计算机代数，数值计算，软件

(2)

91-207

计算机代数介绍(上)

A.M. Cohen J.H. Davenport A.J.P. Heck

Cohen, AM Dave JH

024.6

1. 引言

直到几年前，提到计算机和应用数学的结合，很多人也许只会想到数值计算. 数值计算是研究实数演算的学科；更确切地说，数值计算是寻找适当的有理数去逼近实际问题的实数解. 这类问题往往通过微分、积分或者其它类型的方程组以及适当的初边值条件来表达. 因为计算机不可能准确地表达实数，所以通过数值计算得到的结果是近似的. 应用数值计算的领域包括：物理学、气象学、天文学、地质学和工业生产.

今天，现代计算机 (工作站) 的计算速度和存储空间已提供了比数值计算更多的功能，计算速度和存储空间的指数型增长为符号计算软件的开发提供了技术上的保证. 科学家们已经在计算机上实现了以前只能用纸和笔才能完成的计算. 研究这类软件的数学理论、研制、开发和应用的学科称为计算机代数或者符号计算. 它提供了利用计算机进行准确计算的的工具，例如：表达式代换，函数的微分和积分，准确地解方程等等.

通俗地讲，计算机代数是研制，开发和维护符号软件并研究其数学理论的学科. 更准确地讲，计算机代数是致力于数学求解问题中准确计算自动化的学科 (见 [BC]).

一些多层次、多用途的商用计算机代数软件已经问世. 例如：Derive, 该软件包可在 PC 上做大部分大学一、二年级的数学运算. 本文涉及的软件，例如：Maple, Mathematica, Reduce 不仅可在 PC 上而且可以在工作站上运行.

现代计算机和软件技术还允许用户定义抽象的代数结构 (例如：环和代数) 并对其元素进行操作，具有这一功能的最通用的商业软件是 AXIOM. 毫无疑问，更多具有这类功能的软件在不久的将来会出现，甚至会很常用.

除了通用的计算机代数软件以外，用户自己写的、专用的软件包也可以在现代计算机的工作环境中运行，管理这类具有不同目的的软件正在成为一门专业技术，这正是建立阿姆斯特丹 CAN 职业培训中心的原因.

本文试图说明计算机代数在实际中的应用. 假定我们要解一组来自实际问题的方程，它的实系数已经被适当的有理数、可初始化的参数、或满足适当规则的符号所取代 (例如：$\sqrt{2}$ 可以看作满足 $(\sqrt{2})^2 = 2$ 的符号). 计算机代数可以帮助我们发现这组方程的准确解.

原题：An Overview of Computer Algebra. 译自：*Computer Algebra for Industry* Vol. 16, No. 1, 1994, pp. 1–25.

计算机代数的局限性也很明显，并不是所有的方程组都有准确解，因此计算机代数远不能取代数值计算．另一方面，计算机代数的应用并不局限于求方程组的准确解，计算机代数通常帮助我们把一组方程化为另一组比较容易解的方程．化简的方法包括：级数展开，变量代换等等．事实上，计算机代数软件是进行准确计算的工具箱，它大大提高我们进行数学计算的能力．它的图形功能使我们观察函数，它的数值计算功能使我们可以在必要时进行近似计算，它的数据搜索功能使我们可以对有关的数据进行比较和分析，所有这些功能似乎提供了一个适合于所有数学计算的计算机环境，它只把思维留给用户．

在本文中，我们将简要地描述计算机代数的内容、历史和发展方向，并举若干例子．

2. 示 例

使读者熟悉计算机代数的最好方法也许是通过一个例子说明怎样利用计算机代数来解决具体问题．我们从一个数学练习题开始．

假设我们观察到两架放置在走廊两侧墙壁上的梯子，一架梯子长 7 米，另一架长 5 米，两架梯子相交的高度是 2 米．试求走廊的宽度 (见图 1)．

如图案所示，我们对相关的距离设置变量．根据勾股定理和其它几何知识，我们得到下列方程组：

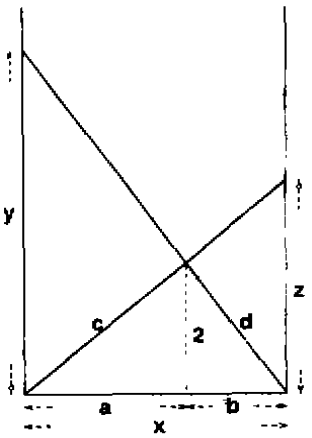

图 1．双梯问题

$$\{x^2+y^2=49,\ \ x^2+z^2=25,$$

$$x=a+b,\ 2x=az,\ 2x=by\}$$

我们将利用计算机代数软件 Maple 解这组方程，以下是 Maple 的输入格式：

```
eqns:  ={x^2+y^2-49,  x^2+z^2-25,
b-x+a,
a * z-2 * x, b * -2 * x }:
```

