第3章 线性方程组直接求解

CHAPTER

万里长征，是一步一步走出来的。——线性方程与线性方程组的直接求解

3.1 引 言

工程技术、科学研究中的很多实际问题，如电学中的网络问题、最小二乘法的曲线拟合问题、三次样条插值问题、自由振动问题等，都需要求解线性方程组。n 阶线性方程组的一般形式为

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \quad\vdots \qquad\quad \vdots \quad\ \vdots \quad\ \vdots \quad\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

式中，a_{ij} 和 b_i 为常数；x_i 为变元，这里 $i,j=1,2,\cdots,n$。

n 阶线性方程组一般形式的矩阵形式为

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

式中，

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

其中，$\boldsymbol{A}$ 为系数矩阵，$\boldsymbol{b}$ 为右端向量，$\boldsymbol{x}$ 为解向量。常把右端向量 $\boldsymbol{b}$ 并入系数矩阵 $\boldsymbol{A}$，作为 $\boldsymbol{A}$ 的第 $n+1$ 列，记为

$$\overline{\boldsymbol{A}} = [\boldsymbol{A} \quad \boldsymbol{b}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix}$$

称 $\overline{\boldsymbol{A}}$ 为增广矩阵。

线性方程组分类的一种方法是按阶数高低分类，阶数在 150 以上的线

性方程组为高阶线性方程组;阶数在 100 以下的线性方程组为低阶线性方程组。线性方程组分类的另一种方法是按系数矩阵 $\boldsymbol{A}$ 中零元素的多少来分类,系数矩阵 $\boldsymbol{A}$ 的大部分元素(大约在 80%以上)是零元素的线性方程组为稀疏线性方程组;系数矩阵 $\boldsymbol{A}$ 的大部分元素(大约在 80%以上)是非零元素的线性方程组为稠密线性方程组。实际应用中,经常见到高阶稀疏线性方程组、低阶稠密线性方程组、对称正定线性方程组和带状线性方程组等。

由克拉默法则可知,若系数行列式 $\det(\boldsymbol{A})\neq 0$,则线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解。但是克拉默法则并不实用。当用克拉默法则求解 n 阶线性方程组时,总乘除次数 MD=$(n+1)$个行列式×$[(n^2-1)\cdot n!$次乘法+n 次除法$]$。随着方程组阶数的增加,求解的总计算量以不低于指数级别的速度增长。例如,当阶数 $n=20$ 时,总乘除次数 MD≈9.7×10^{20};当阶数 $n=30$ 时,MD≈7.4×10^{36};当阶数 $n=40$ 时,MD≈5.35×10^{52},这时就算用每秒做 1 千万亿次乘除法的计算机求解,耗费的年数也是个天文数字。

线性方程组常用的数值解法主要分为两类:直接求解方法和迭代求解方法。

直接求解方法是指经过有限次四则运算,可以求出线性方程组精确解的方法。虽然运算方法在理论上没有误差,但实际计算过程一般会产生舍入误差。有时舍入误差大得无法忽略,因此在设计算法时必须考虑舍入误差的增长,估计运算结果误差的大小。本章将介绍直接求解方法中的消元法和三角分解法,这些方法求解 n 阶线性方程组的时间复杂度为 $O(n^3)$,空间复杂度为 $O(n^2)$,求解低阶稠密矩阵时效率较高。

迭代求解方法是指构造一种迭代方法,由某个(套)迭代初值(粗略解),得到近似解序列,用序列极限逐步逼近线性方程组精确解的方法。与第 2 章非线性方程迭代求根相比较,线性方程组迭代求解需要一套数据才能够启动迭代过程,一次迭代的结果也包含多个数值,却仍为单点迭代法。迭代求解方法存在收敛性和收敛速度的问题。如果收敛,那么随着运算量的增大,误差会趋向无穷小。迭代求解的程序设计简单,内存需求小。求解高阶稀疏矩阵时,迭代求解方法效率较高。第 4 章将介绍线性方程组的迭代求解方法。

3.2 顺序高斯消元法

用顺序高斯消元法求解线性方程组的过程分为消元过程和回代过程。消元过程是自上而下,把原线性方程组化为上三角方程组的过程;回代过程是自下而上,对这个上三角方程组进行求解的过程。

3.2.1 消元过程

消元的过程是对线性方程组做同解变换,把原线性方程组变为上三角方程组的过程。消元的次序,是先用第 1 个方程消去它之下所有方程左边第 1 个变元,并更新其他变元的系数,完成第 1 次循环;然后不再用到第 1 个方程(最上边 1 行)和它下边所有方程的最左边 1 列(变元系数全为 0),对右下剩余的矩形区域,重新消去最上边方程之下所有方程的最左边 1 列,更新其他变元的系数,完成第 2 次循环;如此反复,直到把这个线性方程组化为上三角方程组。具体地说,消去某一个方程的第一个变元,就是让矩形区域最上边方程

等号的左右两端都乘以 1 个因子，加到待消元的方程上，使待消元方程第一个变元的系数为 0，从而实现这一步消元。

为了用程序实现消元和回代，需要用符号表示方程组的一般形式。n 阶线性方程组在消元过程一开始，可以表示为

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1,n}^{(1)}x_n = a_{1,n+1}^{(1)} \\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + \cdots + a_{2,n}^{(1)}x_n = a_{2,n+1}^{(1)} \\ \quad\vdots \qquad\qquad\qquad \vdots \\ a_{n,1}^{(1)}x_1 + a_{n,2}^{(1)}x_2 + \cdots + a_{n,n}^{(1)}x_n = a_{n,n+1}^{(1)} \end{cases}$$

下面说明其中符号的含义：

(1) x_j 为变元。下标 j 为变元序号，且 j 为列号。

(2) $a_{ij}^{(k)}$ 为变元 x_j 的系数。下标 i 为行号，下标 j 为列号。$a_{ij}^{(k)}$ 每更新(改变)一次，上标 k 增 1。k 初值为 1，因此 k 为 $a_{ij}^{(k)}$ 的更新次数+1。

外层循环每循环一轮，就用涉及的矩形区域中上边的第 1 行消去它下边各行左边的第 1 列，此矩形区域最上 1 行和最左 1 列在之后的消元过程中不再涉及(上标不再改变)，除此之外的右下矩形区域中所有的系数被更新一遍(上标 k 增加 1)。第 k 次消元的初始状态为

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1,k}^{(1)}x_k + \cdots + a_{1,n}^{(1)}x_n = a_{1,n+1}^{(1)} \\ \qquad a_{22}^{(2)}x_2 + \cdots + a_{2,k}^{(2)}x_k + \cdots + a_{2,n}^{(2)}x_n = a_{2,n+1}^{(2)} \\ \qquad\qquad \ddots \quad\vdots \qquad\vdots \qquad\vdots \\ \qquad\qquad\qquad a_{k,k}^{(k)}x_k + \cdots + a_{k,n}^{(k)}x_n = a_{k,n+1}^{(k)} \\ \qquad\qquad\qquad\quad \vdots \qquad\vdots \qquad\vdots \\ \qquad\qquad\qquad a_{n,k}^{(k)}x_k + \cdots + a_{n,n}^{(k)}x_n = a_{n,n+1}^{(k)} \end{cases}$$

在第 k 次更新的初始，涉及的待更新矩形区域中，系数的上标都为 k，此矩形区域中最左上角的系数的行标、列标也为 k。更新的过程，就是用第 k 行消去第 $k+1$ 行至第 n 行的第 k 列，并且让第 $k+1$ 行至第 n 行、第 $k+1$ 列至第 $n+1$ 列的矩形区域中的系数更新，上标变为 $k+1$。在此后的消元过程中，第 k 行及之上和第 k 列及其左边的系数不再变化。

