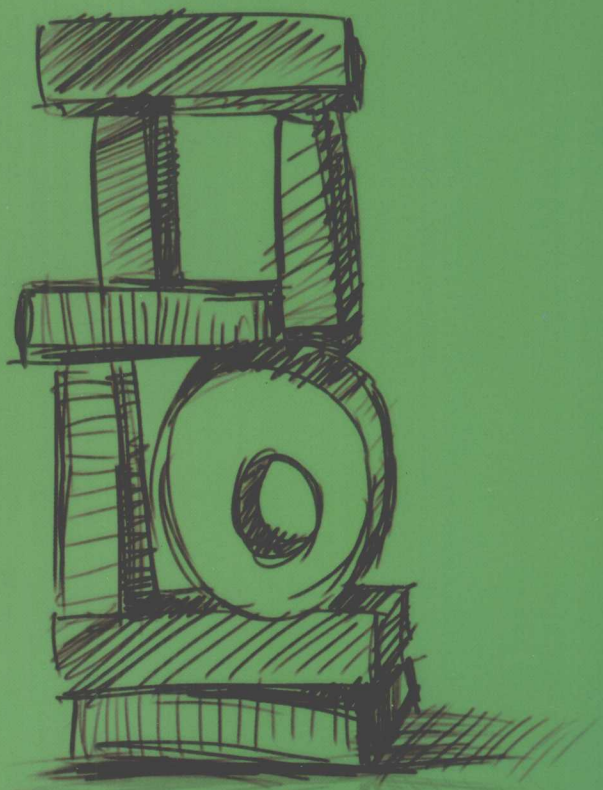
嵌入式系统软件设计基础丛书



周航慈 著

嵌入式系统软件设计中的常用算法

北京航空航天大学出版社

嵌入式系统软件设计基础丛书

嵌入式系统软件设计中的常用算法

周航慈　著

北京航空航天大学出版社

内 容 简 介

本书根据嵌入式系统软件设计需要的常用算法知识编写而成。基本内容有:线性方程组求解、代数插值和曲线拟合、数值积分、能谱处理、数字滤波、数理统计、自动控制、数据排序、数据压缩和检错纠错等常用算法。从嵌入式系统的实际应用出发,用通俗易懂的语言代替枯燥难懂的数学推导,使读者能在比较轻松的条件下学到最基本的常用算法,并为继续学习其他算法打下基础。

本书可作为电子技术人员自学常用算法的教材,也可作为高等院校电子技术类专业本科生、研究生的教学参考书。

图书在版编目(CIP)数据

嵌入式系统软件设计中的常用算法/周航慈著. —北京:
北京航空航天大学出版社,2010.1
ISBN 978-7-81124-943-9

Ⅰ.嵌… Ⅱ.周… Ⅲ.微型计算机—软件设计—算法
Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2009)第 189989 号

嵌入式系统软件设计中的常用算法
周航慈 著
责任编辑 董云凤 张金伟
*
北京航空航天大学出版社出版发行
北京市海淀区学院路 37 号(100191) 发行部电话:010-82317024 传真:010-82328026
http://www.buaapress.com.cn E-mail:bhpress@263.net
北京时代华都印刷有限公司印装 各地书店经销
*
开本:787 mm×960 mm 1/16 印张:12.5 字数:280 千字
2010 年 1 月第 1 版 2010 年 1 月第 1 次印刷 印数:5 000 册
ISBN 978-7-81124-943-9 定价:24.00 元

前 言

嵌入式系统在各行各业的应用越来越广，我国从事嵌入式系统开发的人员也越来越多，从国内主要的几种电子杂志上可以看出，有关嵌入式系统应用的文章也越来越多。

在开发一种嵌入式系统产品时，主要是做两方面的设计：硬件设计和软件设计。在硬件设计方面，各个半导体公司竞相推出各种高性能、低功耗、低成本的 CPU 和外围芯片，这使我们在进行硬件设计时可以很快地得到最先进的芯片。在这种情况下，硬件设计的外部条件越来越好，集成度越来越高，在实现相同功能的情况下线路越来越简化。在软件设计方面，虽然开发工具和程序设计语言也在不断提高，但技术人员本身的软件素质无疑起决定作用。因此，软件设计水平在嵌入式系统产品开发的过程中占有重要的地位，直接影响到产品的功能和竞争能力。