我们想求 x 的值．从用户的角度看，最简单的方法莫过于利用指令：

```
slos:=solve(sqns);
```

Maple 不是万能的，也许 Maple 不能求解，这可能是因为该方程组无解，或者求解对 Maple 来讲过于困难．但对于上术方程组， Maple 解得很漂亮．解答如下：

$$sols := \{b=0,\ a=0,\ y=7,\ x=0\},$$
$$\{b=0,\ a=0,\ y=7,\ x=0\},$$
$$\{b=0,\ a=0,\ y=-7,x=0\},$$
$$\{b=0,\ a=0,\ y=-7,x=0\},$$

$$\{x=(-\%1^2+49)^{(1/2)},\ y=\%1, b=2\frac{(-\%1^2+49)^{1/2}}{\%1}, z=2\frac{\%1}{\%1-2},$$
$$a=\frac{(-\%1^2+49)^{1/2}(\%1-2)}{\%1}\},$$

$$\{x=-(-\%1^2+49)^{(1/2)},\ y=\%1, b=-2\frac{(-\%1^2+49)^{1/2}}{\%1}, z=2\frac{\%1}{\%1-2},$$
$$a=-\frac{(-\%1^2+49)^{1/2}(\%1-2)}{\%1}\},$$

$$\%1 := \text{Root of}(\_z^4-24\_z^2-4\_z^3+96\_z-96)$$

我们得到 6 组不同的解. 在前 4 组解中 $x=0$，后两组解由表达式 1% 给出，这里 1% 表示一个一元四次多项式的根.

剩下的问题是哪一个解是问题的正确答案. 因为所有的变量都代表距离，解必须由正实数组成，这是可以验证的. 例如：利用指令

```
evalf(rhs(sols[1]))
```

计算第 5 个解的第一个分量的值，我们得到 3.95476890，这是 x 的一个近似值. 利用循环：

```
for i to 6 do ok:  =true;
  for j to 5 do
    if evalf(rhs(sols[i][j])) <=0 then ok:  =false fi
  od; if ok then print(i) fi
od;
```

我们可以发现哪个解是由正实数组成，答案是只有第 5 组解，它的近似值为：

$$\{x=3.954768790,\ a=2.585340328,\ y=5.775794648,$$
$$b=1.369428462,\ z=3.059379646\}$$

于是，问题的答案是 $x\approx 3.955$ 更准确地说，x 是在 3.955 的某个小邻域中满足下列条件的 实数.

$$x=\sqrt{-z^2+49},$$

其中 z 是

$$-24z^2-96+4z^3+96z+z^4=0$$

的根.

显然，利用手算解这个方程组将比较繁杂，计算机代数软件使我们把精力集中于做什么，从而把我们从繁琐的计算中解放出来，而且 Maple 给出了该方程组的准确解. 也许有人认为近似解 $x \approx 3.955$ 可以满足所有实际需要而且可以通过几何作图和相似理论得到，为什么要醉心于准确解？准确解的一个优点是我们可以通过它计算任意精度的近似解，而画高精度的几何图形却比较复杂. 当对前问题参数化，准确解的价值就会变得更为明显. 用参数 h,l 和 m 取代原问题中的长度 2, 7, 5, 则输入 Maple 的方程变为：

```
eqns:  = {x^2+y^2=1^2, x^2+z^2=m^2,
x=a+b, h * x=a * z, h * x =b * y }:
```

利用指令 Solve 解方程组，我们需要区别参量 l,m,n 和未知量. Maple 发现 6 组解，记为 sols. 指令：

```
evalf(subs(h − 2, I = 7, m = 5, {sols}));
```

给出原问题的解答. 类似地我们求得：

$$x = \sqrt{-z^2 + l^2}$$

其中 z 满足方程：

$$z^4 - 2hz^3 + (-l^2 + m^2)z^2 + (2hl^2 - 2hm^2)z - l^2h^2 + m^2h^2 = 0.$$

这个解答可以作图，例如：假定 $l = 7, m = 5$ 作 x 与 h 的关系的图形. 以上的解释说明：如果我们要对不同的 l,m,h 计算 x，准确计算只需要解一次方程组，其它运算量较小. 而数值计算则需要对每一组给定的 l,m,h，解方程组.

3. 计算机代数的优点

数学家近几年来的经验证明计算机代数具有下列优点：

1. 它使用户避免大量的繁琐运算；
2. 它使用户容易使用先进的数学技术 (例如：因式分解，符号积分， Gröbner 基等等)；
3. 它帮助研究者完成含有大量繁琐运算的证明；
4. 它帮助研究者通过大量例子进行实验，验证猜想；
5. 它使得一些古老的数学问题获得新生. 例如：大整数的素数判定和分解，该问题在编码中有重要作用；
6. 它促使研究者改进已知的算法和发明新的算法；
7. 从某种意义上说，它恢复了研究有效方法的理论需求 ——"对于哪一类问题我们可以通过算法求解？"

3.1 与手算比较

也许有的读者以为计算机代数只对于真正的工业应用才有价值，这个观点是完全错误的. 计算机代数的用途首先是避免繁琐的手算，进而导致深入的研究. 它时常给研究者灵感. 计算机代数首先是一种计算工具，第 2 节中已经提出令人信服的证据，以下是另一个例子. 利用 Maple 计算：

$$q = \frac{(1+x+x^2+x^6)^3}{(1-x-x^5+x^9)^2}$$

在 $x=0$ 处的泰勒级数. 如不指定展开的阶数，则指令 taylor(g,x) 给出：

$$1+5x+15x^2+32x^3+55x^4+83x^5+o(x^6).$$

除了计算迅速可靠以外，计算机代数软件还可以作图，计算更高阶的展开和取极限等.