若以上标 k 为循环变量，则每循环 1 次，消去 1 列。消元过程中，k 共循环 $n-1$ 次，消去 $n-1$ 列，把原方程组变为上三角方程组。上标 k 每循环 1 次，需要更新第 $k+1$ 行至第 n 行、第 $k+1$ 列至第 $n+1$ 列的矩形区域中的系数，这一过程需要 2 重循环，因此消元过程总共需要 3 重循环，外层循环变量为上标 k，中层循环变量为行标 i，内层循环变量为列标 j。

消元结束之后得到的上三角方程组中，每向下一行，上标 k 增加 1，所以上标 k 等于行标 i，如下所示：

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1,n}^{(1)}x_n = a_{1,n+1}^{(1)} \\ \qquad a_{22}^{(2)}x_2 + \cdots + a_{2,n}^{(2)}x_n = a_{2,n+1}^{(2)} \\ \qquad\qquad \ddots \quad\vdots \qquad\vdots \\ \qquad\qquad\qquad a_{n,n}^{(n)}x_n = a_{n,n+1}^{(n)} \end{cases}$$

下面举例说明消元的具体过程,如表 3.1 所示。

表 3.1 消元的具体过程

实 例	符 号 表 示
$\begin{cases}x+2y-z=2 & ①\\ 3x-y+z=4 & ②\\ 3x+2y-2z=1 & ③\end{cases}$	$\begin{cases}a_{11}^{(1)}x_1+a_{12}^{(1)}x_2+a_{13}^{(1)}x_3=a_{14}^{(1)} & ①\\ a_{21}^{(1)}x_1+a_{22}^{(1)}x_2+a_{23}^{(1)}x_3=a_{24}^{(1)} & ②\\ a_{31}^{(1)}x_1+a_{32}^{(1)}x_2+a_{33}^{(1)}x_3=a_{34}^{(1)} & ③\end{cases}$
$①\times\left(-\frac{3}{1}\right)+②\to②$ $①\times\left(-\frac{3}{1}\right)+③\to③$	令 $l_{21}=-\frac{a_{21}^{(1)}}{a_{11}^{(1)}}$,则$①\times l_{21}+②\to②$ 令 $l_{31}=-\frac{a_{31}^{(1)}}{a_{11}^{(1)}}$,则$①\times l_{31}+③\to③$
$\begin{cases}x+2y-z=2 & ①\\ -7y+4z=-2 & ②\\ -4y+z=-5 & ③\end{cases}$	$\begin{cases}a_{11}^{(1)}x_1+a_{12}^{(1)}x_2+a_{13}^{(1)}x_3=a_{14}^{(1)} & ①\\ a_{22}^{(2)}x_2+a_{23}^{(2)}x_3=a_{24}^{(2)} & ②\\ a_{32}^{(2)}x_2+a_{33}^{(2)}x_3=a_{34}^{(2)} & ③\end{cases}$
$②\times\left(-\frac{-4}{-7}\right)+③\to③$	令 $l_{32}=-\frac{a_{32}^{(2)}}{a_{22}^{(2)}}$,则$②\times l_{32}+③\to③$
$\begin{cases}x+2y-z=2 & ①\\ -7y+4z=-2 & ②\\ -\frac{9}{7}z=-\frac{27}{7} & ③\end{cases}$	$\begin{cases}a_{11}^{(1)}x_1+a_{12}^{(1)}x_2+a_{13}^{(1)}x_3=a_{14}^{(1)} & ①\\ a_{22}^{(2)}x_2+a_{23}^{(2)}x_3=a_{24}^{(2)} & ②\\ a_{33}^{(3)}x_3=a_{34}^{(3)} & ③\end{cases}$

表 3.1 中的 $l_{i,k}$ 称为行乘子,其计算公式为

$$l_{i,k}=-\frac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}\quad i\in[k+1,n]$$

当外层循环为第 k 次循环时,用方程 k 消去方程 i 的变元 x_k,实现对第 i 行的系数 $a_{i,j}^{(k)}$ 的 1 次更新的过程为:将方程 k(第 k 行)的对应系数 $a_{k,j}^{(k)}(j\in[k,n+1])$ 乘以行乘子 $l_{i,k}$,加上 $a_{i,j}^{(k)}$,得到 $a_{i,j}^{(k+1)}$。这一过程的计算公式为

$$a_{i,j}^{(k+1)}=a_{k,j}^{(k)}\times l_{i,k}+a_{i,j}^{(k)}\quad i\in[k+1,n],j\in[k,n+1]\tag{3.1}$$

式(3.1)的主要目的是消去方程 i 的变元 x_k,即要求 $a_{i,k}^{(k+1)}=0$,那么 $a_{i,k}^{(k+1)}=a_{k,k}^{(k)}\times l_{i,k}+a_{i,k}^{(k)}=0$,对此等式变形可得 $l_{i,k}=-\frac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}$,与行乘子 $l_{i,k}$ 的计算公式一致。

当 $a_{k,k}^{(k)}=0$ 时,上述计算过程无法进行;当 $a_{k,k}^{(k)}$ 的绝对值很小时,误差会被放大。

定义 3.1 $a_{k,k}^{(k)}(k=1,2,\cdots,n)$ 对求解起突出作用,称 $a_{k,k}^{(k)}$ 为主元素(主元)。

线性方程组的消元过程可以用矩阵来表示:

$$\bar{\boldsymbol{A}}=[\boldsymbol{A}\quad\boldsymbol{b}]$$

$$=\begin{bmatrix}a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)}\\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2,n}^{(1)} & a_{2,n+1}^{(1)}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ a_{n,1}^{(1)} & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} & a_{n,n+1}^{(1)}\end{bmatrix}\xrightarrow{\text{一系列倍加变换}}\begin{bmatrix}a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)}\\ & a_{22}^{(2)} & \cdots & a_{2,n}^{(2)} & a_{2,n+1}^{(2)}\\ & & \ddots & \vdots & \vdots\\ & & & a_{n,n}^{(n)} & a_{n,n+1}^{(n)}\end{bmatrix}$$

消元过程对应于增广矩阵 $\bar{\boldsymbol{A}}=[\boldsymbol{A}\quad \boldsymbol{b}]$ 的一系列倍加变换，最终把系数矩阵 $\boldsymbol{A}$ 化为上三角矩阵。用方程 k 消去方程 i 的变元 x_k 的过程，对应于增广矩阵 $\bar{\boldsymbol{A}}$ 的 1 次倍加变换，即用行乘子 $l_{i,k}$ 乘以 $\bar{\boldsymbol{A}}$ 的第 k 行，加到第 i 行上去。

3.2.2　回代过程

在消元过程中把原线性方程组变形为上三角方程组，然后在回代过程中，自下而上地求解这个上三角方程组。当回代到某一行时，把已求出的变元代入，就变成求解只有 1 个未知变元的线性方程。

下面举例说明回代的具体过程，如表 3.2 所示。

表 3.2　回代的具体过程

实　例	符 号 表 示
$\begin{cases} x+2y-z=2 & ① \\ -7y+4z=-2 & ② \\ -\dfrac{9}{7}z=-\dfrac{27}{7} & ③ \end{cases}$	$\begin{cases} a_{11}^{(1)}x_1+a_{12}^{(1)}x_2+a_{13}^{(1)}x_3=a_{14}^{(1)} & ① \\ a_{22}^{(2)}x_2+a_{23}^{(2)}x_3=a_{24}^{(2)} & ② \\ a_{33}^{(3)}x_3=a_{34}^{(3)} & ③ \end{cases}$
由③得 $z=\dfrac{-\frac{27}{7}}{-\frac{9}{7}}=3$	由③得 $x_3=\dfrac{a_{34}^{(3)}}{a_{33}^{(3)}}$
把 $z=3$ 代入②得 $y=\dfrac{-2-4z}{-7}=2$	把 x_3 代入②得 $x_2=\dfrac{a_{24}^{(2)}-a_{23}^{(2)}x_3}{a_{22}^{(2)}}$
把 $z=3$，$y=2$ 代入①得 $x=2-(2y-z)=1$	把 x_3 和 x_2 代入①得 $x_1=\dfrac{a_{14}^{(1)}-\sum_{j=2}^{3}(a_{1,j}^{(1)}x_j)}{a_{11}^{(1)}}$

