第七章　计算机代数

Computational Algebra

Symbolic Manipulation

Machine Algebra

……

§7．1 引言

数值计算系统

Fortran 语言

例：

```
……
X=1
Y=2
Z=X+Y
……
```

非数值计算系统

Mathematica 语言

例： 勒让德多项式定义

(1) LegendreP[*n*,*x*] 给出勒让德多项式 $P_n(x)$.

(2) LegendreP[*n*, *m*, *x*] 给出伴随勒让德多项式

$$P_n^m(x)=(-1)^m\left(1-x^2\right)^{m/2}\frac{d^m}{dx^m}P_n(x)$$

也可以由用户采用递推公式定义：

```
P[n_, x_]:=
    module[{temp}, temp=0;
            If [n==0, temp=1 ,
                If [n==1, temp=x ,
                    If [n > 1,
            temp=((2n-1) x P[n-1, x]-(n-1) P[n-2, x])/n , Null
                        ], Null    ], Null
               ]; temp
            ]
```

递推公式

$$P_0(x)=1,\quad P_1(x)=x,\quad P_n(x)=[(2n-1)xP_{n-1}(x)-(n-1)P_{n-2}(x)]/n$$

表面上来看，数值计算语言应当与计算机代数语言是本质上迥然不同的两种语言。其实，两者在本质上是完全一致的。这是因为目前我们使用的计算机仍然是一种二进制的数字计算处理机。文字、字符或符号都只能通过二进制编码才能用计算机进行处理。

由于这种本质联系，所有的数值算法语言经过改造加工以后，都可以发展为计算机代数语言，或者说可以具有非数值处理功能。

所谓计算机代数处理系统实际上是指硬件和软件的综合。

常用的计算机代数系统：

1. MACSYMA。它是用 LISP 语言的一种功能很强的方言 Franz Lisp 写成的。它是一个通用的计算机代数系统。

(2) REDUCE。它是由赫恩(A.C. Hearn)设计的。该语言是用 SLISP (Standard LISP)写成的,通用的代数处理系统，具有相当广泛的基本代数处理功能，并能处理高能物理的计算问题。

(3) Mathematica。该系统是美国 Wolfram 公司开发的一个功能强大的计算机通用数学系统。其基本系统主要是用 C 语言开发的。它是当前运用十分广泛的符号代数处理系统。

(4)Maple. 这是一个商业产品。其优点是使用图形用户界面并支持一些复杂运算，如：因式分解，积分或求和。缺点是 Maple 不适用于处理大量数据。

(5) GiNaC: 这是用 C++的符号计算库。它的主要特征是具备以面向对象的方式实现用户自己的算法的能力。它能处理大量数据，在基准测试下，其运算速度可与下面的 FORM 相当。

(6)SCHOONSCHIP。这是很著名的粒子物理研究用的计算机代数系统。它也能做一般的代数运算，是目前为止运行速度最快的系统。该程序是用 CDC 型 60 位计算机和 6800 系列计算机的汇编语言写成的，因而大大限制了它适用的机型。

(7)FORM:　优点是运算速度高和具有处理大量数据的能力。它被广泛运用于高能物理和涉及大型中间表达式的程序。人们普遍认为它是 SCHOONSCHIP 系统的后继程序

计算机代数系统的发展历史：

- 二十世纪六十年代最早的计算机代数系统几乎完全是基于 LISP 表处理语言.它是用来处理表链的。它对于早期符号计算程序的重要性，就好比同一时期处理数值计算的程序 FORTRAN 系统。在这个阶段，REDUCE 程序对高能物理已经表现出一些特殊的用途。

- SCHOONSCHIP 是 M. Veltman 用汇编语言写的，专门应用于粒子物理领域。汇编代码的应用导致了难以置信的高速计算程序（相对于最初的解释代码），从而使计算更复杂的高能物理散射过程成为可能。由于人们逐渐认识到这个程序的重要性，因而，1998 年 M.Veltman 因此获得了诺贝尔物理奖。

- 同时值得一提的是基于 Franz LISP 的 MACSYMA 系统，它引发了算法的重要发展。

- 从 1980 年以来,新的计算机代数系统开始采用 C 语言编写。这样的系统与解释语言 LISP 相比，能够更好的利用计算机资源，并能保持程序的可移植性，而这正是解释语言所做不到的。