我国目前绝大多数从事嵌入式系统开发的技术人员基本上不是计算机专业毕业的，有的可能还没有上过大学，他们未接受过系统的软件基础理论教育，软件设计水平仍不太高。在软件开发过程中，他们只是不自觉地采用了一些规律性的设计方法，或者模仿别人的程序设计方法，而有更多成熟的基本方法没有掌握，开发出来的软件水平不高，致使产品的功能和可靠性受到一定的制约。

软件设计是一门科学，有其自身的规律，也有很多成熟的理论和算法。要学习就要选教材，而目前所能选到的都是专为计算机专业编写的教材。这些教材起点较高，偏重理论叙述，不考虑嵌入式系统的硬件特点，对于广大嵌入式系统开发人员来说不是十分适合，学起来会感到比较抽象和吃力。

出于提高我国广大嵌入式系统开发人员软件素质的愿望，我们决定编写一本适合自学的关于常用算法的书。该书起点要求不高，只要掌握了 C 语言、学习了“数据结构”有关知识并从事过嵌入式系统开发工作的人员就可以看懂。学完本书后，对软件设计中常用的算法就能初步掌握。在进行软件设计时，可以减少很多盲目性，并为更系统、更深入地学习其他计算机软件设计理论打下良好基础。

本书主要内容如下：

第 1 章介绍常用线性方程组求解算法；

第 2 章介绍常用代数插值和曲线拟合算法；

第 3 章介绍常用数值积分算法；

第 4 章介绍常用能谱处理算法；

第 5 章介绍常用数字滤波算法；

第 6 章介绍常用数理统计算法；

第 7 章介绍常用自动控制算法；

第 8 章介绍常用数据排序算法；

第 9 章介绍常用数据压缩算法；

第 10 章介绍常用检错与纠错算法。

本书编写的原则是：尽量结合嵌入式系统的应用实例，采用通俗易懂的叙述方式，介绍最基本的核心内容，以便读者能够顺利入门，为进一步学习更多的算法打下基础。

在本书的编写过程中，得到北京航空航天大学出版社的大力支持，何立民教授给予了无私帮助，在此表示衷心感谢！周立功先生在本书的策划过程中起了很大促进作用，在此也表示衷心感谢！王冬霞参与了部分算法程序的调试工作，在此一并感谢！

由于作者水平有限，书中一定会有错误及不足之处，敬请广大读者予以指正，不胜感谢！

作　者

于东华理工大学

2009 年 8 月

目 录

第1章

常用线性方程组求解算法

在一个嵌入式系统中，往往需要从外部环境中获取若干个信息（多输入），然后通过数据处理，从中求解出若干个结果（多输出）。如果多输出与多输入之间为线性关系，则将其称为“多变量线性系统”。在多变量线性系统中，经常要碰到求解多元线性方程组的问题，线性方程组的一般形式为：

$$\begin{cases} A_{11}X_1 + A_{12}X_2 + \cdots + A_{1n}X_n = B_1 \\ A_{21}X_1 + A_{22}X_2 + \cdots + A_{2n}X_n = B_2 \\ \quad\quad\quad\quad \vdots \\ A_{n1}X_1 + A_{n2}X_2 + \cdots + A_{m}X_n = B_n \end{cases}$$

写成矩阵形式便是：

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{m} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix}$$

线性方程组的求解算法有直接法和迭代法等，它们各自又有分支。从嵌入式系统的特点出发，在这里只讨论最基本的主元消去法和行列式法，它们是直接法的代表。在本章的应用实例中，将介绍一个直接法和迭代法相结合的算法。

1.1 主元消去法

主元消去法又分总体选主元消去法和按列选主元消去法，由于前者每次选主元的范围大，速度相对慢，所以采用按列选主元消去法。主元消去法又分为无回代过程的主元消去法和有回代过程的主元消去法。

1.1.1 无回代过程的主元消去法

这里，通过求解一个具体的线性方程组来说明这一方法的计算步骤。

$$2x_1 + 3x_2 + x_3 = 1 \tag{1-1}$$

$$x_1 + 2x_2 - x_3 = -3 \tag{1-2}$$

$$4x_1 + 2x_2 - x_3 = 0 \tag{1-3}$$

写成矩阵形式便是：

$$\begin{bmatrix} 2 & 3 & 1 \\ 1 & 2 & -1 \\ 4 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \\ 0 \end{bmatrix}$$

