高等学校教学用书

计算机数值方法实验

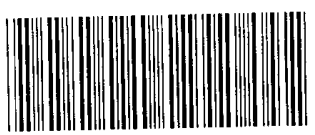
大连理工大学 吴大为 编

化学工业出版社

高等学校教学用书

计算机数值方法实验

大连理工大学　吴大为　编



化学工业出版社
·北　京·

（京）新登字 039 号

图书在版编目（CIP）数据

计算机数值方法实验/吴大为编 .—北京：化学工业出版社，2000
ISBN 7-5025-2647-1

Ⅰ.计… Ⅱ.吴… Ⅲ.电子计算机-数值计算-算法理论 Ⅳ.TP301.6

中国版本图书馆 CIP 数据核字（2000）第 00995 号

高等学校教学用书
计算机数值方法实验
大连理工大学 吴大为 编
责任编辑：程树珍
责任校对：马燕珠
封面设计：蒋艳君

*

化学工业出版社出版发行
（北京市朝阳区惠新里 3 号 邮政编码 100029）
http://www.cip.com.cn

*

新华书店北京发行所经销
北京市密云云浩印制厂印刷
北京市密云同文印刷厂装订
开本 850×1168 毫米 1/32 印张 7½ 字数 196 千字
2000 年 4 月第 1 版 2000 年 4 月北京第 1 次印刷
印 数：1—3000
ISBN 7-5025-2647-1/G·679
定 价：14.00 元

前　言

随着高技术的飞速发展和计算机的广泛普及，越来越多的科学计算问题出现在人们面前。例如方程组的求解，曲线拟合和插值，微分方程数值解，特征值问题，图的算法，最优化问题等等。它们已不是科学家才能掌握的方法，而是早已进入普通高校的课堂，进入各种程序手册，进入许多计算机软件之中。

数值计算的理论和方法并不神秘和高不可攀。只要具有一定的数学基础，通过编程和实际计算，完全可以领悟算法的构造和技巧。在结合具体问题的计算中，甚至还能够发现算法中的问题和不足，这往往成为技术改进和创新的源泉。

目前已有的数值分析教材和程序手册，都是将算法和程序分别介绍。许多抽象的数学符号吓住了初学者，一串串没有说明的程序又难住了读者，两者没有很好结合，使人不易抓住要领，不能学以至用的解决问题，使得许多人望而兴叹。因此迫切需要一本将两者结合起来的书。在大连理工大学教材出版基金的资助下，在应用数学系领导和同事们的鼓励和支持下，编写了本书。

本书以40个工程中最常用的FORTRAN程序为例，密切结合计算方法，对全部程序逐段进行了详细注释。在注释的帮助下，读者可以透彻地理解算法，并在此基础上加以改进，或移植到其他应用程序中。为了便于读者验证程序正确性，每个程序后都有计算实例，每章附有练习题。由于C语言日益普及，本书还介绍了C语言如何调用FORTRAN程序的一般方法。

本书在写作和出版过程中，受到了大连理工大学教务处王续跃处长，应用数学系林建华主任、计算数学教研室施吉林教授、刘淑珍教授、陈桂芝副教授，大连理工大学出版社刘杰编辑的关怀、鼓励和支持，罗远诠教授仔细审查了全部手稿，王宗涛同学为实现C语言对

内 容 提 要

本书针对40个工程中最常用的数值算法，在深入浅出地介绍算法原理的基础上，密切结合FORTRAN语言程序，对全部程序按算法的具体实现步骤，一一逐段进行了详细中文注释。40个算法中，包括数值积分与微分，插值与拟合，线性与非线性方程组求解，微分方程，特征值，图论和最优化算法。

本书每个程序都有计算实例，每章附有练习题，并指出构造实验例题的一般方法，本书可作为数值分析课的配套教材，也可作为计算实习课的独立教材和常用手册，可供工程技术人员学习和使用。

目　　录

第1章 绪 论

随着科学技术的传播和高等教育的发展，特别是近年来电子计算机的广泛使用，微型计算机进入千家万户，人们对计算数学和数值软件表现出日益增长的兴趣。现在较为流行的数值软件如 Mathematica，MATLAB，MathCAD 以及大型统计软件 SPSS，SAS 等，已为中国科教界广泛使用。为了更好地运用各种数值软件，学习一点计算数学的原理，掌握一些常用的数值计算方法是很必要的。

