TONGJIDAXUECHUBANSHE

燃气工程电算方法

冯 良 张 同 全惠君 编著

同济大学出版社

内容提要

本书针对燃气管网计算分析中遇到的一些数值计算原理，较系统地介绍了非线性方程、线性方程组、插值与拟合、数值积分和微积分的计算原理和数值分析方法，并结合燃气管网，重点介绍了网络拓扑的基本概念、拓扑矩阵的计算机形成方法、稀疏矩阵技术和非线性方程组的迭代解法。书中所述算法程序均经实际调试，具有较实用的价值。

本书供燃气工程专业学生作教材使用，同时，也可供燃气工程技术人员参考。

责任编辑　胡兆民
封面设计　陈益平

燃气工程电算方法
冯　良　张　同　全惠君　编著
同济大学出版社出版
（上海四平路1239号）
新华书店上海发行所发行
上海青年报印刷厂印刷
开本：787×1092　1／16　印张：5　字数：125千字
1998年2月第1版　1998年2月第1次印刷
印数：1—1000册　定价：6.00元
ISBN 7-5608-1820-X／TP·201

目　录

第一章　方程求根

本章主要讨论数学的一个基本问题——求给定方程的根。对于高于四次的代数方程，一般不存在通用的求根公式来计算其准确解，超越方程一般也不能求出其准确解，即此时不用解析法求方程的根，而只能用数值方法求方程的近似根。

一般方程式 $f(x)=0$ 的根或解是指使 $f(x)$ 等于零的那些 x 值。方程 $f(x)=0$ 可能无根，也可能有单根、多根甚至无限多根。本章着重讨论多项式方程的数值计算方法求根。设 m 次多项式方程的一般形式为

$$f(x)=a_m x^m + a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 = 0 \tag{1-1}$$

式中，a_m，a_{m-1}，$\cdots$，a_0 是已知常数。

方程式求根通常分两步走：第一步先确定根的某个粗略的近似值所谓“初始近似值”，然后再将初始近似值逐步加工成满足精度要求的结果。本章介绍几种常用的计算方法。

§1.1　二分法

设函数 f 在区间 $[a,b]$ 上连续，且 $f(a)$，$f(b)$ 异号，则在 (a,b) 内至少有一点 x，使得方程 $f(x)=0$。这样的点 x 叫做函数 f 的一个零点（或根）。

令 s 为区间 $[a,b]$ 的中点，若 $f(s)=0$，则 s 是函数 f 的一个零点，若 $f(s)\neq 0$ 则以下不等式之一成立：$f(s)f(a)<0$ 或 $f(s)f(b)<0$。在第一种情况下，区间 (a,s) 至少含有函数 f 的一个零点，在第二种情况下，函数 f 在区间 (s,b) 中至少含有一个零点。对重新起 $[a,b]$ 区间作用的 $[a,s]$ 或 $[s,b]$ 逐次地重复上述步骤，我们可以任意精确地求得 f 的零点。该逼近过程称为二分法。图 1-1 说明了它的应用，图中，x_2，x_3，x_4 表示逐次得到的中点。

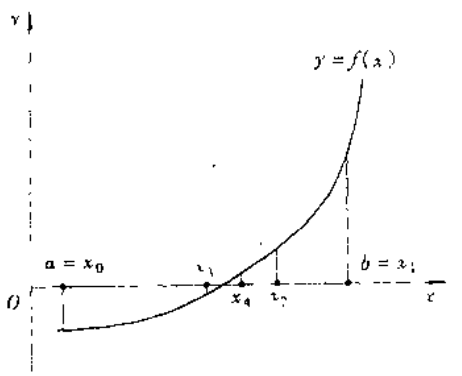

图 1-1　二分法的说明

在实际的方程求解过程中，可采用具体做法如下：

从左端点 $x=a$ 出发，按某个预选的

步长 h，一步一步向右跨，每跨一步进行一次根的“扫描”，即检查每一步的起点 x_0 和终点 x_0+h 的函数值。对于每一个小区间$[x_i, x_{i+1}](x_{i+1}=x_i+h)$：