若行标 i、列标 j 和上标 k 都从 1 开始，则对 n 阶线性方程组回代的一般公式为

$$x_k=\frac{a_{k,(n+1)}^{(k)}-\sum_{j=k+1}^{n}(a_{k,j}^{(k)}x_j)}{a_{k,k}^{(k)}},\quad k=n,n-1,n-2,\cdots,1$$

可以依次求出 $x_n,x_{n-1},x_{n-2},\cdots,x_1$。

3.2.3　顺序高斯消元法的算法和程序

顺序高斯消元法的几点说明如下。

(1) 当用第 k 行消去第 i 行的变元 x_k 时，$a[i][k]$ 变为 0，并且不参与之后的运算。为了节省存储空间，用 $a[i][k]$ 存放行乘子 $l_{i,k}$。

(2) 与上面的叙述不同，算法和程序中列表元素的下标从 0 开始。

(3) 程序没有对线性方程组无解和主元为零时求不出解的情况进行判断。

算法 3.1 顺序高斯消元法的算法。

输入原方程组的阶数 n

输入原方程组的增广矩阵 a[n][n+1]

for k in range(n−1),循环 1 次消去 1 列　　(消元过程)

- for i in range(k+1,n),循环 1 次更新 1 行
  - 行乘子 a[i][k]/=−a[k][k]
  - for j in range(k+1,n+1)
    - a[i][j]+=a[i][k]*a[k][j]

for k in range(n−1,−1,−1)　　(回代过程)

- s=0
- for j in range(k+1,n)
  - s+=a[k][j]*x[j]
- x[k]=(a[k][n]−s)/a[k][k]

输出原方程组的解向量 x

程序 3.1 顺序高斯消元法对应的程序。

```
def inputdata(a,n):
    print("请输入原方程组的增广矩阵:")
    for i in range(n):
        for j in range(n+1):
            a[i][j]=eval(input('a[{}][{}]='.format(i,j)))
def outputdata(x,n):
    print("原方程组的解为:")
    for i in range(n):
        print('x[{}]={}'.format(i,x[i]),end='  ')
n=int(input("请输入原方程组的阶数:"))
a=[[0]*(n+1) for i in range(n)]
inputdata(a,n)
for k in range(n-1):
    for i in range(k+1,n):
        a[i][k]/=-a[k][k]
        for j in range(k+1,n+1):
            a[i][j]+=a[i][k]*a[k][j]
x=[0]*n
for k in range(n-1,-1,-1):
    s=0
    for j in range(k+1,n):
        s+=a[k][j]*x[j]
```

```
    x[k]=(a[k][n]-s)/a[k][k]
outputdata(x,n)
```

上述程序和算法的时间复杂度为 $O(n^3)$，空间复杂度为 $O(n^2)$。具体地说，顺序高斯消元法的乘除次数 MD$=\frac{n^3}{3}+n^2-\frac{n}{3}\approx\frac{n^3}{3}$。

证明　(1) 消元过程乘除次数$=\frac{2n^3+3n^2-5n}{6}$。

消去 $x_k(k=1,2,\cdots,n-1)$一列时，乘法次数＝方程个数$(n-k)$×含右端项时的系数个数$(n+1-k)$；除法次数＝方程个数$(n-k)$。

又因为

$$\sum_{k=1}^{n-1}(n-k)=\sum_{k=1}^{n-1}k,\sum_{k=1}^{n}k=\frac{n(n+1)}{2},\sum_{k=1}^{n}k^2=\frac{n(n+1)(2n+1)}{6}$$

$$\begin{aligned}\text{所以消元过程乘除次数}&=\sum_{k=1}^{n-1}((n-k)(n+1-k))+\sum_{k=1}^{n-1}(n-k)\\&=\sum_{k=1}^{n-1}(k(k+1))+\sum_{k=1}^{n-1}k=\sum_{k=1}^{n-1}k^2+2\sum_{k=1}^{n-1}k\\&=\frac{(n-1)n(2n-1)}{6}+2\times\frac{n(n-1)}{2}=\frac{2n^3+3n^2-5n}{6}\end{aligned}$$

(2) 回代过程乘除次数$=\frac{n(n+1)}{2}$。

当回代求 $x_k(k=1,2,\cdots,n)$时，乘法次数$=n-k$；除法次数$=1$。

所以回代过程乘除次数$=\sum_{k=1}^{n}(n-k)+n=\sum_{k=1}^{n}(k-1)+n=\sum_{k=1}^{n}k=\frac{n(n+1)}{2}$。

(3) 顺序高斯消元法的乘除次数(MD)＝消元过程乘除次数＋回代过程乘除次数

$$\begin{aligned}&=\frac{2n^3+3n^2-5n}{6}+\frac{n(n+1)}{2}\\&=\frac{n^3}{3}+n^2-\frac{n}{3}\approx\frac{n^3}{3}\end{aligned}$$

证毕。

当方程组的阶数增长时，顺序高斯消元法的运算量增长速度，比克拉默法则求解线性方程组的运算量增长速度要慢得多。如果线性方程组的阶数$=20$，那么克拉默法则需要的乘除次数$\approx10^{21}$，而顺序高斯消元法需要的乘除次数≈3000。

上述程序和算法没有对线性方程组无解和主元为零时求不出解的情况进行判断。由克拉默法则可知，如果线性方程组的系数行列式 $\det(\boldsymbol{A})\neq0$，则线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解，否则线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 可能无解(如 $0x+0y=1$)，也可能有无穷多解(如 $0x+0y=0$)。

性质 3.1　当线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解时，使用顺序高斯消元法能够求出这个解的条件是所有的主元素 $a_{k,k}^{(k)}(k=1,2,\cdots,n)$都不为 0。

若 $a_{k,k}^{(k)}=0$,则行乘子 $l_{i,k}=-\dfrac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}$无意义,无法完成消元。

定义 3.2 若系数矩阵 $\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$,则 $\boldsymbol{A}$ 的 r 阶顺序主子式 $\Delta_r=$

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rr} \end{vmatrix} \quad (r=1,2,\cdots,n)。$$

定理 3.1 对于 n 阶线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$,顺序高斯消元法能够求出解的充要条件是:系数矩阵 $\boldsymbol{A}$ 的所有顺序主子式 $\Delta_i(i=1,2,\cdots,n)$ 均不为 0。

证明 (1) 因为 $\det(\boldsymbol{A})=\Delta_n\neq 0$,所以线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解。

(2) 用数学归纳法证明:$\Delta_i(i=1,2,\cdots,n)$ 均不为 0$\Leftrightarrow a_{k,k}^{(k)}(k=1,2,\cdots,n)$ 都不为 0。

① 当 $r=1$ 时,$\Delta_r=\Delta_1=a_{11}^{(1)}=a_{r,r}^{(r)}$,所以当 $r=1$ 时,$\Delta_i(i=1,2,\cdots,r)$ 均不为 0$\Leftrightarrow a_{k,k}^{(k)}$ $(k=1,2,\cdots,r)$ 都不为 0。

② 假设当 $r=1,2,\cdots,k$ 时,$\Delta_i(i=1,2,\cdots,r)$ 均不为 0$\Leftrightarrow a_{k,k}^{(k)}(k=1,2,\cdots,r)$ 都不为 0。

因为消元过程对应于一系列倍加变换,而倍加变换不改变行列式的值,所以

$$\Delta_k=\begin{vmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{vmatrix} \xlongequal{\text{倍加变换(消元)}} \begin{vmatrix} a_{11}^{(1)} & \cdots & a_{1k}^{(1)} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{kk}^{(k)} \end{vmatrix}$$

