$

换言之，*Aitken* 加速技术可以明显提升收敛速度。

✯ **说明 5.2.** 图文框中的条件 $C<1$ 是非常重要的。若 $C=1$ 时，*Aitken* 方法可能没有明显的加速效率。相应的反例可考虑序列

$$x_k = 1/k$$

的 *Aitken* 加速过程。尽管如此，在通常情况下，*Aitken* 方法对真解的近似程度仍会有一定的改善；见教科书的例子。

✍ **论题 5.5.** *Aitken* 加速技术的局部应用可形成 *Steffensen* 迭代法，其迭代函数为

$$\psi(x) = x - \frac{[g(x) - x]^2}{g(g(x)) - 2g(x) + x}.$$

该算法的几何解释是，对不动点迭代的残量

$$(x_k, g(x_k) - x_k)^\top$$

进行线性外推。可以证明：在适当的条件下，*Steffensen* 方法具有局部的平方收敛。

5.2.4 Newton 方法

在非线性问题的数值方法中，Newton-Raphson 方法是最著名和最有效的算法之一。据考证，Newton 最早在 1669 年给出这一方法，当时的算例是 $x^3 - 2x - 5 = 0$. 在 1690 年，Raphson 以略有修改的方式发表了这一算法。

✍ **论题 5.6.** *Newton* 方法又称切线法，其迭代公式是

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.}$$

其几何含义就是用切线方程局部线性化非线性方程，利用线性问题的根作为更好的逼近。我们要指出：局部线性化是一种常用的想法，在非线性问题中广泛使用。

Newton 迭代方法的收敛性研究是非常有趣的。它的收敛速度同待解的真解局部信息有关。下面，我们给出一些著名的结论。

为简单起见，我们均假设 $f(x)$ 是一个充分光滑函数；若 $f(x)$ 是一个非光滑函数，相关的 Newton 方法推广及其相应的理论分析是近代数值方法研究的一个热门研究课题，详略。

定理 5.3. *若 $x_\star$ 是 $f(x)=0$ 的单根，则 Newton 迭代方法是局部二阶收敛的。*

定理 5.4. *若 $x_\star$ 是 $f(x)=0$ 的 m 重根，则 Newton 迭代方法退化为线性收敛，且渐近的误差下降速度为 $1-m^{-1}$.*

✯ **说明 5.3.** *重根的出现会导致严重的舍入误差问题。*

✎ **论题 5.7.** *对于 $f(x)=0$ 的重根，我们可利用如下策略提升 Newton 迭代方法的收敛阶:*

1. *若重数 m 是已知的，可简单修正算法为*

$$x_{k+1}=x_k-\frac{mf(x_k)}{f'(x_k)}.$$

2. *若重数 m 是未知的，我们可以考虑新的函数过滤掉根的重数，即考虑新问题 $F(x)=f(x)/f'(x)$ 的 Newton 迭代;*

$$x_{k+1}=x_k-\frac{f(x_k)f'(x_k)}{[f'(x_k)]^2-f(x_k)f''(x_k)}.$$

   为避免二阶导数的计算，我们还可采用 Steffensen 加速算法。

3. *重根 m 的自动探测: 利用标准 Newtond 迭代，计算*

$$h(x_k)=\frac{\ln|f(x_k)|}{\ln|f(x_k)|-\ln|f'(x_k)|}.$$

   当这个值稳定时，跳转到算法 1.

✎ **论题 5.8.** *在适当的条件下，Newton 迭代方法可以做到所谓的整体收敛。例如，设待解的函数 $f(x)$ 在 $[a,b]$ 上是单调保凸的，且端点*

处函数值异号。若从两个端点出发的迭代位置依旧落在这个区间内，则 *Newton* 方法对于这个区间内的任意初值都是收敛的。证明是简单的数学分析问题；详见教科书。

❀ **思考 5.1.** 重要的应用：开根号运算，倒数运算。

5.2.5 割线法

利用历史数据进行非线性函数 $f(x)$ 的局部线性插值，以这个线性方程的根作为更好的迭代位置，我们可得算法

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}.}$$

这个方法也可解释为 Newton 迭代中的导数被一阶差商所逼近。

定理 5.5. *一般来说，割线法的收敛速度稍慢于 Newton 法。在适当条件下，其收敛阶为 1.618.*

✯ **说明 5.4.** 割线法中可能出现分母为零的情形，从而造成算法的意外停机。这时，我们需要执行迭代的重启，即更换旧的迭代位置，给出一个新的猜测值，重新开始割线法的计算。

5.2.6 实多项式的实根计算

我们可以将上述非线性方程的求解方法，直接应用到实系数多项式

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

的求根计算。首先，我们指出数值计算的困难所在。

✍ **论题 5.9.** 若 n 次多项式 $p(x)$ 的系数有微小扰动，即

$$p_\varepsilon(x) = p(x) + \varepsilon q(x),$$

其中 $q(x)$ 是一个次数不超过 n 的多项式。相应的根扰动可表示为

$$x_k(\varepsilon) = x_k - \frac{q(x_k)}{p'(x_k)}\varepsilon,$$

其中 x_k 是原多项式 $p(x)$ 的根。这里的比值大小反映了多项式是否病态。通常，高次多项式求根是一个病态的问题。

在 Newton 方法，我们要计算多项式 $f(x)$ 及其导数在某点的取值。这涉及到多项式的计算问题。

✍ **论题 5.10.** *Horner* 算法或秦九韶算法是常用的算法。注意到多项式除法运算满足

$$f(x) = (x-\mu)g(x) + b_0 = (x-\mu)\sum_{i=1}^{n} b_n x^{i-1} + b_0.$$

显然 $f(\mu) = b_0$，而 $\{b_i\}_{i=0}^n$ 的计算公式为

$$\boxed{b_n = a_n, \quad b_j = a_j + b_{j+1}\mu, \quad \text{for} \quad j = n-1:0.}$$

