Contents

0 绪论 **7**

1 计算机数和误差 **11**

1.1 计算机数及其表示 11

1.2 舍入误差对计算的影响 12

1.3 减法的计算 13

2 物理学中的常用数值方法 **15**

2.1 数值积分方法 15

2.1.1 梯形法和辛普生方法 15

2.1.2 反常积分的计算 17

2.1.3 高斯积分方法 19

2.1.4 哥西主值的计算 22

2.1.5 急速振荡函数的积分 24

2.1.6 高维积分的计算 26

2.1.7 固体热容量的计算 26

2.2 方程的求根和最优化方法 27

2.2.1 二分法 27

2.2.2 牛顿法和割线法 28

2.2.3 一维最优化, 黄金分割法 29

2.2.4 高维最优化, 共扼梯度方法 31

2.2.5 约束最优化 35

2.2.6 最小二乘法及曲线拟合 36

2.3 函数的求值 38

2.3.1 多项式的求值和级数求和 38

2.3.2 连分式的求值 40

2.4 常微分方程的数值解法 41

2.4.1 Euler方法 . . . 41
2.4.2 龙格—库塔方法 . . . 42
2.4.3 两点边值问题 . . . 46
2.5 插值和逼近 . . . 50
2.5.1 多项式插值 . . . 50
2.5.2 分式有理函数插值和外推 . . . 51
2.5.3 三次样条插值 . . . 52
2.5.4 列表函数的积分 . . . 54
2.5.5 Padé插值与外推 . . . 54
2.5.6 发散级数的Cesáro求和及其推广 . . . 58
2.5.7 Borel求和 . . . 60
2.5.8 Bézier逼近 . . . 61
2.5.9 多元函数的插值及逼近 . . . 63
2.6 快速付里叶变换 . . . 65
2.6.1 付里叶变换 . . . 65
2.6.2 离散付里叶变换 . . . 68
2.6.3 快速付里叶变换 . . . 71
2.7 附录: 特殊函数的计算 . . . 74
2.7.1 误差函数的计算 . . . 76
2.7.2 指数积分的计算 . . . 77
2.7.3 整数阶贝塞尔函数及虚宗量贝塞尔函数的计算 . . . 78
2.7.4 球谐函数的计算 . . . 87
2.7.5 椭圆积分的计算 . . . 89
2.8 附录: 高斯积分的节点和权重 . . . 91

3 数值线性代数 97

3.1 主元消去法解线性代数方程组 . . . 97
3.2 LU分解法 . . . 98
3.3 三对角方程组的求解 . . . 101
3.4 实对称矩阵的本征值和本征向量 . . . 101
3.4.1 Householder 方法 . . . 102
3.4.2 QL 算法 . . . 104
3.5 负本征值方法和递推方法 . . . 106

4 电磁场的计算 109

4.1 Maxwell 方程, 边值问题 . . . 109
4.2 差分和差商 . . . 111
4.3 二维Possion 方程的五点差分格式 . . . 112
4.4 差分方程的求解 . . . 115
4.4.1 简单迭代法—Jacobi 方法 . . . 116
4.4.2 逐次迭代法—Gauss–Seidal 迭代方法 . . . 116
4.4.3 逐次超松弛迭代法(Successive Over-Relaxation) . . . 116
4.5 变分方法复习 . . . 118
4.6 有限元方法的理论基础 . . . 121
4.7 用有限元方法求解二维Laplace 方程 . . . 123

5 Monte Carlo方法 129

5.1 随机变量及其分布 . . . 129
5.2 赝随机数的产生 . . . 131
5.3 用Monte Carlo 方法计算积分 . . . 133
5.4 自相似性与分形 . . . 136
5.5 扩散限制粘结的计算机模拟 . . . 139
5.6 高分子的模拟和无规行走 . . . 139

6 逾渗和统计物理问题 143

6.1 逾渗简介 . . . 143
6.2 逾渗的计算机模拟和分析 . . . 146
6.3 正则系综和统计模型 . . . 147
6.3.1 Ising 模型 . . . 148
6.3.2 Lenard-Jones 模型 . . . 149
6.4 统计物理的Monte Carlo模拟, Metroplis方法 . . . 149
6.5 Ising 模型的Monte Carlo 模拟 . . . 152
6.6 经典统计物理复习 . . . 153
6.6.1 The micro canonical ensemble(NVE) . . . 154
6.6.2 The canonical ensemble(NVT) . . . 154
6.6.3 The NPT ensemble(NPT) . . . 155
6.6.4 The grand canonical ensemble(. . . 155
6.6.5 Thermodynamical relations . . . 156
6.6.6 Correlation functions . . . 156
6.6.7 Time correlation function and transport coefficients . . . 158

6.7 液体模型的Monte Carlo模拟 . . . 159
6.7.1 Monte Carlo Simulation of The Lenard -Jones Model . . . 160
6.7.2 自由能计算 . . . 162
6.7.3 王-Landau 方法及其它 . . . 164
6.8 分子动力学方法 . . . 165
6.8.1 General procedure of MD (NVE ensemble) . . . 165
6.8.2 Simulation of Lennard-Jones liquids . . . 169
6.8.3 Simulation of Hard-sphere systems . . . 170
6.9 Simulation of Langevin dynamics . . . 171
6.10 Long range force and Ewald summation . . . 172

7 原子结构的计算 173
7.1 原子结构问题 . . . 173
7.2 变分法 . . . 174
7.3 Virial 定理和Hellmann-Feynman 定理 . . . 176
7.4 轨道近似和Hartree–Fock 方程 . . . 179
7.5 统计近似和X_α 方程 . . . 181
7.6 径向方程的求解方法 . . . 184
7.7 计算程序 . . . 188
7.8 Sturm-Liouville 问题 . . . 210

Chapter 0

绪论

不论是理论物理, 还是实验物理都离不开数值计算, 例如, 量子力学中简谐振子的波函数可以用厄米多项式来表示, 但为了得到波函数的数值, 仍然需要作数值计算; 实验上测量到的数据要进行数据处理, 也涉及到数据拟合, 分析并计算误差等一系列数值计算.

大家在过去的学习中已经知道, 牛顿力学方程只有二体问题是可解的(自由粒子或小振动等特殊问题除外), 三体以上的问题曾折磨了全世界的许多优秀的数学家和理论物理学家, 但最终也未得到解析解; 麦克斯韦方程组只有在一些非常特殊的条件下才能求得解析解; 量子力学的薛定格方程, 除了氢原子和简谐振子外, 还没有一个真实的物理问题可以找到解析解; 统计物理告诉我们, 只要求得了系统的配分函数, 则一切物理量都可通过求微分得到, 然而, 除了可数的几个二维模型外, 没有一个实际的物理系统的配分函数被求出来过. 一些大规模的物理实验, 耗资在数亿美元, 如果没有充分的论证就去做的话, 其浪费是十分巨大的.对于这样一类问题，大规模的数值计算往往可以发挥重要作用.

大规模数值计算也提供了理论物理和实验物理之间的桥梁. 几乎所有的实际物理问题都无法得到精确解,发展各种理论模型和近似方法就成为理论物理学的重要方面. 理论模型的正确性最终需要经受实践的检验,但对于一个确定的物理模型, 其对应的实验体系往往很难获得,为了检验针对一个确定模型的近似方法,最好能够得到这个模型的精确解,而数值模拟方法正好能够提供这样的结果.

大量的实验体系往往包含多种因素,这些因素有些可以排除,而有些非常难以排除.如果一个理论的出发点包含了所有的实验因素,则这样的理论几乎肯定得不到解析结果. 但是, 数值模拟方法一般总能够把各种因素包含进去,从而能够更好的模拟实验环境.

需要指出的是, 长期以来, 我们总有一些误解, 一讲到计算,似乎就需要超级计算机. 事实上,现在使用的笔记本计算机,各种微型计算机的计算能力往往比10年前的超级计算机的计算能力更强. 用一台目前似乎已经被人看不起的IBM-PC/XT微机(IBM-PC 是基于intel芯片的第一代个人计算机, 大约1980年推出, 其CPU是intel-8086芯片, 有64k内存,

两个5吋软驱,机器带有MS-DOS 1.0和CP/M 两个操作系统，IBM-PC/XT是IBM-PC的扩展，增加了一个10M的硬盘，并且把内存增加到256K), 就可以很容易地解决三体问题, 四体问题, ... 所有元素周期表上的原子的能级和波函数也可以相当精确地计算出来. 如果你拥有一台目前市场上一般配置的微机, 你就可以模拟一些二维和三维模型系统的配分函数, 如Lenard-Jones流体, 三维Ising模型等, 可以计算元素固体和简单化合物晶体的电子结构, 声子谱等. 进一步, 你如果拥有或能够使用到大型计算机, 或利用微机构成计算集群，你就可以作真实流体的流体力学计算, 可以计算复杂边界条件下的三维电磁场, 可以计算三维真实系统的配分函数, ... 同样你也能够在大型物理实验开始前作很多模拟, 对实验的各种情况有一个清楚的概念.

本课程的目的在于对物理中的数值计算进行一些入门介绍, 使大家在学完本课程后, 在组织一些较大规模的计算时心中有数, 少走弯路. 课程分两部分, 一部分是基本算法, 主要介绍计算机数的特点, 舍入误差, 计算稳定性等基本概念和一些常用的数值计算方法; 第二部分则结合物理问题, 讲一, 二个专题, 使大家学到一些组织较大规模计算的步骤和技巧. 学习本课程要求学过四大力学(经典力学, 电动力学, 热力学和统计物理学及量子力学)和数学物理方法, 掌握至少一门计算机高级语言(如C, PASCAL, FORTRAN, Java 等). 计算是一门实践性很强的课程, 上机实习是本课程的一个有机组成部分.

关于本讲义的一点说明:

这是笔者在1993年为开设《计算物理》课程准备的讲义. 1992年,上海交通大学应用物理系代理系主任张仲渊教授找到我，告诉我决定在本科生高年级开设《计算物理》课程, 并安排由笔者开课. 在接到这一任务后, 笔者翻阅了当时国内出版的几本《计算物理》教材, 发现都过于专门, 不适于做为本科高年级的教材, 而更适合于某一专业的研究生教材. 为此, 我只好试图自己写一本讲义, 以应急需. 基本算法部分主要参考了W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery 所著Numerical Recipes, The Art of Scientific Computing 一书，当时，这本书出版时间还不算太长，但已经成为大多数经常从事数值计算的物理学工作者的手头必备书，我自己当时也买了一本，书价是我的两个月的工资。这本书一方面讲解算法，另一方面对每一个算法给出了透明，简洁的源程序。尽管这些程序的效率，安全性方面比一些成熟的软件包要差，但由于其简洁清楚，特别适合于修改后放到自己的程序中去。讲义的内容曾在上海交通大学应用物理系讲授多次, 各个年级的同学积极参与了本课程的学习并对课程的讲法提出了很多非常宝贵的意见和建议, 这些建议在讲义的成文中对笔者有很大帮助, 在此表示感谢.

2000年后,物理系决定把计算物理课程改为双语课程,这本讲义就算完成了其历史使

命. 在教学过程中,我也发现这一讲义对一些问题的讲解过于简洁, 而内容的深度也不太适合本科生的学习.这一段时间,国际上出现了几本比较合适的教材,所以,当时认为这一讲义不需要继续修订,也就放在一边了.

随后,研究生的课程做了一些改革, 增加了36学时的数值计算,任课老师感觉这个讲义的内容作为研究生数值计算的课程比较合适, 遂采用这本讲义的电子版教学.

2013年10月，致远学院的叶曦副院长和负责物理班教学的郑杭教授找我，希望我为致远学院的同学开设一门物理中数值计算的课程。注意到致远学院已经有至少两门相关课程，这门课程应该更多强调物理方面。为了这个课程，利用寒假, 我修订了讲义, 主要是对原来的一些表述做了修改和补充，同时尽可能强调物理方面的内容.

鉴于目前已经存在大量优秀的数学软件, 如Matlab, Maple等. 大量的小规模的计算往往可以通过这些软件方便进行，因此，此次修订时时也适当提到这方面的内容。

讲义中提到的一些程序将放在课程主页上, 这些程序中，有些来自前面提到的Numerical Recipes，有些是我曾经多次使用过的, 也有些是专为此讲义而准备的, 由于时间关系, 没有经过仔细调试和考验, 因此在用于实际问题时, 一定要经过仔细试算, 以免造成不必要的损失, 同时, 我也愿在此声明, 笔者对因使用此讲义中的程序而导致的任何直接的和间接的损失将不负担任何责任. 所有程序都以FORTRAN 77 写出, 鉴于C 语言在实际工作中使用的越来越广泛, 同时也是很多同学喜欢的语言, 部分程序已经翻译为C 语言. 由于程序是用FORTRAN 写的, 翻译中自然留有FORTRAN的痕迹. 在上次讲义完成到这次修订之间，面向对象的程序设计逐步走向主流，其代表性语言是c++和Java，建议有志于从事计算物理的同学注意这一点.

Chapter 1

计算机数和误差

在这一章中, 我们将给出电子数字计算机中数的表示, 计算规则, 舍入误差及算法稳定性等概念, 在处理上, 并不追求数学上的严密, 而是用一些启发式的推导和例子来说明主要概念, 需要进一步钻研的同学, 可以参看有关计算数学的书籍.

1.1 计算机数及其表示

我们在初中的数学上就已经知道, 数有整数, 实数和复数, 整数可以取0, ± 1, ± 2, ..., $\pm n$, ..., $\pm\infty$, 实数则是整个数轴上的连续统. 数学分析就是建立在实数系上的. 实数的运算满足加法和乘法的交换律, 接合律等一般的运算律. 而计算机所能表达的数的全体, 则是一个离散的, 有限的集合. 通常计算机整数的取值范围为$-N$ 到$N-1$, N 的数值随计算机不同而异, 如IBM-PC机的$N = 32768$. 实数也有一个取值范围, 在IBM-PC 上, 实数的绝对值最大约为10^{38}, 最小约为10^{-38}, 大于最大值的数将造成上溢, 小于最小值的数将造成下溢.(在很多计算机系统中, 下溢作为0 来处理). 计算机实数在其范围内的分布是非常不均匀的, 在靠近0 的地方较密, 离开0 越远越稀疏. 计算机实数一般不满足实数的运算律.

计算机中的实数通常用规格化的二进制浮点数来表示, 例如11.001, 0.011001 和0.11001 分别表示为$0.11001 \cdot 2^2$, $0.11001 \cdot 2^{-1}$ 和$0.11001 \cdot 2^0$. 由于计算机只有有限的字长, 所以只能表示有限个实数, 例如, 对于一个字长为五的计算机, 将无法区分0.11001, 0.110010101. 既然计算机的实数是一个有限的集合, 那么初始数据和中间结果都有可能不在这个集合中, 于是必须用计算机实数去近似表示相应的实数, 从而引进舍入误差. 如果按照四舍五入的原则去近似表示, 则由计算机表示一个实数时引进的舍入误差不大于最后一位数的一半, 但很多计算机采用了只舍不入的方法, 则由计算机数表示一个实数时引进的舍入误差最大可为最后一个二进制位.

1.2 舍入误差对计算的影响

在数值计算过程中, 一般每一步都有舍入误差, 舍入误差的积累有可能使整个计算失败. 例如, 考虑下述问题, 计算积分

$$E_n = \int_0^1 x^n e^{x-1} dx, \qquad n = 1, 2, \ldots, 20.$$

利用分部积分法得到

$$\int_0^1 x^n e^{x-1} dx = x^n e^{x-1}|_0^1 - n \int_0^1 x^{n-1} e^{x-1} dx,$$

或

$$E_n = 1 - n\, E_{n-1}$$

容易算出$E_1 = 1/e$. 利用上述递推公式, 在IBM-PC微机上用双精度计算结果如下:

$$E_1 = 0.36787944117144 \qquad E_2 = 0.26424111765712$$
$$E_3 = 0.20727664702865 \qquad E_4 = 0.17089341188538$$
$$E_5 = 0.14553294057308 \qquad E_6 = 0.12680235656152$$
$$E_7 = 0.11238350406936 \qquad E_8 = 0.10093196744509$$
$$E_9 = 0.091612292994171 \qquad E_{10} = 0.083877070058293$$
$$E_{11} = 0.077352229358780 \qquad E_{12} = 0.071773247694637$$
$$E_{13} = 0.066947779969723 \qquad E_{14} = 0.062731080423873$$
$$E_{15} = 0.059033793641902 \qquad E_{16} = 0.055459301729570$$
$$E_{17} = 0.057191870597308 \qquad E_{18} = -.029453670751536$$
$$E_{19} = 1.5596197442792 \qquad E_{20} = 30.192394885584$$

被积函数在积分区间$[0, 1]$ 的内部总是正的, 因此积分值不可能为负, 更进一步, 由于

$$E_n = \int_0^1 x^n e^{x-1} dx \le \int_0^1 x^n dx = \frac{1}{n+1}$$

所以, 必有$E_{20} \le 1/21$. 显然, 上述计算结果是不正确的, 而问题就出在舍入误差上面. 我们对此作一个简单的分析. 双精度数可表示14位10进制数, 所以E_1 的误差约为$\epsilon = 10^{-15}$ (也可能大于此数, 这里仅作一数量级的估计). 在递推过程中, 这个误差将被传播, 放大, 最后导致计算结果的错误. 误差的传播规律为

$$\begin{aligned} E_2 &= 1 - 2(E_1 + \epsilon) = 1 - 2E_1 - 2\epsilon \\ E_3 &= 1 - 3(1 - 2E_1 - 2\epsilon) = 1 - 3(1 - 2E_1) + 3!\epsilon \\ &\cdots \end{aligned}$$

计算到E_{15} 时的误差为$15!\epsilon \approx 10^{-3}$, 而计算到$E_{18}$ 时的误差已为$18!\epsilon \approx 6.0$, 已经远大于E_{18} 的上限$1/19$.

现在我们考虑把递推关系改写成

$$E_{n-1} = \frac{1 - E_n}{n}, \qquad n = \cdots, 4, 3, 2.$$

利用关系

$$E_n \leq \frac{1}{n+1}$$

取$E_{30} = 0.0$ 开始计算, 此时误差为$\epsilon \leq 1/31 \approx 0.032$, 作一次递推后得到$E_{29}$, 其误差减小到

$$\frac{1}{30}\frac{1}{31} \approx 0.0011$$

在计算E_{20} 时, 误差已减小到

$$\frac{20!}{31!} \approx 3 \times 10^{-16}$$

准确到14位有效数字, 初始误差的影响已完全消除了. 而每一步的舍入误差，都在其后的迭代中减小和消除。下表是计算结果.

$E_{30} = 0.0$	$E_{29} = 0.33333333333333$
$E_{28} = 0.033333333333333$	$E_{27} = 0.034523809523810$
$E_{26} = 0.035758377425044$	$E_{25} = 0.037086216252883$
$E_{24} = 0.038516551349885$	$E_{23} = 0.040061810360421$
$E_{22} = 0.041736443027808$	$E_{21} = 0.043557434407827$
$E_{20} = 0.045544884075818$	$E_{19} = 0.047722755796209$
$E_{18} = 0.050119854958094$	$E_{17} = 0.052771119168995$
$E_{16} = 0.055719345931236$	$E_{15} = 0.059017540879298$
$E_{14} = 0.062732163941380$	$E_{13} = 0.066947702575616$
$E_{12} = 0.071773253648030$	$E_{11} = 0.077352228862664$
$E_{10} = 0.083877070103394$	$E_9 = 0.091612292989661$
$E_8 = 0.10093196744559$	$E_7 = 0.11238350406930$
$E_6 = 0.12680235656153$	$E_5 = 0.14553294057308$
$E_4 = 0.17089341188538$	$E_3 = 0.20727664702865$
$E_2 = 0.26424111765712$	$E_1 = 0.36787944117144$

由上例可以看出舍入误差对计算的影响, 如果我们在计算中不去分析算法, 即使编出了正确的程序, 也可能得出错误的, 有时甚至是荒谬的结果.

1.3 减法的计算

也许, 大家认为减法是一种非常简单的计算, 但是在用计算机做减法时, 如不小心, 就很

容易出错, 问题就出在舍入误差上面. 我们考虑下面的例子: 对于大的x, 计算

$$\sqrt{x+1}-\sqrt{x} \tag{1.3.1}$$

取$x = 162835.0$, 在一个具有6位十进制有效数字的计算机上, 计算结果为0.001, 而准确到6位十进制有效数字的精确结果为0.123907×10^{-2}, 计算中损失了5位有效数字. 对于更大的x, 则可能根本得不到正确的结果. 但是, 如果把计算公式改为

$$(\sqrt{x+1}-\sqrt{x})\frac{\sqrt{x+1}+\sqrt{x}}{\sqrt{x+1}+\sqrt{x}} = \frac{1}{\sqrt{x+1}+\sqrt{x}} \tag{1.3.2}$$

就可以得到较好的结果. 又如对于大的N, 计算

$$\int_N^{N+1} \frac{dx}{1+x^2} = \arctan(N+1) - \arctan(N)$$

时, 可以使用等价的公式:

$$\arctan(N+1) - \arctan(N) = \arctan\left(\frac{1}{1+N(N+1)}\right)$$

同理, 对于小的ϵ

$$\sin(x+\epsilon) - \sin(x) = 2\cos\left(x+\frac{\epsilon}{2}\right)\sin\left(\frac{\epsilon}{2}\right)$$

上面分析的是一些简单的例子, 在实际问题的计算中, 常常会碰到一些十分复杂的问题, 即使有经验的计算研究人员也经常在减法计算中犯错误.

练习:

分析下面的表达式, 给出对任何x(只要在所给函数定义域内)都能得出正确结果的计算公式.

$$\cos^2 x - \sin^2 x$$

$$\ln(x) - 1$$

$$e^{2x} - e^x$$

$$\sqrt{1-v^2} - 1$$

$$\frac{\sqrt{1+x^2}-1}{x^2} - \frac{x^2\sin(x)}{x-\mathrm{tg}(x)}$$

Chapter 2

物理学中的常用数值方法

在这一章和随后的几章中, 我们将介绍几种常用算法, 这些内容应该在数学课中讨论,但目前的数学课程并不包含这些属于计算数学课程的内容, 所以我们先作一些介绍.

2.1 数值积分方法

积分分为定积分和不定积分两种, 我们在这里只讨论定积分, 不定积分的计算可作为变上限的定积分来计算. 考虑定积分

$$I = \int_a^b f(x)dx \tag{2.1.1}$$

它代表$f(x)$曲线下的面积.

2.1.1 梯形法和辛普生方法

在微积分中, 我们知道积分是由求和的极限定义的, 这就是第一种计算积分的方法—梯形法. 把区间$[a,b]$分成一些子区间$[a = x_0 < x_1 < x_2 < \cdots < x_n = b]$, 则(2.1.1)的积分$I$可近似为:

$$I = \frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} - x_i)(f(x_i) + f(x_{i+1})) \tag{2.1.2}$$

在实际计算中, 常采用等间距分法, 即$x_{i+1} - x_i = h = \frac{b-a}{n}$为一常数, 上面公式可简化为:

$$I = \frac{h}{2}\sum_{i=0}^{n-1}(f(x_i) + f(x_{i+1}))$$

为了节省工作量, 实际使用的计算公式为:

$$I = h\sum_{i=1}^{n-1} f(x_i) + \frac{h}{2}(f(x_0) + f(x_n)) \tag{2.1.3}$$

当子区间越来越小, $h \to 0$时，求和的结果将趋于积分值。由于我们事先并不知道积分的结果，所以我们也就不知道求和结果和积分结果之间的差别。为此，我们需要估计计算误差，一个直观的估计是比较不同大小的子区间的结果之差。为了在每次子区间变小时能够用到前次的计算结果，我们可以从$n = 1$开始计算, 每次把区间数增加一倍, 则前一次计算的函数结果仍然有用, 通过比较连续两次的计算结果之差是否小于给定的误差限来结束计算.

梯形方法实际上把每一个子区间上的函数近似为线性函数$\alpha x + \beta$, 然后计算其积分值, 再把结果加起来.(作为一个练习, 请自行证明), 因此, 我们说梯形方法是具有线性代数精度的方法. 下面考虑具有二次代数精度的方法, —辛普生方法. 先考虑把积分区间$[a, b]$分为二个子区间, 分点为$(a, a + h, a + 2h = b)$, $h = \frac{b-a}{2}$, 在区间上把被积函数用二次函数来近似, 即取$f(x) \approx \alpha(x - a)^2 + \beta(x - a) + \gamma$, 其中$\alpha, \beta, \gamma$ 可利用分点上的函数值$f(a), f(a + h), f(b)$表示为:

$$\alpha = \frac{f(b) - 2f(a + h) + f(a)}{2h^2}$$

$$\beta = \frac{-f(b) + 4f(a + h) - 3f(a)}{2h}$$

$$\gamma = f(a)$$

简单地计算得到:

$$\int_a^b (\alpha(x - a)^2 + \beta(x - a) + \gamma)\, dx = \frac{h}{3}(f(b) + 4f(a + h) + f(a)) \tag{2.1.4}$$

这就是著名的辛普生求积公式. 显然, 它具有二次代数精度.

实际计算中, 把被积区间$[a, b]$先分为N个子区间, 然后在每个子区间中用辛普生求积公式计算, 并把最后结果加起来. 其计算公式为:

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(a) + 4f(a + h) + 2f(a + 2h) + 4f(a + 3h) + \cdots + f(b)) \tag{2.1.5}$$

这是一个在实际计算中常用的方法, 它对于一般的有限区间的积分都能给出较好的正确结果. 辛普生方法有各种不同的实现, 我们在这里给出简单的一种, 取每个子区间均相等, 从二个子区间开始, 每次子区间数目加倍, 这样可以在每次计算时使用前次的计算值. 这种实现的缺点是在整个积分区间均匀取点, 对于在部分区间变化剧烈而在其余地方变化平缓的被积函数计算效率不高. 这一实现可以充分利用它与梯形法的关系. 假定$n = 2^k$, 用S_k表示用2^k个子区间梯形公式求得的结果. 则式(2.1.5)可写为:

$$\begin{aligned}\int_a^b f(x)dx &\approx \frac{1}{3}[4h(f(a+h)+f(a+2h)+\cdots+f(b-h)+\frac{1}{2}f(a)+\frac{1}{2}f(b))\\ &\quad -2h(f(a+2h)+f(a+4h)+\cdots+f(b-2h)+\frac{1}{2}f(a)+\frac{1}{2}f(b))\\ &\quad +h(f(a)+f(b))-2h(f(a)+f(b))+h(f(a)+f(b))]\\ &= \frac{1}{3}(4S_k-S_{k-1}) \qquad (2.1.6)\end{aligned}$$

计算定积分还有很多方法, 如龙贝格方法, Newton-Cotes方法等, 我们将不在此讨论, 有兴趣的同学可参看有关计算数学的书籍. Quadpack（netlib.org）中包含了若干非常稳定的求积分的程序，建议在实际工作中使用。

在Matlab中，梯形法，辛普森均作为标准函数提供，实际应用时，只要写一个被积函数的m程序，调用积分程序即可。常用的几个求积分的函数是quad, quadl, quadgk等。其中quad使用自适应的辛普森算法，quadl应用龙贝格加速来改善计算效率和提高精度。例如，计算积分

$$\int_0^1 \frac{\sin(x^2)}{1+x^4}\,dx$$

先定义函数func 并存入func.m

```
function y=func(x)
    y=sin(x.^2)./(1.+x.^4);
end
```

然后在Matlab的命令窗口运行

```
>> format long
>> quad(@func,0,1)
```

结果为:

```
ans =

   0.229253354700751
```

2.1.2 反常积分的计算

反常积分分为两类, 一类是积分区间有限, 在积分区间内被积函数有奇点, 另一类是积分区间为无限. 对于二者兼而有之的积分, 可以分为两个或多个积分来处理, 使每个积分为上述两类之一.

积分区间内含有奇点的积分

对于积分区间内含有奇点的积分, 根据奇点的性质, 可采用下面的方法处理.

可去奇点, 设x_0为一可去奇点, 且知道在x_0点被积函数的极限, 则只要在计算函数时, 在x_0的一个邻域内用极限取代函数值即可. 例如, $\frac{\sin(x)}{x}$在$x=0$有可去奇点, 但当$x\to 0$时, $\frac{\sin(x)}{x}\approx 1-\frac{x^2}{6}+\cdots$, 在计算函数时, 可使用下面的语句:

```
IF(ABS(X).LT.1.0E-4) THEN
  F=1.0-X*X/6.0
ELSE
  F=SIN(X)/X
ENDIF
```

极限方法, 有时我们并不知道奇点处函数的极限表达式, 有时甚至不知道奇点是否可去, 此时可用极限方法来逼近. 例如, 若已知x_0为奇点, 对于积分

$$\int_{x_0}^{b} f(x)dx = \lim_{r\to x_0}\int_{r}^{b} f(x)dx$$

定义一个收敛于x_0的序列$b>r_1>r_2>\cdots$, 例如可取$r_n=x_0+2^{-n}$, 则上述积分可写为一个求和:

$$\int_{x_0}^{b} f(x)dx = \int_{r_1}^{b} f(x)dx + \int_{r_2}^{r_1} f(x)dx + \int_{r_3}^{r_2} f(x)dx + \int_{r_4}^{r_3} f(x)dx\cdots$$

等式中每一项都是正常积分, 当$\left|\int_{r_{n+1}}^{r_n} f(x)dx\right|\le\epsilon$时, 终止计算. 如果上式不收敛, 则很可能原积分不存在.

消除奇点, 有些奇点可以通过变量替换或对被积函数进行变换而消去. 我们看几个例子:

$$\int_0^1 \frac{\cos(x)}{\sqrt{x}}dx$$

作变量替换$x=t^2$, $dx=2t\,dt$, 则有:

$$\int_0^1 \frac{\cos(x)}{\sqrt{x}}dx = 2\int_0^1 \cos(t^2)dt$$

或把积分写为:

$$\int_0^1 \frac{\cos(x)}{\sqrt{x}}dx = \int_0^1 \frac{1}{\sqrt{x}}dx + \int_0^1 \frac{\cos(x)-1}{\sqrt{x}}dx = 2+\int_0^1 \frac{\cos(x)-1}{\sqrt{x}}dx$$

第二项积分中的奇点已成为可去奇点.

$$I(x)=\int_0^x \frac{e^{-t}}{1-t}$$

在$t=1$的邻域, 被积函数与$e^{-1}/(1-t)$很相像, 因此变为:

$$I(x)=\int_0^x \frac{e^{-1}}{1-t}dt + \int_0^x \frac{e^{-t}-e^{-1}}{1-t}dt = -\frac{1}{e}\ln|1-x| + \int_0^x \frac{e^{-t}-e^{-1}}{1-t}dt$$

所有的奇异性都在第一项, 而这一项可解析处理.

除上面例子演示的方法外, 还可用分部积分法消除奇点, 这里不再一一举例了.

积分区间为无限的积分

下面我们讨论积分区间为无限的情形. 为具体起见, 只讨论形为

$$\int_0^\infty f(x)dx$$

的积分, 其它形式的无穷区间的积分均可化为上面的形式.

变量替换法: 通过变量替换, 可把无穷区间的积分化为有限区间上的积分, 如令$x = -\ln(t)$, $dx = -\frac{dt}{t}$, 则有:

$$\int_0^\infty f(x)dx = \int_0^1 \frac{f(-\ln(t)}{t}dt = \int_0^1 \frac{g(t)}{t}dt \tag{2.1.7}$$

这里$g(t) = f(-\ln(t))$, 若$\frac{g(t)}{t}$在$t = 0$的邻域内有界, 则上式的积分成为一个正常积分, 否则就是前面讨论过的有限区间上的反常积分, 可以用前面的方法来处理. 上述变换对于当$x \to \infty$时, $f(x)$以e^{-kx}的形式衰减的函数, 计算结果十分令人满意. 其它常用的变换还有$x = \frac{t}{1-t}$, $dx = \frac{1}{(1-t)^2}dt$

$$\int_0^\infty f(x)dx = \int_0^1 f\left(\frac{t}{1-t}\right)\frac{1}{(1-t)^2}dt \tag{2.1.8}$$

以及$t = \tanh x$等.

极限法: 由定义

$$\int_0^\infty f(x)dx = \lim_{r\to\infty}\int_0^r f(x)dx$$

定义一个趋于∞的序列$0 < r_1 < r_2 < \cdots$, 例如可取$r_n = 2^n$, 则上述积分可写为一个求和:

$$\int_0^\infty f(x)dx = \int_0^{r_1} f(x)dx + \int_{r_1}^{r_2} f(x)dx + \int_{r_2}^{r_3} f(x)dx + \cdots \tag{2.1.9}$$

式中每一项都是正常积分, 当$|\int_{r_n}^{r_{n+1}} f(x)dx| \le \epsilon$时, 终止计算. 如果上式不收敛, 则很可能原积分不存在. 如果能找到尾项$\int_R^\infty f(x)dx$的一个较好的渐近表达式, 则上述积分只需进行到某一足够大的r_n处, 再加上尾项.

还有一些其它的有效方法, 这里不再一一列举, 我们下面就要讲到的高斯积分方法提供了另一种处理反常积分的工具.

2.1.3 高斯积分方法

前面的积分公式(梯形公式及辛普生公式)的代数精度都较低, 一个自然的问题是, 能否构造出高代数精度的公式, 回答是肯定的, 这就是高斯积分方法. 我们将不讨论高斯积分方法的理论基础, 而只是给出有关的计算公式, 对理论基础有兴趣的同学可以参看有关的计算数学的书籍.

有限区间的高斯型积分公式:

高斯—勒让德积分公式:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i) \tag{2.1.10}$$

此处, x_i为n阶勒让德多项式的根, 通常称为积分节点, w_i称为积分权重.

等权重切贝雪夫积分公式:

$$\int_{-1}^{1} f(x)dx = \frac{2}{n}\sum_{i=1}^{n} f(x_i) \tag{2.1.11}$$

这里x_i为积分节点, 权重为$2/n$.

第一类高斯—切贝雪夫积分公式:

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}dx = \frac{\pi}{n}\sum_{i=1}^{n} f\left(\cos\left(\frac{(2i+1)\pi}{2n}\right)\right) \tag{2.1.12}$$

第二类高斯—切贝雪夫积分公式:

$$\int_{-1}^{1} f(x)\sqrt{1-x^2}dx = \frac{\pi}{n+1}\sum_{i=1}^{n} \sin^2\left(\frac{i\pi}{n+1}\right) f\left(\cos\left(\frac{i\pi}{n+1}\right)\right) \tag{2.1.13}$$

如果积分区间不是$[-1,1]$, 而是$[a,b]$, 可利用变换

$$x = \left(\frac{b-a}{2}\right)t + \left(\frac{b+a}{2}\right) \tag{2.1.14}$$

将积分区间变为$[-1,1]$, 然后利用上述公式进行计算.

在实际计算时, 常把积分区间分为几个小区间, 在每一个小区间内使用高斯积分公式, 并把结果加起来.

练习*:*

试编写上述四种高斯积分公式的程序*,* 并利用所编写的程序计算积分

$$\int_{-1}^{1} \sqrt{x+1.5}dx, \qquad \int_{-1}^{1} \frac{1}{2+x}dx, \qquad \int_{0}^{1} \frac{dx}{\sqrt{x}}$$

$$\int_{0}^{2} e^{-\cos^2(x)}dx, \qquad \int_{0}^{1} \frac{\sin(x)}{x}dx$$

*(*积分所需的节点及权重在后面的表中给出*).*

无限区间的高斯型积分公式:

高斯—拉盖尔积分公式:

$$\int_{0}^{\infty} e^{-x} f(x)dx = \sum_{i=1}^{n} w_i f(x_i) \tag{2.1.15}$$

高斯—厄米积分公式:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = \sum_{i=1}^{n} w_i f(x_i) \tag{2.1.16}$$

这里x_i分别为n阶拉盖尔多项式和厄米多项式的根. 其数值和对应的权重的数值在附录的表中给出.

练习:

试编写上述二种高斯积分公式的程序, 并利用所编程序计算积分

$$\int_0^{\infty} \frac{e^{-x}}{1+x^4}dx, \qquad \int_0^{\infty} e^{-x^2}\cos(x)dx$$

各阶高斯积分公式的取点并不重和，当增加分点时，所有的函数值都要重算。1965年Kronrod提出了结合高斯方法和加速收敛技巧的算法。这一算法也可以处理有一定奇异性的被积函数，但需要把奇点放在积分限上。在Matlab中已实现这一算法，其调用命令是quadgk.

例如，计算积分

$$\int_0^{\infty} \frac{\sin x}{x} e^{-x}\,dx$$

定义函数funsin, 并存入funsin.m

```
function y=funsin(x)
  y=sin(x).*exp(-x)./x;
end
```

在Matlab命令窗口键入如下命令

```
format long
quadgk(@funsin,0,inf)

ans =

   0.785398163397455
```

这一积分的精确结果是$\frac{\pi}{4} = 0.785398163397448$

2.1.4 哥西主值的计算

在物理计算中, 哥西主值的计算占了很重要的位置, 如物质的介电函数的实部和虚部之间满足著名的Kramers-Kronig关系:

$$\epsilon(\omega) = \epsilon_1(\omega) + i\epsilon_2(\omega)$$

$$\begin{aligned} \epsilon_1(\omega) &= 1 + \frac{2}{\pi} P\int_0^\infty \frac{s\epsilon_2(s)}{s^2-\omega^2} ds \\ \epsilon_2(\omega) &= -\frac{2\omega}{\pi} P\int_0^\infty \frac{\epsilon_1(s)}{s^2-\omega^2} ds \end{aligned} \tag{2.1.17}$$

这个关系是一个严格的关系, 它来源于因果关系, 是非常普遍的. 在物理上, $\epsilon_2(\omega)$的测量和计算都比较容易, 因此, 一般不直接测量或计算$\epsilon_1(\omega)$, 而是利用上述关系由$\epsilon_2(\omega)$的数据进行计算.

上述Kramers-Kronig关系中出现的P$\int$就代表哥西主值积分, 设$a < c < b$, $f(x)$在$x = c$的邻域内无界, 则哥西主值积分定义为:

$$P\int_a^b f(x)dx = \lim_{r\to 0}\left[\int_a^{c-r} f(x)dx + \int_{c+r}^b f(x)dx\right] \tag{2.1.18}$$

不失一般性, 我们取$c = 0$并将积分取成$\int_{-a}^{a} f(x)dx$的形式. 令

$$g(x) = \frac{1}{2}[f(x) - f(-x)], \qquad h(x) = \frac{1}{2}[f(x) + f(-x)]$$

则

$$f(x) = g(x) + h(x)$$

由于$g(x)$是奇函数而$h(x)$是偶函数, 于是:

$$\begin{aligned} & \int_{-a}^{-r} f(x)dx + \int_r^a f(x)dx \\ = & \int_{-a}^{-r} g(x)dx + \int_r^a g(x)dx + \int_{-a}^{-r} h(x)dx + \int_r^a h(x)dx \\ = & 2\int_r^a h(x)dx \end{aligned}$$

因此,

$$P\int_a^b f(x)dx = 2\lim_{r\to 0}\int_r^a h(x)dx = 2\int_0^a h(x)dx = \int_0^a [f(x) + f(-x)]dx$$

这样我们就把哥西主值积分的计算变为在$x = 0$处有奇点的一个反常积分的计算. $h(x)$也可能在$x = 0$处无奇点, 此时只要计算一个常规积分. 例如前面的Kramers-Kronig关系中积分的计算可化为:

$$\frac{2}{\pi} P\int_0^\infty \frac{s\epsilon_2(s)}{s^2-\omega^2} ds = \frac{1}{2\pi} P\int_{-\infty}^\infty \frac{\epsilon_2(s)}{s-\omega} ds = \frac{1}{\pi}\int_0^\infty \frac{\epsilon_2(\omega+x) - \epsilon_2(\omega-x)}{x} dx$$

$$\frac{2\omega}{\pi}P\int_0^\infty \frac{\epsilon_1(s)}{s^2-\omega^2}ds = \frac{1}{2\pi}P\int_{-\infty}^\infty \frac{\epsilon_1(s)}{s-\omega}ds = \frac{1}{\pi}\int_0^\infty \frac{\epsilon_1(\omega+x)-\epsilon_1(\omega-x)}{x}dx$$

在得到上述关系时, 我们利用了ϵ_1为偶函数而ϵ_2为奇函数的事实.

当哥西积分区间内的无界邻域不止一个时, 可以把积分区间分为几段, 使得每一段中只包含一个无界邻域, 然后在每一段用上述方法计算.

另一种计算主值积分的方法是直接从定义出发, 设$f(x)$在积分区间内无奇点, 则积分

$$\begin{aligned} P\int_a^b \frac{f(x)}{x-x_0}dx &= \int_a^{x_0-\Delta} \frac{f(x)}{x-x_0}dx + \int_{x_0+\Delta}^b \frac{f(x)}{x-x_0}dx + P\int_{x_0-\Delta}^{x_0+\Delta} \frac{f(x)}{x-x_0}dx \\ &= I_\Delta + \lim_{\epsilon\to 0} I_{\epsilon,\Delta} \end{aligned} \tag{2.1.19}$$

I_Δ为第一行右方第一和第二项之和, 是正常积分, 可用前述任一方法计算.

$$I_{\epsilon,\Delta} = \int_{x_0-\Delta}^{x_0-\epsilon} \frac{f(x)}{x-x_0}dx + \int_{x_0+\epsilon}^{x_0+\Delta} \frac{f(x)}{x-x_0}dx$$

把$f(x)$在x_0点展开, 并逐项积分, 得到

$$I_{\epsilon,\Delta} = 2f'(x_0)(\Delta-\epsilon) + f^{(3)}(x_0)(\Delta^3-\epsilon^3) + \cdots$$

取极限$\epsilon\to 0$, 得到

$$I_{0,\delta} = 2\sum_{n=0}^\infty f^{(2n+1)}(x_0)\frac{\Delta^{2n+1}}{(2n+1)!(2n+1)}. \tag{2.1.20}$$

这样, 原主值积分为$I_{0,\Delta}+I_\Delta$.

练习:

试用上述方法计算主值积分

$$P\int_0^1 \frac{e^{-x}}{x-0.5},$$

对于不同的Δ, 在*(2.1.20)*中取两项, 比较计算结果与精确结果. *(*到*16*位的精确结果为-0.6150181462805674*)*

另外一种常用的计算主值积分的方法是基于如下公式

$$\lim_{\eta\to 0^+}\frac{1}{x-i\eta} = P\frac{1}{x} + i\pi\delta(x) \tag{2.1.21}$$

式中$\eta\to 0^+$表示η从大于0的方向趋于0. 这一公式当然是在积分的意义下理解。利用上述结果，就可以得到计算主值积分的一个方法：例如要计算

$$P\int_a^b F(x)\,dx$$

取一个小量η，计算$\int_a^b F(x-i\eta)\,dx$, 其实部给出$P\int_a^b F(x)\,dx$ 的依赖于η的近似值，通过减小η，可以得到精度要求下收敛的结果。

例如，对于前面的练习题，分别取不同的η, 计算结果如下

η	积分值
0.01	$-0.596196166026032 + 1.884174006156568i$
0.001	$-0.613114998941270 + 1.903350575685698i$
0.0001	$-0.614827622300549 + 1.905260181139026i$
0.00001	$-0.614999091791223 + 1.905451057233541i$

2.1.5 急速振荡函数的积分

在各种变换特别是付里叶变换中, 经常会遇到急速振荡函数的积分, 例如, 付里叶变换

$$\int_a^b f(x)\cos(nx)dx, \qquad \int_a^b f(x)\sin(nx)dx$$

又如付里叶—贝塞尔变换,

$$\int_0^1 f(x)xJ_n(\gamma_m x)dx$$

这里$0 < \gamma_1 < \gamma_2 < \cdots$ 是贝塞尔函数的根.

一般我们可把这类函数写为:

$$I(t) = \int_a^b f(x)K(x,t)dx, \qquad -\infty < a < b < \infty \tag{2.1.22}$$

式中$K(x,t)$为一振荡核而$f(x)$为非振荡部分. 由于振荡型函数在积分区间内多次取几乎相等的正值和负值, 如果用通常的方法计算, 则由于正负数相加, 造成有效数字的损失, 几乎得不到正确的结果.

下面先给出最常遇到的振荡函数的积分—付里叶系数的一种计算方法, 然后再讨论较为一般的方法.

付里叶系数的计算:

付里叶系数的复数形式为:

$$\int_a^b f(x)e^{isx}dx \tag{2.1.23}$$

对上式分部积分, 假定$f(x)$存在直到n阶的导数, 可以得到:

$$\int_a^b f(x)e^{isx}dx = e^{isa}\sum_{k=0}^{n-1}\left(\frac{i}{s}\right)^{k+1} f^{(k)}(a) - e^{isb}\sum_{k=0}^{n-1}\left(\frac{i}{s}\right)^{k+1} f^{(k)}(b) + R_n \tag{2.1.24}$$

当n趋于无穷大时, 余项R_n趋于0, 而求和项就给出了积分的近似值.

练习*:*

写出*(2.1.24)*的实部和虚部, 即$\int_a^b f(x)\cos sxdx$ 和$\int_a^b f(x)\sin sxdx$ 的积分公式.

上述公式需要计算$f(x)$的直到$n-1$阶的导数, 这对于较为复杂的函数来说, 曾是一个很重的负担. 不过目前计算机代数的发展, 计算导数已经没有多少困难. 为了看出这一方法的效力, 我们来看一个例子. 计算

$$I=\int_0^{2\pi} x\cos(x)\sin 30x dx$$

用公式(2.1.24)的虚部, 取$n=3$, 由$f(x)=x\cos(x)$, $a=0$, $b=2\pi$, $f(0)=0$, $f(2\pi)=2\pi$, $f''(0)=0$, $f''(2\pi)=-2\pi$, 代入得到:

$$I=\int_0^{2\pi} x\cos(x)\sin 30x dx=-\frac{1}{30}2\pi+\left(\frac{1}{30}\right)^3(-2\pi)=-0.20967222$$

而该积分的精确值为-0.20967247.

基于*Euler-Maclaurin*公式的梯形法： 可以证明（见王竹溪，郭敦仁《特殊函数概论》第一章），函数$f(x)$的积分可以写成

$$\int_a^{a+mh} f(x)dx=h\left[\frac{f(a\Psi}{2}+f(a+h)+\cdots+f(a+(m-1)h)+\frac{f(a+mh)}{2}\right] \quad (2.1.25)$$
$$+\sum_{k=1}^{n}\frac{(-1)^k B_k h^{2k}}{(2k)!}\left[f^{2k-1}(a+mh)-F^{2k-1}(a)\right]+R_n$$

其中

$$R_n=\theta\frac{(-1)^{n+1}B_{n+1}}{(2n+2)!}h^{(2n+1)}\left[f^{(2n+1)}(a+mh)-f^{(2n+1)}(a)\right]$$

这里$0\le\theta\le 1$，B_n是Bernoulli数。如果$f(x)$是周期函数，$f(a+mh)=f(a)$, 且$f(x)$存在直到$(2n+1)$阶的导数，则显然有$f^{(2k+1)}(a)=f^{(2k+1)}(a+mh)$, 于是，利用梯形法可以得到精确的结果。作为例子，取不同的点，试用梯形法计算如下积分，并于精确结果比较。

$$\int_0^1\frac{1}{1+\frac{1}{2}\sin 10\pi x}dx,\qquad \text{精确值}=1.15470054$$

Bessel函数的积分表示之一是

$$J_n(t)=\frac{1}{\pi}\int_0^{\pi}\cos\Phi t\sin x-nx)dx$$

计算$n=2$, $t=1$, 3, 7, 10时Bessel函数的值。

由Euler-Maclaurin公式可知，如果一个函数满足$f'(a)=f'(a+mh)$, $f^{(3)}(a)=f^{(3)}(a+mh)$, $\cdots$, 则梯形公式也可以得到很精确的结果。例如函数$f(x)=\exp(x^2(1-x^2))$, 在0和1满足上述关系，从而，可以用梯形公式计算

$$\int_0^1\exp(x^2(1-x^2))dx$$

这个积分的精确结果是1.03414105, 请分别用梯形法和辛普森法计算并比较其结果。

子区间法, 在一般情况下, 如果$K(x,t)=0$的根容易求出, 且记为$a \le x_1 < x_2 < \cdots < x_n \le b$, 则可用普通的积分规则计算每一个子积分$\int_{x_i}^{x_{i+1}}$, 并把结果加起来. 用这种方法得到的一般是一个交错级数, 且相邻项的绝对值比较接近, 其求和可用Euler变换计算. (Euler 变换将在后面讨论)

在讲了样条插值之后, 我们再介绍一种基于样条插值的计算积分(2.1.23)的方法.

更多的方法可以参看（P. J. Davis and P. Rabinowitz，Methods of Numerical Integration, Academic Press, 1984)

2.1.6 高维积分的计算

高维积分的计算原则上可以通过化成累次积分来进行, 例如, 对于积分

$$I = \int_\Omega g(x,y,z)\,dxdydz \tag{2.1.26}$$

其中Ω 为积分区域, 总可以化为

$$I = \int_a^b dx \int_{c(x)}^{d(x)} dy \int_{e(x,y)}^{f(x,y)} g(x,y,z) \tag{2.1.27}$$

这相当于依次计算三个一重积分, 每一积分以其前一积分作为被积函数. 稍加考虑就可发现, 这一过程的计算量随积分重数的增加而变得无法完成. 如每重积分取10个点(这是一个非常小的数字), 则n重积分就要计算10^n个函数值. 因此, 这一方法只能用于积分重数较低的积分.

高维积分还可以用高斯方法计算, 这只要把每重积分用高斯积分公式代入即可. 对于$n>3$的高维积分, 计算量总是相当大的, 如果你对计算精度要求不高, 那么可以利用后面将要讲到的Monte Carlo 方法. 另外, 如果可能, 请尽可能地解析积出内层积分, 以减小计算量.

2.1.7 固体热容量的计算

在德拜近似下, 固体的热容量由下式给出

$$C_V = 9Nk\left(\frac{T}{\Theta_D}\right)^3 \int_0^{\Theta_d/T} \frac{\xi^4 e^\xi}{(e^\xi-1)^2} d\xi$$

式中Θ_D 为德拜温度. 为了求得热容量, 需要计算上式中的积分, 这一积分只有当$\Theta_d/T \ll 1$ 或$\Theta_d/T \gg 1$ 时才能解析积出, 一般情况下只能进行数字计算. 考虑

$$f(x) = \int_0^x \frac{\xi^4 e^\xi}{(e^\xi-1)^2} d\xi$$

利用我们学到的方法, 可以对不同的x求出其值, 并做出$f(x)$的表或图形. 为了计算积分, 注意到$\xi = 0$是被积函数的可去奇点, 因此, 应在该点把被积函数展开为ξ的级数; 当$\xi \to \infty$, 被积函数指数减小. 已知

$$f(\infty) = \int_0^\infty \frac{\xi^4 e^\xi}{(e^\xi - 1)^2} d\xi = \frac{4}{15}\pi^4$$

我们可以根据x的值的大小分别用两种方法计算, 取一值x_c, 当$x < x_c$时, 用原式计算, 当$x > x_c$时, 可以计算$f(\infty) - f_1(x)$, 其中

$$f_1(x) = \int_x^\infty \frac{\xi^4 e^\xi}{(e^\xi - 1)^2} d\xi$$

x_c可以选择的大一点, 这样$f_1(x)$的数值比较小, 计算精度要求可以低一些.

我们选择$x_c = 10$, $x < x_c$时, 用辛普生方法计算$f(x)$, 被积函数为

$$\begin{cases} \xi^2 - \frac{\xi^4}{12} + \frac{\xi^6}{240} & 0 \le \xi < 0.05 \\ \frac{\xi^4 e^\xi}{(e^x - 1)^2} & 0.05 \le \xi \le x_c \end{cases}$$

当$x > x_c$ 时, 计算$f_1(x)$, 对积分作变量变换$\xi = \eta + x$ 则

$$f_1(x) = e^{-x} \int_0^\infty \frac{(\eta + x)^4 e^{-\eta}}{(1 - e^{-(\eta + x)})^2} d\eta$$

这一积分可用高斯—拉盖尔方法计算. 作为练习, 请同学完成余下的计算并作出$f(x) \sim x$的图形

2.2 方程的求根和最优化方法

方程的求根问题是物理计算中最常遇到的问题之一, 对于低次代数方程, 已经找到了解的公式, 但除了二次方程之外, 这些公式并无多少实用价值, 而对于即使如$\cos(x) - x = 0$这样简单的方程, 也无法找到解析形式的解. 数值计算几乎是求解任何有实际意义的问题的唯一方法. 在这一节中, 我们将讨论几种求解$f(x) = 0$的根的数字方法. 最优化方法除了本身的重要性之外, 也是一种求解多元方程的有效方法, 因此, 最优化方法的介绍也是本节的一个重要内容.

2.2.1 二分法

二分法是求解方程的最安全, 也是概念上最简单的方法, 它可以做为后面将要讲到的方法失败后的最后保险.

假定我们知道在区间$a < x < b$ 之内只有$f(x) = 0$ 的一个根, 则显然$f(a)$与$f(b)$异号, 把区间分半, 即令$c = (b + a)/2$, 计算$f(c)$的值, 若$f(a)$与$f(c)$同号, 则根在$[c, b]$ 之间, 否则, 根在$[a, c]$之间, 对根所在的区间重复上述过程直到达到所需精度.

2.2.2 牛顿法和割线法

在所有一元函数求根方法中, 牛顿法也许是最值得推荐的一个. 其想法是这样的, 假定$f(x)=0$ 的根为x^*, 而我们有一个对x^* 的猜测值x_n, $f(x)$ 可在x_n 附近展开为泰勒级数:

$$f(x)=f(x_n)+f'(x_n)(x-x_n)+\cdots$$

略去高次项, 令上式为0, 可解出x, 将此值作为x^* 的一个更好的近似, 记为x_{n+1}, 则

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}. \tag{2.2.28}$$

这样, 给定一个x^* 的初始值x_0, 利用下面的迭代公式反复迭代:

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)} \tag{2.2.29}$$

如果迭代收敛, 则必收敛于$f(x)=0$ 的一个根x^*. 这一迭代公式(2.2.29)就是著名的牛顿迭代公式.

可以证明, 如果迭代到第n步时的误差为$\epsilon_n=x_n-x^*$, 则第$n+1$步的误差为$\epsilon_{n+1}=\epsilon_n^2\frac{f''(x^*)}{2f'(x^*)}$, 也就是说, 牛顿法是平方收敛的. 证明如下: 由迭代公式(2.2.29),

$$\begin{aligned}
x_{n+1} &= x_n-\frac{f(x_n)}{f'(x_n)}\\
&\approx x^*+\epsilon_n-\frac{\epsilon_n f'(x^*)+\frac{1}{2}\epsilon_n^2 f''(x^*)}{f'(x^*)+\epsilon_n f''(x^*)}\\
&\approx x^*+\epsilon_n-\left(\epsilon_n+\frac{1}{2}\frac{f''(x^*)}{f'(x^*)}\epsilon_n^2-\frac{f''(x^*)}{f'(x^*)}\epsilon_n^2\right)\\
&= x^*+\frac{1}{2}\frac{f''(x^*)}{f'(x^*)}\epsilon_n^2
\end{aligned}$$

二分法是线性收敛的, 请证明之. 但是, 另一方面, 对于任一初始值, 牛顿法并不总是收敛, 事实上, 牛顿法的收敛范围是较小的, 因此, 在实际使用时, 必须找一个较好的x_0, 使x_0尽可能地靠近x^*, 这可以把二分法与牛顿法结合起来以完成.

试计算$\sqrt{10}$.

令$f(x)=x^2-10$, $f(x)=0$的根就是所求结果。取初始值$x_0=3$, $f'(x)=2x$.

迭代结果如下：

3.

3.166666666666667

3.162280701754386

3.162277660169842

3.162277660168380

3.162277660168379

牛顿法的一个重要的缺点是需要计算函数的导数, 对于一些复杂的函数来说, 这是一件十分麻烦的工作. 为了即保持较高的收敛速度, 又避免麻烦的导数计算, 可以对牛顿法稍作变形, 用差分代替导数, 从而得到下面的割线法.

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{2.2.30}$$

这一迭代方式需要二个初始值x_0, x_1, 可以取为所猜测的解附近的二个点. 割线法的收敛速度比牛顿法稍慢, 但省去了导数的计算, 对于求导数比较困难的函数特别有用. 作为一个编程练习, 请写出这一算法的程序并作为你自己的库程序.

练习*:*
证明割线法的收敛阶为$\epsilon_{n+1} \sim \epsilon_n^{1.618\cdots}$

另一个不需要计算导数的方法是所谓的Stewenson方法, 算法如下:

$$x_{n+1} = x_n - \frac{f(x_n)^2}{f(x_n) - f(x_n - f(x_n))} \tag{2.2.31}$$

对于单零点, 这一算法的收敛阶为2,(请证明), 而且不需要计算导数, 是一个较好的算法.

试计算$\sqrt{10}$.
令$f(x) = x^2 - 10$, $f(x) = 0$的根就是所求结果。取初始值$x_0 = 3$. 用*Stewenson*法迭代结果如下：

3.
3.142857142857143
3.161965423111920
3.162277578113566
3.162277660168374
3.162277660168380

2.2.3 一维最优化, 黄金分割法

这一节开始我们讨论最优化方法, 或计算一目标函数的极值的方法. 最优化方法在物理学中有十分广泛的应用, 函数的求根问题也可化为一个最优化问题, 例如, 求函数$f(x) = 0$的问题就与求目标函数为$F(x) = f(x)^2$的极小值的问题等价, 而方程组

$$f_i(x_1, x_2, x_3, \cdots, x_n) = 0, \quad i = 1, 2, 3, \cdots, n$$

的求解问题则与

$$F(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} w_i f_i(x_1, x_2, \cdots, x_n)^2$$

的最优化问题等价. 这里w_i, $i = 1, 2, \cdots, n$ 为一组权重参数, 适当的选则可使计算更为容易进行, 但对根的位置没有影响.

这一节我们先考虑一个自变量的目标函数的最优化问题. 假定我们有一目标函数$F(x)$, 我们的任务是寻找一个$x = x^*$, 使得目标函数$F(x)$最小. 这一工作可以分两步来完成, 首先, 我们要找出目标函数的极小点所在的大致位置, 或者更确切一些, 我们要找三个点$a < b < c$, 使得目标函数$F(b) < F(a)$, $F(b) < F(c)$. 第二步则在区间$[a, c]$内寻找$F(x)$的最小值.

第一步原则上是十分简单的, 假定有两点a, b, 我们可以比较$F(a)$ 和$F(b)$ 以确定一个函数减小的方向, 不失一般性, 假定$F(b) < F(a)$, 然后沿减小方向前进一步, 到达新的一点c, 一般取$c = b + 1.618(b - a)$, 如果$F(c) < F(b)$, 则令$a = b$, $b = c$, 继续前进, 寻找新的c; 如果$F(c) > F(b)$, 则我们的目的已经达到.

下面考虑第二步的计算, 即从三个点$a < b < c$, $F(b) < F(a)$, $F(b) < F(c)$出发, 寻找$F(x)$ 的极小点x_{min}. 这里介绍黄金分割法. 寻找极小点的过程, 实际上就是用一种方法不断缩小区间$[a, c]$, 并保证极小点一直位于区间之内. 这一过程可以这样来完成, 假定已经作了n步, 得到三个点a, b, c和对应的目标函数值$F(a) > F(b)$, $F(b) < F(c)$, 不失一般性, 假定$c - b > b - a$, 在区间$[b, c]$内选一点x, 如果$F(x) < F(b)$, 则作代换$a = b$, $b = x$, 新的缩小了的区间为$[a, c]$(即原$[b, c]$, 丢掉的区间为原$[a, b]$); 如果$F(x) > F(b)$, 则作代换$c = x$, 新的缩小了的区间为$[a, c]$(即原$[a, x]$, 丢掉的区间为原$[x, c]$). 问题在于如何选择x, 一个自然的想法是, 不论$F(x) < F(b)$ 或$F(x) > F(b)$, 丢掉的区间都相同, 即选择x, 使得

$$b - a = c - x \equiv w(c - a) \tag{2.2.32}$$

这里w 是一个比例因子, 我们现在来确定它. 由于要求每一步都满足上述关系, 显然有

$$x - b = w(c - b) = w((c - a) - (b - a)) = w(c - a) - w^2(c - a) = (w - w^2)(c - a) \tag{2.2.33}$$

另一方面, 由(2.2.32)可得

$$(b - a) + (c - x) = (c - a) - (x - b) = 2w(c - a) \tag{2.2.34}$$

即

$$x - b = (1 - 2w)(c - a) \tag{2.2.35}$$

由(2.2.33)和(2.2.35), 我们得到:

$$w^2 - 3w + 1 = 0 \tag{2.2.36}$$

由此解得,

$$w = \frac{3 - \sqrt{5}}{2} \approx 0.38197, \quad 1 - w \approx 0.61803$$

$1 - w$ 通常被称为黄金分割数或黄金数. 一般在开始计算时, a, b, c三者并不满足黄金分割关系, 但几次迭代后, 这一关系便可满足, 并在其后一直保持.

2.2.4 高维最优化, 共扼梯度方法

这一节给出多维问题的最优化方法. 对于一个含有n个自变量的目标函数$F(x_1, x_2, \cdots x_n)$, 我们的目的是要求出一组$(x_1^*, x_2^*, \cdots, x_n^*)$ 使目标函数达到极小. 为了讨论问题的方便, 下面用$\mathbf{x}$ 代表n 维空间的一个点, 最优化问题也可以表述为, 寻找n 维空间的一点$\mathbf{x}^*$ 使得目标函数$F(\mathbf{x}^*)$ 最小.