并且

$$\Delta_{k+1}=\begin{vmatrix} a_{11} & \cdots & a_{1k} & a_{1,k+1} \\ \vdots & \ddots & \vdots & \vdots \\ a_{k1} & \cdots & a_{kk} & a_{k,k+1} \\ a_{k+1,1} & \cdots & a_{k+1,k} & a_{k+1,k+1} \end{vmatrix} \xlongequal{\text{倍加变换(消元)}} \left|\begin{array}{ccc:c} a_{11}^{(1)} & \cdots & a_{1k}^{(1)} & a_{1,k+1}^{(1)} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{kk}^{(k)} & a_{k,k+1}^{(k)} \\ \hdashline 0 & \cdots & 0 & a_{k+1,k+1}^{(k+1)} \end{array}\right|$$

$$=\begin{vmatrix} a_{11}^{(1)} & \cdots & a_{1k}^{(1)} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{kk}^{(k)} \end{vmatrix} \cdot a_{k+1,k+1}^{(k+1)} - \begin{vmatrix} a_{1,k+1}^{(1)} \\ \vdots \\ a_{k,k+1}^{(k)} \end{vmatrix} \cdot 0=\Delta_k \cdot a_{k+1,k+1}^{(k+1)}$$

又因为 $\Delta_k\neq 0$,所以 Δ_{k+1} 不为 0$\Leftrightarrow a_{k+1,k+1}^{(k+1)}$ 不为 0,那么 $r=1,2,\cdots,k,k+1$ 时,$\Delta_i(i=1,2,\cdots,r)$ 均不为 0$\Leftrightarrow a_{k,k}^{(k)}(k=1,2,\cdots,r)$ 都不为 0。

③ 由①、②可得,$\Delta_i(i=1,2,\cdots,n)$ 均不为 0$\Leftrightarrow a_{k,k}^{(k)}(k=1,2,\cdots,n)$ 都不为 0。

(3) 按性质 3.1,对于 n 阶线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$,顺序高斯消元法能够求出解的充要条件是:系数矩阵 $\boldsymbol{A}$ 的所有顺序主子式 $\Delta_i(i=1,2,\cdots,n)$ 均不为 0。

证毕。

3.3　列主元高斯消元法

3.3.1　列主元高斯消元法的主要思想

顺序高斯消元法的缺点如下。

(1) 当线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的系数行列式 $\det(\boldsymbol{A})\neq 0$，且存在一个主元素 $a_{k,k}^{(k)}=0$ 时，线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解，但顺序高斯消元法不能求出解。

(2) 小主元(主元素的绝对值远小于其他元素的绝对值)时，能够求出解，但求出的解误差很大。

交换增广矩阵的 2 行(2 个方程的位置互换)或交换 2 列(2 个变元的位置互换)，得到的方程组与原方程组同解。通过交换行、列来避免主元为 0 和小主元，然后再消元求解的方法，称为选主元高斯消元法。列主元高斯消元法是一种常见的选主元高斯消元法，与顺序高斯消元法相比，它的改进之处是：在用主元素 $a_{k,k}^{(k)}$ 消去第 $k+1$ 行至第 n 行的变元 x_k 之前，从第 k 行至第 n 行所有 x_k 的系数(即 $a_{k,k}^{(k)},a_{k+1,k}^{(k)},\cdots,a_{n,k}^{(k)}$)中，寻找绝对值最大者 $a_{\max\_i,k}^{(k)}$，若 $\max\_i\neq k$，则交换第 k 行和第 $\max\_i$ 行，使最大的系数 $a_{\max\_i,k}^{(k)}$ 成为主元。

定理 3.2　当线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解时，列主元高斯消元法必能求出解。

证明　用反证法。假设线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解，但列主元高斯消元法求不出解。这就是说，在消元过程中，当消去某一个变元(设为 x_k)时，$a_{k,k}^{(k)},a_{k+1,k}^{(k)},\cdots,a_{n,k}^{(k)}$ 全部为 0，即使在其中取绝对值最大者为主元，主元仍然为 0。

那么系数行列式

$$\det(\boldsymbol{A})=\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \xlongequal{\text{消去 } x_1\sim x_{k-1}} \left|\begin{array}{ccc:ccc} a_{11}^{(1)} & \cdots & a_{1,k-1}^{(1)} & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & a_{k-1,k-1}^{(k-1)} & a_{k-1,k}^{(k-1)} & \cdots & a_{k-1,n}^{(k-1)} \\ \hdashline 0 & \cdots & 0 & a_{k,k}^{(k)} & \cdots & a_{k,n}^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{n,k}^{(k)} & \cdots & a_{n,n}^{(k)} \end{array}\right|$$

$$=\begin{vmatrix} a_{11}^{(1)} & \cdots & a_{1,k-1}^{(1)} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{k-1,k-1}^{(k-1)} \end{vmatrix}\cdot\begin{vmatrix} a_{k,k}^{(k)} & \cdots & a_{k,n}^{(k)} \\ \vdots & \vdots & \vdots \\ a_{n,k}^{(k)} & \cdots & a_{n,n}^{(k)} \end{vmatrix}-\begin{vmatrix} a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ \vdots & \vdots & \vdots \\ a_{k-1,k}^{(k-1)} & \cdots & a_{k-1,n}^{(k-1)} \end{vmatrix}\cdot 0$$

$$=\begin{vmatrix} a_{11}^{(1)} & \cdots & a_{1,k-1}^{(1)} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{k-1,k-1}^{(k-1)} \end{vmatrix}\cdot\begin{vmatrix} a_{k,k}^{(k)} & \cdots & a_{k,n}^{(k)} \\ \vdots & \vdots & \vdots \\ a_{n,k}^{(k)} & \cdots & a_{n,n}^{(k)} \end{vmatrix}$$

因为 $a_{k,k}^{(k)},a_{k+1,k}^{(k)},\cdots,a_{n,k}^{(k)}$ 全部为 0，所以 $\begin{vmatrix} a_{k,k}^{(k)} & \cdots & a_{k,n}^{(k)} \\ \vdots & \vdots & \vdots \\ a_{n,k}^{(k)} & \cdots & a_{n,n}^{(k)} \end{vmatrix}=0$。

由上可得,系数行列式 $\det(\boldsymbol{A})=\begin{vmatrix} a_{11}^{(1)} & \cdots & a_{1,k-1}^{(1)} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{k-1,k-1}^{(k-1)} \end{vmatrix}\cdot 0=0$。

但是线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解⇔系数行列式 $\det(\boldsymbol{A})\neq 0$,这与 $\det(\boldsymbol{A})=0$ 矛盾,所以假设错误,也就是说,当线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解时,列主元高斯消元法必能求出解。

证毕。

列主元高斯消元法能够避免小主元时舍入误差被放大的情况,提高解的精度。

例 3.1 求解线性方程组$\begin{cases} 0.003x_1+3x_2=2.001 & ① \\ x_1+x_2=1 & ② \end{cases}$,设存储精度为 4 位有效数字。(精确解:$x_1=\dfrac{1}{3}$,$x_2=\dfrac{2}{3}$,比较顺序高斯消元法和列主元高斯消元法的误差大小)

解法 1 顺序高斯消元法。

$①\times\left(-\dfrac{1}{0.003}\right)+②\rightarrow②$,得 $-999x_2=-666$,那么 $x_2=\dfrac{-666}{-999}\approx 0.6667$(因为存储精度为 4 位有效数字,所以 x_2 舍入误差的绝对值$\approx 3.3\times 10^{-5}$)。

把 $x_2\approx 0.6667$ 代入①,得 $x_1=(2.001-3\times 0.6667)/0.003=0.3$(因为除以小主元 0.003,所以 x_1 舍入误差的绝对值$\approx 3.3\times 10^{-2}$,误差被放大 1000 倍)。

解法 2 列主元高斯消元法。

选列主元,得到同解线性方程组$\begin{cases} x_1+x_2=1 & ① \\ 0.003x_1+3x_2=2.001 & ② \end{cases}$