因为 $f'(\mu) = g(\mu)$，导数值的计算转化为一个新的多项式计算。详略。

✍ **论题 5.11.** 简介抛物线法或 *Muller* 方法。其基本思想与弦截法（即割线法）类似。

✯ **说明 5.5.** 对于一个实系数多项式 $f(x)$，有

1. *Lagrange* 法：设 $a_n > 0$，a_{n-k} 为第一个负系数，b 是负系数中的最大绝对值，则 $f(x)$ 的正根上限为 $1 + (b/a_n)^{1/k}$.

2. *Sturm* 序列法：由 $f(x)$ 与 $f'(x)$ 辗转相除得到的序列称为 *Sturm* 序列。这个 *Sturm* 序列在 μ 点的符号变化的次数（删除掉零值），表示严格大于 μ 的实根数目。
3. *Descartes* 符号律：将实系数多项式按降幂方式排列，则它的正根数目等于相邻非零系数的符号改变个数减去一个非负偶数。

具体讨论超出本课程的要求，详略。

✯ **说明 5.6.** 多项式求根可以转化为矩阵特征值问题，在 *Matlab* 中的对应命令是 *root()*。除此之外，关于多项式的求根，还有很多专门设计的特殊算法。因篇幅限制，本课程不做过多讨论。

5.3 方程组的数值求解

对于非线性方程组 $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$，我们将面临如下的困难：(a) 精确解的数学理论（存在性和唯一性）不清楚；(b) 适用于标量方程的算法原理不再成立，例如二分法；(c) 即使适用于标量方程的算法能够推广到方程组，计算效率将成为重要的研究内容。

5.3.1 预备知识

向量值函数的微积分理论，可以简单理解为多元函数微积分理论的推广。我们不想过于纠缠这些内容的细致讨论，而仅仅列出同后续讨论具有紧密联系的概念和结论。

✇ **定义 5.1.** 设 $\boldsymbol{f}(\boldsymbol{x}) = \{f_i(x_1, x_2, \ldots, x_n)\}_{i=1}^m$ 是 $\mathbb{R}^n \to \mathbb{R}^m$ 的向量值函数，相应的 *Frechet* 导数 $\boldsymbol{f}'(\boldsymbol{x})$ 可以精确地定义为

$$\lim_{\|\triangle \boldsymbol{x}\| \to 0} \frac{\|\boldsymbol{f}(\boldsymbol{x} + \triangle \boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}'(\boldsymbol{x}) \triangle \boldsymbol{x}\|}{\|\triangle \boldsymbol{x}\|} = 0. \tag{5.3.5}$$

若 $\boldsymbol{f}(\boldsymbol{x})$ 的所有偏导数都是连续的，则 *Frechet* 导数就是 $m\times n$ 阶 *Jacobi* 矩阵，即

$$\boldsymbol{f}'(\boldsymbol{x})=D\boldsymbol{f}(\boldsymbol{x})=\begin{bmatrix}\dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \cdots & \dfrac{\partial f_1}{\partial x_n}\\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \cdots & \dfrac{\partial f_2}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \dfrac{\partial f_m}{\partial x_1} & \dfrac{\partial f_m}{\partial x_2} & \cdots & \dfrac{\partial f_m}{\partial x_n}\end{bmatrix}(\boldsymbol{x}). \tag{5.3.6}$$

若 $f(\boldsymbol{x})$ 是标量多元函数，有 $f'(\boldsymbol{x})=[\nabla f(\boldsymbol{x})]^\top$。

✯ **说明 5.7.** 同多元函数的主要区别是，向量值函数不一定满足微分中值定理。设 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 是给定的两个位置，即使函数充分光滑，也不存在一个局部位置 $\boldsymbol{\xi}$，使得

$$\boldsymbol{f}(\boldsymbol{y})-\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{f}'(\boldsymbol{\xi})\cdot(\boldsymbol{y}-\boldsymbol{x}).$$

此时，每个分量 f_i 的对应中值通常是不同的。但是，它们造成的分析困难并不可怕，因为我们有积分表示式

$$\boldsymbol{f}(\boldsymbol{y})-\boldsymbol{f}(\boldsymbol{x})=\int_0^1\boldsymbol{f}'(\boldsymbol{x}+s(\boldsymbol{y}-\boldsymbol{x}))\mathrm{d}s\cdot(\boldsymbol{y}-\boldsymbol{x}).$$

这个积分称为 *Jacobi* 矩阵 $\boldsymbol{f}'$ 从 $\boldsymbol{x}$ 到 $\boldsymbol{y}$ 的 *Riemann* 积分，它对应于每个分量函数的积分。

✎ **论题 5.12.** 在后续的算法研究中，我们还会使用向量范数和矩阵范数，作为重要的度量方式。相关的常用不等式有

1. 积分不等式:

$$\left\|\int_0^1\boldsymbol{f}(t)dt\right\|\le\int_0^1\|\boldsymbol{f}(t)\|dt.$$