例如，通过下列步骤可以计算 q 在 ∞ 点附近 12 阶的泰勒展开.

```
> subs (x = 1/y, q):
> normal (");
```

$$\frac{(y^6+y^5+y^4+1^3)}{y^9-y^8-y^4+1^2)}$$

```
> taylor(", y = 0, 12):
> subs (y = 1/x, ");
```

$$1+\frac{5}{x^4}+\frac{3}{x^5}+\frac{3}{x^6}+\frac{14}{x^8}+\frac{10}{x^9}+\frac{15}{x^{10}}+\frac{6}{x^{11}}+o(\frac{1}{x^{12}})$$

这里 ‘‘>’’ 代表输入行，指令 sub 把 q 中的 x 代换为 $1/y$.

3.2 容易应用先进的技术

利用计算机代数系统，我们可以把繁琐的计算交给计算机，而集中精力研究用什么样的计算解决问题. 计算机代数系统还提供诸如做数学实验等很多功能. 通用的计算机代数软件不仅有操作符号的能力，而且还实现了很多强有力的和复杂的算法.

作为第一个例子，我们利用 Maple 的指令 integrate(q,x) 计算 q 对 x 的不定积分. 这一指令调用包括 Risch 算法在内的很多子程序，其结果包括有理部分和超越部分；其

中有理部分为:

$$x+$$

$$\left(\frac{3713744548}{207949239}+\frac{161735065}{69316413}x+\frac{218423287}{69316413}x^{2}+\frac{338981446}{69316413}x^{3}+\frac{391876348}{69136413}x^{4}\right.$$

$$\left.-\frac{2384058994}{207949239}x^{5}-\frac{2732043862}{207949239}x^{6}-\frac{2981806564}{207949239}x^{7}-\frac{3385133998}{207949239}x^{8}\right)$$

$$/(1-x-x^{5}+x^{9}),$$

其中超越部分为:

$$\sum_{\_R=\%1}\_R\ \ln(\text{polynomial in }\_R\text{ of degree 7})-\frac{2032}{27}ln(x-1),$$

$$\%1:=$$

$$\text{RootOf}(33304908281499913685699 7\_z^{8}-250650272696325276330895 52\_z^{7}$$

$$+158918914418075435216744 7\_z^{6}-864356506566901260310756 8\_z^{5}$$

$$+101985185141309175522331 6\_z^{4}-405316726778881362753686\_z^{3}$$

$$-809263540492908269222257\_z^{2}-191617196012011753908 35\_z$$

$$-485323585932933265691)$$

用最近版本的 Maple, 如果指令 integrate 返回输入值, 则 q 对于 x 的不定积分不是初等函数. 如果让 Maple 计算下列积分

$$\int\frac{dx}{\sqrt{x^{2}+1}}dx,\quad\int\frac{xdx}{\sqrt{x^{3}+1}}$$

Maple 分别给出下列答案:

$$ln(x+\sqrt{x^{2}+1}),\quad\int\frac{xdx}{\sqrt{x^{3}+1}}$$

这并不意味着 Maple 不能计算后一个积分.

如果我们设置 Maple 中的 infolevel [integrate] 等于 2, 我们可以看到详细的计算过程如下:

```
> integrate ( x/ sqrt (x^2+1), x);
int/indef:   first-stage indefinite integration
int/algebraic:   algebraic integration
```

$$(x^{2}+1)^{1/2}$$

```
> integrate (x/sqrt (x^3+1), x);

int/indef:  first-stage indefinite integration
int/algebraic:  algebraic integration
int/algebraic/elliptic:  trying ellipitc integration _X _X^3+1 -1/2 _X
int/ellalg/elltype2:  considering expression, domain and
variable _X _X^3+1 -1/2 numeric {_X}
int/ellalg/ellrad:  ellrad entered with _R0 -_X -_X^3-1 numeric _X
newvartable radtable
int/ellalg/ellpolyt:  considering expression, domain and
variable _X^3+1 numeric _X
int/algebraic/elliptic:
the elementary part, the elliptic part, the non-elliptic part
[0, [[_X, _X^3+1, 1/2, [], 1]], 0]
int/algebraic/elliptic:  using Jacobi notation of elliptic
integrals _X _X^3+1 1/2 _X
int/ellalg/ellindef:  indefinite elliptic integration _X _X^3+1 1/2 _X
int/ellalg/ellquo:  apply polyonmial division to
integrand → R1(x) +R2(x)/y 0 _X
int/ellalg/ellquo:  polynomial division step complete 0 _X 0
int/ellalg/hermitel:
partial fraction expansion and Hermite reduction--P1 and P2/Q _X 0
int/ellalg/reduceI: reduction of I_K to basic
functions I _0, I _1 and I _2
array(sparse, 0 .  .  3, [(0)=0, (1)=1]) 1
array(0 .  .  4, [(0)=1, (1)=0, (2)=0, (3)=1, (4)=0]) _X _R0^(1/2)
int/ellalg/reduceI: after reduction of I_k to basic functions 0
array(sparse, 0 .  .  3, [(0)=0, (1)=1]) 1
array (0 ..  4, [(0)=1, (1)=0, (2)=0, (3)=1, (4)=0]) _X _R0^(1/2)
int/ellalg/hermitel:  Hermite reduction completed 0
array(0 ..  2, [(0)=0, (1)=1, (2)=0]) array (1 ..  1, 1 ..  3, [])
int/ellalg/ellindef:  result of indefinite elliptic integration
Int(_X/(_X^3+1)^(1/2), _X)
```