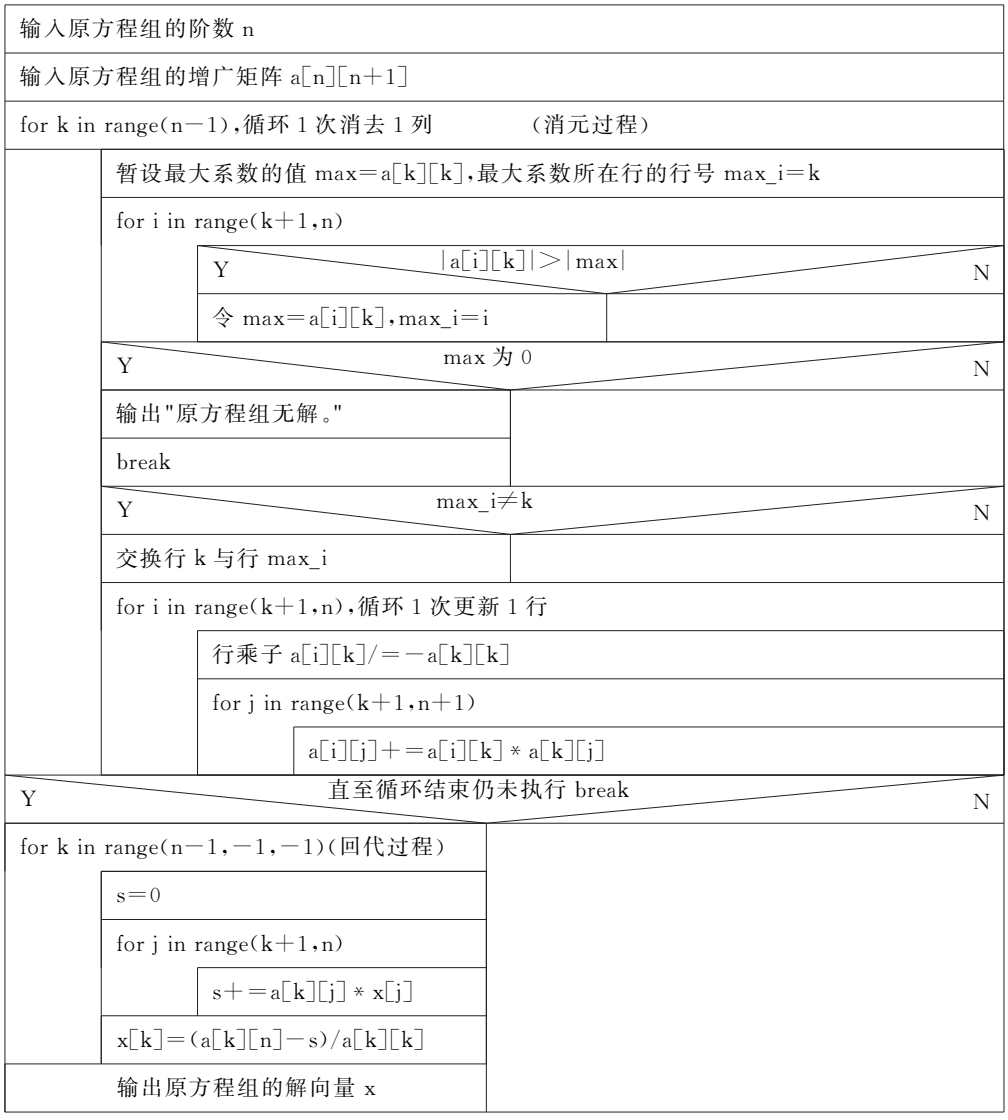
$①\times\left(-\dfrac{0.003}{1}\right)+②\rightarrow②$,得 $2.997x_2=1.998$,那么 $x_2=\dfrac{1.998}{2.997}\approx 0.6667$(因为存储精度为 4 位有效数字,所以 x_2 舍入误差的绝对值$\approx 3.3\times 10^{-5}$)。

把 $x_2\approx 0.6667$ 代入①,得 $x_1\approx 1-0.6667=0.3333$(因为避免了小主元,所以 x_1 的舍入误差没有被放大)。

3.3.2 列主元高斯消元法的算法和程序

用列主元高斯消元法求解 n 阶线性方程组,在消元过程中,使用主元 $a_{k,k}^{(k)}$ 之前,从第 k 行至第 n 行 x_k 的系数中,寻找最大者 $a_{\max\_i,k}^{(k)}$。若 $a_{\max\_i,k}^{(k)}$ 为 0,则方程组无解,否则交换第 k 行和第 $\max\_i$ 行。

算法 3.2 列主元高斯消元法的算法。



程序 3.2　列主元高斯消元法对应的程序。

```
def inputdata(a,n):
    print("请输入原方程组的增广矩阵: ")
    for i in range(n):
        for j in range(n+1):
            a[i][j]=eval(input('a[{}][{}]='.format(i,j)))
def outputdata(x,n):
    print("原方程组的解为: ")
    for i in range(n):
        print('x[{}]={}'.format(i,x[i]),end='  ')
n=int(input("请输入原方程组的阶数: "))
a=[[0]*(n+1) for i in range(n)]
inputdata(a,n)
for k in range(n-1):
```

```
            max,max_i=a[k][k],k
            for i in range(k+1,n):
                if abs(a[i][k])>abs(max):
                    max,max_i=a[i][k],i
            if max==0:
                print("原方程组无解。")
                break
            if max_i! =k:
                a[k],a[max_i]=a[max_i],a[k]
            for i in range(k+1,n):
                a[i][k]/=-a[k][k]
                for j in range(k+1,n+1):
                    a[i][j]+=a[i][k]*a[k][j]
    else:
        x=[0]*n
        for k in range(n-1,-1,-1):
            s=0
            for j in range(k+1,n):
                s+=a[k][j]*x[j]
            x[k]=(a[k][n]-s)/a[k][k]
        outputdata(x,n)
```

3.4 全主元高斯消元法

3.4.1 全主元高斯消元法的主要思想

全主元高斯消元法是另一种选主元高斯消元法。全主元高斯消元法与顺序高斯消元法和列主元高斯消元法相比,它的改进之处是:在用主元素 $a_{k,k}^{(k)}$ 消去第 $k+1$ 行至第 n 行的变元 x_k 之前,从 $a_{k,k}^{(k)}$ 右下方的矩形区域所有 x_k 的系数中,寻找最大者 $a_{\max\_i,\max\_j}^{(k)}$(即 $a_{\max\_i,\max\_j}^{(k)}=\max\limits_{k\leqslant i,j\leqslant n}(|a_{i,j}^{(k)}|)$),若 $\max\_i\neq k$,则交换第 k 行和第 $\max\_i$ 行;若 $\max\_j\neq k$,则交换第 k 列和第 $\max\_j$ 列,使最大的系数 $a_{\max\_i,\max\_j}^{(k)}$ 成为主元。

在程序中不存储变元的名称,确定变元的依据是变元的位置,即变元位于哪一列。因此当交换两列时,需要记录交换后各变元的位置(分别位于哪一列),求解完毕,再按照原方程组各变元的次序输出各变元的值。

定理 3.3 当线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有唯一解时,全主元高斯消元法必能求出解。

证明略。

与列主元高斯消元法相比,全主元高斯消元法也能够避免小主元时舍入误差被放大的情况,而且解的精度一般比列主元高斯消元法高。

例 3.2 求解线性方程组 $\begin{cases}x_1+x_2=2 & ①\\ 2x_1+10^5x_2=10^5 & ②\end{cases}$,设存储精度为 4 位数字。

($x_1\approx-1.0000200004$,$x_2\approx0.9999799996$,比较列主元高斯消元法和全主元高斯消元法的误差大小)。

解法 1 列主元高斯消元法。

选列主元,得到同解线性方程组:

$$\begin{cases} 2x_1+10^5x_2=10^5 & ① \\ x_1+\quad x_2=2 & ② \end{cases}$$

$①\times\left(-\frac{1}{2}\right)+②\to②$，得 $x_2=\dfrac{\left(\frac{10^5}{-2}\right)+2}{\left(\frac{10^5}{-2}\right)+1}\approx 1$（因为存储精度为 4 位有效数字，所以 x_2 舍入误差的绝对值$\approx 2.0\times 10^{-5}$）