2. 设 $\boldsymbol{f}$ 在凸集上处处 *Frechet* 可微，且导数 $\boldsymbol{f}'$ 是 γ-*Lip* 连续，即

$$\|\boldsymbol{f}'(\boldsymbol{y}) - \boldsymbol{f}'(\boldsymbol{x})\| \leq \gamma\|\boldsymbol{y} - \boldsymbol{x}\|, \quad \forall \boldsymbol{y}\, \forall \boldsymbol{x},$$

利用积分表达式，我们有更强的估计

$$\|\boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}'(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})\| \leq \frac{\gamma}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2, \quad \forall \boldsymbol{y}\, \forall \boldsymbol{x}.$$

此时，压缩映像定理依旧是重要的分析工具。

5.3.2 不动点迭代方法

相关概念和分析方法同标量方程类似；此处不再赘述。

5.3.3 Newton 方法

同标量方程的 Newton 方法表述一样，此时的算法是

$$\boxed{\boldsymbol{f}'(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k = -\boldsymbol{f}(\boldsymbol{x}_k), \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k.}$$

换言之，每步 Newton 迭代都涉及到一个线性方程组的求解过程。

Newton 方法的收敛性研究有着非常悠久的历史。在 1829 年,Cauchy 研究了 $n = 1$ 时的局部收敛性，直到 1899 年，Runge 将其推广到 $n \geq 2$ 时的局部收敛性。在 1916 年，Fine 讨论了半局部收敛分析。而后，著名的工作还有 Ostrowski(1936), Willers(1938) 和 Kantovich(1948) 的证明。

定理 5.6. *局部收敛分析: 若 $\boldsymbol{f}'(\boldsymbol{x})$ 在真解 $\boldsymbol{x}_\star$ 附近是连续的非奇异矩阵，则 Newton 方法是局部超线性收敛。*

若进一步假设 $\boldsymbol{f}'$ 在真解 $\boldsymbol{x}_\star$ 附近是 Lipschitz 连续的，则 Newton 方法至少是局部二阶收敛的。

✯ **说明 5.8.** 设序列 $\{\boldsymbol{x}_k\}$ 超线性收敛到 $\boldsymbol{x}_\star$，则其必满足

$$\lim_{k\to\infty}\frac{\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|}{\|\boldsymbol{x}_k-\boldsymbol{x}_\star\|}=1.$$

因此说，在 *Newton* 方法中，我们可以用相邻误差作为迭代误差的估计。

请注意，上述逆命题不成立，如奇偶子列分别定义为

$$\boldsymbol{x}_{2k+1}=\frac{2}{(2k)!},\quad \boldsymbol{x}_{2k}=\frac{1}{(2k)!}.$$

它们虽然满足上述极限关系，但却不是超线性收敛到零。

定理 5.7. *Newton* 方法的半收敛分析：设 $\boldsymbol{f}(\boldsymbol{x})$ 在一个开凸集 D_0 内是处处 *Frechet* 可微的，且满足如下性质：

1. $\|\boldsymbol{f}'(\boldsymbol{x})-\boldsymbol{f}'(\boldsymbol{y})\|\le\gamma\|\boldsymbol{x}-\boldsymbol{y}\|,\forall\boldsymbol{x},\boldsymbol{y}\in D_0$;
2. $[\boldsymbol{f}'(\boldsymbol{x})]^{-1}$ 存在，且 $\|[\boldsymbol{f}'(\boldsymbol{x})]^{-1}\|\le\beta,\forall\boldsymbol{x}\in D_0$;
3. $\|[\boldsymbol{f}'(\boldsymbol{x}_0)]^{-1}\boldsymbol{f}(\boldsymbol{x}_0)\|\le\alpha$.

若 $\boldsymbol{x}_0\in D_0$ 且 $h=\alpha\beta\gamma/2<1$，则 *Newton* 迭代序列至少二阶收敛到问题的某个真解 $\boldsymbol{x}_\star\in S_r(\boldsymbol{x}_0)$，其中 $\gamma=\alpha/(1-h)$。

✯ **说明 5.9.** 由半收敛分析的证明过程可知，收敛条件成立的一个必要条件是

$$\|\boldsymbol{x}_2-\boldsymbol{x}_1\|<\|\boldsymbol{x}_1-\boldsymbol{x}_0\|. \tag{5.3.7}$$

这通常会保证 $\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|$ 的下降。因此，在初始两步的计算中若发现该条件不成立，则我们自动放弃这个没有前途的初值。但是，关于初始解向量 $\boldsymbol{x}_0$ 的选取，依旧是问题的难点。

✯ **说明 5.10.** 称 *Newton* 方法是自校正的，因为 $\boldsymbol{x}_{k+1}$ 仅仅依赖 $\boldsymbol{x}_k$.

✯ **说明 5.11.** 利用 *Newton* 位移的逐次缩减实现所谓的"盲人下山法", 可以找到某个真解的大概位置。然后, 再利用 *Newton* 迭代给出快速的二次收敛。

✯ **说明 5.12.** 当迭代矩阵接近奇异时, 我们可采用基于 *Tiknonov* 正则化方法的修正 *Newton* 算法:

$$\boxed{\Big[\boldsymbol{f}'(\boldsymbol{x}_k)+\lambda_k\mathbb{I}\Big]\Delta\boldsymbol{x}_k=-\boldsymbol{f}(\boldsymbol{x}_k),\quad \boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\Delta\boldsymbol{x}_k.}$$

其中 $\lambda_k>0$ 也称为阻尼因子。

5.3.4 Newton 方法的改进

Newton 方法具有局部的二阶收敛速度, 但是相应的计算效率较差。在每次迭代中, 我们需要计算 Jacobi 矩阵的 n^2 个导数值, 以及求解一个 n 阶的线性代数方程组。下面, 我们考虑 Newton 方法的各种改进方法。