计算数学不仅在科学和工程计算中起着巨大作用，它还以严谨的逻辑、精巧的结构和简洁的形式给人以深刻的启迪。线性方程组的求解如果用 Gram 法则，计算量几乎是天文数字，但是用 Gauss 消去法则很容易；而对系数矩阵的 LU 分解、LDL 分解，更适合连续求解不同右端向量的方程。追赶法、Gauss-Seidel 迭代法和 SOR 迭代法，还可用于大型稀疏方程组。数值计算的收敛性，往往是算法成功的关键。为此，数学家构造了各种有效的计算公式，例如方程求根的 Newton 迭代公式，加速收敛的 Romberg 积分公式，高精度的 Gauss 积分公式，求微分方程解的自适应 Runge-Kutta 公式。为了避免数值不稳定和减少计算量，在求无约束极值中产生了不必求 Hessian 阵的 DFP 法。为避免复数运算，在特征值计算中产生了著名的双步 QR 算法，它巧妙的步骤令人叹为观止。大型网络问题，看起来是那么神秘复杂，但是最短路的算法，最小树的算法和最大流的算法却很简单。学习和揣摩这些算法的思想与构造，真是像在欣赏一件件精美的工艺品。

算法中的许多术语和概念，例如向量、矩阵、特征值、解集、正交、A-共轭等等，是比较抽象，但是当你真正理解了它们的含义，就会发现它是现实世界数量关系的准确投影，它概括了无数具体现象的共同特征，彻底弄懂了它，你就找到了最广泛的用途。长期坚持用

数学的符号和语言来描述事物，用数学的逻辑推理来思考问题，是一种最好的科学训练。

将算法编成程序，用计算机求解，这是数学方法的机械化。不言而喻，要取得正确的结果，首先对算法要有透彻的了解。同时，还要对计算机语言有足够的知识。数值实验就是对这两方面能力综合培养的最佳途径。如果你弄懂了本书的大部分算法和程序，加上你广泛的兴趣和动手能力，它会使你解决问题的本领倍增。

计算机软件的迅速发展，使得许多计算机语言不断淘汰和更新。FORTRAN 语言曾经在数值计算中起过重要的作用，由此产生了许多的程序集和商用软件包，至今它们还在工程和科学计算中广泛使用。但由于非数值计算的大量出现，对各种数据库的管理，图形处理，网络通讯的要求，C 语言正以强劲的势头，争得越来越多的用户。笔者充分注意到了这种发展趋势，但考虑到 FORTRAN 语言在中国仍有众多的用户，各种 FORTRAN 程序集的参考书也容易找到，FORTRAN 语言又接近英语习惯，比 C 语言容易理解。同时这里的主要目的是弄清算法和数值实验，因此，目前还是用 FORTRAN 语言书写，随着今后的发展，将根据情况，改写成其他更好的语言形式。

1.1 常用计算技巧

1.1.1 累加的技巧

在数值计算中，有许多技巧。这可以追溯到久远的年代。例如，人们经常要求多项式的值 $P_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$，这里对每给一个 x 值，都要计算出 $P_n(x)$ 的对应值。由 $P_n(x)$ 的表达式可以看到，如果给出许多不同的 x 的值，求出相应的 $P_n(x)$ 的值，是一件很繁琐的事。如何计算 $P_n(x)$ 呢？一种不动脑筋的方法是从左到右，先乘后加；先将 x 自乘 $n-1$ 次得到 x^n，然后乘上 a_0；再将 x 自乘 $n-2$ 次得到 x^{n-1}，又乘 a_1；以此类推，直至 xa_{n-1}，最后将乘法得到的结果相加得到 $P_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ 这种作法，要做 x 的乘幂共 $(n-1) + (n-2) + \cdots + 1 + 0 =$

$\frac{n(n-1)}{2}$次，x^{n-i}与a_i的乘法共n次，总共做$\frac{1}{2}n(n+1)$次乘法，还要做n次加法。人们知道，无论是手工计算还是机器计算，乘法都比加法复杂得多。那么在上面的计算中，能否改进呢?

首先，可以观察到x的自乘有大量的重复，x^n在自乘时的中间结果恰好可以作为x^2，x^3，…，x^{n-1}使用。因此，可利用另一种格式。

$$P_1(x) = a_0x + a_1, P_2(x) = P_1(x)x + a_2, P_3(x) = P_2(x)x + a_3 \cdots, P_n(x) = P_{n-1}(x)x + a_n$$

这个格式计算出的$P_n(x)$与前面的结果是完全相同的，只是，它将乘法和加法的次序作了巧妙的安排，使得乘法次数大大减少，只需要n次乘法和n次加法。这就是著名的秦九韶（1202～1261）多项式算法。