把 $x_2=1$ 代入①得：$2x_1+10^5\times 1=10^5$，所以 $x_1=0$（因为 x_2 乘以系数 10^5，所以 x_1 舍入误差的绝对值≈ 1，误差被放大 10^5 倍）。

解法 2　全主元高斯消元法。

选全主元，得到同解线性方程组：

$$\begin{cases} 10^5x_2+2x_1=10^5 & ① \\ x_2+x_1=2 & ② \end{cases}$$

$①\times\left(-\frac{1}{10^5}\right)+②\to②$，得 $x_1=\dfrac{\left(\frac{10^5}{-10^5}\right)+2}{\left(\frac{2}{-10^5}\right)+1}\approx 1$（因为存储精度为 4 位有效数字，所以 x_1 舍入误差的绝对值$\approx 2.0\times 10^{-5}$）

把 $x_1=1$ 代入①得：$10^5x_2+2\times 1=10^5$，所以 $x_2=\dfrac{10^5-2}{10^5}\approx 1$（因为回代时避免了乘以大系数，所以 x_1 的舍入误差没有被放大）。

与顺序高斯消元法相比，列主元高斯消元法增加的运算量主要如下：

(1) 为寻找最大系数而增加的比较次数 $\approx\sum\limits_{k=1}^{n-1}(n-k)=\sum\limits_{k=1}^{n-1}k=\dfrac{(n-1)n}{2}\approx\dfrac{n^2}{2}$。

(2) 行交换增加的系数交换次数 $\approx\sum\limits_{k=1}^{n-1}(n-k+2)=\sum\limits_{k=1}^{n-1}k+2(n-1)=\dfrac{(n-1)n}{2}+2n-2\approx\dfrac{n^2}{2}$。

与顺序高斯消元法相比，全主元高斯消元法增加的运算量主要如下。

(1) 为寻找最大系数而增加的比较次数 $\approx\sum\limits_{k=1}^{n-1}(n-k+1)^2=\sum\limits_{k=1}^{n-1}(k+1)^2=\dfrac{n(n+1)(2n+1)}{6}-1\approx\dfrac{n^3}{3}$。

(2) 行交换增加的系数交换次数 $\approx\sum\limits_{k=1}^{n-1}(n-k+2)=\sum\limits_{k=1}^{n-1}k+2(n-1)=\dfrac{(n-1)n}{2}+2n-2\approx\dfrac{n^2}{2}$。

(3) 列交换增加的系数交换次数 $\approx\sum\limits_{k=1}^{n-1}n=n(n-1)\approx n^2$。

总之，与全主元高斯消元法增加的运算量相比，列主元高斯消元法增加的运算量可以忽略。全主元高斯消元法的程序结构比列主元高斯消元法要复杂得多。只要原方程组有

解,这两种方法都能求出解。一般而言,全主元高斯消元法比列主元高斯消元法精度高。

3.4.2 全主元高斯消元法的算法和程序

在程序中用列表 x_j 记录对增广矩阵做一系列交换的过程中各变元的位置(变元位于哪一列)。增广矩阵的列 j(列序号从 0 开始)所对应变元的下标 $=x_j[j]$。也就是说,当 $x_j[j]=k$ 时,解向量中的元素 $x[j]$存放的是变元 x_k。程序开始时,原方程组各变元的次序表示为 $x_j[j]=j$;增广矩阵的列 j 移动到列 max_j,表示为赋值 $x_j[\text{max\_}j]=x_j[j]$;交换列 j 和列 max_j,表示为交换 $x_j[j]$和 $x_j[\text{max_}j]$的值。求解完毕,按照原方程组各变元的次序输出各变元的值。

算法 3.3 全主元高斯消元法的算法。

输入原方程组的阶数 n

输入原方程组的增广矩阵 a[n][n+1],记录各变元的初始次序 xj[j]=j,(j=0,1,2,…,n−1)

for k in range(n−1),循环 1 次消去 1 列　　(消元过程)

- 暂设最大系数的值 max=a[k][k],max 的行号 max_i=k,列号 max_j=k
- for i in range(k,n)(找绝对值最大的系数及其位置)
  - for j in range(k,n)
    - |a[i][j]|>|max|?
      - Y: 令 max=a[i][j],max_i=i,max_j=j
      - N:
- max 为 0?
  - Y: 输出"原方程组无解。"; break
  - N:
- max_i≠k?
  - Y: 交换行 k 与行 max_i
  - N:
- max_j≠k?
  - Y: 交换列 k 与列 max_j; 交换 xj[k]与 xj[max_j]
  - N:
- 消元模块的 2 重循环,同列主元高斯消元法,此处略

直至循环结束仍未执行 break?

- Y: 原方程组无解
- N: 回代模块的 2 重循环,同列主元高斯消元法,此处略
  - for k in range(n)(按原方程组各变元的次序输出各变元的值)
    - 在列表 xj 中,元素值为 k 的元素下标 i=xj.index(k)
    - 输出原方程组变元 xk 的值 x[i]

程序 3.3　全主元高斯消元法对应的程序。

```
def inputdata(a,n):
    print("请输入原方程组的增广矩阵：")
    for i in range(n):
        for j in range(n+1):
            a[i][j]=eval(input('a[{}][{}]='.format(i,j)))
def outputdata(x,n,xj):
    print("原方程组的解为：")
    for k in range(n):
        i=xj.index(k)
        print('x[{}]={}'.format(k,x[i]),end='  ')
n=int(input("请输入原方程组的阶数："))
a=[[0]*(n+1) for i in range(n)]
inputdata(a,n)
xj=[i for i in range(n)]
for k in range(n-1):        #消元过程开始
    max=a[k][k]
    max_i=max_j=k
    for i in range(k,n):
        for j in range(k,n):
            if abs(a[i][j])>abs(max):
                max,max_i,max_j=a[i][j],i,j
    if max==0:
        print("原方程组无解。")
        break
    if max_i!=k:
        a[k],a[max_i]=a[max_i],a[k]
    if max_j!=k:
        for i in range(k,n):
            a[i][k],a[i][max_j]=a[i][max_j],a[i][k]
        xj[k],xj[max_j]=xj[max_j],xj[k]
    for i in range(k+1,n):  #真正的消元,循环 1 次更新 1 行
        a[i][k]/=-a[k][k]
        for j in range(k+1,n+1):
            a[i][j]+=a[i][k]*a[k][j]
else:
    x=[0]*n       #回代过程开始
    for k in range(n-1,-1,-1):
        s=0
        for j in range(k+1,n):
            s+=a[k][j]*x[j]
        x[k]=(a[k][n]-s)/a[k][k]
    outputdata(x,n,xj)
```

3.5 高斯约当消元法

3.5.1 高斯约当消元法的主要思想

与顺序高斯消元法的求解过程相比,高斯约当消元法求解过程的特点如下。

(1) 在消去变元 x_k 所在的那一列时,不仅把主元之下的 x_k 消去,也把主元之上的 x_k 消去,也就是把系数矩阵 $\boldsymbol{A}$ 除主对角线之外的元素都化为 0。

(2) 在消去变元 x_k 所在的那一列时,先把主元 $a_{k,k}^{(k)}$ 化为 1(让方程 k 所有系数都除以 $a_{k,k}^{(k)}$),再用它去消其他方程的变元 x_k,也就是把系数矩阵 $\boldsymbol{A}$ 主对角线上的元素都化为 1。

消元完毕,原线性方程组同解变换成如下形式:

$$\begin{cases} x_1 & & & = a_{1,n+1}^{(1)} \\ & x_2 & & = a_{2,n+1}^{(2)} \\ & & \ddots & \vdots \quad \vdots \\ & & x_n & = a_{n,n+1}^{(n)} \end{cases}$$