高维最优化问题有很多求解方法, 其基本思路是相同的, 这就是从一点$\mathbf{x}_0$ 出发, 按照某种规定的方向$\mathbf{p}_0$, 用一维最优化的方法寻找函数$F(\mathbf{x})$ 的极小点$\mathbf{x}_1$, 继续这一过程直到达到最优点. 若第i步到达$\mathbf{x}_i$, 设选定方向$\mathbf{p}_i$, 则新的$\mathbf{x}_{i+1}$ 可如下求得. 实际上, 我们只要找一参数t_i, 使得

$$F(\mathbf{x}_i + t_i\mathbf{p}_i) = \min_t F(\mathbf{x}_i + t\mathbf{p}_i) \tag{2.2.37}$$

则$\mathbf{x}_{i+1} = \mathbf{x}_i + t_i\mathbf{p}_i$便为所求. 因此, 高维最优化的的不同方法实际上是选择不同的一维寻找方向. 一个直观的推断就是在每一点选择该点目标函数下降最快的方向

$$\mathbf{p}_i = -\nabla F(\mathbf{x}_i) \tag{2.2.38}$$

作为寻找方向, 这一方法称为最陡下降法. 直观的想象可能会认为最陡下降方向是一种理想的寻找方向, 但事实并非如此, 最陡下降方向只表示在$\mathbf{x}_i$附近的下降性质, 而对整个最优化过程来说, 我们寻找的是全局的最优方向. 作为一个例子, 考虑如下函数的最优化:

$$F = x^2 + 10y^2 \tag{2.2.39}$$

显然, 这一函数的极小点是$(0, 0)$. 函数(2.2.39)的负梯度为

$$-\mathbf{g} = -(2x, 20y)^T \tag{2.2.40}$$

如果从点$(2, 1)$出发, 利用最陡下降法计算所得逐$\mathbf{x}$如下:

$(2, 1)$, $(1.792829, -0.035857)$, $(0.461013, 0.230507)$,
$(0.413259, -0.008265)$, $(0.106267, 0.053133)$, $(0.095259, -0.001905)$,
$(0.024495, 0.012248)$, $(0.021958, -0.000439)$, $(0.005646, 0.002823)$,
$(0.005061, -0.000101)$, $(0.001302, 0.000651)$, $(0.001167, -0.000023)$,
$(0.000300, 0.000150)$, $(0.000269, -0.000005)$, $(0.000069, 0.000035)$,
$(0.000062 - 0.000001)$.

显然, 收敛是很慢的.

最陡下降方向并不是理想的下降方向, 为了改进最优化的收敛速度, 需要寻求其他方向. 为了判断一个寻找方向的好坏, 应该有一个标准, 注意到一般函数在最小点附近近似于二次函数, 因此如果一个方法对二次函数比较有效, 则可望其对一般函数也比较有效, 如果一个方法对二次函数的效果都不好的话, 很难期望它对一般函数会有好的效果. 计

算实践也证实了这一论断. 下面, 我们就从二次函数寻求一种比较有效的下降方向. 为此, 我们先给出几个数学定理;

定理一: 若N维欧几里德空间中的向量$\mathbf{q}$与N个线性独立向量

$$\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_N}$$

都正交, 则向量$\mathbf{q}$必为0.

这一定理的结论是显然的.

定义: A**共轭向量**. 设A是一个$N \times N$的对称正定矩阵, $\mathbf{p}$, $\mathbf{q}$ 是两个N维向量, 若,

$$\mathbf{p}^T A\mathbf{q} = 0 \tag{2.2.41}$$

则称向量$\mathbf{p}$和向量$\mathbf{q}$互为A共轭或互为A正交.

定理二: 若A为$N \times N$对称正定矩阵, $\mathbf{p}_1$, $\mathbf{p}_2$, $\cdots$, $\mathbf{p}_N$为A共轭的N维非零向量, 则此向量组必为线性独立.

证明: 设向量组$\mathbf{p}_1$, $\mathbf{p}_2$, $\cdots$, $\mathbf{p}_N$之间存在线性关系

$$\alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \cdots + \alpha_N\mathbf{P}_N = 0$$

对$i = 1, 2, \cdots, N$, 用$\mathbf{p}_i^T A$左乘上式得

$$\alpha_i\mathbf{p}_i^T A\mathbf{p}_i = 0$$

因为$\mathbf{p}_i \neq 0$, A正定, 所以

$$\mathbf{p}_i^T A\mathbf{p}_i > 0$$

从而必有

$$\alpha_i = 0, \quad i = 1, 2, \cdots, N$$

因此, 向量组$\mathbf{p}_1$, $\mathbf{p}_2$, $\cdots$, $\mathbf{p}_N$线性独立.

有了以上数学准备, 我们来考虑二次函数的最优化问题, 我们的目的是找一种下降方向, 能够在有限步达到二次函数的极小点. 考虑二次N元函数

$$F(\mathbf{x}) = a + \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T A\mathbf{x} \tag{2.2.42}$$

其中A为一正定矩阵. 函数(2.2.44)的梯度为

$$\mathbf{G}(\mathbf{x}) = \mathbf{b} + A\mathbf{x} \tag{2.2.43}$$

记$\mathbf{x}_i$点的梯度为$\mathbf{g_i} = \mathbf{G}(\mathbf{x}_i)$, 从$\mathbf{x}_0$点出发, 对于第一个寻找方向, 除梯度外没有别的信息, 我们就取$\mathbf{p}_0 = -\mathbf{g}_0$, 沿此方向找到极小点$\mathbf{x}_1$并可求得这一点的梯度$\mathbf{g}_1$, 因为$\mathbf{x}_1$是沿$\mathbf{p}_0$ 方向的极小点, 这一点的梯度方向一定与$\mathbf{p}_0$垂直，$\mathbf{g}_1 \perp \mathbf{p}_0$, 从而$\mathbf{g}_1 \perp \mathbf{g}_0$. 现在有了两个方

向，$\mathbf{g}_1$和$\mathbf{g}_0$, 最陡下降法选取新的方向为$-\mathbf{g}_1$. 我们保留原来方向的一点信息，选新的寻找方向为这两个方向的组合

$$\mathbf{p}_1 = -\mathbf{g}_1 - \alpha_0 \mathbf{g}_0 \tag{2.2.44}$$

并要求$\mathbf{p}_1$ 与$\mathbf{p}_0$ 为A共轭, 则应有

$$(-\mathbf{g}_1 - \alpha_0 \mathbf{g}_0)^T A \mathbf{p}_0 = 0 \tag{2.2.45}$$

由于

$$\mathbf{x}_1 = \mathbf{x}_0 + t_0 \mathbf{p}_0$$

则

$$\mathbf{g}_1 - \mathbf{g}_0 = A(\mathbf{x}_1 - \mathbf{x}_0) = t_0 A \mathbf{p}_0$$

一般, 若第i步的寻找方向为$\mathbf{p}_i$, 则同理可证

$$\mathbf{g}_{i+1} - \mathbf{g}_i = A(\mathbf{x}_{i+1} - \mathbf{x}_i) = t_i A \mathbf{p}_i \tag{2.2.46}$$

因此,方程(2.2.45)成为

$$(-\mathbf{g}_1 - \alpha_0 \mathbf{g}_0)^T (\mathbf{g}_1 - \mathbf{g}_0) = 0 \tag{2.2.47}$$

由此解得

$$\alpha_0 = \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0} \tag{2.2.48}$$

这里假定$\mathbf{g}_0 \neq 0$, 否则, 若$\mathbf{g}_0 = 0$，则$\mathbf{x}_0$就是所求的极值点.

把α_0代入(2.2.44)，便得到$\mathbf{p}_1$. 沿$\mathbf{p}_1$求出极小点$\mathbf{x}_2$并算出$\mathbf{g}_2$. 由于$\mathbf{p}_0$和$\mathbf{p}_1$为A共轭, 因此由(2.2.46)

$$\mathbf{g}_0^T (\mathbf{g}_2 - \mathbf{g}_1) = 0$$

考虑到$\mathbf{g}_0$与$\mathbf{g}_1$正交, $\mathbf{g}_0^T \mathbf{g}_1 = 0$, 由上式可知$\mathbf{g}_0$与$\mathbf{g}_2$也正交. 因$\mathbf{x}_2$是沿$\mathbf{p}_1$ 方向的极小点，故$\mathbf{g}_2 \perp \mathbf{p}_1$, 也就是

$$\mathbf{g}_2^T (-\mathbf{g}_1 - \alpha_0 \mathbf{g}_0) = 0$$

即$\mathbf{g}_2 \perp \mathbf{g}_1$. $\mathbf{g}_0$, $\mathbf{g}_1$和$\mathbf{g}_2$构成一个正交矢量组, 我们可以在这个正交组构成的三维空间寻求与$\mathbf{p}_0$, $\mathbf{p}_1$均为A 共轭的寻找方向$\mathbf{p}_2$. 令

$$\mathbf{p}_2 = -\mathbf{g}_2 - \alpha_1 \mathbf{g}_1 - \beta_0 \mathbf{g}_0 \tag{2.2.49}$$

要求$\mathbf{p}_2$分别与$\mathbf{p}_0$和$\mathbf{p}_1$为A共轭, 也就是要求如下方程成立

$$(-\mathbf{g}_2 - \alpha_1 \mathbf{g}_1 - \beta_0 \mathbf{g}_0)^T (\mathbf{g}_1 - \mathbf{g}_0) = 0 \tag{2.2.50}$$

$$(-\mathbf{g}_2 - \alpha_1 \mathbf{g}_1 - \beta_0 \mathbf{g}_0)^T (\mathbf{g}_2 - \mathbf{g}_1) = 0 \tag{2.2.51}$$

$$\tag{2.2.52}$$

由此解出

$$\alpha_1 = \frac{\mathbf{g}_2^T\mathbf{g}_2}{\mathbf{g}_1^T\mathbf{g}_1}, \quad \beta_0 = \alpha_0\alpha_1. \tag{2.2.53}$$

于是

$$\mathbf{p}_2 = -\mathbf{g}_2 + \alpha_1\mathbf{p}_1 \tag{2.2.54}$$

现假定已求得$\mathbf{p}_0$, $\mathbf{p}_1$, $\cdots$, $\mathbf{p}_j$, 其中

$$\begin{aligned} \mathbf{p}_0 &= -\mathbf{g}_0 \\ \mathbf{p}_i &= -\mathbf{g}_i + \alpha_{i-1}\mathbf{p}_{i-1} \\ \alpha_{i-1} &= \mathbf{g}_i^T\mathbf{g}_i/\mathbf{g}_{i-1}^T\mathbf{g}_{i-1} \\ & \quad i = 1, 2, \cdots, j \end{aligned}$$

这里假定$\mathbf{g}_0$, $\mathbf{g}_1$, $\cdots$, $\mathbf{g}_j$都不为零且构成一个正交系. 沿$\mathbf{p}_j$求出$\mathbf{x}_{j+1}$并算出$\mathbf{g}_{j+1} = \mathbf{G}(\mathbf{x}_{j+1})$, 设$\mathbf{g}_{j+1} \neq 0$, 则仿前可证, $\mathbf{g}_0$, $\mathbf{g}_1$, $\cdots$, $\mathbf{g}_{j+1}$ 也构成正交系. 令

$$\mathbf{p}_{j+1} = -\mathbf{g}_{j+1} - \beta_0\mathbf{g}_0 - \beta_1\mathbf{g}_1 - \cdots - \beta_{j-1}\mathbf{g}_{j-1} - \alpha_j\mathbf{g}_j \tag{2.2.55}$$

并要求$\mathbf{p}_{j+1}$与$\mathbf{p}_0$, $\mathbf{p}_1$, $\cdots$, $\mathbf{p}_j$为A共轭, 即

$$(-\mathbf{g}_{j+1} - \beta_0\mathbf{g}_0 - \beta_1\mathbf{g}_1 - \cdots - \beta_{j-1}\mathbf{g}_{j-1} - \alpha_j\mathbf{g}_j)^T(\mathbf{g}_{j+1} - \mathbf{g}_j) = 0 \tag{2.2.56}$$

得到

$$\begin{aligned} \alpha_j &= \mathbf{g}_{j+1}^T\mathbf{g}_{j+1}/\mathbf{g}_j^T\mathbf{g}_j, \\ \beta_{j-1} &= \alpha_j\alpha_{j-1}, \\ & \cdots\cdots \\ \beta_1 &= \alpha_j\alpha_{j-1}\cdots\alpha_1, \\ \beta_0 &= \alpha_j\alpha_{j-1}\cdots\alpha_1\alpha_0. \end{aligned}$$

且

$$\begin{aligned} -\beta_0\mathbf{g}_0 - \beta_1\mathbf{g}_1 &- \cdots - \beta_{j-1}\mathbf{g}_{j-1} - \alpha_j\mathbf{g}_j \\ &= -\alpha_j\cdots\alpha_1(\alpha_0\mathbf{g}_0 + \mathbf{g}_1) - \beta_2\mathbf{g}_2 - \cdots - \alpha_j\mathbf{g}_j \\ &= \alpha_j\cdots(\alpha_1\mathbf{p}_1 - \mathbf{g}_2) - \cdots - \alpha_j\mathbf{g}_j \\ &= \cdots \\ &= \alpha_j\mathbf{p}_j \end{aligned}$$

即

$$\mathbf{p}_{j+1} = -\mathbf{g}_{j+1} + \alpha_j \mathbf{p}_j, \quad \alpha_j = \mathbf{g}_{j+1}^T \mathbf{g}_{j+1} / \mathbf{g}_j^T \mathbf{g}_j \tag{2.2.57}$$

根据归纳法, 在(2.2.57)中令$j = 0, 1, 2, \cdots$, 所得方向为A 共轭方向. 上述方法最多只要N步就可找到函数(2.2.44)的极小点, 这是因为所有的$\mathbf{g}_j$互相正交, 即$\mathbf{g}_N^T \mathbf{g}_j = 0$对所有$j = 0, 1, 2, \cdots, N-1$都成立, 而我们的问题在$N$维空间, 故必有$\mathbf{g}_N = 0$, 亦即$\mathbf{x}_N$ 为函数(2.2.44)的极小点.

这就是所谓共轭梯度方法, 理论上, 对于二次函数, 只要N次迭代就可以达到极小点, 但实际计算时, 由于舍入误差的影响, 一般N次迭代并不能达到极小点. 对于一般的目标函数, 有限次(N次)的迭代也一般不能达到极小点, 因此实际计算时, 从一初始点出发, 迭代N次后, 以所得结果为出发点, 重新开始计算, 直到收敛为止.

共轭梯度方法由于占用内存小, 方法简单, 编程方便, 目前在物理问题中应用比较多.

作为例子, 我们再考虑前面的例题, 对函数$x^2 + 10y^2$, 取$\mathbf{x}_0 = (2, 1)^T$, 则$\mathbf{g}_0 = (4, 20)^T$, 取$\mathbf{p}_0 = -\mathbf{g}_0$, 求得$\mathbf{x}_1 = (1.79283, -0.0358566)^T$, $\mathbf{g}_1 = (3.58566, -0.717131)^T$, $\alpha_0 = 0.0321423$, 由此可构造出$\mathbf{p}_1 = (-3.71423, 0.0742845)^T$, 沿$\mathbf{p}_1$方向求得$\mathbf{x}_2 = (0.000000, 0.000000)^T$, 即已经达到极小点.

习题*:*

编制共轭梯度法的程序并求下面函数的极小值点及极小值,

$$F(x_1, \cdots, x_n) = 1 + \sum_{i=2}^{n} \left[100(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2\right] \tag{2.2.58}$$

取$n = 10$, 选择初始点为$x_i = i/10$, $i = 1, 2, \cdots, 10$.

2.2.5 约束最优化

前面几节所讲的是无约束最优化问题, 在实际物理问题中, 还经常会遇到有约束最优化问题, 其数学提法是, 给定一目标函数$F(x_1, x_2, \cdots, x_n)$, 在一定的约束下求解其极小值, 约束一般写成一组函数. 即

$$\begin{cases} F(x_1, x_2, \cdots, x_n) = \text{Min} \\ G_i(x_1, x_2, \cdots, x_n) = 0 \qquad (i = 1, 2, \cdots, m) \end{cases} \tag{2.2.59}$$

式中$m < n$, m 个方程$G_i(x_1, x_2, \cdots, x_n)$ 代表m 个约束. 解决约束优化问题的通常做法是构造一个新的函数, 把约束优化问题改为一无约束最优化问题, 然后利用前面的方法求解. 通常的做法是, 定义

$$F' = F + \sum_{i=1}^{m} \lambda_i G_i^2 \tag{2.2.60}$$

对于合适选择的λ_i, $(i = 1, 2, \cdots, m)$, F' 的极小值就代表了由方程(2.2.59)定义的约束最优化问题.

例题 在参数x_1, x_2, $\cdots$, x_n 的一个限制区域D内求解F 的极小值. 为此, 我们把求解区域扩展到整个区域, 如果F 在D外无定义, 则应扩充其定义. 同时定义函数G为

$$G = \begin{cases} 0 & \{x\} \in \mathrm{D} \\ \sum_i \lambda_i x_i^2 & \{x\} \notin \mathrm{D} \end{cases} \tag{2.2.61}$$

则通过计算$F + G$的极小值就可得到问题的解. 在使用这种方法时应注意如果在D外F会变小时, G的选择应使得$F + G$在D外总是较快增加的. 式(2.2.61) 给出的仅仅是一种可能的选择, 实际上G的选择可以有无穷多种.

2.2.6 最小二乘法及曲线拟合

在物理实验中, 常常要测量一对物理量之间的关系, 在有些情况下, 一对物理量之间的函数关系是已知的, 但包含若干个未知参数, 若用x和y代表这一对物理量, 则有

$$y = f(x; a_1, a_2, \cdots, a_m) \tag{2.2.62}$$

式中a_1, a_2, $\cdots$, a_m 为一组未知参量, f的函数形式是已知的, 在这种情况下, 曲线拟合的任务, 就是根据(x, y)的N个观测点

$$(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$$

来确定参数a_i, $(i = 1, 2, \cdots, m)$, 如果测量中没有误差, 则每一组(x_i, y_i) 都应准确地落在理论曲线上, 即应有

$$y_i = f(x_i; a_1, a_2, \cdots, a_m) \quad (i = 1, 2, \cdots, N) \tag{2.2.63}$$

为了确定诸a_i 的值, 只要选出(2.2.63)中的m 个方程求解即可, 但实际测量总是有误差的, (x_i, y_i) 并不准确落在理论曲线上, 在$N > m$ 的情况下, 方程组(2.2.63) 成为矛盾方程组, 不能用解方程的方法求参数$a_1, a_2, \cdots, a_m$, 只能用曲线拟合的方法根据带有误差的观测值点求得理论曲线参数的估计值.

在很多实际问题中, x和y这两个物理量中总有一个量的测量精度比另一个高得多, 其测量误差可以忽略. 把可以忽略测量误差的量选作自变量x, 其观测值可看作是准确的, 对于每个x_i, 对应的y 的观测值y_i 是一个随机变量(见第5章), 其误差可由其方差来估计.

物理中还经常遇到另外一种曲线拟合的任务: 物理量y 和x 间的函数形式未知, 需要从观测点求出y 和x 的函数关系的一个经验公式. 一般的作法是选取某一函数族{ $\phi_i(x)$ } 的一个线性组合,

$$y = \sum_i a_i \phi_i(x) \tag{2.2.64}$$

然后用曲线拟合方法求出系数a_i. 函数族{ $\phi_i(x)$ } 通常从物理考虑来选取, 一种常用的方法是选取{ 1, x, x^2, $\cdots$, x^{m-1} } 用于拟合m 个参数; 也可选取三角函数{ 1, $\sin x$, $\cos x$, $\sin 2x$, $\cos 2x$, $\cdots$, $\sin 2mx$, $\cos 2mx$} 用于拟合$2m+1$ 个参数. 这种方法称为线性拟合方法, 即y 线性地依赖于参数a_i, 对于有些物理问题, y和x 的函数关系中非线性地依赖于参数a_i, 例如

$$y = a_1 e^{-a_2 x} + a_3 e^{-a_4 x} \sin(a_5 x + a_6)$$

就是一个具有6个参数的非线性拟合问题.

曲线拟合通常采用最小二乘法, 如果x 的测量误差可以忽略, 观测点i 的残差δ_i 定义为在该点y的观测值与理论值之差, 即

$$\delta_i = y_i - f(x_i; a_1, a_2, \cdots, a_m) \tag{2.2.65}$$

若观测值y_i 的方差为σ_i^2, 则定义其权重因子为

$$w_i = \frac{1}{\sigma_i^2}$$

记χ^2 为观测点残差的加权平方和,

$$\chi^2 = \sum_{i=1}^{N} w_i \delta_i^2 = \sum_{i=1}^{N} w_i (y_i - f(x_i; a_1, a_2, \cdots, a_m))^2 \tag{2.2.66}$$

曲线拟合的最小二乘法就是以χ^2 为目标函数的优化问题, 即寻找一组参数a_i, $(i = 1, 2, \cdots, m)$ 使χ^2 为最小. 在观测点的方差未知的情况下, 可以取$w_i = 1$, 也可以在每一观测点多测量几次, 以其平均值$\langle y_i \rangle$作为该点的观测值, 而以$\langle y_i^2 \rangle - \langle y_i \rangle^2$ 作为其方差.

在曲线拟合问题中, 最困难的是选取合适的经验公式, 这需要结合物理考虑和对测量数据的观察, 也依赖于对各种函数及其图像的明确概念.

例题 在某一能谱测量中, 得到下表所示数据, 试给出经验公式并作出曲线拟合.

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6
y	2.117	2.634	3.459	4.671	6.600	9.978	15.893
x	0.7	0.8	0.9	1.0	1.1	1.2	1.3
y	25.108	30.979	25.768	17.363	12.031	10.454	10.280
x	1.4	1.5	1.6	1.7	1.8	1.9	
y	11.012	11.138	9.745	7.576	5.598	4.165	

把表中数据点到坐标纸上(见图2.1中圈点), 可以看出数据有二个峰, 为此, 我们试用Lorentz 形曲线去拟合, 即取经验公式为

$$y = \frac{a_1}{(x - a_2)^2 + a_3^2} + \frac{a_4}{(x - a_5)^2 + a_6^2}$$

用上式和上表中的数据计算χ^2, 并用共轭梯度方法优化χ^2, 求得各参数值为

$$a_1 = 1.210 \quad a_2 = 0.800 \quad a_3 = 0.202$$
$$a_4 = 0.776 \quad a_5 = 1.502 \quad a_6 = 0.295$$

在图2.1中, 我们用实线画出了拟合的结果, 虚线给出了二个峰的图像.

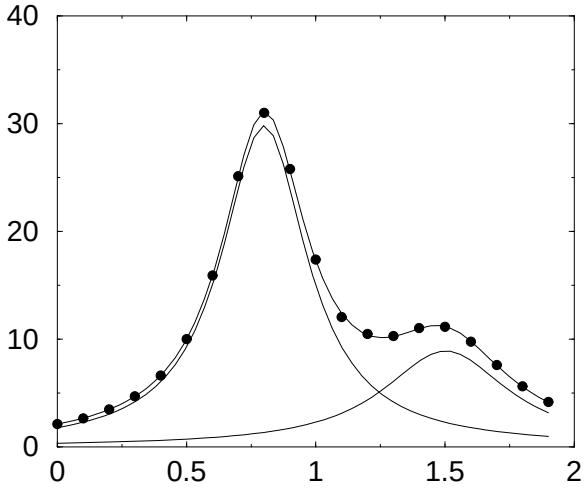


Figure 2.1:

2.3 函数的求值

在数值计算中, 经常要求一些函数的值, 一些常用的初等函数如三角函数, e指数函数, 双曲函数, 对数函数等, 均已写入计算机语言的定义做为内部函数, 在使用时只需调用即可, 而其它的函数和表达式则要自己计算, 本章将给出一些在物理计算中经常遇到的函数及表达式的计算方法和部分程序. 一般说来, 计算机的内部函数都是由专家设计的, 其效率和精度都很好, 因此, 如果计算机语言已经提供了所需的函数, 则应尽量使用这些函数, 下面提供的方法仅仅作为计算机系统不提供这些函数时的补充.

2.3.1 多项式的求值和级数求和

所谓多项式是指由下式给出的表达式:

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n$$

为了计算$p(x)$在给定x时的数值, 我们用$c(n)$, $n = 0, 1, 2, \cdots, n$ 来代表多项式的系数, 则可以写出计算语句. 以$n = 4$为例, 直接用FORTRAN语言写出的算式为:

```
p=c(0)+c(1)*x+c(2)*x**2+c(3)*x**3+c(4)*x**4
```

这显然不是一个好的算法(为什么?). 较好的算法是下面二者之一:

```
p=c(0)+x*(c(1)+x*(c(2)+ x*(c(3)+x*c(4))))

p=(((c(4)*x+c(3))*x+c(2))*x+c(1))*x+c(0)
```

当n较大时, 可采用下面的语句段:

```
      P=C(N)
      DO 11 J=N-1,1,-1
        P=P*X+C(J)
11 CONTINUE
```

在科研中, 还常常碰到级数求和的问题, 例如计算

$$S = \sum_{i=0}^{\infty} u_i$$

我们当然不可能计算到无穷多项, 所以上述级数必需在某处截断, 如果级数收敛很慢, 就要计算很多项, 一方面占用大量的CPU时间, 更重要的是当计算的项数非常多时, 舍入误差的积累有可能完全破坏整个计算. 为此必须使用加速收敛的技巧. Aitken 算法是常用的加速技巧之一. 它是对相邻的三个部分和进行处理, 得到一个更好的近似. 记部分和为:

$$S_n = \sum_{i=0}^{n} u_i \tag{2.3.67}$$

则由S_{n-1}, S_n, S_{n+1} 可计算出

$$S'_n = S_{n+1} - \frac{(S_{n+1} - S_n)^2}{S_{n+1} - 2S_n + S_{n-1}} \tag{2.3.68}$$

对$n = 1, 2, 3, \cdots$, 先求出S_n, 同时算出S'_n, 一般S'_n更快地收敛于S. 在实际使用时, (2.3.68)最好用写出的式子计算, 一些代数上等价的式子可能导致较大的舍入误差.

Euler 变换

对于由下式定义的交错级数

$$\sum_{s=0}^{\infty}(-1)^s u_s \tag{2.3.69}$$

这里$u_s > 0$. Euler 变换是一个强有力的算法, 它由下式给出(取n为偶数)

$$\sum_{s=0}^{\infty}(-1)^s u_s = u_0 - u_1 + u_2 \cdots - u_{n-1} + \sum_{s=0}^{\infty} \frac{(-1)^s}{2^{s+1}}(\Delta^s u_n) \tag{2.3.70}$$

其中:

$$\begin{aligned} \Delta u_n &\equiv u_{n+1} - u_n \\ \Delta^2 u_n &\equiv \Delta(\Delta u_n) = u_{n+2} - 2u_{n+1} + u_n \\ &\cdots\cdots \end{aligned} \tag{2.3.71}$$

为向前差分. 式(2.3.70)可推导如下:

定义位移算符E为:

$$u_{n+1} \equiv E u_n$$

由于$(1+\Delta)u_n = u_n + u_{n+1} - u_n = u_{n+1} = Eu_n$, 因此有算符等式

$$E = 1 + \Delta$$

而

$$\begin{aligned} & u_0 - u_1 + u_2 - u_3 + \cdots = (1 - E + E^2 - E^3 + \cdots)u_0 = \frac{1}{1+E}u_0 \\ = \quad & \frac{1}{2}\frac{1}{1+\frac{\Delta}{2}}u_0 = \sum_{s=0}\frac{(-1)^s}{2^{s+1}}\Delta^s u_0 \end{aligned}$$

2.3.2 连分式的求值

所谓连分式, 是指由下式给出的表达式,

$$f = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4 + \cfrac{a_5}{b_5 + \cdots}}}}}$$

数学家们对它做过很多研究, 这是一个非常有趣的函数表示方法. 除了上面的表达式外, 人们还经常使用另一种记法.

$$f = b_0 + \frac{a_1}{b_1+}\frac{a_2}{b_2+}\frac{a_3}{b_3+}\frac{a_4}{b_4+}\frac{a_5}{b_5+}\cdots$$

这里, 系数a_n 和b_n 也可以是x 的函数.

由于连分式只能从右向左求值, 因此在具体计算之前, 很难判断一个无限的连分式应该在何处截断, 一种直接的方法就是猜一个截断点, 求出一个值, 再猜一个更大一点的截断点计算, 通过比较不同截断处的结果来判断是否收敛. 这显然不是一个好的方法. 另一

种在实践中常用的方法是用一个分式有理函数来近似连分式, 用f_n 表示计算到a_n和b_n的连分式的值, 则:

$$f_n = \frac{A_n}{B_n} \tag{2.3.72}$$

这里A_n, B_n由下面的递推公式给出:

$$\begin{aligned} A_{-1} &\equiv 1 \qquad & B_{-1} &\equiv 0 \\ A_0 &\equiv b_0 \qquad & B_0 &\equiv 1 \\ A_j &= b_j A_{j-1} + a_j A_{j-2} \quad & B_j &= b_j B_{j-1} + a_j B_{j-2} \quad j = 1, 2, \cdots, n \end{aligned} \tag{2.3.73}$$

练习

试证明递推关系*(2.3.73)*并写出计算机程序

2.4 常微分方程的数值解法

常微分方程在物理学中具有重要意义, 牛顿第二定律就表示为一组二阶常微分方程, 第一章中提到的著名的三体问题就是由九个二阶常微分方程组成的方程组; 在量子力学中和电动力学中, 由薛定谔方程及Maxwell方程分离变量得到的所谓Sturm–Liouville 本征值问题也是二阶常微分方程的求解问题. 在这一章中, 我们将给出几种数值求解常微分方程的常用方法.

2.4.1 Euler方法

对于常微分方程的初值问题

$$\begin{aligned} \frac{dy}{dx} &= f(x, y) \\ y(0) &= \eta \end{aligned} \tag{2.4.74}$$

数值求解的第一步是把方程(2.4.74)变为一个差分方程, 为此, 让x 取一组分立值x_n, $n = 0, 1, 2, \cdots$, $x_0 = 0$, $x_{n+1} = x_n + h$, 这里h 称为积分的步长, 把$y(x)$在x_n处展开, 有

$$y(x_{n+1}) = y(x_n) + h \left.\frac{dy}{dx}\right|_{x=x_n} + \mathcal{O}(h^2) \tag{2.4.75}$$

利用方程(2.4.74), 便可得到

$$y(x_{n+1}) = y(x_n) + hf(x_n, y_n) + \mathcal{O}(h^2) \tag{2.4.76}$$

这就是Euler差分格式, 从$x = 0$出发, 利用这一方程一步一步地向前计算, 就可以得到任一x_n处的函数值$y_n \equiv y(x_n)$. Euler 方法是一个显式方法, 即第$n + 1$步的y_{n+1} 只依赖

于x_n, y_n 的值, 而且每一步只需计算一次$f(x_n, y_n)$ 的值, 这是Euler 方法的优点, 但另一方面, Euler 方法也存在严重的不足, 因为这一方法是一次的(每一步的误差项为h^2 的数量级, $|\Delta y(x_N)| \sim Nh^2 \sim h$) , 所以$h$ 必须取得足够小, 以保证有满意的精度, 但步长太小时, 积分到一预定的x 则要做很多步的计算, 舍入误差的积累将会使计算结果严重失真. 由于这一原因, 在实际计算中几乎不使用Euler方法, 而常常利用其推导上的简单和概念上非常清楚而用于教学.

2.4.2 龙格—库塔方法

具有二阶以上精度的直接方法称为龙格—库塔方法. 做为一种训练, 我们在此给出二阶龙格—库塔方法的推导. 下面先给出一种比较直接的推导, 然后再介绍一种易于推广到高阶情形的较为系统的推导方法, 希望同学仔细体会这里介绍的方法和思路, 以求举一反三, 为自己设计算法做准备.

第一种推导方法: 如果计算$x_{n+\frac{1}{2}} \equiv x_n + \frac{h}{2}$ 处y 的一阶导数, 可以得到

$$y'_{n+\frac{1}{2}} = \frac{y_{n+1} - y_n}{h} + \mathcal{O}(h^2) \tag{2.4.77}$$

这可用Taylor 展开加以验证. 把上式稍加整理, 并代入$y'_{n+\frac{1}{2}} = f(x_{n+\frac{1}{2}}, y_{n+\frac{1}{2}})$, 得到

$$y_{n+1} = y_n + hf(x_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}) + \mathcal{O}(h^3) \tag{2.4.78}$$

上式中出现了半个分点处的值, 我们必须设法用分点n和$n+1$处的量来表示$n + \frac{1}{2}$ 处的量, 为此, 把$f(x_{n+\frac{1}{2}}, y_{n+\frac{1}{2}})$ 和$f(x_{n+1}, y_{n+1})$ 在n 点展开:

$$\begin{aligned} f(x_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}) &= f(x_n, y_n) + \frac{h}{2}\frac{df}{dx}\bigg|_{x_n} + \mathcal{O}(h^2) \\ f(x_{n+1}, y_{n+1}) &= f(x_n, y_n) + h\,\frac{df}{dx}\bigg|_{x_n} + \mathcal{O}(h^2) \end{aligned} \tag{2.4.79}$$

把上面的第一式乘以2减去第二式, 整理后得

$$f(x_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}) = \frac{1}{2}(f(x_n, y_n) + f(x_{n+1}, y_{n+1})) + \mathcal{O}(h^2) \tag{2.4.80}$$

注意到

$$y_{n+1} = y_n + hf(x_n, y_n) + \mathcal{O}(h^2)$$

及

$$f(x_{n+1}, y_n + hf(x_n, y_n) + \mathcal{O}(h^2)) = f(x_n + h, y_n + hf(x_n, y_n)) + \mathcal{O}(h^2)$$

我们得到

$$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n)) + \mathcal{O}(h^3) \tag{2.4.81}$$

这就是二阶龙格—库塔格式的一种, 由它可以通过n 点的解给出$n+1$ 点的解. 它显然具有二阶精度.

第二种推导方法: 因为我们的目的是从n 点的解给出$n+1$ 点的解并要求每一步的误差为$\mathcal{O}(h^3)$, 所以可以一般的把差分格式写为

$$y_{n+1} = y_n + h\Delta(x_n, y_n; h) \tag{2.4.82}$$

其中

$$\Delta(x_n, y_n; h) \equiv y_n' + \frac{h}{2}y_n'' + \frac{h^2}{3!}y_n^{(3)} + \cdots$$

如果我们用另一函数$\Phi(x_n, y_n; h)$代替$\Delta(x_n, y_n; h)$, 且使得

$$\Phi(x_n, y_n; h) - \Delta(x_n, y_n; h) = \mathcal{O}(h^p)$$

则可得到p 阶差分格式.

对于$p = 2$, 取

$$\Phi(x, y; h) = c_1 f(x, y) + c_2 f(x + ha_2, y + b_{21}hf(x, y))$$

把上式对h 展开

$$\Phi(x, y; h) = c_1 f(x, y) + c_2 f(x, y) + hc_2\left[a_2\frac{\partial f}{\partial x} + b_{21}\frac{\partial f}{\partial y}f(x, y)\right] + \mathcal{O}(h^2) \tag{2.4.83}$$

另一方面:

$$\Delta(x, y; h) = f(x, y) + \frac{1}{2}h\left[\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}f(x, y)\right] + \mathcal{O}(h^2) \tag{2.4.84}$$

比较式(2.4.83) 和(2.4.84), 只要(2.4.83) 中的系数满足下述关系, 就能达到我们的要求.

$$\begin{cases} c_1 + c_2 = 1 \\ c_2 a_2 = \dfrac{1}{2} \\ c_2 b_{21} = \dfrac{1}{2} \end{cases} \tag{2.4.85}$$

方程(2.4.85) 的解并不唯一, 也就是说, 有无穷多组系数可以满足要求. 为此, 令$c_2 = \alpha$, 则

$$\begin{cases} c_1 = 1 - \alpha \\ a_2 = \dfrac{1}{2\alpha} \\ b_{21} = \dfrac{1}{2\alpha} \end{cases} \tag{2.4.86}$$

取$\alpha = \frac{1}{2}$, 得到

$$c_1 = \frac{1}{2}, \quad a_2 = 1, \quad b_{21} = 1$$

差分格式为

$$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n)) \tag{2.4.87}$$

这与第一种方法得到的结果相同. 若取$\alpha = 1$, 则

$$c_1 = 0, \quad a_2 = \frac{1}{2}, \quad b_{21} = \frac{1}{2}$$

差分格式为

$$y_{n+1} = y_n + h(f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(x_n, y_n)) \tag{2.4.88}$$

当α 取不同值时, 我们还可得到其它形式的二阶差分格式.

计算实践表明, 对于绝大多数简单问题, 四阶龙格—库塔方法是最好的选择. 这一方法的一种常用差分格式可用下面一组公式来表示:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned} \tag{2.4.89}$$

$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + \mathcal{O}(h^5) \tag{2.4.90}$$

实际上, 四阶的龙格—库塔方法的形式并不唯一, 这里给出的是常用的一种. 四阶龙格—库塔方法的推导比较繁琐, 但并不难, 其基本思路与前面推导二阶方法的第二种方法相同, W. C. Gear 的名著*Numerical initial Value Problems in Ordinary Differential Equations* (Englewood Cliffs, N. J. Prentice–Hall, 1971)给出了一般龙格—库塔方法的推导, 同时也列出了其它四阶龙格—库塔方法的形式, 有兴趣的同学可以参考.

上面给出了用四阶龙格—库塔方法积分一步的方法, 我们当然可以利用这一方法, 选定一个步长h, 从初给点开始一步一步地积分到所需的点. 那么, 步长如何选取呢? 选多大才是合适的, 是否每一步都选相同的步长? 为此, 我们需要一个判定精度的方法, 同时, 直观告诉我们, 在函数变化剧烈的地方, 步长应该取得小一点以保证精度, 而在函数变化平缓的地方, 步长可以取的大一点以减小计算量. 对于每一步, 判定步长是否合适的一个简单的方法就是把步长减半, 比较步长为$2h$ 的一步的积分结果和步长为h 的两步的计算结果, 确切地说,

$$\begin{aligned} y(x + 2h) &= y_1 + O(h^5) \\ y(x + h + h) &= y_2 + O(h^5) \end{aligned} \tag{2.4.91}$$

第二个方程是指用二步步长为h 的积分结果. 二者之差

$$\Delta \equiv y_2 - y_1 \tag{2.4.92}$$

显然可以作为误差的一个合适的量度. 由方程(2.4.91)可知, Δ 正比于h^5, 因此, 如果步长为h_1 时的误差为Δ_1, 而当步长为h_0 时的误差为Δ_0, 则显然应有

$$h_0 = h_1 \left| \frac{\Delta_0}{\Delta_1} \right|^{1/5} \tag{2.4.93}$$

如果我们令Δ_0 为充许误差, 则方程(2.4.93)可以用以调整步长, 如果$|\Delta_1|$ 大于$|\Delta_0|$, 则h_0 可作为新的步长重新计算, 否则, 如果$|\Delta_1|$ 小于$|\Delta_0|$, 则h_0 可作为下一步步长的一个合理的推断.

下面考虑如何选取Δ_0, 你可以把它取为某个小的数字, 例如, 10^{-6}, 但略加思考就会发现这是不行的, 理由如下, 所求函数y 的绝对值可以很大, 也可以很小, 因此, 绝对精度是没有意义的; 所以, 我们可以考虑取$\Delta_0 = \epsilon y$, 而ϵ 为某一小的数字; 但另一方面, 有些函数是振荡的, 在某一极大和极小之间, 并且经常穿过0点, 对于这类问题, Δ_0 的一个合适的选则应为某一小量乘以y 的极大值的绝对值; 为了照顾到上述两种情况, 我们可以取

$$\Delta_0 = \epsilon\, y_{\rm scal} \tag{2.4.94}$$

其中

$$y_{\rm scal} = |y| + |h \times \frac{dy}{dx}|$$

上面是以一步为基础来考虑的, 在有些问题中, 我们可能要求解满足某种全局的精度, 由于舍入误差的积累(在最坏的情况下, 舍入误差全部相加, 总体误差为$N\Delta_0 \sim \frac{L}{h}\Delta_0$, 其中$L$为对$x$ 的积分的长度), 当h 缩小时, Δ_0 也要相应的缩小, 以保证总的舍入误差固定, 即选取$y$scal 正比于h, 所以当h_0 变化时, $y_{\rm scal}$ 也相应的变化, 从而方程(2.4.93)中的指数成为1/4, 另外, 我们只考虑了误差的主导项, 高阶项也应有一些贡献. 基于所有这些考虑, 我们把方程(2.4.93) 改写为

$$\begin{aligned} h_0 &= Sh_1 \left| \frac{\Delta_0}{\Delta_1} \right|^{1/5} \qquad \Delta_0 \geq \Delta_1 \\ &= Sh_1 \left| \frac{\Delta_0}{\Delta_1} \right|^{1/4} \qquad \Delta_0 < \Delta_1 \end{aligned} \tag{2.4.95}$$

这里S 为一考虑了高阶效应的安全系数, 指数的选择从最安全出发. 当放大步长时, 用小指数1/5 (少放一点); 当缩小步长时, 用大指数1/4(多缩一些). Δ_0由(2.4.94)给出.

为了简洁起见, 这里以一个方程为基础给出了四阶龙格—库塔方法, 但不难把它们推广到多个方程.

在结束本小节前, 我们指出, 虽然这里讲的是一阶常微分方程组的解法, 但实际上已经包括了高阶常微分方程, 因为任何一个高阶常微分方程均可化为一组等价的一阶常微分方程组, 如果有同学对这一表述还不清楚, 那么请立即复习《高等数学》的有关内容.

本节的方法基本上可以满足实际计算的需要. 但在实际工作中, 也会碰到一些特殊的问题, 比较重要的一类问题是所谓刚性微分方程求解问题, 关于这一点, 请参看W. C. Gear 的著作. 另外一些特殊问题将在后面结合具体物理问题进行处理.

练习:

单摆的方程为

$$\ddot{x} + \omega^2 \sin x = 0$$

设初始位移为$x = 0$, 试用二阶龙格-库塔方法对不同的初始速度求解此方程, 并讨论所得结果

2.4.3 两点边值问题

所谓两点边值问题是指下面的定解问题

$$\frac{dy_i}{dx} = g_i(x; y_1, y_2, \cdots, y_n) \quad i = 1, 2, \cdots, n \tag{2.4.96}$$

在点x_1, 有边界条件

$$B_{1j}(x_1; y_1, y_2, \cdots, y_n) = 0, \quad j = 1, 2, \cdots, n_1 \tag{2.4.97}$$

在点x_2, 有边界条件

$$B_{2j}(x_2; y_1, y_2, \cdots, y_n) = 0, \quad j = 1, 2, \cdots, n - n_1 \tag{2.4.98}$$

这样一类问题与前节所处理的问题的最大差别在于定解条件是在x 的两个不同的点(端点)处给出的, 而前节的定解条件是在一个点给出的, 因此, 前节的问题称为初值问题而本节将要解决的问题称为两点边值问题. 初值问题一般来说总是有解的, 但边值问题的解有时并不存在, 边值问题中常常包含一个参数, 只有对参数的某些值, 边值问题的解才存在, 这类问题称为微分方程的本征值问题, 使得方程有解的那些参数值称为本征值.

边值问题有各种不同的解法, 我们将在本节介绍打靶法. 并通过一个具体算例给出处理边值问题的一些技巧和方法. 在后面的章节中, 我们将结合具体物理问题, 再回到本节的问题. 这里要强调指出的是, 由于边值问题的特殊性, 试图像初值问题那样给出通用程序是几乎不可能的, 在实际工作中应根据具体的物理问题设计合适的方法.

打靶法的基本思路是从x_1点出发, 利用n_1 个已知的边界条件并指定n_2 个条件, 积分到x_2 点, 看是否与x_2 点的n_2 个条件一致, 若否, 则调节在x_1 指定的条件直到符合要求

为止. 现在我们给出仔细的分析, 令$V=[V_1,V_2,\cdots,V_{n_2}]^T$ 为一n_2 维空间的矢量, 它给出了x_1 点的n_2 个指定的条件, 则方程(2.4.97) 可以写为

$$y_i(x_1)=y_i(x_1;V_1,V_2,\cdots,V_{n_2})\quad i=1,2,\cdots,n \tag{2.4.99}$$

因此, 给定V, 我们就有了一个完整的初值问题, 利用前节的方法在区间$[x_1,x_2]$ 之间积分方程(2.4.96), 便可得到一组$y_i(x_2;V_1,V_2,\cdots,V_{n_2})$, 在$x_2$, 定义一个$n_2$ 维的偏离矢量F, F的取法并不唯一, 可视方便而定, 例如可取

$$F_k=B_{2k}(x_2;y_1,y_2,\cdots,y_n)\quad k=1,2,\cdots,n_2 \tag{2.4.100}$$

如果$F=0$, 则说明V的选则是正确的. 因此, 问题转化为求解方程$F=0$, 这里有n_2 个未知量V_k, n_2个方程$F_k=0$. 为了求解这一组方程, 我们假定V 与真实的解相差不大, 从而可以把方程(2.4.100)线性化, 即令

$$F_k(V^*-\delta V)=-\sum_{j=1}^{n_2}A_{kj}\delta V_j \tag{2.4.101}$$

这里

$$A_{kj}=\frac{\partial F_k}{\partial V_j} \tag{2.4.102}$$

由方程(2.4.101)求出δV_j, 并令新的V 为

$$V_k^{new}=V_k^{old}+\delta V_k \tag{2.4.103}$$

通过反复迭代, 最终得到结果.

现在, 让我们回到(2.4.102), 由于F_k 是由数值给出的, 因此A_{kj} 的计算也只能是数值的, 最简单的方法是令

$$\frac{\partial F_k}{\partial V_j}\approx\frac{F_k(V_1,V_2,\cdots,V_j+\delta V_j,\cdots)-F_k(V_1,V_2,\cdots,V_j,\cdots)}{\delta V_j} \tag{2.4.104}$$

这样, 每完成一次迭代需要求解n_2+1 次初值问题, 一次计算$F_k(V_1,V_2,\cdots,V_{n_2})$, n_2 次计算偏导数, 显然, 与初值问题相比, 边值问题的计算量是相当大的. V 的初始叠代值的选择是一个困难的问题, 如果选择的不好, 则可导致计算的失败, 通常应根据对被求解问题的了解程度尽量选择接近其解的量作为初始叠代值. 在后面的例子中, 将给出一个针对具体问题的选法.

为了帮助理解打靶法的应用, 这里给出一个例子, 即旋转椭球谐振子. 当在旋转椭球坐标系中对某些偏微分方程如Laplace方程分离变量时, 将会导致下面的方程

$$\frac{d}{dx}\left[(1-x^2)\frac{dS}{dx}\right]+\left(\lambda-c^2x^2-\frac{m^2}{1-x^2}\right)S=0 \tag{2.4.105}$$

这里m 为一整数. 这是一个Sturm–Liouville本征值问题, λ 是方程的本征值. 方程在$x = \pm 1$ 具有奇点, 从Sturm–Liouville本征值问题的一般理论可知(参见梁昆淼, 《数学物理方法》, 如果我们寻求在$x = \pm 1$ 为正则的解, 则只有对λ 的某些特殊值才有可能, 这些特殊值称为本征值, 且所有的本征值均大于0. 当$c = 0$ 时, 方程化为球谐振子的方程, 其解为$P_n^m(x)$, 本征值为$\lambda_{mn} = n(n+1)$, $n = m, m+1, \cdots$, 当$c \neq 0$ 时, 我们记(2.4.105)的本征值为$\lambda_{mn}(c)$, 对应的本征函数记为$S_{mn}(x)$. $\lambda_{mn}(c)$和$S_{mn}(x)$的计算曾经是非常困难的问题, 其十分复杂的级数解, 递推关系等可在一些非常专门的书籍中查到. (例如, 可参看M. Abramowitz and I Stegun *Handbook of Mathematical Functions* (Dover Publications, New York, 1968)) 其数值解相对来说要容易得多. 首先, 我们作代换(从天上掉下来的?, 请复习《数学物理方法》中常微分方程的有关内容.)

$$S = (1 - x^2)^{m/2} y$$

则原方程化为

$$(1 - x^2)\frac{d^2y}{dx^2} - 2(m+1)x\frac{dy}{dx} + (\mu - c^2x^2)y = 0 \tag{2.4.106}$$

这里

$$\mu \equiv \lambda - m(m+1) \tag{2.4.107}$$

方程(2.4.104)和(2.4.106)均在代换$x \to -x$ 下不变, 因此S 及y 除了一个常数因子外也应在同一变换下不变, 由于解最终将归一化, 因此这一常数因子只能是± 1. 方程(2.4.106) 也是一个Sturm–Liouville 方程, 其本征值$\mu \geq 0$, 参照球谐振子的结果, 如果令$\lambda_{mn} = \alpha_{mn} + n(n+1)$, 且要求$\alpha_{mn} \geq 0$, 则由(2.4.107) 可知$n \geq m$, $n = m$ 对应于最低本征值. 利用Sturm–Liouville 方程的从小到大排列的本征值与其对应的本征函数的零点的定理可知, 当$n - m$ 为偶数时, y_{mn}在$[-1, 1]$之间有偶数个零点, 所以我们取$y(x) = y(-x)$, 否则, 取$y(x) = -y(-x)$, 于是

$$y_{mn}(-x) = (-1)^{n-m} y_{mn}(x) \tag{2.4.108}$$

在$x \to -1$ 时, 可以解得

$$y'(-1) = -\frac{\mu - c^2}{2(m+1)} y(-1) \tag{2.4.109}$$

在原点, 我们有

$$\begin{aligned} y(0) &= 0, \quad n - m\text{为奇数} \\ y'(0) &= 0, \quad n - m\text{为偶数} \end{aligned} \tag{2.4.110}$$

最后, 我们令S_{mn} 在$x \to -1$时与P_n^m有相同的极限行为. (这相当于选则一个特定的归一化条件)

$$\lim_{x \to -1} (1 - x^2)^{-m/2} S_{mn}(x; c) = \lim_{x \to -1} (1 - x^2)^{-m/2} P_n^m(x) \tag{2.4.111}$$

现在, 令

$$\begin{aligned} y_1 &= y \\ y_2 &= y' \\ y_3 &= \mu \end{aligned} \tag{2.4.112}$$

这样, 方程(2.4.106)变为

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= \frac{1}{1-x^2}\left[2x(m+1)y_2 - (y_3 - c^2x^2)y_1\right] \\ y_3' &= 0 \end{aligned} \tag{2.4.113}$$

我们选择区间$[-1,0]$ 来求解(为什么不选$[0,1]$?), 边界条件为, 在$x=-1$, 有

$$\begin{aligned} y_2 &= -\frac{y_3 - c^2}{2(m+1)}y_1 \\ y_1 &= \lim_{x\to -1}(1-x^2)^{-m/2}P_n^m(x) = \frac{(-1)^m(n+m)!}{2^m m!(n-m)!} \end{aligned} \tag{2.4.114}$$

在$x=0$, 有:

$$\begin{aligned} y_1 &= 0, \quad n-m\text{为奇数} \\ y_2 &= 0, \quad n-m\text{为偶数} \end{aligned} \tag{2.4.115}$$

在这一问题中, 未知量是本征值μ_{mn}, 为了给出一个好的初值, 我们注意到在方程(2.4.106)中, 如果把$\mu - c^2x^2$ 作为本征值看待, 则为球谐振子方程, 其本征值已知为$n(n+1)-m(m+1)$, 因为$0 \le x^2 \le 1$, 因此, 把x^2 代之以$\frac{1}{2}$ 得到的结果应是原问题的一个良好的近似, 为此, μ_{mn} 的初始叠代值可选为$n(n+1)-m(m+1)+\frac{1}{2}c^2$.

在这个例子中, 我们不仅示范了打靶法的使用, 同时也示范了如何把高阶方程化为一阶方程以及如何处理本征值问题, 请同学仔细体会, 掌握这一方法并应用于解决实际问题.

练习*:*

对于方程

$$y'' + \frac{1}{x}y' + ky = 0$$

在边界条件$y(0)$ 有限, $y(1)=0$ 下计算本征值k.

提示, 为了得到k 的叠代初值, 考虑k 很大时, 方程成为$y''+ky=0$, 其解为$y \sim \cos(\sqrt{k}x+\delta)$, 若令$\delta = 0$ *(*δ 应在区间$[0,\frac{\pi}{2}]$ 内*)*, 则$k=(n+\frac{1}{2})^2\pi^2$.

2.5 插值和逼近

给定一个列表函数, $(x_1, f(x_1)), (x_2, f(x_2)), \cdots, (x_n, f(x_n))$, 要求寻找一个函数$\phi(x)$ 来近似代替$f(x)$, 使得在$(x_1, x_2, \cdots, x_n)$, $f(x_i) = \phi(x_i)$, 而对于其它的x, $\phi(x)$ 是$f(x)$ 的一个很好的近似. 这样的问题就称为插值问题. 如果读者稍为细心一点, 就会发现我们的最后一个要求是有问题的, 因为我们仅仅知道$f(x)$ 在一些分立点x_i 上的值, 事先并不知道$f(x)$ 在这些点以外如何变化, 因此除了要求$\phi(x)$ 在诸x_i 点上与$f(x)$ 相等外, 无法提出更多的要求. 然而, 在物理上和数学上, 假如我们认为一组分立点的数值大致决定了$f(x)$, 这就隐含着我们假定了$f(x)$ 在任何两个给定点之间是光滑变化的. 因此, 我们的目的实际上是寻找一条光滑地通过n 个给定点$(x_1, f(x_1)), (x_2, f(x_2)), \cdots, (x_n, f(x_n))$ 的一条曲线.

2.5.1 多项式插值

通过两点可以唯一地作一条直线, 通过三点可以唯一地作一条抛物线, ... 通过n 个数据点$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ 可以唯一地决定一个$n-1$ 阶的多项式$P_n(x)$, 这一多项式就称为$y = f(x)$ 的插值多项式. 它由著名的Lagrange 公式给出.

$$\begin{aligned} P(x) &= \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)}y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)}y_2 + \\ &\quad \cdots \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}y_n \end{aligned} \tag{2.5.116}$$

它共有n 项, 每一项为一$n-1$ 次多项式, 且第i 项除在点x_i 为$y_i = f(x_i)$ 外, 在其余诸x_j 处均为0.

直接对(2.5.116)编写程序虽然不是一个很好的算法, 但也是完全可行的. 读者可作为一个编程练习来写出计算程序. 我们在这里将介绍一个更好的算法, Neville算法, 这一算法的基本思想是从一个低次多项式出发, 逐次用递推方式加高并最终得到结果. 下面给出这一算法的一个简短的说明. 对于给定的x, 令P_{11}^0 是通过(x_1, y_1) 点的唯一的0 次多项式, 即$P_{11}^0 = y_1$, (同样定义$P_{22}^0, P_{33}^0, \cdots P_{nn}^0$); 定义$P_{12}^1$ 为通过$(x_1, y_1), (x_2, y_2)$ 的唯一一次多项式. 同理有$P_{23}^1, P_{34}^1, P_{n-1,n}^1$; $\cdots\cdots$, 定义$P_{i,i+m}^m$ 为通过$(x_i, y_i), (x_{i+1}, y_{i+1}), \cdots (x_{i+m}, y_{i+m})$ 的m 次多项式, 则P_{1n}^{n-1} 即为通过$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ 的唯一$n-1$ 次多项式, 即为(2.5.116)的等价形式. 所有这些P 可以排成一个表, 其"祖代"排在左边, "子代"在右边并最终导致唯一的"后代"即为我们的解. 如下式

所示(以$n=5$ 为例):

$$\begin{array}{lllllll}
x_1: & y_1 = P_{11}^0 & & & & & \\
 & & P_{12}^1 & & & & \\
x_2: & y_2 = P_{22}^0 & & P_{13}^2 & & & \\
 & & P_{23}^1 & & P_{14}^3 & & \\
x_3: & y_3 = P_{33}^0 & & P_{24}^2 & & P_{15}^4 & \\
 & & P_{34}^1 & & P_{25}^3 & & \\
x_4: & y_4 = P_{44}^0 & & P_{35}^2 & & & \\
 & & P_{45}^1 & & & & \\
x_5: & y_5 = P_{55}^0 & & & & &
\end{array} \tag{2.5.117}$$

“子代”与“父代”之间满足如下递推关系:

$$P_{i,i+m}^m = \frac{(x - x_{i+m})P_{i,i+m-1}^{m-1} + (x_i - x)P_{i+1,i+m}^{m-1}}{x_i - x_{i+m}} \tag{2.5.118}$$

练习

证明上述递推关系

直接用(2.5.118)计算是完全可行的, 但还可进一步改进, 一种改进的方法是在计算中做小量 “子代”和“父代”之差的迭代. 即定义

$$\begin{aligned}
C_{mi} &\equiv P_{i,i+m}^m - P_{i,i+m-1}^{m-1} \\
D_{mi} &\equiv P_{i,i+m}^m - P_{i+1,i+m}^{m-1}
\end{aligned} \tag{2.5.119}$$

从(2.5.118)可容易推出C_{mi} 和D_{mi} 所满足的递推关系为:

$$\begin{aligned}
D_{m+1,i} &= \frac{(x_{i+m+1} - x)(C_{m,i+1} - D_{mi})}{x_i - x_{i+m+1}} \\
C_{m+1,i} &= \frac{(x_i - x)(C_{m,i+1} - D_{mi})}{x_i - x_{i+m+1}}
\end{aligned} \tag{2.5.120}$$

在每一次递推中, C 和D 是对插值的更高一次的修正, 最后的结果P_{1n}^{n-1} 是任一y_i 加上一系列从y_i 出发而最终到达“家族” 终点的任一路径上的C 及D.

2.5.2 分式有理函数插值和外推

有些函数(大部分函数?)不能很好地用多项式来近似, 我们在后面要举一个著名的例子, 但大部分函数都能用分式有理函数来近似, 这里边有一些深入的数学原理, 有兴趣的同学可以研读有关的数学书籍.

一个通过$m+1$ 个点$(x_0,y_0),(x_1,y_1),\cdots,(x_m,y_m)$ 的分式有理函数可记为

$$R_{\mu,\nu}(x)=\frac{P_\mu(x)}{Q_\nu(x)}=\frac{p_0+p_1x+\cdots+p_\mu x^\mu}{1+q_1x+\cdots+q_\nu x^\nu} \tag{2.5.121}$$

由于有$\mu+\nu+1$ 个未知参数p 及q, 因此, 必有$m+1=\mu+\nu+1$. 在做分式有理函数插值时, 还需要给定分子或分母的阶.

方程(2.5.121)两边乘以$Q_\nu(x)$, 分别代入逐插值点, 得到一组线性代数方程组, 可由此解得p_i和q_i, 得到有理插值函数. 类似于多项式插值, 我们也可以构造一个递推算法, 这一算法称为Bulirsch—Stoer算法. 这里不予介绍, 有兴趣的同学可参看(J. Stoer, R. Bulirsch, Introduction to Numerical Analysys, Springer-Verlag, 1992, Chapter 2).

2.5.3 三次样条插值

三次样条插值是目前应用最广泛的一种方法, 它可以保持插值函数具有二次连续导数, 而且不会出现多项式插值中的一些问题. 三次样条插值问题可以表述如下, 对于给定的n 个点$(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$, 寻找一个分段三次多项式

$$S(x)=\begin{cases} S_1(x) & (x_1\le x\le x_2)\\ S_2(x) & (x_2\le x\le x_3)\\ \cdots & \cdots\\ S_{n-1}(x) & (x_{n-1}\le x\le x_n)\end{cases} \tag{2.5.122}$$

其中, $S_i(x)$ 为一定义在区间$[x_i,x_{i+1}]$ 上的三次多项式并要求$S(x)$ 在诸插值点上的一阶及二阶导数连续.

我们先来分析这一插值问题的解是否存在, 由于决定每个三次多项式需要4 个条件, 所以总共需要$4(n-1)$ 个条件, 由每个多项式都要通过两个端点, 给出$2(n-1)$ 个条件, 一阶导数和二阶导数连续给出$2(n-2)$ 个条件, 这样, 我们总共有$4(n-1)-2$ 个条件. 因此, 为了确定插值问题, 还需要补充二个条件, 由这两个条件的不同, 又有几种不同的三次样条插值方法, 这里我们将介绍二种, 一种是再给定两个端点x_1,x_n 的一阶导数值y_1',y_n', 另一种是令两个端点上的二阶导数为0 , 即令$y_1''=y_n''=0$.

现在来求解这一问题, 由于二阶导数连续, 所以$y_i'',i=2,3,\cdots,n-1$ 是存在的, 在任一子区间$[x_i,x_{i+1}]$, 有

$$S_i''(x)=\frac{x-x_i}{x_{i+1}-x_i}y_{i+1}''+\frac{x_{i+1}-x}{x_{i+1}-x_i}y_i'' \tag{2.5.123}$$

对上式积分二次, 得到

$$\begin{aligned} S_i(x) &= \frac{1}{6}\frac{(x_{i+1}-x)^3}{x_{i+1}-x_i}y_i'' + \frac{1}{6}\frac{(x-x_i)^3}{x_{i+1}-x_i}y_{i+1}'' \\ &\quad + (y_{i+1} - \frac{1}{6}y_{i+1}''(x_{i+1}-x_i)^2)\frac{x-x_i}{x_{i+1}-x_i} \\ &\quad + (y_i - \frac{1}{6}y_i''(x_{i+1}-x_i)^2)\frac{x_{i+1}-x}{x_{i+1}-x_i} \end{aligned} \tag{2.5.124}$$

上式包含n 个未知量$y_i'', i = 1, 2, 3, \cdots, n$, 为了决定这些未知量, 对上式微分一次, 得到

$$\begin{aligned} S_i'(x) &= -\frac{1}{2}\frac{(x_{i+1}-x)^2}{x_{i+1}-x_i}y_i'' + \frac{1}{2}\frac{((x-x_i)^2}{x_{i+1}-x_i}y_{i+1}'' \\ &\quad + (y_{i+1} - \frac{1}{6}y_{i+1}''(x_{i+1}-x_i)^2)\frac{1}{x_{i+1}-x_i} \\ &\quad - (y_i - \frac{1}{6}y_i''(x_{i+1}-x_i)^2)\frac{1}{x_{i+1}-x_i} \end{aligned} \tag{2.5.125}$$

由$S_i'(x)$ 的连续性条件, 可得$n-2$ 个方程如下(对$i = 2, 3, \cdots, n-1$)

$$\frac{x_i - x_{i-1}}{6}y_{i-1}'' + \frac{x_{i+1}-x_{i-1}}{3}y_j'' + \frac{x_{i+1}-x_i}{6}y_{i+1}'' = \frac{y_{i+1}-y_i}{x_{i+1}-x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{2.5.126}$$

如前所述, 我们还需要2 个补充条件以决定n 个未知量y_i'', 其选择方法是, 如果给定端点的一阶导数, 则把y_1' 和y_n' 代入方程(2.5.125)可得到两个关于y'' 的方程, 与(2.5.126)一起构成一个完整的线性代数方程组; 如果给定端点的二阶导数为0, 则(2.5.126)足以决定$n-2$ 个未知量. 一旦求得了诸y_i'', 则对给定的x, 可利用(2.5.124)计算其对应的y 值.

下面我们看一个例子, 考虑下面的函数

$$y = \frac{1}{1+25x^2} \tag{2.5.127}$$

我们在$[-1, 1]$ 之间取11 个点, 分别用多项式, 分式有理函数和三次样条函数进行插值, 计算结果见图(7.1), 其中实线为原函数, 点线为多项式插值的结果, 除了在插值点外, 与原函数的差别是相当大的, 点虚线和虚线分别为分式有理函数和三次样条插值的结果, 在图上几乎完全与准确值重合.