解：

第1步：为了提高计算的精度，在第1列找出绝对值最大的系数作为第1列的主元，这里是第3行的4。为了使主元位于主对角线上，把方程组中的式(1-1)和式(1-3)交换位置，得到等效的方程组：

$$4x_1 + 2x_2 - x_3 = 0 \tag{1-4}$$

$$x_1 + 2x_2 - x_3 = -3 \tag{1-5}$$

$$2x_1 + 3x_2 + x_3 = 1 \tag{1-6}$$

写成矩阵形式便是：

$$\begin{bmatrix} 4 & 2 & -1 \\ 1 & 2 & -1 \\ 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 1 \end{bmatrix}$$

第2步：用主元除式(1-4)中各个元素，则方程组变为：

$$x_1 + 0.5x_2 - 0.25x_3 = 0 \tag{1-7}$$

$$x_1 + 2x_2 - x_3 = -3 \tag{1-8}$$

$$2x_1 + 3x_2 + x_3 = 1 \tag{1-9}$$

写成矩阵形式便是：

$$\begin{bmatrix} 1 & 0.5 & -0.25 \\ 1 & 2 & -1 \\ 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 0 \end{bmatrix}$$

第3步：用高斯消去法将式(1-8)和式(1-9)中 x_1 的系数变为0。

高斯消去法的算法原理如下：

对于一个 n 阶线性方程组，设当前主元位于第 i 行第 i 列(主元一定在主对角线上)，通过前面若干步骤的处理，已经将主元调整为1，主元左边各个系数均已为零，则主元所在的第 i 行成为：

$$X_i + A_{i,i+1}X_{i+1} + \cdots + A_{i,n-1}X_{n-1} + A_{i,n}X_n = B_i \tag{1-10}$$

即：

$$X_i = B_i - (A_{i,i+1}X_{i+1} + \cdots + A_{i,n-1}X_{n-1} + A_{i,n}X_n) \tag{1-11}$$

这时待消元对象所在的第 k 行是：

$$A_{k,i}X_i + A_{k,i+1}X_{i+1} + \cdots + A_{k,n-1}X_{n-1} + A_{k,n}X_n = B_k \tag{1-12}$$

即：

$$X_i = [B_k - (A_{k,i+1}X_{i+1} + \cdots + A_{k,n-1}X_{n-1} + A_{k,n}X_n)]/A_{k,i} \tag{1-13}$$

由式(1-11)和式(1-13)可得：

$$(A_{k,i+1}X_{i+1} + \cdots + A_{k,n-1}X_{n-1} + A_{k,n}X_n) - A_{k,i} \times (A_{i,i+1}X_{i+1} + \cdots + A_{i,n-1}X_{n-1} + A_{i,n}X_n) = B_k - A_{k,i} \times B_i$$

即：

$$(A_{k,i+1} - A_{k,i} \times A_{i,i+1})X_{i+1} + \cdots + (A_{k,n} - A_{k,i} \times A_{i,n})X_n = B_k - A_{k,i} \times B_i \tag{1-14}$$

这时就得到一个 X_i 的系数为零的式(1-14)，用式(1-14)取代式(1-12)式便完成了第 k 行的消元过程。仔细观察式(1-14)，就可以得到消元的具体过程为：

$$\text{新式}(k) = \text{原式}(k) - A_{k,i} \times \text{式}(i)$$

对于这个线性方程组，目前主元是 A_{11}，并且 A_{11} 已经为1，现在需要将 A_{21} 和 A_{31} 进行消元处理，具体过程如下：

$$\text{式}(1-8) - A_{21} \times \text{式}(1-7) = \text{式}(1-8) - \text{式}(1-7)$$

$$\text{式}(1-9) - A_{31} \times \text{式}(1-7) = \text{式}(1-9) - 2 \times \text{式}(1-7)$$

消元后得到线性方程组为：

$$x_1 + 0.5x_2 - 0.25x_3 = 0 \tag{1-15}$$

$$1.5x_2 - 0.75x_3 = -3 \tag{1-16}$$

$$2x_2 + 1.5x_3 = 1 \tag{1-17}$$

写成矩阵形式便是：

$$\begin{bmatrix} 1 & 0.5 & -0.25 \\ 0 & 1.5 & -0.75 \\ 0 & 2 & 1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 1 \end{bmatrix}$$