若 $f(x_i)=0$，则 x_i 为一实根，且从 $x_i+h/2$ 开始再往后搜索；

若 $f(x_i)f(x_{i+1})=0$，则 x_{i+1}为一实根，且从 $x_{i+1}+h/2$ 开始再往后搜索；

若 $f(x_i)f(x_i+h)>0$，则说明当前小区间内无实根，从 x_{i+1}开始再往后搜索；

若 $f(x_i)f(x_i+h)<0$，则说明当前小区间内有一实根。此时，反复将区间减半，直到区间长度小于 ϵ 为止，区间的中点即为方程的一个实根。然后再从 x_{i+1}开始往后搜索。其中 ϵ 为预先给定的精度要求。

以上过程一直作到整个区间端点 b 为止。

在求方程的实根时，要注意步长 h 的选择。若步长 h 选得过大，可能会导致某些实根的丢失；若步长 h 选得过小，则计算工作量太大。

需要注意的是，由于函数的符号在一个偶数重根的邻域内不改变，所以通常不能用二分法求得偶数重根。

二分法的 FORTRAN 子程序如下：

哑元说明：

A——实型变量，输入参数。求根区间的左端点。

B——实型变量，输入参数。求根区间的右端点。

H——实型变量，输入参数。搜索求根时采用的步长。

EPS——实型变量，输入参数。预先给定的精度要求。

X——实型一维数组，长度为 N，输出参数存放所有的单实根（前 M 个）。

N——整型变量，输入参数。方程根个数的预估值，也即一维数组 X 的长度。

M——整型变量，输出参数。求根区间内方程实根的实际个数。若 M = N，则有可能在求根区间(a,b)内实根围搜索完。

F——实型函数名，输入参数。用于计算方程 f(x) = 0。

```
subroutine dlurt(a,b,h,eps,x,n,m,f)
real h
dimension x(n)
m=0
x1=a
100 f1=f(x1)
x2=x1+h/2.0
f2=f(x2)
if(abs(f1).lt.eps)then
  m=m+1
  x(m)=x1
  x1=x1+h/2.0
elseif(abs(f2).lt.eps)then
  m=m+1
  x(m)=x2
  x1=x2+h/2.0
```

```
      elseif(f1 * f2.gt.0.0)then
        x1 = x2
      elseif(f1 * f2.lt.0.0)then
200     x3 = (x1 + x2)/2.0
        f3 = f(x3)
        if(f1 * f3.gt.0.0)then
          x1 = x3
          f1 = f3
        else
          x2 = x3
          f2 = f3
        endif
        if(abs(x2 - x1).gt.eps.and.abs(f3).gt.eps)goto 200
        m = m + 1
        x(m) = x3
        x1 = x3 + h/2.0
      endif
      if(x1.lt.b + h/2.0.and.m.ne.n)goto 100
      return
      end
```

【例】 求方程

$$f(x) = x^6 - 5x^5 + 3x^4 + x^3 - 7x^2 + 7x - 20 = 0$$

在区间[-2,5]内的全部单实根。取步长 $H = 0.2$，$\varepsilon = 0.000001$。方程中各系数均为准确值。

因为这是一个6次方程，最多有6个实根，取 $N = 6$。

要求计算结果取到小数点后5位。

主程序及函数子程序如下：

```
      dimension x(6)
      external f
      call dhrt(-2.0,5.0,0.2,1e-6,x,6,m,f)
      write(*,10)m
10    format(1x,'m = ',i2)
      do 20 i = 1,m
20    write(*,30)i,x(i)
30    format(1x,'x('i2,') = ',e15.6)
      end

      function f(x)
      f = (((((x - 5) * x + 3) * x + 1) * x - 7) * x + 7) * x - 20.0
      return
      end
```

运算结果如下：

```
m = 2
x(1) =      - .140246E + 01
x(2) =        .433376E + 01
```

§1.2 定点迭代法

如果一个单变量的非线性代数方程

$$f(x)=0$$

可以表达为

$$f(x)=x-F(x)=0$$

的形式，则解方程就等效于解方程

$$x=F(x) \tag{1-2}$$

我们称上式为函数 $f(x)$ 的定点式，并且称 $f(x)$ 的正确解 x^* 为函数 $F(x)$ 的定点。

定点式的求解可以通过逐步猜试的办法来实现。比如，先任选初始猜值 $x=x^0$，显然 x^0 不会恰好就是 x^*，所以需要继续猜试。在一定的条件下，可以证明，如果将 $x^1=F(x^0)$ 作为第二个猜值，它会比 x^0 更接近 x^*，即有 $|x^*-x^1|<|x^*-x^0|$。所以，依次类推，就可以得到所谓的定点迭代算法，其步骤如下：