修正 Newton 法

这种方法也称为沙文基思方法, 主要手段是局部锁定 Jacobi 矩阵。对于这些局部同型的线性方程组, 我们只需执行一次 LU 分解, 使线性方程组的求解效率得到明显的提高。

$$\boxed{\boldsymbol{x}_{k,0}=\boldsymbol{x}_k,\ \boldsymbol{x}_{k,j}=\boldsymbol{x}_{k,j-1}-[\boldsymbol{f}'(\boldsymbol{x}_k)]^{-1}\boldsymbol{f}(\boldsymbol{x}_{k,j-1}),\ \boldsymbol{x}_{k+1}=\boldsymbol{x}_{k,m}.}$$

在适当的假设下, 可证修正 Newton 法可达 $m+1$ 阶收敛速度。在实际应用中, 较为常用的是 $m=2$ 的情形。

割线法

割线法改进了 Jacobi 矩阵的计算困难。它使用历史数据，通过适当的方式，给出 Jacobi 矩阵中的导数值近似。依据不同的构造思想，割线法主要包含两大类实现方法：

论题 5.13. 离散的 *Newton* 方法：直接用差商矩阵 $\mathbb{J}(\boldsymbol{x}_k, \boldsymbol{h}_k)$ 代替 *Newton* 方法中的迭代矩阵，可得如下算法

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \Big[\mathbb{J}(\boldsymbol{x}_k, \boldsymbol{h}_k)\Big]^{-1} \boldsymbol{f}(\boldsymbol{x}_k),}$$

其中

$$(\boldsymbol{h}_k)_{ij} = \bar{h}_{ij} \|\boldsymbol{f}(\boldsymbol{x}_k)\|$$

为趋于零的离散化参数。这里的 $\bar{h}_{ij}$ 是事先给定的常数，通常 $\bar{h}_{ij} = \bar{h}_i$。

论题 5.14. 利用非线性方程的局部线性插值思想，或者曲面的局部平面化思想，我们可以将非线性问题的计算，局部转化为一个线性问题的计算。利用线性方程组的解作为非线性方程组的一个近似解，我们可得如下的**割线算法**：

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathbb{A}_k^{-1} \boldsymbol{f}(\boldsymbol{x}_k), \quad \mathbb{A}_k^{-1} = \mathbf{H}_k \boldsymbol{\Gamma}_k^{-1}.}$$

其中

$$\mathbf{H}_k = [\boldsymbol{x}_{k,1} - \boldsymbol{x}_{k,0}, \cdots, \boldsymbol{x}_{k,n} - \boldsymbol{x}_{k,0}], \tag{5.3.8a}$$

$$\boldsymbol{\Gamma}_k = [\boldsymbol{f}_{k,1} - \boldsymbol{f}_{k,0}, \cdots, \boldsymbol{f}_{k,n} - \boldsymbol{f}_{k,0}], \tag{5.3.8b}$$

是由 $n+1$ 个辅助点 $\{\boldsymbol{x}_{k,\ell}, \boldsymbol{f}_{k,\ell}\}_{\ell=0}^{n}$ 生成的矩阵。为使算法具有可操作性，要求辅助点信息处于一般位置，即它们不共处一个平面，或者说 $\mathbf{H}_k$ 和 $\boldsymbol{\Gamma}_k$ 均可逆。否则，我们需要重启算法。

依据辅助点的选取策略,我们有两个著名的割线方法。一直设 $\boldsymbol{x}_{k,0} = \boldsymbol{x}_k$ 为当前位置，而其它辅助点的设置是不同的。

1. 两点序列割线法：

$$\boldsymbol{x}_{k,\ell} = \boldsymbol{x}_k + \{(\boldsymbol{x}_{k-1})_\ell - (\boldsymbol{x}_k)_\ell\}\boldsymbol{e}_\ell, \tag{5.3.9a}$$

$$\boldsymbol{x}_{k,\ell} = \boldsymbol{x}_k + \sum_{j=1}^{\ell}\{(\boldsymbol{x}_{k-1})_j - (\boldsymbol{x}_k)_j\}\boldsymbol{e}_j, \tag{5.3.9b}$$

其中 $(\boldsymbol{x}_k)_\ell$ 表示 $\boldsymbol{x}_k$ 的第 ℓ 个分量。在每次迭代中，前者需要计算 n^2+n 个函数值，而后者仅仅需要计算 n^2 个函数值。

2. $(n+1)$ 点序列割线法：

$$\boldsymbol{x}_{k,\ell} = \boldsymbol{x}_{k-\ell}, \quad \ell = 1:n.$$

在每次迭代中，我们仅仅需要计算 n 个函数值。

无论哪种方法，我们可证：在适当的条件下，p 点序列割线法的收敛阶为 $\lambda^p = \lambda^{p-1} + 1$ 的最大正根。

✯ **说明 5.13.** 虽然 $(n+1)$ 点序列割线法收敛阶占优，但其容易产生数值不稳定。

✯ **说明 5.14.** 效率的比较：从高到低，依次为 $(n+1)$ 点序列割线法、两点割线法、和离散 *Newton* 方法。

5.3.5 拟 Newton 方法

拟 Newton 法是上世纪 60 年代发展起来的高效解法。它继承了 $(n+1)$ 点序列割线法的优点，避免了导数值的计算。同时，拟 Newton 方法克服了 Newton 迭代中线性方程组求解的困难。