同样的思想可以推广，它对许多类似计算也是适用的。例如求和$S_n = 1 + (1+2) + (1+2+3) + \cdots + (1+2+3+\cdots+n)$，其中每个括号内的求和就有大量重复，$(1+2+3+\cdots+n)$的中间结果正好可被前面各括号求和使用，因此如换用下列格式：

$$\begin{aligned}
&P_1 = 1, && S_1 = P_1;\\
&P_2 = P_1 + 2, && S_2 = S_1 + P_2;\\
&P_3 = P_2 + 3, && S_3 = S_2 + P_3;\\
&\quad\vdots && \quad\vdots\\
&P_n = P_{n-1} + n, && S_n = S_{n-1} + P_n;
\end{aligned}$$

那么可以迅速提高计算效率。

如果说上面的S_n只是节省了重复的加法，那么下面的求和则看出该方法的优点。

$$S_n = \sin1 + (\sin1 + \sin2) + (\sin1 + \sin2 + \sin3) + \cdots + (\sin1 + \sin2 + \sin3 + \cdots + \sin n)$$

如果依次先做括号内的$\sin i$的求和，那就不仅要增加重复求和次数，而且要增加大量重复计算$\sin i$的次数，两者相比，后者不知要优越多少。

像这种求和用紧凑的符号可记为$S_n=\sum_{k=1}^{n}\sum_{i=1}^{k}i$，$S_n=\sum_{k=1}^{n}\sum_{i=1}^{k}\sin i$，一般地可写为$S_n=\sum_{k=1}^{n}\sum_{i=1}^{k}f(i)$，其中$f(i)$可以是以$i$为自变量的任何函数。例如$f(i)=i^2$，$f(i)=\frac{1}{i^2+1}$，$f(i)=\tan(i)$等等，$f(i)$越复杂，改进的计算格式效率越高。

这种求和可推广到三重求和，例如$S_n=\sum_{m=1}^{n}\sum_{k=1}^{m}\sum_{i=1}^{k}i$，$S_n=\sum_{m=1}^{n}\sum_{k=1}^{m}\sum_{i=1}^{k}\sin i$，$S_n=\sum_{m=1}^{n}\sum_{k=1}^{m}\sum_{i=1}^{k}f(i)$，根据上面作法，利用计算机编一个小程序，只要使用两个中间变量，安排一个单重循环即可。

```
      S=0:P=0:Q=0
      DO  10 I=1,N
      P=P+F(I)
      Q=Q+P
      S=S+Q
10    CONTINUE
```

若采用原方法，计算机使用三重循环，计算时间耗费很多，作为比较，列举于此。

```
      S=0
      DO  10 M=1,N
      Q=0
      DO  20 K=1,M
      P=0
      DO  30 I=1,K
      P=P+F(I)
30    CONTINUE
      Q=Q+P
20    CONTINUE
      S=S+Q
10    CONTINUE
```

当然这种方法还可推广到多重求和，重数越多，$f(i)$越复杂，其效

率越高。

作为该方法一个最常用的应用，是求 e^x，由 Maclaurin 级数可知：

$$e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!}$$

随 n 的增大 e^x 的逼近公式更精确。

利用上面所述方法，一个简洁而实用的程序是：

```
EX=1: E0=1
DO  10 I=1, N
E0=E0*X/I
EX=EX+E0
10  CONTINUE
```

循环结束后 EX 就是 e^x 的近似值。

1.1.2 公式变换的技巧

数学方法在数值计算中有着巨大的作用，许多复杂的计算过程，可以先利用数学方法将其化难为易，成百上千倍地提高计算效率。

人们知道，$f(\theta) = 1 + \cos\theta + \cos2\theta + \cdots + \cos n\theta$ 是一个三角函数的求和，如果逐项计算 $\cos i$，则要做 n 次余弦值，然后才能相加，如果 n 很大，例如 $n = 10^6$，计算时间就会很长。

由复变函数的 De Moivre 公式，$f(\theta)$ 恰是一个复级数的部分和的实部。

$$\begin{aligned}
&1 + (\cos\theta + i\sin\theta) + (\cos2\theta + i\sin2\theta) + \cdots + (\cos n\theta + i\sin n\theta) \\
&= 1 + e^{i\theta} + e^{2i\theta} + \cdots + e^{ni\theta} = \frac{1 - e^{(n+1)i\theta}}{1 - e^{i\theta}} \\
&= \frac{1 - \cos[(n+1)\theta] - i\sin[(n+1)\theta]}{1 - \cos\theta - i\sin\theta}
\end{aligned}$$

分子分母同乘 $1 - \cos\theta + i\sin\theta$ 并将实部分离出来就得到

$$1 + \cos\theta + \cos2\theta + \cdots + \cos n\theta = \frac{1}{2} + \frac{\sin\frac{2n+1}{2}\theta}{2\sin\frac{\theta}{2}}$$