- 这个时期还出现了最早的商业计算机代数系统，其中 Mathematica 和 Maple 最为著名。另外，少量的专用程序也出现了，J.Vermaseren 编写的 FORM 就是一个用于粒子物理研究的程序。它是可移植的，并认为是 SCHOONSCHIP 系统的后继程序。

- 近几年，有关大型程序可维护性的问题变得越来越重要。全部的设计范例都由过程设计变到了面向对象设计。反映在编程语言上从 C 变到 C++。这样 GiNaC 库随之发展起来，它支持 C++环境下的符号计算。

§7.2 粒子物理研究中计算机代数的应用

- 利用计算机代数系统使微扰计算步骤自动化。
- 粒子物理是应用计算机代数的一个重要领域，它充分发挥了计算机代数系统的潜力。
  SCHOONSCHIP, REDUCE, Mathematica,
  FORM 和 GiNaC
- 采用计算机代数系统来计算的复杂量子场论问题可以分为两类：

(1)第一类问题是仅仅只需要系统的基本支持的计算。

(2)复杂的计算方法。既可以采用标准化的非局部操作也可以采用针对特定问题的计算算法，如采用由用户发展起来的专用算法。

有关粒子物理研究中微扰计算的计算机代数算法：

一、图形产生

Thomas Hahn,　FeynArts

以 FeynArts 程序包为例：使用时，首先加载该程序包：<<FeynArts.m （或 init.m 文件）

一旦用户给定初、末态粒子，微扰计算的阶数，适当的模型，利用 FeynArts 就可以得到非零贡献的费曼图。用如下函数的操作指令就可以得到费曼图.

（1） $CreateTopologies[l,e]$ 产生 l 圈 e 个外腿的拓扑。

（2） $InsertFields[top,ext,Model->\{\mathrm{mod}1,\mathrm{mod}2,...\}]$ 将 $\mathrm{mod}1,\mathrm{mod}2$,...模型中的场加到外腿数为 ext 的拓扑 top 中去。

（3） $Paint[\exp r]$ 在屏幕上画出 InsertFields 输出 expr 的费曼图形。

二、费曼幅度和指标收缩计算

FeynArts 程序包中，采用函数

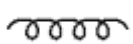
CreateFeynAmp[expr]

将 InsertFields 函数输出 expr 的一组费曼图产生对应的解析幅度。

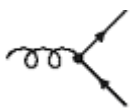
图形与幅度的转换过程是按照费曼图图形技术中所对应的规则进行的：

外线对应自由波函数，内线对应着传播函数，顶点对应相互作用顶点。

例：协变规范中的胶子传播函数的规则和夸克-胶子作用顶点的规则：



$$= \frac{-i}{k^2}\left(g_{\mu\nu} - (1-\xi)\frac{k_\mu k_\nu}{k^2} \right)\delta_{ab}$$



$$= ig\gamma^\mu T^a_{ij}$$

由上面的费曼规则得到的不变振幅的表达式中将包含了重复指标求和。

例如:洛仑兹指标的收缩可以通过不断运用以下几个规则来实现：

$$g_{\mu\nu}g^{\nu\rho}=g_{\mu}^{\rho}\ ,\quad g_{\mu\nu}p^{\nu}=p_{\mu}\ ,\quad g_{\mu\nu}\gamma^{\mu}=\gamma_{\mu}\ ,\quad g_{\mu}^{\mu}=D\ ,$$

$$p_{\mu}q^{\mu}=pq\ ,\quad p_{\mu}\gamma^{\mu}=\not{p}\ ,\quad \gamma_{\mu}\gamma^{\mu}=D$$

如果要对一串狄拉克（Dirac）矩阵乘积的重复指标收缩，例如$\gamma_{\mu}\gamma_{\nu}\gamma^{\mu}$，我们可以先用狄拉克矩阵的反对易关系$\{\gamma_{\mu},\gamma_{\nu}\}=2g_{\mu\nu}$，使两个狄拉克矩阵用同一个指标，然后再用上面公式化简。