基本思想

拟 Newton 方法可视为割线法的推广。拟 Newton 法的基本结构为：任取初始向量 $\boldsymbol{x}_0$ 和初始迭代矩阵 $\mathbb{B}_0$。假设 $\boldsymbol{x}_k$ 和 $\mathbb{B}_k$ 是已知的，我们按公式

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathbb{B}_k^{-1}\boldsymbol{f}(\boldsymbol{x}_k),}$$

计算出新的位置 $\boldsymbol{x}_{k+1}$。我们希望利用相邻两步的信息，构造下一步的迭代矩阵 $\mathbb{B}_{k+1}$。继承 $(n+1)$ 点序列割线法的主要特征，迭代矩阵满足拟 Newton 方程

$$\boxed{\boldsymbol{y}_k = \mathbb{B}_{k+1}\Delta\boldsymbol{x}_k,}$$

其中 $\boldsymbol{y}_k = \boldsymbol{f}_{k+1} - \boldsymbol{f}_k := \boldsymbol{f}(\boldsymbol{u}_{k+1}) - \boldsymbol{f}(\boldsymbol{u}_k)$ 和 $\Delta\boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$。拟 Newton 方程包含 n^2 个未知数，但仅有 n 个约束条件，所以当 $n > 1$ 时它有无穷多解。

为提高线性方程组的求解效率，我们希望新的迭代矩阵 $\mathbb{B}_{k+1}$ 同旧的迭代矩阵 $\mathbb{B}_{k+1}$ 具有一定的相似性。让回忆 $(n+1)$ 点序列割线法的迭代矩阵所满足的性质

$$\mathbb{A}_k\boldsymbol{p}_j = \boldsymbol{q}_j, \quad j = k-1, k-2, \ldots, k-n; \tag{5.3.10a}$$

$$\mathbb{A}_{k+1}\boldsymbol{p}_j = \boldsymbol{q}_j, \quad j = k, k-1, \ldots, k-n+1, \tag{5.3.10b}$$

其中 $\boldsymbol{p}_j = \boldsymbol{x}_{j+1} - \boldsymbol{x}_j, \boldsymbol{q}_j = \boldsymbol{f}_{j+1} - \boldsymbol{f}_j$. 这些蕴含着

$$(\mathbb{A}_{k+1} - \mathbb{A}_k)\underbrace{[\boldsymbol{p}_{k-1}, \ldots, \boldsymbol{p}_{k-n+1}]}_{\mathbf{H}_k} = [0, 0, \ldots, 0]. \tag{5.3.11}$$

因为矩阵 $\mathbf{H}_k$ 是列满秩的，因此 $\mathbb{A}_{k+1}$ 是 $\mathbb{A}_k$ 的秩一修正矩阵。鉴于此，我们将重点介绍基于秩一修正的拟 Newton 方法，即 Broyden (1965) 方法。

定理 5.8. 给定方向 $\boldsymbol{v}_k$，设相邻的两个 *Jacobi* 矩阵满足所谓的秩一修正，即

$$\mathbb{B}_{k+1} = \mathbb{B}_k + \boldsymbol{u}_k \boldsymbol{v}_k^\top,$$

其中 $\boldsymbol{u}_k$ 是待定的方向。显然 $\mathbb{B}_{k+1}$ 和 $\mathbb{B}_k$ 作用在 $span^\perp(\boldsymbol{v}_k)$ 的像是一样的。若其满足拟 *Newton* 方程，则应取

$$\boldsymbol{u}_k = (\boldsymbol{y}_k - \mathbb{B}_k \Delta \boldsymbol{x}_k)/\boldsymbol{v}_k^\top \Delta \boldsymbol{x}_k.$$

此时，$\mathbb{B}_{k+1}$ 是极小问题 $\min\{\|\mathbb{S} - \mathbb{B}_k\|_F : \mathbb{S}\Delta \boldsymbol{x}_k = \boldsymbol{y}_k\}$ 的解。

✎ **论题 5.15.** 记 $\mathbb{H}_k$ 是 $\mathbb{B}_k$ 的逆矩阵。在 *Broyden* 方法中，利用 *Sherman-Morrison* 公式，迭代矩阵的逐步修正策略是

$$\boxed{\mathbb{H}_{k+1} = \mathbb{H}_k + \frac{(\Delta \boldsymbol{x}_k - \mathbb{H}_k \boldsymbol{y}_k)\boldsymbol{d}_k^\top}{\boldsymbol{d}_k^\top \boldsymbol{y}_k} = \mathbb{H}_k - \frac{\mathbb{H}_k \boldsymbol{f}_{k+1} \boldsymbol{d}_k^\top}{\boldsymbol{d}_k^\top \boldsymbol{y}_k},}$$

其中 $\boldsymbol{d}_k = \mathbb{H}_k^\top \boldsymbol{v}_k$。关于 $\boldsymbol{v}_k$ 的选取主要有两种方式，即

$$\boldsymbol{v}_k = \Delta \boldsymbol{x}_k, \quad \text{或} \quad \boldsymbol{v}_k = \boldsymbol{f}(\boldsymbol{x}_{k+1}).$$

后者更适宜所谓的对称问题求解，它可以一直保持 $\mathbb{H}_k$ 的对称性。

将上述公式融合到拟 *Newton* 迭代中，为获得新的迭代位置，我们共需要 $\mathcal{O}(n^2)$ 次乘除法运算。

✯ **说明 5.15.** 拟 *Newton* 方法的收敛阶不如 *Newton* 方法高。我们可以证明：在适当的条件下，*Broyden* 方法具有局部的超线性收敛。