第4步：与第1步相同，在第2列找出绝对值最大的系数作为第2列的主元，不过查找范围是从第2行开始的以下各行。查找结果是第3行的2。为了使主元位于主对角线上，把方程组中的式(1-16)和式(1-17)交换位置，得到等效的方程组：

$$x_1 + 0.5x_2 - 0.25x_3 = 0 \tag{1-18}$$

$$2x_2 + 1.5x_3 = 1 \tag{1-19}$$

$$1.5x_2 - 0.75x_3 = -3 \tag{1-20}$$

写成矩阵形式便是：

$$\begin{bmatrix}1 & 0.5 & -0.25\\ 0 & 2 & 1.5\\ 0 & 1.5 & -0.75\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}0\\ 1\\ -3\end{bmatrix}$$

第5步：与第2步相同，用主元除式(1-19)各个元素，则方程组变为：

$$x_1+0.5x_2-0.25x_3=0 \tag{1-21}$$

$$x_2+0.75x_3=0.5 \tag{1-22}$$

$$1.5x_2-0.75x_3=-3 \tag{1-23}$$

写成矩阵形式便是：

$$\begin{bmatrix}1 & 0.5 & -0.25\\ 0 & 1 & 0.75\\ 0 & 1.5 & -0.75\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}0\\ 0.5\\ -3\end{bmatrix}$$

第6步：与第3步相同，用高斯消去法将式(1-21)和式(1-23)中x_2的系数变为0。具体过程如下：

式(1-21)$-A_{12}\times$式(1-22)=式(1-21)$-0.5\times$式(1-22)

式(1-23)$-A_{32}\times$式(1-22)=式(1-23)$-1.5\times$式(1-22)

消元后得到线性方程组为：

$$x_1-0.625x_3=-0.25 \tag{1-24}$$

$$x_2+0.75x_3=0.5 \tag{1-25}$$

$$-1.875x_3=-3.75 \tag{1-26}$$

写成矩阵形式便是：

$$\begin{bmatrix}1 & 0 & -0.625\\ 0 & 1 & 0.75\\ 0 & 0 & -1.875\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}-0.25\\ 0.5\\ -3.75\end{bmatrix}$$

第7步：与第1步相同，在第3列找出绝对值最大的系数作为第3列的主元，不过查找范围是从第3行开始的以下各行。因为第3行是最后一行，主元就是第3行的-1.875。

第8步：与第2步相同，用主元除式(1-26)中各个元素，则方程组变为：

$$x_1-0.625x_3=-0.25 \tag{1-27}$$

$$x_2+0.75x_3=0.5 \tag{1-28}$$

$$x_3=2 \tag{1-29}$$

写成矩阵形式便是：

$$\begin{bmatrix}1 & 0 & -0.625\\ 0 & 1 & 0.75\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}-0.25\\ 0.5\\ 2\end{bmatrix}$$

第 9 步：与第 3 步相同，用高斯消去法将式(1－27)和式(1－28)中 x_3 的系数变为 0。具体过程如下：

$$式(1-27)-A_{13}\times式(1-29)=式(1-27)-(-0.625)\times式(1-29)$$

$$式(1-28)-A_{23}\times式(1-29)=式(1-28)-0.75\times式(1-29)$$

消元后得到线性方程组为：

$$x_1 = 1 \tag{1-30}$$

$$x_2 = -1 \tag{1-31}$$

$$x_3 = 2 \tag{1-32}$$

写成矩阵形式便是：

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}1\\ -1\\ 2\end{bmatrix}$$

至此，方程组的 3 个根全部解出。分析总结以上求解过程，可以看出：每个未知数的系数对应一列系数，为了提高计算精度，从中找出绝对值最大的系数作为主元，并将主元调整到主对角线上。为了减少消元过程中的运算量，先将主元除该行各个系数和常数项，使主元归一化。其他各行通过和主元所在行进行线性运算，就可以将其他各行中主元对应列的系数变为 0(消元过程)。最后除主对角线各个系数均为 1 外，其他系数全部为零，成为单位系数矩阵，这时各个常数项的值就是各个根的值。无回代过程的主元消去法解线性方程组的算法流程图如图 1－1 所示。