显然，求和公式只要做两次正弦值即可得到正确结果。

再例如 chebyshev 多项式的求值，如果按递推公式 $T_0(x)=1$，$T_{n+1}(x)=2xT_n(x)-T_{n-1}(x)$ 求 T_{10000}，那么就要花较多时间，若直接利用通项公式 $T_n(x)=\cos(n\arccos x)$ 就简便多了。类似的还有像 Fibonacci 级数的通项公式等等带有双递推或多递推的计算公式。

有些场合，不仅仅是个公式的变换问题，甚至会有完全矛盾的结果。

例如调和级数 $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$，随着 n 的无限增加，尽管对总和的贡献 $1/n$ 越来越小，但这个级数不收敛，而是可以变得无穷大。这是因为 $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}>\int_1^n\frac{1}{x}\mathrm{d}x=\ln n$，当$n\to\infty$时调和级数就可以任意大。如果用计算机计算该级数的$n$项和，可以预见到，由于计算机字长是有限的，当 $1/n$ 小到一定程度，并与前面相对大的部分和相加时，则 $1/n$ 就被“吃”掉了，从而以后的计算再也不会使级数增加。因此，只要 n 达到一定值后，不论 n 如何增大，级数都将“收敛”到一个常数，这显然与调和级数发散的结论矛盾。如果利用 $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}=C+\ln n+\varepsilon_n$ 其中 C 为 Euler 常数，$C=0.577216\cdots$，则 $C+\ln n$ 就是调和级数最好的近似，ε_n 将很快趋于零，在与 $\ln n$ 的比较中，这点误差是微不足道的。这里可以看到计算机的速度不是万能的，它一定要结合很好的公式和算法才能奏效。

还有其他的数值计算的技巧，也是利用一些简单的恒等变换，使计算精度大大提高。

例如$\sqrt{x+1}-\sqrt{x}$，当 x 很大时，两个大数相减就丢失了很多有效数字，如果变换成$\frac{1}{\sqrt{x+1}+\sqrt{x}}$来计算就要好得多；再比如 $(x+2)x+1$，当 x 很小时，用它的恒等变换 $(x+1)^2$ 来计算就好些；当利用中心差分作导数求根时，计算 $\frac{f(x_n)}{\frac{f(x_n+h)-f(x_n-h)}{2h}}$，不

如写成 $\dfrac{2hf(x_n)}{f(x_n+h)-f(x_n-h)}$，如果分母的差分能够设法约掉则更好。凡此种种变换，在编程实践中不断注意积累和灵活运用，在大规模的计算中也许会带来很大的收益。

1.1.3 防止溢出的技巧

计算机是有限字长的机器，数值的绝对值太大或太小都会超出它所能表达的范围，使计算过程被迫中断，或者产生不正确的答案。

例如求多项式 $x^9+x^7+x^5+x^3+x-10^{10}=0$ 的实根，它在[10, 11]间有正实根，这样在计算多项式时，即会遇到大数乘法，有可能产生上溢；又会遇到大数相减使有效数字丧失；这都使计算难以精确。但是，如果在方程两边除一个适当的数 10^5，使原方程变为

$$10000\left(\frac{x}{10}\right)^9+100\left(\frac{x}{10}\right)^7+\left(\frac{x}{10}\right)^5+\frac{1}{10000}\left(\frac{x}{10}\right)-10^5=0$$

那么在计算多项式值时，就不会出现上述两个问题，从而使求根很准确。这种方法在方程求根中是一种实用技巧。

由于计算格式的改变，既防止了溢出又提高了精度的例子还有很多。例如求样本平均值 $\overline{x}=\dfrac{1}{n}(x_1+x_2+\cdots+x_n)$，一般做法是先将括号内的 x_i 相加，然后再除以 n，得到平均值。但是，如果 x_i 都比较大，同时样本又非常多，这样在第一步的累加过程中就有可能溢出，即使不溢出，一个很大的数与相对很小的数相加，也会产生“大数吃小数”的现象。同时，样本取值 x_i 往往有很多非随机的误差，例如在一群很大的 x_i 中，混有一个很奇怪的小数，它很可能是填写错误或输入错误造成的。如果不加区分，统统累加后除 n，会使平均值偏低。

如果采用下列格式

$$\overline{x}_1=x_1,\quad \overline{x}_{n+1}=\frac{n}{n+1}\overline{x}_n+\frac{x_{n+1}}{n+1}$$