(1) 给定某允许误差值 ε，并任选初始猜值为 x^0。令 $j=0$，$x^{(j)}=x^0$。

(2) 计算 $x^{(j+1)}=F(x^{(j)})$。

(3) 若 $|x^{(j+1)}-x^{(j)}|<\varepsilon$，则算法终止，否则转(4)。

(4) 令 $j=j+1$，转(2)。

定点迭代法的几何意义可以用图 1-2 来加以说明。图中曲线 $y=F(x)$ 与直线 $y=x$ 的交点所对应的就是方程的正确解 x^*，即函数 $F(x)$ 的定点。定点迭代法的全部过程对应于图中的折线路径，其中箭头代表过程进行的方向。首先在曲线上任选一点起始点 a(对应于初始猜值 x^0)，然后过 a 点作平行线交直线于 b 点(它的横坐标 x^1 就是第二个猜值)，再过 b 点作垂直线交曲线于 c 点，然后再过 c 点作平行线交直线于 d 点(它的横坐标 x^2 就是第三个猜值)，……如此继续下去，直到这些交点逐步逼近到定点为止。

可以验证图 1-2 所示的例子。不管选择什么起点，定点迭代法总会收敛到定点。但是，这并不表明对于所有的情况应用定点迭代法都会收敛。例如，对于图 1-3 所示的情况，选择任何起点算法都是发散的。

【例】 用迭代法求以下方程的根：

$$f(x)=x^2-\sqrt{x}-3=0$$

精度要求为 $\varepsilon=0.000001$，最大迭代次数为 60。

我们取 $x=F(x)=(3+\sqrt{x})^{1/2}$，因此它的迭代公式为

$$x_{n+1}=(3+\sqrt{x_n})^{1/2}$$

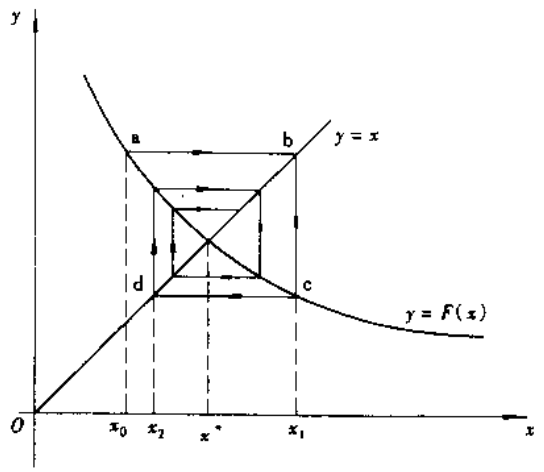

图 1-2　定点迭代法的几何意义

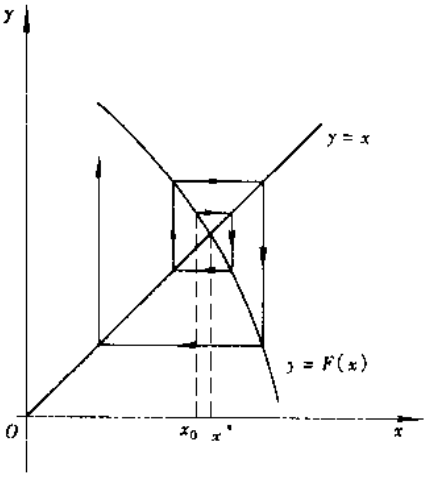

图 1-3　定点迭代法的发散情况

FORTRAN 程序如下：

```
      f(x) = sqrt(3 + sqrt(x))
      1 = 60
      x0 = 2.0
      eps = 0.000001
10    x1 = f(x0)
      if(abs(x1 - x0).gt.eps.and.1.gt.0)then
          1 = 1 - 1
          x0 = x1
          goto 10
      endif
      write( * ,20)x1
20    format(1x,'x = ',e12.6)
      end
```