无回代过程的主元消去法解线性方程组的程序如下：

程序清单 1－1　无回代过程的主元消去法程序

```
#include <stdio.h>
#include <math.h>   //需要使用其中的绝对值函数 fabs()
#define N   3       //线性方程组的阶数
float A[N][N+1];    //线性方程组的增广系数矩阵(将常数项 B 移到系数矩阵内,作为最后一列)
float X[N];         //线性方程组的解

void findmain(int i)  //寻找第 i 列的主元,并将其所在行交换到当前处理行位置上
{
    int j,k;
    float c;
    c=fabs(A[i][i]);k=i;                    //初始化主元在第 i 行
    for(j=i+1;j<N;j++)                      //寻找主元(绝对值最大)所在行
        if(fabs(A[j][i])>c){
            c=fabs(A[j][i]);
```

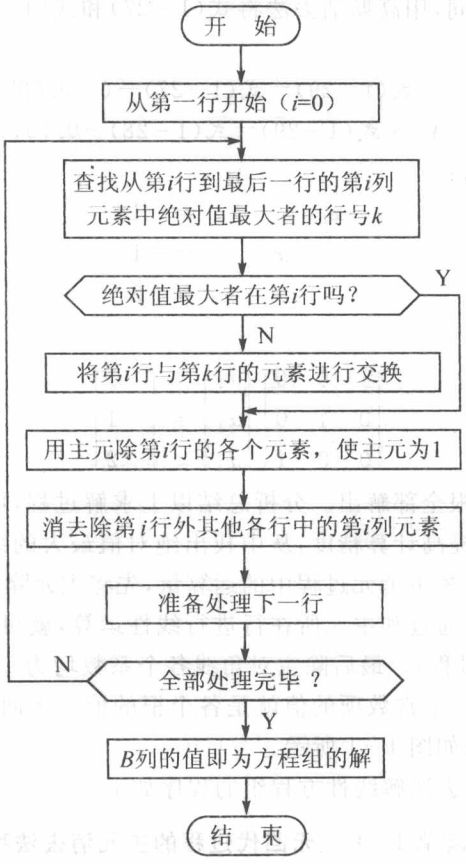


图 1-1　无回代过程主元消去法流程图

```
            k = j;
            }
    if (k! = i) for (j = 0;j<= N;j ++ ) {          //将主元所在行交换到当前处理行位置上
        c = A[k][j];
        A[k][j] = A[i][j];
        A[i][j] = c;
        }
}

void divmain (int i)                               //将主元所在行的各个系数除以主元,使主元为 1
{
    int j;
```

```
    float c;
    c = A[i][i];                                    //取出主元的原数值
    A[i][i] = 1.0;                                  //使主元为 1
    for (j = i + 1;j<= N;j ++ ) A[i][j]/ = c;       //将主元所在行的各个系数除以主元的
                                                    //原数值
}

void del (int i)                                    //进行第 i 列的消元处理
{
    int j,k;
    float c;
    for (j = 0;j<N;j ++ ) if (j! = i && A[j][i]) {
                                    //只处理除 i 行之外且 i 列系数非零的行(行下标为 j)
        c = A[j][i];A[j][i] = 0;        //记录被消系数的原数值,然后将其消零
        for (k = i + 1;k<= N;k ++ ) A[j][k] - = c * A[i][k];    //调整同行的其他系数
        }
}

int main ( )
{
    int i,j;
    printf("\n 请输入线性方程组的增广系数矩阵: \n");
    for(i = 0;i<N;i ++ ) {
        for(j = 0;j<N;j ++ ) {
            printf("A % d % d = ",i + 1,j + 1);
            scanf(" % f",&A[i][j]);
            }
        printf("B % d = ",i + 1);
        scanf(" % f",&A[i][N]);
        }
    getchar();
    for (i = 0;i<N;i ++ ) {                 //按行处理
        if (i<N - 1) findmain (i);          //寻找主元,并将其交换到当前处理行位置上
        divmain (i);                        //将当前处理行的各个系数除以主元,使主元为 1
        del (i);                            //进行消元处理
        }
    for (i = 0;i<N;i ++ ) X[i] = A[i][N];   //保存 n 阶线性方程组的解
    printf("\n 线性方程组的解: \n");
    for (i = 0;i<N;i ++ ) printf("X % d = % f     ",i + 1,X[i]);
```

```
    getchar();
    return 0;
}
```