$$\int \frac{X}{(X^3+1)^{1/2}}dx$$

该过程表明 Maple 判断第二个积分是椭圆积分，不存在初等函数解. 系统尽可能把该积分利用 Legendre's 积分表示.

另一个优美而且更为初等的算法是计算格的约化基底. 格是向量空间一组基底的所有整系数线性组合的集合. 这组向量空间的基底的坐标通常是整数，但有时也可能不是整数. 请注意基底的选择不是唯一的. 所谓约化基很像正交基，但要求所有的坐标都是正整数，而且约化基的向量长度都比较小. 详见 [LLL], 以发明者 Lenstra 、 Lenstra, Lovász 命名的 LLL 算法以 n 维空间中的一组整系数向量为输入，以输入向量所张成的格的一组约化基作为它的输出，下面的例子说明当 $n=3$ 时 Maple 怎样运行该算法. 指令 readlib(latlice) 使 Maple 装入 LLL 算法，

```
> readlib(lattice):
> S: [1, 2, 4], [3, 9, 27], [4, 16, 64]:
> lattice(S);
                [-2, 1, 1], [0, -1, 3], [1, 3, 1]
```

显然，输出的基底比输入的基底要 "好".LLL 算法可以用于分解整系数多项式，这里它给出较好的最坏复杂度，但经验证明经典算法的平均复杂度比应用 LLL 算法的平均复杂度低.

LLL 的另一个应用是关于求多项式在某个区间中的实根的逆问题. 更为精确地说，假定给定下参数：

r—— 一个近似根；

n—— 所求一元多项式次数的上界；

ε—— 给出所要求的精度 $f(\varepsilon)$, 其中 f 由 Maple 决定.

Maple 的函数 minpoly 利用 LLL 方法求一个非零的次数最小的 (不大于 n) 的整系数多项式，使得该多项式在 $(r-f(\varepsilon),\ r+f(\varepsilon))$ 内有一个根.

```
> minpoly (x, 4, 10^7);
```

$$16 - 20\_X + 47\_X^2 + \_X^3 - 3\_X^4$$

```
> minpoly(x, 4, 19^9);
```

$$36 - 88\_X + 103\_X^2 - 21\_X^3$$

```
> fsolve(");
                              3.954768786
```

请注意我们用了第二节梯子问题中 x 的近似值，但所发现的极小多项式与梯子问题中以 x 为根的多项式不同，最后一个指令检查所求得的多项式的确有一个足够接近 x 的根.

3.3 帮助完成证明

四色定理的证明显示了先进的计算机有更大的潜力. 为了证明所有的地图可由四种颜色染成，使得任意两个相邻的国家的颜色不同，需要验证大约 2000 多种地图满足某些性质. 虽然艰苦的思维把四色问题化为上述验证问题，但是最后的一步只能在计算机

上完成.

下面的例子更接近于实用 (见 (CGL)), 证明一个李群的某种有限子群的存在性有关的问题可以归结为在有 1831 个元素的有限域上解一个有 240 个未知数的线性方程. 利用计算机很容易证明该方程组存在唯一解, 而用手算几乎是不可能的.

3.4 实验工具

计算机代数系统有进行数学实验的功能, 例如: 数值计算和作图.

很多计算机代数系统可以把它们程序自动翻译为高级程序语言. 例如 FORTRAN, PASCAL 或 C, 也可以提供更为复杂的符号 —— 数值接口. 例如: Maple 和 Macsyma 的 Macrofort, Reduce 和 Macsyna 的 GENTRAN, Rednce 的 IRENA, Mathematica 的 INTERCALL. 我们还可以利用计算机代数软件写高级语言的程序. 这种方法比直接用高级语言写程序写得快而且很容易设置参数, 高级语言程序的输出也可以很容易地读入计算机代数系统以便进行进一步的数学研究. 这些优点部分是由于符号编程的环境更接近于数学, 从而比高级语言更为容易; 另一部分是因为计算机代数软件具有人机对话的功能.

近年来, 专门的计算工具也越来越多. 例如: 康乃尔大学开发的 DSTOOL, 不仅能对动力系统的研究提供计算, 而且可以通过图形显示它们的几何性质. 在 Aèrospatiale 开发的 JAMES, 可对运动进行符号 —— 数值分析, 利用它可以对轨道卫星的空间站进行分析.

4. 计算机代码的局限性

在利用计算机代数软件时, 会比预期更早遇到在通常的时间和空间的限制. 其原因部分在于运用的是准确运算和符号表达式, 部分在于缺乏有效的算法. 另一方面, 懒惰也很容易导致这些限制, 用户经常让计算运行直到它占用了所有的计算资源, 而不是首先考虑有效的计算方法.