运行结果为

```
x =        .211012E + 01
```

§1.3　牛顿法

牛顿法是求解方程 $f(x)=0$ 的一种重要的迭代法。这种方法的基本思想是设法将非线性方程 $f(x)=0$ 逐步转化为某种线性方程来求解。

假定 $x=x^{(j)}$ 为第 j 次的猜值，且假定它还不是正确解，即 $f(x^{(j)})\neq 0$，于是，可以将 $f(x)$ 在 $x=x^{(j)}$ 处进行泰勒展开：

$$f(x)=f(x^{(j)})+f'(x^{(j)})(x-x^{(j)})+\text{高次项}$$

若令下一个猜值为 $x=x^{(j+1)}$ 很接近正确解，从而使得 $x^{(j+1)}-x^{(j)}$ 非常小，忽略上式中

的高次项，得近似式：

$$f(x^{(j+1)}) = f(x^{(j)}) + f'(x^{(j)})(x^{(j+1)} - x^{(j)})$$

因为我们希望 $x^{(j+1)}$ 是方程的解，所以应该选择 $x^{(j+1)}$，使得 $f(x^{(j+1)}) = 0$，于是有

$$f(x^{(j)}) + f'(x^{(j)})(x^{(j+1)} - x^{(j)}) = 0 \tag{1-3}$$

再从上式解出 $x^{(j+1)}$，就得到牛顿迭代公式：

$$x^{(j+1)} = x^{(j)} - f(x^{(j)})/f'(x^{(j)}) \tag{1-4}$$

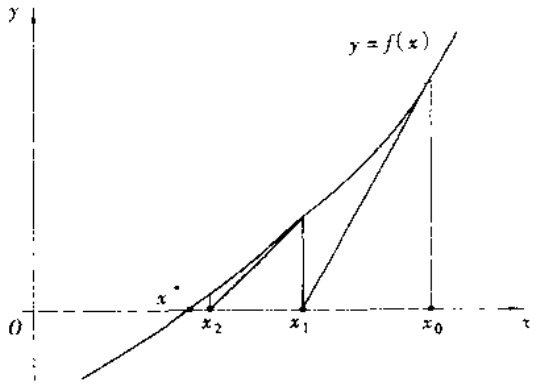

图 1-4 Newton 法的说明

牛顿迭代法的几何意义可以用图 1-4 来加以说明。图中曲线 $y = f(x)$ 与 x 轴的交点就是一个非线性代数方程 $f(x) = 0$ 的解点。假定 $P^{(j)}$ 为曲线上与 $x^{(j)}$ 相对应的点，在这一点的曲线斜率为：$f'(x^{(j)})$。根据牛顿迭代公式，得下一个迭代点的 $x^{(j+1)}$ 是从原点到曲线上过 $P^{(j)}$ 点切线与 x 轴交点的距离，从而得到下一个迭代点 $P^{(j+1)}$。重复上述步骤，迭代可以很快收敛到解点。