1.1.2 有回代过程的主元消去法

在前面介绍的无回代过程的主元消去法中，主对角线右上方的系数在被消元前需要多次进行运算调整。以最右上方的常数项 B_1 为例，在式(1-4)中开始是0，当将 A_{12} 进行消元时 B_1 被调整为式(1-24)中的−0.25，当将 A_{13} 进行消元时 B_1 又被调整为式(1-30)中的1。当线性方程组的未知数较多时，这种调整运算量就会明显增加。为了减少运算量，人们对前面的算法进行了改进，得到求解高阶线性方程组的有回代过程的主元消去算法。我们仍然以前面的方程组为例来说明算法过程：

第1步～第5步：算法与无回代过程算法相同。

第6步：与第3步相同，用高斯消去法将式(1-23)中 x_2 的系数变为0。具体过程如下：

$$式(1-23)-A_{32}\times式(1-22)=式(1-23)-1.5\times式(1-22)$$

式(1-21)和式(1-22)维持不变，则方程组变为：

$$x_1+0.5x_2-0.25x_3=0 \tag{1-33}$$

$$x_2+0.75x_3=0.5 \tag{1-34}$$

$$-1.875x_3=-3.75 \tag{1-35}$$

写成矩阵形式便是：

$$\begin{bmatrix}1 & 0.5 & -0.25\\ 0 & 1 & 0.75\\ 0 & 0 & -1.875\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}\begin{bmatrix}0\\ 0.5\\ -3.75\end{bmatrix}$$

这里开始显示出和无回代过程算法的区别，它不是将主元所在列的其他系数全部消元，只是将主元下方的系数进行消元处理，即只将主对角线左下方的系数进行消元处理。

第7步：与第1步相同，在第3列找出绝对值最大的系数(即第3列的主元)，不过查找范围是从第3行开始的以下的各行。因为第3行是最后一行，主元就是第3行的−1.875。

第8步：与第2步相同，用主元除式(1-35)中各个元素，则方程组变为：

$$x_1+0.5x_2-0.25x_3=0 \tag{1-36}$$

$$x_2+0.75x_3=0.5 \tag{1-37}$$

$$x_3=2 \tag{1-38}$$

写成矩阵形式便是：

$$\begin{bmatrix} 1 & 0.5 & -0.25 \\ 0 & 1 & 0.75 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \\ 2 \end{bmatrix}$$

至此，主对角线左下方的全部系数均为零，主对角线系数全部为1，这时最后一行的常数项就是一个根，即 $X_n=B_n$，消元过程结束。

第9步：从倒数第2行开始回代过程，将式(1-38)的结果代入式(1-37)可得：

$$x_2=0.5-0.75x_3=0.5-0.75\times 2=-1$$

则方程组变为：

$$x_1+0.5\,x_2-0.25x_3=0 \tag{1-39}$$

$$x_2=-1 \tag{1-40}$$

$$x_3=2 \tag{1-41}$$

写成矩阵形式便是：

$$\begin{bmatrix} 1 & 0.5 & -0.25 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}$$

第10步：继续回代过程，将式(1-40)和式(1-41)的结果代入式(1-39)可得：

$$x_1=0-0.5\,x_2+0.25x_3=-0.5\times(-1)+0.25\times 2=1$$

则方程组变为：

$$x_1=1 \tag{1-42}$$

$$x_2=-1 \tag{1-43}$$

$$x_3=2 \tag{1-44}$$

写成矩阵形式便是：

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

至此方程求解完毕。我们仔细分析回代过程可以看出，由于回代过程是从下向上进行的，当进行第 i 行的回代运算(求解 X_i)时，从第 $i+1$ 行到最后一行均已成为 $X_k=B_k$ 的形式，而：

$$X_i+A_{i,i+1}X_{i+1}+A_{i,i+2}X_{i+2}+\cdots+A_{i,n}X_n=B_i$$

故：

$$\begin{aligned} X_i&=B_i-(A_{i,i+1}X_{i+1}+A_{i,i+2}X_{i+2}+\cdots+A_{i,n}X_n) \\ &=B_i-(A_{i,i+1}B_{i+1}+A_{i,i+2}B_{i+2}+\cdots+A_{i,n}B_n) \end{aligned}$$

有回代过程的算法显然比无回代过程的算法要麻烦一些，故这种算法只有在求解高阶线性方程组时才使用，以减少总的运算次数。