使用计算机代数软件时常遇到的另一困难是没有封闭形式解的方程的普遍存在. 于是, 计算机代数似乎局限于决定给的方程组是否存在封闭解, 如果存在则求解. 但是, 在随后的近似分析中, 计算机代数系统也还能通过结合所有的计算方法提供合适的数学工具来帮助用户.

最后, 一个单一的指令不总能有效地导致正确的结果. 例如: taylor$(q, x_x = infinity, 12)$ 并不给出 §3.1 节中所求的结果 (见 §4.3 节的讨论).

本节我们将详细讨论上述问题.

4.1 中间表达式膨胀

在运用计算机代数软件时, 一个常见的问题是中间表达式膨胀. 这是因为我们试图

用计算机代数系统解决类似于手算但规模更大的问题. 但我们没有仔细分析中间表达式的规模与输入表达式规模的依赖关系. 它们之间的关系可能是线性的, 也可能是指数的, 或者是双指数的. 假设我们用 "无分式" 除法, 或称伪除法, 计算两个整系数一元多项式的最大公因子. 例如: 考虑 $A = x^2 - 2x$, $B = 2x + 1$. 在整数环内, 我们不能用 B 除 A, 但可以有 B 除 $4A$. 其余式为 13, 于是 A 和 B 互素. 但是考虑两个系数为一位数的 6 次多项式:

$$A = -7x^6 - 5x^5 - 8x^4 + 4x^3 + 3x^2 + 8x - 3;\ B = 5x^6 - 4x^5 - 2x^4 + 3x^3 - 10x^2 - 9x + 4.$$

其结果颇令人吃惊.

利用 B 伪除 A, 我们得到伪除式:

$$C = -53x^5 - 54x^4 + 41x^3 - 55x^2 - 23x + 13;$$

用 C 伪除 B 得到:

$$D = 31275x^4 - 25910x^3 - 7675x^2 - 10750x + 4970;$$

用 C 伪除 D 得到:

$$E = -51957230300x^3 - 95117304624x^2 - 47176101625x + 27934170725;$$

用 E 伪除 D 得到

$$F = 31362393463136423585859375x^2 + 22021906977910511306085937 5x - 10728692414741589825585937 5;$$

用 F 伪除 E 得到

$$G = -233884917231758694117375545526990779754650521326884765625000000x + 774700330477497432812839078891679179323402691078146972656250000;$$

最后一步, 我们用一次多项式 G 伪除 F, 得到一个 150 位的整数, 从而证明 A 和 B 互素. 什么地方错了呢? 事实上, 没有任何错误, 这仅仅是因为辗转伪除法是指数型的, 即系数的位数是输入多项式的次数的指数函数. 自从六十年代后期, 数学家研究出不少方法解决这一问题. 他们要用到比欧基里得方法更为先进的知识 (见 §7.3).

4.2 难于管理的输出

在人机对话的过程中, 无辜的用户可能发现系统返回难于处理的大型输出. 例如一个一般 (generic)4 次多项式的完全解集, 其公式会超出整个屏幕. 大多数计算机代数软件都可避免打印输出. 例如: 在 Maple 中可以用 ":" 表示一个指令的终结. 它与 ";" 的区别以 ":" 为终止符的指令不在屏幕上打印输出. 当用户知道如何继续计算或知道如何

简洁地表示结果时，这个终止符很有帮助．然而，怎样友好地表示大型公式的问题仍未解决．打印全部的公式不是正确的答案，但系统本身很难决定什么是重要的．

虽然有很多例子说明这一现象，我们在这里仅仅给出一个例子．

表达式

$$x^{17}-34x^{16}+544x^{15}-5440x^{14}+38080x^{13}-198016x^{12}+792064x^{11}-2489344x^{10}$$
$$+6223360x^{9}-12446720x^{8}+19914752x^{7}-25346048x^{6}+25346048x^{5}-19496960x^{4}$$
$$+11141119x^{3}-4456451x^{2}+1114109x-131073$$

看上去很复杂，令它为 f．利用指令 factor(f) 对 f 因式分解，得到的是 f 本身．但 f 的前几项使我们联想到 $(x-2)^{17}$．现在，我们利用指令：

```
> factor (f-(x-2)^17);
```

得到

$$-(x+1)^3$$

于是 $f=(x-2)^{17}-(x+1)^3$，这是 f 的一个简洁的表达式． Maple 中化简表达式的工具有 Simplify, normal, combine 和 compoly.