这时系数矩阵 $\boldsymbol{A}$ 被化为单位矩阵,右端向量就是解向量,不再需要回代过程。因此高斯约当消元法又称为无回代过程消元法。

例 3.3 用高斯约当消元法求解线性方程组 $\begin{cases} 2x_1 - x_2 - 3x_3 = -2 & ① \\ 2x_1 - 3x_2 - 2x_3 = -3 & ② \\ -x_1 + x_2 + x_3 = 1 & ③ \end{cases}$。

解 ①×0.5,得 $\begin{cases} x_1 - 0.5x_2 - 1.5x_3 = -1 & ① \\ 2x_1 - 3x_2 - 2x_3 = -3 & ② \\ -x_1 + x_2 + x_3 = 1 & ③ \end{cases}$

①×(−2)+②→②,①+③→③,得 $\begin{cases} x_1 - 0.5x_2 - 1.5x_3 = -1 & ① \\ -2x_2 + x_3 = -1 & ② \\ 0.5x_2 - 0.5x_3 = 0 & ③ \end{cases}$

②×(−0.5),得 $\begin{cases} x_1 - 0.5x_2 - 1.5x_3 = -1 & ① \\ x_2 - 0.5x_3 = 0.5 & ② \\ 0.5x_2 - 0.5x_3 = 0 & ③ \end{cases}$

②×0.5+①→①,②×(−0.5)+③→③,得 $\begin{cases} x_1 - 1.75x_3 = -0.75 & ① \\ x_2 - 0.5x_3 = 0.5 & ② \\ -0.25x_3 = -0.25 & ③ \end{cases}$

③×(−4),得 $\begin{cases} x_1 - 1.75x_3 = -0.75 & ① \\ x_2 - 0.5x_3 = 0.5 & ② \\ x_3 = 1 & ③ \end{cases}$

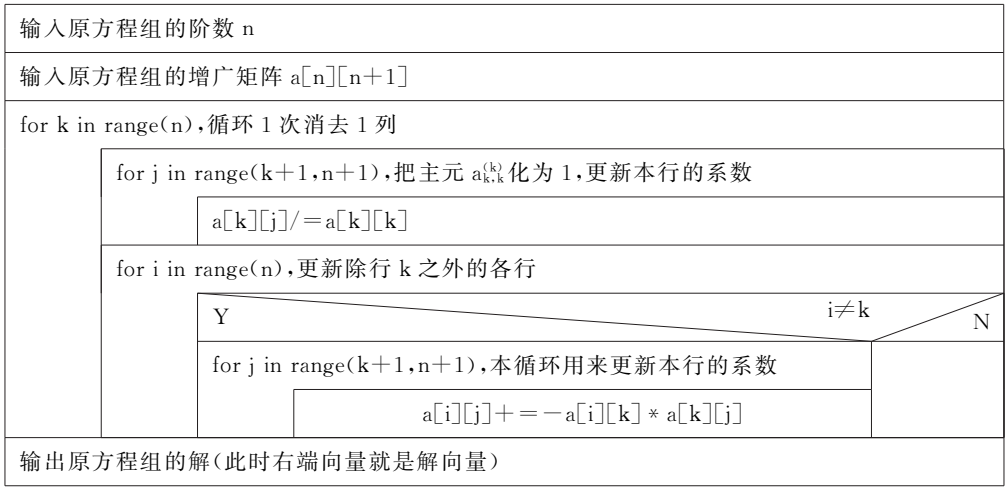
③×1.75+①→①,③×0.5+②→②,得 $\begin{cases} x_1 = 1 & ① \\ x_2 = 1 & ② \\ x_3 = 1 & ③ \end{cases}$

所以原方程组的解为：$x_1=1, x_2=1, x_3=1$。

3.5.2　高斯约当消元法的算法和程序

与顺序高斯消元法相同，高斯约当消元法也存在主元为 0 时求不出解，小主元时误差较大等问题。解决方法与顺序高斯消元法类似，可以采用交换列的方法，保证在有解时一定能求出解。下面的程序没有对线性方程组无解和主元为零时求不出解的情况进行判断。

算法 3.4　高斯约当消元法的算法。



程序 3.4　高斯约当消元法对应的程序。

```
def inputdata(a,n):
    print("请输入原方程组的增广矩阵: ")
    for i in range(n):
        for j in range(n+1):
            a[i][j]=eval(input('a[{}][{}]='.format(i,j)))
def outputdata(a,n):
    print("原方程组的解为: ")
    for i in range(n):
        print('x[{}]={}'.format(i,a[i][n]),end='  ')
n=int(input("请输入原方程组的阶数: "))
a=[[0]*(n+1) for i in range(n)]
inputdata(a,n)
for k in range(n):                          #循环 1 次消去 1 列
    for j in range(k+1,n+1):                #把主元 a(k)k,k 化为 1,更新本行的系数
        a[k][j]/=a[k][k]
    for i in range(n):                      #更新除行 k 之外的各行
        if i!=k:
            for j in range(k+1,n+1):        #本循环用来更新本行的系数
```

```
                a[i][j]+=-a[i][k]*a[k][j]
    outputdata(a,n)
```

3.5.3 一次求解出多个线性方程组

行乘子 $l_{i,k}$ 的计算公式 $l_{i,k}=-\dfrac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}$ 中，不出现右端向量 $\boldsymbol{b}$。当用高斯约当消元法求解多个系数矩阵 $\boldsymbol{A}$ 相同，而右端向量 $\boldsymbol{b}$ 不同的线性方程组时，消元过程对应的一系列倍加变换是相同的。也就是说，以什么次序，哪一行(方程)乘以什么因子(行乘子)，加到哪一行(方程)上，不受右端向量 $\boldsymbol{b}$ 的影响。因此，可以把这些方程组的右端向量合并到一个增广矩阵中，每多一个线性方程组，增广矩阵就多增加一列。当对增广矩阵做倍加变换，把系数矩阵 $\boldsymbol{A}$ 化为单位矩阵时，右端向量就被化为对应方程组的解。这样，原来求解一个线性方程组的倍加变换，可以把这些线性方程组全部求出，减少了总的运算量。

例 3.4 用高斯约当消元法，一次消元求解下列多个线性方程组：

$$①\begin{cases}2x+y-z=2\\-x+3z=2,\\-2x+y+z=0\end{cases}\quad②\begin{cases}2x+y-z=1\\-x+3z=8,\\-2x+y+z=3\end{cases}\quad③\begin{cases}2x+y-z=7\\-x+3z=0\\-2x+y+z=-3\end{cases}$$

解 构造方程组①、②、③的增广矩阵，得

$$\overline{\boldsymbol{A}}=[\boldsymbol{A}\quad\boldsymbol{b}]=\left[\begin{array}{ccc:ccc}2&1&-1&2&1&7\\-1&0&3&2&8&0\\-2&1&1&0&3&-3\end{array}\right]\xrightarrow{r_1/2}\left[\begin{array}{ccc:ccc}1&0.5&-0.5&1&0.5&3.5\\-1&0&3&2&8&0\\-2&1&1&0&3&-3\end{array}\right]$$

$$\xrightarrow[r_1\times 2+r_3]{r_1+r_2}\left[\begin{array}{ccc:ccc}1&0.5&-0.5&1&0.5&3.5\\0&0.5&2.5&3&8.5&3.5\\0&2&0&2&4&4\end{array}\right]\xrightarrow{r_2\times 2}\left[\begin{array}{ccc:ccc}1&0.5&-0.5&1&0.5&3.5\\0&1&5&6&17&7\\0&2&0&2&4&4\end{array}\right]$$

$$\xrightarrow[r_2\times(-2)+r_3]{r_2/(-2)+r_1}\left[\begin{array}{ccc:ccc}1&0&-3&-2&-8&0\\0&1&5&6&17&7\\0&0&-10&-10&-30&-10\end{array}\right]$$