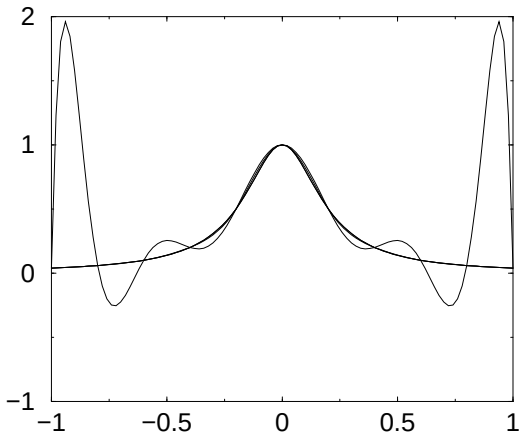


Figure 2.2: 多项式, 分式有理函数和三次样条函数插值的比较

2.5.4 列表函数的积分

在物理上, 经常会遇到计算列表函数的积分. 实验测量的结果总是在一些分立点上, 大型数值计算的结果也只能是一些分立值. 样条插值方法可以用来计算此类积分. 对于列表函数, 可首先建立其样条插值, 求出由式(2.5.122) 所指定的函数$S(x)$, 由于诸$S_i(x)$ 都是三次多项式, 其积分可解析求出, 把每一分段上的积分值加起来就得到最终结果.

练习
分别对于端点和内点计算*(2.5.124)*的积分, 并最终给出样条插值方法计算列表函数的算法和程序

练习
对于函数

$$f(x) = \frac{x\cos x}{1+x^3}$$

(i), 用辛普生方法计算$\int_{1.5}^{2} f(x)dx$
(ii), 在区间$[1,2]$ 上分*50* 个子区间, 计算$f(x_i)$的值, 利用这些数值并用样条函数方法*(*见上题*)*计算上述程分, 比较所得结果

2.5.5 Padé插值与外推

在理论物理中, 微扰方法是进行实际计算的最重要的方法之一(在力学界称为摄动方法), 粗略地讲, 微扰方法就是在计算某一物理量时, 利用这一物理问题中的某个小参量作级

数展开, 例如量子力学中的微扰论, 弹性力学及天体力学中的摄动理论, 相变理论中的高温展开和低温展开等都是典型的例子. 但是, 级数展开毕竟是级数展开, 它只在展开参量很小时有效(如果级数不收敛, 甚至在参数很小时也是有问题的), 但另一方面, 在物理问题中, 小参量有时并不小, 其选取有时(在大多数情况下)纯粹是为了计算方便, 那么, 有没有可能从这样一个级数出发, 得到对参数的从小到大的所有值都有意义的结果呢? 这似乎是一个失去理智的要求. 然而, 这个问题的回答是肯定的(如果没有一批似乎是失去理智的人的开创性工作, 也就不会有今天科学的高度进步).

对于一个函数$f(x)$, 假定我们知道它的幂级数为

$$f(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots \tag{2.5.128}$$

这一幂级数所反映的, 当然只是$f(x)$ 在原点的局部性质, 如果用它来计算函数在原点附近的值, 在上面的幂级数收敛的前提下, 一般取前几项就可得到较好的结果. 倘若要求$|x|$较大时的值或(2.5.128)不收敛, 则这一级数对于计算是没有什么用处的. Padé逼近就是针对这样一个没有用处的形式级数(并不要求收敛), 从中得到函数$f(x)$的性质, 并想法计算任一x 下$f(x)$ 的值. 之所以能够做到这一点是因为一个形式的幂级数已经包含了关于函数$f(x)$ 的很多信息. 对于一个确定的函数来说, 函数的各点之间应该具有某种联系, 也就是说, 函数在一点的行为是与其整体形为有关的, 这就是Padé 逼近可以成功的基础. 函数的一点与整体的关系问题是一个很深奥的问题, 我们不打算(也不可能)在此给出详细的解释, 但需要强调指出的是, Padé逼近虽然最早是由数学家提出的, 但其理论的发展和广泛的应用则是由物理学家完成的.

Padé逼近的定义是, 对于由(2.5.128)定义的函数, 寻找一个分式有理函数

$$R_{m,n}(x) = P(x)/Q(x) \tag{2.5.129}$$

其中$P(x)$ 为一m 次多项式, $Q(x)$ 为一n 次多项式且$Q(0) = 1$, 使得

$$f(x) - P(x)/Q(x) = \mathcal{O}(x^{m+n+1}) \tag{2.5.130}$$

即$R_{m,n}(x)$ 的泰勒展开的前$m + n + 1$ 项(包括常数项)与$f(x)$ 的泰勒展开是一样的. 这样的分式有理函数就称为$f(x)$ 的一个Padé逼近, 简记为$[m/n]$. 需要注意的是, 这样定义的padé逼近并不总是存在的. Padé逼近有很多十分有趣的性质, 有兴趣的同学可以参看由著名理论物理学家G. A. Baker Jr.所写的*The essentials of Padé approximations* (Academic Press, New York , 1975)一书.

对于一个函数$f(x)$, 其Padé 逼近$[m/n]$ 可以排成一个表, 称为Padé 表, 如下式所示.

m\ n	0	1	2	3	4	5	⋯
0	[0/0]	[0/1]	[0/2]	[0/3]	[0/4]	[0/5]	⋯
1	[1/0]	[1/1]	[1/2]	[1/3]	[1/4]	[1/5]	⋯
2	[2/0]	[2/1]	[2/2]	[2/3]	[2/4]	[2/5]	⋯
3	[3/0]	[3/1]	[3/2]	[3/3]	[3/4]	[3/5]	⋯
4	[4/0]	[4/1]	[4/2]	[4/3]	[4/4]	[4/5]	⋯
5	[5/0]	[5/1]	[5/2]	[5/3]	[5/4]	[5/5]	⋯
⋯			⋯	⋯			

Padé 表有很多有趣的性质, 其中之一是Wynn 恒等式, 对于Padé 表上相邻的五个元素:

$$\begin{array}{ccc} & N & \\ W & C & E \\ & S & \end{array}$$

有

$$\frac{1}{E-C}+\frac{1}{W-C}=\frac{1}{N-C}+\frac{1}{S-C} \tag{2.5.131}$$

下面我们讲述计算Padé表的方法, 如果我们的目的是求出Padé表的数值, 则可使用ε算法, 如果已知

$$f(x)=\sum_{i=0}^{\infty}c_i x^i$$

这一算法首先定义一组

$$\begin{aligned} \varepsilon_0^j &= \sum_{i=0}^{j}c_i x^i \quad j=0,1,2,\cdots \\ \varepsilon_{-1}^j &= 0 \quad j=0,1,2,3,\cdots \\ \varepsilon_{2k}^{-k-1} &= 0 \quad k=0,1,2,3,\cdots \end{aligned}$$

则可以建立下面的ε表

	ε_0^{-1}		ε_2^{-2}		ε_4^{-3}		⋯
ε_{-1}^0		ε_1^{-1}		ε_3^{-2}		ε_5^{-3}	⋯
	ε_0^0		ε_2^{-1}		ε_4^{-2}		⋯
ε_{-1}^1		ε_1^0		ε_3^{-1}		ε_5^{-2}	⋯
	ε_0^1		ε_2^0		ε_4^{-1}		⋯
ε_{-1}^2		ε_1^1		ε_3^0		ε_5^{-1}	⋯
	ε_0^2		ε_2^1		ε_4^0		⋯
ε_{-1}^3		ε_1^2		ε_3^1		ε_5^0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

其中诸ε 之间满足如下的递推关系

$$\varepsilon_{k+1}^{j}=\varepsilon_{k-1}^{j+1}+\frac{1}{\varepsilon_{k}^{j+1}-\varepsilon_{k}^{j}}\quad(j,k=0,1,2,\cdots)\tag{2.5.132}$$

因此, 从ε 表的第一行及第一, 二两列出发, 便可得到表中的所有元素. 可以证明, ε 表与Padé表之间有下述关系:

$$\varepsilon_{2k}^{j}=[(k+j)/k]\tag{2.5.133}$$

因此, 一旦求得了ε 表, 也就得到了Padé 表.

在(2.5.132) 中, 如果第二项的分母变为零(在机器精度内), 则计算将无法进行下去, 通常计算机将报告一个除以零(Divide by Zero) 错误, 这一般是由于在计算中所取项数较多, 原级数已经收敛. 在此情况下, 可少取几项进行计算, 并不断增加项数比较所得结果. 也可以用下面将要介绍的方法求出Padé 逼近的系数, 用其解析式进行计算.

练习

对较小的k,j, 验证式*(2.5.133)*, 并进而证明它. *(*提示*:* 用归纳法*)*

练习

*1,*编写利用ε 表计算*padé*逼近的程序, 利用$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\cdots$, 用*padé*逼近方法计算$\ln(5)$的值并与准确值比较

2, 考虑π 的一个著名的慢收敛级数,

$$\pi=4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\cdots=\sum_{k=0}^{\infty}(-1)^k\frac{4}{2k+1}$$

用*Padé*近似计算π 的数值.

3, 考虑积分

$$I(x)=\int_0^{\infty}\frac{e^{-t}}{1+xt}\,dt$$

试证明此积分可写为下面的形式级数

$$I(x)=1-1!x+2!x^2-3!x^3+4!x^4-5!x^5+\cdots$$

从这一发散级数出发, 利用*Padé*近似计算$I(1)$, *(*$I(1)$的准确值可用数值积分方法求出为$0.596347\cdots$*)*

4, 对于$x=0.5$, 利用e^x 的幂级数展开式, 从小到大改变所取级数的项数N并用前面的程序计算其*Padé* 逼近, 观察所得结果并与准确值比较. 当N 较大时会发生什么情况*?*

在很多问题中, 我们并不仅仅需要Padé逼近的数值, 而且需要知道Padé逼近的解析形式, 即多项式$P(x)$ 和$Q(x)$ 的系数. 为此, 考虑方程(2.5.130), 记

$$\begin{aligned} f(x) &= c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots + c_{m+n} x^{m+n} + \mathcal{O}(x^{m+n+1}) \\ P(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_m x^m \\ Q(x) &= 1 + b_1 x + b_2 x^2 + b_3 x^3 + \cdots + b_n x^n \end{aligned} \tag{2.5.134}$$

这里, $c_i, i = 0, 1, 2, \cdots, m+n$ 已知, 我们的目的是求$a_i, i = 0, 1, 2, \cdots, m$ 和$b_i, i = 1, 2, \cdots, n$ 使得方程(2.5.130)满足. 可以证明, 如果函数$f(x)$ 的Padé逼近存在, 则必定是唯一的. 因此, 一旦求得了a_i 及b_i, 则$P(x)/Q(x)$ 就是$f(x)$ 的$[m/n]$ Padé逼近. 为了计算系数a_i 及b_i, 把方程(2.5.130)改写为

$$f(x)Q(x) - P(x) = \mathcal{O}(x^{m+n+1}) \tag{2.5.135}$$

把(2.5.134)代入(2.5.135), 比较方程两边x 同次幂的系数, 可以得到下面的两组线性代数方程.

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} c_0 & 0 & 0 & \cdots & 0 \\ c_1 & c_0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ c_m & c_{m-1} & c_{m-2} & \cdots & c_{m-n} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{2.5.136}$$

$$\begin{bmatrix} c_m & c_{m-1} & \cdots & c_{m-n+1} \\ c_{m+1} & c_m & \cdots & c_{m-n+2} \\ \vdots & \vdots & \cdots & \vdots \\ c_{m+n-1} & c_{m+n-2} & \cdots & c_m \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = - \begin{bmatrix} c_{m+1} \\ c_{m+2} \\ \vdots \\ c_{m+n} \end{bmatrix} \tag{2.5.137}$$

在上面的方程及后面的讨论中, 我们规定, 当$j < 0$ 时, $c_j = 0$. 如果方程(2.5.137)的解存在, 我们可利用下一章将要介绍的求解线性代数方程组的方法求得b_i, 再代入(2.5.136)求出a_i, 从而得到了Padé逼近的解.

顺便指出, 同ε 算法一样, 也有一些高效率的迭代算法, 鉴于物理上感兴趣的问题中最困难的部分在于求出$f(x)$ 的形式级数, 而Padé逼近的计算在总的计算中所占机时完全可以忽略不计, 因此我们不再做进一步的介绍.

2.5.6 发散级数的Cesáro求和及其推广

除了前述Padé逼近, 在物理上还常用另外一种计算发散级数的方法, Cesáro求和方法及其推广. 我们先看几个经典的例子.

考虑如下求和

$$S = 1 - 1 + 1 - 1 + 1 - 1 + 1 - 1 + \cdots \tag{2.5.138}$$

按照通常级数收敛的判据, 这是一个发散级数. 另一方面, 它是下面的展开式中$x = 1$的情形

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + x^4 - x^5 + \cdots \tag{2.5.139}$$

上式左边对任何$x \neq -1$都有意义, 当$x = 1$时, 为1/2. 现在我们试图从(2.5.138)得到这个结果. 注意到(2.5.138)的结构, 我们有

$$\begin{aligned} S &= 1 - 1 + 1 - 1 + 1 - 1 + \cdots \\ &= 1 - S \end{aligned}$$

于是可得到$S = 1/2$. 进一步考虑

$$\begin{aligned} S &= 1 - 2 + 4 - 8 + 16 - \cdots \\ &= 1 - 2S \end{aligned}$$

由此得到$S = 1/3$, 与(2.5.139)左边的结果相同. 这里, 我们实际上利用了所给级数是一个函数的形式展开级数取特定值这样一个事实, 一般这样一种方法在应用时要十分小心, 如果所给级数并不是某一函数的形式展开, 那么上述处理可能给出错误的结果.

一般来说, 如果我们要计算

$$s = a_0 + a_1 + a_2 + \cdots \tag{2.5.140}$$

记部分和为

$$s_n = a_0 + a_1 + a_2 + \cdots + a_{n-1} + a_n \tag{2.5.141}$$

则

$$s = \lim_{n\to\infty} s_n \tag{2.5.142}$$

另一方面, 对任何收敛级数, 其部分和的平均值当$n \to \infty$时收敛于原级数的和s. 因此我们可以用下述Cesáro变换计算级数和.

$$s = \lim_{n\to\infty} \frac{s_0 + s_1 + s_2 + \cdots + s_n}{n+1} \tag{2.5.143}$$

对于一个不收敛的级数, 若上式极限存在, 则称级数可Cesáro求和并把s作为级数的和.

对于

$$1 - 1 + 1 - 1 + 1 - 1 + \cdots$$

部分和为

$$1, 0, 1, 0, 1, 0, \cdots$$

其Cesáro和为$s=\frac{1}{2}$. 又对于

$$1-2+3-4+5-\cdots$$

部分和为

$$1,-1,2,-2,3,-3,\cdots$$

并不给出收敛的结果. 为此, 可以考虑Cesáro方法的推广, 定义

$$\begin{aligned} H_n^1 &= \frac{s_0+s_1+s_2+\cdots+s_n}{n+1} \\ H_n^2 &= \frac{H_0^1+H_1^1+H_2^1+\cdots+H_n^1}{n+1} \\ &\cdots \\ H_n^{m+1} &= \frac{H_0^m+H_1^m+H_2^m+\cdots+H_n^m}{n+1} \end{aligned}$$

若到某一级m, 极限$\lim_{n\to\infty} H_n^m$存在, 则可把此极限作为原级数的和. 对于前面的例子, 我们有

$$\begin{aligned} H_n^1 &: 1,0,\frac{2}{3},0,\frac{3}{5},0,\frac{4}{7},\cdots,0,\frac{n+2}{2(n+1)},\cdots \\ H_n^2 &: 1,\frac{1}{2},\frac{5}{9},\frac{5}{12},\frac{34}{75},\cdots,\frac{1}{4} \end{aligned}$$

注意到$H_n^2=\frac{1}{n+1}\sum_{i=1}^{[n/2]}\frac{2i+2}{2(2i+1)}=\frac{1}{n+1}\frac{1}{2}\left[\frac{n}{2}\right]+\frac{1}{n+1}\frac{1}{2}\sum_{i=1}^{[n/2]}\frac{1}{2i+1}$, 当$n\to\infty$, 第一项的极限是1/4, 第二项的求和的发散部分与$\int_1^{n/2}\frac{1}{x}dx=\ln n/2$相当，于是$\lim_{n\to\infty}\frac{\ln n/2}{n+1}\to 0$.

2.5.7 Borel求和

Borel求和是一种强有力的发散级数的求和方法, 比前一小节的简单方法有更大的适用范围. 在物理上有广泛应用.

设有一函数$J(x)$, 其级数表示

$$J(x)=\sum p_n x^n$$

对所有x都收敛. 考虑函数

$$S(x)=\sum p_n s_n x^n$$

这里s_n为欲求和的发散级数的部分和, 如果

$$\lim_{x\to\infty}\frac{S(x)}{J(x)}=s$$

存在, 则称级数"J"收敛, $s(J)$为"J"收敛的结果. 实际计算时, 我们一般不是取x趋于无穷大的极限, 而是寻找x的一个$S(x)$与$J(x)$之比为常数的范围, 并把此常数作为我们的结果. 最常用的函数为e^x, 对应的$p_n = 1/n!$. 此时我们寻找一个使函数

$$e^{-x}\sum_0^\infty s_n\frac{x^n}{n!}$$

稳定的区域.

例题, 考虑级数

$$1 - a + a^2 - a^3 + a^4 - a^5 + \cdots$$

当$a = 2$时, 即为前一节考虑过的的例子. 这一级数的部分和为

$$s_n = \frac{1-(-a)^{n-1}}{1+a}$$

因此

$$\frac{S(x)}{J(x)} = e^{-x}\sum_0^\infty \frac{1-(-a)^{n+1}}{1+a}\frac{x^n}{n!} = \frac{1}{1+a} + \frac{ae^{-x}}{1+a}\sum_0^\infty \frac{(-ax)^n}{n!}$$

$$= \frac{1}{1+a} + \frac{ae^{-x}e^{-ax}}{1+a}$$

上式对$\mathrm{Re}(a) > -1$收敛, 当$a = 2$时, 得到1/3. 这一方法称为Borel微分求和方法.

另一种Borel求和方法是Borel积分求和方法, 代替部分求和, 考虑

$$s(B) = \int_0^\infty e^{-x}\left[\sum_0^N \frac{a_n x^n}{n!}\right]dx$$

很显然, 如果我们对上式逐项积分, 将得到原级数, 而先对被积函数中的级数求和再计算积分, 则往往可得到正确的结果. 我们仍然考虑前一例子, 注意到

$$\sum_0^\infty \frac{(-a)^n x^n}{n!} = e^{-ax},$$

则当$\mathrm{Re}(a) > -1$时,

$$\int_0^\infty e^{-x}e^{-ax}dx = \frac{1}{1+a}.$$

当这一方法用于只有有限项的级数时，积分限一般不是选无限大，而是选取当所得结果几乎恒定时的积分上限为所要上限，恒定的结果为带求的值。

2.5.8 Bézier逼近

在这一节, 我们将处理一个近年来变得十分重要的逼近问题, 即对于一条曲线, 寻找一个函数形式去逼近它. 这在计算机辅助设计(CAD)中有较重要的应用. 这种逼近称为Bézier逼近, 它是由Bézier和De Casteljau大约在1962年分别独立发展起来的.

我们首先介绍Bernstein 多项式, 这是Bézier 逼近的基础. 假定我们考虑区间$a \le x \le b$, 第i 个n 次Bernstein 多项式定义为

$$B_i^n(x) = \left(\begin{array}{c} n \\ i \end{array} \right) \frac{(b-x)^{n-i}(x-a)^i}{(b-a)^n}, \quad i = 0, 1, 2, \cdots, n. \tag{2.5.144}$$

这里,

$$\left(\begin{array}{c} n \\ i \end{array} \right) \equiv \frac{n!}{i!(n-i)!}$$

为二项式系数. 实际计算时, 可以使用下面的递推关系.

$$\begin{aligned} B_0^0(x) &= 1, \quad B_{-1}^n = 0, \\ B_i^n(x) &= \frac{(b-x)B_i^{n-1}(x) + (x-a)B_{i-1}^{n-1}(x)}{b-a}, \quad i = 0, 1, 2, \cdots, n \\ B_{n+1}^n(x) &= 0 \end{aligned} \tag{2.5.145}$$

现在考虑由下式定义的线性组合

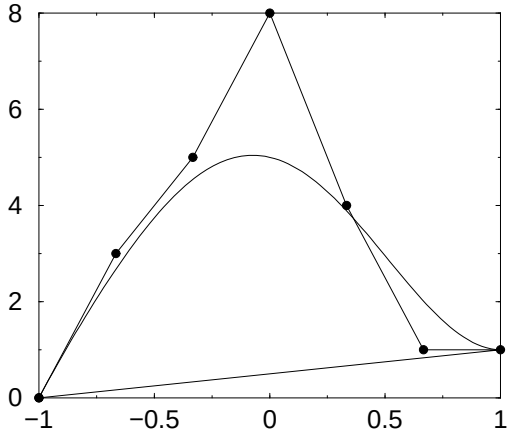
$$P_n(x) = \sum_{i=0}^{n} p_i B_i^n(x) \tag{2.5.146}$$

并称其为Bézier 多项式, 由此式决定的曲线称为Bézier 曲线. 因为每一个$B_i^n(x)$ 都是n 次多项式, 所以$P_n(x)$ 也是一个n 次多项式, 这个多项式的好处在于, 由p_i 的数值就可以大致看出$P_n(x)$ 的图像, 例如, $x = a$ 时, 由于当$i \neq 0$ 时$B_i^n(a) = 0$, $B_0^n(a) = 1$, 所以p_0 给出了$P_n(a)$ 的值. 同样, $P_n(b) = p_n$. 为了看出其它p_i 与由$P_n(x)$ 所决定的曲线之间的关系, 我们定义一组控制点如下, 令

$$t_j = a + \frac{j\,(b-a)}{n}$$

并指定一组点$(t_i, p_i), i = 0, 1, 2 \cdots, n$ 为一组控制点, 则由(t_0, p_0)出发, 经$(t_1, p_1), (t_2, p_2), \cdots, (t_n, p_n)$再回到$(t_0, p_0)$ 的线段构成的多边形称为Bézier 多边形, 而由这一组p_i 所决定的$P_n(x)$ 的图像则与这一多边形的形状紧密相关. 图(7.2) 给出了一个例子, 图中虚线代表Bézier 多边形, 圆点为控制点, 实线为对应的Bézier 曲线, 几者之间的关系是一目了然的.

Figure 2.3: 一组控制点及对应的Bézier多项式和Bézier曲线



CAD 的任务通常是你的大脑中有关于某条曲线的一个设想, 这时便可设置一组控制点, 用Bézier 多项式画出曲线, 通过不断地调节控制点而达到设计要求.

这里只对Bézier 逼近做了一个十分简要地介绍, 需要更多了解这一内容的同学可参看有关资料.

2.5.9 多元函数的插值及逼近

多元函数的逼近问题原则上可以通过推广前述各种方法来进行, 例如, 如果定义

$$l_k(x) = \prod_{i=1, i\neq k}^{n} \frac{x - x_i}{x_k - x_j} \tag{2.5.147}$$

则Lagrange 公式(2.5.116)可写为

$$P(x) = \sum_{k=1}^{n} y_k \, l_k(x) \tag{2.5.148}$$

这一公式可直接推广到二维, 如果一个二元函数$z = f(x, y)$ 在某一二维区域内的一组点$(x_i, y_j), i = 1, 2, \cdots, n; j = 1, 2, \cdots, m$ 上的值z_{ij} 已知, 则其Lagrange 插值公式为

$$f(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} z_{ij} l_i(x) l_j(y) \tag{2.5.149}$$

练习

证明由*(2.5.147)*定义的$l_i(x)$ 满足关系$l_i(x_j) = \delta_{ij}$, 并进而证明公式*(2.5.148)*和*(2.5.149)*.

三次样条插值也可以推广到二维, 假定函数$z = f(x, y)$ 在矩形区域$a \le x \le b, c \le y \le d$ 上的一组网格分点$a = x_1 < x_2 < \cdots < x_n = b,\ c = y_1 < y_2 < \cdots < y_m = d$ 上给定, 我们可以构造一组分片三次多项式$R_{ij}(x, y)$(对x 和y 均为三次), 定义在子区域$x_{i-1} \le x \le x_i, y_{j-1} \le y \le y_j$ 上, 则

$$\begin{aligned} S(x,y) &= R_{ij}(x,y) \ \text{如果}(x_{i-1} \le x \le x_i, y_{j-1} \le y \le y_j) \\ & i = 2, 3, \cdots, n; j = 2, 3, \cdots, m \end{aligned} \tag{2.5.150}$$

$R_{ij}(x, y)$ 可以类似于一维三次样条函数通过要求由$S(x, y)$ 给出的曲面适当的光滑而决定, 其推导过程较繁, 但并不难. 我们只给出最终结果. 定义

$$\begin{aligned} h_i &= x_{i+1} - x_i \\ \tau_j &= y_{j+1} - y_j \\ u_i &= \frac{x - x_{i-1}}{h_{i-1}} \\ v_j &= \frac{y - y_{j-1}}{\tau_{j-1}} \end{aligned} \tag{2.5.151}$$

则

$$R_{ij}(x,y) = \begin{bmatrix} u_i^3 & u_i^2 & u_i & 1 \end{bmatrix} A G_{ij} A^T \begin{bmatrix} v_j^3 \\ v_j^2 \\ v_j \\ 1 \end{bmatrix} \tag{2.5.152}$$

其中

$$A = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.5.153}$$

$$G_{ij} = \begin{bmatrix} z_{i-1,j-1} & z_{i-1,j} & \tau_{j-1} g_{i-1,j-1} & \tau_{j-1} g_{i-1,j} \\ z_{i,j-1} & z_{i,j} & \tau_{j-1} g_{i,j-1} & \tau_{j-1} g_{i,j} \\ h_{i-1} m_{i-1,j-1} & h_{i-1} m_{i-1,j} & h_{i-1}\tau_{j-1} l_{i-1,j-1} & h_{i-1}\tau_{j-1} l_{i-1,j} \\ h_{i-1} m_{i,j-1} & h_{i-1} mi, j & h_{i-1}\tau_{j-1} l_{i,j-1} & H_{i-1}\tau_{j-1} l_{i,j} \end{bmatrix} \tag{2.5.154}$$

而$m_{ij} = \frac{\partial S(x,y)}{\partial x}_{|x=x_i, y=y_j}$, $g_{ij} = \frac{\partial S(x,y)}{\partial y}_{|x=x_i, y=y_j}$, $l_{ij} = \frac{\partial^2 S(x,y)}{\partial x \partial y}_{|x=x_i, y=y_j}$, 分别满足下面的方程组

$$\begin{aligned} & h_i m_{i-1,j} + 2(h_i + h_{i-1}) m_{ij} + h_{i-1} m_{i+1,j} \\ &= \frac{3}{h_{i-1} h_i} [h_{i-1}^2 z_{i+1,j} + (h_i^2 - h_{i-1}^2) z_{ij} - h_i^2 z_{i-1,j}] \\ & i = 2, 3, \cdots, n-1; \ \ j = 1, 2, \cdots, m \end{aligned} \tag{2.5.155}$$

$$\begin{aligned}&\tau_j g_{i,j-1} + 2(\tau_j + \tau_{j-1})g_{ij} + \tau_{j-1}g_{i,j+1}\\ &= \frac{3}{\tau_{j-1}\tau_j}[\tau_{j-1}^2 z_{i,j+1} + (\tau_j^2 - \tau_{j-1}^2)z_{ij} - \tau_j^2 z_{i,j-1}]\\ &i = 1, 2, \cdots, n; \;\; j = 2, 3, \cdots, m-1\end{aligned} \tag{2.5.156}$$

$$\begin{aligned}&\tau_j l_{i,j-1} + 2(\tau_j + \tau_{j-1})l_{ij} + \tau_{j-1}l_{i,j+1}\\ &= \frac{3}{\tau_{j-1}\tau_j}[\tau_{j-1}^2 m_{i,j+1} + (\tau_j^2 - \tau_{j-1}^2)m_{ij} - \tau_j^2 m_{i,j-1}]\\ &i = 1, 2, \cdots, n; \;\; j = 2, 3, \cdots, m-1\end{aligned} \tag{2.5.157}$$

$$\begin{aligned}&h_i l_{i-1,j} + 2(h_i + h_{i-1})l_{ij} + h_{i-1}m_{i+1,j}\\ &= \frac{3}{h_{i-1}h_i}[h_{i-1}^2 g_{i+1,j} + (h_i^2 - h_i - 1^2)g_{ij} - h_i^2 g_{i-1,j}]\\ &i = 2, 3, \cdots, n-1; \;\; j = 1, m\end{aligned} \tag{2.5.158}$$

下面我们来分析一下插值问题是否确定. (2.5.155)共有$m(n-2)$ 个方程, (2.5.156)共有$(m-2)n$ 个方程, (2.5.157)共有$(m-2)n$ 个方程, (2.5.158)共有$2(n-2)$ 个方程, 总共有$3mn - 2(m+n) - 4$ 个方程, 而未知数的数目为$3mn$ 个, 因此, 为了求解, 尚需补充$2(m+n)+4$ 个条件, 这些条件通常通过给定边界上的一组函数$z = f(x, y)$ 的导数值来补足, 即给定

$$\begin{aligned}&\frac{\partial z}{\partial x}_{|x=x_i, y=y_j} && i = 1, n; \;\; j = 1, 2, \cdots, m\\ &\frac{\partial z}{\partial y}_{|x=x_i, y=y_j} && i = 1, 2, \cdots, n; \;\; j = 1, m\\ &\frac{\partial^2 z}{\partial x \partial y}_{|x=x_i, y=y_j} && i = 1, n; \;\; j = 1, m\end{aligned} \tag{2.5.159}$$

上面第一行共有$2m$ 个条件, 第二行共有$2n$ 个条件, 第三行共有4 个条件, 加上原有的方程, 构成一个完整的线性代数方程组求解问题. 关于这一问题的算法实现, 我们将不再讨论, 有兴趣的同学可参看有关专著.

2.6 快速付里叶变换

2.6.1 付里叶变换

付里叶变换在物理学的很多问题中占有十分重要的地位. 这包括实验数据的处理, 很多理论问题的计算等. 有一些物理问题本身就是用付里叶变换的语言来表述的, 利如量子力学中波函数在坐标空间和动量空间的表示就对应于一对付里叶变换. 这一节我们简要复习一下付里叶变换的理论.

Table 2.1: 付里叶变换的对称性质

如果...	则...
$h(t)$ 为实函数	$H(-f) = H^*(f)$
$h(t)$ 为虚函数	$H(-f) = -H^*(f)$
$h(t)$ 为偶函数	$H(-f) = H(f)$
$h(t)$ 为奇函数	$H(-f) = -H(f)$
$h(t)$ 为实偶函数	$H(f)$ 为实偶函数
$h(t)$ 为实奇函数	$H(f)$ 为虚奇函数
$h(t)$ 为虚偶函数	$H(f)$ 为实偶函数
$h(t)$ 为虚奇函数	$H(f)$ 为实奇函数

一个物理过程可以在“时间区域”上给予描述, 其自变量取为t, 物理过程用函数$h(t)$ 表示; 也可以在“频率区域”上描述, 其自变量取为f , 物理过程用函数$H(f)$ 表示, 这里$H(f)$ 通常是一个复数. “时间” 和“频率”可以代表各种互相对偶的一对变量, 如信号处理中的时间和频率, 波动过程中的坐标和波数, 量子力学中的坐标和动量, 时间和能量等.

在物理学中, 一般把$h(t)$ 和$H(f)$ 看为同一物理过程的两种不同的表示, 如量子力学中的$\psi(x)$ 和$\phi(p)$ 等. 两种表示之间的变换通过付里叶变换来表示

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt \\ h(t) &= \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df \end{aligned} \tag{2.6.160}$$

如果t 用秒表示, 则f 的单位为赫兹. 在物理学中, 更常用的习惯是用$\omega = 2\pi f$ 表示频率, 相应地, $H(\omega) \equiv H(f)|_{f=\omega/2\pi}$, 方程(2.6.160) 成为

$$\begin{aligned} H(\omega) &= \int_{-\infty}^{\infty} h(t) e^{i\omega t} dt \\ h(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} H(\omega) e^{-i\omega t} d\omega \end{aligned} \tag{2.6.161}$$

当然, 上式也可以写成对称的形式, 因为从数值计算的角度看, 方程(2.6.160)中包含有较少的2π 因子, 计算上比较方便, 因此在下面的讨论中我们将采用方程(2.6.160), 在实际应用时, 可把所用的形式变为这里讨论的形式, 计算完后再变换回去.

从定义显见, 付里叶变换是线性的, 同时, 付里叶变换还有表(2.1)所列的性质, 这些性质可以通过定义来证明. 在实际工作中，这些性质可以用来提高计算效率.

在以后的讨论中, 我们用下面的记号表示由方程(2.6.160)定义的一对付里叶变换

$$h(t) \Longleftrightarrow H(f) \tag{2.6.162}$$

利用定义，可以得到下面对应的变换对

$$h(at) \iff \frac{1}{|a|}H\left(\frac{f}{a}\right) \quad \text{时间标度} \tag{2.6.163}$$

$$\frac{1}{|b|}h\left(\frac{t}{b}\right) \iff H(bf) \quad \text{频率标度} \tag{2.6.164}$$

$$h(t-t_0) \iff H(f)e^{2\pi i f t_0} \quad \text{时间平移} \tag{2.6.165}$$

$$h(t)e^{-2\pi i f_0 t} \iff H(f-f_0) \quad \text{频率平移} \tag{2.6.166}$$

对于两个函数$g(t)$ 和$h(t)$, 其对应的付里叶变换分别为$G(f)$ 和$H(f)$, 我们可以定义两个合成函数, 一个是卷积$g * h$, 由下式定义

$$g * h \equiv \int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau \tag{2.6.167}$$

$g * h$ 是时间区域上的函数, 其付里叶变换满足下面的卷积定理

$$g * h \Longleftrightarrow G(f)H(f) \Longleftrightarrow h * g \quad \text{卷积定理} \tag{2.6.168}$$

另一个是相关函数$C[g,h]$, 定义为

$$C[g,h] \equiv \int_{-\infty}^{\infty} g(\tau+t)h(\tau)d\tau \tag{2.6.169}$$

$C[g,h]$是时间区域上的函数, 其付里叶变换由下面的相关定理给出

$$C[g,h] \Longleftrightarrow G(f)H(-f) \quad \text{相关定理} \tag{2.6.170}$$

如果g 和h 是实函数, 则上式可写为

$$C[g,h] \Longleftrightarrow G(f)H^*(f) \tag{2.6.171}$$

一个函数与其自身的相关函数称为自相关函数, 在此情况下, 方程(2.6.171) 成为

$$C[g,g] \Longleftrightarrow |G(f)|^2 \tag{2.6.172}$$

自相关函数通常称为功率谱, 而总功率定义为

$$P \equiv \int_{\infty}^{\infty} |h(t)|^2 dt = \int_{\infty}^{\infty} |H(f)|^2 df \tag{2.6.173}$$

上式通常称为Parseval 定理.

2.6.2 离散付里叶变换

为了在计算机上实现付里叶变换, 需要对付里叶变换进行离散化处理. 这一节介绍离散付里叶变换, 首先介绍对连续付里叶变换离散化的抽样方法. 在非常一般的情况下, 对函数$h(t)$ 在时间的等间距上进行取样. 令Δ 为取样间隔, 则对$h(t)$ 的取样成为

$$h_n = h(n\Delta) \quad n = \cdots, -3, -2, -1, 0, 1, 2, 3, \cdots \tag{2.6.174}$$

Δ 的倒数称为取样频率, 它反映了取样所能代表的频率上限. 对于一给定的取样间隔Δ , 存在一个截止频率f_c, 由下式给出,

$$f_c = \frac{1}{2\Delta} \tag{2.6.175}$$

这一频率通常称为Nyquist 临界频率, 如果对一个具有Nyquist 临界频率正弦(或余弦)函数进行取样, 则如果一个取样点位于正的波峰, 则下一个取样点将位于负的波峰, 再下一个又位于正的波峰, ... 与Nyquist 临界频率相连系, 有下面的重要结论:
(1) 如果一连续函数$h(t)$ 只有有限带宽, 且带宽小于f_c, 也就是说, 对所有的频率$|f| > f_c$, 都有$H(f) = 0$, 则函数$h(t)$ 由其取样值h_n 完全决定. 实际上, $h(t)$ 可由下式显式给出

$$h(t) = \Delta \sum_{-\infty}^{\infty} h_n \frac{\sin[2\pi f_c(t - n\Delta)]}{\pi(t - n\Delta)} \tag{2.6.176}$$

证明如下, $h(t)$ 与$H(f)$ 之间由下式联系

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt \\ h(t) &= \int_{-f_c}^{f_c} H(f) e^{-2\pi i f t} df \end{aligned} \tag{2.6.177}$$

上式中的第二个积分限为有限值, 这是由于在积分限外, $H(f) = 0$. 把$H(f)$ 延拓到整个频率域上, 记延拓后的函数为$\tilde{H}(f)$, 满足

$$\begin{cases} \tilde{H}(f) = H(f) & -f_c \le f \le f_c \\ \tilde{H}(f + 2f_c) = \tilde{H}(f) \end{cases} \tag{2.6.178}$$

则可得到一对新的付里叶变换

$$\begin{aligned} \tilde{H}(f) &= \int_{-\infty}^{\infty} \tilde{h}(t) e^{2\pi i f t} dt \\ \tilde{h}(t) &= \int_{-\infty}^{\infty} \tilde{H}(f) e^{-2\pi i f t} df \end{aligned} \tag{2.6.179}$$

(2.6.179) 的第二式可写为

$$\begin{aligned}\tilde{h}(t) &= \int_{-\infty}^{\infty}\tilde{H}(f)e^{-2\pi ift}df \\ &= \sum_{m=-\infty}^{\infty}\int_{(2m-1)f_c}^{(2m+1)f_c}\tilde{H}(f)e^{-2\pi ift}df \\ &= \sum_{m=-\infty}^{\infty}e^{-2\pi i2mf_ct}\int_{-f_c}^{f_c}H(f)e^{-2\pi ift}df \\ &= \sum_{m=-\infty}^{\infty}e^{-2\pi i2mf_ct}h(t) \end{aligned} \tag{2.6.180}$$

在得到上式时, 我们利用了式(2.6.178), 上式中的求和当$f_ct=\frac{n}{2}$, $n=0,\pm1,\pm2,\cdots$ 时, 为无穷大, 对其它t 值, 求和为0, 可以证明(请同学作为一个练习, 证明下述关系)

$$\sum_{m=-\infty}^{\infty}e^{-2\pi i2mf_ct}=\Delta\sum_{n=-\infty}^{\infty}\delta(t-n\Delta) \tag{2.6.181}$$

在得出上式时, 使用了关系$f_c=\frac{1}{2\Delta}$, 从而有

$$\tilde{h}(t)=\Delta\sum_{n=-\infty}^{\infty}\delta(t-n\Delta)h(t) \tag{2.6.182}$$

$$\begin{aligned}\tilde{H}(f) &= \int_{-\infty}^{\infty}\tilde{h}(t)e^{2\pi ift}dt \\ &= \Delta\sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty}\delta(t-n\Delta)h(t)e^{2\pi ift}dt \\ &= \Delta\sum_{n=-\infty}^{\infty}h(n\Delta)e^{2\pi ifn\Delta} \end{aligned} \tag{2.6.183}$$

在区间$[-f_c,f_c]$ 内, 上式就等于$H(f)$, 于是

$$\begin{aligned}h(t) &= \int_{-f_c}^{f_c}H(f)e^{-2\pi ift}df \\ &= \Delta\sum_{n=-\infty}^{\infty}h(n\Delta)\int_{-f_c}^{f_c}e^{-2\pi if(t-n\Delta)}df \\ &= \Delta\sum_{n=-\infty}^{\infty}h(n\Delta)\frac{\sin[2\pi f_c(t-n\Delta)]}{\pi(t-n\Delta)} \end{aligned} \tag{2.6.184}$$

这是一个十分重要的定理, 它告诉我们, 一个有限带宽的信号所包含的信息较之一个一般的连续函数是非常小的(严格的讲, 二者的信息量之比为零). 在通常情况下, 物理上遇到的信号都是有限带宽的, 在这种情况下, 只要取取样间隔Δ 小于或等于二倍最大频率

的倒数, 则可以完整的地得到这一信号所包含的信息.

(2) 如果一个信号不是有限带宽的或Nyquist 临界频率f_c 小于信号的最大频率, 则位于$[-f_c, f_c]$ 外的付里叶分量将移动到$[-f_c, f_c]$ 区间内并叠加在这一区间的分量上面, 这一现象称为波形重叠. 当一个信号进行离散取样后, 消除波形重叠几乎是不可能的. 克服这一困难的方法是, 首先找到信号的自然带度, 或者先用某种方法对信号进行滤波, 把信号限制于一频率间隔内, 然后由最大频率确定取样间隔.

现在, 我们来讨论如何由有限数目的样点来计算其付里叶变换. 假定我们有N 个相继的取样信号

$$h_k \equiv h(t_k), \quad t_k \equiv k\Delta \quad k = 0, 1, 2, \cdots, N-1 \tag{2.6.185}$$

为简单起见, 我们进一步假定N 为偶数. 如果$h(t)$ 只在有限时间区间内非零, 则$h(t)$ 的取样应包括这个非零区间, 如果$h(t)$ 在整个时间区间中非零, 则取样应包括足够大的区间以使区间内的$h(t)$ 尽可能好的代表整个时间区间上的$h(t)$.

如果我们只有N 个数作为输入, 则显然最多只能得到N 个独立的数作为输出. 因此, 要从N 个样品点计算$[-f_c, f_c]$ 内的所有f 值的付里叶变换$H(f)$ 是不可能的, 因此, 我们只计算一系列离散点

$$f_n \equiv \frac{n}{N\Delta}, \quad n = -\frac{N}{2}, -\frac{N}{2}+1, \cdots, \frac{N}{2} \tag{2.6.186}$$

上式中n 的上限值正好是Nyquist 临界频率f_c , 上式中的n 的总数实际上为$N+1$ 个, 而不是N 个, 但n 的上限值和下限值并不独立, 实际上, 它们代表同一个点, 这样, 独立频率的总数为N 个.

把(2.6.160) 中的积分变为一个离散求和

$$H(f_n) = \int_{-\infty}^{\infty} h(t)e^{2\pi ift}dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi i f_n t_k}\Delta = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi ikn/N} \tag{2.6.187}$$

在最后一个等式中, 使用了式(2.6.174)和(2.6.186), 上式的最后求和称为N 个数据点h_k 的离散付里叶变换. 记为

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{2\pi ikn/N} \tag{2.6.188}$$

它把N 个数h_k 映射为N 个数H_n, 这一求和不依赖于任何量纲参数(h_k 与H_n 具有相同的量纲) , 连续付里叶变换在离散点上的值与离散付里叶变换由式(2.6.187) 给出为

$$H(f_n) = \Delta H_n \tag{2.6.189}$$

到此为此, 我们总是取(2.6.188)中的n 的取值范围为$-N/2$ 到$N/2$. 实际上, 从(2.6.188)可知, H_n 对于n 具有周期N, 即$H_{-n} = H_{N-n}$, 因此, 我们也可以取$n = 0, 1, 2, \cdots, N-1$,

这样, n 和k 的变化范围完全相同. 在这种取法下, $0 < f < f_c$ 对应于$1 \le n \le N/2-1$, 负频率$-f_c < f < 0$ 对应于$N/2+1 \le n \le N-1$, 而f_c 和$-f_c$ 均对应于$n = N/2$.

离散付里叶变换的逆变换由下式给出

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n e^{-2\pi ikn/N} \tag{2.6.190}$$

这一表达式与(2.6.188)的差别在于改变e 指数上的符号并把结果除以N, 因此计算离散付里叶变换的方法(或程序)只需做微小修改便可用来计算离散付里叶变换的逆变换.

离散付里叶变换具有与连续付里叶变换几乎完全相同的性质, 例如表2.1 中所列的对称性质对于离散付里叶变换也成立, 只是此时h_k 对应于$h(t)$, h_{N-k} 对应于$h(-t)$, H_n 对应于$H(f)$, 而H_{N-n} 对应于$H(-f)$. 例如说H_n 为偶时, 我们是指$H_n = H_{N-n}$, 等等.

2.6.3 快速付里叶变换

首先, 让我们看一下对N 个数据点作一次离散付里叶变换所需的计算次数, 定义

$$W \equiv e^{2\pi i/N} \tag{2.6.191}$$

则(2.6.188)可写为

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k \tag{2.6.192}$$

即如果把H_n 和h_k 看作矢量, 则上式相当于一矩阵与矢量的乘积, 矩阵的第(n,k) 个分量等于W 的nk 次方. 显然, 矩阵与矢量的乘积需要做N^2 次乘法, 总的计算量等于N^2 次乘法加上形成矩阵所需的计算次数, 其阶为N^2. 直到1960年中期, 这是通常的看法. 1965年, IBM 公司的J. W. Cooley 和J. W. Tukey 提出了一个离散付里叶变换的快速算法, 把计算量从N^2 降低到$N\log_2 N$. 为了看出这一改进的重要意义, 考虑一个$N = 524288$ 的离散付里叶变换, 这一变换用快速方法在一个486/33 计算机上约需60 秒, 而如果用(2.6.192)直接计算, 则需要大约20天.

事实上, 离散付里叶变换的快速算法可以追溯到1903年, 当时, Runge 提出了12点和24点的算法. 1942年, Danielson 和Lanczos 提出了一种最优算法, 但因为当时还没有电子计算机, 因而没有引起重视, 只是在J. W. Cooley和J. W. Tukey 提出后, 才引起了广泛的注意. 下面我们给出这一算法的一个推导, 首先我们证明, 一个N 点(N 为偶数)的离散付里叶变换可以写为二个$N/2$ 点的离散付里叶变换之和, 其中一个由处于原来N 个点的偶数位置的点构成, 另一个则由处于原来N 个点的奇数位置的点构成, 这一结论称

为Danielson—Lanczos 定理. 证明如下:

$$\begin{aligned} H_n &= \sum_{k=0}^{N-1} e^{2\pi ink/N} h_k \\ &= \sum_{k=0}^{N/2-1} e^{2\pi in(2k)/N} h_{2k} + \sum_{k=0}^{N/2-1} e^{2\pi in(2k+1)/N} h_{2k+1} \\ &= \sum_{k=0}^{N/2-1} e^{2\pi ink/(N/2)} h_{2k} + W^n \sum_{k=0}^{N/2-1} e^{2\pi ink/(N/2)} h_{2k+1} \\ &= H_n^e + W^n H_n^o \end{aligned} \tag{2.6.193}$$

上式称为Danielson—Lanczos公式. 在上式中, W 由式(2.6.191) 给出, H_n^e表示由长度为$N/2$, 由原h_k 的偶分量组成的数据序列的离散付里叶变换; 而H_n^o表示由长度为$N/2$, 由原h_k 的奇分量组成的数据序列的离散付里叶变换. 请注意上式中最后一行中n 的取值仍然是0 到N, 但H_n^e 和H_n^o 的周期为$N/2$, 因此实际上只要各计算$N/2$ 个分量, 另外$N/2$ 个由周期性得到.

在有了上述结果后, 我们可以分别对H_n^e 和H_n^o 进行同样的变换, 即代替计算长度为$N/2$ 的离散付里叶变换, 我们可以计算两个长度为$N/4$ 的离散付里叶变换, 分别对应于$N/2$ 序列中的偶数分量序列和奇数分量序列. 也就是说, 我们可以定义H_n^{ee}, H_n^{eo}, $\cdots$ 分别为由顺次划分中的偶偶序列, 偶奇序列, $\cdots$ 的离散付里叶变换. 为了这一划分可以一直进行下去, 我们要求N 应为2 的某个幂次, 如果在实际工作中得到的数据不是2 的幂次, 则可在原数据上增加若干个0使分量的总数成为2 的幂次以便使用快速方法. 在N为2 的幂次的条件下, 这种划分可以一直进行到长度为1 的变换, 而长度为1 的离散付里叶变换就是该数本身. 这样, 我们得到N 个一点离散付里叶变换, 每一个对应于一种$\log_2 N$ 个e 和o 的排列, 即

$$H_n^{eeoe\cdots oo} = h_k \tag{2.6.194}$$

实际上, 对于一点变换, 其周期为1, 因而上式左边的下标n 是多余的. 上式右边的下标k与左边的上标对应, 对于一个给定的e 和o 的排列, 唯一地确定了方程右边的下标k.

下面的问题是给出k 与e 和o 的排列之间的对应关系, 这个关系是十分简单的, 首先, 令$e = 0$, $o = 1$, 则e 和o 的一种排列与一个2进制数相对应, 我们把这个2进制数前后颠倒, 其结果就给出k 的2 进制表示. 例如, 对于$N = 2^3$,

$$\begin{aligned} &eee \to 000 \to k = (000)_2 = 0 \quad eeo \to 001 \to k = (100)_2 = 4 \\ &eoe \to 010 \to k = (010)_2 = 2 \quad eoo \to 011 \to k = (110)_2 = 6 \\ &oee \to 100 \to k = (001)_2 = 1 \quad oeo \to 101 \to k = (101)_2 = 5 \\ &ooe \to 110 \to k = (011)_2 = 3 \quad ooo \to 111 \to k = (111)_2 = 7 \end{aligned} \tag{2.6.195}$$

为了说明这个结果, 我们仍然以$N = 2^3$ 为例, 下面是每一步变换步骤

$$\begin{aligned} H_n &= H_n^e + W^n H_n^o \\ &= (H_n^{ee} + W^{2n} H_n^{eo}) + W^n (H_n^{oe} + W^{2n} H_n^{oo}) \\ &= (H_n^{eee} + W^{4n} H_n^{eeo}) + W^{2n} (H_n^{eoe} + W^{4n} H_n^{eoo}) + \\ &\quad W^n (H_n^{oee} + W^{4n} H_n^{oeo}) + W^{3n} (H_n^{ooe} + W^{4n} H_n^{ooo}) \end{aligned} \tag{2.6.196}$$

另一方面,

$$H_n = h_0 + W^n h_1 + W^{2n} h_2 + W^{3n} h_3 + W^{4n} h_4 + W^{5n} h_5 + W^{6n} h_6 + W^{7n} h_7 \tag{2.6.197}$$

比较(2.6.196)和(2.6.197) 可知, k 与e 和o 的排列之间的关系由前面的论断给出, 我们在此仅仅对N 的一个特殊值给出了验证, 但可以一般地证明它的正确性. 式(2.6.196)也给出了一个计算步骤, 利用关系$W^0 = 1$ 及$W^{N/2} = -1$, 我们看到, 如果把h_k 按照0, 4, 2, 6, 1, 5, 3, 7 的次序排列(这一排列方式正好是k 的2 进制表示前后颠倒后的数的顺序排列), 则第一步可对相邻的数作二点变换, 如式(2.6.196)中最后一行的每一个括号所示, 在这一步, 只需计算$n = 0$ 和$n = 1$,$n = 2, \cdots, 7$ 由周期性条件给出, 需要做$2 \times 4 = 8$ 次乘法; 第二步由式(2.6.196) 的第二行给出, 此时需计算$n = 0, 1, 2, 3$, 需要做$4 \times 2 = 8$ 次乘法; 最后一步由(2.6.196)的第一行给出, 需要做$8 \times 1 = 8$ 次乘法. 因此, 总的乘法数为$8 \times 3 = 8 \log_2 8$ 次. 上面的算法可立即推广到一般N(为2 的幂次), 我们给出计算离散付里叶变换的算法如下

(1) 把h_k 按照k 的2 进制表示前后颠倒后的数的顺序重新排列;
(2) 对$max = 2, 4, 8, \cdots, 2^i, \cdots, N$;
对于$m = 0, 1, \cdots, max$;
对于相邻的一对数做二点离散付里叶变换.

到此为止, 我们讨论了$N = 2^\gamma$, γ 为一整数的快速算法(在一般文献中称为基2 算法), 在实现时, 我们先对数据进行二进位次序颠倒, 再使用Danielson—Lanczos 公式. 当然, 我们也可以先使用Danielson—Lanczos 公式, 再做二进位次序颠倒. 除了基2 算法外, 也可以进行基4 算法, 基8 算法等, 也可以通过推广Danielson—Lanczos 公式, 建立以小的素数为基的算法. 对于绝大多数问题, 基2 算法应作为首选算法, 如前面已经指出过的, 若原始数据的个数不是2 的幂次, 则可通过补零的办法凑齐.

作为例子, 我们来考虑区间$[-1, 1]$ 上Runge 函数的付里叶变换. Runge 函数由下式给出

$$h(x) = \frac{1}{1 + 25x^2}$$

把$h(x)$ 延拓到整个实轴上, 成为周期为2 的周期函数, 为了使用前面的程序, 考虑区间$[0,2]$, 由于已经进行了延拓, 因此$[-1,1]$ 和$[0,2]$ 均为一个周期, 其付里叶变换是相同的. 图(2.4)给出付里叶变换的计算结果

关于快速付里叶变换, 已有不少专著对其进行讨论, 希望进一步了解的同学可参看这些书籍. 但对于大部分实际应用, 本章的内容应已够用.

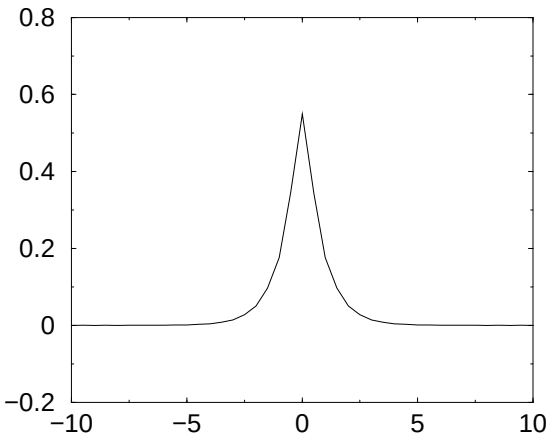


Figure 2.4: Runge 函数的付里叶变换

2.7 附录: 特殊函数的计算

Γ函数, B函数, 不完全Γ函数的计算

Γ函数由下面的积分定义:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{2.7.198}$$

如果z为一整数, 则有

$$n! = \Gamma(n+1) \tag{2.7.199}$$

Γ函数满足如下递推关系:

$$\Gamma(z+1) = z\Gamma(z) \tag{2.7.200}$$

如果已经知道了Γ函数在$\mathrm{Re}(z) > 1$时的值, 则可利用下述关系求得$\mathrm{Re}(z) < 1$时的值.

$$\Gamma(1-z) = \frac{\pi}{\Gamma(z)\sin(\pi z)} = \frac{\pi z}{\Gamma(1+z)\sin(\pi z)} \tag{2.7.201}$$

$\Gamma(z)$ 在$0, -1, -2, \cdots$,上有单极点, 在复平面的其它地方解析.

Lanczos方法似乎是计算Γ函数的数值的最漂亮的方法, 对于一定的整数γ 和N 以及一定的系数$c_1, c_2, \cdots, c_N$, Lanczos的近似公式由下式给出:

$$\begin{aligned}\Gamma(z+1) &= \left(z+\gamma+\frac{1}{2}\right)^{z+\frac{1}{2}} e^{-\left(z+\gamma+\frac{1}{2}\right)} \\ &\times \sqrt{2\pi}\left[c_0+\frac{c_1}{z+1}+\frac{c_2}{z+2}+\cdots+\frac{c_N}{z+N}+\epsilon\right] \\ &\quad (\mathrm{Re}(z)>0) \end{aligned} \tag{2.7.202}$$

这里: $c_0 \approx 1$, ϵ代表误差, 对于$\gamma = 5, N = 6$可以得到$|\epsilon| < 2 \times 10^{-10}$

当$|z|$较大时, Γ函数的数值非常大, 极易上溢, 在实用上, 很多物理公式常常表示为Γ函数相除的形式而代表一个合理的数字. 因此, 代替计算Γ函数本身, 我们通常计算Γ函数的对数$\ln(\Gamma(z))$. 由上式可得到$\ln(\Gamma(z))$的公式如下:

$$\ln(\Gamma(z+1)) = \left(z+\frac{1}{2}\right)\ln\left(z+\gamma+\frac{1}{2}\right)-\left(z+\gamma+\frac{1}{2}\right) \tag{2.7.203}$$

$$+ \ln\left[\left(\sqrt{2\pi}(c_0+\frac{c_1}{z+1}+\frac{c_2}{z+2}+\cdots+\frac{c_N}{z+N}+\epsilon\right)\right] \tag{2.7.204}$$

这里

$$\begin{aligned} c_0 &= 1 \\ c_1 &= 76.18009173 \\ c_2 &= -86.50532033 \\ c_3 &= 24.01409822 \\ c_4 &= -1.231739516 \\ c_5 &= 0.120858003 \times 10^{-2} \\ c_6 &= -0.536382 \times 10^{-5} \end{aligned}$$

B函数由下式积分定义:

$$B(z,w) = B(w,z) = \int_0^1 t^{z-1}(1-t)^{w-1}dt \tag{2.7.205}$$

它与Γ函数之间满足如下简单的关系:

$$B(z,w) = \frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)} \tag{2.7.206}$$

因此, 很容易用Γ函数的结果计算B函数.

不完全Γ函数定义为:

$$P(a,x) \equiv \frac{\gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt \quad (a>0) \tag{2.7.207}$$

它具有下面的极限

$$P(a,0) = 0, \qquad P(a,\infty) = 1 \tag{2.7.208}$$

$P(a,x)$ 的补函数$Q(a,x) \equiv 1 - P(a,x)$ 的积分形式为:

$$Q(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt \quad (a>0) \tag{2.7.209}$$

$\gamma(a,x)$可以用级数表示出来:

$$\gamma(a,x) = e^{-x} x^a \sum_{n=0}^{\infty} \frac{\Gamma(a)}{\Gamma(a+1+n)} x^n \tag{2.7.210}$$

实际计算时, 并不需要对每个n都计算$\Gamma(a+1+n)$, 而通常是利用前一次的值递推计算.

$\Gamma(a,x)$可以用连分式表示出来,

$$\Gamma(a,x) = e^{-x} x^a \left(\frac{1}{x+} \frac{1-a}{1+} \frac{1}{x+} \frac{2-a}{1+} \frac{2}{x+} \cdots \right) \quad (x>0) \tag{2.7.211}$$

(2.7.210)对于较小的x $(x < a+1)$收敛较快, 而(2.7.211)对于大的x $(x > a+1)$ 收敛较快, 把二个式子结合起来, 就得到计算不完全Γ函数的方法.

2.7.1 误差函数的计算

误差函数和余误差函数定义为

$$\mathrm{erf(x)} = \frac{2}{\sqrt{\pi}} \int_0^{\mathrm{x}} \mathrm{e}^{-\mathrm{t}^2}\,\mathrm{dt} \tag{2.7.212}$$

$$\mathrm{erfc(x)} \equiv 1 - \mathrm{erf(x)} = \frac{2}{\sqrt{\pi}} \int_{\mathrm{x}}^{\infty} \mathrm{e}^{-\mathrm{t}^2}\,\mathrm{dt} \tag{2.7.213}$$

它们具有下面的极限形式

$$\mathrm{erf}(0) = 0, \quad \mathrm{erf}(\infty) = 1, \quad \mathrm{erfc}(0) = 1, \quad \mathrm{erfc}(\infty) = 0 \tag{2.7.214}$$

及对称关系

$$\mathrm{erf(-x)} = -\mathrm{erf(x)} \qquad \mathrm{erfc(-x)} = 2 - \mathrm{erfc(x)} \tag{2.7.215}$$

误差函数可用下面的近似逼近式计算其值, 误差小于10^{-7}.

$$\begin{aligned} \mathrm{erfc(x)} &= (1.061405429t^5 - 1.453152027t^4 + 1.421413741t^3 \\ &\quad -0.284496736t^2 + 0.254829592t)\exp(-x^2) \\ t &= \frac{1}{1+0.3275911x} \end{aligned}$$

这一函数目前已经在很多计算机上作为内部函数给出, 因此在使用本程序前, 请先检查你的机器上是否有此函数, 如果没有, 再使用本节的算法计算.

误差函数也可作为不完全Γ 函数的特殊形式给出,

$$\begin{aligned} \mathrm{erf(x)} &= P\left(\frac{1}{2}, x^2\right) \quad (x \geq 0) \\ \mathrm{erfc(x)} &= Q\left(\frac{1}{2}, x^2\right) \quad (x \geq 0) \end{aligned} \tag{2.7.216}$$

例题 硅片置于恒定杂质浓度(N_0)的气体中, 试确定其内部浓度随时间的变化.

解: 本问题的定解方程为:(梁昆淼, 数学物理方法, P259)

$$\begin{cases} u_t - a^2 u_{xx} = 0 \\ u|_{x=0} = N_0 \\ u|_{t=0} = 0 \end{cases}$$

其解可写为:

$$u(x,t) = N_0 \mathrm{erfc}(\frac{\mathrm{x}}{2\mathrm{a}\sqrt{\mathrm{t}}})$$

为了得到解的图形, 可利用本节的程序计算余误差函数的值. 作为练习, 请同学完成具体计算.

2.7.2 指数积分的计算

指数积分是指由下式定义的函数

$$E_1(z) = \int_z^\infty \frac{e^{-t}}{t} dt \quad (|\arg z| < \pi) \tag{2.7.217}$$

这里z 为一辐角小于π 的复数, 与此相关的函数还有

$$E_n(z) = \int_1^\infty \frac{e^{-zt}}{t^n} dt \quad (n = 2, 3, \cdots; \mathrm{Re} z > 0) \tag{2.7.218}$$

它们之间满足下面的关系

$$E_{n+1}(z) = \frac{1}{n}[e^{-z} - zE_n(z)] \quad (n = 1, 2, 3, \cdots) \tag{2.7.219}$$

对$E_n(x)$, 有下面的不等式

$$\frac{1}{x+n} < e^x E_n(x) \leq \frac{1}{x+n-1} \tag{2.7.220}$$

对于实数宗量, 当$0 \leq x \leq 1$ 时, $E_1(x)$ 可用多项式来近似; 当$x > 1$ 时, $xe^x E_1(x)$ 可用分式有理函数来近似, 下面是有关公式:

$$\begin{aligned} E_1(x) &= 0.00107857x^5 - 0.00976004x^4 + 0.05519968x^3 \\ &\quad -0.24991055x^2 + 0.99999193x - 0.57721566 - \ln(x) \quad (0 \leq x \leq 1) \end{aligned}$$

$$
\begin{aligned}
E_1 &= \frac{P(t)}{Q(t)}\frac{\exp(-x)}{x} \quad (x>1)\\
t &= \frac{1}{x}\\
P(t) &= 0.2677737343t^4 + 8.6347608925t^3 + 18.0590169730t^2 + 8.5733287401t + 1\\
Q(t) &= 3.9584969228t^4 + 21.0996530827t^3 + 25.6329561486t^2 + 9.5733223454t + 1
\end{aligned}
$$

2.7.3 整数阶贝塞尔函数及虚宗量贝塞尔函数的计算

贝塞尔函数在物理研究中经常遇到, 对于任意实数ν, 贝塞尔函数定义为:

$$J_\nu(z) = (\frac{z}{2})^\nu \sum_{k=0}^{\infty} \frac{\left(-\frac{z^2}{4}\right)^k}{k!\Gamma(\nu+k+1)} \tag{2.7.221}$$

这个级数对于任何z 都是收敛的, 但当$z \gg 1$ 时, 对于实际计算并无多大用处.