4.3 寻找适当的工具

在用计算机代数软件时，我们所需的程序常常要通过递归查询找到．假如我们要计算某个函数的泰勒展开，我们也许会寻找含有 Taylor 的命令． AXIOM 系统有功能齐备的浏览器．通常我们利用索引，即想一个适当的名字，然后寻找与这个名字有关的信息．事实上，大部分计算机代数系统都具有引索查询能力．但贮存的信息和我们所想的名字很可能不同．由于函数名无法对应，我们很可能找不到所需要的函数．回到 §3.1 中的例子，指令 asympt($q,x,30$) 也许也是计算 q 的 30 阶近似的方法，而实际却不是这样．同样，计算机代数系统的很多功能也是较难发现的．

通用的计算机系统含有很多标准的和不太标准的算法．但是更多的算法还没有在计算机代数系统里实现．据我们所知，路径积分和李代数的 Buchberger 算法就还没有在通用软件中实现．

于是，我们可能买了一个计算机代数软件，但发现它并不是最适合我们需要的，因为它并不提供要求解的问题的基本工具，或者不如专用软件有效．下面的例子说明符号软件 Form(见 [Verm]) 比 Maple 更适合于非交换代数的计算．在下面的例子中我们对四元数体中的元素进行计算．该体的基元素为 i,j,k，并满足 $ij=k$(以及对 i,j,k 的轮换),$ji=-ij, ki=-ki, kj=-jk$，和 $i^2=g^2=k^2=-1$.

```
FORM version 2.0 2-jan-1992

#procedure simplify
repeat;
```

```
id i*j = k;
id j*k = i;
id k*i = j;
id j*i = -k;
id k*j = -i;
id i*k = -j;
id i*i = -1;
id j*j = -1;
id k*k = -1;
endrepeat;
#endprocedure
*nwrite statistics;
Functions i,j,k;
Symbols  t,u;
Local  F=(u+3*i-k)^3 * (-1+t*j);
#call simplify
bracket i,j,k;
print;
.end

Time  =      0.09  sec   Generated terms =     54
               F          Terms in output =     12
                          Bytes used      =    218
  F =
      + i * (30 + 3*t*u^2 - 10*t - 9*u^2)
      + j * ( - 30*t*u + t*u^3)
      k * ( -10 + 9*t*u^2 - 30*t + 3*u^2)
      + 30*u - u^3;
canb.can.nl> maple
    |\^/|      MAPLE V
 _|\|   |/|_.  Copyright (c) 1981-1990 by the University of Waterloo.
 \  MAPLE  /   All rights reserved.  MAPLE is a registered trademark of
 <____ ____>   Waterloo Maple Software.
      |        Type ? for help.

> `&*` := proc(a,b) local answer, coa, cob, terma, termb;
> # id is the identity.
> # The procedure treats all variables as constants.
> # To provide a simplified answer we use collect.
> #
```

```
> if nargs>2 then
>   RETURN( `&*`(`&*`(a,b), args[3..nargs]) )
> elif nargs=0 then
>   RETURN(`1d`);
> elif nargs=1 then
>   RETURN(a)
> fi;
> if type(a,`+`) then
>   answer := map( `&*`, a, b )
> elif type(b,`+`) then
>   answer := map( proc(c, d) d &* c end, b, a )
> elif a=1d then
>   answer := b
> elif b=1d then
>   answer := a
> elif not has( a, {1d,1,j,k} ) or not has( b, {id,1,j,k} ) then
>   answer := a*b
> else
>   coa := coeffs( a, {1d,1,j,k}, terma );
>   cob  = coeffs( b, {1d,1,j,k}, termb );
>   answer := coa*cob*`&*`(terma,termb);
> fi;
> collect( answer, {id,1,j,k} );
> end:
```

Having defined the usual procedure '&^' := proc(x,n) for computing powers, we may continue with.

```
> `&*`(i,i) := -1d: `&*`(1,j) := k:   `&*`(1,k) := -j:
> `&*`(j,1) := -k:  `&*`(j,j) := -id: `&*`(j,k) := 1:
> `&*`(k,i) := j:   `&*`(k,j) := -i:  `&*`(k,k) := -id:
>
> time(): (u+3*1-k)&^3 &* (-1d+t*j); time():
```

$$(30u - u^3)\,1d + (-9u^2 + 30 - (-3u^2 + 10)\,t)\,i + (3u^2 - 10 + (9u^2 - 30)\,t)\,k + (-30ut + u^3 t)\,j$$

```
> cpu_time := ("-""")*second;
```

$$cpu\_time := 1.000\ second$$

4.4 错误

显然，软件中所含的错误是一个重要的问题．大多数通用系统的基本运算，例如大整数加法和乘法，都是非常可靠的，但关于复杂算法的软件很可能有错误． Maple 系统绕过这个难点的方法是：保证少量基本运算的正确性，并把它们贮在软件的核心 (kernel)

部分. 普通用户无法阅读核心内的程序，而所有其它的程序都由类似于 Pascal 的程序组成，并且可以阅读. 如果用户觉得某个矩阵的迹计算的不对，或者可以用更有效的方法计算，他可以修改程序以适应自己的需要，或者写自己所需要的程序.

然而，在软件的核心中，我们会遇到怎样化简表达式的问题. 例如，如果输入 3*8*7，则大部分系统会输出 168. 这看上去很合理，因为我们希望计算它的值. 假如一个系统不计算 3*8*7 的值，我们很快会被巨大数量的表达式搞得晕头转向. 当然，有的系统也有程序写出： 168 的分解，或者把它写成字符串 3*7*8.