✯ **说明 5.16.** 我们要指出：为保证拟 *Newton* 方法具有超线性收敛，我们并不需要求 $\mathbb{B}_k$ 具有收敛性。事实上，若它是收敛的，为保证其具有超线性收敛，我们只需一个较弱的充要条件

$$\lim_{k\to\infty} \frac{\|[\mathbb{B}_k - \boldsymbol{f}'(\boldsymbol{x}_\star)]\Delta\boldsymbol{x}_k\|}{\|\Delta\boldsymbol{x}_k\|} = 0.$$

在适当的条件下，上述条件等价于

$$\begin{aligned}\boldsymbol{s}_k^{\mathrm{Qn}} - \boldsymbol{s}_k^{\mathrm{Nt}} &= \Delta\boldsymbol{x}_k + [\boldsymbol{f}'(\boldsymbol{x}_k)]^{-1}\boldsymbol{f}'(\boldsymbol{x}_k)\\ &= [\boldsymbol{f}'(\boldsymbol{x}_k)]^{-1}\{\boldsymbol{f}'(\boldsymbol{x}_k) - \mathbb{B}_k\}\Delta\boldsymbol{x}_k \to 0,\end{aligned}$$

其中 $\boldsymbol{s}_k^{\mathrm{Qn}}$ 是拟 *Newton* 方法中的修正方向，而 $\boldsymbol{s}_k^{\mathrm{Nt}}$ 是原始 *Newton* 迭代方法的修正方向。

❀ **思考 5.2.** 换言之，要得到所谓的超线性收敛算法，我们不必要求迭代矩阵 $\mathbb{B}_k$ 趋向于真解 $\boldsymbol{x}_\star$ 处的 *Jacobi* 矩阵 $\boldsymbol{f}'(\boldsymbol{x}_\star)$. 考虑

$$\boldsymbol{f}(\boldsymbol{x}) = (x_1, x_2^2 + x_2)^\top$$

的求根问题，其真解为 $\boldsymbol{x}_\star = (0,0)^\top$。若初始猜测值和初始迭代矩阵分别取为

$$\boldsymbol{x}_0 = (0,\varepsilon)^\top, \quad \mathbb{B}_0 = \begin{bmatrix} 1+\delta & 0 \\ 0 & 1 \end{bmatrix}.$$

请验证：$\mathbb{B}_k$ 在 $(1,1)$ 位置处的元素不收敛到 $\boldsymbol{f}'(\boldsymbol{x}_\star)$ 的对应元素。

我们也可以假设 $\mathbb{B}_{k+1}$ 是 $\mathbb{B}_k$ 的秩二修正矩阵,构造相应的拟 Newton 方法。著名的算法有 DFP 方法和 BFS 方法，略。

5.3.6 极值算法

非线性方程组 $\boldsymbol{f}(\boldsymbol{x}) = 0$ 等价于最优化问题 $\min_{\forall\boldsymbol{x}} \|\boldsymbol{f}(\boldsymbol{x})\|_2^2$，我们可采用各种优化方法，例如最速下降方法等进行计算；详略。

事实上，很多最优化问题同非线性方程组的求根是密切相关的，因为目标函数在极值点处的梯度通常是等于零的。

5.3.7 延拓方法

略。

第 6 章

数值实验

在本学期的数值实验中，我们重点考虑下面的两个矩阵。它们均对应一个椭圆方程的有限差分离散过程。

例 1. 考虑单位区间 $[0,1]$ 上的两点边值问题

$$-u''(x) = g(x), \quad x \in (0,1); \qquad u(0) = u(1) = 0,$$

我们在在网格点 $\{ih\}_{i=1:n}$ 处，利用差商代替导数，可得差分方程

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 g(ih), \quad i = 1:n,$$

其中 $h = 1/(n+1)$ 为网格步长，u_i 是 $u(ih)$ 的数值近似。注意到零边值条件，差分方程的全体对应一个线性方程组

$$\mathbb{T}_n \boldsymbol{x} = \boldsymbol{b}_n, \tag{6.0.1}$$

其中 $\boldsymbol{x} = (u_1, u_2, \ldots, u_n)^\top$ 是数值解向量，

$$\mathbb{T}_n = \mathrm{tridiag}(-1, 2, -1) = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \tag{6.0.2}$$

是 n 阶的三对角对称方阵。

例 2. 考虑单位正方形区域 $(0,1)\times(0,1)$ 上的 Poisson 方程

$$-\triangle u(x,y) = f(x,y), \quad (x,y) \in (0,1)^2;$$
$$u(x,y) = 0, \quad x = 0, 1 \text{ 或者 } y = 0, 1.$$

在网格点 $\{(ih,jh)\}_{i=1:n}^{j=1:n}$ 处，利用差商代替导数，我们可以给出 Poisson 方程的五点差分格式

$$4u_{ij}-u_{i-1,j}-u_{i+1,j}-u_{i,j-1}-u_{i,j+1}=h^2f(ih,jh),$$

其中 $h=1/(n+1)$ 为网格步长，u_{ij} 是 $u(ih,jh)$ 的近似。注意到零边值条件，我们可得线性方程组

$$\mathbb{A}_n\boldsymbol{x}=\boldsymbol{b}_n, \tag{6.0.3}$$

其中 $\boldsymbol{x}$ 是 $\{u_{ij}\}_{i=1:n}^{j=1:n}$ 按照从下到上、从左到右的次序，逐行构成的数值解向量。系数矩阵 $\mathbb{A}_n$ 是一个 n^2 阶的对称正定矩阵

$$\begin{aligned}\mathbb{A}_n&=\begin{bmatrix}\mathbb{T}_n+2\mathbb{I}_n & -\mathbb{I}_n & & \\ -\mathbb{I}_n & \mathbb{T}_n+2\mathbb{I}_n & & \\ & \ddots & \ddots & -\mathbb{I}_n \\ & & -\mathbb{I}_n & \mathbb{T}_n+2\mathbb{I}_n\end{bmatrix}\\ &=\mathbb{T}_n\otimes\mathbb{I}_n+\mathbb{I}_n\otimes\mathbb{T}_n,\end{aligned} \tag{6.0.4}$$