当ν 不是整数时, 第二类贝塞尔函数$Y_\nu(z)$ 由下式定义:

$$Y_\nu(z) = \frac{J_\nu(z)\cos(\nu\pi) - J_{-\nu}(z)}{\sin(\nu\pi)} \tag{2.7.222}$$

当ν 为整数n 时, Y_n 可由下面的极限得到.

$$Y_n(z) = \lim_{\nu\to n} Y_\nu(z) \tag{2.7.223}$$

对于小的x, (例如, 当$x \ll \nu$ 时), 贝塞尔函数具有如下渐近表达式:

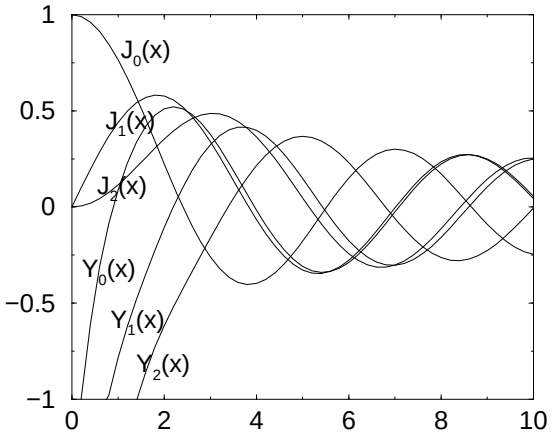
$$
\begin{aligned}
J_\nu(x) &\sim \frac{1}{\Gamma(\nu+1)}\left(\frac{x}{2}\right)^\nu \quad (\nu \geq 0)\\
Y_0(x) &\sim \frac{2}{\pi}\ln(x)\\
Y_\nu(x) &\sim -\frac{\Gamma(\nu)}{\pi}\left(\frac{x}{2}\right)^{-\nu} \quad (\nu > 0)
\end{aligned}
\tag{2.7.224}
$$

而对于大的x, (如$x \gg \nu$), 贝塞尔函数定性与三角函数一致, 但其振幅按照$1/\sqrt{x}$ 的方式衰减, 其渐近形式为:

$$
\begin{aligned}
J_\nu(x) &\sim \sqrt{\frac{2}{\pi x}}\cos\left(x - \frac{1}{2}\nu\pi - \frac{1}{4}\pi\right)\\
Y_\nu(x) &\sim \sqrt{\frac{2}{\pi x}}\sin\left(x - \frac{1}{2}\nu\pi - \frac{1}{4}\pi\right)
\end{aligned}
\tag{2.7.225}
$$

图(4.1)给出了前几个贝塞尔函数的图形.

Figure 2.5: $J_0(x), \cdots, J_2(x)$, $Y_0(x), \cdots, Y_2(x)$ 的图形



贝塞尔函数满足如下的递推关系:

$$\begin{aligned} J_n'(x) &= \frac{nJ_n(x)}{x} - J_{n+1}(x) \\ J_{n+1}(x) &= \frac{2n}{x}J_n(x) - J_{n-1}(x) \\ Y_{n+1}(x) &= \frac{2n}{x}Y_n(x) - Y_{n-1}(x) \end{aligned} \tag{2.7.226}$$

第一个可用来计算贝塞尔函数的导数, 第三个递推式对于n 增加的递推是稳定的, 第二个递推式当$x > n$ 时对于n 增加的递推是稳定的, 而当$x < n$ 时对于n 减小的递推是稳定的.

计算整数阶贝塞尔函数的任务可由两步来完成, 第一步先计算$J_0(x), J_1(x), Y_0(x), Y_1(x)$ 的数值, 然后利用递推关系求得其它n 的值. 从小x 时的渐近关系我们知道, $Y_n(x)$ 在$x = 0$ 附近有奇异性, 因此, 我们应该计算其正规部分.

$$Y_0(x) - \frac{2}{\pi}J_0(x)\ln(x) \quad \text{and} \quad Y_1(x) - \frac{2}{\pi}\left[J_1(x)\ln(x) - \frac{1}{x}\right] \tag{2.7.227}$$

当$0 \le x \le 8$ 时, $J_0(x), J_1(x), Y_0(x), Y_1(x)$ 可以用分式有理函数来表示, 而当$8 < x < \infty$ 时, 对于$(n = 0, 1)$, 有下面的近似式:

$$\begin{aligned} J_n(x) &= \sqrt{\frac{2}{\pi x}}\left[P_n\left(\frac{8}{x}\right)\cos(X_n) - Q_n\left(\frac{8}{x}\right)\sin(X_n)\right] \\ Y_n(x) &= \sqrt{\frac{2}{\pi x}}\left[P_n\left(\frac{8}{x}\right)\sin(X_n) + Q_n\left(\frac{8}{x}\right)\cos(X_n)\right] \end{aligned} \tag{2.7.228}$$

其中

$$X_n \equiv x - \frac{2n+1}{4}\pi$$

P_0, P_1, Q_0 和Q_1是其宗量的多项式. (这里选择8 作为两种近似表达式的分界线并无特殊的原因, 选7 或9 也可, 当然多项式的系数将会不同). 代替给出表示式, 这里给出计算程序, 读者应该很容易从程序中看出上述各多项式的形式和系数.

```
      Function bessj0(x)
      Real*8 y,p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,
     *    s1,s2,s3,s4,s5,s6
      Data p1,p2,p3,p4,p5 /1.D0,-.1098628627D-2,.2734510407D-4,
     *    -.2073370639D-5,.2093887211D-6/,
     *     q1,q2,q3,q4,q5 /-.1562499995D-1,.1430488765D-3,
     *    -.6911147651D-5,.7621095161D-6,-.934945152D-7/
      Data r1,r2,r3,r4,r5,r6 /57568490574.D0,-13362590354.D0,
     *    651619640.7D0,-11214424.18D0,77392.33017D0,
     *   -184.9052456D0/,
     *    s1,s2,s3,s4,s5,s6 /57568490411.D0,1029532985.D0,
     *    9494680.718D0,59272.64853D0,267.8532712D0,1.D0/
      If(abs(x).LT.8.) Then
        y=x**2
        bessj0=(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))
     *        /(s1+y*(s2+y*(s3+y*(s4+y*(s5+y*s6)))))
      Else
        ax=abs(x)
        z=8./ax
        y=z**2
        xx=ax-.785398164
        bessj0=sqrt(.636619772/ax)*(cos(xx)*(p1+y*(p2+y*(p3+y*(p4+
     *         y*p5))))-z*sin(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5)))))
      Endif
      Return
      End

C
C
      Function bessy0(x)
      Real*8 y,p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,
     *    s1,s2,s3,s4,s5,s6
      Data p1,p2,p3,p4,p5 /1.D0,-.1098628627D-2,.2734510407D-4,
     *    -.2073370639D-5,.2093887211D-6/,
     *     q1,q2,q3,q4,q5 /-.1562499995D-1,.1430488765D-3,
     *    -.6911147651D-5,.7621095161D-6,-.934945152D-7/
      Data r1,r2,r3,r4,r5,r6 /-2957821389.D0,7062834065.D0,
     *    -512359803.6D0,10879881.29D0,-86327.92757D0,
     *     228.4622733D0/,
     *     s1,s2,s3,s4,s5,s6 /40076544269.D0,745249964.8D0,
     *     7189466.438D0,47447.26470D0,226.1030244D0,1.D0/
      If(x.LT.8.) Then
        y=x**2
        bessy0=(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))/(s1+y*(s2+y
     *      *(s3+y*(s4+y*(s5+y*s6)))))+.636619772*bessj0(x)*log(x)
      Else
        z=8./x
        y=z**2
        xx=x-.785398164
        bessy0=sqrt(.636619772/x)*(sin(xx)*(p1+y*(p2+y*(p3+y*(p4+y*
     *      p5))))+z*cos(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5)))))
      Endif
      Return
```

```
      End
C
C
      Function bessj1(x)
      Real*8 y,p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,
     *    s1,s2,s3,s4,s5,s6
      Data r1,r2,r3,r4,r5,r6 /72362614232.D0,-7895059235.D0,
     *    242396853.1D0,-2972611.439D0,15704.48260D0,
     *   -30.16036606D0/,
     *    s1,s2,s3,s4,s5,s6 /144725228442.D0,2300535178.D0,
     *    18583304.74D0,99447.43394D0,376.9991397D0,1.D0/
      Data p1,p2,p3,p4,p5 /1.D0,.183105D-2,-.3516396496D-4,
     *    .2457520174D-5,-.240337019D-6/,
     *     q1,q2,q3,q4,q5 /.04687499995D0,-.2002690873D-3,
     *    .8449199096D-5,-.88228987D-6,.105787412D-6/
      If(abs(x).LT.8.) Then
        y=x**2
        bessj1=x*(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))
            /(s1+y*(s2+y*(s3+y*(s4+y*(s5+y*s6)))))
      Else
        ax=abs(x)
        z=8./ax
        y=z**2
        xx=ax-2.356194491
        bessj1=sqrt(.636619772/ax)*(cos(xx)*(p1+y*(p2+y*(p3+y*(p4+y
     *      *p5))))-z*sin(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5)))))
     *      *sign(1.,x)
      Endif
      Return
      End
C
C
      Function bessy1(x)
      Real*8 y,p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,
     *    s1,s2,s3,s4,s5,s6,s7
      Data p1,p2,p3,p4,p5 /1.D0,.183105D-2,-.3516396496D-4,
     *    .2457520174D-5,-.240337019D-6/,
     *     q1,q2,q3,q4,q5 /.04687499995D0,-.2002690873D-3,
     *    .8449199096D-5,-.88228987D-6,.105787412D-6/
      Data r1,r2,r3,r4,r5,r6 /-.4900604943D13,.1275274390D13,
     *   -.5153438139D11,.7349264551D9,-.4237922726D7,
     *    .8511937935D4/,
     *     s1,s2,s3,s4,s5,s6,s7 /.2499580570D14,.4244419664D12,
     *    .3733650367D10,.2245904002D8,.1020426050D6,
     *    .3549632885D3,1.D0/
      If(x.LT.8.) Then
        y=x**2
        bessy1=x*(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))/(s1+y*(s2+y*
     *      (s3+y*(s4+y*(s5+y*(s6+y*s7))))))+.636619772
     *      *(bessj1(x)*log(x)-1./x)
      Else
        z=8./x
        y=z**2
        xx=x-2.356194491
        bessy1=sqrt(.636619772/x)*(sin(xx)*(p1+y*(p2+y*(p3+y*(p4+y
     *      *p5))))+z*cos(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5)))))
```

```
      Endif
      Return
      End
```

高阶第二类贝塞尔函数的值可以通过递推关系求得, $J_n(x)$的计算比较麻烦, 对于小n 大x, 按照n 增加的方向递推是稳定的, (实际工作中碰到的大都是这种情况), 但对于大n 小x, 则必须按照n 减小的方向递推, 这就必须选则一个较大的n, 一般取为$n+\sqrt{\text{Const}\times n}$, 向下递推至0, 再按照恒等式

$$1 = J_0(x) + 2J_2(x) + 2J_4(x) + 2J_6(x) + \cdots$$

对计算结果归一化. 下面就是这一想法的实现.

```
      Function bessy(n,x)
      If(n.LT.2) Pause'bad argument N in BESSY'
      tox=2./x
      by=bessy1(x)
      bym=bessy0(x)
      Do 11 j=1,n-1
        byp=j*tox*by-bym
        bym=by
        by=byp
   11 Continue
      bessy=by
      Return
      End
C
C
      Function bessj(n,x)
      Parameter(iacc=40,bigno=1.E10,bigni=1.E-10)
      If(n.LT.2) Pause'bad argument N in BESSJ'
      ax=abs(x)
      If(ax.EQ.0.) Then
        bessj=0.
      Elseif(ax.GT.float(n)) Then
        tox=2./ax
        bjm=bessj0(ax)
        bj=bessj1(ax)
        Do 11 j=1,n-1
          bjp=j*tox*bj-bjm
          bjm=bj
          bj=bjp
   11   Continue
        bessj=bj
      Else
        tox=2./ax
        m=2*((n+int(sqrt(float(iacc*n))))/2)
        bessj=0.
        jsum=0
        sum=0.
        bjp=0.
        bj=1.
        Do 12 j=m,1,-1
          bjm=j*tox*bj-bjp
```

```
        bjp=bj
        bj=bjm
        If(abs(bj).GT.bigno) Then
          bj=bj*bigni
          bjp=bjp*bigni
          bessj=bessj*bigni
          sum=sum*bigni
        Endif
        If(jsum.NE.0) sum=sum+bj
        jsum=1-jsum
        If(j.EQ.n) bessj=bjp
12    Continue
      sum=2.*sum-bj
      bessj=bessj/sum
    Endif
    If(x.LT.0..AND.mod(n,2).EQ.1) bessj=-bessj
    Return
    End
```

现在我们讨论虚宗量贝塞尔函数的计算. 虚宗量贝塞尔函数定义为:

$$\begin{aligned} I_n(x) &= (-i)^n J_n(ix) \\ K_n(x) &= \frac{\pi}{2} i^{n+1} [J_n(ix) + iY_n(ix)] \end{aligned} \tag{2.7.229}$$

在小x极限下, 它们具有如下渐近形式:

$$\begin{aligned} I_n(x) &\approx \frac{1}{n!}\left(\frac{x}{2}\right)^n \quad n \geq 0 \\ K_0(x) &\approx -\ln(x) \\ K_n(x) &\approx \frac{(n-1)!}{2}\left(\frac{x}{2}\right)^{-n} \quad n > 0 \end{aligned} \tag{2.7.230}$$

而在大x极限下, 其渐近形式为:

$$\begin{aligned} I_n(x) &\approx \frac{1}{\sqrt{2\pi x}} \exp(x) \\ K_n(x) &\approx \frac{\pi}{\sqrt{2\pi x}} \exp(-x) \end{aligned} \tag{2.7.231}$$

虚宗量贝塞尔函数的递推关系与贝塞尔函数的略有不同, 由下式给出:

$$\begin{aligned} I_n'(x) &= \frac{nI_n(x)}{x} + I_{n+1}(x) \\ I_{n+1}(x) &= -\left(\frac{2n}{x}\right) I_n(x) + I_{n-1}(x) \\ K_{n+1}(x) &= +\left(\frac{2n}{x}\right) K_n(x) + K_{n-1}(x) \end{aligned} \tag{2.7.232}$$

归一化关系为

$$1 = I_0(x) - 2I_2(x) + 2I_4(x) - 2I_6(x) + \cdots$$

考虑到上述改变, 我们给出计算程序如下:

```
      Function bessi0(x)
      Real*8 y,p1,p2,p3,p4,p5,p6,p7,
     *    q1,q2,q3,q4,q5,q6,q7,q8,q9
      Data p1,p2,p3,p4,p5,p6,p7 /1.0D0,3.5156229D0,3.0899424D0,
     *    1.2067492D0,0.2659732D0,0.360768D-1,0.45813D-2/
      Data q1,q2,q3,q4,q5,q6,q7,q8,q9 /0.39894228D0,0.1328592D-1,
     *    0.225319D-2,-0.157565D-2,0.916281D-2,-0.2057706D-1,
     *    0.2635537D-1,-0.1647633D-1,0.392377D-2/
      If(abs(x).LT.3.75) Then
        y=(x/3.75)**2
        bessi0=p1+y*(p2+y*(p3+y*(p4+y*(p5+y*(p6+y*p7)))))
      Else
        ax=abs(x)
        y=3.75/ax
        bessi0=(exp(ax)/sqrt(ax))*(q1+y*(q2+y*(q3+y*(q4
     *   +y*(q5+y*(q6+y*(q7+y*(q8+y*q9))))))))
      Endif
      Return
      End
C
C
      Function bessi1(x)
      Real*8 y,p1,p2,p3,p4,p5,p6,p7,
     *    q1,q2,q3,q4,q5,q6,q7,q8,q9
      Data p1,p2,p3,p4,p5,p6,p7 /0.5D0,0.87890594D0,0.51498869D0,
     *    0.15084934D0,0.2658733D-1,0.301532D-2,0.32411D-3/
      Data q1,q2,q3,q4,q5,q6,q7,q8,q9 /0.39894228D0,-0.3988024D-1,
     *    -0.362018D-2,0.163801D-2,-0.1031555D-1,0.2282967D-1,
     *    -0.2895312D-1,0.1787654D-1,-0.420059D-2/
      If(abs(x).LT.3.75) Then
        y=(x/3.75)**2
        bessi1=x*(p1+y*(p2+y*(p3+y*(p4+y*(p5+y*(p6+y*p7))))))
      Else
        ax=abs(x)
        y=3.75/ax
        bessi1=(exp(ax)/sqrt(ax))*(q1+y*(q2+y*(q3+y*(q4+
     *      y*(q5+y*(q6+y*(q7+y*(q8+y*q9))))))))
        If(x.LT.0.) bessi1=-bessi1
      Endif
      Return
      End
C
C
      Function bessk0(x)
      Real*8 y,p1,p2,p3,p4,p5,p6,p7,
     *    q1,q2,q3,q4,q5,q6,q7
      Data p1,p2,p3,p4,p5,p6,p7 /-0.57721566D0,0.42278420D0,
     *    0.23069756D0,0.3488590D-1,0.262698D-2,0.10750D-3,0.74D-5/
      Data q1,q2,q3,q4,q5,q6,q7 /1.25331414D0,-0.7832358D-1,
     *    0.2189568D-1,-0.1062446D-1,0.587872D-2,-0.251540D-2,
     *    0.53208D-3/
      If(x.LE.2.0) Then
        y=x*x/4.0
        bessk0=(-log(x/2.0)*bessi0(x))+(p1+y*(p2+y*(p3+
     *   y*(p4+y*(p5+y*(p6+y*p7))))))
      Else
```

```
        y=(2.0/x)
        bessk0=(exp(-x)/sqrt(x))*(q1+y*(q2+y*(q3+
     *        y*(q4+y*(q5+y*(q6+y*q7))))))
       Endif
       Return
       End
C
C
       Function bessk1(x)
       Real*8 y,p1,p2,p3,p4,p5,p6,p7,
     *    q1,q2,q3,q4,q5,q6,q7
       Data p1,p2,p3,p4,p5,p6,p7 /1.0D0,0.15443144D0,-0.67278579D0,
     *    -0.18156897D0,-0.1919402D-1,-0.110404D-2,-0.4686D-4/
       Data q1,q2,q3,q4,q5,q6,q7 /1.25331414D0,0.23498619D0,
     *    -0.3655620D-1,0.1504268D-1,-0.780353D-2,0.325614D-2,
     *    -0.68245D-3/
       If(x.LE.2.0) Then
        y=x*x/4.0
        bessk1=(log(x/2.0)*bessi1(x))+(1.0/x)*(p1+y*(p2+
     *      y*(p3+y*(p4+y*(p5+y*(p6+y*p7))))))
       Else
        y=2.0/x
        bessk1=(exp(-x)/sqrt(x))*(q1+y*(q2+y*(q3+
     *      y*(q4+y*(q5+y*(q6+y*q7))))))
       Endif
       Return
       End

       Function bessk(n,x)
       If(n.LT.2) Pause'bad argument N in BESSK'
       tox=2.0/x
       bkm=bessk0(x)
       bk=bessk1(x)
       Do 11 j=1,n-1
        bkp=bkm+j*tox*bk
        bkm=bk
        bk=bkp
   11  Continue
       bessk=bk
       Return
       End
C
C
       Function bessi(n,x)
       Parameter(iacc=40,bigno=1.0E10,bigni=1.0E-10)
       If(n.LT.2) Pause'bad argument N in BESSI'
       If(x.EQ.0.) Then
        bessi=0.
       Else
        tox=2.0/abs(x)
        bip=0.0
        bi=1.0
        bessi=0.
        m=2*((n+int(sqrt(float(iacc*n)))))
        Do 11 j=m,1,-1
         bim=bip+float(j)*tox*bi
         bip=bi
```

```
        bi=bim
        If(abs(bi).GT.bigno) Then
          bessi=bessi*bigni
          bi=bi*bigni
          bip=bip*bigni
        Endif
        If(j.EQ.n) bessi=bip
11    Continue
      bessi=bessi*bessi0(x)/bi
      If(x.LT.0..AND.mod(n,2).EQ.1) bessi=-bessi
    Endif
    Return
    End
```

练习

运行上面的程序, 并与你所能得到的贝塞尔函数的数据表比较

例题 均匀圆柱, 半径为R, 高为H, 柱侧有均匀分布的恒定热流进入, 其强度为q_0, 圆柱上下底面保持为恒定的u_0 度, 求解柱内稳定温度分布.

这一问题的答案是*(*见梁昆淼, 数学物理方法, *P376)*

$$u-u_0=\frac{4Hq_0}{k\pi^2}\sum_{l=0}^{\infty}\frac{1}{(2l+1)^2}\frac{I_0\left(\frac{(2l+1)\pi}{H}\rho\right)}{I_0'\left(\frac{(2l+1)\pi R}{H}\right)}\sin\frac{(2l+1)\pi z}{H}$$

利用$I_0'(x)=I_1(x)$, 由上面给出的贝塞尔函数的计算程序, 可以直接用上式求得$u-u_0$, 由于I_0 和I_1 在其宗量较大时指数上升, 为了避免上溢, 实际计算时对前面所给程序作了很小的改动, 即把指数上升部分提了出来. 图*(2.6)* 给出了$R=1,H=2$ 时$u-u_0$ 与z 的关系的计算结果. 从下至上, 曲线依次对应于$\rho=0,0.2,0.4,0.6,0.8,1.0$, 从图中可见, 在柱的轴线方向, 中间温度最高, 沿柱的径向, 则表面温度最高, 这些特点, 在解析式子中是很难看出来的.

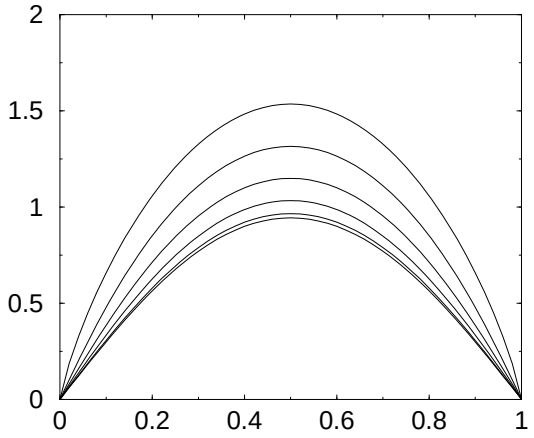


Figure 2.6:

2.7.4 球谐函数的计算

球谐函数在物理学的各个子领域中都经常遇到, 它是Laplace方程在球坐标中分离变量的自然结果. 球谐函数通常通过连带勒让德多项式来定义:

$$Y_{lm}(\theta,\phi)=\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)e^{im\phi} \tag{2.7.233}$$

利用关系式

$$Y_{l,-m}(\theta,\phi)=(-1)^m Y_{lm}^*(\theta,\phi) \tag{2.7.234}$$

我们总可以把一个任意球谐函数与$m\geq 0$的连带勒让德多项式联系起来. 令$x\equiv\cos\theta$, 连带勒让德多项式定义为:

$$P_l^m(x)=(-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_l(x) \tag{2.7.235}$$

$P_l(x)$ 为l 阶勒让德多项式. 连带勒让德多项式有关系式

$$P_l^{-m}(\cos\theta)=(-1)^m\frac{(l-m)!}{(l+m)!}P_l^m(\cos\theta)$$

因此我们只需要计算$m>0$ 的情形.

连带勒让德多项式满足如下的递推关系:

$$(l-m)P_l^m(x)=x(2l-1)P_{l-1}^m-(l+m-1)P_{l-2}^m \tag{2.7.236}$$

且

$$\begin{aligned} P_m^m(x) &= (-1)^m(2m-1)!!(1-x^2)^{m/2} \\ P_{m+1}^m(x) &= x(2m+1)P_m^m(x) \end{aligned} \tag{2.7.237}$$

由此, 我们得到计算连带勒让德多项式的程序如下:

```
      Function plgndr(l,m,x)
      If(m.LT.0.OR.m.GT.l.OR.abs(x).GT.1.) Pause'bad arguments'
      pmm=1.
      If(m.GT.0) Then
        somx2=sqrt((1.-x)*(1.+x))
        fact=1.
        Do 11 i=1,m
          pmm=-pmm*fact*somx2
          fact=fact+2.
11      Continue
      Endif
      If(l.EQ.m) Then
        plgndr=pmm
      Else
        pmmp1=x*(2*m+1)*pmm
        If(l.EQ.m+1) Then
          plgndr=pmmp1
        Else
          Do 12 ll=m+2,l
            pll=(x*(2*ll-1)*pmmp1-(ll+m-1)*pmm)/(ll-m)
            pmm=pmmp1
            pmmp1=pll
12        Continue
          plgndr=pll
        Endif
      Endif
      Return
      End
```

当$m=0$ 时, 程序给出勒让德多项式的值. 球谐函数则可通过连带勒让德多项式乘以一个因子得到.

例题: 设半径为a 的球面上的势给定为*(*取球坐标, 坐标原点取到球心*)*

$$V(\theta) = \begin{cases} +V & 0 \le \theta < \frac{\pi}{2} \\ -V & \frac{\pi}{2} < \theta \le \pi \end{cases}$$

其球外的电势可解出为*(*见*D. J.* 杰克逊, 经典电动力学, 朱培豫译, *P101)*

$$\Phi(r,\theta) = \frac{V}{\sqrt{\pi}} \sum_{j=1}^{\infty} (-1)^{j-1} \frac{(2j-\frac{1}{2})\Gamma\left(j-\frac{1}{2}\right)}{j!} \left(\frac{a}{r}\right)^{2j} P_{2j-1}(\cos\theta)$$

利用本节提供的计算Γ函数和勒让德多项式的子程序, 直接对上式求和, 就可得到电势的图像, 图*(2.7)*是对五个$r, (r = 1.1a, 1.6a, 2.1a, 2.6a, 3.1a)$ 的数值所做的计算结果.

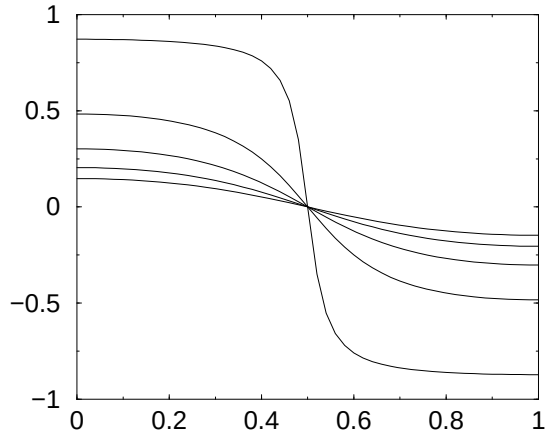


Figure 2.7:

2.7.5 椭圆积分的计算

我们知道, 形如

$$\int dx \frac{A(x)+B(x)\sqrt{S(x)}}{C(x)+D(x)\sqrt{S(x)}} \tag{2.7.238}$$

的积分总可以表示为椭圆积分. 这里$A, B, C,$ 和D 为任意多项式而S 是一个三次或四次多项式. 为了计算椭圆积分的数值, 我们定义广义第二类不完全椭圆积分如下:

$$el2(y, k_c, a, b) \equiv \int_0^y \frac{(a+bx^2)dx}{(1+x^2)\sqrt{(1+x^2)(1+k_c^2x^2)}} \tag{2.7.239}$$

这里$y \geq 0$, a, b, k_c可以是任何实数. 做变量替换

$$x = \tan\phi \quad k^2 = 1 - k_c^2 \tag{2.7.240}$$

我们得到广义第二类不完全椭圆积分的等价表示:

$$\begin{aligned} el2(y, k_c, a, b) &= \int_0^{\tan^{-1} y} \frac{(a+b\tan^2\phi)d\phi}{(1+\tan^2\phi)\sqrt{(1+\tan^2\phi)(1+k_c^2\tan^2\phi)}} \\ &= \int_0^{\tan^{-1} y} \frac{a+(b-a)\sin^2\phi}{\sqrt{1-k^2\sin^2\phi}} d\phi \end{aligned} \tag{2.7.241}$$

另一个有用的函数是广义完全椭圆积分, 定义为:

$$\begin{aligned} cel(k_c, p, a, b) &\equiv \int_0^\infty \frac{(a+bx^2)dx}{(1+px^2)\sqrt{(1+x^2)(1+k_c^2x^2)}} \\ &= \int_0^{\pi/2} \frac{a\cos^2\phi + b\sin^2\phi}{(\cos^2\phi + p\sin^2\phi)\sqrt{\cos^2\phi + k_c^2\sin^2\phi}} d\phi \end{aligned} \tag{2.7.242}$$

利用上述两个函数, 则第一类勒让德椭圆积分可表示为:

$$F(\phi,k)=\int_0^{\phi}\frac{d\phi}{\sqrt{1-k^2\sin^2\phi}}=\int_0^{x}\frac{dx}{\sqrt{(1+x^2)(1+k_c^2x^2)}}=el2(x,k_c,1,1) \quad (2.7.243)$$

这里

$$x=\tan\phi,\quad k_c^2=1-k^2$$

第一类完全椭圆积分可表示为:

$$K(k)\equiv F(\frac{\pi}{2},k)=cel(k_c,1,1,1) \quad (2.7.244)$$

第二类勒让德椭圆积分可表示为:

$$E(\phi,k)=\int_0^{\phi}\sqrt{1-k^2\sin^2\phi}d\phi=\int_0^{x}\frac{\sqrt{1+k_c^2x^2}}{(1+x^2)\sqrt{(1+x^2)}}dx=el2(x,k_c,1,k_c^2) \quad (2.7.245)$$

第二类完全椭圆积分可表示为:

$$E(k)\equiv E(\frac{\pi}{2},k)=cel(k_c,1,1,k_c^2) \quad (2.7.246)$$

这里不给出算法, 讲义所附计算$el2$ 和cel的程序可作为一个"黑匣子"来使用.

例题: 考虑一个单摆, 由一质量为m 的重锤和一长为l 的轻绳构成, 平衡时重锤竖直向下, 取摆绳偏离平衡位置的夹角为θ, 则摆的运动方程为

$$\ddot{\theta}=\frac{g}{l}\sin\theta$$

其中g 为重力加速度, 当$\theta<<1$ 时, $\sin\theta\approx\theta$, 上面方程可容易解出, 由此可求得单摆的周期为

$$T_0=2\pi\sqrt{\frac{l}{g}}$$

为了求得一般结果, 把原方程改写为

$$T_0^2\ddot{\theta}=(2\pi)^2\sin\theta$$

上式两边乘以$\dot{\theta}$ 并积分得:

$$\frac{d\theta}{dt}=\pm\frac{2\pi}{T_0}\sqrt{2(\cos\theta-\cos\alpha)}$$

式中α 为一积分常数, 代表单摆的最大摆角, $+$ 号表示θ 增加方向, $-$ 号表示θ 减小方向. 上式可进一步化为

$$dt=\pm\frac{T_0}{4\pi}\frac{d\theta}{\sqrt{\sin^2\frac{\alpha}{2}-\sin^2\frac{\theta}{2}}}$$

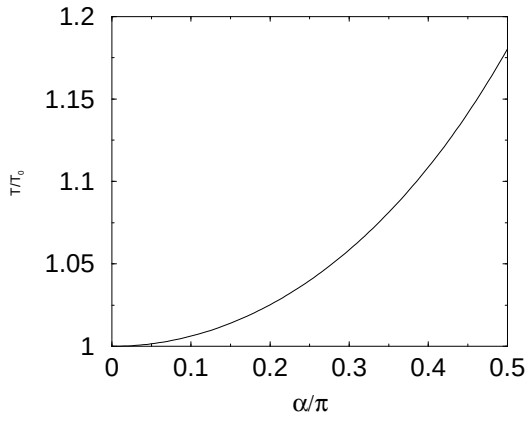


Figure 2.8: 单摆的周期与最大偏角之间的关系

若设$t = 0$ 时, $\theta = 0$, 考虑$0 \le t \le T/4$, $0 \le \theta \le \alpha$ 之间的运动, 作代换$\sin\frac{\theta}{2} = \sin\frac{\alpha}{2}\sin\phi$, 则

$$t = T_0\frac{1}{2\pi}\int_0^{\phi}\frac{d\phi}{\sqrt{1-\sin^2\frac{\alpha}{2}}} = T_0\frac{1}{2\pi}F(\phi,k) = T_0\frac{1}{2\pi}el2(x,k_c,1,1)$$

其中$k = \sin\frac{\alpha}{2}$, $x = \mathrm{tg}\phi$, $k_c = \sqrt{1-k^2}$, 当$\theta = \alpha$ 时, $\phi = \frac{\pi}{2}$, 由此得摆的严格周期为

$$T = 4T_0\frac{1}{2\pi}F(\frac{2\pi}{2},k) = T_0\frac{2}{\pi}cel(k_c,1,1,1)$$

图(2.8)是按照上式求出的$T \sim \alpha$ 关系.

2.8 附录: 高斯积分的节点和权重

这里给出若干常用的高斯积分的节点和权重表，更多的数表可在M. Abramowitz & A. Stegun 编写的*Handbook of Mathematical Functions* 中找到.

Table 2.2: 高斯—勒让德积分的节点和权重

$\pm x_i$	w_i
n=2	
0.57735 02691 89626	1.00000 00000 00000
n=3	
0.00000 00000 00000	0.88888 88888 88889
0.77459 66692 41483	0.55555 55555 55556
n=4	
0.33998 10435 84856	0.65214 51548 62546
0.86113 63115 94055	0.34785 48451 37454
n=5	
0.00000 00000 00000	0.56888 88888 88889
0.53846 93101 05683	0.47862 86704 99366
0.90617 98459 38664	0.23692 68850 56189
n=6	
0.23861 91860 83197	0.46791 39345 72691
0.66120 93864 66265	0.36076 15730 48139
0.93246 95142 03152	0.17132 44923 79170
n=7	
0.00000 00000 00000	0.41795 91836 73469
0.40584 51513 77397	0.38183 00505 05119
0.74153 11855 99394	0.27970 52914 89277
0.94910 79123 42759	0.12948 49661 68870
n=8	
0.18343 46424 95650	0.36268 37833 78362
0.52553 24099 16329	0.31370 66458 77887
0.79666 64774 13627	0.22238 10344 53374
0.96028 98564 97536	0.10122 85362 90376

Table 2.3: 等权重切贝雪夫积分的节点

n	$\pm x_i$	n	$\pm x_i$	n	$\pm x_i$
2	0.57735 02692	5	0.83249 74870	7	0.88386 17008
			0.37454 14096		0.52965 67753
			0.00000 00000		0.32391 18105
					0.00000 00000
3	0.70710 67812				
	0.00000 00000			9	0.91158 93077
					0.60101 86554
		6	0.86624 68181		0.52876 17831
4	0.79465 44723		0.42251 86538		0.16790 61842
	0.18759 24741		0.26663 54015		0.00000 00000

Table 2.4: 高斯—拉盖尔积分的节点和权重, 表示形式为$(E)B = B \times 10^E$

x_i	w_i	$w_i e^{x_i}$
n=2		
0.58578 64376 27	(-1)8.53553 390593	1.53332 603312
3.41421 35623 73	(-1)1.46446 609407	4.45095 733505
n=5		
0.26356 03197 18	(-1)5.21755 610583	0.67909 40422 08
1.41340 30591 07	(-1)3.98666 811083	2.04810 243845
4.53662 02969 21	(-2)7.59424 496817	2.76944 324237
7.08581 00058 59	(-3)3.61175 867992	4.31565 690092
12.64080 08442 76	(-5)2.33699 723858	7.21918 635435
n=6		
0.22284 66041 79	(-1)4.58964 673950	0.57353 55074 23
1.18893 21016 73	(-1)4.17000 830772	1.36925 259071
2.99273 63260 59	(-1)1.13373 382074	2.26068 459338
5.77514 35691 05	(-2)1.03991 974531	3.35052 458236
9.83746 74183 83	(-4)2.61017 202815	4.88682 680021
15.98287 39806 02	(-7)8.98547 906430	7.84901 594560
n=7		
0.19304 36765 60	(-1)4.09318 951701	0.49647 75975 40
1.02666 48953 39	(-1)4.21831 277862	1.17764 306086
2.56787 67449 51	(-1)1.47126 348658	1.91824 978166
4.90035 30845 26	(-2)2.06335 144687	2.77184 863623
8.18215 34445 63	(-3)1.07401 014328	3.84124 912249
12.73418 02917 98	(-5)1.58654 643486	5.36067 820792
19.39572 78622 63	(-8)3.17031 547900	8.40543 248683
n=8		
0.17027 96323 05	(-1)3.69188 589342	0.43772 34104 93
0.90370 17767 99	(-1)4.18786 780814	1.03386 934767
2.25108 66298 66	(-1)1.75794 986637	1.66970 976566
4.26670 01702 88	(-2)3.33434 922612	2.37692 470176
7.04590 54023 93	(-3)2.79453 623523	3.20854 091335
10.75851 60101 81	(-5)9.07650 877336	4.26857 551083
15.74067 86412 78	(-7)8.48574 671627	5.81808 336867
22.86313 17368 89	(-9)1.04800 117487	8.90622 621529

Table 2.5: 高斯—厄米积分的节点和权重, 表示形式为$(E)B = B \times 10^E$

$\pm x_i$	w_i	$w_i e^{x_i^2}$
n=2		
0.70710 67811 86548	(-1)8.86226 92545 28	1.46114 11826 611
n=5		
0.00000 00000 00000	(-1)9.45308 72048 29	0.94530 87204 829
0.95857 24646 13819	(-1)3.93619 32315 22	0.98658 09967 514
2.02018 28704 56086	(-2)1.99532 42059 05	1.18148 86255 360
n=6		
0.43607 74119 27617	(-1)7.24629 59522 44	0.87640 13344 362
1.33584 90740 13697	(-1)1.57067 32032 29	0.93558 05576 312
2.35060 49736 74492	(-3)4.53000 99055 09	1.13690 83326 745
n=7		
0.00000 00000 00000	(-1)8.10264 61755 68	0.81026 46175 568
0.81628 78828 58965	(-1)4.25607 25261 01	0.82868 73032 836
1.67355 16287 67471	(-2)5.45155 82819 13	0.89718 46002 252
2.65196 13568 35233	(-4)9.71781 24509 95	1.10133 07296 103
n=8		
0.38118 69902 07322	(-1)6.61147 01255 82	0.76454 41286 517
1.15719 37124 46780	(-1)2.07802 32581 49	0.79289 00483 864
1.98165 67566 95843	(-2)1.70779 83007 41	0.86675 26065 634
2.93063 74202 57244	(-4)1.99604 07221 14	1.07193 01442 480

Chapter 3

数值线性代数

在这一章里, 我们将讨论物理学中非常重要的一类计算问题, 即关于线性代数的计算问题. 线性代数计算也是计算数学中研究的最透彻的问题之一, 目前已经有一些十分完整的程序包运行在从微机到大型机的各种不同档次的计算机上, 其中最著名的是Linpack和Eispack, 第一个是求解线性方程组的通用软件包, 第二个是求解各种矩阵本征值和本征向量的通用软件包.* 目前, 这两个包已经合并为LAPACK软件包, 与基本线性代数包BLAS联合, 可以解决几乎所有的线性代数问题. 这些软件包可在http://www.netlib.org 免费获得, 在校园里十分流行的工具软件Matlab为上述软件包做了很好的界面, 可以帮助方便使用.

但另一方面, 大型软件包由于包含的程序多, 调用路径复杂, 对于一些较为简单的问题, 使用起来颇有牛刀杀鸡之感. 因此, 在这一章里, 我们将介绍几种常见的线性代数问题的计算方法, 对于在研究中碰到的较为复杂的问题, 强烈建议使用前面所提软件包中的标准程序.

3.1 主元消去法解线性代数方程组

一个n 元线性代数方程组记为

$$\mathbf{A}\cdot\mathbf{X}=\mathbf{B} \tag{3.1.1}$$

这里,

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

*国内最早对这两个软件包进行系统介绍的是郭富印, 冯国环, 石中岳和朱耀祖所编《FORTRAN 算法汇编, 第三分册》, 并对软件包做了一些扩充, 但所附程序有个别印刷错误.

为一$n \times n$矩阵,

$$\mathbf{X} = [x_1, x_2, \cdots, x_n]^T, \quad \mathbf{B} = [b_1, b_2, \cdots, b_n]^T$$

为列向量. 显然, 方程(3.1.1)的解可形式地写为

$$\mathbf{X} = \mathbf{A^{-1}B}$$

但这一形式对于实际计算没有什么用处. 一个实用的算法是消去法, 其计算过程是相当简单的, 首先, 用矩阵**A** 的第一行分别乘以某一适当的数与其它各行相加, 使得矩阵**A** 的第一列除a_{11} 外全为0, 同时, 对列矩阵**B** 施行同样的运算, 由线性代数知道, 这样的运算相当于用第一个方程分别乘以某一数值与其它方程相加, 而这样的运算并不改变原方程组的解. 用**A** 的第二行分别乘以某一适当数值与其它各行相加,同时对**B** 施行同样的运算, 便可以把**A** 的第二列除a_{22} 外的元素化为0, 这一过程可以一直(?)进行下去, 最后得到一个对角矩阵的方程, 其解可直接写出.

然而, 上述过程并不总是能够成功, 事实上, 直接用这一方法计算的话, 大部分问题都无法求解, 一个极端的例子是, 如果$a_{11} = 0$, 则显然第一步就完不成. 解决这一问题的方法自然是十分简单的, 只要交换第一个方程和另一个x_1 的系数不为0 的方程就行了, 实际上, 应该选择x_1 的系数的绝对值最大的方程与之交换, 这样可使舍入误差最小. 既然如此, 对于$a_{11} \neq 0$ 的情形, 我们也可以作这样一个交换, 把x_1的系数的绝对值最大的方程放在第一个, 这一过程称为选列主元. 当然, 我们对于其后的运算也要进行选主元计算. 完成选列主元的过程等价于交换矩阵**A** 的二行, 这一过程并不改变原方程的解, 所以无需保留任何交换的信息. 也许大家可能已经想到, 如果在每一个方程内, 在计算的每一步把系数最大的那一项放在第一个, 可能数值性态会更好. 事实确是如此, 实现这一过程称为选行主元, 可以通过交换矩阵的两列来实现. 由于选行主元改变了原方程的结构, 所以在计算的每一步都要保留行主元的信息, 在最后恢复到原方程的解. 这一过程叫做全主元高斯消去法. 我们立刻可以看到, 这一过程对于求解一组方程和求解**A** 相同而方程右边不同的数组方程一样容易, 更进一步, **A** 的逆矩阵也可同时求出(如何求?).

3.2 LU分解法

前节讲述的方法基本可以满足一般问题的需要, 但也有一个严重的缺点, 这就是必须事先知道方程的右边**B**, 有时候, 我们需要求解这样一类方程组, 其系数矩阵是给定的, 但**B**随问题而变且事先并不知道. 对这样一类问题, LU分解法提供了最好的解决办法.

假定我们可以把矩阵**A**分解为

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{A} \tag{3.2.2}$$

这里**L**为一下三角矩阵, **U**是一个上三角矩阵, 其形式为

$$\mathbf{L}=\begin{bmatrix}\alpha_{11} & 0 & \cdots & 0\\ \alpha_{21} & \alpha_{22} & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots\\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn}\end{bmatrix} \tag{3.2.3}$$

$$\mathbf{U}=\begin{bmatrix}\beta_{11} & \beta_{12} & \cdots & \beta_{1n}\\ 0 & \beta_{22} & \cdots & \beta_{2n}\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & \beta_{nn}\end{bmatrix} \tag{3.2.4}$$

利用(3.2.2), 方程(3.1.1)可写为

$$\mathbf{A}\cdot\mathbf{X}=\mathbf{L}\cdot\mathbf{U}\cdot\mathbf{X}=\mathbf{B} \tag{3.2.5}$$

定义一个矢量**Y** 为下面方程的解

$$\mathbf{L}\cdot\mathbf{Y}=\mathbf{B} \tag{3.2.6}$$

则

$$\mathbf{U}\cdot\mathbf{X}=\mathbf{Y} \tag{3.2.7}$$

的解即为原方程(3.1.1)的解. 方程(3.2.6) 和(3.2.7)的求解是平庸的, (3.2.6)可用所谓向前代入法求解, 即

$$\begin{aligned} y_1 &= \frac{b_1}{\alpha_{11}}\\ y_i &= \frac{1}{\alpha_{ii}}\left[b_i-\sum_{j=1}^{i-1}\alpha_{ij}y_i\right] \quad i=2,3,\cdots,n \end{aligned} \tag{3.2.8}$$

而(3.2.7)可用向后代入法求解,

$$\begin{aligned} x_n &= \frac{y_n}{\beta_{nn}}\\ x_i &= \frac{1}{\beta_{ii}}\left[y_i-\sum_{j=i+1}^{n}\beta_{ij}x_j\right] \quad i=n-1,n-2,\cdots,2 \end{aligned} \tag{3.2.9}$$

上述计算是直接了当的, 因此, 只要有了矩阵**A** 的LU分解, 对于任一**B**, 我们只需按(3.2.8) 和(3.2.9)作简单计算便可求得其解.

现在的问题是要找出矩阵**A** 的LU分解, 为此, 我们写出方程(3.2.2)的分量形式

$$\begin{aligned} \alpha_{i1}\beta_{1j}+\alpha_{i2}\beta_{2j}+\cdots+\alpha_{ii}\beta_{ij} &= a_{ij} \text{ 当 } i<j\\ \alpha_{i1}\beta_{1j}+\alpha_{i2}\beta_{2j}+\cdots+\alpha_{ii}\beta_{jj} &= a_{ij} \text{ 当 } i=j\\ \alpha_{i1}\beta_{1j}+\alpha_{i2}\beta_{2j}+\cdots+\alpha_{ij}\beta_{jj} &= a_{ij} \text{ 当 } i>j \end{aligned} \tag{3.2.10}$$

上式共有n^2 个方程, 而有n^2+n 个未知数, 由于未知数的数目大于方程的数目, 我们可以任意指定n 个条件, 然后求解这组方程. 如果指定

$$\alpha_{ii}=1 \quad i=1,2,\cdots,n \tag{3.2.11}$$

则方程(3.2.10)可以十分方便的求出. 其算法通常称为Crout算法, 可表述为:

对每一个$j=1,2,3,\cdots,n$, 作如下两步运算:

1. 对$i=1,2,\cdots,j$, 利用(3.2.10)的一, 二两式及(3.2.11)求解β_{ij}, 即

$$\begin{aligned}\beta_{1j} &= a_{1j}\\ \beta_{ij} &= a_{ij}-\sum_{k=1}^{i-1}\alpha_{ik}\beta_{kj} \quad i=2,3,\cdots,j\end{aligned} \tag{3.2.12}$$

2. 对$i=j+1,j+2,\cdots,n$, 利用(3.2.10)的第三式求解α_{ij}, 即

$$\begin{aligned}\alpha_{i1} &= \frac{1}{\beta_{11}}a_{i1} \quad j=1\\ \alpha_{ij} &= \frac{1}{\beta_{jj}}\left[a_{ij}-\sum_{k=1}^{j-1}\alpha_{ik}\beta_{kj}\right] \quad j=2,3,\cdots,n\end{aligned} \tag{3.2.13}$$

请读者用铅笔和纸按照上面的算法做几步, 以验证:

- 方程(3.2.12) 和(3.2.13)右边的诸α,β在需要时已经算出.
- 每一个a_{ij}只使用一次且用过后不再需要, 也就是说, α_{ij}或β_{ij} 可以取代a_{ij} 以节约存储空间.

当完成LU分解之后, 矩阵$\mathbf{A}$ 变为如下形式:

$$\begin{bmatrix}\beta_{11} & \beta_{12} & \cdots & \beta_{1n}\\ \alpha_{21} & \beta_{22} & \cdots & \beta_{2n}\\ \vdots & \vdots & \cdots & \vdots\\ \alpha_{n1} & \alpha_{n2} & \cdots & \beta_{nn}\end{bmatrix} \tag{3.2.14}$$

在描述Crout算法时, 我们有意回避了选主元的问题. 这并不意味着主元的选取不重要, 而是为了表述清楚. 因为这是本讲义中很少的几个给出详细描述的算法之一. 在Crout算法中, 只选列主元, 因为这样做并不会改变原方程组, 而仅仅是交换方程而已. 因此, 选列主元后, 我们得到的实际上是$\mathbf{A}$ 的进行了某些行交换之后的矩阵的LU分解.

练习

请证明, 在LU分解下, 矩阵$\mathbf{A}$的行列式由下式给出

$$det(\mathbf{A})=\Pi_{j=1}^{n}\beta_{jj}$$

如何利用LU分解计算矩阵$\mathbf{A}$的逆矩阵? 请给出关键程序段

3.3 三对角方程组的求解

前面两节的方法可以解决任何线性代数方程组的求解问题, 但在实际工作中常常遇到一些特殊形式的方程组, 这些方程组当然可以利用前面的一般方法求解, 但如果我们充分利用它们的特殊性, 则可以更快地得到结果, 或者在同样的计算条件下求解更大的问题.

三对角方程组是最常见的特殊形式的方程组之一, 它在后面将要讲到的微分方程求解问题, 在固体物理的紧束缚近似方程中均会出现, 因此我们在此给出其求解方法. 三对角方程组是指由下式给出的方程组

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & & & \\ a_2 & b_2 & c_2 & \cdots & & & \\ & & & \cdots & & & \\ & & & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & \cdots & 0 & a_n & b_n \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_{n-1} \\ r_n \end{bmatrix} \tag{3.3.15}$$

对上面方程利用高斯消去法求解，就可以得到所谓的“追赶法”, 描述如下,

1. 令$u_1 = r_1/b_1$, $\gamma_1 = 0$, 对$j = 2, 3, \cdots, n$, 作运算

$$\begin{aligned} \gamma_j &= \frac{c_{j-1}}{b_{j-1} - a_{j-1}\gamma_{j-1}} \\ u_j &\leftarrow \frac{r_j - a_j u_{j-1}}{b_j - a_j \gamma_j} \end{aligned} \tag{3.3.16}$$

2. 对$j = n-1, n-2, \cdots, 1$, 作运算

$$u_j \leftarrow u_j - \gamma_{j+1} u_{j+1} \tag{3.3.17}$$

练习

试证明上面的算法确实给出*(3.3.15)*的解

在这一算法中, 我们没有施行选主元的运算, 因为主元的选取将破坏原方程的结构, 而原方程的特殊结构正是我们构造上述算法的基础, 在大部分实际问题中, 三对角方程组的对角线元素本身已经是主元. 因此, 这一缺陷对大部分应用不会有影响, 如果在某一问题的求解中上述方法失败, 则建议改用LU分解方法, 全主元消去法可作为最后的选择.

3.4 实对称矩阵的本征值和本征向量

计算矩阵本征值问题的最佳办法是, 首先, 把矩阵约化为一简单形式, 然后用能够保持这一简单形式的叠代方法计算. 常用的约化方法有Givens 方法和Householder 方法. 常用的叠代方法有QR 方法和QL 方法等. 这一节我们介绍Householder 方法和QL方法. 本节给出的程序均取自Eispack, 这是一个经过多年考验的软件包.

3.4.1 Householder 方法

Householder 方法通过$n-2$ 步正交变换把一$n \times n$ 实对称矩阵$\mathbf{A}$约化为三对角形式, 即除了主对角线和次对角线外的元素均为0. 这一变换的最基本的部分是Householder 矩阵$\mathbf{P}$, 它具有下面的形式

$$\mathbf{P} = \mathbf{1} - \mathbf{2w} \cdot \mathbf{w}^T \tag{3.4.18}$$

这里$\mathbf{w}$ 为一实矢量且满足$|\mathbf{w}|^2 \equiv \mathbf{w}^T \cdot \mathbf{w} = 1$. 矩阵$\mathbf{P}$ 是正交的, 这可以简单证明如下

$$\begin{aligned} \mathbf{P^2} &= (\mathbf{1} - \mathbf{2w} \cdot \mathbf{w}^T) \cdot (\mathbf{1} - \mathbf{2w} \cdot \mathbf{w}^T) \\ &= \mathbf{1} - \mathbf{4w} \cdot \mathbf{w}^T + \mathbf{4w} \cdot (\mathbf{w}^T \cdot \mathbf{w}) \cdot \mathbf{w}^T \\ &= 1 \end{aligned}$$

因此$\mathbf{P} = \mathbf{P^{-1}}$. 由(3.4.18) 可知$\mathbf{P}^T = \mathbf{P}$, 所以$\mathbf{P^{-1}} = \mathbf{P}^T$ 即$\mathbf{P}$ 为一正交矩阵.

把$\mathbf{P}$ 写为

$$\mathbf{P} = 1 - \frac{\mathbf{u} \cdot \mathbf{u}^T}{H} \tag{3.4.19}$$

此处$H \equiv \frac{1}{2}|\mathbf{u}|^2$, $\mathbf{u}$ 为一任意非零实矢量. 如果$\mathbf{x}$ 是由A 的第一列构成的矢量, 选择

$$\mathbf{u} = \mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1 \tag{3.4.20}$$

这里$\mathbf{e}_1 = [1, 0, 0, \cdots, 0]^T$ 为一单位矢量, 正负号的选择以减小舍入误差. 则

$$\begin{aligned} \mathbf{P} \cdot \mathbf{x} &= \mathbf{x} - \frac{\mathbf{u}}{H} \cdot (\mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1)^T \cdot \mathbf{x} \\ &= \mathbf{x} - \frac{2\mathbf{u} \cdot (|x|^2 \mp |x|x_1)}{2|x|^2 \mp 2|x|x_1} \\ &= \mathbf{x} - \mathbf{u} \\ &= \pm|\mathbf{x}|\mathbf{e}_1 \end{aligned} \tag{3.4.21}$$

这一结果证明, 当把$\mathbf{P}$ 作用在定义$\mathbf{P}$ 的给定矢量$\mathbf{x}$ 上时, 将把$\mathbf{x}$ 的除第一个元素外的所有元素变为0.

为了把实对称矩阵$\mathbf{A}$ 变换为三对角形式, 我们选择构成第一个Householder 矩阵的矢量$\mathbf{x}$ 为$\mathbf{A}$ 的第一列的后$n-1$ 个元素, 即取

$$\mathbf{x} = [0, a_{21}, a_{31}, \cdots, a_{n1}]^T \tag{3.4.22}$$

于是$\mathbf{P}_1$ 具有形式

$$\mathbf{P}_1 = \begin{bmatrix} 1 & 0 & 0 & & \cdots & 0 \\ 0 & & & & & \\ 0 & & & & & \\ \vdots & & & \mathbf{P}_1^{[n-1]} & & \\ 0 & & & & & \end{bmatrix} \tag{3.4.23}$$

这里$\mathbf{P}_1^{[n-1]}$ 为一$(n-1)\times(n-1)$ 维的Householder 矩阵. 用$\mathbf{P}$ 对$\mathbf{A}$ 进行变换, 得到

$$\mathbf{A}' = \mathbf{P_1}\cdot\mathbf{A}\cdot\mathbf{P} = \begin{bmatrix} a_{11} & k & 0 & \cdots & 0 \\ k & & & & \\ 0 & & & & \\ \vdots & & \mathbf{A}^{[n-1]} & & \\ 0 & & & & \end{bmatrix} \tag{3.4.24}$$

式中k 为(3.4.22) 给出的矢量$\mathbf{x}$ 的模的正值或负值, $\mathbf{A}^{[n-1]}$ 为一$(n-1)\times(n-1)$ 维实对称矩阵.

现在, 选择构成第二个Householder 矩阵的矢量$\mathbf{x}$ 为$\mathbf{A}^{[n-1]}$ 的第一列的后$n-2$ 个元素, 于是$\mathbf{P}_1$ 具有形式

$$\mathbf{P}_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & \mathbf{P}_2^{[n-2]} & & \\ 0 & 0 & & & \end{bmatrix} \tag{3.4.25}$$

左上角的单位矩阵保证了第一步所得到的三对角形式不会在这一步的变换中被改变. 用$\mathbf{P}_2$ 对$\mathbf{A}'$ 进行变换将产生第二个三对角输出. 显然, 经过$n-2$ 步这样的变换, 我们将得到一个三对角矩阵.

实际计算$\mathbf{P}\cdot\mathbf{A}\cdot\mathbf{P}$ 时, 并不需要做两次矩阵乘法, 而用下面的技巧, 计算矢量

$$\mathbf{p} \equiv \frac{\mathbf{A}\cdot\mathbf{u}}{H} \tag{3.4.26}$$

于是

$$\begin{aligned} \mathbf{A}\cdot\mathbf{P} &= \mathbf{A}\cdot\left(1-\frac{\mathbf{u}\cdot\mathbf{u}^T}{H}\right) = \mathbf{A}-\mathbf{p}\cdot\mathbf{u}^T \\ \mathbf{P}\cdot\mathbf{A}\cdot\mathbf{P} &= \mathbf{A}-\mathbf{p}\cdot\mathbf{u}^T-\mathbf{u}\cdot\mathbf{p}^T+2K\mathbf{u}\cdot\mathbf{u}^T \end{aligned}$$

这里K 由下式定义为

$$K = \frac{\mathbf{u}^T\cdot\mathbf{p}}{2H} \tag{3.4.27}$$

记

$$\mathbf{q} \equiv \mathbf{p}-K\mathbf{u} \tag{3.4.28}$$

则

$$\mathbf{P}\cdot\mathbf{A}\cdot\mathbf{P} = \mathbf{A}-\mathbf{q}\cdot\mathbf{u}^T-\mathbf{u}\cdot\mathbf{q}^T \tag{3.4.29}$$

这是实际使用的计算公式.

如果在计算的某一步$\mathbf{x}$ 的模为0, 则这一步变换无需进行, 为此, 可引入如下判据, 定义

$$\epsilon = \sum_{k=1}^{i-1} |a_{ik}| \tag{3.4.30}$$

如果在机器精度级$\epsilon = 0$, 则跳过这一次变换, 否则, 把a_{ik} 换为a_{ik}/ϵ, 并利用新的a_{ik} 进行变换, 因为Householder 变换只与元素的相对大小有关.

3.4.2 QL 算法

QL 算法的基本思想是, 任何实矩阵可以分解为

$$A = Q \cdot L \tag{3.4.31}$$

其中Q 为一正交矩阵, 而L 是一下三角矩阵. 现在把上式中前后两项互换相乘, 则有

$$A' = L \cdot Q \tag{3.4.32}$$

方程(3.4.31) 意味着$L = Q^T \cdot A$, 于是

$$A' = Q^T \cdot A \cdot Q \tag{3.4.33}$$

即A' 是A 的一个正交变换. 对于一般实矩阵, 可以证明, 如果反复进行上述变换, 则A 将依不同情况最终收敛于下面的形式:

(i), 如果A 的本征值的绝对值$|\lambda_i|$ 各不相同, 则A 收敛于一下三角矩阵, 其本征值按绝对值大小沿对角线排列.

(ii), 如果A 的某一本征值的绝对值$|\lambda_i|$ 具有简并度p, 则除了一个p 阶的位于对角位置的方阵外, A 收敛于一下三角矩阵, 且对角方阵的本征值趋于λ_i .

这一定理的证明较繁, 这里不亦给出. 对于实对称矩阵, 则A 收敛于一对角矩阵及代表相同本征值的对角块. 上三角元素趋于零的速度为

$$a_{ij}^{(s)} \sim \left(\frac{\lambda_i}{\lambda_j}\right)^s$$

这里上标s 代表叠代次数. 尽管$|\lambda_i| < |\lambda_j|$, 但当$|\lambda_i|$ 与$|\lambda_j|$ 很接近时, 收敛将是很慢的. 为了解决这一问题, 可采用移位叠代的方法. 我们不在此讨论这种方法. 针对对称矩阵, 可以采用隐式位移叠代方法. 关于这些方法的介绍, 请参看J. H. Wilkinson and C. Reinsch *Linear Algebra* 一书.

物理中常见的另一种矩阵是厄米矩阵, 这在量子力学问题中经常出现. 如果C 为一厄米矩阵, $C = A + i\,B$, 则显然有, $A^T = A$, $B^T = -B$, 因此, 矩阵C 的本征值问题可以转

化为一个实对称矩阵的问题. C的本征值问题是

$$(A + i\,B)(u + i\,v) = \lambda(u + i\,v) \tag{3.4.34}$$

代替上面的方程, 我们考虑下面的本征值问题

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.4.35}$$

这是一个实对称矩阵的本征值问题, 若C 的本征值为$\lambda_1, \lambda_2, \cdots, \lambda_n$, 本征向量为$u + i\,v$, 则方程(3.4.35)的本征值为$\lambda_1, \lambda_1, \lambda_2, \lambda_2, \cdots, \lambda_n, \lambda_n$, 本征向量为$[u, v]^T$ 和$[-v, u]^T$, 因此, 由上述对应关系及我们已经知道的实对称矩阵问题的解法, 就可以求得厄米矩阵的本征值问题.