在振幅平方的计算中，一般有狄拉克矩阵的求阵迹计算。由于结果是相对论性不变量，所以人们很少有必要去研究公式中各个矢量的分量，而只研究这些矢量的标积。这样计算就没有原来那么复杂了。原则上这个计算方法是很明确的：

- 奇数个狄拉克矩阵的阵迹是零；
- Tr1=4；
- 偶数(2n)个 γ 矩阵的乘积的阵迹计算可以用公式：

$$\mathrm{Tr}\,\gamma_{\mu_1}\gamma_{\mu_2}\cdots\gamma_{\mu_{2n}} = g_{\mu_1\mu_2}\mathrm{Tr}\,\gamma_{\mu_3}\cdots\gamma_{\mu_{2n}} - g_{\mu_1\mu_3}\mathrm{Tr}\,\gamma_{\mu_2}\gamma_{\mu_4}\cdots\gamma_{\mu_{2n}} + \cdots + g_{\mu_1\mu_{2n}}\mathrm{Tr}\,\gamma_{\mu_2}\cdots\gamma_{\mu_{2n-1}}.$$

将(2n)个γ矩阵的乘积的阵迹计算转变为对(2n-2)的γ矩阵的乘积求阵迹。理论上反复运用上式，(2n)个狄拉克矩阵的阵迹会产生(2n-1)(2n-3)…3·1=(2n-1)!!项。这个数目随n增加而指数增长。如果在指标$\mu_1,\ldots,\mu_{2n}$之间没有别的关系，则这实际上就是(2n)个γ矩阵的乘积的阵迹最后结果的项数。

FORM 程序是处理费曼幅度化简、洛仑兹指标收缩和γ矩阵阵迹的最好工具之一。运用′t Hooft－Veltman 方案对狄拉克代数的计算在 TRACER 程序中得以实现。

三、单圈积分函数计算和 Passarino－Veltman 化简

Passarino 和 Veltman 已经首先给出了一圈标量和张量积分函数系统的算法。其计算的基本思想就是将张量积分计算化解为标量积分函数的计算，而标量积分函数最终化为 Spence 函数的计算。

例：介绍张量圈积分化简到一个标量圈积分的集合。以下面的三点积分计算为例，

$$I_3^{\mu\nu} = \int \frac{d^D k}{i\pi^{D/2}} \frac{k^\mu k^\nu}{k^2 \left(k - p_1\right)^2 \left(k - p_1 - p_2\right)^2}$$

其中 p_1 和 p_2 表示外线动量。Passarino 和 Veltman 采用将 $I_3^{\mu\nu}$ 写成最一般的形式，即用形状因子乘以外部动量和（或）度规张量表示的形式。在以上我们的例子里，我们将该张量积分写为

$$I_3^{\mu\nu} = p_1^\mu p_1^\nu C_{21} + p_2^\mu p_2^\nu C_{22} + \{p_1^\mu, p_2^\nu\} C_{23} + g^{\mu\nu} C_{24} .$$

这里 $\{p_1^\mu, p_2^\nu\} = p_1^\mu p_2^\nu + p_2^\nu p_1^\mu$。然后通过方程两边与外部动量 $p_1^\mu p_2^\nu, p_2^\nu p_1^\mu, \{p_1^\mu, p_2^\nu\}$ 和度规张量 $g^{\mu\nu}$ 收缩，来解出形状因子 C_{21}, C_{22}, C_{23} 和 C_{24}。左边的结果是按照传播子重写成圈动量 k^μ 与和外部动量之间的标量乘积的形式，例如：

$$2 p_1 \cdot k = k^2 - (k - p_1)^2 + p_1^2 .$$

右边的前两项消去了传播子，而最后一项根本与圈动量无关。接着得到线性方程求解形状因子。在这一步，我们会遇到计算 Gram 行列式

$$\Delta_3 = 4\begin{vmatrix} p_1^2 & p_1 \cdot p_2 \\ p_1 \cdot p_2 & p_2^2 \end{vmatrix}.$$

这个算法的缺点就在出现在对这个行列式的计算上面。如果在一个 p_1 和 p_2 共线的相空间域中，Gram 行列式趋向与零，这时稳定的数值计算程序就变得十分困难。目前已经有对 Passarino－Veltman 算法的改进的方法，这在很大程度上避免了 Gram 行列式出现。