其中 $\mathbb{I}_n$ 为 n 阶单位矩阵，而 $\mathbb{T}_n$ 由 (6.0.2) 给出。

6.1 线性方程组的直接解法

❖ **练习 6.1.** 设 $\boldsymbol{x}_\star=(1,1,1,\ldots,1)^\top$ 为线性方程组 (6.0.3) 的真解，右端向量由这个真解给出。

1. 编制高斯消元法和 $LDL^\top$ 算法，求解线性方程组。

2. 利用纵轴上的对数坐标，绘制数值误差同参数 n 的关系图。

3. 绘制所用的 *CPU* 时间同参数 n 的关系图。

4. 绘制矩阵条件数与参数 n 的关系。请问：摄动理论给出的舍入误差估计 (1.5.18) 是否完美地刻画了相对误差的大小？

❖ **练习 6.2.** 矩阵非零元素的分布对数值计算的影响，这个问题最优答案的确定是个 *NP* 问题。考虑行列重排后相等的两个矩阵

$$\mathbb{B}_1=\begin{bmatrix}1 & & & & a\\ & 1 & & & a\\ & & \ddots & & \vdots\\ & & & 1 & a\\ a & a & \cdots & a & 1\end{bmatrix},\quad \mathbb{B}_2=\begin{bmatrix}1 & a & \cdots & a & a\\ a & 1 & & & \\ \vdots & & \ddots & & \\ a & & & 1 & \\ a & & & & 1\end{bmatrix},\qquad (6.1.5)$$

比较三角分解后的非零元素分布。在 *Matlab* 中观测矩阵结构图的命令是 *spy()*。利用矩阵的元素分布特点修改 *Crout* 算法，努力尝试省掉那些无用的零运算时间，观察是否有 *CPU* 时间的节省。

❖ **练习 6.3.** 求 $\mathbb{T}_n$ 的逆矩阵。

❖ **练习 6.4.** 考虑线性方程组 $\mathbb{D}_n\mathbb{T}_n\boldsymbol{x}=\boldsymbol{b}$，其中 $\mathbb{D}_n=diag(2^{-i})$。取真解为 $\boldsymbol{x}_\star=(1,2,3,\ldots,n)^\top$，令 n 从 *500* 变换到 *2000.*

1. 观测追赶法的数值误差与矩阵阶数 n 的关系。

2. 考虑等价的问题 $\mathbb{T}_n\boldsymbol{x}=\mathbb{D}_n^{-1}\boldsymbol{b}$，追赶法的计算结果能否得到改善？

❖ **练习 6.5.** 随机构造大量的 n 阶可逆矩阵 $\mathbb{A}$，进行列主元高斯消元分解。观测主元增长因子 $\eta(\mathbb{A})$ 是否处于 $n^{2/3}$ 或 $n^{1/2}$ 的量级。

6.2 线性方程组的迭代解法

考虑线性方程组 (6.0.3)，取真解为 $\boldsymbol{x}_\star=(1,1,\ldots,1)^\top$。在本章的数值实验中，初始向量均取为零，以 ℓ_2 范数或无穷范数为度量工具。取误

差指标为 $\mathcal{E} = 10^{-6}$。

❖ **练习 6.6.** 编制程序实现 *Jacobi* 迭代方法和 *Gauss-Seidel* 方法。对应不同的停机标准（例如残量，相邻差量，后验误差停机标准），比较迭代次数以及算法停止时的真实误差。

❖ **练习 6.7.** 编写程序实现 *SOR* 迭代方法。以真实误差作为停机标准，数值观测 *SOR* 迭代方法中松弛因子 ω 对迭代次数的影响，找到最佳迭代因子的取值。

❖ **练习 6.8.** 对于 *J* 方法、*GS* 方法和（带有最佳松弛因子的）*SOR* 方法，分别绘制误差下降曲线以及残量的下降曲线（采用对数坐标系），绘制（按真实误差）迭代次数与矩阵阶数倒数的关系；

❖ **练习 6.9.** 编制变系数 *Richardson* 迭代方法，绘制误差下降曲线以及残量的下降曲线。观测循环指标 m 对收敛速度的影响。

❖ **练习 6.10.** 对 *Jacobi* 迭代进行半迭代加速（考虑 m=5 的循环迭代），绘制误差下降曲线以及残量的下降曲线。

❖ **练习 6.11.** 共轭梯度算法的研究。比较 $n = 100, 101$ 时 *CG* 算法与 *SOR* 算法的迭代次数；绘制误差下降曲线以及残量的下降曲线；绘制迭代次数与矩阵阶数的关系。