练习

请证明前段的论断

3.5 负本征值方法和递推方法

在物理应用上，有时候并不需要精确计算出本征值和本征矢量，而是需要计算出本征值的分布（谱分布或态密度），这一节介绍一种计算态密度的简单方法，这个方法特别适合于三对角（或更一般地，带状矩阵）态密度的计算。

考虑如下带状矩阵

$$\mathbf{D}=\begin{bmatrix} \mathbf{A_1} & \mathbf{B_2} & & & \\ \mathbf{B_2}^T & \mathbf{A_2} & \mathbf{B_3} & & \\ & \mathbf{B_3}^T & \mathbf{A}_3 & \mathbf{B}_4 & \\ & & \cdots & \cdots & \cdots \\ & & & \mathbf{B}_n^T & \mathbf{A_n} \end{bmatrix} \tag{3.5.36}$$

此处$\mathbf{A_i}$为$m_i \times m_i$的子矩阵，而$\mathbf{B_i}$为$m_i \times m_{i-1}$的子矩阵。各个m_i无需相等，但我们假定m_i均较小，精确处理这样大小的矩阵没有困难。

P. Dean 和M. D. Bacon在1965年（在1960年Deen和Martin的文章的基础上）证明了如下定理。(P. Dean and M.D. Bacon, Proc. R. Soc. London A, 283,1392, 64-82(1965), P. Dean and J. L. Martin, Proc. R. Soc. London A, 259,1298, 409-418(1960)) 如果用$\eta(\mathbf{M})$表示矩阵$\mathbf{M}$的负本征值的数目(小于零的本征值的数目），那么

$$\eta(\mathbf{D}-E\mathbf{I})=\sum_i^n \eta(\mathbf{U}_i) \tag{3.5.37}$$

其中，$\mathbf{I}$是单位矩阵，

$$\mathbf{U}_i=\mathbf{A}_i-E\mathbf{I}_i-\mathbf{B}_i^T\mathbf{U}_{i-1}^{-1}\mathbf{B}_i \quad (i=2,3,\cdots,n) \tag{3.5.38}$$

$$\mathbf{U}_1=\mathbf{A}_1-E\mathbf{I}_1 \tag{3.5.39}$$

我们不证明这个定理（证明过程并不难），有兴趣的同学可以参看原文。利用这个定理，就可以计算矩阵$\mathbf{D}$的本征值分布（态密度），计算方法如下，先取一个足够小的E，使得矩阵$\mathbf{D}-E\mathbf{I}$的负本征值数目为零，然后，不断增加E, 计算每个E对应的$\mathbf{D}-E\mathbf{I}$的负本征值数目。由此数目之差就可以得到在E附近ΔE范围内的本征值数目，从而得到态密度。

对于三对角实对称矩阵，这一方法特别简单。设三对角矩阵为

$$\mathbf{D}=\begin{bmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & b_3 & & \\ & b_3 & a_3 & b_4 & \\ & & \cdots & \cdots & \cdots \\ & & & b_n & a_n \end{bmatrix} \tag{3.5.40}$$

则公式(3.5.38), (3.5.39)成为

$$U_i = a_i - E - b_i^2/U_{i-1} \tag{3.5.41}$$

$$U_1 = a_1 - E \tag{3.5.42}$$

习题：

一维斐波那契链可以通过如下规则形成。设想一个一维原子链，原子的格点能量为a, 近邻格点间的跃迁可以取值b和c. b和c以斐波那契方式排列。其生成规则如下：从c出发，$c \to b$, 变为b; $b \to bc$, 变为bc; $b \to bc$, $c \to b$, 成为bcb. 继续这一变换，可以得到

$$bcb$$

$$bcbbc$$

$$bcbbcbcb$$

$$bcbbcbcbbcbbc$$

$$\cdots$$

上述序列也可以通过不断把前一序列加到当前序列的后面而得到。

试产生不同长度的序列（*2000–200000*），取$a = 0$, $b = 1$, c在1–2之间取若干值, 利用负本征值方法计算其态密度及积分态密度。

在很多物理问题中，我们实际上并不需要得到所有的本征值，而是只需要最小的本征值（基态）和次小本征值（第一激发态）。对这样的问题，可以使用递推方法计算。

设我们研究的问题的矩阵为对称矩阵$\mathbf{D}$, 其基态为$|g\rangle$. 取一个包含基态的归一化出发态$|0\}$, 这是容易做到的，一般取随机矢量就能满足要求。

把$\mathbf{D}$作用于矢量$|0\}$得到新的矢量，并由此定义一个与$|0\}$正交的矢量$|1\}$，

$$b_1|1\} = \mathbf{D}|0\} - a_0|0\}$$

用$|0\}$左乘上式，得到

$$a_0 = \{0|\mathbf{D}|0\}$$

b_1由归一化确定。再作用一次

$$b_2|2\} = \mathbf{D}|1\} - a_1|1\} - b_1|0\}$$

要求$|2\}$与$|1\}$和$|0\}$垂直，$\{2|1\} = 0$, $\{2|0\} = 0$, 得到

$$a_1 = \{1|\mathbf{D}|1\}$$

b_2由归一化条件确定。

继续这一过程，

$$b_3|3\} = \mathbf{D}|2\} - a_2|2\} - b_2|1\}$$

要求|3}与|2}和|1}正交，得到

$$a_2 = \{2|\mathbf{D}|2\}$$

再由归一化条件确定b_3。这样确定的|3}也与|0}正交，因为

$$\{0|3\} \propto \{0|\mathbf{D}|2\} \propto b_1\{1|2\} + a_0\{0|2\} = 0$$

这一过程可以一直进行下去，并得到一系列相互正交的矢量

$$|0\}, |1\}, |2\}, \cdots$$

以及一系列数

$$a_0, a_1, a_2, \cdots$$

$$b_1, b_2, b_3, \cdots$$

且这些数构成一个三对角矩阵

$$\mathbf{D}' = \begin{bmatrix} a_0 & b_1 & & & \\ b_1 & a_1 & b_2 & & \\ & b_2 & a_2 & b_3 & \\ & & \cdots & \cdots & \cdots \\ & & & b_n & a_n \end{bmatrix}$$

这一矩阵实际上来源于原矩阵的正则变换，所以其本征值是相同的。但是，上述算法并不稳定，每次递推，可能混入一点其它状态的成分，从而导致正交性被破坏. 因此，通常只能递推20次左右，这样得到的是一个比原矩阵小了很多的矩阵，不可能反映原矩阵的所有信息. 好在这一矩阵的最低本征值与原矩阵的最低本征值比较接近，最大本征值与原矩阵的最大本征值也比较接近，中间的本征值的分布与原矩阵本征值的分布也有很强的联系。因此可以用来近似计算基态和态密度。

Chapter 4

电磁场的计算

4.1 Maxwell 方程, 边值问题

电磁场的分析和计算在物理和工程应用中都占有重要的地位. 它的基础是Maxwell 方程组. 从数学上看, 电磁场的分析和计算问题归结为偏微分方程边值问题的求解. 具体的计算方法可以分为两大类, 一类是解析方法, 另一种是数值方法. 解析方法就是把要求的解表示成解析形式或无穷级数, 它只适用于边界形状比较简单(如球, 柱等)的问题, 应用范围相当有限. 这一方法在《数学物理方法》课程中已经做了详细讨论. 数值方法是解决实际问题最有效最普遍的方法, 随着电子计算机的高速发展, 它已成为工程中的一种常规分析方法. 这一章将以静电场的计算为例, 简单介绍有限差分法和有限元方法, 这是目前应用最多的两种方法.

电磁场计算的理论基础是Maxwell 方程, 它的微分形式为:

$$\begin{cases} \nabla \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{D} = \rho \\ \nabla \times \mathbf{H} = \mathbf{J} + \dfrac{\partial \mathbf{D}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 \end{cases} \tag{4.1.1}$$

式中, $\mathbf{E}$为电场强度, $\mathbf{D}$为电位移矢量, $\mathbf{B}$为磁感应强度, $\mathbf{H}$为磁场强度, ρ和$\mathbf{J}$分别为电荷密度和电流密度. 对于各向同性的线性介质, 上式中各量之间满足下面的本构方程

$$\begin{cases} \mathbf{D} = \varepsilon \mathbf{E} \\ \mathbf{B} = \mu \mathbf{H} \\ \mathbf{J} = \sigma \mathbf{E} \end{cases} \tag{4.1.2}$$

其中ε, μ, σ分别为介质的介电常数, 导磁系数和电导率. 在两种介质的分界面上, 介质的特性系数如ε, μ, σ将发生突变. 在这种情况下将出现面电荷, 面电流的分布, 从而使场矢

量也发生突变, 这种突变由下面的联接条件给出

$$\begin{cases} \mathbf{n}\cdot(\mathbf{D_2}-\mathbf{D_1})=\rho_s \\ \mathbf{n}\times(\mathbf{E_2}-\mathbf{E_1})=0 \\ \mathbf{n}\cdot(\mathbf{B_2}-\mathbf{B_1})=0 \\ \mathbf{n}\times(\mathbf{H_2}-\mathbf{H_1})=\mathbf{J}_s \\ \mathbf{n}\cdot(\mathbf{J_2}-\mathbf{J_1})=-\dfrac{d\rho_s}{dt} \end{cases} \tag{4.1.3}$$

式中$\mathbf{n}$为界面上从介质1 指向介质2 的单位矢量, ρ_s为自由面电荷密度, $\mathbf{J_s}$为自由面电流密度.

在静电场的情况下, Maxwell 方程组简化为

$$\begin{cases} \nabla\times\mathbf{E}=0 \\ \nabla\cdot\mathbf{E}=\dfrac{\rho}{\varepsilon} \end{cases} \tag{4.1.4}$$

由于(4.1.4)的第一式, 可以引进一标量势函数ϕ. 令

$$\mathbf{E}=-\nabla\phi \tag{4.1.5}$$

则(4.1.4)的第一式自动满足, 代入(4.1.4)的第二式可得ϕ 满足的偏微分方程为

$$\nabla^2\phi=-\frac{\rho}{\varepsilon} \tag{4.1.6}$$

这时介质分界面上的联接条件为

$$\begin{cases} \phi_1=\phi_2 \\ \varepsilon_1\dfrac{\partial\phi_1}{\partial n}-\varepsilon_2\dfrac{\partial\phi_2}{\partial n}=\rho_s \end{cases} \tag{4.1.7}$$

对于静电场, 通常考虑下面三种边界条件:

1. 给定整个场域边界Γ上的电势值$\phi_{|\Gamma}$. 这通常称为第一类边值问题或Dirichlet边值问题.

2. 给定边界Γ上电势的法向导数$\frac{\partial\phi}{\partial n}_{|\Gamma}$, 这通常称为第二类边值问题或Neumann边值问题.

3. 在边界的一部分给定电势值$\phi_{|\Gamma_1}$, 另一部分边界上给定电势的法向导数$\frac{\partial\phi}{\partial n}_{|\Gamma_2}$. 这通常称为混合边值问题.

可以证明, 静电问题的上述三种边值问题的解是存在而且唯一的(例如见D. J. Jakson *Classical Electrodynamics* John Wiley & Sons, Inc., 1975, P42)

4.2 差分和差商

这一节简单复习一下差分和差商的概念. 这些内容在前面的章节中曾分散地出现过, 这里则综合起来理一下.

设有函数$f(x)$, 若其宗量的增量为$\Delta x = h$, 则对应函数的增量为

$$\Delta f(x) = f(x+h) - f(x) \tag{4.2.8}$$

称为函数$f(x)$ 在x 点的一阶向前差分, 或简称为差分. 差分也可定义为

$$\Delta f(x) = f(x) - f(x-h) \tag{4.2.9}$$

称为函数$f(x)$ 在x 点的一阶向后差分及

$$\Delta f(x) = f\left(x + \frac{h}{2}\right) - f\left(x - \frac{h}{2}\right) \tag{4.2.10}$$

称为函数$f(x)$ 在x 点的一阶中心差分.

一阶差分除以增量h称为一阶差商, 根据不同的差分定义, 我们有

$$\begin{aligned}
&\text{一阶向前差商:}\ \frac{\Delta f}{\Delta x} = \frac{f(x+h) - f(x)}{h} \\
&\text{一阶向后差商:}\ \frac{\Delta f}{\Delta x} = \frac{f(x) - f(x-h)}{h} \\
&\text{一阶中心差商:}\ \frac{\Delta f}{\Delta x} = \frac{f(x+h/2) - f(x-h/2)}{h}
\end{aligned} \tag{4.2.11}$$

当h趋向于零时, 上述三种差商均趋于微商$\frac{df}{dx}$, 因此, 一阶差商在实际计算中常常做为一阶微商的近似值. 下面我们来分析一下用差商代替微商的近似程度. 由Taylor展开式

$$\begin{aligned}
f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \cdots \\
f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \cdots
\end{aligned}$$

可以得到:

$$\begin{aligned}
\frac{f(x+h) - f(x)}{h} - f'(x) &= \frac{h}{2!}f''(x) + \cdots = \mathcal{O}(h) \\
\frac{f(x) - f(x-h)}{h} - f'(x) &= -\frac{h}{2!}f''(x) + \cdots = \mathcal{O}(h) \\
\frac{f(x+h/2) - f(x-h/2)}{h} - f'(x) &= \frac{h^2}{3!}f''(x) + \cdots = \mathcal{O}(h^2)
\end{aligned} \tag{4.2.12}$$

由上式可知, 中心差商的截断误差与h^2同阶, 比向前差商和向后差商高一阶, 因而精度最高. 同一阶差商类似, 我们还可定义二阶差商和高阶差商. 例如二阶中心差商定义为

$$\frac{\Delta^2 f(x)}{h^2} \equiv \frac{\Delta f(x+h/2) - \Delta f(x-h/2)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{4.2.13}$$

容易验证

$$\frac{\Delta^2 f(x)}{h^2} = f''(x) + \mathcal{O}(h^2)$$

用有限差分方法求微分方程的解的过程就是用有限差分代替微分, 从而把微分方程化为差分方程, 然后对所得差分方程进行数值求解.

4.3 二维Possion 方程的五点差分格式

这一节以二维Possion方程的第一类边值问题为例来具体说明有限差分方法的原理和实现方法. 定解问题为

$$\begin{cases} \nabla^2\phi = -\dfrac{\rho}{\varepsilon} \\ \phi_{|\Gamma} = f \end{cases} \tag{4.3.14}$$

对于平面问题, 在直角坐标系中, (4.3.14)中的第一个式子为

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = -\frac{\rho}{\varepsilon} \tag{4.3.15}$$

对于具有旋转对称的静电问题, 可以采用柱坐标系(z, r, θ), 由于对称性, 电势与θ无关, 因此Posion方程成为

$$\frac{\partial^2\phi}{\partial z^2} + \frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} = -\frac{\rho}{\varepsilon} \tag{4.3.16}$$

三维问题退化成一个二维问题. 如果引进一个参量α, 则可把(4.3.15)和(4.3.16)写成下面的统一形式, 从而可进行统一处理.

$$\frac{\partial^2\phi}{\partial z^2} + \frac{\partial^2\phi}{\partial r^2} + \frac{\alpha}{r}\frac{\partial\phi}{\partial r} = -\frac{\rho}{\varepsilon} \tag{4.3.17}$$

当$\alpha = 1$时, 对应于旋转对称的场, 此时(r, z)是圆柱坐标系中的径向和轴向坐标; 当$\alpha = 0$时, 相应于二维平面问题, 此时(r, z)对应于直角坐标系中的(y, x). 假定我们的求解区域如图4.1, 则我们可用两族平行于坐标轴的直线对区域进行划分, 如图4.1所示.

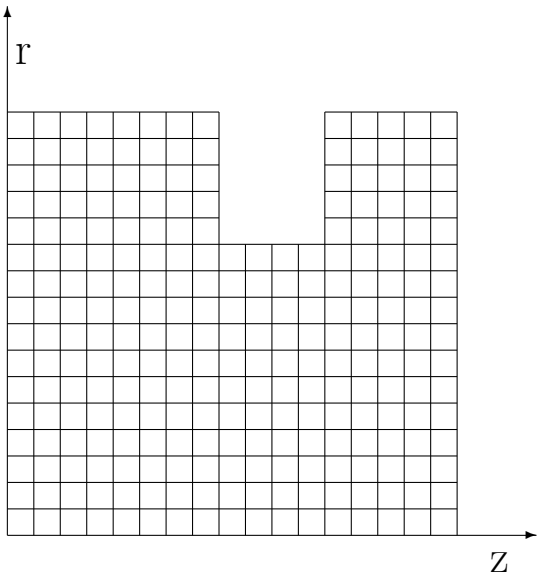


Figure 4.1: 二维空间的网格划分

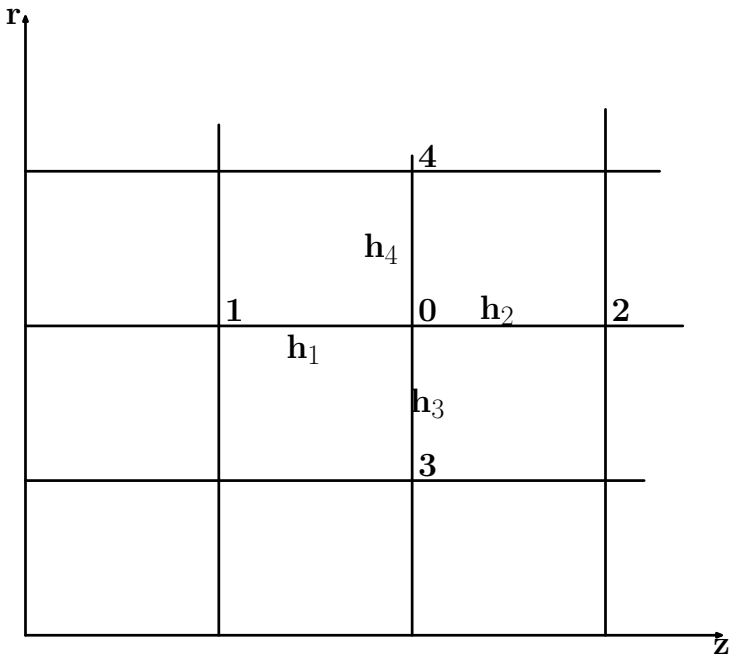


Figure 4.2:

对于上述划分的每一个网格点, 可以建立其差分方程, 为具体起见, 考虑图4.2所示一个网格点(记为0)及其近邻点(记为1,2,3,4), 则点1,2,3,4 的电势值可以用0点的电势值表示出来, 对1,2,3,4点的电势作Taylor展开, 有

$$\begin{aligned}
\phi_1 &= \phi_0 - h_1\left(\frac{\partial\phi}{\partial z}\right)_{|0} + \frac{h_1^2}{2}\left(\frac{\partial^2\phi}{\partial z^2}\right)_{|0} + \cdots \\
\phi_2 &= \phi_0 + h_2\left(\frac{\partial\phi}{\partial z}\right)_{|0} + \frac{h_2^2}{2}\left(\frac{\partial^2\phi}{\partial z^2}\right)_{|0} + \cdots \\
\phi_3 &= \phi_0 - h_3\left(\frac{\partial\phi}{\partial r}\right)_{|0} + \frac{h_3^2}{2}\left(\frac{\partial^2\phi}{\partial r^2}\right)_{|0} + \cdots \\
\phi_4 &= \phi_0 + h_4\left(\frac{\partial\phi}{\partial r}\right)_{|0} + \frac{h_4^2}{2}\left(\frac{\partial^2\phi}{\partial r^2}\right)_{|0} + \cdots
\end{aligned} \tag{4.3.18}$$

由上式可解得:

$$\begin{aligned}
\left(\frac{\partial^2\phi}{\partial z^2}\right)_{|0} &= 2\frac{h_2\phi_1 + h_1\phi_2}{h_1h_2(h_1+h_2)} - \frac{2\phi_0}{h_1h_2} + \mathcal{O}(h^2) \\
\left(\frac{\partial^2\phi}{\partial r^2}\right)_{|0} &= 2\frac{h_4\phi_3 + h_3\phi_4}{h_3h_4(h_3+h_4)} - \frac{2\phi_0}{h_3h_4} + \mathcal{O}(h^2) \\
\left(\frac{\alpha}{r}\frac{\partial\phi}{\partial r}\right)_{|0} &= \frac{\alpha}{r_0(h_3+h_4)}(\phi_4 - \phi_3) + \mathcal{O}(h^2)
\end{aligned} \tag{4.3.19}$$

在得出上式时, 我们假定h_1, h_2, h_3, h_4为同阶小量, 且任二者之差为高一阶的小量, 其阶

统一记为$\mathcal{O}(h)$. 代入Possion方程, 并略去$\mathcal{O}(h^2)$以下的小量, 我们得到

$$2\frac{h_2\phi_1+h_1\phi_2}{h_1h_2(h_1+h_2)}+2\frac{h_4\phi_3+h_3\phi_4}{h_3h_4(h_3+h_4)}-\frac{2\phi_0}{h_1h_2}-\frac{2\phi_0}{h_3h_4}+\frac{\alpha}{r_0(h_3+h_4)}(\phi_4-\phi_3)=-\frac{\rho_{|0}}{\varepsilon}$$

整理后得

$$C_1\phi_1+C_2\phi_2+C_3\phi_3+C_4\phi_4-C_0\phi_0=-\frac{\rho_{|0}}{\varepsilon} \tag{4.3.20}$$

其中

$$\begin{cases} C_1=\dfrac{2}{h_1(h_1+h_2)} \\ C_2=\dfrac{2}{h_2(h_1+h_2)} \\ C_3=\dfrac{2}{h_3(h_3+h_4)}-\dfrac{\alpha}{r_0(h_3+h_4)} \\ C_4=\dfrac{2}{h_4(h_3+h_4)}+\dfrac{\alpha}{r_0(h_3+h_4)} \\ C_0=C_1+C_2+C_3+C_4=\dfrac{2}{h_1h_2}+\dfrac{2}{h_3h_4} \end{cases} \tag{4.3.21}$$

式(4.3.20)与(4.3.21)一起称为Possion方程的五点差分格式.

对于旋转对称场, 在原点($r_0=0$), 方程(4.3.20)不成立. 注意到如果电荷密度在原点没有奇异性, 则$\frac{\partial\phi}{\partial r}$在$r=0$时趋于0, 因而当$r\to 0$时, 有

$$\lim_{r\to 0}\frac{1}{r}\frac{\partial\phi}{\partial r}=\left(\frac{\partial^2\phi}{\partial r^2}\right)_{|0}$$

因此, 在$r=0$附近, 旋转对称场的Possion方程为

$$\frac{\partial^2\phi}{\partial z^2}+2\frac{\partial^2\phi}{\partial r^2}=-\frac{\rho}{\varepsilon} \tag{4.3.22}$$

在$r=0$点, (见图4.3)

$$\left(\frac{\partial^2\phi}{\partial r^2}\right)_{|0}=\frac{2}{h_4^2}\phi_4-\frac{2}{h_4^2}\phi_0$$

从而得到轴上的与式(4.3.20)形式相同的差分公式, 但其系数为

$$\begin{cases} C_1=\dfrac{2}{h_1(h_1+h_2)} \\ C_2=\dfrac{2}{h_2(h_1+h_2)} \\ C_3=0 \\ C_4=\dfrac{4}{h_4^2} \\ C_0=C_1+C_2+C_3+C_4=\dfrac{2}{h_1h_2}+\dfrac{4}{h_4^2} \end{cases} \tag{4.3.23}$$

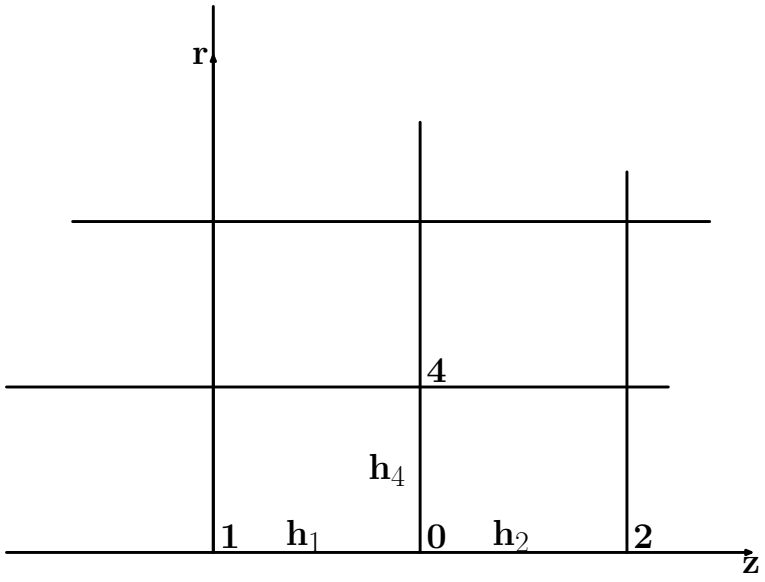


Figure 4.3:

对于求解区域的每个内节点(指不在区域边界上的点), 都可建立类似的差分方程, 方程中的C_0, C_1, C_2, C_3, C_4 是决定于步长的已知函数, 而各个$\phi_0, \phi_1, \phi_2, \phi_3, \phi_4$是待求的未知量. 每个内节点有一个方程, 一个未知量, 因此未知量的数目等于方程的数目. 邻近边界的节点的差分方程中, 包含有边界点的电势值, 这是已知的边界条件. 这样, 如果有n 个内节点, 我们就建立了n个线性代数方程, 可以一般地写为

$$\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{B} \tag{4.3.24}$$

通过求解上面的方程便可得到所需的各ϕ值.

这一节我们建立了二维问题的五点差分格式, 这一方法很容易推广到三维, 其差分格式称为七点差分格式. 同样的方法也可用来处理一维问题, 为三点差分格式.

4.4 差分方程的求解

在上一节, 我们得到了Possion方程的差分方程, 这一节讨论其解法. 方程(4.3.24)是一线性代数方程组, 原则上可以用前面介绍过的求解线性代数方程组的一般方法求解, 但是, 由于此方程的一些特殊的性质, 一般解法将会遇到困难. 为此, 我们先分析一下方程(4.3.24) 的系数矩阵**A**, 它具有下面的性质:

1. **A**是一个稀疏矩阵, 事实上, **A**的每一行或每一列的非零元素数目最多只有五个.
2. **A**是一个高阶矩阵, 由于每个内点有一个方程, 因此**A**的阶数为为内点的数目, 如果内点的数目为n, 则**A**为一$n \times n$ 矩阵, 一般在计算中为了提高精度, 网格分得较细, 内点数目较多, 因而**A** 的阶数也就较高.

由于**A**的上述性质, 用通常的解法在时间和存储空间都是十分浪费的, 更严重的是, 由于**A**的阶数较高, $n \times n$个元素可能根本无法放入内存. 根据**A**的性质, 一般用迭代法求解. 下面介绍三种常用的迭代方法

4.4.1 简单迭代法—Jacobi 方法

直接利用差分方程(4.3.20), 把其改写为

$$\phi_0 = \frac{C_1}{C_0}\phi_1 + \frac{C_2}{C_0}\phi_2 + \frac{C_3}{C_0}\phi_3 + \frac{C_4}{C_0}\phi_4 - \frac{\rho_{|0}}{C_0\varepsilon} \tag{4.4.25}$$

上式就可作为一迭代格式, 为明确起见, 把迭代格式写为

$$\phi_0^{(k+1)} = \frac{C_1}{C_0}\phi_1^{(k)} + \frac{C_2}{C_0}\phi_2^{(k)} + \frac{C_3}{C_0}\phi_3^{(k)} + \frac{C_4}{C_0}\phi_4^{(k)} - \frac{\rho_{|0}}{C_0\varepsilon} \tag{4.4.26}$$

给定初始的$\phi^{(0)}$, 利用上式对每一个内点进行迭代, 直到迭代前后的值小于某一给定的的误差限为止. 可以证明, 不论$\phi^{(0)}$如何选取, 当$k \to \infty$时, $\phi^{(k)}$必收敛于差分方程的解. 由式(4.4.26)可知, 计算第$k+1$次的某一点的电势值时所用的其邻近点的电势是第k次迭代的结果, 因此只有当第k 次的n个内点计算完后, 才可开始第$k+1$次的计算, 这样, 就必须为每个内点分配两个存储单元, 以储存相邻两次迭代的电势值. 另外, 实际计算表明, 这种迭代方法的收敛速度也很慢.

4.4.2 逐次迭代法—Gauss–Seidal 迭代方法

这是对简单迭代的一种直接的改进, 在迭代中, 如果对某一格点作第$k+1$次迭代时, 如果相邻格点中部分点的第$k+1$次电势值已算出, 则利用新的数值. 对于如图4.1所示的网格, 如果在z方向的计算顺序是从左到右, 在r方向的计算顺序是从下到上, 则Gauss—Seidal迭代公式为

$$\phi_0^{(k+1)} = \frac{C_1}{C_0}\phi_1^{(k+1)} + \frac{C_2}{C_0}\phi_2^{(k)} + \frac{C_3}{C_0}\phi_3^{(k+1)} + \frac{C_4}{C_0}\phi_4^{(k)} - \frac{\rho_{|0}}{C_0\varepsilon} \tag{4.4.27}$$

逐次迭代法的收敛速度比简单迭代法要快, 同时可以节省计算机存储量, 因为每个格点只需存储一套电势值.

4.4.3 逐次超松弛迭代法(Successive Over-Relaxation)

逐次超松弛迭代法是加快收敛的一种方法, 它介于前述两种方法之间. 具体作法是, 把第$k+1$次Gauss–Seidal迭代的结果与第k次迭代的结果进行加权平均作为新的第$k+1$次的迭代值. 记$k+1$次Gauss–Seidal迭代的结果为$\tilde{\phi}$, 则有

$$\tilde{\phi}_0 = \frac{C_1}{C_0}\phi_1^{(k+1)} + \frac{C_2}{C_0}\phi_2^{(k)} + \frac{C_3}{C_0}\phi_3^{(k+1)} + \frac{C_4}{C_0}\phi_4^{(k)} - \frac{\rho_{|0}}{C_0\varepsilon} \tag{4.4.28}$$

$$\phi_0^{(k+1)} = \omega\tilde{\phi}_0 + (1-\omega)\phi_0^{(k)} \tag{4.4.29}$$

将(4.4.28)代入(4.4.29), 得到

$$\phi_0^{(k+1)} = (1-\omega)\phi_0^{(k)} + \omega\left(\frac{C_1}{C_0}\phi_1^{(k+1)} + \frac{C_2}{C_0}\phi_2^{(k)} + \frac{C_3}{C_0}\phi_3^{(k+1)} + \frac{C_4}{C_0}\phi_4^{(k)} - \frac{\rho_{|0}}{C_0\varepsilon}\right) \tag{4.4.30}$$

当$\omega = 1$时, 上式就成为Gauss–Seidal 迭代. ω称为迭代松弛因子, 又叫阻尼因子. 可以证明, 当$0 < \omega < 2$时, 迭代(4.4.30)是收敛的. 当$1 < \omega < 2$时, 迭代(4.4.30)称为超松弛迭代, 相应地ω称为超松弛迭代因子. 迭代因子的选取对收敛速度的影响很大, 一般它与网格形式, 节点数目, 迭代次序等因素有关. 在计算数学的理论书中, 对最佳因子的选取有不少讨论, 这里介绍一种选取的方法.(证明见南京大学数学系计算数学专业编《**偏微分方程数值解法**》, 科学出版社, 1979)

记$\phi_i^{(k)}$为第i个内节点第k次迭代的电势值(这里$i = 1, 2, \cdots n$代表n个内节点, 请与前面$i = 0$, 1, 2, 3, 4代表一点及其近邻相区别), 取相邻两次迭代的误差的平均为

$$\bar{\epsilon}^{(k)} = \frac{1}{n}\sum_{i=1}^{n}\left|\phi_i^{(k)} - \phi_i^{(k-1)}\right| \tag{4.4.31}$$

记

$$\lambda^{(k)} = \frac{\bar{\epsilon}^{(k+1)}}{\bar{\epsilon}^{(k)}} \tag{4.4.32}$$

若第k次迭代所用的超松弛因子为$\omega^{(k)}$, 定义

$$\mu^2 = \frac{(\lambda^{(k)} + \omega^{(k)} - 1)^2}{\lambda^{(k)}(\omega^{(k)})^2} \tag{4.4.33}$$

则第$k+1$次迭代的超松弛因子为$\omega^{(k+1)}$可取为

$$\omega^{(k+1)} = \frac{2}{1 + \sqrt{1 - \mu^2}} \tag{4.4.34}$$

一般可取$\omega^{(0)}$为1.5 左右, 利用上述过程不断改进ω的取值, 当$|\omega^{(k+1)} - \omega^{(k)}|$小于某一误差限(例如$10^{-2}$)时, 就用所得的$\omega$作为最佳迭代因子, 进行计算.

当相邻两次迭代的误差小于某一事先给定的误差限时, 就可终止计算. 通常误差可选为平均相对误差

$$\frac{\sum_{i=1}^{n}\left|\phi_i^{(k+1)} - \phi_i^{(k)}\right|}{\sum_{i=1}^{n}\phi_i^{(k)}}$$

或最大绝对误差

$$\mathrm{Max}\left\{\left|\phi_i^{(k+1)} - \phi_i^{(k)}\right|\right\}$$

最后需要指出的是, 为了达到较高的计算精度, 减小截断误差, 就要减小步长h, 从而增加节点数, 增大计算量. 但随着计算量的增加, 舍入误差将随之增加, 因此, 对于实际的计算, 并非网格分得越小越好, 对具体问题, 应根据问题本身的特点和计算要求, 选则合适的网格.

有限差分方法概念清楚, 方法简单, 计算量也较小. 但对边界的处理不够灵活, 对于复杂的边界条件, 很难处理, 另外对于部分求解区域电势变化比较大而另外的区域变化比较平缓的问题, 处理起来也不灵活. 这些困难, 在我们后面要讲的有限元方法中可以得到较好的处理.

4.5 变分方法复习

为了给下一节的有限元方法作准备, 这一节复习一下变分方法. 考虑一个满足一定条件的函数的集合$\mathcal{F}:\{y(x)\}$ 和一个实数的集合R. 如果存在一个对应关系J, 使得对于$\mathcal{F}$中的每一个元素$y(x)$, 有R中的一个数J与之对应, 则称J 为函数$y(x)$的泛函, 记为

$$J = J[y(x)] \tag{4.5.35}$$

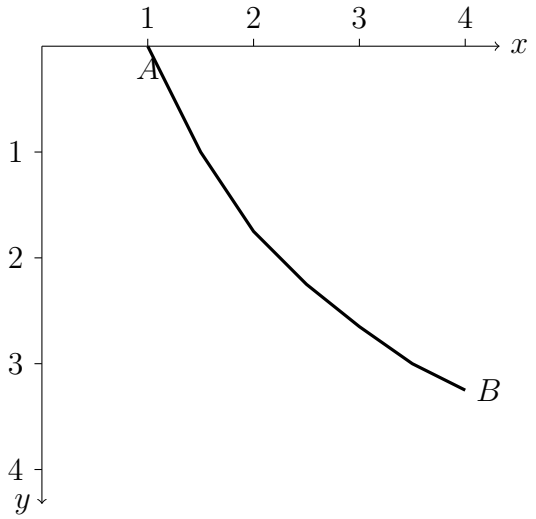


Figure 4.4:

我们来看一个泛函的例子, 如图4.4所示, 一质点沿曲线$y = y(x)$ 以初速度0从A降落到B, 其所需的时间为,

$$T = \int_A^B dt = \int_A^B \frac{ds}{v} = \int_A^B \frac{\sqrt{1+y'^2}}{\sqrt{2gy}} dx \tag{4.5.36}$$

当A, B固定时, 对于不同的轨道$y(x)$, 将有不同的时间T, 因此, T 是轨道$y(x)$的泛函, $T = T[y(x)]$.

对于泛函$J[y(x)]$, 如果函数$y(x)$改变$\delta y(x)$, 则其对应的泛函的改变为

$$\Delta J = J[y(x) + \delta y(x)] - J[y(x)] \tag{4.5.37}$$

当$\delta y(x) \to 0$ 时, 式(4.5.37)的极限称为泛函$J[y(x)]$的变分,记为δJ. 类似于函数的微分, 泛函的变分可写为

$$\delta J = \int \frac{\delta J}{\delta y(x)} \delta y(x) dx \tag{4.5.38}$$

其中$\frac{\delta J}{\delta y(x)}$ 称为泛函J 的变分导数, 它既是$y(x)$的泛函, 又是x的函数. 如果泛函在$y(x) = y_0(x)$时取极值, 则

$$\frac{\delta J}{\delta y(x)}_{|y(x)=y_0(x)} = 0 \tag{4.5.39}$$

下面看几个例子.

例一: $J = y(x_0)$, 泛函为$x = x_0$点上函数$y(x)$的值. 为了计算变分, 注意到J 可写为

$$J = y(x_0) = \int_{-\infty}^{+\infty} y(x)\delta(x - x_0)dx$$

其变分为

$$\begin{aligned}\delta J &= \int_{-\infty}^{+\infty} (y(x) + \delta y(x))\delta(x - x_0)dx - \int_{-\infty}^{+\infty} y(x)\delta(x - x_0)dx \\ &= \int_{-\infty}^{+\infty} \delta(x - x_0)\delta y(x)dx\end{aligned}$$

与(4.5.38)比较得

$$\frac{\delta y(x_0)}{\delta y(x)} = \delta(x - x_0)$$

例二:

$$J = \int_a^b y(x)^n dx$$

其变分为

$$\delta J = \int_a^b (y(x) + \delta y(x))^n dx - \int_a^b y(x)^n dx = \int_a^b n y(x)^{n-1}\delta y(x)dx + O(\delta y(x)^2)$$

而变分导数为

$$\frac{\delta J}{\delta y(x)} = n y(x)^{n-1}$$

例三:

$$J = \int_a^b e^{y(x)} dx$$

其变分为

$$\delta J = \int_a^b e^{(y(x)+\delta y(x))}dx - \int_a^b e^{y(x)}dx = \int_a^b e^{y(x)}\delta y(x)dx + \mathcal{O}(\delta y(x)^2)$$

而变分导数为

$$\frac{\delta J}{\delta y(x)} = e^{y(x)}$$

从这些例子我们看到, 泛函与函数有很多相似的性质, 有兴趣更进一步了解泛函及其性质的同学可研读专门的书籍. 下面我们研究一种特殊的泛函—积分泛函, 并指出求其极值的方法. 这种泛函在物理的各个领域都有重要应用.

为了下面的应用, 我们考虑定义在多元函数空间上的积分泛函

$$J = \int F\left(x_1, x_2, \cdots, x_n, y, \frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \cdots, \frac{\partial y}{\partial x_n}\right) dx_1 dx_2 \cdots, dx_n \tag{4.5.40}$$

这里, $y = y(x_1, x_2, \cdots, x_n)$是一$n$元函数. 本节开始处质点沿不同路径的时间就是这种类型泛函的一个例子. 现在, 我们来求泛函(4.5.40)的极值. 所谓极值, 就是要在函数y的集

合$\mathcal{F}$中寻找一个特殊的函数y, 使得泛函取极大值或极小值. 这通常有两种方法, 一种是直接法, 另一种是微分方程方法, 分别介绍如下.

一, 直接法: 在集合$\mathcal{F}$中构造一个完备的函数集合$\{\varphi_i(x_1, x_2, \cdots, x_n)\}$, 一般这是一个无穷集合, 取其前$N$个函数展开$y$, 我们得到

$$y = \sum_{i=1}^{N} c_i \varphi_i \tag{4.5.41}$$

把上式代入(4.5.40)并完成积分, 则泛函成为N个变量$c_1, c_2, \cdots, c_N$ 的函数, 而计算泛函极值的问题就转化为求多元函数极值的问题. 对于给定的N, 求得的与J 的极值对应的y记为y_N, 则

$$y(x_1, x_2, \cdots, x_n) = \lim_{N\to\infty} y_N(x_1, x_2, \cdots, x_n) \tag{4.5.42}$$

就趋于极值问题的解.

二, 微分方程法: 由于泛函的极值由其变分为0给出, 所以我们求J的一阶变分. 对于式(4.5.40),

$$\delta J = \int \left[\frac{\partial F}{\partial y}\delta y + \sum_{i=1}^{n} \frac{\partial F}{\partial\left(\frac{\partial y}{\partial x_i}\right)} \delta\left(\frac{\partial y}{\partial x_i}\right) \right] dx_1 dx_2 \cdots dx_n \tag{4.5.43}$$

若取δy在积分区域的边界上固定为0, 对上式分部积分得到

$$\delta J = \int \left[\frac{\partial F}{\partial y} - \sum_{i=1}^{n} \frac{\partial F}{\partial\left(\frac{\partial y}{\partial x_i}\right)} \right] \delta y dx_1 dx_2 \cdots dx_n \tag{4.5.44}$$

如果y使得J取极值, 则此y必使$\delta J = 0$, δy除要求在边界上为0 外是任意的, 因而必有

$$\frac{\partial F}{\partial y} - \sum_{i=1}^{n} \frac{\partial}{\partial x_i}\left(\frac{\partial F}{\partial\left(\frac{\partial y}{\partial x_i}\right)} \right) = 0 \tag{4.5.45}$$

这一方程称为Euler 方程, 一般是一偏微分方程. 这样, 泛函极值问题就变为微分方程的求解问题. 这一过程也可以反过来进行, 如果要求某一微分方程的解, 而这一微分方程是某一泛函的Euler 方程, 我们就可以把微分方程的求解问题转化为泛函的极值问题.

对于本节开始处的问题, 其Euler 方程为

$$-\frac{\sqrt{1+y'^2}}{2\sqrt{2gy}\,y} - \frac{d}{dx}\left(\frac{y'}{\sqrt{2gy(1+y'^2)}} \right) = 0$$

注意到$\frac{d}{dx} = y'\frac{d}{dy}$, 上式可改写为

$$\frac{\sqrt{1+y'^2}}{y^{3/2}} + 2y'\frac{d}{dy}\left(\frac{y'}{\sqrt{y(1+y'^2)}} \right) = 0$$

乘以$y^{3/2}(1+y'^2)^{3/2}$, 并整理得

$$\frac{dy}{2y} + \frac{y'dy'}{1+y'^2} = 0$$

其解为,

$$x = -\sqrt{2c_1 y - y^2} + c_1 \arccos\frac{c_1 - y}{c_1} + c_2$$

c_1, c_2为二积分常数, 由A, B 二点决定, 上式给出了所用时间最短的轨迹.

4.6 有限元方法的理论基础

这一节以二阶椭圆型偏微分方程为例, 讲述有限元方法的理论基础. 考虑如下的边值问题,

$$\begin{cases} \nabla^2\phi = -f \\ \phi_{|\Gamma} = 0 \end{cases} \tag{4.6.46}$$

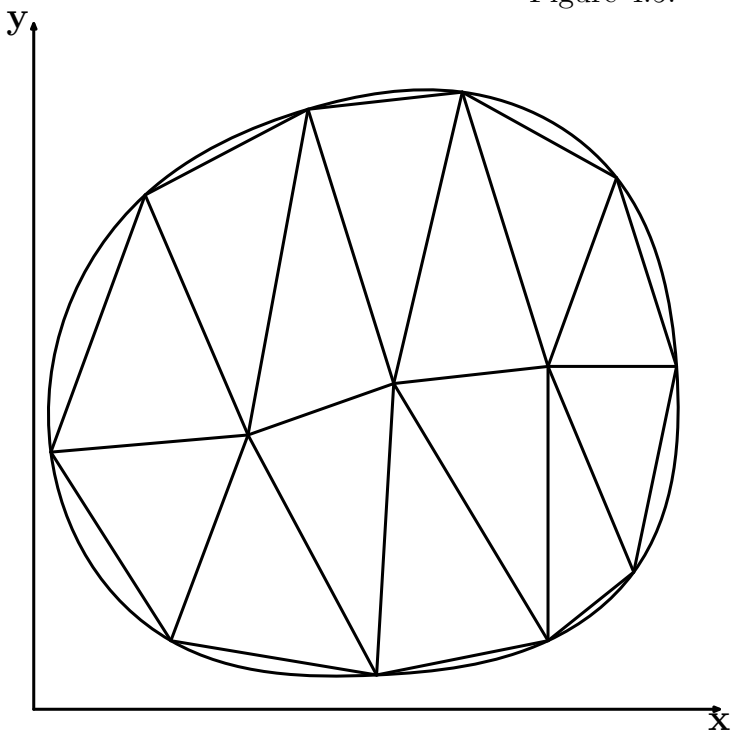
这一边值问题是泛函

$$F[\phi] = \frac{1}{2}\int_\Omega (\nabla\phi\cdot\nabla\phi - 2f\phi)\, dV \tag{4.6.47}$$

的Euler 方程. 这里Ω为边界Γ 所包围的区域. 求解边值问题(4.6.46)与(4.6.47)给出的泛函的极值问题等价. 因此, 代替求解边值问题(4.6.46), 我们可以用直接法求(4.6.47)给出的泛函的极值问题, 其结果就是原问题的解.

为了用直接法计算泛函极值, 就需要构造一组完备函数集. 有限元方法的原理就是把待求解区域分为有限个小的单元, 在每个单元上构造插值函数, 再通过总体合成得到函数集合$\{\varphi_i\}$, 把待求的函数ϕ用$\{\varphi_i\}$展开, 用函数求极值的方法求出展开系数, 得到问题的解.

Figure 4.5:



下面以二维问题为例具体说明有限元方法的基本思路, 图4.5给出了一个具体的二维求解区域及单元划分. 在图中, 我们使用了三角形划分, 这是实际计算中最常使用的分法, 但有限元方法并不限定单元的划分方法, 使用别的分法也是可以的. 不过, 下节我们将看到, 三角形分法正好可以决定线性插值函数, 简单易行. 三角形分法的三维推广是四面体划分方法. 假定我们把求解区域划分为m个单元, 现在拿出其中任一单元(假定为第e个单元)进行分析. 在这一单元上, 取一组插值函数$\{w_i(p), i = 1, 2, \cdots, k\}$, 其中$k$为单元的顶点数, 对于三角形单元, $k = 3$, p为单元中的坐标. 则单元e中的待求函数可写为

$$\phi^{(e)}(p) = \sum_{i=1}^{k} a_i w_i(p) \tag{4.6.48}$$

这里, w_i具有一定的选择自由度. 在单元的k个顶点上, 其坐标值分别为$p_1, p_2, \cdots, p_k$, 设对应的待求函数值为$\phi_1^{(e)}, \phi_2^{(e)}, \cdots, \phi_k^{(e)}$, 于是有

$$\begin{cases} \phi_1^{(e)} = \sum_{i=1}^{k} a_i w_i(p_1) \\ \phi_2^{(e)} = \sum_{i=1}^{k} a_i w_i(p_2) \\ \quad \cdots\cdots \\ \phi_k^{(e)} = \sum_{i=1}^{k} a_i w_i(p_k) \end{cases} \tag{4.6.49}$$

$w_i(p_j)$是已知的, 反解上式, 就可以把系数a_i用待求函数ϕ在单元顶点(网格节点)上的值表示出来.

$$\begin{cases} a_1 = \sum_{i=1}^{k} c_{1i} \phi_i^{(e)} \\ a_2 = \sum_{i=1}^{k} c_{2i} \phi_i^{(e)} \\ \quad \cdots\cdots \\ a_k = \sum_{i=1}^{k} c_{ki} \phi_i^{(e)} \end{cases} \tag{4.6.50}$$

再把上式代回到(4.6.48), 得到

$$\phi^{(e)}(p) = \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij} \phi_j^{(e)} w_i(p) \tag{4.6.51}$$

$$= \sum_{j=1}^{k} \left(\sum_{i=1}^{k} c_{ij} w_i(p) \right) \phi_j^{(e)} \tag{4.6.52}$$

$$= \sum_{j=1}^{k} \phi_j^{(e)} N_j^{(e)}(p) \tag{4.6.53}$$

这里, 可以把$N_j^{(e)}$, $j=1,2,\cdots,k$作为第e个单元(记为Δ_e) 上的基函数. 对于每一个单元, 都可以得到一组k个基函数, 可以把这些基函数从每个个别单元推广到整个区域上.

$$N_j^e(p)=\begin{cases} N_j^{(e)}(p) & p\in\Delta_e \\ 0 & p\notin\Delta_e \end{cases} \tag{4.6.54}$$

注意整个区域上的基函数用不带括号的上标. 在每个顶点上

$$\phi^{(e)}(p_i)=\sum_{j=1}^{k}\phi_j^{(e)}N_j^e(p_i) \tag{4.6.55}$$

因此,

$$N_j^e(p_i)=\delta_{ij} \tag{4.6.56}$$

这样, 我们对每个单元构造了基函数并推广到了整个区域上, 但一般这些基函数并不是自洽的, 也就是说, 在单元与单元之间的交接线上, 由两个单元分别构造的基函数一般并不连续. 因此, 这样的基函数并不满足作为基函数的要求(例如, 当被展开函数取这些交接线上的点时, 基函数是双值的). 为此, 我们可通过适当选择函数$w(p)$的形式使得基函数在任何相邻单元的交接线上连续, 这一过程称为总体合成. 经过合成后, 得到n个独立的基函数$N_i(p), i,=1,2,\cdots,n$, 此处n为内节点的总数. 这些基函数应满足关系

$$N_i(p_j)=\delta_{ij}$$

这样, 就可以把待求函数用这一基函数展开得到

$$\phi=\sum_{i=1}^{n}\phi_iN_i \tag{4.6.57}$$

代入方程(4.6.47) 得到

$$F[\phi]=F(\phi_1,\phi_2,\cdots,\phi_n) \tag{4.6.58}$$

令

$$\frac{\partial F}{\partial\phi_i}=0,\qquad i=1,2,\cdots,n$$

就得到一组n个确定ϕ_i的方程, 求得诸ϕ_i后代回(4.6.57)就得到了问题的解.

4.7 用有限元方法求解二维Laplace 方程

这一节通过一个具体的简单例子, 说明有限元方法计算静电场问题的过程. 我们将要计算的是求解二维Laplace方程的第一边值问题, 其方程为

$$\begin{cases} \nabla^2\phi=0 \\ \phi_{|\Gamma}=f \end{cases} \tag{4.7.59}$$

这一方程对应的变分问题为

$$F[\phi] = \frac{1}{2}\int_{\Omega}\left[\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2\right]dxdy = \text{Min} \tag{4.7.60}$$

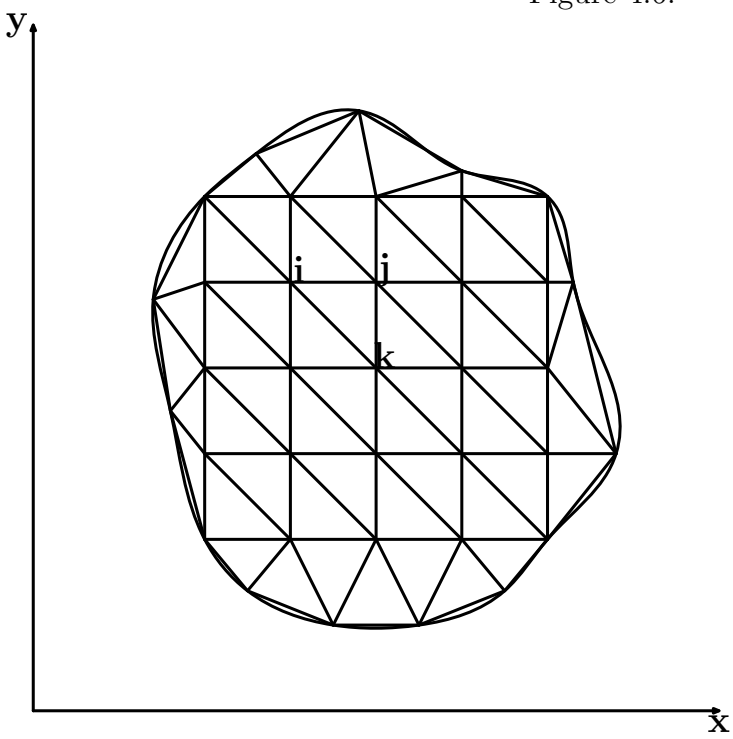
这一变分问题具有明确的物理意义, 注意到电场强度为$\mathbf{E} = -\nabla\phi$, 上式给出的恰好是电场的能量, 而变分原理在这里则表示真实的解使得电场的能量取极值(实际上是极小值).

有限元方法的计算步骤如下

一, 划分单元:

用有限元方法计算时, 先把待求区域划分为有限个互不重叠的基本单元. 现取为三角形单元. 这些单元的顶点(节点)位置的选取原则上是任意的, 一般在需要求解精度较高的区域或电势变化较剧烈的区域取得密一些, 反之则可取的疏一些. 对于大部分情况, 则可按某种规律划分, 以方便编程. 例如可先用平行坐标轴的直线族划分成四边形网格, 然后再把每个四边形对分成三角形, 如图4.6所示. 对边界上的特殊点如拐折点等都应取为节点, 弯曲的较历害的边界, 节点应取得密一点.

Figure 4.6:



所有的节点可以分为两类, 一类是内节点, 位于区域内, 其上的电势值未知待求, 另一类是边界点, 其上电势已知. 设有l_0个内节点, $l_1 - l_0$个边界点, 对节点的编号可以这样进行, 对内节点编号$1, 2, \cdots, l_0$, 对外节点编号$l_0 + 1, l_0 + 2, \cdots, l_1$. 以这些节点为顶点连成一个由不重叠的三角形组成的网. 在构成三角形时, 要避免太尖或太扁, 而且要求所有的节点都是三角形的顶点而不能是任何一个三角形的边上的点. 每个三角形称为一个单元, 第e个单元记为Δ_e, 节点间的连线称为线元. 区域边界的曲线只能以线元来近似, 图4.6中的三角形Δ_{ijk}就是一个典型的单元.

二, 构造单元上的插值函数:

下面我们来分析一个典型的单元, 为简单起见, 记三角形的三个顶点为$1,2,3$. 在单元内, 函数w可选为$\{1,x,y\}$, 于是, 单元Δ_e中的电势可表为

$$\phi^{(e)}(x,y)=\alpha_1+\alpha_2x+\alpha_3y \tag{4.7.61}$$

设单元的三个顶点$1,2,3$对应的坐标为$(x_1,y_1),(x_2,y_2),(x_3,y_3)$, 电势为$\phi_1,\phi_2,\phi_3$, 则由(4.7.61)得

$$\begin{aligned}\phi(x_1,y_1) &= \alpha_1+\alpha_2x_1+\alpha_3y_1\equiv\phi_1\\ \phi(x_2,y_2) &= \alpha_1+\alpha_2x_2+\alpha_3y_2\equiv\phi_2\\ \phi(x_3,y_3) &= \alpha_1+\alpha_2x_3+\alpha_3y_3\equiv\phi_3\end{aligned} \tag{4.7.62}$$

由上式可解出

$$\begin{aligned}\alpha_1 &= \frac{1}{2\Delta}(a_1\phi_1+a_2\phi_2+a_3\phi_3)\\ \alpha_2 &= \frac{1}{2\Delta}(b_1\phi_1+b_2\phi_2+b_3\phi_3)\\ \alpha_3 &= \frac{1}{2\Delta}(c_1\phi_1+c_2\phi_2+c_3\phi_3)\end{aligned} \tag{4.7.63}$$

其中

$$\left\{\begin{array}{l}a_1=x_2y_3-x_3y_2\\ b_1=y_2-y_3\\ c_1=x_3-x_2\end{array}\right.\quad\left\{\begin{array}{l}a_2=x_3y_1-x_1y_3\\ b_2=y_3-y_1\\ c_2=x_1-x_3\end{array}\right.\quad\left\{\begin{array}{l}a_3=x_1y_2-x_2y_1\\ b_3=y_1-y_2\\ c_3=x_2-x_1\end{array}\right. \tag{4.7.64}$$

在上式中, Δ 为单元的面积, 其值为

$$\Delta=\frac{1}{2}\left|\begin{array}{ccc}1&x_1&y_1\\1&x_2&y_2\\1&x_3&y_3\end{array}\right|=\frac{1}{2}(b_1c_2-b_2c_1) \tag{4.7.65}$$

把(4.7.64)中的α回代到(4.7.61)中, 就得到这一单元中的电势的线性插值函数为

$$\phi=\frac{1}{2\Delta}\sum_{i=1}^{3}(a_i+b_ix+c_iy)\phi_i \tag{4.7.66}$$

若定义

$$\begin{aligned}N_1^{(e)}&=\frac{1}{2\Delta}(a_1+b_1x+c_1y)\\ N_2^{(e)}&=\frac{1}{2\Delta}(a_2+b_2x+c_2y)\\ N_3^{(e)}&=\frac{1}{2\Delta}(a_3+b_3x+c_3y)\end{aligned} \tag{4.7.67}$$

则第e个单元中的电势$\phi^{(e)}$ 可写为

$$\phi^{(e)}=\sum_{i=1}^{3}\phi_iN_i^{(e)} \tag{4.7.68}$$

在写出此式时, 我们有意加上了上标e以表示是第e个单元. 对每一个单元都可做完全相同的分析. 由于我们在这里使用的是线性插值函数, 因此每个线元上的电势只决定于其两个端点的电势值, 因而自洽性要求是自动满足的.

三, 单元分析:

现在考虑第e个单元对泛函(4.7.60)的贡献. 插值函数的导数为

$$\begin{aligned}\frac{\partial\phi}{\partial x} &= \frac{1}{2\Delta}(b_1\phi_1+b_2\phi_2+b_3\phi_3)\\ \frac{\partial\phi}{\partial y} &= \frac{1}{2\Delta}(c_1\phi_1+c_2\phi_2+c_3\phi_3)\end{aligned}\tag{4.7.69}$$

上式右边为常数, 这表明在取线性插值函数时, 每一单元内的电场强度为常数, 这当然只有在单元取得很小时才成立, 同时这也为我们选取网格的大小提供了一个参考, 即网格的选取应使单元的范围比把电场强度可看作常数的范围稍小.

由(4.7.69), 可求得第e个单元对泛函(4.7.60)的贡献为

$$\begin{aligned}F^{(e)} &= \frac{1}{2}\int_{\Delta_e}\left[\left(\frac{\partial\phi}{\partial x}\right)^2+\left(\frac{\partial\phi}{\partial y}\right)^2\right]dxdy\\ &= \frac{1}{2}\sum_{p=1}^{3}\sum_{q=1}^{3}a_{pq}^{(e)}\phi_p\phi_q\end{aligned}\tag{4.7.70}$$

其中,

$$a_{pq}^{(e)}=\frac{1}{4\Delta}(b_pb_q+c_pc_q)\tag{4.7.71}$$

为了下面的应用, 我们恢复单元e的顶点的本来编号, 把$1,2,3$换为$j_1[e],j_2[e],j_3[e]$, 于是, 式(4.7.70)可写成

$$F^{(e)}=\frac{1}{2}\sum_{p=1}^{3}\sum_{q=1}^{3}a_{pq}^{(e)}\phi_{j_p[e]}\phi_{j_q[e]}\tag{4.7.72}$$

总体合成:

把所有的单元对泛函的贡献加起来, 就得到了泛函(4.7.60)的值

$$F[\phi]=\sum_{\text{all }e}F^{(e)}=\frac{1}{2}\sum_{\text{all }e}\sum_{p,q=1}^{3}a_{pq}^{(e)}\phi_{j_p[e]}\phi_{j_q[e]}\tag{4.7.73}$$

上式可以通过以节点的值ϕ_i进行归并, 整理得到

$$F[\phi]=\frac{1}{2}\sum_{i,j=1}^{l_1}K_{ij}\phi_i\phi_j\tag{4.7.74}$$

为了计算$F[\phi]$的极值, 可对上式对$\phi_i,=1,2,\cdots,l_0$求偏导数并令其为零, 得到

$$\sum_{j=1}^{l_1}K_{ij}\phi_j=0\qquad i=1,2,\cdots,l_0$$

或, 把与边界有关的已知项移到右边

$$\sum_{j=1}^{l_0} K_{ij}\phi_j = -\sum_{j=l_0+1}^{l_1} K_{ij}\phi_j \equiv u_i \tag{4.7.75}$$

写成矩阵形式, 有

$$\mathbf{K}\phi = \mathbf{u} \tag{4.7.76}$$

这是一个线性代数方程组, 可用前面介绍的方法求解, 一旦求得了诸ϕ_i, 就可计算出区域中任一点的电势.

在实际计算中, 通常要用一些特殊的方法求解有限元方程(4.7.76). 矩阵$\mathbf{K}$ 一般是一个稀疏带状距阵, 且具有正定, 对称的特点. 对于这类矩阵, 通常采用一维表示, 使内存要求大大降低. 下面简单介绍一下一维表示方法, 由于$\mathbf{K}$ 是对称的, 因此我们只需要存储对角线上的元素和上三角的元素. 在第i 列上取出最上面的非零元素, 设其位于第k_i 行, 则第i 列的带宽为

$$t_i = i - k_i, \qquad (i = 1, 2, \cdots, l_0) \tag{4.7.77}$$

显然, $t_1 = 0$, 把矩阵$\mathbf{K}$ 的上三角部分和对角部分先按行再按列排成一个一维数组, 记为$A[1:s]$, 称为带内数据数组. 再建立一个数组$N[1:l_0]$, 其第i 个元素定义为:

$$N[i] = (t_1 + 1) + (t_2 + 1) + \cdots + (t_i + 1), \quad (i = 1, 2, \cdots, l_0) \tag{4.7.78}$$

这一数组称为累计带宽数组, $N[1] = 1$, 它给出了$\mathbf{K}$ 中元素在数组A 中的位置, 例如$A[N[i]]$ 为$\mathbf{K}$ 的第i 个对角元素. 一般地

$$\mathbf{K}_{ji} = A[N[i] - i + j] \tag{4.7.79}$$

用A 和N 就可以完整的表示出$\mathbf{K}$. 如果矩阵$\mathbf{K}$ 的最大带宽($\mathrm{Max}\{t_i\}$) 为m, 则A 的元素数$s \le ml_0$, N 的元素数为l_0, 最多需$(m+1) \times l_0$ 个存储单元, 而直接存储$\mathbf{K}$ 则需$l_0 \times l_0$ 个单元, 二者在l_0 较大时相差是很大的.

对于$l_0 = 6$的一种情形示于(4.7.80), 其中方块是$\mathbf{K}$中必须存储的的非零元素，$\times$中的元素可以通过对称性得出，方块中的数字是一维存储时的编号。

$$\mathbf{K} = \begin{bmatrix} \boxed{1} & \boxed{2} & \boxed{4} & 0 & 0 & 0 \\ \times & \boxed{3} & \boxed{5} & 0 & 0 & 0 \\ \times & \times & \boxed{6} & \boxed{7} & \boxed{9} & 0 \\ \times & \times & \times & \boxed{8} & \boxed{10} & \boxed{12} \\ \times & \times & \times & \times & \boxed{11} & \boxed{13} \\ \times & \times & \times & \times & \times & \boxed{14} \end{bmatrix} \tag{4.7.80}$$

在此表示下, 可以方便地用消去法求解方程(4.7.76).