牛顿法的收敛性与初始猜值的好坏有关，这是在推导牛顿迭代公式时所作假定的必然结果。当初始猜值远离解点时，算法可能发散。

显然，用牛顿迭代法只能求方程的一个实根。可以证明，当方程 $f(x) = 0$ 满足以下条件：

(1) $f(x)$ 在闭区间 $[a, b]$ 上，$f'(x)$ 与 $f''(x)$ 均存在，且各自保持固定符号；

(2) $f(a) \cdot f(b) < 0$；

(3) $f(x_0) \cdot f''(x) > 0$，x、$x_0 \in [a, b]$

则方程 $f(x) = 0$ 在区间 $[a, b]$ 上有且只有一个实根，取初值 x_0，有牛顿迭代格式：

$$x_{n+1} = x_n - f(x_n)/f'(x_n) \tag{1-5}$$

计算得到的序列 $x_0, x_1, \cdots, x_n, \cdots$，收敛于方程 $f(x) = 0$ 的根。

结束迭代过程的条件为

$$|f(x_{n+1})| < \varepsilon \text{ 和 } |x_{n+1} - x_n| < \varepsilon$$

其中，ε 为预先给定的精度要求。

牛顿法 FORTRAN 子程序如下：

哑元说明：

x——实型变量，输入兼输出参数。调用时存放迭代初值；返回时存放方程实根的近似值。

eps——实型变量，输入参数。预先给定的精度要求。

fs——子程序名，输入参数。用于计算方程 f(x) = 0 的左端函数的值与导数值。

l——整型变量，输入兼输出参数。输入时用于控制迭代次数，输出时，如果 l = 0，则说明求根没满足精度或迭代发散。

q——逻辑变量，输出参数，当导数为 0 时，q 为假。

```
      subroutine newton(x,eps,fs,l,q)
      logical p,q
      call fs(x,f,df)
      p=.true.
      if(df.ne.0.0)then
        q=.true.
      else
        q=.false.
      endif
100   if(l.gt.0.and.p.and.q)then
        x1=x-f/df
        if(abs(x1-x).lt.eps.or.abs(f).lt.eps)then
          p=.false.
        else
          l=l-1
          x=x1
          call fs(x,f,df)
          if(df.eq.0.0)q=.false.
        endif
        goto 100
      endif
      return
      end
```

【例】 设方程

$$f(x)=x^3-x^2-1=0$$

导函数为

$$f'(x)=3x^2-2x$$

求此方程在 $x_0=1.5$ 附近的一个实根，取 $\varepsilon=0.000001$，最大迭代次数为 60 次。

```
      logical q
      external fs
      x=1.5
      eps=1e-6
      l=60
      call newton(x,eps,fs,l,q)
      if(q)then
        if(l.eq.0)then
          write(*,*)'err1'
        else
          write(*,10)x
        endif
      else
```

```
      write( * , * )'err2'
    endif
10  format(1x,'x = ',e15.6)
    end

    subroutine fs(x,f,df)
    f = x * x * (x - 1.0) - 1.0
    df = 3.0 * x * x - 2.0 * x
    return
    end
```

运行结果为：

```
x =        .146557E + 01
```

§1.4 弦截法

牛顿法的突出优点是收敛速度快，但它有个明显的缺点：需要计算导数 $f'(x)$，如果函数 $f(x)$ 比较复杂，使用牛顿迭代公式是不方便的。

在执行二分法时，我们每次把区间分为两个相等的部分。现对二分法加以修正，依照区间端点上函数值之比来划分区间，即成为所谓的试位法。因为在一个小区间上，可把 f 近似看成为一线性多项式，所以在 f 是可微的情况下，该过程应该是合理的。

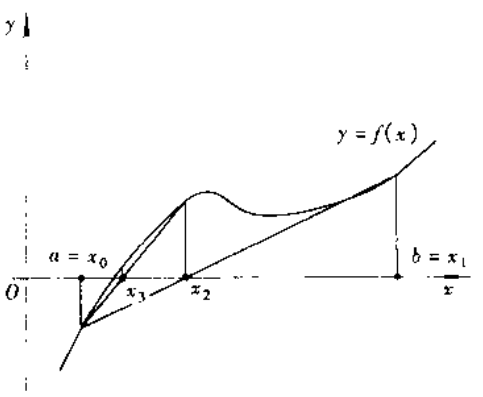

图 1-5 试位法的说明

令 $[a,b]$ 是初始区间，满足条件 $f(a)f(b)<0$；则 s 定义为 x 轴和联结两点 $(a,f(a))$、$(b,f(b))$ 的弦的交点。若 $f(s)=0$，则 f 的一个零点已经找到，故算法结束，若 $f(s)f(a)<0$，则区间 (a,s) 含有 f 的一个零点，而若 $f(s)f(b)<0$，则区间 (s,b) 至少含有 f 的一个零点。对缩减的区间逐次地重复上述过程，则从初始项 $x_0=a$，$x_1=b$ 出发，可获得一个序列 $x_k(k=0,1,2,\cdots)$，如图 1-5 所示，考虑上述方法的一般步骤，设在 k 步获得的区间以点 x_j 和 x_k 作为端点，则连结点 $(x_k,f(x_k))$ 和点 $(x_j,f(x_j))$ 的弦的方程形式为

$$y-f(x_k)=\frac{f(x_j)-f(x_k)}{x_j-x_k}(x-x_k)$$

由此推得弦和 x 轴的交点是 x_{k+1}，其中

$$x_{k+1}=x_k-\frac{f(x_k)(x_k-x_j)}{f(x_k)-f(x_j)} \tag{1-6}$$

能够证明，对连续函数 f，序列 $\{x_k\}_{k=0}^{\infty}$ 收敛且其极限等于函数 f 的一个零点。

如果从牛顿法出发，为了避免导数的计算，我们用差商

$$\frac{f(x^{(j)})-f(x^{(j-1)})}{x^{(j)}-x^{(j-1)}}$$

替换牛顿迭代公式中的导数 $f'(x^{(j)})$，得到下列离散化形式：

$$x^{(j+1)}=x^{(j)}-f(x^{(j)})(x^{(j)}-x^{(j-1)})/(f(x^{(j)})-f(x^{(j-1)})) \tag{1-7}$$