那就要好得多，这里 $\overline{x}_{n+1}$是在前 n 个样本均值 $\overline{x}_n$ 的基础上，加上 x_{n+1}的信息得到的，它不仅可以避免溢出，而且可以随时掌握平均

值的变化情况。如果 x_{n+1} 与 $\overline{x}_n$ 相差太大，是否可以认为要对 x_{n+1} 进行分析，以防不良数据混淆了真实情况。

计算过程中，数值运算的情况是千变万化的，分母为零显然是要停机，分子很大而分母很小，尽管它不为零，但所得的商也很可能溢出，这样计算就难以进行。怎样使数值软件自动适应这种情况呢？可以充分利用计算机运算中“大数吃小数”的矛盾来解决。在运算中，两个数相加，首先要将其化为具有相同指数阶的表达形式，然后浮点部分相加。例如 98721.05 + 0.023496 在相加之前计算机先将 98721.05 变为 0.9872105×10^5，将 0.023496 也变为具有相同指数阶 10^5 的浮点数 0.00000023496×10^5，然后小数部分相加得到 0.98721073496×10^5。在这个对阶的过程中可以看到，如果小数部分字长限制在 7 位，那么 0.00000023496×10^5 就变成了 0.0000002×10^5，后面的小数就丢失了，如果与它相加的数再大 10 倍是 987210.5，那么在 10^6 上对阶，这时就连最后一个 2 也不存在了，变成 $0.9872105\times10^6+0.0000000\times10^6$，从而 0.023496 就被大数“吃”掉了。

既然数的量级相差到一定程度能“以大吃小”，这说明如果出现大数吃小数现象，则两数一定相差相当的数量级别。如果这样的两个数放在一起，大的做分子，小的做分母，则商要溢出，为了增强程序的“健壮性”，有必要在这种关键地方加上一条保护性语句。

```
IF  ((ABS(X)+Y-ABS(X)).NE.0.0)  THEN  Z=X/Y
```

这里 X 代表绝对值很大的数，Y 代表绝对值很小的数，如果 Y 在第一步与ABS(X)相加时就被吃掉了，那么第二步的相减就变成了ABS(X)-ABS(X)，这时结果就等于零，它说明 Y 不适合做分母。只有在相加时，Y 不被吃掉时才允许做分母，计算 Z=X/Y。

1.1.4 存储的技巧

由于计算机做数据处理时，是一步步依次进行的，许多中间结果就需要保存，这种存储量随着问题的复杂越来越大；与此密切相联的，对庞大数据库中的数据进行检索、修改和删除所花费的时间也越来越多。近年来，计算机硬件的迅速更新，使得其存储量不断增大，

对数据的操作速度日益提高，如果选择适当的数据存储方式，利用一些有效的结构，使软硬件有机结合，就会更高效地解决问题。

在数值计算中，数据存储最常见的方法就是利用数组。如一维数组它包含数组的名字和下标两部分。例如：$A(1)$，$A(2)$，…，$A(N)$，在这个数组中，每个元素都具有相同的名字“A”，而不同的下标将其区分开来，最大的下标 N 代表了数组的长度，也表明了该数组占据的存储空间的大小。二维数组包含数组名和两个下标，例如：$B(1,1)$，$B(1,2)$，…，$B(N,M)$，这样的数组可以排列成一个矩阵，

$$\boldsymbol{B}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}$$

它的第一个下标代表矩阵的行号，第二个下标代表矩阵的列号。只要给出某个元素的行列号，就可以确定它在矩阵中的位置。n 和 m 是行和列的最大值，$n\times m$ 代表数组占据的存储空间。三维数组有三个下标，例如：$C(i,j,k)$ 是三维数组中第（i，j，k）个元素。类推下去多维数组有多个下标，它占据的存储空间是各个下标最大值的乘积。

数组的形式虽然简单，但是灵活的运用却有丰富的内容。例如上面的矩阵 $\boldsymbol{B}$，它的每一行就是一个一维数组，而该行的 m 个元素可以代表 m 个不同的特征，这些特征可以是运动员的年龄、身高、体重、速度等参数，也可以是某种化工产品的纯度、组分和价格数据，n 个同类的个体，也就是 $\boldsymbol{B}$ 的 n 行存储在该二维数组之后，就可以利用统计方法对它们进行分析，比如分类、概括主要特征、找出共同规律等等。

矩阵不仅可以存储常见的十进制数，也经常存储二进制数。例如汉字的点阵，当笔划经过时为 1，不经过时为 0，这样由若干 1 和背景的 0 组成一个个字模，比如 16×16 点阵，当然这里的 0 和 1 并不是用通常的浮点数表示的，而是用二进制的“位”（bit）表示，8 个