最后, 具体说明一下构成矩阵$\mathbf{K}$和矢量$\mathbf{u}$的算法

1. 对$i, j = 1, 2, \cdots, l_0$

$$\begin{cases} K_{ij} \Longleftarrow 0 \\ u_i \Longleftarrow 0 \end{cases} \tag{4.7.81}$$

2. 对所有的单元Δ_e, 令$i = j_p[e], j = j_q[e], p, q = 1, 2, 3$

   (a) 如果$i > l_0$, 不作处理;

   (b) 如果$i \le l_0$ 同时$j \le l_0$, 则$K_{ij} \Longleftarrow a_{pq}^{(e)} + K_{ij}$

   (c) 如果$i \le l_0$ 同时$j > l_0$, 则$u_i \Longleftarrow -a_{pq}^{(e)}\phi_j + u_i$

在一维表示下, 上面的算法变为

1. 对$i = 1, 2, \cdots, l_0$

$$\begin{cases} t_i \Longleftarrow 0 \\ u_i \Longleftarrow 0 \end{cases} \tag{4.7.82}$$

2. 对所有的单元Δ_e, 令$i_1 = j_1[e], i_2 = j_2[e], i_3 = j_3[e]$

   (a) 如果$i_p > l_0$, $p = 1, 2, 3$, 不作处理;

   (b) 计算t_i:

$$\begin{cases} \text{如果} i_1 \le l_0 \quad \text{则} t_{i_1} \Longleftarrow \mathrm{Max}(t_{i_1}, i_1 - i_2, i_1 - i_3) \\ \text{如果} i_2 \le l_0 \quad \text{则} t_{i_2} \Longleftarrow \mathrm{Max}(t_{i_2}, i_2 - i_3, i_2 - i_1) \\ \text{如果} i_3 \le l_0 \quad \text{则} t_{i_3} \Longleftarrow \mathrm{Max}(t_{i_3}, i_3 - i_1, i_3 - i_2) \end{cases} \tag{4.7.83}$$

3. 计算N_i:

   (a) $N_1 \Longleftarrow 1$;

   (b) 对$i = 2, 3, \cdots, l_0$, $N_i \Longleftarrow N_{i-1} + t_i + 1$

4. 对所有的单元Δ_e, 令$i = j_p[e]$, $j = j_q[e]$, $p, q = 1, 2, 3$

   (a) 如果$i > l_0$, 不作处理;

   (b) 如果$i \le l_0$ 同时$j \le l_0$, 则

      i. 如果$i > j$, 不作处理;

      ii. 如果$i \le j$,

$$\begin{aligned} \alpha &= N_j - j + i \\ K_\alpha &\Longleftarrow a_{pq}^{(e)} + K_\alpha \end{aligned}$$

   (c) 如果$i \le l_0$ 同时$j > l_0$, 则$u_i \Longleftarrow -a_{pq}^{(e)}\phi_j + u_i$

这一节比较详细地处理了有限元方法求解二维Laplace 方程的问题, 从讲解中我们可以看到, 有限元方法节点配置灵活, 特别适用于处理复杂边界的问题, 但另一方面, 有限元方法形成的**K**矩阵的形式往往比较复杂, 当节点较多时, 需要用特殊的方式储存和运算, 增加了求解的复杂性. 在实际工作中, 应学会根据具体问题选择合适的方法.

本章没有给出具体程序, 这是因为这里只是对电磁场的计算做了非常初等的介绍. 如果要作实际计算, 还必须考虑很多细节问题. 同学们在作计算时可参看有关的专著和一些较成熟的软件包.

Chapter 5

Monte Carlo方法

数值模拟方法在计算物理中具有十分重要的地位, 这一章, 我们将从理论物理的角度来介绍这一方法.

5.1 随机变量及其分布

在这一节中, 我们介绍随机变量及几个相关的问题, 随机变量(以下用ξ 表示)可分为两类, 一类是离散型随机变量, 它可以取一系列分立值$x_1, x_2, \cdots, x_n, \cdots$, 其对应的取某一值的概率为$p_1, p_2, \cdots, p_n, \cdots$. p_i称为ξ的概率分布; 另一类是连续型随机变量, ξ可连续取值, 设ξ在区间$[x, x+\delta x]$ 内取值的概率为$p(x \le \xi < x+\delta x)$, 令

$$f(x) = \lim_{\delta x \to 0} \frac{p(x \le \xi < x+\delta x)}{\delta x}$$

$f(x)$称为ξ的分布概率密度, 而ξ处于区间$[a, b]$内的概率由下式给出

$$p(a \le \xi < b) = \int_a^b f(x)\, dx$$

概率应归一化, 即

$$\sum_{i=1}^{\infty} p_i = 1 \qquad \text{对离散型随机变量}$$

$$\int_{-\infty}^{\infty} f(x)\, dx = 1 \qquad \text{对连续型随机变量}$$

定义分布函数

$$F(x) = p(\xi \le x) = \int_{-\infty}^{x} f(x)\, dx$$

代表ξ处于$[-\infty, x]$的概率. 显然, $f(x) = \frac{dF}{dx}$.

随机变量的数学期望定义为

$$E(\xi)=\sum_i x_i p_i \qquad \text{对离散型随机变量}$$

$$E(\xi)=\int_{-\infty}^{\infty} x f(x)\,dx \qquad \text{对连续型随机变量}$$

方差定义为

$$D(\xi)=\sum_i (x_i-E(\xi))^2 p_i \qquad \text{对离散型随机变量}$$

$$D(\xi)=\int_{-\infty}^{\infty} (x-E(\xi))^2 f(x)\,dx \qquad \text{对连续型随机变量}$$

下面介绍两个重要定理:

大数定理: 设$\xi_1,\xi_2,\cdots,\xi_N,\cdots$为一随机变量序列, 相互独立, 具有同样分布, 且$E(\xi_i)=a$ 存在, 则对任意小量$\epsilon>0$, 有

$$\lim_{N\to\infty} p\left\{\left|\frac{1}{N}\sum_{i=1}^{N}\xi_i-a\right|<\epsilon\right\}=1$$

这一定理指出, 不论随机变量的分布如何, 只要N足够大, 则算术平均与数学期望值可无限接近, 也就是说, 算术平均以概率收敛于其数学期望值.

中心极限定理: 设$\xi_1,\xi_2,\cdots,\xi_N,\cdots$为一随机变量序列, 相互独立, 具有同样分布, 且$E(\xi_i)=a, D(\xi_i)=\sigma^2$ 存在, 则当$N\to\infty$时,

$$p\left\{\frac{1}{N}\sum_{i=1}^{N}\xi_i-a<\frac{X_\alpha\sigma}{\sqrt{N}}\right\}\to\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{X_\alpha} e^{-x^2/2}\,dx$$

推论:

$$p\left\{\left|\frac{1}{N}\sum_{i=1}^{N}\xi_i-a\right|<\frac{X_\alpha\sigma}{\sqrt{N}}\right\}\to\frac{2}{\sqrt{2\pi}}\int_{0}^{X_\alpha} e^{-x^2/2}\,dx$$

令

$$\frac{2}{\sqrt{2\pi}}\int_{0}^{X_\alpha} e^{-x^2/2}\,dx=1-\alpha$$

则当N很大时,

$$\left|\frac{1}{N}\sum_{i=1}^{N}\xi_i-a\right|<\frac{X_\alpha\sigma}{\sqrt{N}} \tag{5.1.1}$$

成立的概率为$1-\alpha$, $1-\alpha$称为可信水平. α, $1-\alpha$和X_α的数值关系见表5.1:

由表可见, 当$X_\alpha=2.5758$时, 式(5.1.1)成立的概率已经为99%, 也就是说, 该式的可靠性已相当高.

α	0.5	0.05	0.02	0.01
$1-\alpha$	0.5	0.95	0.98	0.99
X_α	0.6745	1.9600	2.3263	2.5758

Table 5.1:

5.2 赝随机数的产生

在数值模拟中, 总是要用到随机数, 因此, 我们首先讨论如何产生随机数. 产生随机数的方法一般有两种, 一种是物理的方法, 一种是数学的方法. 我们只讨论产生随机数的数学方法. 用数学的方法, 在计算机上产生随机数, 这似乎是不可能的, 因为一个程序在运行中算得的结果完全是确定的, 不具有任何随机性. 但另一方面, 确实可以用确定的程序, 产生出可以重复的, 满足随机数的各个检验要求的数列, 为了与真实的随机数区别, 我们通常把数学方法产生的随机数称为赝随机数. 由于赝随机数序列是用确定的方法产生的, 因此它有一定的周期, 即这一序列在它的第N个数开始重复. 从实用的角度讲, 周期要尽可能的长, 这样, 在一次数值模拟中所用的随机数的个数小于其周期, 否则, 如果在一次计算中所用的随机数的数目超过了它的周期, 则可能产生虚假的结果.

最常用, 最基本的产生赝随机数的方法是同余法, 即下面的递推关系:

$$I_{j+1} = aI_j + c(\mathrm{mod}\ m) \tag{5.2.2}$$

这里, a, c, m 是三个整数, 由上述关系可推出一个序列I_1, I_2, I_3, $\cdots$, 对于适当选定的a, c, m, 这一序列可以作为赝随机数使用且具有足够长的周期. 因此, 对于大部分实际应用, 下面两句FORTRAN语句放到任何需要随机数的地方就可以满足要求

```
JRAN = MOD(JRAN*IA+IC, IM)
RAN = FLOAT(JRAN)/FLOAT(IM)
```

这里RAN 就是一个在区间[0,1]上的赝随机数. 关于系数的一个特别的选择是

$$a = 16807, \quad c = 0$$

于是，在32位机器上，下面的语句就可以产生$(1, 2^{31})$之间的随机数。

```
linux=linux*16807
if(linux.lt.0) linux=linux+2147483647+1
```

对于大型的计算项目, 需要质量较好且与机器无关的随机数发生器, 一种笔者常用的随机数发生器是Fibonacci发生器, 其新的随机数由前面的两个已经得到的随机数通过加, 减或乘得到, 一种简单的实现是:

$$x_{k+1} = x_{k-17} - x_{k-5}$$

当$x<0$时, $x\leftarrow x+1$. 这一实现称为17位Fibonacci发生器, 可以通过下面的程序段来实现, 使I和J分别初始化为17和5, 并生成17个随机数存在数组U中

```
UNI=U(I)-U(J)
IF(UNI.LT.0.0) UNI=UNI+1.0
U(I)=UNI
I=I-1
J=J-1
IF(I.EQ.0) I=17
IF(J.EQ.0) J=17
```

这一随机数发生器的周期可以达到10^{12}, 基本能满足大部分实际计算的需要. 目前广泛使用的一个赝随机数发生器是Shift-Register发生器，其程序实现称为R250，这似乎是一个质量更高的发生器。R250假定已经有了250个随机整数数，然后，第251个为

$$x_{251}=x_1\oplus x_{148}$$

或一般的

$$x_k=x_{k-250}\oplus x_{k-103}$$

这是一类随机数发生器中的一个，这类发生器均可写为

$$x_k=x_{k-p}\oplus x_{k-q}$$

这里$\oplus$表示对整数按照二进制位做xor运算。

其它的(p,q)组合有

(98, 27) (1279,216) (1279,418) (9689, 84)

(9689,471) (9689, 1836) (9689, 2444) (9689, 4187)

对于其它分布的赝随机数, 只要对上面程序产生的序列做变换就可得到. 这里做一个简短的介绍。

对于一个给定的分布$P(y)$, 我们希望由均匀分布的随机数x获得满足分布$P(y)$ 的随机数。即我们需要找到一个x的函数$f(x)$, 对于一系列在$[0,1]$ 按照均匀分布的x, 由$y=f(x)$ 得到按照$P(y)$分布的y. 注意到

$$\int_{-\infty}^{y}P(y')\,dy'\equiv F(y)=x,$$

则显然

$$y=F^{-1}(x).$$

下面是一个例子：

1,指数分布

$$P(y)=\begin{cases}0, & y<0\\ \lambda e^{-\lambda y}, & y>0\end{cases}.$$

$$F(y)=\int_{-\infty}^{y}P(y')\,dy'=\begin{cases}0, & y<0\\ 1-e^{-\lambda y}, & y>0\end{cases},$$

则

$$y=f(x)\equiv F^{-1}(x)=-\frac{1}{\lambda}\ln(1-x).$$

不幸的是，对于绝大多数分布而言，上面的积分无法解析积出，因此变换无法完成，因此，需要寻找其它方法。

5.3 用Monte Carlo 方法计算积分

在第三章中, 我们曾指出, 高维积分的工作量之大, 是通常积分方法无法完成的, 这是因为计算量对积分重数的依赖是指数关系, 即为a^N , 这里a 是一个数, 大体为计算一重积分的计算量, N为积分重数. 如果要求的精度不是很高, 则Monte Carlo 方法计算高维积分可以大大减小计算量. 由中心极限定理可知, 如果我们在一个多维体积V中均匀随机地选N个点, $\mathbf{x_1},\mathbf{x_2},\mathbf{x_3},\cdots,\mathbf{x_N}$, 则

$$\int_V f dV\approx V\langle f\rangle\pm V\sqrt{\frac{\langle f^2\rangle-\langle f\rangle^2}{N}}\tag{5.3.3}$$

这里$\langle\cdots\rangle$表示取算术平均,

$$\langle f\rangle\equiv\frac{1}{N}\sum_{i=1}^{N}f(\mathbf{x_i})\quad\langle f^2\rangle\equiv\frac{1}{N}\sum_{i=1}^{N}f^2(x_i)$$

在方程(5.3.3)中, 右边第一项是积分的近似值, 而第二项为误差估计(取$X_\alpha=1$并以$\langle f\rangle$代替数学期望值). 注意到误差项反比于$\sqrt{N}$, 正比于被积函数的方差. 因此, 为了达到较高的精度, 一种方法就是增大N, 但显然, 为了增加一位有效数字, 就要增加100倍的计算量, 下面是一个例子, 我们用Monte Carlo 方法计算积分

$$\int_0^1 dx_1\cdots\int_0^1 dx_{10}\sum_{i=1}^{10}x_i^2$$

这个积分可以解析积出来, 其结果为$10/3\approx 3.333333\cdots$, 下表是用Monte Carlo 方法计算的结果, 可以清楚地看到其缓慢收敛的过程.

N	<f>	<f>-10/3	sqrt((<f^2>-<f>^2)/N)
200	3.26346	0.698731E-01	0.659725E-01

2000	3.33163	0.170660E-02	0.209839E-01
20000	3.33331	0.267029E-04	0.667506E-02
200000	3.33352	-0.184298E-03	0.210897E-02

另一种提高精度的方法是减小方差, 为此, 考虑不在积分域内均匀取点, 而是按照某一分布取点, 用$g(\mathbf{x})$记这一分布, 则方程(5.3.3)成为

$$\int_V f dV \approx \langle f/g \rangle \pm \sqrt{\frac{\langle (f/g)^2 \rangle - \langle f/g \rangle^2}{N}} \tag{5.3.4}$$

其中$\langle \cdots \rangle$为对于按照分布$g(x)$取样的点所做的平均。如果选择

$$g = \frac{f}{\langle f/g \rangle} \tag{5.3.5}$$

则方程中的误差项为零! 但仔细观察上式就会发现, 这实际上是做不到的, 因为上式中包含有待求的项$\langle f/g \rangle$, 而在g未给出之前$\langle f/g \rangle$根本无法计算. 但另一方面，$g(x)$与被积函数成比例，如果选择一个形状非常接近于被积函数的分布，就可能减小方差。

不过, 还有一个更本质的困难, 一般我们给出的是均匀分布的随机数, 因此实现均匀抽样完全是直接了当的, 但当我们要实现一个指定分布的抽样时, 就面临把均匀分布的随机数变为所需分布的问题, 具体地说, 就是要找一变换

$$\mathbf{y} = \mathbf{y}(\mathbf{x})$$

使得对于均匀分布的$\mathbf{x}$有以g分布的$\mathbf{y}$, 这种变换的寻找也许比原积分的计算更难,(至少对绝大多数问题是如此!). 因此, 这一想法似乎是无用的. 不过, (5.3.5)毕竟给我们一些提示, 由于g 正比于f, 这就提示我们在被积函数值较大的地方多选取样点. 因此可以构造一些简单(变换容易) 而又满足上述要求的分布进行取样. 关于这一方面的内容, 需进一步了解者可参看(裴鹿成, 张孝泽著, 蒙特卡罗方法及其在粒子输运问题中的应用).

下一章我们将介绍另外一种基于这样一种观察的方法来模拟统计物理问题.

我们再来看一个例子，考虑积分

$$I = \int_0^{10} dx\, x^{68} \exp(-25x)$$

这个积分的精确值是

$$I^{\text{exact}} = 0.86416626117$$

用简单抽样法计算，结果如下

N	I	$\frac{\lvert I-I^{\text{exact}}\rvert}{I^{\text{exact}}}$	$\sqrt{N}\frac{\lvert I-I^{\text{exact}}\rvert}{I^{\text{exact}}}$	$\frac{\Delta I}{I}$	$\frac{\sqrt{N}\Delta I}{I}$
10^2	1.0757	0.245	2.44	0.235	2.35
10^3	0.83834	0.299×10^{-1}	0.945	0.874×10^{-1}	2.76
10^4	0.84711	0.197×10^{-1}	1.97	0.277×10^{-1}	2.77
10^5	0.86775	0.414×10^{-2}	1.31	0.867×10^{-2}	2.74
10^6	0.86343	0.850×10^{-3}	0.850	0.274×10^{-2}	2.75
10^7	0.86423	0.686×10^{-4}	0.217	0.868×10^{-3}	2.75
10^8	0.86380	0.420×10^{-3}	4.20	0.275×10^{-3}	2.75
10^9	0.86426	0.111×10^{-3}	3.50	0.868×10^{-4}	2.75
10^{10}	0.86420	0.383×10^{-4}	3.83	0.275×10^{-4}	2.75

其中，$\frac{\Delta I}{I}$ 是相对误差的估计，一种估算方法是直接把被积函数的抽样值作为随机变量，由此得到:

$$\frac{\Delta I}{I}=\sqrt{\frac{\langle f^2\rangle-\langle f\rangle^2}{N\langle f\rangle^2}}$$

另一种方法是把总的步数分为若干段，在每一段计算I, 并把其看做为随机变量，然后求误差。取大约10 – 100段，所得结果与第一种方法的结果大致相同。这里的结果来自第一种方法。从最后一列可以看出, 误差估值反比于$\sqrt{N}$, 其比例系数与方差相联系。

为了减少方差，我们尝试用一个与被积函数比较接近的分布来取样，取此分布为

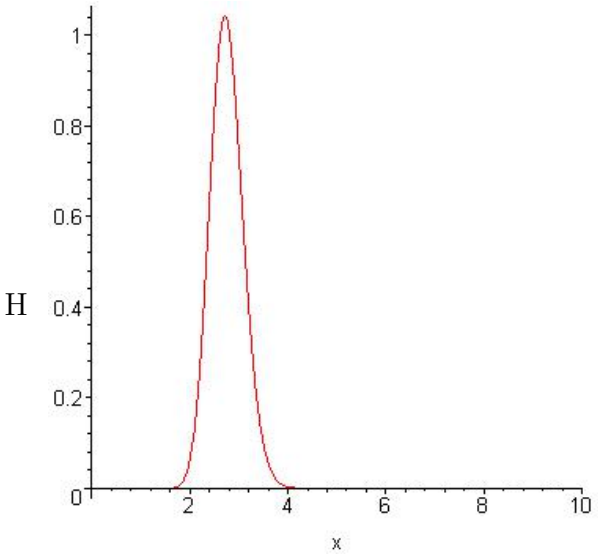
$$g(x)=0.8920620583\exp(-2.5(x-2.8)^2)$$

然后，利用这个分布取样，并计算

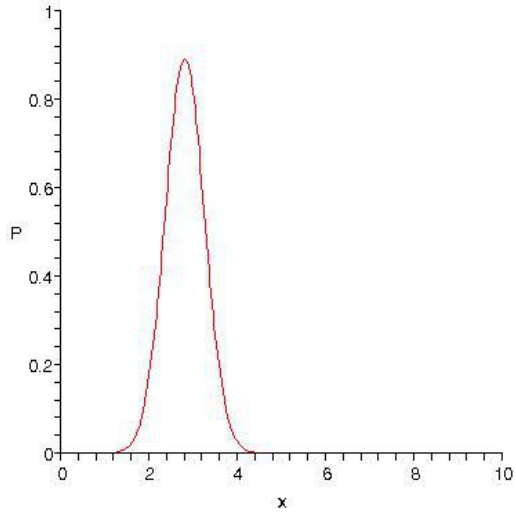
$$\frac{x^{68}\exp(-25x)}{0.8920620583\exp(-2.5(x-2.8)^2)}$$

的平均值和误差，计算结果如下。注意到最后一列比简单抽样小差不多一个数量级。

N	I	$\frac{\lvert I-I^{\text{exact}}\rvert}{I^{\text{exact}}}$	$\sqrt{N}\frac{\lvert I-I^{\text{exact}}\rvert}{I^{\text{exact}}}$	$\frac{\Delta I}{I}$	$\frac{\sqrt{N}\Delta I}{I}$
10^2	0.86741	0.376×10^{-2}	0.038	0.346×10^{-1}	0.300
10^3	0.86192	0.260×10^{-2}	0.082	0.117×10^{-1}	0.320
10^4	0.86409	0.907×10^{-4}	0.009	0.365×10^{-2}	0.315
10^5	0.86395	0.247×10^{-3}	0.078	0.116×10^{-2}	0.318
10^6	0.86460	0.505×10^{-3}	0.505	0.367×10^{-3}	0.317
10^7	0.86422	0.617×10^{-4}	0.195	0.116×10^{-3}	0.317
10^8	0.86417	0.154×10^{-5}	0.015	0.367×10^{-4}	0.317
10^9	0.86418	0.170×10^{-4}	0.538	0.116×10^{-4}	0.317
10^{10}	0.86416	0.315×10^{-5}	0.315	0.367×10^{-5}	0.317



$f(x)$



分布$g(x)$

为了实现对于$g(x)$的抽样，我们利用如下Metropolis算法，关于这一算法的解释将在下一章给出。算法如下：

1. 任取$x_0 \in [0, 10]$。
2. 设已经得到x_i, 选$y = x_i + \delta\,(ran() - 0.5)$. 其中$ran()$给出$(0, 1)$之间均匀分布的随机数，$\delta$为一可调参数。
3. 如果$y < 0$或$y > 10$, 则令$x_{i+1} = x_i$. 否则，计算$g(y)$, 产生随机数$\xi = ran()$, 如果$g(y)/g(x_i) \geq \xi$, 则$x_{i+1} = y$, 否则$x_{i+1} = x_i$.

这样得到的序列x_0, x_1, ..., x_N 满足分布$g(x)$. 建议写一个实现上述算法的小程序，产生一个序列，做出其分布的直方图并检验上述论断。

5.4 自相似性与分形

在本章的其余几节我们将介绍几种与随机数和模拟有关的物理过程. 这一节先介绍分形的概念. 对于任一物体, 其形状本身就是一个很有趣的性质, 我们日常生活中经常遇到的如圆形, 椭圆, 球, 长方体等都代表某种形状. 与这些形状相关的是其维数, 如我们知道圆面, 椭圆面, 长方形面都是二维的而球, 长方体等则是三维的. 从数学上看, 如果我们把一个二维物体的线度增加倍, 则其面积将变为原来的四倍, 即

$$S \sim R^2.$$

这里S为所研究物体的面积而R是其特征长度. 对于三维物体, 我们有,

$$V \sim R^3.$$

作为一个一般的关系, 我们考虑所研究的物体具有均匀面密度或体密度, 于是有

$$M \sim R^d$$

这里d为所研究物体的维数.

近年来, 人们对于另外一类形状发生了很大的兴趣, 这类物体的形状通常都非常复杂, 自然界如雪花, 海岸等。数学上，利用简单的规则，可以生成很多种复杂图形。物理上如扩散限制粘结的图样等。这样一类形状被称为分形. 我们以一个直观的简单例子开始讨论, 考虑一个正方形, 显然, 它的维数是二. 现把它分为9个相等的小正方形, 并把中心的小正方形拿掉, 对剩下的8个小正方形重复这一过程直到无穷, 这样得到的形状成为Sierpinski地毯. Sierpinski地毯是一种确定性分形, 这是因为产生Sierpinski地毯的规则完全是确定的, 后面我们还会看到随机分形的例子, 这类分形是通过一定的随机过程而生成的. 分形的一个显著特点是自相似性, 如果我们从不同的空间尺度去观察分形, 看到的形状特点都是一样的, 或者, 如果我们仅仅只看分形, 则无法知道所看到的部分是否被放大了, 放大了多少倍. Sierpinski地毯就是这样一个典型的例子. 标志分形的关键量是它的维数, 对于规则形状, 直观上很容易给出其维数, 这个维数实际上就是可以把此形状嵌入的最小空间维数, 如一个平面物体可以嵌入二维空间, 是二维的, 一个空间物体如球, 立方体等可以嵌入三维空间, 是三维的. 而分形的另一个特点是很难从直观上给出其维数, 再以Sierpinski地毯为例, 它的维数是多少? 与前一段的分析类似, 我们可以一般地用下述公式

$$M \sim R^{d_f}$$

定义一个物体的维数. 对于大多数复杂形状, d_f通常并不是一个整数, 而是一个分数或无理数, 这就是分形这一名称的来源.

对于一个d_f维的形状, 由上面的定义可知, 如果我们用某个长度单位u来量度其特征长度, 则面积的单位为u^2, 体积的单位为u^3. 一个物体的大小可以用能容纳该物体的最小正规维数的空间来量度, 如前面的Sierponski地毯可容纳于二维空间, 因此可以用u^2来量度其大小, 所谓大小, 在这一例子中是指至少用多少个单位面积可以完全覆盖物体. 设在此选定单位下特征长度的值为L, 或有L个长度单位. 如果我们把所用的长度单位减少到原来的$1/l$, 则显然L变为原来的l倍, 而面积和体积分别变为原来的l^2及l^3倍. 或d维形状的大小变为原来的$N = l^d$倍. 如果一个维数为d_f的形状嵌入d维空间, 此处d称为正规维数, 只能取整数值. 则此形状在单位变换下其大小变为原来的$N = l^{d_f}$倍. 于是维数

$$d_f = \frac{\ln N}{\ln l}.$$

我们可以用这样一种关系来计算分形的维数. 对于Sierpinski地毯, 设我们取其边长为单位, 则其大小为1, 把长度单位缩小到原来的1/3, 大小成为8, 再缩小到1/3, 大小成

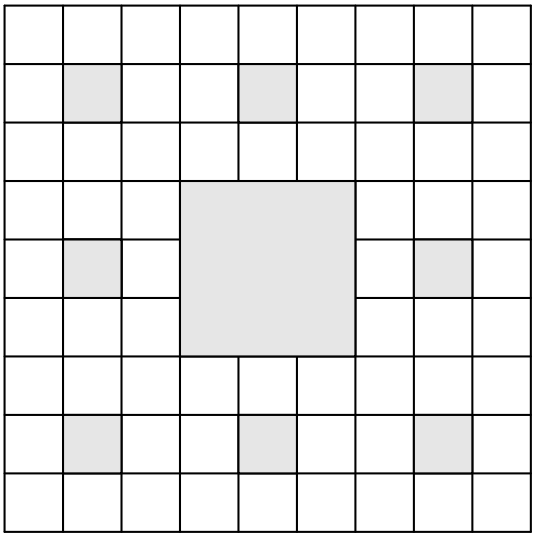


Figure 5.1: Sierpinski地毯.

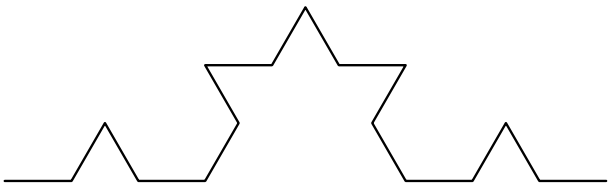


Figure 5.2: Koch 曲线.

为8^2, 或一般地, 长度单位每缩小到原来1/3, 则大小增加到8倍. 这里$N = 8$, $l = 3$, 于是Sierpinski 地毯的分形维数为$d_f = \ln 8/\ln 3 = 1.89278\cdots$.

与Sierpinski地毯有关的还有Koch曲线, 它是把一单位长度的线段三等分, 把中间的一段拿掉, 连上由长度为1/3的线段构成的上翘三角形, 然后对每一线段重复上述过程至无穷, 得到的结果即为Koch曲线. 如图所示.

练习*:* 计算*Koch*曲线的分形维数*.(*提示*,* 考察*Koch*曲线的总长度与长度单位改变之间的关系*).*

在自然界中, 一个与分形有关的著名例子是英国的海岸线有多长? 如果我们拿一张地图, 分别用不同长度的尺子来量度英国的海岸线, 则可以得到不同的长度, 随着尺子不断变短, 则我们的到的海岸线的长度变长. 一个自然的想法是当尺子足够短时, 这一长度应该趋于一个恒定值, 并可把这一恒定值定义为英国海岸线的长度. 然而, 这一直观的想法并不成立, 事实上, 当我们的尺子越来越短时, 我们可能达到一张地图分辨率的极限, 此时我们需要一张分辨率更高的地图, 但我们并不能得到一个确定的长度, 最后, 也许我们需要沿着英国海岸, 在距离水边一米处测量, 即便如此, 随着尺子长度的改变, 我们依然不能得到确定的海岸线长度. 事实上, 海岸线的长度并不是一个一维量, 而是一个分形, 它的维数大于一. 为了计算其维数, 我们可以用不同长度的尺子测量海岸线的长度, 并把

测得的长度的数值的对数与尺子长度的对数作图, 则可得到一条直线, 直线的斜率就给出海岸线的维数.

练习*:* 找一张较大的中国地图, 试测量并计算中国海岸线的分形维数.

5.5 扩散限制粘结的计算机模拟

扩散限制粘结(Diffusion limited aggregation), 简称DLA, 是近三十年来引起广泛兴趣, 得到大量研究的物理过程。其开端是由Witten和Sander为了充分使用一台当时最新的绘图仪而引发，且一发而不可收拾。(Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon, Phys. Rev. Lett. 47, 1400–1403 (1981)) 全面介绍这样一个重要的领域大大超出了本讲义的范围, 这一节我们介绍一种用计算机生成DLA集团的方法, 同学门可由此体会一点DLA的初步知识.

考虑一个正方格子, 现在格子的中心放一个粒子, 作为DLA的种子. 从格子的边缘随机选定一个出发点, 并使一粒子从该点出发, 在格子上做无规行走, 当此粒子到达作为种子的近邻时, 则使其停下, 并随机在格子边缘选一出发点, 继续这一过程; 如果粒子在无规行走中到达边界, 则放弃这一粒子. 重复这一过程多次直到生成一个较大的集团, 如果我们把这个集团显示到计算机显示器上或打印出来, 我们就会看到这一图案与很多自然现象有相似之处, 如雪花, 闪电的图样, 山上岩石的裂纹等.

DLA图样是随机分形的典型例子, 为了计算其分形维数, 我们可以考虑以种子所在的点为圆心, 不断改变半径并数出圆内所具有的粒子数M, 则应有

$$M \sim R^{d_f},$$

作出$\ln M$与$\ln R$ 的图, 就可得到分形维数.

计算机实习题*:* 编写生成*DLA*的程序并生成*DLA*集团*,* 分析其分形维数.

5.6 高分子的模拟和无规行走

无规行走可以作为高分子链的简单模型，为了简单直观起见，这一节我们只考虑二维空间的无规行走，推广到三维是直接了当的。

简单无规行走(RW)

考虑一个二维的正方格子（其它格子亦可），从格子上的一个格点出发，随机地在四个可能的方向上选择一个方向，前进一步。然后以新的位置为出发点，重复前进，走N步后，其所走的轨迹构成一个一维链。这个链可以看作最简单的高分子链的模型。因为每一步有$z = 4$种等价的选择，而相邻的步是独立的，因此，无规行走的配分函数

（总的状态数）可以方便写出

$$Q = z^N$$

这里，z是格点的最近邻格点数，对于正方格子，$z = 4$. 用$\mathbf{r}_i$表示第i步的位移，N步后的位移是

$$\mathbf{R} = \sum_{i=1}^{N} \mathbf{r}_i$$

$R = |\mathbf{R}|$是起点与终点之间的距离，称为末端距。显然

$$\langle \mathbf{R} \rangle = 0$$

而末端距的平方的平均为

$$\langle R^2 \rangle = \sum_{i,j=1}^{N} \langle \mathbf{r}_i \cdot \mathbf{r}_j \rangle = \sum_{i=1}^{N} \langle r_i^2 \rangle = Na^2$$

这里a是晶格常数。随后将会看到，这个严格的结果在模拟计算中非常有用。

不回头无规行走(NRRW)

与简单无规行走类似，除了每一步不容许立即回头，即只能在$z-1$个可能的方向中选择。对于N步行走，其配分函数为

$$Q = (z-1)^N$$

而末端距平方的平均值是

$$\langle R^2 \rangle = \frac{4}{3} Na^2$$

自回避无规行走(SAW) N 如果考虑到高分子链具有体积效应，即每个格点最多只能访问一次，那么，在无规行走中，如果遇到已经访问过的点，就必须回避。一种实际的计算方法是，产生NRRW链，一旦选择的新格点被占据，就停止。显然，这样一种方法很难得到N比较大的链，事实上，在长度为N的NRRW链中，自回避链所占的比例由其配分函数的比值给出

$$w_N = \frac{Q^{\mathrm{SAW}}}{Q^{\mathrm{NRRW}}}$$

这也是计算SAW配分函数的一个方法。事实上，SAW的配分函数无法解析得到，在$N \to \infty$时，可以得到其渐进形式为

$$Q^{\mathrm{SAW}} = N^{\gamma-1} z_{\mathrm{eff}}^N$$

其中，γ是一个临界指数，$z_{\mathrm{eff}} < z-1$为有效近邻数，通常不是一个整数。这连个数一般只能通过数值模拟求得，目前已知的结果是：三维空间$\gamma \approx \frac{7}{6}$，二维空间$\gamma \approx \frac{4}{3}$. 利用这个结果，可以得到

$$w_N = N^{\gamma-1} \left(\frac{z_{\mathrm{eff}}}{z-1} \right)^N = \exp\left(-N \ln \frac{z-1}{z_{\mathrm{eff}}} + (\gamma-1) \ln N \right)$$

从上式可以看出，随着N增加，SAW链出现的概率指数衰减，因此，当N较大，例如达到100时，从NRRW中挑出一个SAW是非常难的。所以，直接生成的方法通常限于$N < 100$的SAW链的研究。

除了自回避外，如果在最近邻的占据格点上加上一点吸引相互作用，则会得到非常有趣的结果。此时，SAW链存在一个相变，其来源是自回避排斥作用，熵和吸引相互作用的竞争。存在一个临界温度T_c, 当$T > T_c$时，SAW的末端距平方的平均值在大N极限下为

$$\langle R^2 \rangle \sim N^{2\nu}, \quad \nu \approx 0.59$$

这里ν是另外一个临界指数。当$T < T_c$时，

$$\langle R^2 \rangle \sim N^{2/3}$$

而当$T = T_c$时

$$\langle R^2 \rangle \sim N$$

Chapter 6

逾渗和统计物理问题

6.1 逾渗简介

逾渗问题本身不是一个物理问题, 但这一问题的研究包含了现代物理的很多重要概念, 如相变, 临界指数, 重整化群等. 由于逾渗问题比实际统计物理问题简单很多, 因此成为介绍这些重要物理概念的一个合适的入门模型. 逾渗的很多概念和模型在无序固体及软物质的研究中也有重要应用. 这一节我们简单介绍点逾渗模型及一些概念, 下一节给出一种计算机分析逾渗的方法.

为简单起见, 我们考虑二维问题, 把平面划分为正方格子, 如图所示. 每一个小的方格可以处于"占据"和"空"两种状态, 在图中以灰和白表示. 设每一个格点被占据的概率为p, 这一概率与格点的周围环境无关. 这样一种模型称为点逾渗模型. 对于给定的占据概率p, 每一占据格点或者是孤立的, 或者与最近邻占据格点形成集团. 我们定义若二格点为最近邻且均被占据, 则此二格点直接相连, 若二格点之间可以找到一个通路, 沿通路最近邻格点相连, 则此二格点间接相连. 集团定义为这样一组占据格点, 其中的任何二个都直接或间接相连. 集团的大小是逾渗问题中的一个关键量. 设想若每个占据格点为导体而空格点为绝缘体, 则如果存在一个大集团从格子的一边延伸到另一边, 则逾渗样品将导通, 否则为绝缘体.

我们可以很容易用计算机生成逾渗集团, 对于每一个格点, 产生一个在区间$[0,1]$上均匀分布的随机数r, 若$p > r$, 则此格点标记为占据, 否则为空. 如果占据概率p很小, 我们期望只有一些孤立的小集团, 如图A所示, 如果占据概率$p \sim 1$, 我们期望大多数占据格点形成一个跨越边界的大集团, 我们称此集团为跨越集团. 对于中等大小的p, 如p处于0.4—0.7之间, 会发生什么情况? 下面我们将看到, 对于一个无穷大的格点, 存在一个确定的临界概率p_c, 当$p > p_c$时, 存在一个无穷大的跨越集团; 当$p < p_c$时, 所有集团都是有限的, 不存在跨越集团.

逾渗的本征量是其连通性, 随着格点占据概率的连续变化, 连通性在一个特定值p_c发

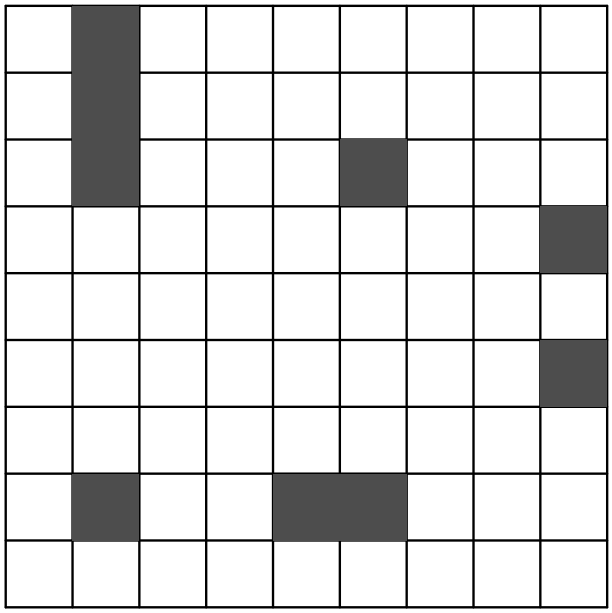


Figure 6.1:

生突变. 这种由无跨越集团到有跨越集团的变化是一种相变, 现在一般称为几何相变.

逾渗并不仅仅是一个有趣的模型, 实际上有很多与之相关的物理问题. 一个有实际意义的问题是随机复合介质问题, 如果我们把金属球和橡胶球均匀的随机放入一个容器中, 如果金属球没有形成通路, 则样品整体表现出绝缘性, 而当金属球形成通路时, 则样品在整体上表现为导体. 我们用金属球在容器中所占的体积与容器的体积之比, 体积分数ϕ来表示金属球的多少, 当ϕ很小时, 样品的电导为0, 在一个临界体积分数ϕ_c, 电导变为非零并随ϕ的增加而急剧增加. 这是一个金属绝缘体转变的典型例子. 其他的应用还有如磁性物质的非磁性掺杂, 森林火灾的扩展, 流行病的扩展等等.

为了定量研究逾渗问题, 我们定义下面一些量. 集团尺寸s, 定义为集团内的格点数; 平均集团尺寸分布$n_s(p)$, 定义为尺寸为s的集团的数目(当$p > p_c$时, 我们排除跨越集团); 显然, sn_s为尺寸为s的集团所占据的格点的总数, 而

$$w_s = \frac{sn_s}{\sum_s sn_s}$$

为随机抽取一个占据格点时, 该格点处于某一尺寸为s的集团的概率, 平均集团尺寸则由下式给出

$$S = \sum_s sw_s = \frac{\sum_s s^2 n_s}{\sum_s sn_s}.$$

另一个描述逾渗问题的重要量是跨越集团概率P_∞, 定义为跨越集团内包含的格点总数与占据格点总数之比. 对于一个无穷大的格点, 当$p < p_c$时, $P_\infty = 0$ 而当$p = 1$时, $P_\infty = 1$, 在$p = p_c$, P_∞由零变为非零且随p增加而增加.

在《热力学统计物理》中我们学过热力学相变和临界现象, 一个比较简单但具有代表

性的例子是铁磁系统的临界现象, 当温度T小于临界温度T_c时, 系统为铁磁体, 具有自发磁化, 而当$T > T_c$时, 系统为顺磁体. 系统的相变由参数温度引起, 在临界温度附近, 系统具有长程关联, 关联长度在临界点发散, 其宏观表现为一系列热力学量在临界点的非解析性质. 逾渗相变与热力学临界现象有很多相似之处, 下面我们给出一些定义和分析, 这些分析对理解更为复杂和丰富多采的热力学临界现象可提供帮助.

首先定义一个与逾渗有关的长度, 即平均连接长度$\xi(p)$, 代表集团的空间尺寸. 这一长度可以有多种定义方式, 这里我们考虑一个大小为s的集团的回转半径R_s, 定义为

$$R_s^2 = \frac{1}{s}\sum_{i=1}^{s}(\mathbf{r}_i - \overline{\mathbf{r}})^2$$

其中

$$\overline{\mathbf{r}} = \frac{1}{s}\sum_{i=1}^{s}\mathbf{r}_i$$

而$\mathbf{r}_i$是第i个格点的位置矢量. $\overline{\mathbf{r}}$是集团的质心. 我们可以把非跨越集团回转半径的平均值定义为ξ, 也可把最大的非跨越集团的回转半径定义为ξ. 当$p < p_c$时, ξ随p的增加而增加, 当$p > p_c$时, ξ随p的减小而增加, 在$p = p_c$处ξ发散. 当$|p - p_c| \ll 1$时, 我们称系统处于临界区域, ξ的发散行为是非解析的, 满足如下的指数关系

$$\xi(p) \sim |p - p_c|^{-\nu}$$

ν是我们遇到的第一个重要的数, 称为关联长度的临界指数. 在临界区域, 我们期望当$p > p_c$时, 有

$$P_\infty \sim (p - p_c)^\beta$$

在临界区域, 集团的尺寸发散, 我们期望其行为是

$$S(p) \sim |p - p_c|^{-\gamma}$$

这样我们定义了三个临界指数, 对于二维问题, 这些临界指数可以解析求出, 其值为$\beta = 5/36$, $\gamma = 43/18$, $\nu = 4/3$. 对于三维问题, 只能用各种近似方法来得到, 其结果为$\beta \approx 0.4$, $\gamma \approx 1.8$, $\nu \approx 0.9$. 这些指数具有很大的普适性, 与晶格结构以及与关联长度的不同定义均无关. 除此之外, 它们之间还满足一定的关系, 如

$$2\beta + \gamma = \nu d$$

式中d为空间维数, 这种关系称为标度关系. 有关这些关系及其在相变和临界现象中的重要意义将在《高等统计物理》课程中详细讨论. 有兴趣的同学也可参看有关书籍和文献. 相对于临界指数, 临界点p_c则不是一个普适量, 对于二维正方格点, $p_c \approx 0.5927\cdots$而对于三角格子$p_c = 1/2$.

6.2 逾渗的计算机模拟和分析

上一节我们简单介绍了逾渗的概念, 这一节将讨论用计算机研究逾渗问题的一些方法. 研究逾渗问题大体上分为三步, 第一步是生成逾渗集团, 这是最简单的一步, 已经在上节介绍过了; 第二步是标定集团, 即找出每个集团所包含的格点, 并对集团编号, 以便进一步分析, 这是整个分析中最困难的一步; 第三步是根据第二步的结果, 具体计算临界概率, 临界指数等.

一种直观的标定集团的算法是从一个占据格点出发, 找出与此格点直接和间接相连的所有格点, 并给这些格点赋予数1; 然后从剩下的占据格点中选出一个, 找出所有与这一格点相连的格点并赋予数2; 重复这一过程直到所有占据过格点全部用完. 请同学试写出这一算法的程序并对从小到大的格点进行试算, 看会有什么结果? 实际上, 随着格点数目的增加, 计算量将指数增加, 从而使计算实际上成为不可能. 下面我们介绍一种由Hoshen和Kopelman发展的算法. 这一算法可通过下面的例子来介绍. 考虑图6.2所示的一个逾渗样本, 我们从左下角开始, 并从左到右, 从下到上进行编号. 因格点(1,1)被占据, 我们给它赋予集团标号1, 下一个没有占据, 无需赋值, 下一个占据格点是(3,1), 由于其左边的格点为空, 我们给它赋予下一个集团标号2, 格点(4,1)左边占据, 与其左边的格点同属一个集团, 也赋予2, 这一行其余格点的分析是直接了当的. 然后我们到第二行,考察格点(1,2), 由于这一格点被占据, 它的上一行的近邻(1,1)的集团标号为1, 因此我们对它赋值1, 并从左到右检查这一行, 如果一个格点为空, 则跳过该格点, 如果格点被占据, 则考察其左边和下边的最近邻, 若此2近邻均空, 则给此格点赋予下一个集团标号, 如果只有一个近邻被占据, 则给此格点赋予其近邻的集团标号, 如格点(2,2)赋予1, 因为只有其左边的近邻被占据且具有标号1. 如果格点的左和下两个近邻均被占据, 则意味着原标定为不同标号的两个集团实际上是一个集团, 我们必需进行合并并重新编号. 如格点(6,2), 其左边格点(5,2)标号为4, 而其下边格点(6,1)标号为3, 显然我们应该对(6,2)赋予较小的标号3并把(5,2)的标号更新为3, 但是在后面可能还会有进一步的重复和合并, 我们把这一更新过程推迟到后面再进行. 为了记录这种合并信息, 我们引入一个正规标定数组n, 数组的每一个元素与一个集团标号相联系, 对独立的集团标号, 数组的对应元素赋值为该数组的下标, 否则赋值对应将要合并的集团的标号. 对于本例, 在到达格点(6,2)前, 分别有

$$np(1) = 1, \qquad np(2) = 2, \qquad np(3) = 3, \qquad np(4) = 4.$$

而到达格点(6,2), 我们知道标号为4的集团实际上与标号为3的集团是同一集团, 为了记住这一信息, 我们有$np(4) = 3$. 这一赋值告诉我们标号4是非正规的, 而且与正规标号3相连.

这样一个过程仍然有不确定性, 当我们到达格点(5,4), 此格点的左边标号为5, 下边标号为4, 我们已经知道标号4为非正规标号(因为$np(4) \neq 4$), 此时我们并不是按照上面的

7	7					
7				8	3	3
	6	3			3	3
6	6	5	5	3	3	3
		5		4	3	3
1	1			4	3	3
1		2	2		3	

7	7					
7				3	3	3
	3	3			3	3
3	3	3	3	3	3	3
		3		3	3	3
1	1			3	3	3
1		2	2		3	

Figure 6.2: 集团的构造算法

规则为(5,4)标号4, 而是标号为较小标号所指的正规标号, 这里为3(因为$np(4) = 3$), 同时我们知道标号5也是非正规的, 其所对应的正规编号为3, 故应令$np(5) = 3$. 当检查完所有格点后, 我们可以根据数组np的值对非正规标号进行更新.

实际计算时, 我们总是分析有限的格点, 对每一个给定大小的格点和在给定p后生成的逾渗样本, 计算得到的P_∞一般并不相同, 因此必需对多个样本进行平均. 这样计算得到的结果与所取样本大小有关, 为了得到无穷大格点的结果, 应对不同大小的格点进行计算, 然后外推到无穷大样本. 为了计算临界指数, 我们可在p_c附近对不同的p计算P_∞, 设$P_\infty \sim (p - p_c)^\beta$, 则

$$\ln P_\infty = \beta \ln(p - p_c) + C$$

这里C为一个常数, 通过调节p_c, 使得$\ln P_\infty$ 与$\ln(p - p_c)$ 成为一条直线, 则直线的斜率就给出β.

作为一个计算实习题, 请同学根据前面二节的内容分析二维正方格子上的逾渗问题, 试通过计算P_∞, ξ等量来得到临界概率p_c及临界指数.

6.3 正则系综和统计模型

在《热力学与统计物理》中我们学过, 如果一个多体系统的Hamiltonian 为H, 则对于给定系统的粒子数N, 体积V 和温度T 之后, 系统的平衡分布由正则分布给出为:

$$P(\mathbf{x}) = \frac{1}{Z} \exp[-\beta H(\mathbf{x})] \tag{6.3.1}$$

式中$\mathbf{x}$ 代表系统的任一位形, 而$P(\mathbf{x})$则给出在平衡态时系统处于该位形的概率.

$$Z = \int d\mathbf{x} \exp(-\beta H(\mathbf{x})) \tag{6.3.2}$$

为系统的正则配分函数(与标准定义差一因子$N!$, 这在Monte Carlo 计算中并不重要, 因为后面将会看到, Monte Carlo 方法几乎不用来计算配分函数本身) 在分立变量的情况下, 式(6.3.2)中的积分应理解为对各种位形的求和. 物理量是位形的函数, 对于物理量$b(\mathbf{x})$, 其平均值B 为

$$B = \int d\mathbf{x} b(\mathbf{x}) P(\mathbf{x}) \tag{6.3.3}$$

下面我们看几个具体的例子.

6.3.1 Ising 模型

Ising 模型的Hamiltonian 为

$$H = -J \sum_{nn} s_i s_j - h \sum_i s_i \tag{6.3.4}$$

式中J称为交换积分, h为外场, s_i 可取值± 1, 称为自旋变量. Ising 模型是最简单的非平庸统计物理模型, 它是由德国物理学家Lenz 在二十年代提出的, 这一模型可用来描述单轴各向异性磁性系统, 合金等物理体系, 同时也是一个十分有兴趣的理论模型. Ising 最早给出了这一模型在一维情况下的严格解, 证明了在一维下这一模型不存在相变. Onsager 于1945 年做出了零场下这一模型在二维空间的严格解并计算了它的相变温度, 比热在相变点的行为等热力学量. 杨振宁在1952 年解出了外场很小时二维空间的Ising 模型, 求出了序参量的临界行为. 由于对这一模型的很多形为目前了解的比较透彻, 因此它经常被用来做为检验各种数值方法或解析近似方法的标准.

1969年，A. E. Ferdinand和M. E. Fisher（Bounded and Inhomogeneous Ising Models. I. Specific-Heat Anomaly of a Finite Lattice，Phys. Rev. 185, 832–846 (1969) ）求得了有限尺寸2D Ising 模型在周期性边界条件下的严格解，成为检验模拟结果的一个有效标准。

对于Ising模型, 人们通常感兴趣的热力学量是能量$E = \langle H \rangle$, 序参量$m = \frac{1}{N} \sum_i s_i$, 能量的涨落$\langle H^2 \rangle - \langle H \rangle^2$, 序参量的涨落$\langle m^2 \rangle - \langle m \rangle^2$等. 能量的涨落与系统的比热成正比, 而序参量的涨落则正比于系统的磁化率.

与Ising 模型类似的还有Heisenberg 模型:

$$H = -\sum_{\langle ij \rangle} (J_x S_x^i S_x^j + J_y S_y^i S_y^j + J_z S_z^i S_z^j) \tag{6.3.5}$$

式中, J_x, J_y, J_z 称为交换积分, $\mathbf{S} = (S_x, S_y, S_z)$为自旋矢量. $\mathbf{S}$ 可为量子算符, 也可取为经典量, 分别对应于量子Heisenberg 模型和经典Heisenberg模型. 在这里, 我们只讨论经典问题.

若$J_x = J_y = J_z \equiv J$, 则上式成为各向同性的Heisenberg 模型:

$$H = -J \sum_{\langle ij \rangle} \mathbf{S}^i \cdot \mathbf{S}^j \tag{6.3.6}$$

通过适当调整J, 式中的$\mathbf{S}$可以看作三维空间的单位矢量. 如果把矢量$\mathbf{S}$的维数降为二, 我们得到所谓的XY模型; 当矢量$\mathbf{S}$的维数趋于无穷大时, 对应于球模型, 而当矢量$\mathbf{S}$的维数趋于零时, 可用来描写高分子溶液.

在Ising 模型中, 如果s的取值不是± 1, 而可以取多个值, 则对应于所谓Potts 模型.

6.3.2 Lenard-Jones 模型

在高温下, 量子效应是不重要的, 可以用经典统计物理来描述. 最简单的多粒子系统是惰性元素组成的系统, 除了氦之外, 其它惰性元素组成的系统在变成固体前的整个温度范围内都可用经典方法来描述. 惰性元素的电子是满壳层的, 其相互作用由两部分组成, 一部分是由原子的瞬时偶极矩导致的与距离如6次方成正比的吸引相互作用, 另一部分是来源于量子效应的强排斥芯. 其Hamiltonian 可写为

$$H = \frac{1}{2m} \sum_{i=1}^{N} p_i^2 + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} u(r_{ij}) \tag{6.3.7}$$

式中第一项为动能项, 第二项是相互作用, N 为系统的粒子数. $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ 为第i个粒子与第j个粒子之间的距离.

$$u(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

为Lennard-Jones 相互作用, 其中$-\epsilon$为势的极小值, σ 是长度的单位.

在粒子问题中感兴趣的热力学量通常是能量$E = \langle H \rangle$, 能量的涨落$\langle H^2 \rangle - \langle H \rangle^2$, 以及对相关函数$g(r)$, 很多热力学量如压强, 化学势等均可用对相关函数$g(r)$ 来表示. $g(r)$定义为$\delta(\mathbf{r} - \mathbf{r}_{ij})$的平均值. 对于相互作用各向同性的系统, 这一平均值只与$\mathbf{r}$的大小有关.

实际的粒子系统的相互作用常常十分复杂, 相互作用不仅与距离有关, 一般也与方向有关, 除了二体相互作用外, 通常还有多体相互作用.

6.4 统计物理的Monte Carlo模拟, Metroplis方法

由(6.3.3)可知, 为了计算热力学量, 我们需要计算一个多重积分(对于分立变量的系统, 是一个对各种位形的求和), 如果所考虑的系统有N个原子, 就需要计算$3N$重的积分, N 通常是一个较大的数, 例如, 为了得到可靠的结果, N一般取100—10000左右. 为了计算这样高重的积分, 除非可以解析求解, 否则Monte Carlo 方法就是唯一的选择.

从第5.3 可知, 对于(6.3.3)式的积分, 可以在积分区域内取一系列均匀分布的随机点x_1, x_2, x_3, $\cdots$, x_n, 则积分就成为

$$B = \frac{\sum_{i=1}^{n} b(\mathbf{x}_i)\exp(-\beta H(\mathbf{x}_i))}{\sum_{i=1}^{n}\exp(-\beta H(\mathbf{x}_i))} \tag{6.4.8}$$

稍一分析, 我们就可发现这种作法是很难行得通的. 由于指数因子的存在, 整个被积函数在积分区域内剧烈变化, 因而其方差是非常大的, 这将导致误差变大. 实际情况比这还要糟糕, 因为被积函数相差太大, 对于有限位数的计算机来说, 式(6.4.8)的求和根本无法进行. 根据5.3的讨论, 为了减小方差, 可以不在积分区域内均匀取点, 而是按照某种分布$P(\mathbf{x})$取点, 此时式(6.4.8)成为

$$B = \frac{\sum_{i=1}^{n} b(\mathbf{x}_i)\exp(-\beta H(\mathbf{x}_i))/P(\mathbf{x}_i)}{\sum_{i=1}^{n}\exp(-\beta H(\mathbf{x}_i))/P(\mathbf{x}_i)} \tag{6.4.9}$$

如果取

$$P(\mathbf{x}_i) = \frac{\exp(-\beta H(\mathbf{x}))}{Z} \tag{6.4.10}$$

为正则分布, 方差可大为减小, 且式(6.4.9)简化成为

$$B = \frac{1}{n}\sum_{i=1}^{n} b(\mathbf{x}_i) \tag{6.4.11}$$

现在的问题在于, 如何实现按正则分布的抽样? 为此, 我们先简单介绍一下Markov过程.

对于一个我们所要研究的统计物理系统, 它可以处于各种微观状态. 例如对于Ising模型, 给定了每个自旋的取值就对应于一种微观状态, 如果系统由N个自旋构成, 则总的微观状态数就有2^N 个; 又如对于一个由N 个粒子组成的系统, 给定了每个粒子的位置和动量就决定了一个状态, 这可用由$3N$维空间和$3N$维动量空间构成的$6N$维相空间的一个点来表示. 从经典角度来讲, 系统可有无穷多个微观状态, 但量子力学的不确定关系限制了相空间一个"点"最小为h^{3N}, h为Plank 常数. 因而系统总的微观状态数约为$\left(\frac{4\pi}{3}\bar{p}^3V\right)^N \frac{1}{h^{3N}N!}$ 式中$\bar{p}$为单个粒子动量大小的平均值, V 为体积, 分母上的$N!$因子是由于考虑了粒子的全同性. 我们注意到, 当N较大时, 系统的微观状态数是非常大的, 因而对所有状态的求和也是无法办到的.

现在我们构造一个过程, 从系统的某一微观状态出发, 并在过程的每一步转移到一个新的状态. 为了确定起见, 下面用$\mathbf{x}_i$代表系统的微观状态, 如果从$\mathbf{x}_0$出发, 则这一过程产生一系列状态$\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x}_i, \cdots$, 这一系列状态构成一个链. 所谓Markov 过程, 是指这样一种过程, 在过程的每一步所达到的状态只与前一状态有关, 从一状态r到另一状态s的转移通过一转移概率$w(\mathbf{x}_r \to \mathbf{x}_s)$ 来实现. 由Markov 过程产生的一系列状态所构成的链称为Markov 链.

为了实现按照正则分布抽样, 我们可以构造这样一个Markov 链, 使得无论从何状态出发, 存在一个大数M, 在丢掉链的前面M个状态后, 链上其余的状态满足正则分布. 现在我们来证明, 只要取$w(\mathbf{x}_r \to \mathbf{x}_s)$ 满足如下条件, 就可达到我们的要求.

$$P(\mathbf{x}_r)w(\mathbf{x}_r \to \mathbf{x}_s) = P(\mathbf{x}_s)w(\mathbf{x}_s \to \mathbf{x}_r) \tag{6.4.12}$$

式中$P(\mathbf{x})$为正则分布(6.4.10). 这一式子又称为细致平衡条件. 为了证明上式, 我们考虑很多个平行的Markov链, 在一个给定的某一步, 有N_r个链处于第r个态, N_s个链处于第s个态. 于是在下一步从r态到s态的数目为

$$N_{r\to s} = N_r w(\mathbf{x}_r \to \mathbf{x}_s)$$

从s态到r态的数目为

$$N_{s\to r} = N_s w(\mathbf{x}_s \to \mathbf{x}_r)$$

从r态到s态的净转移的数目为

$$\Delta N_{r\to s} = N_r w(\mathbf{x}_s \to \mathbf{x}_r)\left[\frac{w(\mathbf{x}_r \to \mathbf{x}_s)}{w(\mathbf{x}_s \to \mathbf{x}_r)} - \frac{N_s}{N_r}\right] \tag{6.4.13}$$

若$w(\mathbf{x}_r \to \mathbf{x}_s)$满足式(6.4.12), 则上式成为

$$\Delta N_{r\to s} = N_r w(\mathbf{x}_s \to \mathbf{x}_r)\left[\frac{P(\mathbf{x}_s)}{P(\mathbf{x}_r)} - \frac{N_s}{N_r}\right] \tag{6.4.14}$$

这是一个十分重要的结果, 上式表明, 如果二个状态之间不满足正则分布, 则这一Markov过程的演化结果将总是使其趋于满足. 这样, 就证明了我们的论断.

有了上面的定理, 就可以实现正则分布的抽样了, 我们只要选择一个满足细致平衡条件的转移概率, 产生一个Markov 链, 丢掉链的前而面M个状态, 并用其余状态进行计算即可. 这一算法是五十年代初由Metropolis 提出来的, 因此现在一般称为Metropolis 算法. 考虑从r态到s态的转移, 若二状态的能量差为

$$\delta H \equiv H(\mathbf{x}_s) - H(\mathbf{x}_r)$$

则

$$\frac{w(\mathbf{x}_r \to \mathbf{x}_s)}{w(\mathbf{x}_s \to \mathbf{x}_r)} = \exp[-\beta\delta H]$$

当年Metropolis 选择

$$w(\mathbf{x}_r \to \mathbf{x}_s) = \begin{cases} \exp[-\beta\delta H] & \text{如果} \quad \delta H > 0 \\ 1 & \text{如果} \quad \delta H \le 0 \end{cases} \tag{6.4.15}$$

目前常用的另一种选择是

$$w(\mathbf{x}_r \to \mathbf{x}_s) = \frac{1}{2}\left[1 - \tanh\left(\frac{\beta\delta H}{2}\right)\right] \tag{6.4.16}$$

应当注意的是, w的选择并不唯一, 只要满足(6.4.12)的要求即可, 但不同的w收敛速度往往差别很大, 如何选择合适的w以达到尽可能快的收敛速度和尽可能高的计算精度仍然是当前Monte Carlo算法研究的前沿课题之一.

6.5 Ising 模型的Monte Carlo 模拟

这一节以Ising模型为例, 介绍分立变量模型Monte Carlo模拟方法. 为了模拟Ising 模型, 首先要确定一个晶格和晶格的尺寸. 例如, 我们可以取一个简单立方格子, 取三个方向的大小均为L; 其次, 我们要指定一个初始位形, 一般来说, 如果在临界温度之下进行计算, 通常取所有自旋沿同一方向为初始位形, 而如果在临界温度之上进行计算, 通常取随机分布的自旋取向为初始位形, 也可取所有自旋沿同一方向为初始位形. 这是由于在临界温度之下的平衡位形是有序的, 若从一无序位形出发, 系统在演变中将形成若干个有序的区域, 相邻区域的边界上将出现畴壁. 当然, 如果计算的目的是为了研究畴壁, 则必须从一自旋取向为随机分布的初始位形出发; 第三件事是要确定适当的边界条件, 因为计算总是对有限大小的系统进行的, 选择合适的边界条件对于得到好的结果是十分重要的, 如果我们感兴趣的是系统的体性质, 则应尽量消除边界的影响, 此时一般取周期性边界条件, 即在每个方向上, 取$s_{L+1} \equiv s_1$. 然而, 周期性边界条件对于写并行计算的程序不太方便, 而并行计算是当前计算机发展的一个重要方向, 因此, 另一种称为螺旋周期边界条件得到了较多的应用, 以二维正方格子为例, 这种边界条件是让每一行的最后一个自旋与其下一行的第一个自旋相同; 最后一个需要确定的问题是选取产生一系列状态的方式, 一般来说, Markov 链中的每一个状态与其前一个状态相差应较小, 因为如果两个状态相差过大, 其能量差亦较大, 从而转移概率太小, 计算很容易陷入相空间中初态附近一个很小的子空间内, 通常有两种作法, 一种是一次翻转一个自旋, 这是一种不保持总自旋守恒的计算, 另一种是每次交换一对相邻自旋, 这种计算将保持总自旋守恒.