上式即为弦截法的迭代公式。

图 1-6 为弦截法前三步的几何意义。曲线 $y=f(x)$ 上取两点 P_{j-1}，P_j，对应的横坐标为 x_{j-1}，x_j，则差商

$$\frac{f(x^{(j)})-f(x^{(j-1)})}{x^{(j)}-x^{(j-1)}}$$

即为弦 P_{j-1}-P_j 的斜率。因此，按迭代公式求得的 $x^{(j+1)}$ 实际上是弦 P_{j-1}-P_j 与 x 轴的交点，因此，这种算法称为弦截法。

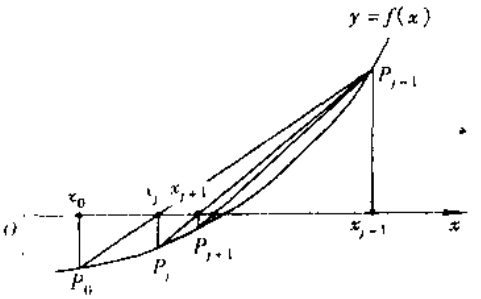

图 1-6 为弦截法的说明

在试位法中，每一步对应函数值的符号决定缩减区间的选择。但在凸函数或凹函数的情况下，每一个缩减区间的一个端点与初始区间的一个端点相重合，所以缩减区间的长度不趋于零。弦截法实际上是每一步选择 $j=k-1$ 的试位法。显然，弦截法和牛顿法一样具有较快的收敛速度，但是它也与初始猜值的好坏有关。当初始猜值远离解点时，算法可能发散。另外，虽然弦截法省略了计算导数，但它需要两个初值。

弦截法 FORTRAN 子程序如下：

哑元说明：

x0——实型变量，输入参数。调用时存放迭代初值 1；

x1——实型变量，输入参数。调用时存放迭代初值 2；

eps——实型变量，输入参数。预先给定的精度要求。

f——子程序名，输入参数。用子计算方程 f(x) = 0 的左端函数的值与导数值。

l——整型变量，输入兼输出参数。输入时用于控制迭代次数，输出时，如果 l = 0，则说明求根没满足精度或迭代发散。

```
      subroutine xj(x0,x1,eps,l,x2)
      n=1
      fx0=f(x0)
      fx1=f(x1)
10    x2=x0-fx0*(x0-x1)/(fx0-fx1)
      fx2=f(x2)
      if(abs(x2-x1).gt.eps.and.n.le.l)then
        x0=x1
        fx0=fx1
```

```
        x1 = x2
        fx1 = fx2
        n = n + 1
        goto 10
    endif
    return
    end
```

【例】 求方程

$$f(x) = x^3 - 7.8x^2 + 18.5x - 11.3 = 0$$

在 1.0 附近的一个实根，取精度 $\varepsilon = 0.000001$，最大迭代次数为 60 次。

```
    external f
    eps = 1e - 6
    1 = 60
    call xj(0.0,1.0, eps,1,x2)
    write( * ,10)x2
10    format(1x,'x = ',e12.6)
    end

    function f(x)
    f = ((x - 7.8) * x + 18.5) * x - 11.3
    return
    end
```