设计中的主要问题是把一个表达式化简到什么程度. 例如在 Maple 中把 0*f(1) 简化为 0, 从效率的角度看，这是必要的. 但这也许是不对的，因为 f(1) 也许未定义或者是 ∞. 在利用化简规则 $0^i = 0$ 时，对下面的数组 a 产生了以下结果：

```
> sum(a[i] * x^i, i=0 ..  n)
```

$$\sum_{i=0}^{n} a[i]x^i$$

```
> subs(x=0,");
```

$$\sum_{i=0}^{n} 0$$

```
> expand(");
```

$$0$$

另一个自动化简的规则产生了下述灾难性的结果：

```
> subs(y=-z, sqrt(y^2))=sqrt((-z)^2);
```

$$-z=z$$

在这个例子中， sqrt 是取平方根的算子，但自动化简 $\sqrt{2^2}$ 时，符号出了错.

上述例子说明了运算效率和数学上的正确性之间的矛盾. 计算机代数软件的用户必须了解绝大多数（如果不是全部） 计算机代数软件中在进行表达式化简时，并不是 100% 的正确. 见 [Stout] 中关于这一问题的讨论.

4.5 空间局限性

最常见的困难是运行某些算法需要巨大的存贮空间和计算时间，因此计算机代数软件对它能处理的问题的规模有较苛刻的限制. 中间表达式膨胀的例子说明任意大整数的运算可能导致这样的限制. 如果可能的话，可以对整数进行模素数的运算，因为大整数问题在适当的有限域中并不出现. 但中间表达式膨胀的问题也可能出现在其它情形，例如在多元多项式的运算中.

Buchberger 算法 (见 §7.2) 适用于任何可计算域. 但是用这一方法解较小规模的方程组，都会产生巨大的中间多项式. 利用这一方法解方程组失败的原因往往是存贮空间不够，而不是运行时间太长.

解决问题所需空间和时间是问题参数的函数，其增长之快使得人们不能寄希望仅

仅靠硬件的改进来提高算法效率. 本世纪计算能力的迅速提高可以用较为模糊的概念 MIPS 来刻化，这里 MIPS 是 Million Instructions Per Second 的缩写. 当然，一个指令可由很多种方法来定义，于 MIPS 是一种比较粗糙的度量. MIPS 的一个用途是用来比较不同机器的能力. 设定八十年代 VAX-11/780 为标准单位，即 1 MIPS. 人类大约每秒处理 1.5 个指令，则我们得到下面的表：

Year	Machine	no. MIPS
1920	human	.0000015
1960	IBM360	.01
1970	IBM370	.1
1980	VAX-11/780	1
1989	Sparc1	17
1991	indigo	30
1992	crimson	85

在 [HP] 的表格说明：一次世界大战后，中型机器的 MIPS 大约每 10 年增加 10 倍 (根据 BYTE, 1992 2 月号， p.137–138, 由 INTEL 制造的 P_5 微处理器希望突破 100 MIPS 的限制，本世纪末可能会出现 1000 MIPS 的处理器).

我们的观点是即使这样改进将在近几十年内持续下去，仅仅是硬件的改进也不可能对解多元多项式方程组带来实质上的提高. 为了说明我们的观点：假定某个算法处理一类依赖于维数 d 的问题，假定

—— 该算法关于 d 维问题需要 $f(d)$ 条指令.

—— 当 $d = 4$ 时，算法可以在较为合理的时间内解决问题，

—— 计算机的运算速度每 10 年提高 10 倍.

我们的问题是：多长时间以后我们才能处理 $d + 1 = 5$ 维的问题. 如果我们在 t 年后将能够处理 $f(d + 1)$, 则 $f(d) \geq f(d + 1)10^{-t/10}$, 于是，我们需要等待的时间是

$$t = 10(\log_{10} f(d + 1) - \log_{10} f(d))$$

下面的表格说明当 $f(d)$ 为线性 ($f \sim 10d$), 指数型 ($f \sim 10^d$), 和双指数型 ($f \sim 10^{10^d}$) 时 t 的值.

$f(d)$	years t needed until $d = 5$ is feasible
$10d$	1
10^d	10
10^{10^d}	900,000

于是，如果 Buchberger 方法是双指数型的，寄希望通过改进硬件提高该方法的效率是不现实的. 更广义地讲，结论是需要更好的算法.

5 计算机代数简史

由于计算机代数年轻的形象，我们略去它的古老的历史(尽管代数和算法两个词都有很长的历史).Ada Lovelace 女士扮演了预言家的角色，她于 1842 年在为 Charles Babbage 的一位客人作翻译时提到利用解析机器来操作符号演算的可能性. 从此以后，计算机代数的里程碑大致如下：

1842 年 Ada 预言计算机代数的存在，
1953 年 第一个计算机程序问世，
60 年代 专用程序问世 ——Reluace-1, CAMAL, Schoonschip,
1970 年 算法方面的突破性进展：积分，因式分解，等等.
70 年代 通用计算机代数系统问世，包括 Macsyma, Reduce-2,
1980 年 小型计算机兴起： Vaxima,
1983 年 IBM PC:Mumath,
1985 年 商业化： Maple, Derive, Mathematica,
1990 年 专业化软件： GAP. Macaulay, Pari 等,
1991 年 HP-95. 计算机代数袖珍计算器，装有 Derive,
1992 年 AXIOM: 一个根据数学结构设计的计算机代数软件.