有了这些讨论, 我们给出一个算法如下:

重复如下步骤多次:

1. 选择一个格点i, 其自旋将考虑作翻转$s_i \to -s_i$.
2. 计算与此翻转相联系的能量变化δH.
3. 用式(6.4.15)计算这一翻转的转移概率w.
4. 产生一在[0,1] 之间均匀分布的随机数z.
5. 如果$z < w$, 则翻转该自旋, 否则, 保持不变. 不论何种情况, 其结果都作为一新的状态.

6. 分析该状态, 为计算平均值收集数据.

计算平均值, 输出结果.

在具体给出这一算法的实现之前, 我们先作一些讨论, 关于每一步要翻转的格点i的选择, 一般来说可有很多种不同的方法, 最常用的有两种, 一种是顺序取每一个格点, 另一种是随机的选取. 在随机选取时, 应使每个格点平均说来被访问的次数相同, 通常每个格点被访问一次称为一个Monte Carlo 步(Monte Carlo Step or MCS), 一次有价值的计算通常需要做几千或几万个MCS. 有时, 为了得到高精度的结果, 甚至要作百万MCS以上的计算.

由于每一个状态与其前导状态最多相差一个自旋翻转, 因而其物理性质具有很强的关联. 这样, 上述过程的第6步不必对每次自旋都进行, 而是每间隔一个或数个MCS(视问题的关联时间的大小)进行一次. 另外, 如在前面已经指出过的, 前面若干个MCS应舍弃.

计算能量差是最费时的工作, 对于Ising 模型, 由于能量差只能取很少几个数值, 我们可以予先算好存起来以节省计算量. 这一技巧不仅适用于Ising 模型, 也适用于其它分立变量的模型如Potts 模型等.

下面我们以三维简单立方格子为例来给出算法的实现. 取一个$L \times L \times L$的简单立方格子, 每个格点放一个自旋, 可取值± 1. 对于一个给定的自旋, 共有6个最近邻格点, 因此, 对其最近邻的求和只取7个可能的值, 也就是说

$$s_i \sum_{<ij>} = \{-6, -4, -2, 0, 2, 4, 6\} \tag{6.5.17}$$

对应于s_i的6个近邻格点中全与s_i反向, 5个反向, ..., 1个反向, 全不反向. 将(6.5.17)中的数加7, 可得到$\{1, 3, 5, 7, 9, 11, 13\}$, 这样, 前面7 个数就代表了一格点周围的近邻自旋相对于该格点取向的全部信息. 如果我们再规定数的正负号与自旋的向上和向下相对应, 则$\{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 11, \pm 13\}$ 可代表无外场时一个格点的自旋及其与其近邻自旋相对取向的全部信息. 为了计及自旋与外场的相对关系, 我们利用前面数之间的空档, 例如, 可令上面的数代表与外场平行的情形, 则可取每一数的绝对值大一的数代表相同位形但自旋与外场反平行的情形. 对于所有$\{\pm 1, \pm 2, \cdots \pm 14\}$共28个数, 实际使用的只有一半, 因为若外场沿自旋向下, 则负奇数代表自旋向下, 正偶数代表自旋向上.

6.6 经典统计物理复习

Statistical mechanics is the studies of many particle systems under external conditions, the number of particles is typically 10^{23}.The basis of statistical mechanics is the ensemble theory, there are three very important ensembles, the micro-canonical, canonical and grand canonical, correspond to fixed total energy, temperature and chemical potential

respectively, for each ensemble there exist a distribution function $P(X)$ from it the average of physical quantities are calculated by

$$\langle A\rangle = \frac{\sum_X A(X) P(X)}{\sum_X P(X)}.$$

Where X represents points in the phase space and

$$\sum_X = \left(\frac{1}{h}\right)^{3N} \int_V d^3r_1 d^3r_2 \cdots d^3r_N \int d^3p_1 d^3p_2 \cdots d^3p_N$$

is the summation over the phase space.

6.6.1 The micro canonical ensemble(NVE)

In this ensemble the distribution is given by

$$P(X) = \delta(H(X) - E).$$

The total number of states with energy E is

$$\Omega(N,V,E) = \frac{1}{N!}\sum_X \delta(H(X) - E).$$

The entropy is defined as

$$S(N,V,E) = k_B \ln \Omega(N,V,E).$$

Thermodynamic quantities are obtained by

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_{N,V} \quad \mu = -T\left(\frac{\partial S}{\partial N}\right)_{V,E} \quad P = T\left(\frac{\partial S}{\partial V}\right)_{N,E}.$$

6.6.2 The canonical ensemble(NVT)

In this ensemble the energy is allowed to vary but the system is in contact with a heat bath at temperature T. The distribution function is

$$P(X) = \exp\left[-\frac{H(X)}{k_B T}\right].$$

The canonical partition function is

$$\begin{aligned} Z(N,V,T) &= \frac{1}{N!}\sum_X \exp\left[-\frac{H(X)}{k_B T}\right] \\ &= \sum_E e^{-\beta E}\Omega(N,V,E). \end{aligned}$$

Where $\beta = 1/k_B T$, The free energy

$$F(N,V,T) = -k_B T \ln Z(N,V,T),$$

and

$$\mu = \left(\frac{\partial F}{\partial N}\right)_{V,T} \quad P = -\left(\frac{\partial F}{\partial V}\right)_{N,T} \quad S = -\left(\frac{\partial F}{\partial T}\right)_{N,V}.$$

6.6.3 The NPT ensemble(NPT)

Here the pressure is fixed while volume is allowed to vary

$$P(V,X) = \exp\left[-\beta\left(PV + H(X)\right)\right].$$

The partition function

$$\begin{aligned} Q(N,P,T) &= \int dV \frac{1}{N!} \sum_X \exp\left[-\beta\left(PV + H(X)\right)\right] \\ &= \int dV \exp\left[-\beta PV\right] Z(N,V,T). \end{aligned}$$

Gibbs free energy

$$G(N,P,T) = -k_B T \ln Q(N,P,T),$$

with

$$\mu = \left(\frac{\partial G}{\partial N}\right)_{p,T} \quad V = \left(\frac{\partial G}{\partial P}\right)_{N,T} \quad S = -\left(\frac{\partial F}{\partial T}\right)_{N,P}.$$

6.6.4 The grand canonical ensemble(μVT)

The particle number is not fixed while the chemical potential is fixed,

$$P(N,X) = \exp\left[-\beta\left(H(X) - \mu N\right)\right],$$

the grand partition function

$$\begin{aligned} Z_G(\mu,V,T) &= \sum_N \frac{1}{N!} \sum_X \exp\left[-\beta\left(H(X) - \mu N\right)\right] \\ &= \sum_N \exp\left[\beta\mu N\right] Z(N,V,T), \end{aligned}$$

the grand canonical potential

$$\Omega_G(\mu,V,T) = -k_B T \ln Z_G(\mu,V,T),$$

with

$$N = -\left(\frac{\partial \Omega_G}{\partial \mu}\right)_{V,T} \quad P = -\left(\frac{\partial \Omega_G}{\partial P}\right)_{V,T} \quad S = -\left(\frac{\partial \Omega_G}{\partial T}\right)_{\mu,V}.$$

6.6.5 Thermodynamical relations

There are relations between thermodynamic potentials

$$\begin{aligned} F &= E - TS, \\ G &= F + PV = N\mu, \\ \Omega_G &= F - \mu N = -PV. \end{aligned}$$

There are two kinds of physical quantities, one is the mechanical quantities, which can be obtained by average with respect to distributions, the typical one is $H(X)$:

$$E = \langle H(X) \rangle .$$

The other is thermal quantities which can only be obtained through the derivatives of the thermodynamic potentials, the typical one is chemical potential μ and free energy itself. This kind of quantities are difficult to calculate in simulations and often obtained through integration of mechanic quantities

$$\begin{aligned} F(T, V_1) &= F(T, V_0) - \int_{V_0}^{V_1} P(T, V)\, dV; \\ \frac{F(T_1, V)}{T_1} &= \frac{F(T_0, V)}{T_0} - \int_{T_0}^{T_1} \frac{E(T, V)}{T^2} dT. \end{aligned}$$

6.6.6 Correlation functions

A very important quantity in statistical mechanics is the pair correlation function $g(\mathbf{r}, \mathbf{r}')$, defined as

$$g(\mathbf{r}, \mathbf{r}') = \frac{V^2}{Z_R} \int_V d^3 r_3 d^3 r_4 \cdots d^3 r_N \exp\left[-\beta V_N(\mathbf{r}, \mathbf{r}', \mathbf{r}_{3,} \cdots, \mathbf{r}_N)\right],$$

where

$$Z_R = \int_V d^3 r_1 d^3 r_2 \cdots d^3 r_N \exp\left[-\beta V_N(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_{3,} \cdots, \mathbf{r}_N)\right].$$

It may also be written as

$$g(\mathbf{r}, \mathbf{r}') = \frac{V^2}{N(N-1)} \left\langle \sum_{i,j;i \neq j} \delta(\mathbf{r} - \mathbf{r}_i)\, \delta(\mathbf{r}' - \mathbf{r}_j) \right\rangle .$$

For a homogeneous system the pair correlation function depends only on the distance between $\mathbf{r}$ and $\mathbf{r}'$. In this case we denote it as $g(r)$.

The $g(\mathbf{r}, \mathbf{r}')$ is proportional to the probability that given a particle at point $\mathbf{r}$ and find another particle at point $\mathbf{r}'$. At large distances $g(r)$ tends to 1, we may define the total correlation function

$$h(\mathbf{r}) = g(\mathbf{r}) - 1.$$

The Fourier transform of the above function gives the static structure function

$$S(\mathbf{k}) = 1 + n\int g(\mathbf{r})\, e^{i\mathbf{k}\cdot\mathbf{r}} d^3r.$$

The structure function is defined as the correlation function of Fourier component of density fluctuations

$$\begin{aligned} S(\mathbf{k}) &= \frac{1}{N}\langle \Delta n_{-\mathbf{k}} \Delta n_{\mathbf{k}}\rangle = \frac{1}{N}\sum_{ij;i\neq j}\left\langle e^{i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j)}\right\rangle + 1 \\ &= \int d^3r d^3r' e^{i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}')}\frac{1}{N}\sum_{ij;i\neq j}\langle \delta(\mathbf{r}-\mathbf{r}_i)\,\delta(\mathbf{r}-\mathbf{r}_j)\rangle + 1 \\ &= \frac{N-1}{V^2}\int d^3r d^3r' e^{i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}')} g(\mathbf{r}-\mathbf{r}') + 1 \\ &= 1 + n\int d^3r g(\mathbf{r}) e^{i\mathbf{k}\cdot\mathbf{r}} \\ &= 1 + n\int d^3r h(\mathbf{r}) e^{i\mathbf{k}\cdot\mathbf{r}} + n\delta(\mathbf{k}). \end{aligned}$$

The structure function can be measured directly by scattering experiments and can also be calculated by simulations.

Many physical quantities can be expressed in terms of the pair correlation functions, for example the energy in NVT ensemble is

$$E = \frac{3}{2}Nk_BT + \frac{N}{2}n\int d^3r\, V(\mathbf{r})\, g(\mathbf{r}).$$

The pressure is

$$\begin{aligned} \frac{\beta P}{n} &= 1 - \frac{\beta}{3N}\left\langle \sum_{i=1}^{N}\hat{\mathbf{r}}_i\cdot\nabla_i V_N\right\rangle \\ &= 1 - \frac{\beta}{6}n\int d^3r\, \hat{\mathbf{r}}\cdot\nabla V(\mathbf{r})\, g(\mathbf{r}). \end{aligned}$$

The compressibility

$$k_BT\left(\frac{\partial n}{\partial p}\right)_T = 1 + n\int d^3r\,(g(\mathbf{r})-1).$$

This expression can be derived from the fluctuations of particle numbers

$$\left\langle N^2\right\rangle - \langle N\rangle^2 = k_BT\left(\frac{\partial\langle N\rangle}{\partial\mu}\right)_{T,V} = -k_BT\frac{\langle N\rangle^2}{V^2}\left(\frac{\partial V}{\partial P}\right)_{T,\langle N\rangle}.$$

Since $n = \frac{\langle N\rangle}{V}$, $dV = -\frac{V^2}{\langle N\rangle}dn$, so

$$\left\langle N^2\right\rangle - \langle N\rangle^2 = \langle N\rangle k_BT\left(\frac{\partial n}{\partial P}\right)_T.$$

On the other hand, it can be proved that

$$\left\langle N^2\right\rangle - \langle N\rangle^2 = \langle N\rangle \left(1 + n\int d^3r\left(g\left(\mathbf{r}\right) - 1\right)\right).$$

We have the final results.

6.6.7 Time correlation function and transport coefficients

The time correlation function: Consider the correlations of two physical quantities at different times,

$$C_{AB}(t,t') = \langle A(t)B(t')\rangle .$$

For systems at equilibrium the time correlation function is a function of the time difference only and can be written as

$$C_{AB}(t) = \langle A(t)B(0)\rangle .$$

One example of time correlation function is the velocity auto correlation function of the ith particle

$$C_{\mathbf{vv}}\left(t\right) = \left\langle \mathbf{v}_i\left(t\right)\cdot\mathbf{v}_i\left(0\right)\right\rangle .$$

Transport coefficients are defined in terms of the response of a system to a perturbation. For example, the diffusion coefficient relates the particle flux to a concentration gradient, while the shear viscosity is a measure of the shear stress induced by an applied velocity gradient. By introducing such perturbations into the Hamiltonian, or directly into the equations of motion, their effect on the distribution function P may be calculated. Generally, a time-dependent, non-equilibrium distribution $P_{eq} + \delta P\left(t\right)$ is produced. Hence, any non-equilibrium ensemble average may be calculated. By retaining the linear terms in the perturbation and comparing the equation for the response with a macroscopic transport equation, we may identify the transport coefficients. This is usually the infinite time integral of an equilibrium time correlation function of the form

$$\gamma = \int_0^\infty dt\left\langle \dot{A}\left(t\right)\dot{A}\left(0\right)\right\rangle$$

where γ is the transport coefficient, and A is a physical variable appearing in the perturbation Hamiltonian. There is also an Einstein relation associated with this kind of expression

$$\left\langle \left(A\left(t\right) - A\left(0\right)\right)^2\right\rangle = 2\gamma t$$

which holds for large t , ($t \gg \tau$, where τ is the correlation time of A).

The diffusion coefficient D is given by

$$D = \frac{1}{3}\int_0^\infty dt\ \langle \mathbf{v}_i(t)\cdot\mathbf{v}_i(0)\rangle$$

and the Einstein relation is

$$\frac{1}{3}\left\langle \left(\mathbf{r}_i(t)-\mathbf{r}_i(0)\right)^2\right\rangle = 2Dt.$$

Where $\mathbf{r}_i(t)$ and $\mathbf{v}_i(t)$ are the position and velocity of the ith particle.

The shear viscosity η is given by

$$\eta = \frac{V}{k_BT}\int_0^\infty dt\ \langle P_{\alpha\beta}(t)\,P_{\alpha\beta}(0)\rangle$$

or

$$\frac{V}{k_BT}\langle Q_{\alpha\beta}(t)\,Q_{\alpha\beta}(0)\rangle = 2\eta t.$$

Here

$$\begin{aligned} P_{\alpha\beta} &= \frac{1}{V}\sum_i\left(\frac{p_{i\alpha}p_{i\beta}}{m_i}+r_{i\alpha}f_{i\beta}\right)\\ Q_{\alpha\beta} &= \frac{1}{V}\sum_i r_{i\alpha}p_{i\beta}. \end{aligned}$$

The negative of $P_{\alpha\beta}$ is often called stress tensor.

6.7 液体模型的Monte Carlo模拟

相对于气体和固体，液体是最复杂的一相。描写液体的最简单的模型是硬球模型，这个模型把每个分子看成直径为σ的硬球，并假定除此之外没有其它相互作用。即便是这样一个简单的模型，也没有找到它的精确解。尽管如此，通过数十年来大量的研究，我们对这个模型已经有了非常多的了解。硬球系统的相互作用势只有0或∞两个可能值，因此，模拟计算非常简单。另一个稍微实际一点的模型是Lennard-Jones模型，这个模型假定分子是球形的，其二体相互作用为：

$$V_{LJ}(r) = \varepsilon\left[\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r}\right)^{6}\right]$$

这一模型由两部分构成，一部分是吸引相互作用，与距离的6次方成反比，来自于中性原子的涨落偶极矩的相互作用，是一个纯量子力学效应，可以通过二阶微扰得到；另一部分是与距离的12次方成反比的排斥相互作用，这一部分主要来自于泡利不相容原理，但并没有很好的量子力学计算，基本上是唯像引入。

这一节主要介绍这两个模型的模拟。

6.7.1 Monte Carlo Simulation of The Lenard -Jones Model

Lets consider the Lenard -Jones model of fluids, we have already met this model before. In this model one consider a collection of structureless particles interacting through the pairwise potentials, Its interaction energy is

$$V = \sum_{\langle i,j \rangle} 4\varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]. \tag{6.7.18}$$

where $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$, σ and ε are two parameters, the summation is over all the pairs of particles. The configuration of the system is specified by the set of all the particle positions. The Monte Carlo simulation by Metropolis algorithm is given by the following steps

1. Initialize the particles on a FCC lattice.

2. Choose a particle k at random and propose to move: $\mathbf{r}'_k = \mathbf{r}_k + \delta\mathbf{r}_k$, where

$$\begin{aligned} \delta x_k &= \Delta \left(\text{rand}\,() - 0.5 \right) \\ \delta y_k &= \Delta \left(\text{rand}\,() - 0.5 \right) \\ \delta z_k &= \Delta \left(\text{rand}\,() - 0.5 \right). \end{aligned} \tag{6.7.19}$$

   Here Δ is a pre-specified value and adjusted so that the acceptance ratio(see bellow) will be roughly 0.5(not always true), $rand\,()$ is any good random number generator generates uniformly distributed random number between 0 and 1.

3. Calculate the energy increment

$$\Delta E = V\left(\{\mathbf{r}'\}\right) - V\left(\{\mathbf{r}\}\right). \tag{6.7.20}$$

4. Accept the proposed move as the new state if a uniformly distributed random number (between 0 and 1) is less than $\exp\left(-\beta\Delta E\right)$; retain the old configuration as the new configuration otherwise.

5. Go to 2.

In the Monte Carlo simulations using the Metropolis algorithm, the next configuration depends on the previous one. Due to this correlation between Monte Carlo steps, the formula for the statistical error $\varepsilon = \frac{\sigma}{\sqrt{N}}$ underestimates the true error, and it should be replaced by

$$\varepsilon = \sigma \sqrt{\frac{1+2\tau}{N}}. \tag{6.7.21}$$

The quantity τ is called correlation time, which is roughly the number of Monte Carlo steps needed to generate independent configurations. We can compute the correlation time τ as follows: define the time-dependent correlation function

$$f(t) = \frac{\langle A(t') A(t'+t)\rangle - \langle A\rangle^2}{\langle A^2\rangle - \langle A\rangle^2}, \tag{6.7.22}$$

where time t is measured in terms of Monte Carlo steps. $A(t')$ is the quantity at steps t' whose error we want to estimate. The angular brackets denote average over Monte Carlo steps. The correlation time is defined by

$$\tau = \sum_{t=1}^{\infty} f(t). \tag{6.7.23}$$

Just as many physical quantities become singular at the critical point, the correlation time is also becomes singular at the critical point. On an infinite lattice it diverges with a power law

$$\tau \propto |T - T_c|^{-\nu z}. \tag{6.7.24}$$

Here ν is the correlation length exponent and z is called dynamic critical exponent. This phenomena that the correlation time becomes very large is called critical slowing down. It not only happens in computer simulations but also happens in real systems. On a finite system of linear size L, the correlation time will not go to infinity, but it will grow with size as

$$\tau \propto L^z, \quad \text{at } T = T_c. \tag{6.7.25}$$

Substituting this result into the error formula, we find that

$$\varepsilon \approx \frac{\sigma L^{z/2}}{\sqrt{N}} \tag{6.7.26}$$

for large system near T_c. For the two dimensional Ising model with Metropolis algorithm , $z = 2.1$, we see typically that increasing the size leads to a large error in the quantity to be calculated.

The Metropolis algorithm makes changes locally one site at a time, This is the cause of the critical slowing down. The Swendsen-Wang algorithm and its extensions solved this problem by the cluster methods.

In 1987, Swendsen and Wang proposed a multi-cluster algorithm to solve the problem of the critical slowing down, their algorithm reduced the dynamical exponent z from 2.1 to around 0 for 2D Ising model. The algorithm goes:

1. Start from some arbitrary state $\{s\}$.

2. Go through each nearest neighbor connection of the lattice, create a bond between the two neighboring sites i and j with probability $p = 1 - \exp(-2\beta J)$ when the two spins are the same. Never put a bond between the sites if the spin values are different.

3. Identify clusters as a set of sites connected by bonds, or isolated sites. Two sites are said to be in the same cluster if there is a connected path of bonds joining them. Every site has to belong to one of the clusters. After the clusters are found, each cluster is assigned a new Ising spin chosen with equal probability between $+1$ and -1. The old spin values now can be forgotten.

4. One MCS finished. Go to step 2 for the next step.

The performance of the algorithm in terms correlation time in comparison with Metropolis algorithm is remarkable. For two dimensional Ising model the dynamic exponent is less than 0.3, or possibly 0 (but $\tau \propto \ln L$). In three dimensional it is about 0.5, At and above four dimensions it is 1.

In 1989 Wollf proposed a algorithm based on the Swendsen-Wang algorithm, in stead of generate many clusters each time, Wollf proposed that one picks a site at random, and then generate one single cluster by growing a cluster from the seed. The neighbors of the seed will belong to the cluster if the spins are parallel to the seed and a random number is less than $p = 1 - \exp(-2\beta J)$. Neighbors of each new site are tested for membership. The recursive process will eventually terminate. The spins in the cluster are turned over with probability 1.

Wollf algorithm is more efficient than the multi-cluster Swendsen-Wang algorithm and easier to implement.

There are many ways to extend the cluster method both to Ising like models and continuous models, it seems for Ising like models Wollf algorithm is still the best choice.

6.7.2 自由能计算

The calculation of free energy, entropy or chemical potential is much more difficult than the calculation of the averages of mechanical quantities in Monte Carlo methods. We give here a very brief discussion of the calculation of free energy by Monte Carlo simulations.

The most common way is the thermodynamical integration method, which was discussed in the context of molecular dynamics simulations. We give here the Bennett

method. In this method one calculates the difference of two systems described by potentials U_0 and U_1 respectively. The free energy is

$$\beta\Delta F = \beta F_1 - \beta F_0 = -\ln\frac{Z_1}{Z_0}. \tag{6.7.27}$$

Where

$$\begin{aligned} Z_1 &= \int \exp\left(-\beta U_1\left(x\right)\right)\, dx, \\ Z_0 &= \int \exp\left(-\beta U_0\left(x\right)\right)\, dx. \end{aligned} \tag{6.7.28}$$

are partition functions for potential U_1 and U_0. The ratio may be transformed as

$$\begin{aligned} \frac{Z_1}{Z_0} &= \frac{Z_1\int W\left(x\right)\exp\left\{-\beta\left[U_0\left(x\right)+U_1\left(x\right)\right]\right\}dx}{Z_0\int W\left(x\right)\exp\left\{-\beta\left[U_0\left(x\right)+U_1\left(x\right)\right]\right\}dx} \\ &= \frac{\left\langle W\exp\left[-\beta U_1\right]\right\rangle_0}{\left\langle W\exp\left[-\beta U_0\right]\right\rangle_1}. \end{aligned} \tag{6.7.29}$$

Where the subscript 0 and 1 means average with respect to canonical distributions of potential U_0 and U_1. The W is an arbitrary function. Now we choose

$$W = \text{const.}\left(\frac{Z_1}{n_1}\exp\left[-\beta U_0\right] + \frac{Z_0}{n_0}\exp\left[-\beta U_1\right]\right), \tag{6.7.30}$$

where n_0 and n_1 are two arbitrary constant, we have

$$\frac{Z_1}{Z_0} = \frac{Z_1 n_0}{Z_0 n_1}\frac{\left\langle\left(1+\frac{Z_1 n_0}{Z_0 n_1}\exp\left[\beta\left(U_1-U_0\right)\right]\right)^{-1}\right\rangle_0}{\left\langle\left(1+\frac{Z_0 n_1}{Z_1 n_0}\exp\left[\beta\left(U_0-U_1\right)\right]\right)^{-1}\right\rangle_1}. \tag{6.7.31}$$

If we set

$$\frac{Z_1 n_0}{Z_0 n_1} = e^C \tag{6.7.32}$$

then we get

$$\frac{Z_1}{Z_0} = e^C\frac{\left\langle f\left(\beta U_1-\beta U_0+C\right)\right\rangle_0}{\left\langle f\left(\beta U_0-\beta U_1-C\right)\right\rangle_1}. \tag{6.7.33}$$

where $f\left(x\right) = \left(1+e^x\right)^{-1}$ is the Fermi function(there is nothing to do with Fermi distribution!). So that the free energy difference is

$$\Delta F = -\ln\frac{Z_1}{Z_0} = \ln\frac{\left\langle f\left(\beta U_1-\beta U_0+C\right)\right\rangle_0}{\left\langle f\left(\beta U_0-\beta U_1-C\right)\right\rangle_1} - C. \tag{6.7.34}$$

6.7.3 王-Landau 方法及其它

与自由能计算相关的的是态密度的计算，在统计物理中，态密度定义为单位能量间隔内的微观状态数，通常记为$\Omega(E)$. 如果已知一个系统的态密度，那么，配分函数可以写成

$$Q(\beta) = \int \Omega(E) e^{-\beta E}\, dE$$

对Q求对数并乘以$-kT$就得到了自由能。

1987年，Ferrenberg 和Swendsen （Physical Review Letters, 61,2635,1988）提出了一个直方图算法，实际上用到了态密度。利用这一方法，可以通过很少几个点的模拟而得到整个温区的结果。其基本思路非常简单：注意到给定温度下系统按照能量的分布是

$$P(\beta, E) = \frac{\Omega(E) e^{-\beta E}}{Q(\beta)}$$

对于一个确定的温度，由Metropolis算法，可以得到这个分布的直方图。直方图$H(\beta, E)$与$P(\beta, E)$成正比，或

$$H(\beta, E) \propto \Omega(E) e^{-\beta E}$$

于是

$$\Omega(E) \propto H(\beta, E) e^{\beta E}$$

对于另一个温度β', 其分布可由上述$\Omega(E)$得到

$$P(\beta', E) = \frac{\Omega(E) e^{-\beta' E}}{Q(\beta')} = \frac{H(\beta, E) e^{(\beta - \beta')E}}{\int dE\, H(\beta, E) e^{(\beta - \beta')E}}$$

这样，在一个温度下的模拟结果，原则上可以用来计算所有温度的平均值。而上式的分母，则可以用来估算配分函数从而估算自由能（相差一个任意常数）。

这一方法取得了巨大成功，大致可以把Monte Carlo的计算时间减少一个量级，同时也提供了计算自由能的思路和方法。

另一个直接针对态密度的计算的方法是由Berg在1992年提出来的多正则系综方法。这是一个迭代算法，基于如下观察。设系统的态密度为$\Omega(E)$, 现在，如果我们已经知道了态密度，并利用$\frac{1}{\Omega(E)}$取样做模拟计算，那么我们将得到一个平的分布，也就是说模拟结果对能量做直方图$H(E)$, 将得到一个常数。

Berg算法的思路是，先假设一个态密度$\Omega(E)$, 然后由此态密度的导数取样，得到直方图$H(E)$, 然后，把态密度变为$\Omega(E)/H(E)$, 开始一轮新的迭代，知道直方图变平（在给定的精度下），最后得到的$\Omega(E)$就是所求的态密度。

这一方法思路很清楚，但实现起来非常困难。首先，方法本身的稳定性不好，对于一个比较任意的初始态密度，很难得到收敛的结果。其次，需要认真处理能量的上界和下界处的问题，在这些地方，态密度很小，容易出问题。

2005年由王福高和Landau提出的算法，克服了Berg算法的缺点，这一算法的基本思路与Berg的思路相同，但采取了一个不满足细致平衡条件的做法。在模拟的每一步都修正态密度。其基本步骤是：1,给定一个初始态密度$\Omega(E)$，从一个初始位形（对应于一个确定能量）出发；2，按照$\frac{1}{\Omega(E)}$ 取样，前进一步，到达新的位形和对应能量；3，立刻把这个新的能量对应的态密度乘以一个放大因子，继续计算。在计算中同时统计直方图，当直方图基本变平时，缩小放大因子，并把所得态密度作为新的初始态密度，开始计算。当放大因子缩的足够小，且直方图在精度要求下变平时，得到最终的态密度计算结果。

这个方法的稳定性非常好，几乎对于所有模型，任何初始态密度都能得到收敛的结果。这个算法的发现，被看成是Monte Carlo历史上具有里程碑意义的事件。但是，这个算法也有比较严重的问题，虽然能够保证收敛且计算非常稳定，但收敛的精度似乎有一个极限，到达此极限后，似乎无法通过加大计算量来提高精度。另外，一个显而易见的问题是，即便以精确态密度作为初始态密度，一旦开始计算，也开始引进误差，最终得到的是有一定误差的态密度。

6.8 分子动力学方法

6.8.1 General procedure of MD (NVE ensemble)

Now I describe the general procedure of the MD in the context of the NVE ensemble. A molecular dynamics simulation includes the following basic steps:

1. Initialize;
2. Start simulation and let the system reach equilibrium;
3. Continue simulation and store results.

Now we discuss these steps in detail.

Initialize: The number of particles and interactions between particles are specified; The simulation box is setup, the total energy of the system is specified, usually temperature is more important than energy so it can be an input parameter, we will see how can we tune the system to a desired temperature.

We assign to each particle its position and momentum. In many cases we assign particles in a FCC lattice, which is a closed packing structure and usually the ground state of many systems. If we use cubic unit cell and cube box then the number of particles per unit cell is 4, and the total number of particles are $4M^3$, $M = 1, 2, 3, \cdots$.

That is we may simulation systems with total number of particles $N = 108, 256, 500, 864, \cdots$. The velocities of particles are draw from a Maxwell distribution with the specified temperature. This is accomplished by drawing the three components of the velocity from the Gaussian distribution, for example, the distribution of the x-component is

$$\exp\left[-\frac{mv_x^2}{2k_BT}\right].$$

Draw numbers from a Gaussian distribution may be done in this way, consider the distribution

$$P(v_x, v_y) \propto \exp\left[-\frac{mv_x^2}{2k_BT}\right]\exp\left[-\frac{mv_y^2}{2k_BT}\right] = \exp\left[-\frac{m\left(v_x^2+v_y^2\right)}{2k_BT}\right].$$

Then

$$P(v_x, v_y)dv_xdv_y = P(v)vdvd\phi,$$

where $v^2 = v_x^2 + v_y^2$ and

$$P(v) \propto \exp\left[-\frac{mv^2}{2k_BT}\right].$$

So the distribution of v_x and v_y may be obtained from v and ϕ, which has distribution $v\exp\left[-\frac{mv^2}{2k_BT}\right]$ and uniform in the interval $[0, 2\pi]$. Since

$$\int v\exp\left[-\frac{mv^2}{2k_BT}\right]dv = -\frac{2k_BT}{m}\exp\left[-\frac{mv^2}{2k_BT}\right] + C.$$

If we draw random numbers x uniformly distributed in the interval $[0, 1]$, then

$$v = \sqrt{-\frac{2k_BT}{m}\ln(x)}$$

will has distribution $v\exp\left[-\frac{mv^2}{2k_BT}\right]$, now if we draw random numbers ϕ uniformly distributed in the interval $[0, 2\pi]$. We can get two random numbers $v_x = v\cos\phi,\ v_y = v\sin\phi$ satisfy the Gaussian distribution.

Another method of draw random numbers in the Gaussian distribution is through the following *empirical* methods. Consider the distribution

$$\exp\left[-\frac{x^2}{2}\right].$$

According to the center limit theorem, if we draw uniform random numbers r_i in interval $[0, 1]$, and define a variable

$$\xi = \frac{\frac{1}{n}\sum_{i=1}^{n} r_i - \frac{1}{2}}{\sqrt{\frac{1}{12n}}},$$

when $n \to \infty$ the distribution of ξ is the Gaussian distribution $\exp\left[-\frac{\xi^2}{2}\right]$. And the distribution of $\xi' = \xi\sigma$ is $\exp\left[-\frac{\xi'^2}{2\sigma^2}\right]$. If we take $n = 12$, we get

$$\xi = \sum_{i=1}^{12} r_i - 6.$$

That is we may generate each random number distributed according to Gaussian from 12 random numbers in an uniform distribution.

Homework:

1, Write programs for the two methods to generate Guassian random numbers.

2, Compare the two methods for efficiency and quality.

After the generation of the velocity of each particle, we may shift the velocity so that the total momentum is zero

Start simulation and let the system reach equilibrium: The system we prepared in the way described above are not in the equilibrium state, we now use an algorithm to integrate it. There are many methods used in molecular dynamics simulations, we use a very simple one to describe the concepts here and describe the algorithms in the lecture follows.

The standard Verlet algorithm is the first successful method in history and still wide used today in different forms. It is

$$\mathbf{r}(t+h) = 2\mathbf{r}(t) - \mathbf{r}(t-h) + h^2\mathbf{F}(\mathbf{r}(t))/m.$$

Where $\mathbf{r}(t)$ is the position of particle at time $t = nh$. To start the integration we need $\mathbf{r}(h)$, given by

$$\mathbf{r}(h) = \mathbf{r}(0) + h\mathbf{v}(0) + h^2\mathbf{F}(\mathbf{r}(0))/m.$$

During the integration, the velocity can be calculated by

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+h) - \mathbf{r}(t-h)}{2h}.$$

Variations of this method are

$$\begin{aligned} \mathbf{v}(t+h/2) &= \mathbf{v}(t-h/2) + h\mathbf{F}(\mathbf{r}(t)) \\ \mathbf{r}(t+h) &= \mathbf{r}(t) + h\mathbf{v}(t+h/2). \end{aligned}$$

And

$$\begin{aligned} \mathbf{r}(t+h) &= \mathbf{r}(t) + h\mathbf{v}(t) + h^2\mathbf{F}(\mathbf{r}(t)) \\ \mathbf{v}(t+h) &= \mathbf{v}(t) + h\frac{\mathbf{F}(\mathbf{r}(t+h)) + \mathbf{F}(\mathbf{r}(t))}{2}. \end{aligned}$$

Both of these variations are mathematically equivalent to the original one but more stable under finite precision arithmetic.

The PBC are usually treated in two ways, one is the minimum image method, that is, in each direction, we only sum the force due to particles which is not farther than $L/2$ in that direction. The other is the cut off method, one simply cut the interaction at a distance r_{cutoff} and neglect interactions beyond this distance. In this method the discontinuity at r_{cutoff} may cause inaccuracies in the integration, this may be solved by shift the potential in the way as

$$V_{\text{sh}}(r) = V(r) - V(r_{\text{cutoff}}).$$

For long range interactions like electrostatic interactions, special method should be used, we will discuss it latter.

The temperature of the system is given by the equal partition theorem, that is the average of kinetic energy of each degree of freedom is half k_BT, that is

$$\frac{3}{2}k_BT_D = \frac{1}{N-1}\sum_{i=1}^{N}\left\langle \frac{1}{2}mv_i^2 \right\rangle.$$

The $N-1$ is due to the conservation of the total momentum reduce the degree of freedom by 3. To reach the desired temperature we may scale the velocity at every few steps of integration

$$\mathbf{v}_i(t) \to \lambda \mathbf{v}_i(t),$$

and the scaling factor λ is chosen as

$$\lambda = \sqrt{\frac{(N-1)\,3k_BT}{\sum_{i=1}^{N} mv_i^2}}.$$

Continue simulation and store results: After the system reaches equilibrium the integration continue in the same way as above without scaling of velocity. The data are stored or accumulated for the calculating physical properties. The static properties of physical quantity A is given by a time average

$$\bar{A} = \frac{1}{n-n_0}\sum_{\nu>n_0}^{n} A_\nu,$$

here A_ν is the value of A at νth time step. Usually the data stored in each step include:

1, the kinetic energy $\sum_{i=1}^{N}\frac{1}{2}mv_i^2$;

2, the potential energy $U = \sum_{\langle i,j\rangle} V(r_{ij})$;

3, the virial $\sum_{\langle i,j\rangle} r_{ij}\frac{\partial V(r_{ij})}{\partial r_{ij}}$.

We also needs data to calculate the pair correlation function, this is done by divide the interval $[0, r_{\max}]$ into sub intervals $[i\Delta r, (i+1)\Delta r]$, at each stage of updating, add the number of pairs with separation in the interval to an array $n(i)$ and find the average value after simulation, the pair correlation function is given by

$$g(r) = \frac{V}{N(N-1)/2} \frac{\langle n(r) \rangle}{4\pi r^2 \Delta r}.$$

Tail corrections:

$$\begin{aligned} \langle U \rangle &= \langle U \rangle_{\text{cutoff}} + 4\pi \frac{N}{V} \int_{r_{\text{cutoff}}}^{\infty} r^2 dr V(r) g(r) \\ \frac{P}{k_B T} &= 1 - \frac{1}{3Nk_B T} \left\langle \sum_{\langle i,j \rangle} r_{ij} \frac{\partial V}{\partial r_{ij}} \right\rangle_{\text{cutoff}} - \frac{N}{6k_B TV} \int_{r_{\text{cutoff}}}^{\infty} r^3 dr \frac{\partial V}{\partial r} g(r). \end{aligned}$$

The pair correlation function in the above expression may be taken to be the asymptotic value $g(r) = 1$.

6.8.2 Simulation of Lennard-Jones liquids

Consider the system of particles interact by the potential

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right]$$

which are good approximation for inert gases. Some values of ε and σ are

	Ar	Kr	Xe	CH_4	N_2	CO_2	C_2H_4
$\varepsilon/k_B(K)$	120	165	230	143	85	216	202
$\sigma(A)$	3.43	3.72	4.05	3.77	3.72	3.88	4.26

.

We may use ε/k_B as the unit of temperature, energy in unit of ε, length in unit of σ, and $\tau = \sqrt{\frac{m\sigma^2}{\varepsilon}}$ as the unit of time. A good time step is about $h = 0.004$. The cut off distance may be set to 2.5σ.

Homework:

1, Write a MD program use the cut off potential to simulate LJ system.

2, Calculate the pressure and energy of Ar at 3-5 different densities from, say 0.9 *to* 1.2 *under temperature* 1. *(note: the averaged temperature may be slightly different to the desired value).*

6.8.3 Simulation of Hard-sphere systems

The simulation of hard sphere systems historically was the first Molecular simulation on digital computers(Alder and Wainwright, *J. Chem. Phys,* 27, 1208(1957); 31,459(1959)). It consists of the following steps:

1, Locate the next collision;

2, Move all particles forward until collision occurs;

3, Implementing the collision dynamics of the collision pair(usually elastic);

4, Calculate quantities interest.

We describe the steps, consider two spheres, i and j, of diameter σ, whose positions at time t are $\mathbf{r}_i$ and $\mathbf{r}_j$, and velocities are $\mathbf{v}_i$ and $\mathbf{v}_j$, respectively. If these particles are to collide at time $t + t_{ij}$ then the following equation will be satisfied

$$|\mathbf{r}(t + t_{ij})| = |\mathbf{r}_{ij}(t) + \mathbf{v}_{ij} t_{ij}| = \sigma$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. If we define $b_{ij} = \mathbf{r}_{ij} \cdot \mathbf{v}_{ij}$, then this equation becomes

$$v_{ij}^2 t_{ij}^2 + 2 b_{ij} t_{ij} + r_{ij}^2 - \sigma^2 = 0.$$

There will be no real positive solution of this equation if $b_{ij} > 0$, or $b_{ij}^2 - v_{ij}^2 \left(r_{ij}^2 - \sigma^2\right) < 0$. Otherwise we have

$$t_{ij} = \frac{-b_{ij} - \left(b_{ij}^2 - v_{ij}^2 \left(r_{ij}^2 - \sigma^2\right)\right)^{1/2}}{v_{ij}^2}.$$

This is the time that particle i and j will collide. For each pair we may get a time t_{ij}, or never collide, the collide time is then $t + \min(t_{ij})$.

The collision dynamics of smooth hard spheres is the conservation of kinetic energy and linear momentum, which for two spheres i and j having the same mass we obtain

$$\begin{aligned} \mathbf{v}_i^{new} &= \mathbf{v}_i^{old} + \delta \mathbf{v}_i \\ \mathbf{v}_j^{new} &= \mathbf{v}_j^{old} - \delta \mathbf{v}_i. \end{aligned}$$

Where $\delta \mathbf{v}_i$ is given by

$$\delta \mathbf{v}_i = -\frac{b_{ij}}{\sigma^2} \mathbf{r}_{ij}.$$

b_{ij} is evaluated at the moment of collide.

Homework

Write (or find) a program for the hard-sphere MD simulations and calculate the pair correlation function of the system.

6.9 Simulation of Langevin dynamics

The motion of Brownian particles are described by the Langevin equation

$$m\frac{d\mathbf{v}}{dt} = -\gamma\mathbf{v}(t) + \mathbf{F}(t) + \mathbf{R}(t),$$

the difference between Langevin dynamics and the ordinary particle dynamics is the *random force* $R(t)$. This force is a result of random collisions by molecules of solvent to Brownian particles, it satisfy the following correlation assumption

$$\langle R_i(t) R_j(0)\rangle = q\delta(t).$$

Here $q = 2k_BT\gamma$ from Einstein relation.

To simulate this dynamics, we need a way to treat the random force, usually the method of verlet method may be used in the integration of the equation, i.e.

$$\begin{aligned}\mathbf{v}(t+h/2) &= \mathbf{v}(t-h/2) + h\mathbf{F}(\mathbf{r}(t)) + \Delta\mathbf{v}\\ \mathbf{r}(t+h) &= \mathbf{r}(t) + h\mathbf{v}(t+h/2).\end{aligned}$$

Where $\Delta\mathbf{v}$ is the contribution from random force and can be calculated in the following way, the correlation of $R(t)$ indicates that the distribution of random force R is Guassian(in the following we consider only one component),

$$P(R(t)) \propto \exp\left(-\frac{R^2}{2\langle R^2\rangle}\right).$$

If we solve for the Langevin equation without external force we obtain

$$v(t) = v(0)\exp\left(-\frac{\gamma t}{m}\right) + \frac{1}{m}\int_0^t \exp\left(-\frac{(t-\tau)\gamma}{m}\right) R(\tau)\, d\tau.$$

When $t \gg m/\gamma$ the first term is zero, since m/γ is typically $10^{-11}s$ (for example, consider a Brownian particle of diameter 1μ suspended in water, $m \sim 4\pi/3\,(d/2)^3\,\rho \sim 0.5\times10^{-18}$, $\gamma = 6\pi\eta\,(d/2) \sim 6\pi\times0.01\times0.5\times10^{-6} \sim 10^{-7}$, $m/\gamma \sim 10^{-11}s$), so if we use a time step much larger than this value, the above condition is satisfied. The mean square of velocity is

$$\left\langle v(t)^2\right\rangle = v(0)^2\exp\left(-\frac{2\gamma t}{m}\right) + \frac{q}{2\gamma m}\left(1-\exp\left(-\frac{2\gamma t}{m}\right)\right),$$

which for large t tends to

$$\left\langle v(\infty)^2\right\rangle = \frac{q}{2\gamma m} \equiv \frac{k_BT}{m}.$$

The last equality comes from definition of temperature. This gives

$$q = 2k_BT\gamma.$$

Since the velocity v is linear in the random force R, so the distribution of v is also a Gaussian

$$P(v) \propto \exp\left(-\frac{mv^2}{2k_BT}\right).$$

We may just sample Δv according to the above distribution.

If the time step is small enough so that the term $\exp\left(-\frac{2\gamma t}{m}\right)$ is not 0 in a practical calculation, we may simply replace the k_BT in the distribution to $k_BT\left(1-\exp\left(-\frac{2\gamma t}{m}\right)\right)$ to obtain

$$P(v) \propto \exp\left(-\frac{mv^2}{2k_BT\left(1-\exp\left(-\frac{2\gamma t}{m}\right)\right)}\right).$$

However，this replacement is a little bit inconsistent with the above treatment since we ignored the contribution of the initial velocity $v(0)$ part. A consistent way of the simulation is to use a time step much larger than the m/γ.

6.10 Long range force and Ewald summation

In the case of Coulomb interactions the interaction potential in the PBC is given by

$$U = \sum_{\mathbf{R}} \sum_{\langle ij \rangle} \frac{q_i q_j}{|\mathbf{r}_i - \mathbf{r}_j + \mathbf{R}|}; \quad \text{with} \ \sum_i q_i = 0.$$

Cut off the above potential will result large errors, one way to avoid the difficulty of the long range interaction is through the use of the curved space, that is, we may consider our finite system is the surface of a four dimension hypersphere, this method involves the use of non-Euclidean geometry in calculating distances. The traditional method is to use the so called Ewald summation and its variants. The derivation of the method for Coulomb interaction and higher order multipole interactions is given separately, we quote here only the results

$$\begin{aligned} U_{PBC} &= \frac{2\pi}{V} \sum_{\mathbf{K}\neq\mathbf{0}} \left| \sum_i q_i \exp\left(i\mathbf{K}\cdot\mathbf{r}_i\right) \right|^2 \frac{\exp\left(-\frac{K^2}{4\alpha}\right)}{K^2} \\ &\quad + \sum_{\langle ij \rangle} q_i q_j \frac{\operatorname{erfc}\left(\sqrt{\alpha}\mathrm{r_{ij}}\right)}{r_{ij}} - \left(\frac{\alpha}{\pi}\right)^{1/2} \sum_i q_i^2. \end{aligned}$$

Chapter 7

原子结构的计算

7.1 原子结构问题

原子由原子核和在原子核周围运动的电子构成, 在原子核静止的坐标系中原子的Hamiltonian 为

$$H = -\sum_{i=1}^{Z} \frac{\hbar^2 \nabla_i^2}{2\mu} - Ze^2 \sum_{i=1}^{Z} \frac{1}{r_i} + \frac{e^2}{2} \sum_{i,j;i\neq j} \frac{1}{|\mathbf{r_i} - \mathbf{r_j}|} \tag{7.1.1}$$

式中μ 为电子质量, e 为基本电荷, $\hbar$为Plank 常数除以2π. 原子的Schödinger 方程为

$$H\psi = E\psi$$

式中ψ是Z个电子的多体波函数. 由量子力学我们知道, 一旦求得了系统的波函数, 则任何力学量都可求出.

在数值计算中, 通常需要将表示物理问题的方程无量纲化, 在原子, 分子以及固体的电子结构的计算中, 通常采用原子单位. 下面就介绍一下原子单位. 对于原子系统来说, 长度用Bohr半径为单位来量度时, 所出现的数字一般为1 的数量级, 因此, 我们自然选择Bohr 半径作为长度的单位, 基于同样的理由, 我们选择氢原子的电离能作为能量的单位, 通常称为Rydberg, 也有选Rydberg的两倍作为能量单位的, 称为Hartree, 而质量的单位就选为电子的质量. 由于在定态问题中不出现时间, 所以选质量, 长度和能量作为三个基本单位. 这些单位与基本物理常数的关系及其数值为:

$$\begin{aligned}
\text{Bohr 半径:} \quad & a_0 = \frac{\hbar^2}{\mu e^2} = 0.5292\text{Å} \\
\text{Rydberg:} \quad & \text{Rydberg} = \frac{\mu e^4}{2\hbar^2} = 13.606\text{eV} \\
\text{Hartree:} \quad & \text{Hartree} = 2\text{Rydberg} = 27.21\text{eV} \\
\text{电子质量:} \quad & \mu = 9.1095 \times 10^{-28} g
\end{aligned} \tag{7.1.2}$$

对式(7.1.1)表示的Hamiltonian 作代换$r = (\hbar^2/\mu e^2)\, r'$, 得

$$\begin{aligned} H &= -\sum_{i=1}^{Z} \frac{\hbar^2 \nabla_i'^2}{2\mu} \frac{1}{\frac{\hbar^2}{\mu e^2}} - Ze^2 \sum_{i=1}^{Z} \frac{1}{r_i'} \frac{\mu e^2}{\hbar^2} \\ &\quad + \frac{e^2}{2} \sum_{i,j;i\neq j} \frac{1}{|\mathbf{r_i'} - \mathbf{r_j'}|} \frac{\mu e^2}{\hbar^2} \end{aligned} \tag{7.1.3}$$

提出因子$\frac{\mu e^4}{2\hbar^2}$ 并令其为1 (单位能量), 略去r' 上面的“ ′ ”, 得到以原子单位表示的Hamiltonian 为:

$$H = -\sum_{i=1}^{Z} \nabla_i^2 - 2Z \sum_{i=1}^{Z} \frac{1}{r_i} + \sum_{i,j;i\neq j} \frac{1}{|\mathbf{r_i} - \mathbf{r_j}|} \tag{7.1.4}$$

在原子单位下, 定态Schrödinger 方程取如下形式:

$$\left(-\sum_{i=1}^{Z} \nabla_i^2 - 2Z \sum_{i=1}^{Z} \frac{1}{r_i} + \sum_{i,j;i\neq j} \frac{1}{|\mathbf{r_i} - \mathbf{r_j}|} \right) \Psi = E\Psi \tag{7.1.5}$$

原子结构计算的任务就是求解上述Schrödinger 方程, 求得本征值和本征函数, 由此便可计算电荷密度, 电离能, 各种跃迁距阵元等物理上感兴趣的量.

7.2 变分法

直接求解上节导出的原子物理问题几乎是不可能的, 而且, 能够用Schrödinger 方程来严格求解的问题也是不多的. 大多数问题要依靠近似方法来解决. 变分法就是一种有效的近似方法.*

定态Schödinger 方程可以从变分原理导出. 考虑泛函

$$I[f] = \frac{\int f^* \hat{H} f dq}{\int f^* f dq} \tag{7.2.6}$$

其中$\hat{H}$ 是所研究的系统的Hamiltonian, $\int \cdots dq$ 表示对系统的全体独立坐标积分(或求和, 假如有这样的情况的话). 我们要证: **当函数f 满足**Schödinger **方程时**, I **有极值**. 证明如下: 设$f = \psi$ 时, I 有极值E

$$E = \frac{\int \psi^* \hat{H} \psi dq}{\int \psi^* \psi dq} \tag{7.2.7}$$

则对于任何与ψ 相差微小变分的函数$f = \psi + \delta\psi$, 应有

$$\delta I = I[\psi + \delta\psi] - I[\psi] = 0 \tag{7.2.8}$$

*本节几乎全部取之于蔡建华先生所著⟨!⟨量子力学⟩⟩ 一书第26节

略去二阶小项, 并注意$\hat{H}$ 是厄密的而且E 是实数, 由(7.2.6)和(7.2.7) 式有

$$\begin{aligned}
\delta I &= \frac{\int (\psi^* + \delta\psi^*)\, \hat{H}\, (\psi + \delta\psi)\, dq}{\int (\psi^* + \delta\psi^*)\, (\psi + \delta\psi)\, dq} - \frac{\int \psi^* \hat{H} \psi dq}{\int \psi^* \psi dq} \\
&= \frac{\int \psi^* \hat{H} \psi dq + \int \delta\psi^* \hat{H} \psi dq + \int \psi^* \hat{H} \delta\psi dq}{\int \psi^* \psi dq + \int \delta\psi^* \psi dq + \int \psi^* \delta\psi dq} - \frac{\int \psi^* \hat{H} \psi dq}{\int \psi^* \psi dq} \\
&= \frac{\int \psi^* \hat{H} \psi dq + \int \delta\psi^* \hat{H} \psi dq + \int \psi^* \hat{H} \delta\psi dq}{\int \psi^* \psi dq} \left[1 - \frac{\int \delta\psi^* \psi dq + \int \psi^* \delta\psi dq}{\int \psi^* \psi dq}\right] \\
&\quad - \frac{\int \psi^* \hat{H} \psi dq}{\int \psi^* \psi dq} \\
&= \frac{\int \delta\psi^* \hat{H} \psi dq + \int \psi^* \hat{H} \delta\psi dq}{\int \psi^* \psi dq} - \frac{\int \psi^* \hat{H} \psi dq}{\int \psi^* \psi dq} \frac{\int \delta\psi^* \psi dq + \int \psi^* \delta\psi dq}{\int \psi^* \psi dq} \\
&= \frac{\int \delta\psi^* \left(\hat{H} - E\right) \psi dq + \int \psi^* \left(\hat{H} - E\right) \delta\psi dq}{\int \psi^* \psi dq}
\end{aligned}$$

由于$\delta\psi$是任意变分, 并且ψ 是复数, $\delta\psi$ 和$\delta\psi^*$ 互相独立, 因此(7.2.8) 要求

$$\hat{H}\psi = E\psi$$

根据上述变分原理, 就有了一个决定能量本征值和定态波函数的近似方法. 假如用某个试探函数f 代入(7.2.6) 式中计算I, 则当f 与严格的定态波函数ψ 有差别Δ 时, 所得的I数值与准确的能量本征值之差至多是Δ^2 的量级. 如果我们选择的试探函数含有若干个参数$\alpha_1, \alpha_2, \cdots, \alpha_t$, 代入积分后, 得到$I = I(\alpha_1, \alpha_2, \cdots, \alpha_t)$, 设当$\alpha_k = \alpha_k^0, k = 1, 2, \cdots, t$时, f 最接近ψ, 则此时$I(\alpha_1^0, \alpha_2^0, \cdots, \alpha_t^0)$ 也最接近于E. 用一个含参数的试探函数f, 实际上就是用某种有限度的变化来代替最一般的任意变分$f = \psi + \delta\psi$. 因为E 是对于任意变分的极值, $I(\alpha_1^0, \alpha_2^0, \cdots, \alpha_t^0)$ 也是$I(\alpha_1, \alpha_2, \cdots, \alpha_t)$ 对于变数$\alpha_1, \alpha_2, \cdots, \alpha_t$ 的极值. 因此, 作为一种近似方法的变分法如下:

猜测Schödinger 方程$\hat{H}\psi = E\psi$ 的解的形式. 用一个具有所猜测的形式, 满足ψ 所必须的边界条件, 并且含若干参数$\alpha_k, k = 1, 2, \cdots, t$ 的试探函数$f(\alpha_1, \alpha_2, \cdots, \alpha_t)$ 代入(7.2.6), 计算$I(\alpha_1, \alpha_2, \cdots, \alpha_t)$, 并相对于$\alpha_k, k = 1, 2, \cdots, t$, 求I 的极值, 即解方程组

$$\frac{\partial I}{\partial \alpha_k} = 0, \quad k = 1, 2, \cdots, t \tag{7.2.9}$$

设方程组的解是$\alpha_k = \alpha_k^0, k = 1, 2, \cdots, t$, 则$f(\alpha_1^0, \alpha_2^0, \cdots, \alpha_t^0)$ 和$I(\alpha_1^0, \alpha_2^0, \cdots, \alpha_t^0)$ 分别是ψ 和E 的近似. 如$|f(\alpha_1^0, \alpha_2^0, \cdots, \alpha_t^0) - \psi| \leq \Delta$, 则$|I(\alpha_1^0, \alpha_2^0, \cdots, \alpha_t^0) - E| \leq \Delta^2$.

可以证明定态的能量本征值是泛函(7.2.6)的极小值, 即二阶变分$\delta^2 I > 0$, 特别是基态能量为(7.2.6)的绝对极小值. 实际应用中, 常常以一个比较粗糙的试探波函数, 就能够得到相当准确的基态能量. 因为基态能量是泛函(7.2.6)的对于最广义的变分$\delta\psi$ 的绝对极小值, 显然, 用上述近似的变分方法求出的基态能量近似值, 总比真正的基态能量高一些.

试探波函数所含参数愈多, 变化范围愈广, 则一般说来所得到的基态能量近似值愈接近于准确值, 但绝不会比它低.

7.3 Virial 定理和Hellmann-Feynman 定理

Virial 定理和Hellmann-Feynman 定理在量子力学用于原子及分子的研究中有重要的应用, 下面我们分别予以介绍.

设一量子系统的定态Schödinger 方程为

$$\hat{H}\psi = E\psi \tag{7.3.10}$$

又设$\hat{G}$ 为一不含时的线性算符, 则

$$\int \psi^*[\hat{G}, \hat{H}]\psi dq = 0 \tag{7.3.11}$$

这一结论称为超Virial 定理(Hyper-virial Theorem), 现在我们来证明这一论断: 因为ψ 满足方程(7.3.10), 所以

$$\begin{aligned}\int \psi^*[\hat{G}, \hat{H}]\psi dq &= \int \psi^*\hat{G}\hat{H}\psi dq - \int \psi^*\hat{H}\hat{G}\psi dq \\ &= E\int \psi^*\hat{G}\psi dq - E\int \psi^*\hat{G}\psi dq \\ &= 0\end{aligned}$$

式中利用了$\hat{H}$是Hermite 算符这一事实.

对于N 个粒子构成的系统, 粒子的直角坐标为$x_1, x_2, \cdots, x_{3N}$, 设

$$\hat{G} = \sum_{i=1}^{3N} \hat{x}_i\hat{p}_i = \frac{\hbar}{i}\sum_{i=1}^{3N} x_i\frac{\partial}{\partial x_i} \tag{7.3.12}$$

则

$$[\hat{G}, \hat{H}] = \sum_{i=1}^{3N}[\hat{x}_i\hat{p}_i, \hat{H}] = \sum_{i=1}^{3N}\hat{x}_i[\hat{p}_i, \hat{H}] + \sum_{i=1}^{3N}[\hat{x}_i, \hat{H}]\hat{p}_i \tag{7.3.13}$$

把Hamiltonian 写为

$$\hat{H} = \hat{T} + V$$

其中$\hat{T} = \sum_{i=1}^{3N}\frac{\hat{p}_i^2}{2\mu_i}$ 为系统的动能, 而V 为势能, 包括外场中的势能和相互作用势能, 只与系统中粒子的坐标有关. 则可容易证明:

$$[\hat{x}_i, \hat{H}] = \frac{p_i}{\mu_i} \quad [\hat{p}_i, \hat{H}] = -\frac{\partial V}{\partial x_i} \tag{7.3.14}$$

于是,

$$[\hat{G},\hat{H}]=\sum_{i=1}^{3N}\frac{\hat{p}_i^2}{\mu_i}+\sum_{i=1}^{3N}-\left(x_i\frac{\partial V}{\partial x_i}\right)=2\hat{T}-\sum_{i=1}^{3N}x_i\frac{\partial V}{\partial x_i} \tag{7.3.15}$$

利用(7.3.11) 可得

$$\int\psi^*2\hat{T}\psi dq-\int\psi^*\sum_{i=1}^{3N}x_i\frac{\partial V}{\partial x_i}\psi dq=0 \tag{7.3.16}$$

或, 用算符上面加一横代表对定态的平均值, 上式写为

$$2\overline{T}=\overline{\sum_{i=1}^{3N}x_i\frac{\partial V}{\partial x_i}} \tag{7.3.17}$$

这就是Virial 定理.

如果系统的势能是坐标$x_i,i=1,2,\cdots,3N$ 的n 次齐次函数, 则

$$\sum_{i=1}^{3N}x_i\frac{\partial V}{\partial x_i}=nV$$

Virial 定理可以简化为

$$2\overline{T}=n\overline{V} \tag{7.3.18}$$

又, 由于

$$\overline{T}+\overline{V}=E$$

所以有

$$\begin{cases}\overline{V}=\dfrac{2E}{n+2}\\ \overline{T}=\dfrac{nE}{n+2}\end{cases} \tag{7.3.19}$$

对于原子问题, 势能是坐标的(-1) 次齐次函数, 因此有

$$\begin{cases}\overline{V}=2E\\ \overline{T}=-E\\ 2\overline{T}=-\overline{V}\end{cases} \tag{7.3.20}$$

因为Virial 定理是严格的, 因此它可以用来作为检验近似方法的近似程度的标准之一, 如果用某种近似方法求得了原子系统的波函数, 就可以用这些波函数计算$\overline{T},\overline{V}$ 和E, 如果其不满足(7.3.20), 则计算结果显然不可靠.

Hellmann-Feynman 定理是在1937—1939年分别由H. Hellmann 和R. P. Feynman 独立提出的. 这一定理的内容为: 若ψ为Hamiltonian $\hat{H}$ 的归一化本征函数, E 是相应的本征能量, 而λ 是出现在$\hat{H}$ 中的任何一个参数, 则

$$\frac{\partial E}{\partial\lambda}=\int\psi^*\frac{\partial\hat{H}}{\partial\lambda}\psi dq \tag{7.3.21}$$

λ 可以是核间距, 电荷等. 这一定理可证明如下:

$$E = \int \psi^* \hat{H} \psi dq$$

由上式对λ 求偏微商, 得

$$\frac{\partial E}{\partial \lambda} = \int \frac{\partial \psi^*}{\partial \lambda} \hat{H} \psi dq + \int \psi^* \frac{\partial \hat{H}}{\partial \lambda} \psi dq + \int \psi^* \hat{H} \frac{\partial \psi}{\partial \lambda} dq$$

利用Schödinger 方程$\hat{H}\psi = E\psi$ 及$\hat{H}$ 的Hermite 性, 可得

$$\begin{aligned} \frac{\partial E}{\partial \lambda} &= \int \psi^* \frac{\partial \hat{H}}{\partial \lambda} \psi dq + E \frac{\partial}{\partial \lambda} \int \psi^* \psi dq \\ &= \int \psi^* \frac{\partial \hat{H}}{\partial \lambda} \psi dq \end{aligned}$$

上式的最后一个等式使用了波函数的归一化关系.

作为Hellmann–Feynman 定理的应用, 我们来看二个例子.

对于一维谐振子, Hamiltonian $\hat{H}$ 和能量E_n 分别为

$$\begin{aligned} \hat{H} &= -\frac{\hbar^2}{2\mu} \frac{d^2}{dx^2} + \frac{1}{2} \mu \omega^2 x^2 \\ E_n &= \left(n + \frac{1}{2} \right) \hbar \omega \end{aligned}$$

选ω 为参数, 得

$$\left(n + \frac{1}{2} \right) \hbar = \int \psi_n^* \mu \omega x^2 \psi dx$$

由此求得x^2 在定态ψ_n 的平均值为

$$\overline{x^2} = \left(n + \frac{1}{2} \right) \frac{\hbar}{\mu \omega}$$

这样, 就省去了对波函数的积分运算.