运行结果为

```
x =      .935621E + 00
```

第二章　线性方程组的数值解法

在工程技术中很多计算问题最后归结为求解线性方程组，求解线性方程组的方法有很多，本章中将介绍几种常用的数值解法。

§2.1　迭代法

迭代法是一种近似方法，是通过某种极限过程去逼近精确解，求得满足误差要求的近似解。本节介绍两个迭代方法——简单迭代法和逐个迭代法（即高斯-赛德尔法）。

一、简单迭代法

设有如下一般形式的线性方程组：

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

经改写后化成迭代形式：

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

于是，它的一般迭代公式如下：

$$\begin{cases} x_1^{(k+1)} = a_{11}x_1^{(k)} + a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} + b_1 \\ x_2^{(k+1)} = a_{21}x_1^{(k)} + a_{22}x_2^{(k)} + \cdots + a_{2n}x_n^{(k)} + b_2 \\ \cdots \\ x_n^{(k+1)} = a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{nn}x_n^{(k)} + b_n \end{cases} \tag{2-2}$$

首先，任意给定初始近似解 $x_1^{(0)}$，$x_2^{(0)}$，$\cdots$，$x_n^{(0)}$，并代入公式(2-2)的右端，得第一次近似解 $x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}$；再以第一次近似解 $x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}$ 代入迭代公式(2-2)，得到第二次近似解；这样做下去，可得到第 k 次近似解。在迭代过程中，随时将前后两次所得到的近似值进行比较，如果发现有

$$\left| x_i^{(k)} - x_i^{(k+1)} \right| < \varepsilon$$

$$\left|x_2^{(k)}-x_2^{(k+1)}\right|<\varepsilon$$
$$\cdots$$
$$\left|x_n^{(k)}-x_n^{(k+1)}\right|<\varepsilon$$

其中，ε 为预先要求的误差，我们的迭代过程就停止。并且取第 $k+1$ 次近似值为方程的近似解，即

$$x_1\approx x_1^{(k+1)},\quad x_2\approx x_2^{(k+1)},\cdots,\quad x_n\approx x_n^{(k+1)}$$

一个已知线性方程组化为迭代形式不是唯一的，根据任意一个迭代式进行计算，不一定都能得到逼近精确解的近似解。一个迭代公式，如果根据它能够计算出逼近精确解的近似解，也就是说近似解序列 $x_i^{(k)}(i=1,\cdots,n)$ 有极限：

$$\lim_{k\to\infty}x_i^{(k)}=x_i^*,\quad i=1,\cdots,n$$

这里，$x_i=x_i^*(i=1,\cdots,n)$ 是方程组的精确解，那么，迭代公式叫做收敛的，否则叫做发散的。

怎样的迭代公式是收敛的呢？下面介绍两个简单的判别法则。

【定理 1】 假如方程组(2-1)中每个方程右端各系数绝对值的和都不大于一个正数 μ，而 $\mu<1$，即

$$|a_{i1}|+|a_{i2}|+\cdots+|a_{in}|\leqslant\mu\quad(i=1,2,\cdots,n)$$

那么，简单迭代公式(2-2)对任取初始值都是收敛的。

【定理 2】 假如方程组(2-1)右端同一未知量的各系数绝对值的和都不大于一个正数 γ，而 $\gamma<1$，即

$$|a_{1i}|+|a_{2i}|+\cdots+|a_{ni}|\leqslant\gamma\quad(i=1,2,\cdots,n)$$

那么，简单迭代公式(2-2)对任取初始值都是收敛的。

从前面定理 1，不难推出下面的结论：

如果线性方程组

$$\begin{cases}a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1\\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2\\ \cdots\\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n\end{cases}\tag{2-3}$$

的系数 $a_{ii}(i=1,\cdots,n)$ 的绝对值都比同行其余各系数绝对值的和还大，即

$$|a_{ii}|>\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}|$$

那么，迭代公式

$$\begin{cases}x_1^{(k+1)}=-\dfrac{a_{12}}{a_{11}}x_2^{(k)}-\cdots-\dfrac{a_{1n}}{a_{11}}x_n^{(k)}+\dfrac{b_1}{a_{11}}\\ x_2^{(k+1)}=-\dfrac{a_{21}}{a_{22}}x_1^{(k)}-\cdots-\dfrac{a_{2n}}{a_{22}}x_n^{(k)}+\dfrac{b_2}{a_{22}}\\ \cdots\\ x_n^{(k+1)}=-\dfrac{a_{n1}}{a_{nn}}x_1^{(k)}-\dfrac{a_{n2}}{a_{nn}}x_2^{(k)}-\cdots+\dfrac{b_n}{a_{nn}}\end{cases}\tag{2-4}$$