❖ **练习 6.12.** 编制预处理 *CG* 算法，采用 *SSOR* 做为预处理因子。取定矩阵阶数，考察迭代次数与 ω 的关系。

6.3 线性最小二乘问题

❖ **练习 6.13.** 随机构造一个可逆方阵，利用不同方法给出它的 *QR* 分解。观测所得列向量的正交性、*CPU* 时间和所谓的向后稳定性。

❖ **练习 6.14.** 实现本章的两张图。

❖ **练习 6.15.** 考虑列满秩的最小二乘问题 $\mathbb{A}_{n\times(n-1)}\boldsymbol{x}_{n-1}=\boldsymbol{b}_n$，其中系数矩阵和右端项分别为

$$\mathbb{A}_{n\times(n-1)}=\mathbb{T}_n(1:n,1:n-1)=\begin{bmatrix}2&-1&&&\\-1&2&-1&&\\&\ddots&\ddots&\ddots&\\&&-1&2&-1\\&&&-1&2\\&&&&-1\end{bmatrix},\quad(6.3.6)$$

$$\boldsymbol{b}_n=\left(1+\frac{1}{n},\frac{2}{n},\frac{3}{n},\ldots,\frac{n-2}{n},1+\frac{n-1}{n},0\right)^{\top}.\quad(6.3.7)$$

对应的最小二乘解为 $\boldsymbol{x}_{LS}=(1,1,1,\ldots,1,1)^{\top}$. 分别用

(a) 法方程组; *(b)MGS* 方法; *(c)Householder* 法; 和 *(d)Givens* 法

四种方法求解这个最小二乘问题。

令 n 从 *5* 增加到 *30*, 绘图比较上述四种算法的计算工作量 (可用 *cpu* 时间表示) 和计算精度与 n 的关系。

6.4 矩阵特征值问题

考虑三对角对称矩阵 $\mathbb{T}_n$, 见 (6.0.2)。矩阵阶数分别取为 $n=100$ 和 $n=101$, 要求精确计算到准确特征值的小数点后第 6 位。

❖ **练习 6.16.** 取初始向量为 $\boldsymbol{v}_0=(1,1,1,\ldots,1)^{\top}$, 用乘幂法计算主特征值及其相应的特征向量。请绘制主特征值误差的下降曲线, 以及特征子空间距离的下降曲线。然后, 请采用 *Atiken* 加速技巧和 *Rayleigh* 商技术分别对算法进行加速, 并完成类似的工作。

❖ **练习 6.17.** 用反幂法求解最靠近 *2* 的特征值，及其对应的特征向量。观察是否有所谓的“一次迭代”特性。

❖ **练习 6.18.** 用幂法求解第二主特征值及其特征向量；用同时迭代方法求解 $\mathbb{T}_n$ 的前两个主特征值。比较两者的计算效果。

❖ **练习 6.19.** 分别用古典 *Jacobi* 方法、循环 *Jacobi* 和阈值 *Jacobi* 方法求解 $\mathbb{T}_n$ 的全部特征值；绘制相应的收敛过程。

❖ **练习 6.20.** 用二分法加原点位移反幂法求解在 (*1*, *2*) 间的特征值；

❖ **练习 6.21.** 用对称隐式 *QR* 方法求解全部特征值。

❖ **练习 6.22.** 阈值 *Jacobi* 方法具有求解小特征值的优势。考虑对称正定矩阵

$$\mathbb{A} = \begin{bmatrix} 10^{40} & 10^{29} & 10^{19} \\ 10^{29} & 10^{20} & 10^{9} \\ 10^{19} & 10^{9} & 1 \end{bmatrix}$$

直接计算可知其特征值为 $10^{40}, 9.9 \times 10^{19}, 9.81818 \times 10^{-1}$。请用阈值 *Jacobi* 方法求解三个特征值。在 *Matlab* 中，我们可以利用 *eig()* 计算矩阵的特征值。请比较它们的数值结果。

6.5 非线性方程 (组) 的数值方法

❖ **练习 6.23.** 用 *Newton* 方法求解方程 $x^3 - x^2 - 8x + 12 = 0$ 的根，并绘制误差下降曲线。然后，试用割线法重复上述工作。

❖ **练习 6.24.** 编制 *Newton* 方法和 *Broyden* 方法求解非线性方程

组

$$\begin{cases} (x+3)(y^2-7)+18=0, \\ \sin(ye^x-1)=0, \end{cases} \tag{6.5.8}$$

取相同的初值 $(-0.15, 1.4)$，比较两者的迭代次数；若初值为 $(0,1)$ 呢？观察 *Broyden* 中的 $\mathbb{B}_k$ 是否收敛到 *Jacobi* 矩阵呢？若非线性方程组为

$$\begin{cases} x+y-3=0, \\ x^2+y^2-9=0, \end{cases} \tag{6.5.9}$$

呢？初值取为 $(2,4)$, 观察 *Broyden* 中的 $\mathbb{B}_k$ 是否收敛到 *Jacobi* 矩阵呢？

❖ **练习 6.25.** 编制修正 *Newton* 法（与 m 的关系）、离散 *Newton* 法和两点序列割线法求解(6.5.9)。

❖ **练习 6.26.** 非线性方程组的应用：*Newton* 法可应用于特征值问题的求解，方式如下

$$\begin{cases} \mathbb{T}\boldsymbol{x}-\lambda\boldsymbol{x}=0, \\ \boldsymbol{x}^\top\boldsymbol{x}=1. \end{cases} \tag{6.5.10}$$

取 $\mathbb{T}$ 为以前的三对角矩阵，阶数为 $n=5$；取任意的初值 $\boldsymbol{x}_0$ 和 $\lambda_0=\boldsymbol{x}_0^\top\mathbb{T}\boldsymbol{x}_0$，做 *Newton* 迭代。将得到的结果与幂法做比较。

qzh_nk@aliyun.com