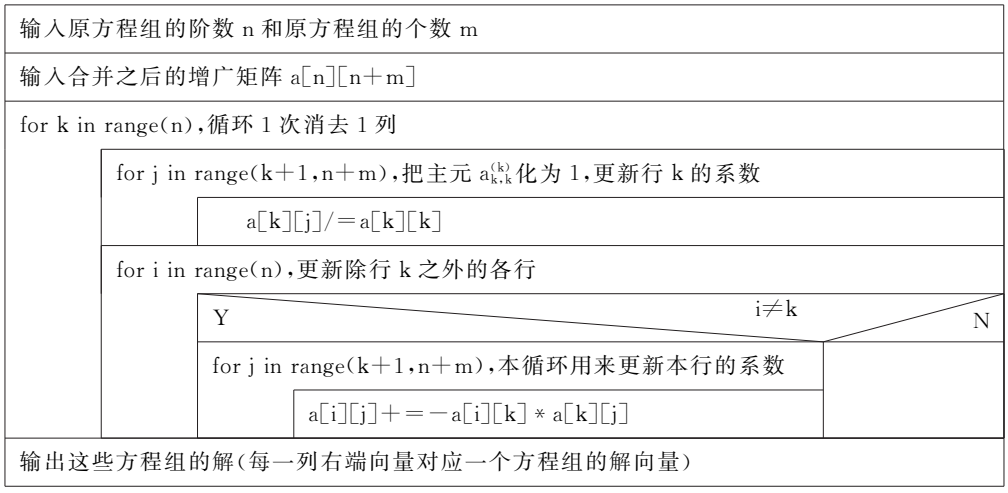
$$\xrightarrow{r_3/(-10)}\left[\begin{array}{ccc:ccc}1&0&-3&-2&-8&0\\0&1&5&6&17&7\\0&0&1&1&3&1\end{array}\right]$$

$$\xrightarrow[r_3\times(-5)+r_2]{r_3\times 3+r_1}\left[\begin{array}{ccc:ccc}1&0&0&1&1&3\\0&1&0&1&2&2\\0&0&1&1&3&1\end{array}\right]$$

所以方程组①的解为 $\begin{cases}x=1\\y=1\\z=1\end{cases}$，方程组②的解为 $\begin{cases}x=1\\y=2\\z=3\end{cases}$，方程组③的解为 $\begin{cases}x=3\\y=2\\z=1\end{cases}$。

3.5.4 一次求解多个线性方程组的算法和程序

算法 3.5 用高斯约当消元法一次消元求解多个线性方程组的算法。



程序 3.5　用高斯约当消元法一次消元求解多个线性方程组对应的程序。

```
def inputdata(a,n,m):
    print("请输入合并之后的增广矩阵({}行、{}列): ".format(n,n+m))
    for i in range(n):
        for j in range(n+m):
            a[i][j]=eval(input('a[{}][{}]='.format(i,j)))
def outputdata(a,n,m):
    for k in range(m):
        print("\n方程组{}的解为: ".format(k))
        for i in range(n):
            print('x{}[{}]={}'.format(k,i,a[i][n+k]),end=' ')
n=int(input("请输入原方程组的阶数: "))
m=int(input("请输入原方程组的个数: "))
a=[[0]*(n+m) for i in range(n)]
inputdata(a,n,m)
for k in range(n):  #循环 1 次消去 1 列
    for j in range(k+1,n+m):  #把主元 a(k)k,k 化为 1,更新行 k 的系数
        a[k][j]/=a[k][k]
    for i in range(n):  #更新除行 k 之外的各行
        if i!=k:
            for j in range(k+1,n+m):  #本循环用来更新本行的系数
                a[i][j]+=-a[i][k]*a[k][j]
outputdata(a,n,m)
```

3.6　消元形式的追赶法

3.6.1　消元形式的追赶法的主要思想

带状对角矩阵是一种常见的高阶稀疏矩阵。3 对角矩阵是带状对角矩阵的一种。在

做三次样条插值、求解微分方程等问题时,经常需要解 3 对角方程组。

定义 3.3 如果在方阵 $\boldsymbol{A}$ 中,除主对角线和两条次对角线之外的元素都为 0,那么 $\boldsymbol{A}$ 是 3 对角矩阵,即 $\boldsymbol{A}$ 可以表示为下面的形式:

$$\boldsymbol{A}=\begin{bmatrix} a_{1,0} & a_{1,1} & & & \\ a_{2,-1} & a_{2,0} & a_{2,1} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1,-1} & a_{n-1,0} & a_{n-1,1} \\ & & & a_{n,-1} & a_{n,0} \end{bmatrix}$$

其中,$a_{i,j}$ 的左下标 i 为行号,右下标 j 为 0 表示 $a_{i,j}$ 位于主对角线,j 为 1 表示 $a_{i,j}$ 位于右上次对角线,j 为 -1 表示 $a_{i,j}$ 位于左下次对角线。系数矩阵 $\boldsymbol{A}$ 中各个系数下标之间的关系是,$a_{i,j}$ 的下方为 $a_{i+1,j-1}$,$a_{i,j}$ 的右方为 $a_{i,j+1}$。

定义 3.4 如果线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的系数矩阵 $\boldsymbol{A}$ 是 3 对角矩阵,那么 $\boldsymbol{Ax}=\boldsymbol{b}$ 是 3 对角方程组。

定义 3.5 如果在方阵 $\boldsymbol{A}$ 中,除主对角线、左下 lw 条次对角线和右上 uw 条次对角线之外的元素都为 0,那么 $\boldsymbol{A}$ 是带状对角矩阵,称 lw 为下带宽,uw 为上带宽,$bw=uw+lw+1$ 为带宽,即 $\boldsymbol{A}$ 可以表示为下面的形式:

$$\boldsymbol{A}=\begin{bmatrix} a_{1,1} & \cdots & a_{1,uw+1} & & \\ \vdots & \ddots & & \ddots & \\ a_{lw+1,1} & & \ddots & & a_{n-uw,n} \\ & \ddots & & \ddots & \vdots \\ & & a_{n,n-lw} & \cdots & a_{n,n} \end{bmatrix}$$

其中,左下标为行号,右下标为列号。

定义 3.6 如果线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的系数矩阵 $\boldsymbol{A}$ 是带状对角矩阵,那么 $\boldsymbol{Ax}=\boldsymbol{b}$ 是带状对角方程组。

追赶法用于求解 3 对角方程组。在求解 3 对角方程组时,原来的求解过程被大大简化。要求解的 3 对角方程组可以表示为

$$\begin{cases} a_{1,0}x_1+ & a_{1,1}x_2 & & & =a_{1,2} \\ a_{2,-1}x_1+ & a_{2,0}x_2 & +a_{2,1}x_3 & & =a_{2,2} \\ \quad\ddots & \quad\ddots & \quad\ddots & & \vdots \\ & a_{n-1,-1}x_{n-2}+ & a_{n-1,0}x_{n-1}+ & a_{n-1,1}x_n & =a_{n-1,2} \\ & & a_{n,-1}x_{n-1}+ & a_{n,0}x_n & =a_{n,2} \end{cases}$$

求解 3 对角方程组,不需要存储系数矩阵 $\boldsymbol{A}$ 中除 3 条对角线之外的零元素,只需要存储 3 条对角线、右端向量和解向量,使存储空间需求从 $O(n^2)$ 降为 $O(n)$。

这里按照顺序高斯消元法的求解思路求解这个 3 对角方程组。在消元过程中,当消去 x_i 一列时,主元素为 $a_{i,0}$。$a_{i,0}$ 下方只有一个系数 $a_{i+1,-1}$ 非 0,因此当消去 x_i 一列时,只需要更新第 $i+1$ 行中变元的系数。这一行中,也只有 $a_{i+1,0}$ 和 $a_{i+1,1}$ 需要更新。