对任取的初始值是收敛的。

类似的，还有一个结论：如果线性方程组(2-3)的系数 $a_{ii}(i=1,\cdots,n)$ 的绝对值都比同列的其余各系数绝对值的和还大，即

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ji}|$$

那么，迭代公式(2-4)是收敛的。

二、高斯-赛德尔迭代法

高斯-赛德尔迭代法与简单迭代法几乎完全一样，唯一的差别是它们所用的迭代公式不同。这里不用上面的迭代公式(2-2)，而是用下面所谓的高斯-赛德尔迭代公式：

$$\begin{cases} x_1^{(k+1)} = a_{11}x_1^{(k)} + a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \cdots + a_{1n}x_n^{(k)} + b_1 \\ x_2^{(k+1)} = a_{21}x_1^{(k+1)} + a_{22}x_2^{(k)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)} + b_2 \\ x_3^{(k+1)} = a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k)} + \cdots + a_{3n}x_n^{(k)} + b_3 \\ \cdots \\ x_n^{(k+1)} = a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + a_{n3}x_3^{(k+1)} + \cdots + a_{nn}x_n^{(k)} + b_n \end{cases}$$

这两个迭代公式的差别是在用第 k 次近似值 $x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)}$ 求第 $k+1$ 次近似值 $x_1^{(k+1)}, x_2^{(k+1)}, \cdots, x_n^{(k+1)}$ 时，简单迭代公式是将 $x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)}$ 整个地代入(2-1)的右端；而高斯-赛德尔迭代公式则是将 $x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)}$ 逐次代入(2-1)中，也就是说，在计算 $x_i^{(k+1)}$ 时，及时地引用新算出的 $x_1^{(k+1)}, x_2^{(k+1)}, \cdots, x_{i-1}^{(k+1)}$。一般说来，在计算过程中，及时地引用新值会比用旧值更接近精确解，收敛速度要比简单迭代快。因此逐个迭代法是更为常用的迭代方法。

显然，定理 1 和定理 2 同样也是判别高斯-赛德尔收敛公式的收敛定理。

高斯-赛德尔迭代法 FORTRAN 子程序如下：

哑元说明：

x——实型变量，输入兼输出参数。输入时放初始近似值，输出时，放计算结果。

f——实型变量，输入参数。存放线性方程组的常数项。

a——实型变量，输入参数。存放线性方程组的系数矩阵。

n——整型变量，输入参数。存放线性方程组的阶数。

eps——实型变量，输入参数。求解方程组的预先给出的精度要求。

```
      subroutine gase(x,f,a,n,eps)
      dimension x(n),f(n),a(n.n)
10    dalt=0.0
      do 100 i=1,n
         s=x(i)
         s1=0.0
         do 20 j=1,n
            if(j.ne.i)then
               s1=s1+a(i,j)*x(j)
            endif
20       continue
```

```
        x(i) = (f(i) - s1)/a(i,i)
        if(abs(x(i) - s).gt.dalt)then
            dalt = abs(x(i) - s)
        endif
100     continue
        if(dalt.gt.eps)goto 10
        end
```

【例】 求解线性方程组

$$\begin{aligned} 13x_1 + 4x_2 &= 8 \\ 6x_1 - 16x_2 + 5x_3 &= 0 \\ 5x_1 - 5x_2 + 6x_3 &= 5 \end{aligned}$$

精度要求为 $\varepsilon = 0.000001$.

主程序如下：

```
dimension a(3,3),x(3),b(3)
  data a/13.0,6.0,5.0,4.0,-16.0,-5.0,0.0,5.0,6.0/
  b/8.0,0.0,5.0/,eps/0.000001/
  data x/3*0.0/
  call gase(x,b,a,3,eps)
  do 10 i=1,3
      write(*,20)i,x(i)
10    continue
20    format(1x,'x(',i2,')=',f12.6)
      end
```

运行结果为

```
x(1) =      .483971
x(2) =      .427094
x(3) =      .785935
```

§2.2 列主元高斯消去法

高斯消去法的基本思想是：将一个方程乘或除以某个常数，或者将两个方程相加减。通过这两种步骤减少方程中变元的数目，最终使每个方程只含一个变元，从而得到所求的解。

设有如下一般形式的线性方程组：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{2-6}$$