近年来出现了解析或数值计算一圈图的一点、二点、三点、四点，甚至五点积分函数的程序包。LoopTools就是这样一个计算标量和张量单圈积分的程序包。由于多点单圈图的动量积分具有可以最终归结为Spence函数计算的共性，因而这样的积分是可以用计算机代数系统来解决复杂计算的困难。然而二圈以上的动量积分计算就不再具有这一共性，不具备统一的算法，因此用计算机代数系统来解决复杂计算还存在困难。

四、高阶量子修正计算的计算机代数程序

目前比较著名的,用于高阶电弱和 QCD 理论微扰计算，基于计算机代数的程序包有以下四组。每组中的不同程序包大都有相同的语法，它们还可以鏈接起来使用。

(1)FeynArts, FeynCalc, FormCalc, LoopTools, TwoCalc 和 s2lse。

(2)GEFICOM,QGRAF,MATAD 和 MINCER。

(3)DIANA,QGRAF 和 ON-SHELL2,

(4)xloops 和 GiNaC。

如前面所列举的高阶精确计算程序包表明：计算机代数系统与其余程序部份间的有效交流是极为重要。这些外部程序可能是数值计算，文字处理，数据库，专家库类型的程序包，也可能是特别适合问题的某一部份计算的计算机代数系统。将来人们会更多地强调对程序集成的某种程度的标准化，以及要求具备不同系统间数据传递功能。随着将来硬件和软件的发展，计算机代数系统功能的扩展会向两个不同方向发展：一方面，计算机代数系统可以进行远远超出现在水平的非常复杂的计算，另一方面计算机代数系统越来越成为解决常规问题的软件环境中的一部份。

§7．3 Mathematica 语言编程

单词，词汇 ---→ 语句，文章

1．过程

一般形式为：

Module[{<局部变量名表>},表达式 1；表达式 2；...表达式 n]

例如：

```
unit[x_, y_, z_] := Module[{len}, len = Sqrt[x^2 + y^2 + z^2];
                                  N[{x/len, y/len, z/len }]  ]
```

Block[{<局部变量名表>},表达式 1；表达式 2；...表达式 n]

2．控制选择

(i)　顺序控制　　分号“ ;”

(ii) 条件控制

If 语言结构。

If[逻辑表达式，表达式]

If[逻辑表达式，表达式 1，表达式 2]

If[逻辑表达式，表达式 1，表达式 2，表达式 3]

例如：

```
f[x_] := If[(x > 0) || (x=0) , N[Sqrt[x]], Print[“x is negative。 ”],
                Print[“x is not numerical.”]
```

Which语句结构。

Which[条件1，表达式1，条件2，表达式2，..., 条件n，表达式n]

例： Which[2= =3,x,3= =3,y]

其结果为y。这是由于条件2==3的结果为False,而条件3==3的结果为True.

Switch语句结构。

Switch[判别表达式，模式 1，表达式 1，模式 2，表达式 2，..., 模式 n，表达式 n]

例：　　i=1

　　Switch[i^2,0,x,1,y,2,z]

最后结果为y。

（iii）循环控制

Do语句结构

Do[表达式，{循环描述}]

例：　Do[Print[2^i],{i,1,5}]

该指令的结果是循环打印出2^i，（i=1,2,3,4,5）的值2，4，8，16，32。

For语句结构

For[初始表达式，条件，步进表达式，循环表达式]

例如： For[i=0,i<=10,++1,Print[i]]

值为0，1，2，3，4，5，6，7，8，9，10。

While语句结构

While[条件，循环表达式]

例如：

```
i=0
While[ i<=10,Print[i];i++ ]
```

3.程序包结构

一般的程序包具有如下的基本框架：

BeginPackage[“程序包名称 ’ ”]

名称::usage=“ 字符串，程序包中定义在包外可以使用的函数、变量等的名字和使用说明。”

……

Begin[“ 'Private' ”]

……

程序包主体.(包括外部可用的函数及变量,一些内部函数变量的定义.)

……

End[ ]

EndPackage[ ]