1953 年，在美国的两篇关于形式微分的论文(分别由 J.F.Nolan 和 H.G.Kahri-manian 撰写). 导致了一些计算机代数的程序，与此同时，英国的 Hazelgrore 利用 EDSAC-1 进行了群论中的 Toss-.Coxtex 计算. 在六十年代早期，用于表处理的计算机语言 LISP 在美国开发成功，LISP 与 FORTRAN 和 ALGOL60 不同，后两种语言比 LISP 开发稍早. 尽管后来的计算机代数系统大部分由 C 语言写成，LISP 在计算机代数软件中起了重要作用. 由 James Slagle 写的第一个符号积分程序，以及稍后由 Joel Moses 写的符号积分程序都是用 LISP 写成的. 这些程序和 Nilliam Martin 的努力是 Macsyma 项目的前奏. Macsyma 是第一个基于 LISP 的通用计算机代数软件. Macsyma 的第一版于 1971 年问世，它提供了计算极限和解方程的功能. 在八十年代，麻省理工学院把 Macsyma 转让给 Symbolic 公司，该公司专门开发专家系统和 LISP 机器. 最近，Macsyma 又被转让给专业公司 Macsyma Inc.Macsyma 系统的命运代表了过去三十年来符号计算软件的三个阶段：六十年代的专门化程序，七十年代的通用程序和八十年代的商业化.

在六十年代，还有一些其它的专业化程序. 它们通常致力于实现初等函数的运算. 例如：由 IBM 公司 Jean Sammet 和 Robert Tobey 等开的 FORMAC, 由剑桥大学 David Barton, Steve Bourne 和 John Fitch 开发的 CAMAL,CAMAL 来自于 CAMbridge ALgebra, 它致力于天文学和相对论的计算. M.Veltman 开发的 Schoonschip, 它为高能物理中的数学计算提供了必要的工具.

类似的原因使 A.C.Hearn 用 LISP 开发了计算机代数系统 Reduce, 后来 Reduce 成为一个广泛应用的通用软件.

自 PC 问世以来, David Stoutemyer 把计算机代数系统实现在这些小型机上. 鉴于 PC 的限制, 这些计算机代数系统的功能是惊人的. 这一软件先命名为 MuMATH, 而后以 Derive 为名字投入市场. 这一系统甚至已经装入 HP 公司的袖珍计算器 HP-95 并能够处理大约 80% 的大学数学计算.

一个广泛应用的通用软件是由 C 语言写成的 Maple. 它由加拿大 Waterloo 大学 Keith Geddes 和 Gaston Gonnet 在八十年代初发起的一个科研项目演变而成, 该项目的目的本来是为用户提供应用计算机代数的工具. 与其他计算机代数系统比较, Maple 是效率较高的; 这是因为它的设计特点: 系统的核心由尽可能小的关于最基本运算程序组成. 这些运算包括: 指令翻译, 整数, 有理数和多项式运算, 空间管理. 该软件的其他部分是由 Maple 语言写成的软件包. 这些数学软件包的管理很灵活, 用户可以加入. 改变和删除函数. 目前, 已有大量的专用软件包.

最引人注目的商业系统是由 Stephen Wolfram 组织编写的 Mathematica, 它由 C 语言写成. Stephen Wolfram 早期曾编写 8MP 系统. Mathematica 有很新颖的特点. 例如: "代数发动机" 和用户接口有本质的区别; 它综合了符号计算, 数值计算和作图的功能; 它具有结构清晰的用户编程语言; 在某些机器上 (如 Macintosh, NeXT,PC MS-DOS Windows) 有 "笔记本" 的功能, 利用该功能可以编译数学公式, Mathematica 程序, 预先设计好的计算和图形等. 与其他系统相比, Mathematica 成功地吸引了很多学术界以外的注意, 这一成功完全不在于那个高度专业化的商业组织, 也得到了大量用户的支持.

另一个利用 LISP 编写的软件是 SCRATCHPAD, 它由 IBM 公司的 Jenks 和 Griem-er 组织编写. 它综合了重写规则和幂级数动态赋值的想法. SCRATCHPAD 导致了 SCRATCHPAD-II 后者能够系统地处理诸多类型的代数数据. SCRATCHPAD-II 型的下一代是由 NAG Ltd 开发的 AXIOM. 它的新功能包括: 图形功能和关于 SCRATCHPAD-II 代数功能的信息系统. 目前, AXIOM 只能在为数不多的计算机上运行 (主要在 IBM RS6000), 但这一情形会在不久的将来变化.

到目前为止, 我们提供了通用系统. 还有很多专用软件, 它们处理某一类数学对象. 一个著名系统是由 John Canon 在悉尼开发的 CAYLEY 系统, 它主要用于群论和组合学. 在近几年来, 我们注意很多学者编写软件包, 并通过文件和 UNIX 系统互相交流.

由于篇幅限制, 很多方面我们还未触及, 例如并行计算, 感兴趣的读者可以参考 [DDF].

(未完待续)

(李子明 译　戴宗铎 校)