在中心力场中的粒子, 能量本征态可以取为$\left(\hat{H}, \hat{\mathrm{l}}^2, \hat{l}_z \right)$ 的共同本征态, 即

$$\psi = R(r) Y_{lm}(\theta, \phi) = \frac{\chi(r)}{r} Y_{lm}(\theta, \phi)$$

$\chi(r)$ 满足径向方程

$$\left(-\frac{\hbar^2}{2\mu} \frac{d^2}{dr^2} + V(r) + \frac{l(l+1)\hbar^2}{2\mu r^2} \right) \chi(r) = E\chi(r)$$

这一方程与相当于Hamiltonian 为

$$\hat{H} = -\frac{\hbar^2}{2\mu} \frac{d^2}{dr^2} + V(r) + \frac{l(l+1)\hbar^2}{2\mu r^2}$$

的一维定态Schödinger 方程. 把径向方程的本征值记为$E_{n_r l}$, 它依赖于径向量子数$n_r(=0,1,2,\cdots)$ 和角量子数$l(=0,1,2,\cdots)$. 视l 为参数,

$$\overline{\frac{\partial \hat{H}}{\partial l}} = (2l+1)\frac{\hbar^2}{2\mu}\overline{\frac{1}{r^2}} > 0$$

(因为$\frac{1}{r^2}$ 为正定算符). 因此, 按照Hellmann–Feynman 定理

$$\frac{\partial E_{n_r l}}{\partial l} > 0$$

即给定n_r 的情况下, $E_{n_r l}$ 随l 增大而增大, 因此, 中心力场的基态必为s 态$(l=0)$

7.4 轨道近似和Hartree–Fock 方程

除了氢原子或类氢离子外, 即使像氦原子这样简单的原子, 我们也无法求出其本征函数和本征值的解析解, 因此, 在原子结构的定量研究中, 数值计算是十分基本的. 但是, 为了严格求解原子结构问题, 就要求解方程(7.1.5), 这是一个有$3Z$ 个变量的偏微分方程的本征值问题, 当Z 稍大时, 直接对其进行数值计算几乎是不可能的, 因此必须发展近似方法. 在《数学物理方法》课程中, 我们学过分离变量法, 通过把一个多变量方程分离为多个单变量方程, 求解过程可大大简化. 在原子结构计算中, 我们也希望用这一方法来简化我们的问题, 然而, 不幸的是, 方程(7.1.5)是不能分离变量的. 一般来说, 一个多体问题的Hamiltonian 如果可以写成对单体Hamiltonian 的求和形式, 则对应的定态Schödinger 方程是可分离的, 否则是不可分离的. 具体地说, 如果

$$\hat{H} = \sum_i \hat{h}_i \tag{7.4.22}$$

其中h_i 只与第i 个粒子的坐标有关, 则Schödinger 方程$\hat{H}\Psi = E\Psi$ 的解可以写为

$$\begin{aligned} E &= \textstyle\sum_i \varepsilon_i \\ \Psi &= \textstyle\prod_i \psi_i \end{aligned} \tag{7.4.23}$$

对于全同粒子, 还要考虑波函数的对称性, 即对于Bose 子, 取对称波函数而对于Fermi 子取反对称波函数. 电子是Fermi 子, 因此原子中电子的波函数应取反对称的形式. 尽管方程(7.1.5)是不能分离变量的, 但分离变量方法是如此的强有力, 我们仍然试图寻找分离变量形式的近似解, 通过使用变分法, 使近似解尽可能接近实际的解. 我们先给出计算方法的推导, 关于这一近似的理论问题将不做过多的讨论, 留给读者进一步钻研.

考虑具有Z 个电子的原子, 我们假定其波函数可写为分离变量的形式, 注意到电子是Fermi 子, 可把波函数写成Slater 行列式的形式, 这样反对称性可自动满足. 我们用$\phi_i(r)$ 表示单个电子的空间波函数, η 表示单个电子的自旋波函数, 则$\psi_i = \phi_i(r)\eta_i$ 为单

个电子的波函数, 通常称为电子轨道. 每个电子轨道可以容纳一个电子, 如果Z 个电子占据了第$i_1, i_2, \cdots, i_Z$ 个轨道, 则用电子轨道表示的Slater 行列式形式的原子波函数为

$$\Psi = \frac{1}{\sqrt{Z!}} \begin{vmatrix} \psi_{i_1}(r_1) & \psi_{i_1}(r_2) & \cdots & \psi_{i_1}(r_2) \\ \psi_{i_2}(r_1) & \psi_{i_2}(r_2) & \cdots & \psi_{i_2}(r_2) \\ \cdots & \cdots & \cdots & \cdots \\ \psi_{i_Z}(r_1) & \psi_{i_Z}(r_2) & \cdots & \psi_{i_Z}(r_2) \end{vmatrix} \tag{7.4.24}$$

在原子单位下, 原子的Hamiltonian 由(7.1.4) 给出, 把式(7.4.24)和(7.1.4)代入下式

$$E = \int \Psi^* \hat{H} \Psi d^3r_1 d^3r_2 \cdots d^3r_Z \tag{7.4.25}$$

经过一些较为繁琐但直接了当的代数计算, 可以求得

$$\begin{aligned} E &= \sum_{k=1}^{Z} \int \phi_{i_k}^*(r) \left(-\nabla^2 - \frac{2Z}{r} \right) \phi_{i_k}(r) d^3r \\ &\quad + \sum_{k=1}^{Z} \sum_{k'=1}^{Z} \int \phi_{i_k}^*(r_1) \phi_{i_k}(r_1) \frac{1}{|r_1 - r_2|} \phi_{i_{k'}}^*(r_2) \phi_{i_{k'}}(r_2) d^3r_1 d^3r_2 \\ &\quad - \sum_{k=1}^{Z} \sum_{k'=1}^{Z}{}' \int \phi_{i_k}^*(r_1) \phi_{i_{k'}}(r_1) \frac{1}{|r_1 - r_2|} \phi_{i_{k'}}^*(r_2) \phi_{i_k}(r_2) d^3r_1 d^3r_2 \end{aligned} \tag{7.4.26}$$

式中最后一项求和上的“$'$” 表示只对自旋相同的电子对求和. 把上式对ϕ^* 求变分, 并注意到约束条件$\int \phi^*(r)\phi(r)d^3r = 1$, 引入Lagrange 乘子$\varepsilon_i$, 令

$$\delta E - \delta \left(\sum_{k=1}^{Z} \varepsilon_{i_k} \int \phi_{i_k}^*(r) \phi_{i_k}(r) d^3r \right) = 0$$

得到如下形式的方程

$$\hat{h}_{i_k} \phi_{i_k} = \varepsilon_{i_k} \phi_{i_k} \tag{7.4.27}$$

其中

$$\begin{aligned} \hat{h}_{i_k} &= -\nabla^2 - \frac{2Z}{r} \\ &\quad + \sum_{k'=1}^{Z} \int \phi_{i_{k'}}^*(r_1) \phi_{i_{k'}}(r_1) \frac{2}{|r_1 - r|} d^3r_1 \\ &\quad - \frac{\sum_{k'=1}^{Z}{}' \int \phi_{i_{k'}}^*(r_1) \phi_{i_k}(r_1) \frac{2}{|r_1 - r|} d^3r_1 \phi_{i_k}^*(r) \phi_{i_{k'}}(r)}{\phi_{i_k}^*(r) \phi_{i_k}(r)} \end{aligned} \tag{7.4.28}$$

式(7.4.27)和(7.4.28) 称为Hartree-Fock 方程, 式(7.4.28) 为Hartree-Fock 近似下的单电子Hamiltonian, 其中第一项为单个电子的动能, 第二项为电子在核势场中的势能, 第三

项为其它电子对所考虑电子的直接库仑相互作用能(简称为库仑能), 第四项是由于电子的全同性导致的交换库仑相互作用能(简称为交换能).

我们立刻注意到, 虽然方程(7.4.27)是一个单粒子的Schödinger 方程, 但由于其Hamiltonian (7.4.28)中的库仑能和交换能与待求波函数ϕ 有关, 因此只能自洽求解, 可以先假定一组ϕ, 计算出式(7.4.28), 然后求解方程(7.4.27), 得到一组新的ϕ, 再利用这组新的ϕ计算(7.4.28) 并求解方程(7.4.27), 反复迭代, 直至二次迭代所得结果小于某一给定的误差限为止.

7.5 统计近似和X_α 方程

方程(7.4.28) 中前三项与下标i_k 无关, 第四项与i_k 有关. 为了进一步简化计算, 对第四项取平均, 即把该项代之以其对每一占据态的算术平均值

$$V_{ex} \equiv -\frac{1}{Z}\sum_{k=1}^{Z}\frac{\sum_{k'=1}^{\prime Z}\int \phi_{i_{k'}}^*(r_1)\phi_{i_k}(r_1)\frac{2}{|r_1-r|}d^3r_1\phi_{i_k}^*(r)\phi_{i_{k'}}(r)}{\phi_{i_k}^*(r)\phi_{i_k}(r)} \tag{7.5.29}$$

用$V_c(\mathbf{r})$ 记方程(7.4.28) 中的第三项, 则方程(7.4.28) 可以写为

$$\hat{h}_{i_k} = -\nabla^2 + V(\mathbf{r}) \tag{7.5.30}$$

其中

$$V(\mathbf{r}) = -\frac{2Z}{r} + V_c(\mathbf{r}) + V_{ex}(\mathbf{r}) \tag{7.5.31}$$

如果$V(\mathbf{r})$ 具有中心对称性, 则方程(7.4.27) 的求解可以大为简化. 为此, 我们对式(7.5.31)作方向平均, 使其只依赖于径向坐标r, 在这一近似下, 方程(7.4.27) 变为

$$[-\nabla^2 + V(r)]\phi(\mathbf{r}) = \varepsilon\phi(\mathbf{r}) \tag{7.5.32}$$

上式的解可以写为

$$\phi(\mathbf{r}) = R(r)Y_{lm}(\theta,\varphi)$$

而$R(r)$ 满足下面的径向方程

$$-\frac{1}{r}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) + V(r)R + \frac{l(l+1)}{r^2}R = \varepsilon R \tag{7.5.33}$$

$R(r)$ 称为径向波函数, 在做进一步的推导之前, 我们先对径向方程及其解做一些一般性的讨论.

方程是一个二阶常微分方程的本征值问题, 因为我们这里感兴趣的是束缚态问题, 由波函数的平方可积要求可知, 当$r \to \infty$, 应有$rR(r) \to 0$; 又由波函数的统计解释及二

阶微分方程的一般理论† 可知当$r \to 0$, $R(r) \sim r^l$, 或$rR(r) \to 0$. 在这些边界条件下, 方程(7.5.33) 的本征值只能取一些分立值, 可用径向量子数$n_r(= 0, 1, 2, 3, \cdots)$ 编号, 分别对应于$R(r)$ 无节点, 一个节点, 二个节点, ... 本征值只与(n_r, l) 有关, 而与量子数m 无关, 因此, 在中心力场近似下, 电子的轨道能量对于确定的l 是$2l + 1$ 重简并的. 在原子问题中, 通常用另外一套量子数来标志状态, 定义主量子数为

$$n = n_r + l + 1 \tag{7.5.34}$$

则主量子数可取值$n = 1, 2, 3, \cdots$, 对于给定的n, l 可取值$0, 1, 2, \cdots, n-1$, 通常用$s, p, d, \cdots$表示. 对一个确定的状态可以由四个量子数(n, l, m, s) 来标记, 其中s代表自旋, 可取向上和向下两个值. 把方程(7.5.33) 的本征值记为ε_{nls}, 本征函数记为$\phi_{nlms}(\mathbf{r}) = R_{nls}(r)Y_{lm}(\theta, \varphi)$. 所谓电子组态是指电子占据轨道的情况, 例如, $(1s)^2(2s)^2(2p)^6$ 代表一种电子组态, 它表明$n = 1, l = 0$ 的轨道有二个电子, $n = 2, l = 0$ 的轨道有二个电子, $n = 1, l = 1$ 的轨道有六个电子. 这种组态中的每个填充的简并轨道均已填满, 为闭壳层组态. 若有某些简并轨道没有填满, 则为开壳层组态, 例如$(1s)^2(2s)^2(2p)^3$ 就是一个开壳层组态.

现在考虑(7.5.31)的方向平均. 对于给定的电子组态, 电荷密度为

$$\rho_s(\mathbf{r}) = \sum_{nlm} W_{nlms} |\phi_{nlm}(\mathbf{r})|^2 \tag{7.5.35}$$

式中W_{nlms} 是轨道ϕ_{nlms} 的占据数, 当占据时取1, 未占据时取0. 由于在中心力场近似下相同(nl) 的态是简并的, 因此可以定义(nls) 态的占据数为

$$W_{nls} = \sum_m W_{nlms}$$

把式(7.5.35)对方向取平均, 得到方向平均后的电荷密度为

$$\begin{aligned} \rho_s(r) &= \frac{1}{4\pi} \int d\Omega \rho_s(\mathbf{r}) \\ &= \frac{1}{4\pi} \sum_{nlm} W_{nlms} R_{nls}(r)^2 \int d\Omega |Y_{lm}(\theta, \varphi)|^2 \\ &= \frac{1}{4\pi} \sum_{nl} W_{nls} R_{nls}(r)^2 \end{aligned} \tag{7.5.36}$$

总电荷密度为

$$\rho(r) = \rho_\uparrow(r) + \rho_\downarrow(r)$$

在给定的电子组态下, 库仑能为

$$V_c(\mathbf{r}) = \int d^3 r_1 \frac{2\rho(\mathbf{r})}{|\mathbf{r_1} - \mathbf{r}|} \tag{7.5.37}$$

†见曾谨言著《量子力学》卷I , P259

利用

$$\frac{1}{|\mathbf{r_1}-\mathbf{r}|}=\sum_{lm}\frac{4\pi}{2l+1}\frac{r_<^l}{r_>^{l+1}}Y_{lm}(\theta,\varphi)Y_{lm}(\theta_1,\varphi_1)$$

其中

$$r_<=\min(r,r_1)\quad r_>=\max(r,r_1)$$

则V_c 的方向平均为

$$\begin{aligned}V_c(r) &= \frac{1}{4\pi}\int d\Omega V_c(\mathbf{r})\\ &= \frac{1}{4\pi}\int d^3r_1 2\rho(\mathbf{r_1})\sum_{lm}\frac{4\pi}{2l+1}\frac{r_<^l}{r_>^{l+1}}Y_{lm}(\theta_1,\varphi_1)\int d\Omega Y_{lm}(\theta,\varphi)\\ &= 2\int d^3r_1\frac{\rho(\mathbf{r_1})}{r_>}\\ &= \frac{8\pi}{r}\int_0^r r_1^2\rho(r_1)dr_1+8\pi\int_r^\infty r_1\rho(r_1)dr_1 \qquad (7.5.38)\end{aligned}$$

在推导中用到了关系式$\int d\Omega Y_{lm}(\theta,\varphi)=\sqrt{4\pi}\delta_{l0}\delta_{m0}$.

交换项的处理要复杂得多, 通常的做法是, 先对均匀系统(即外场为0 的系统) 求出V_{ex} 依赖于电荷密度的函数关系(对于均匀系统, 电荷密度是一与位置无关的常数), 再把其中的电荷密度换成有外场的电荷密度而保持函数形式不变. 在取方向平均时, 也只要把电荷密度换成对方向平均的电荷密度即可. 这一过程叫做统计平均近似. 在均匀情形下, 可求出

$$V_{ex\uparrow}=-6\left(\frac{3}{4\pi}\rho_\uparrow\right)^{1/3} \qquad (7.5.39)$$

对自旋向下的电子也有相同的结果. 把上式中的$\rho_\uparrow$ 换成$\rho_\uparrow(r)$, 便得到了交换势的形式. 另一方面, 如果对式(7.4.26)中的交换项作统计平均近似, 然后取变分, 则得到的交换势为

$$V_{ex\uparrow}(r)=-6\frac{2}{3}\left(\frac{3}{4\pi}\rho\uparrow\right)^{1/3} \qquad (7.5.40)$$

与(7.5.39)相比, 多出一个因子$\frac{2}{3}$, 这表明统计平均与变分操作是不可对易的. 为此, Slater 建议把交换势写成

$$V_{ex\uparrow}(r)=-6\alpha\left(\frac{3}{4\pi}\rho\uparrow\right)^{1/3} \qquad (7.5.41)$$

其中$\frac{2}{3}\le\alpha\le 1$ 为一参数, 令

$$V_s(r)=-\frac{2Z}{r}+V_c(r)+V_{ex,s}(r) \qquad (7.5.42)$$

则方程

$$\left(-\nabla^2+V_s(r)\right)\phi_{nlm}=\varepsilon_{nls}\phi_{nlms} \qquad (7.5.43)$$

为决定电子轨道的方程, 由于在交换势中引入了一个参数α, 因此这一方程称为X_α 方程. 实际计算分为自旋限制和自旋极化两种, 在自旋限制计算中, 认为两种自旋的能级是简并的且密度相同, 并具有相同的交换势, 取为

$$V_{ex}(r) = -6\alpha \left(\frac{3}{8\pi}\rho\right)^{1/3} \tag{7.5.44}$$

在自旋极化计算中, 则是对自旋的两种取向分别求解X_α 方程.

通常确定α的方法有如下几种, 一种是调整α 使计算结果满足Viral 定理; 第二种方法是严格求解Hartree-Fock 方程, 并比较Hartree-Fock 方程的结果与X_α 方程的结果来确定α; 第三种是把α 看作一个变分参数, 通过使原子的总能量取极小来确定α. 上述三种方法所得到结果基本上是相同的.

现在的第一性计算都基于密度泛函理论和Kahn-Sham的实现格式。其中的交换关联势基本上来自对于均匀系统的计算，然后做各种各样的近似。最早，也是使用的最多的是局域密度近似（LDA），也就是把均匀系统的拟合表达式拿来，直接把其中的密度换成与位置有关的密度。在此基础上的改进包括梯度展开，GW方法，等等。对于强关联系统，通过引进参数，用所谓+U的方法，也能得到较好的结果。这一方面的文献非常多，有兴趣进一步探讨的同学可以参考相关文献。

7.6 径向方程的求解方法

方程(7.5.43) 的解可写为$\phi_{nlm} = R_{nl}(r)Y_{lm}(\theta,\varphi)$,[‡] 其中$R_{nl}(r)$ 满足下面的径向Schödinger方程:

$$-\frac{1}{r}\frac{d}{dr}\left(r^2\frac{dR_{nl}}{dr}\right) + V(r)R_{nl} + \frac{l(l+1)}{r^2}R_{nl} = \varepsilon_{nl}R_{nl} \tag{7.6.45}$$

这一节我们介绍求解径向方程(7.6.45)的数值方法.

令$R_{nl}(r) = \frac{y_{nl}}{r}$ 并代入方程(7.6.45), 我们得到

$$\frac{d^2y_{nl}}{dr^2} = F_{nl}(r)y_{nl} \tag{7.6.46}$$

其中

$$F_{nl}(r) = \frac{l(l+1)}{r^2} + [V(r) - \varepsilon_{nl}]$$

为了简化符号, 在不引起误解的情况下, 后面的讨论中我们将省略下标(nl). 方程(7.6.46)可用Numerov 格式求解, 下面介绍这一方法. 为了更一般起见, 考虑在区间$[a,b]$ 上求解二阶微分方程

$$\frac{d^2y}{dr^2} = F(r)y + G(r) \tag{7.6.47}$$

[‡]在本节中, 为简单计, 将略去自旋下标, 需要时, 很容易在最终结果中恢复

以h 为步长划分区间$[a,b]$, 令$r_i = a + ih, i = 0, 1, 2, \cdots$, 记$y_i \equiv y(r_i)$ 则:

$$\begin{aligned} y_{i+1} &= y_i + hy_i' + \frac{h^2}{2!}y_i'' + \frac{h^3}{3!}y_i^{(3)} + \frac{h^4}{4!}y_i^{(4)} + \frac{h^5}{5!}y_i^{(5)} + O(h^6) \\ y_{i-1} &= y_i - hy_i' + \frac{h^2}{2!}y_i'' - \frac{h^3}{3!}y_i^{(3)} + \frac{h^4}{4!}y_i^{(4)} - \frac{h^5}{5!}y_i^{(5)} + O(h^6) \end{aligned} \tag{7.6.48}$$

上面二式相加得:

$$y_{i+1} - 2y_i + y_{i-1} = h^2\left(y_i'' + \frac{1}{12}h^2y_i^{(4)}\right) + O(h^6) \tag{7.6.49}$$

上式左边为y 在r_i 点的二阶中心差分, 记为

$$\Delta^2 y_i \equiv y_{i+1} - 2y_i + y_{i-1}$$

于是, (7.6.49) 可写为

$$\Delta^2 y_i = h^2\left(y_i'' + \frac{1}{12}h^2y_i^{(4)}\right) + O(h^6) \tag{7.6.50}$$

对于y'', 显然有相同的关系:

$$\Delta^2 y_i'' = h^2\left(y_i^{(4)} + \frac{1}{12}h^2y_i^{(6)}\right) + O(h^6) \tag{7.6.51}$$

由式(7.6.50) 和(7.6.51) 可得,

$$\Delta^2 y_i = h^2(y_i'' + \frac{1}{12}\Delta^2 y_i'') + O(h^6) \tag{7.6.52}$$

又, 由于$y_i'' = F_iy_i + G_i$, $\Delta^2 y_i'' = \Delta^2(F_iy_i + G_i)$, 式中$F_i \equiv F(r_i)$, $G_i \equiv G(r_i)$, 从而

$$\Delta^2 y_i = h^2\left[F_iy_i + G_i + \frac{1}{12}\left(F_{i+1}y_{i+1} + G_{i+1} + F_{i-1}y_{i-1} + G_{i-1} - 2F_iy_i - 2G_i\right)\right]$$

整理得

$$\left(1 - \frac{h^2}{12}F_{i+1}\right)y_{i+1} - 2\left(1 - \frac{h^2}{12}F_i\right)y_i + \left(1 - \frac{h^2}{12}F_{i-1}\right)y_{i-1} = h^2\left(F_iy_i + G_i + \frac{1}{12}\Delta^2 G_i\right)$$

令$Y = \left(1 - \frac{h^2}{12}F\right)y$, 最后得到

$$Y_{i+1} - 2Y_i + Y_{i+1} = h^2\left(\frac{F_i}{1 - \frac{h^2}{12}F_i}Y_i + G_i + \frac{1}{12}\Delta^2 G_i\right) \tag{7.6.53}$$

上式就是Numerov 差分格式, 这是一个二步方法, 即为了得到Y_{i+1}, 需要知道Y_i 和Y_{i-1} 的值.

径向方程不含对y 的一阶导数, 是方程(7.6.47)当$G(r) = 0$ 时的特例, 可用前述Numerov 方法求解. 具体计算时, 可分别从两个边界点出发, 逐步向中间计算. 从$r = 0$ 出发, 向r 增加的方向的计算过程称为外向计算, 从$r = \infty$ 出发(实阶上是一有限的足够大的值), 向r 减小的方向的计算过程称为内向计算. 外向计算和内向计算应该在某一中间点$r = r_m$ 处光滑的连接起来, 使二者光滑连接的过程称为外向计算与内向计算的匹配.

对于给定的(nl), 径向方程的解应该有$n_r = n - l - 1$ 个节点, 由径向方程可知, 解的节点只能出现在$F(r) < 0$ 的区域, 这是由于在节点上有$y = 0$ 及$y'' = 0$, 因此, 节点也是函数$y(r)$ 的拐点, 在节点的外侧, 随着r 的增大, $y(r)$ 或是趋于另一节点, 或者随$r \to \infty$ 而趋于0. 如果在节点外侧$y > 0$, 因要求$y(r)$ 是上凸的(否则将不断增长), 所以应有$y'' < 0$; 反之, 如果在节点外侧$y < 0$, 因要求$y(r)$ 是下凸的, 所以应有$y'' > 0$. 不论何种情况, 都要求y 与y'' 异号, 这只有当$F(r) < 0$ 时才有可能. 一般r_m 取在y 的最后一个节点之外, 即$F(r) > 0$ 处. 为了进行外向积分, 我们还需要知道最初二个点的y 的数值. $r = 0$ 时, 径向方程无定义, 积分可从$r = h$ 开始. 当$r \to 0$ 时, $V(r) \to \frac{-2Z}{r}$, 这是类氢离子的势, 因此, 我们可以用类氢离子的径向波函数(除以r)在$r = h, r = 2h$ 的值作为y_1, y_2. 给定一个ε_{nl} 的数值, 作外向积分至r_m (注意, r_m 一般与ε_{nl} 有关), 如果节点的数目不等于n_r, 说明ε_{nl} 的选择不好, 若节点数大于n_r, 说明$|\varepsilon_{nl}|$ 太小, 此时可用1.25 或其它大于1 的合适数字乘以ε_{nl}, 继续计算; 若节点数小于n_r, 说明$|\varepsilon_{nl}|$ 太大, 此时可用0.75 或其它小于1 的合适数字乘以ε_{nl}, 继续计算. 重复上述过程直至节点数目正好为n_r 时为至, 这一过程称为对ε_{nl} 的粗调. 在作完ε_{nl}的粗调后, 可交替进行向内计算和向外计算, 通过要求在r_m 点y 的光滑连接再细调ε_{nl}. 向内积分可取一足够大的r_∞ 开始计算, 为了适用于离子的计算, 我们假定原子的总电子数不必等于Z, 设电子数为N, 则当$r \to \infty$ 时, $V(r) \to -2\frac{Z-N+1}{r}$, $F(r) \to -\varepsilon_{nl}$, 径向方程的解为$y(r) \to \exp(-\sqrt{-\varepsilon_{nl}}r)$, 在$r_\infty$ 附近, 可取

$$y(r) = \exp(-\sqrt{F(r_\infty)}r)$$

并用Numerov 方法作内向积分, 为了y 在r_m 点光滑连接, 应要求外向积分和内向积分的结果在r_m 点的$\frac{y'}{y}$ 连续. 这可以通过细调ε_{nl} 来做到. 若记y_{in} 为内向积分的结果, y_{out} 为外向积分的结果, 在计算出y_{in}和y_{out}之后, 计算

$$\Delta\varepsilon_{nl} = \frac{1}{M}\left[\frac{y'_{out}(r_m)}{y_{out}(r_m)} - \frac{y'_{in}(r_m)}{y_{in}(r_m)}\right] \tag{7.6.54}$$

其中

$$M = \left[\frac{\int_0^{r_m} y^2_{out}(r)dr}{y^2_{out}(r_m)} + \frac{\int_{r_m}^{\infty} y^2_{in}(r)dr}{y^2_{in}(r_m)}\right]$$

把ε_{nl} 调整成$\varepsilon_{nl} + \Delta\varepsilon_{nl}$ 再重复上述计算, 直到$\left|\frac{\Delta\varepsilon_{nl}}{\varepsilon_{nl}}\right|$ 小于某一误差限时, 结束叠代.

最后, 我们给出式(7.6.54)的证明, 把$F(r)$ 写为两项之和,

$$F(r) = F'(r) - \varepsilon_{nl}$$

则方程(7.6.46) 成为

$$y'' = F'(r)y - \varepsilon_{nl}y \tag{7.6.55}$$

当ε_{nl} 变为$\varepsilon_{nl} + \Delta\varepsilon_{nl}$ 时, 设y 变为$y + \Delta y$, 在线性近似下, 有

$$y'' + \Delta y'' = F'(r)y + F'(r)\Delta y - \varepsilon_{nl}y - \Delta\varepsilon_{nl}y - \varepsilon_{nl}\Delta y \tag{7.6.56}$$

将(7.6.56)与(7.6.55)相减得

$$\Delta y'' = F'(r)\Delta y - \Delta\varepsilon_{nl}y - \varepsilon_{nl}\Delta y \tag{7.6.57}$$

(7.6.57) 乘以y 与(7.6.55)乘以Δy 相减得

$$\Delta y y'' - \Delta y'' y = \Delta\varepsilon_{nl}y^2 \tag{7.6.58}$$

或

$$\frac{d}{dr}(\Delta y y' - \Delta y' y) = \Delta\varepsilon_{nl}y^2$$

两边积分得

$$(\Delta y y' - \Delta y' y)|_{r_1}^{r_2} = \Delta\varepsilon_{nl}\int_{r_1}^{r_2} y^2 dr$$

分别用$(0, r_m)$ 及(r_m, ∞) 代替(r_1, r_2), 并注意到

$$\Delta\frac{y'}{y} = \frac{y\Delta y' - \Delta y y'}{y^2}$$

我们得到

$$\begin{aligned}
\Delta\frac{y'_{out}(r_m)}{y_{out}(r_m)}y^2_{out}(r_m) &= -\Delta\varepsilon_{nl}\int_0^{r_m} y^2_{out}(r)dr \\
\Delta\frac{y'_{in}(r_m)}{y_{in}(r_m)}y^2_{in}(r_m) &= \Delta\varepsilon_{nl}\int_{r_m}^{\infty} y^2_{in}(r)dr
\end{aligned} \tag{7.6.59}$$

若要求对应于$\varepsilon_{nl} + \Delta\varepsilon_{nl}$ 的解满足光滑连接, 即

$$\frac{y'_{out}}{y_{out}} + \Delta\frac{y'_{out}}{y_{out}} = \frac{y'_{in}}{y_{in}} + \Delta\frac{y'_{in}}{y_{in}} \tag{7.6.60}$$

把(7.6.59) 代入(7.6.60), 就得到方程(7.6.54)

前面我们讨论了径向方程的求解方法, 在讨论中假定$V(r)$ 是已知的, 事实上, $V(r)$ 是与径向方程的解有关的, 具体关系由式(7.5.38)及式(7.5.41)给出. 因此, 实阶计算时, 应先给定一初始势和初始本征值ε_{nl}, 然后求解径向方程求得一组本征值及本征函数, 用求得的本征函数代入式(7.5.38)及式(7.5.41)求得新的势$V(r)$, 用新的势和老的势的适当混合做为势函数, 再求解径向方程, 这样反复叠代直至前后两次势之差小于某一给定的误差限为止, 这一过程称为自洽过程, 一般来说, 如果初始势选择的较好, 只要15次左右的叠代便可达到很高的计算精度.

7.7 计算程序

这一节我们给出一个完整的计算原子结构的程序, 它用我们在前面几节描述的X_α 方法计算原子的总能量, 轨道能量及原子周围的势$V(r)$ 和电荷密度$\rho(r)$等.

第一个比较完整的原子结构计算程序是由Herman-Skillman编写的，此后的几乎所有原子结构计算程序基本上都是在这个程序的基础修改的。这里所附程序是一个非常接近于最初程序的版本。很多新的发展并没有包含进去。包含了最新交换关联势的一些程序通常都作为大型第一性原理计算包的一个辅助程序，如果对于精确计算原子结构有兴趣，请参看这类程序。

下面是P 原子计算的输入文件, 分别对应于自旋限制和自旋极化的计算. 现对每一参数解释如下, 第一行为描述性文字; 第二行(格式为2F10.5)为所用的α 的数值和每次叠代时所得的势混入老的势中的分数; 第三行(格式为10I5) 给出计算控制值, 分列如下: 第一个数: 输出文件打印控制, $=0$ 时打印每次叠代势的最大变化点和轨道能量, 收敛的总能量, 径向网格点, 电荷密度. $=1$ 还打印每次叠代时计算轨道能量的过程信息.
第二个数: 起始势的输入方式, 本程序中总取为1.
第三个数: 势收敛的指数阈值, 取-3 或-4.
第四个数: 轨道能量收敛的指数阈值, 取-5 或-6.
第五个数: 原子径向网格点的数目, 本程序中取为441.
第六个数: 允许的叠代次数, 取为50 左右.
第七个数: 原子序数.
第八个数: 原子的电荷数(即原子序数与所带电子数之差), 对中性原子取0.
第九个数: 轨道能级的数目.
第十个数: $=1$ 表示自旋限制计算; $=2$ 表示自旋极化计算.

第四行及紧跟的10行(对自旋限制计算) 或21行(对自旋极化计算) 为起始势的输入值.

紧限起始势的输入每行代表一个能级, 第一个数表示(nl), 以$100\times n+10\times l$ 表示; 第二个数对自旋限制计算取1 , 对自旋极化计算时取1 表示自旋向上, 取2 表示自旋向下; 第三个数为能级上的电子占据数; 第四个数为估计的轨道能量.

下面的两个输入文件分别计算磷的自旋限制基态和自旋极化基态及对应的轨道能量的输出值(能量以Rydberg 为单位).

```
Phosphorus Spin_Restricted state
 .72569       .20
 1    1   -4   -6  441   60   15    0    5    1
1.000  0.995  0.900  0.985  0.980  0.975  0.970  0.965  0.960  0.955
0.950  0.945  0.940  0.935  0.930  0.925  0.920  0.915  0.910  0.905
0.900  0.895  0.890  0.885  0.880  0.875  0.870  0.865  0.860  0.855
0.850  0.845  0.840  0.835  0.830  0.825  0.820  0.815  0.810  0.800
0.795  0.790  0.785  0.780  0.775  0.770  0.765  0.760  0.755  0.750
```

```
 0.745  0.740  0.735  0.730  0.725  0.720  0.715  0.710  0.705  0.700
 0.695  0.690  0.685  0.680  0.675  0.670  0.665  0.660  0.655  0.650
 0.645  0.640  0.635  0.630  0.625  0.620  0.615  0.610  0.605  0.600
 0.595  0.590  0.585  0.580  0.575  0.570  0.565  0.560  0.555  0.550
 0.545  0.540  0.535  0.530  0.525  0.520  0.515  0.510  0.505  0.500
 0.495  0.490  0.485  0.480  0.475  0.470  0.465  0.460  0.455  0.450
100    1  2.0      -195.0
200    1  2.0       -41.8
210    1  6.0       -40.0
300    1  2.0       -20.0
310    1  3.0       -18.0


                  ****   ATOMIC ENERGIES    ****
                  ***********************************

                  Kinetic Energy=           .6813768D+03
                  Nuc-El Coulomb=          -.1624688D+04
                  El--El  Energy=           .3071392D+03
                  Spin up Exchg =          -.4507193D+02


                  Total  Energy =          -.6812438D+03
                  Virial ratio  =           .1999805D+01


  spin-orbitals   nl  spin  ocup           E
                  100    1    2.0       -152.6651305
                  200    1    2.0        -12.7209732
                  210    1    6.0         -9.2141014
                  300    1    2.0          -.9833246
                  310    1    3.0          -.3684071



Phosphorus Spin_Polarized state
  .72569       .20
  1    1   -4   -6  441   60   15    0    9    2
 1.000  0.995  0.900  0.985  0.980  0.975  0.970  0.965  0.960  0.955
 0.950  0.945  0.940  0.935  0.930  0.925  0.920  0.915  0.910  0.905
 0.900  0.895  0.890  0.885  0.880  0.875  0.870  0.865  0.860  0.855
 0.850  0.845  0.840  0.835  0.830  0.825  0.820  0.815  0.810  0.800
 0.795  0.790  0.785  0.780  0.775  0.770  0.765  0.760  0.755  0.750
 0.745  0.740  0.735  0.730  0.725  0.720  0.715  0.710  0.705  0.700
 0.695  0.690  0.685  0.680  0.675  0.670  0.665  0.660  0.655  0.650
 0.645  0.640  0.635  0.630  0.625  0.620  0.615  0.610  0.605  0.600
 0.595  0.590  0.585  0.580  0.575  0.570  0.565  0.560  0.555  0.550
 0.545  0.540  0.535  0.530  0.525  0.520  0.515  0.510  0.505  0.500
 0.495  0.490  0.485  0.480  0.475  0.470  0.465  0.460  0.455  0.450
 1.000  0.995  0.900  0.985  0.980  0.975  0.970  0.965  0.960  0.955
 0.950  0.945  0.940  0.935  0.930  0.925  0.920  0.915  0.910  0.905
 0.900  0.895  0.890  0.885  0.880  0.875  0.870  0.865  0.860  0.855
 0.850  0.845  0.840  0.835  0.830  0.825  0.820  0.815  0.810  0.800
 0.795  0.790  0.785  0.780  0.775  0.770  0.765  0.760  0.755  0.750
 0.745  0.740  0.735  0.730  0.725  0.720  0.715  0.710  0.705  0.700
 0.695  0.690  0.685  0.680  0.675  0.670  0.665  0.660  0.655  0.650
```

```
 0.645  0.640  0.635  0.630  0.625  0.620  0.615  0.610  0.605  0.600
 0.595  0.590  0.585  0.580  0.575  0.570  0.565  0.560  0.555  0.550
 0.545  0.540  0.535  0.530  0.525  0.520  0.515  0.510  0.505  0.500
 0.495  0.490  0.485  0.480  0.475  0.470  0.465  0.460  0.455  0.450
100    1  1.0      -195.0
200    1  1.0       -41.8
210    1  3.0       -40.0
300    1  1.0       -20.0
310    1  3.0       -18.0
100    2  1.0      -195.0
200    2  1.0       -41.8
210    2  3.0       -40.0
300    2  1.0       -20.0


                  ****   ATOMIC ENERGIES    ****
                  ************************************

                  Kinetic Energy=          .6815644D+03
                  Nuc-El Coulomb=         -.1625375D+04
                  El--El  Energy=          .3076826D+03
                  Spin up Exchg =         -.2386447D+02
                  Spin Dn Exchg =         -.2143792D+02


                  Total  Energy =         -.6814307D+03
                  Virial ratio  =          .1999804D+01


   spin-orbitals   nl  spin  ocup           E
                  100    1    1.0       -152.6333667
                  200    1    1.0        -12.6974850
                  210    1    3.0         -9.1942103
                  300    1    1.0         -1.0708087
                  310    1    3.0          -.4509173
                  100    2    1.0       -152.6050145
                  200    2    1.0        -12.6619486
                  210    2    3.0         -9.1503035
                  300    2    1.0          -.7940065
```

下面是程序清单

```
C*****************************************************************
C
C              Atomic X-alpha Program
C
C*****************************************************************



C
      Implicit Real*8 (a-h,o-z)
      Logical watson,prnt,convg
      Dimension ichg(12)
      Dimension ihdr(20)
```

```
      Dimension dumcom(2084),u(521,2),unew(521,2)
      Dimension r(521),x(521)
      Dimension v(521,2),fn(521),a(521)
      Dimension po(521),p(521,10),rhosp(521,2),rho(521)
      Dimension nl(30),ocup(30),e(30),nspin(30),kmx(30),delta(2),
     *    kdel(2)
      Dimension rwk(521),vwk(521),pwk(521)
      Common dumcom,r,v,rhosp,rho,e,ocup,kmx,nspin,p
      Equivalence(dumcom(1),unew(1,1))
      Equivalence(dumcom(1043),fn(1),x(1))
      Equivalence(dumcom(1564),a(1))
      Data zero,one,two,three,four /0.0D0,1.0D0,2.0D0,3.0D0,4.0D0/
      Data pi /3.141592653589793D0/
      Data xincr /0.0025D0/
      Data ichg /40,80,120,160,200,240,280,320,360,400,440,480/
      Data watson,prnt,convg /.FALSE.,.FALSE.,.FALSE./

C
C*****************************************************************
C
C         Read Header Card
C*****************************************************************
C
      Open(5,File='fi')
      Open(6,File='out.dat')
      Read(5,5000) ihdr
      Write(6,5100) ihdr
C
C--------------------Read Input Parameters and Flags----------------
C
      Read(5,5200) exfact,radion,ratio,fracu,iprt,key,ntol,nthrsh,
     *        mesh,maxit,nz,ion,nsts,nspins
      z=dfloat(nz)
      xion=dfloat(ion)
      facs=dfloat(nspins)
      tol=10.0D0**(ntol)
      thresh=10.0D0**(nthrsh)
      zion=xion
      twoion=xion+xion
      mesh=mesh-1
      Write(6,5300) nz,ion,nsts,nspins,exfact,radion,ratio,fracu,
     *    key,ntol,nthrsh,mesh
      If(radion.GT.zero) watson=.TRUE.
      If(iprt.NE.0) prnt=.TRUE.
      nblock=mesh/40
C
C*****************************************************************
C         Constant for Conversion from R(Ao) to Dimensionless
C           Thomas-Fermi radial variable x
C*****************************************************************
C
      ctfd=two*z**(one/three)*(four/(three*pi))**(two/three)
C
C----------Construct X and R Meshes----------------------------------
C
      k=1
```

```
      x(k)=zero
      r(k)=zero
      deltax=xincr
      Do 20 nb=1,nblock
        Do 10 i=1,40
          k=k+1
          x(k)=x(k-1)+deltax
          r(k)=x(k)/ctfd
   10   Continue
        deltax=deltax+deltax
   20 Continue
C
C-----110 point mesh potential read (key=1)-------------------------
C
      twoz=z+z
      Do 50 is=1,nspins
        Read(5,5400) (u(k,is),k=1,437,4)
        Do 30 k=1,437,4
          u(k,is)=-twoz*u(k,is)
   30   Continue
        u(441,is)=u(437,is)
        u(445,is)=u(437,is)
        n=9
        Do 40 k=1,437,4
          n=n-1
          If(n.LT.0) Then
            u(k+1,is)=(22.0D0*u(k,is)+11.0D0*u(k+4,is)
     *                        -u(k+8,is))/32.0D0
            u(k+2,is)=(10.0D0*u(k,is)+15.0D0*u(k+4,is)
     *                        -u(k+8,is))/24.0D0
            u(k+3,is)=( 6.0D0*u(k,is)+27.0D0*u(k+4,is)
     *                       -u(k+8,is))/32.0D0
            n=9
          Else
            u(k+1,is)=(21.0D0*u(k,is)+14.0D0*u(k+4,is)
     *                  -3.0D0*u(k+8,is))/32.0D0
            u(k+2,is)=( 3.0D0*u(k,is)+6.0D0*u(k+4,is)
     *                        -u(k+8,is))/8.0D0
            u(k+3,is)=( 5.0D0*u(k,is)+30.0D0*u(k+4,is)
     *                        -3.0D0*u(k+8,is))/32.0D0
          Endif
   40   Continue
   50 Continue
C
C
C----setup of V mesh: V(R)=U(R)/R,
C                  factor 2Z already absorbed in U(R)
C
      Do 80 is=1,nspins
        v(1,is)=-9.9D35
        kmax=min0(441,mesh)
        Do 60 k=2,kmax
          v(k,is)=u(k,is)/r(k)
   60   Continue
        If(mesh.GT.kmax) Then
          Do 70 k=442,mesh
```

```
            v(k,is)=-twoion/r(k)
   70     Continue
        Endif
   80 Continue
c
c------Read state(spin-orbital) Information
c
      spnup=zero
      spndn=zero
      Do 90 nst=1,nsts
        Read(5,5500) nl(nst),nspin(nst),ocup(nst),e(nst)
        If(nspin(nst).EQ.1) spnup=spnup+ocup(nst)
        If(nspin(nst).EQ.2) spndn=spndn+ocup(nst)
   90 Continue
c
c---Begin SCF iteration
      Do 270 iter=1,maxit
        Do 100 k=1,mesh
          rhosp(k,1)=zero
          rhosp(k,2)=zero
          rho(k)=zero
  100   Continue
c
c----Solve Schroedinger equation form each spin-orbital
c
        Do 160 nst=1,nsts
          netry=0
          etry=e(nst)
          n=nl(nst)/100
          l=nl(nst)/10-10*n
          is=nspin(nst)
          Do 110 kk=1,520
            rwk(kk)=r(kk+1)
            vwk(kk)=v(kk+1,is)
            pwk(kk)=po(kk+1)
  110     Continue
          Call numrov(etry,nst,z,n,l,rwk,vwk,pwk,ichg,kmax,mesh,
     *     thresh,ntries,prnt)
          Do 120 kk=2,521
            r(kk)=rwk(kk-1)
            v(kk,is)=vwk(kk-1)
            po(kk)=pwk(kk-1)
  120     Continue
          netry=max0(netry,ntries)
          kmax=kmax+1
c
c----Add this orbital density to spin density
c
          po(1)=zero
          If(is.NE.2) Then
            Do 130 k=1,kmax
              rhosp(k,1)=rhosp(k,1)+ocup(nst)*po(k)**2
  130       Continue
          Else
            Do 140 k=1,kmax
              rhosp(k,2)=rhosp(k,2)+ocup(nst)*po(k)**2
```

```
  140       Continue
          Endif
c
c---store P*R for this spin-orbital
c
          Do 150 k=1,mesh
            p(k,nst)=po(k)
  150     Continue
          kmx(nst)=kmax
          e(nst)=etry
  160   Continue
c
c---end of schroedinger loop
c     find total charge density and generate new potential
c
        Do 170 k=1,mesh
          rho(k)=rhosp(k,1)+rhosp(k,2)
  170   Continue
        Call vgener(z,exfact,facs,ichg,mesh,nsts,nspins,watson,
     *   radion,zion,ratio,convg)
c
c----find point of largest change in U potential
        delta(1)=zero
        delta(2)=zero
        Do 190 is=1,nspins
          Do 180 k=1,mesh
            dif=dabs(u(k,is)-unew(k,is))
            If(dif.GT.delta(is)) Then
              delta(is)=dif
              kdel(is)=k
            Endif
  180     Continue
          Write(6,5700) iter,delta(is),kdel(is)
  190   Continue
        delt=dmax1(delta(1),delta(2))
        If(convg) Goto 280
        If(tol.GT.delt) convg=.TRUE.
c
c--- SCF potential not converged: Calculate next trial pot--
c
        If(netry.LE.2) fracu=fracu+(one-fracu)/two
        If(fracu.GT.0.5D0) fracu=0.5D0
        Do 220 is=1,nspins
          Do 200 k=1,mesh
            u(k,is)=(one-fracu)*u(k,is)+fracu*unew(k,is)
  200     Continue
          Do 210 k=2,mesh
            vlast=v(k,is)
            v(k,is)=u(k,is)/r(k)
            unew(k,is)=v(k,is)-vlast
  210     Continue
  220   Continue
        unew(1,1)=zero
        unew(1,2)=zero
c
c---next trial eigenvalues from 1st order pert:: <p| DEL V |p>
```

```
c
        Do 260 nst=1,nsts
          is=nspin(nst)
          kmax=kmx(nst)
          Do 230 k=1,kmax
            fn(k)=unew(k,is)*p(k,nst)**2
  230     Continue
          Do 240 kk=1,520
            rwk(kk)=r(kk+1)
            vwk(kk)=fn(kk+1)
            pwk(kk)=a(kk+1)
  240     Continue
          Call integr(vwk,rwk,kmax,ichg,pwk,1)
          Do 250 kk=2,521
            r(kk)=rwk(kk-1)
            fn(kk)=vwk(kk-1)
            a(kk)=pwk(kk-1)
  250     Continue
          dele=a(kmax)
          e(nst)=e(nst)+dele
  260   Continue
        Write(6,5600) (nl(nst),nspin(nst),ocup(nst),
     *                 e(nst),nst=1,nsts)
  270 Continue
c
c---- end of SCF iteration loop
c
       Write(6,5800)
       Stop
  280 Continue
c
c--- successful convergence of potential
c
       ncards=mesh/8+1
       If(mesh.EQ.(8*ncards)) ncards=ncards-1
       Write(6,5600) (nl(nst),nspin(nst),ocup(nst),
     *                e(nst),nst=1,nsts)
       Write(6,5900)
c
c--- print R. V meshes
c
       kf=1
       Do 290 nc=1,ncards
         kl=kf+7
         If(kl.GT.mesh) kl=mesh
         Write(6,6500) kf,(r(k),k=kf,kl)
         kf=kl+1
  290 Continue
       Write(6,6000)
       Do 310 is=1,nspins
         Write(6,6100) is
         kf=1
         Do 300 nc=1,ncards
           kl=kf+7
           If(kl.GT.mesh) kl=mesh
```

```
          Write(6,6500) kf,(v(k,is),k=kf,kl)
          kf=kl+1
  300   Continue
  310 Continue
      If(iprt.GE.1) Then
c
c--- print radial functions
c
        Do 330 nst=1,nsts
          n=nl(nst)/100
          l=nl(nst)/10-10*n
          Write(6,6200) n,l,e(nst),ocup(nst),nspin(nst),kmx(nst)
          kmax=kmx(nst)
          kf=1
          Do 320 nc=1,ncards
            kl=kf+7
            If(kl.GT.kmax) kl=kmax
            Write(6,6500) kf,(p(k,nst),k=kf,kl)
            kf=kl+1
  320     Continue
  330   Continue
      Endif
c
c--- print charge density
c
      Write(6,6300)
      If(nspins.NE.1) Then
        Do 350 is=1,nspins
          If(is.EQ.2) Write(6,6700)
          Write(6,6400) is
          kf=1
          Do 340 nc=1,ncards
            kl=kf+7
            If(kl.GT.mesh) kl=mesh
            Write(6,6500) kf,(rhosp(k,is),k=kf,kl)
            kf=kl+1
  340     Continue
  350   Continue
      Endif
      Do 360 k=2,mesh
        rho(k)=r(k)*rho(k)
  360 Continue
      kf=1
      Write(6,6700)
      Write(6,6600)
      Do 370 nc=1,ncards
        kl=kf+7
        If(kl.GT.mesh) kl=mesh
        Write(6,6500) kf,(rho(k),k=kf,kl)
        kf=kl+1
  370 Continue
      Close(5)
      Close(6)
      Stop
c
 5000 Format(20A4)
```

```
 5100 Format(' ',20A4
     *  //T20,'*****     input parameters     *****'
     *   /20X,36('*')//)
 5200 Format(4F10.5/10I5)
 5300 Format(T30,'At  # =',I9/T30,'charge=',I9
     *   /T30,'# orbs=',I9
     *   /T30,'# spns=',I9
     *   /T30,'exfact=',F9.5
     *   /T30,'radion=',F9.5
     *   /T30,'ratio =',F9.5
     *   /T30,'fracu =',F9.5
     *   /T30,'key   =',I9
     *   /T30,'ntol  =',I9
     *   /T30,'nthrsh=',I9
     *   /T30,'mesh  =',I9//)
 5400 Format(F8.5,9F7.5)
 5500 Format(2I5,2F10.0)
 5600 Format(//T5,'spin-orbitals',T21,'nl',T25,'spin',T31,'ocup',
     *          T46,'E'
     *   /(T20,I3,T27,I1,T31,F4.1,T41,F13.7))
 5700 Format(//T19,'ITER #',I5,5X,'delta=',1PD14.7,2X,'- point',I4)
 5800 Format('MAXIMUM # OF ITERATIONS EXCEEDED')
 5900 Format('RADIAL MESH FOR THIS ATOM:'//)
 6000 Format('XALPHA V POTENTIAL FOR THIS ATOM:'//)
 6100 Format(//'   SPIN',I2//)
 6200 Format('SPIN-ORBITAL W/ N=,'I2,'   L=',I2,'   E=',1PD14.7,
     *    '  OCUP=',1X,F5.2,'   NSPIN=',I2,'   TO POINT',I5,
     *    ' : FORM',' R*RADLFN'/)
 6300 Format('1 CHARGE DENSITY AT THE MESH POINTS:'/)
 6400 Format(/'  SPIN',I2/)
 6500 Format(I5,1P8D16.6)
 6600 Format(/'  TOTAL CHARGE DENSITY, FORM RHO*(R**2) :'/)
 6700 Format('1')
      End
C
      Function blatt(e,p,r,vradl,hsqo12)
c*****************************************************************
c     find the first derivative of function generated by Numorov
c  interpolation,  the  radial  increment  between  the  five
c  interpolation  points must be constant.
c
c        r= radial mesh
c    vradl= radial potential
c        p= radial function obtained by integrating with energy
c           E through the radial potential vradl(k) on mesh R(k)
c    blatt= numerical derivative of p
c   see: jm blatt , j comp phys 1:385 (1967)
c*****************************************************************
c
      Implicit Real*8 (a-h,o-z)
      Dimension p(5),vradl(5),r(5)
c
      If(r(5)-r(1).LE.4.1D0*(r(2)-r(1))) Then
        h=dsqrt(12.D0*hsqo12)
        dif1=(p(4)-p(2))/2.D0
        dif2=(p(5)-p(1))/2.D0
```

```
        ddif1=((vradl(4)-e)*p(4)-(vradl(2)-e)*p(2))*hsqo12
        ddif2=((vradl(5)-e)*p(5)-(vradl(1)-e)*p(1))*hsqo12
        blatt=16.D0*(-dif1+37.D0*dif2/32.D0-37.D0*ddif1/5.D0
     *   -17.D0*ddif2/40.D0)/(21.D0*h)
        Return
       Endif
       Write(6,5000)
       Stop
 5000 Format(' ERROR -- BLATT DERIVATIVE FORMULA IS CALLED',
     *                        'OVER A MESH DOUBLEING INTERVAL')
       End
C
c
       Subroutine integr(f,r,kmx,ichg,a,md)
c ****************************************************************
c      integrates function F on 1-D mesh R by quadrtures, stores
c  partial integrals in A
c  See   K S Kunz, Numerical Analysis
c****************************************************************
c
       Implicit Real*8 (a-h,o-z)
       Dimension f(*),r(*),ichg(12),a(*)
       Data s720,s646,s456,s346,s264,s251,s106,s74,s30,s25,s24,
     *  s19,s11,s10,s9,s7,s6,s5,s4,s2 /720.D0,646.D0,456.D0,
     *  346.D0,264.D0,251.D0,106.D0,74.D0,30.D0,25.D0,24.D0,
     *  19.D0,11.D0,10.D0,9.D0,7.D0,6.D0,5.D0,4.D0,2.D0/
       Data zero /0.D0/
c
       h=r(2)-r(1)
       If(md.EQ.1) Then
c
c----begin integration at zero
c
        k0=0
        f0=zero
       Else
c
c--- begin integration at first point
        k0=1
        a(1)=zero
        f0=f(1)
       Endif
       n=1
       k0p1=k0+1
       k0p2=k0+2
       k0p3=k0+3
       k0p4=k0+4
c
c--- quadrature formulas q41(0),q41(1),q41(2) for first 3 points
c
       a(k0p1)=h*(s251*f0+s646*f(k0p1)-s264*f(k0p2)+s106*f(k0p3)
     *   -s19*f(k0p4))/s720
       a(k0p2)=a(k0p1)+h*(-s19*f0+s346*f(k0p1)+s456*f(k0p2)
     *   -s74*f(k0p3)+s11*f(k0p4))/s720
       a(k0p3)=a(k0p2)+h*(s11*f0-s74*f(k0p1)+s456*f(k0p2)
     *   +s346*f(k0p3)-s19*f(k0p4))/s720
```

```
      k0=k0p4
      Do 20 k=k0,kmx
        km1=k-1
        km2=k-2
        km3=k-3
        kich=k-ichg(n)
        If(kich.NE.1) Then
          If(kich.EQ.2) Goto 10
c
c--- main part of integration: q31(2)
c
          a(k)=a(km1)+h*(s9*f(k)+s19*f(km1)-s5*f(km2)+f(km3))/s24
          Goto 20
        Endif
c
c-------if mesh interval just changed , special formulas
c
        h=h+h
        a(k)=a(km1)+h*(s2*f(k)+s7*f(km1)-s4*f(km2)+f(km3))/s6
        Goto 20
   10   n=n+1
        a(k)=a(km1)+h*(s11*f(k)+s25*f(km1)-s10*f(km2)
     *      +s4*f(km3))/s30
   20 Continue
      If(mod(md,2).NE.0) Return
c
c---- if MD even, reverse A
c
      Do 30 k=1,kmx
        a(k)=a(kmx)-a(k)
   30 Continue
      Return
      End

c
c
      Subroutine numrov(e,nst,z,n,l,r,v,p,ichg,kmax,mesh,thresh,
     *    ntries,prnt)
c
c*****************************************************************
c
c      solves schodinger equation numerically by numerov method
c     V= partial potential including L*(L+1)/R**2
c     P= partial wave function in the form R*(Radlfn)
c   assume V=infinity, P= zero at origin
c*****************************************************************
c
      Implicit Real*8 (a-h,o-z)
      Logical cutset,onedun,prnt
      Dimension dumcom(2084)
      Dimension ichg(12),elast(30)
      Dimension r(520),p(520),vradl(520),v(520),a(520),fn(520)
      Common dumcom
      Equivalence(dumcom(1),vradl(1))
      Equivalence(dumcom(522),a(1))
      Equivalence(dumcom(1043),fn(1))
      Data ntrymx,nrstmx,nmtplm /50,10,20/
```

```
      Data zero,one,two,four,five,ten,twlve /0.D0,1.D0,2.D0,4.D0,
     *    5.D0,10.D0,12.D0/
c
      ntries=0
      nreset=0
      nmtply=0
      nodes=n-(l+1)
      angm=dfloat(l*(l+1))
      onedun=.FALSE.
c
c--- setup radial EQN potential (V plus ANGMOM pot. L)
c
      Do 10 k=1,mesh
        vradl(k)=v(k)+angm/(r(k)*r(k))
   10 Continue
      If(elast(nst).EQ.zero) elast(nst)=1.25D0*e
      delast=dabs((elast(nst)-e)/e)
      Goto 30
   20 Continue
c
c--- E has been reset non variationaly:
c                       reset DE-control parameter
c
      If(prnt) Write(6,5500) nst,ntries,e
      delast=dabs(0.25D0*e)
c
c-- outward integration: To start exponential tail
c                                (where curvature goes negtive)
c
   30 Continue
      ntries=ntries+1
      If(ntries.LE.ntrymx) Then
        gkm1=-one
        cutset=.FALSE.
        h=r(2)-r(1)
        ncross=0
        Call pstart(h,z,l,e,v,p(1),p(2))
        hsqo12=h*h/twlve
        pkm2=p(1)
        pkm1=p(2)
        dkm2=-(e-vradl(1))*pkm2*hsqo12
        dkm1=-(e-vradl(2))*pkm1*hsqo12
        i=1
        Do 50 k=3,mesh
          gk=(e-vradl(k))*hsqo12
          pk=(two*(pkm1+five*dkm1)-pkm2+dkm2)/(one+gk)
          p(k)=pk
c
c--- test for sign changes in WFN
c
          If(pk*pkm1.LT.zero) ncross=ncross+1
          If(.NOT.cutset) Then
c
c--- test for chg of sign in curvature  (outer turning point)
c
            If(gkm1.LT.zero.OR.gk.GE.zero) Then
```

```
              gkm1=gk
              Goto 40
            Endif
c
c--- Have reached turning point:
c                 set KSTOP for cutoff integration
c
            cutset=.TRUE.
            kstop=k+2
            If(kstop.EQ.ichg(i)-1) kstop=ichg(i)
            If(k-2.LT.ichg(i-1).AND.k+2.GT.ichg(i-1))
     *         kstop=ichg(i-1)+4
            If(k-2.LT.ichg(i).AND.k+2.GT.ichg(i))
     *         kstop=ichg(i)+4
          Else
c
c--- KSTOP set: check if it is reached
c
            If(k.EQ.kstop) Goto 60
          Endif
   40     If(k.GE.ichg(i)) Then
c
c--- numerov increment for H just doubled
c
            i=i+1
            hsqo12=four*hsqo12
            dkm2=four*dkm2
            dkm1=-four*gk*pk
            pkm1=pk
          Else
c
c---numerov increment for H unchanged
c
            dkm2=dkm1
            dkm1=-gk*pk
            pkm2=pkm1
            pkm1=pk
          Endif
   50   Continue
c
c--- Error exit from outwards integration: Tail not reached
c       Reset E=VRADL(k) inside mesh and try again
c
        If(prnt) Write(6,5600) nst,ntries,k
        nreset=nreset+1
        k=mesh-40*nreset
        e=vradl(k)
        If(nreset.GT.nrstmx) Goto 140
        Goto 20
c
c--- Successful exit from outwards integration:
c                                  Test # radial nodes
c
   60   Continue
        If(ncross.NE.nodes) Then
c
```

```
c---- # of nodes in error:  if too few, decrease ABS(E)
c                           if too many, increase ABS(E)
c
          If(prnt) Write(6,5700) nst,ntries,ncross
          nmtply=nmtply+1
          If(nmtply.GT.nmtplm) Goto 160
          If(ncross.LT.nodes) fact=0.75D0
          If(ncross.GT.nodes) fact=1.25D0
          e=fact*e
          Goto 20
        Endif
c
c--- # of nodes correct: find P from blatt formula
c
        drvout=blatt(e,p(kstop-4),r(kstop-4),
     *                             vradl(kstop-4),hsqo12)
        kmatch=kstop-2
        pmatch=p(kmatch)
c
c--- end outward integration
c
        kstop=kstop-4
        If(.NOT.onedun) Then
c
c--- skip searchfor kmax if one succesful inwards integral done
c
          bigex=69.08D0
          in=11
          kmax=mesh+40
   70     Continue
          in=in-1
          If(in.EQ.0) Goto 150
          kmax=kmax-40
          hsq=(h*h)*four**in
c
c--- set KMAX farther IN , if H too large for stabilit
c                          or RADLFN is in an underflow region
c
          If(hsq.GE.6.214D0/dabs(e)) Goto 70
          ex=r(kmax)*dsqrt(vradl(kmax)-e)
          If(ex.GT.bigex) Goto 70
          Do 80 k=kmax,mesh
            p(k)=zero
   80     Continue
        Endif
c
c--- inward integration
c
        k=kmax
        i=in
c
c-- first two points by exponential approx
c
        hsqo12=(h*h/twlve)*four**i
        pkm2=dexp(-r(k)*dsqrt(vradl(k)-e))
        dkm2=-(e-vradl(k))*pkm2*hsqo12
```

```
        p(k)=pkm2
        k=k-1
        pkm1=dexp(-r(k)*dsqrt(vradl(k)-e))
        dkm1=-(e-vradl(k))*pkm1*hsqo12
        p(k)=pkm1
c
c--- integrat down to kstop
c
   90   k=k-1
        gk=(e-vradl(k))*hsqo12
        If(-gk.GT.one) Write(6,5300) e,gk
        pk=(two*(pkm1+five*dkm1)-pkm2+dkm2)/(one+gk)
        p(k)=pk
        If(k.NE.kstop) Then
          If(k.LE.ichg(i)) Then
c
c--- increment if interval just halved: calcls 2 points
c
            i=i-1
            dk=-pk*gk
            km1=k-1
            km2=k-2
            gkm1=(e-vradl(k))*hsqo12
            pkm1=(two*(pk+five*dk)-pkm1+dkm1)/(one+gkm1)
            dkm1=-pkm1*gkm1/four
            hsqo12=hsqo12/four
            gkm2=(e-vradl(k))*hsqo12
            dk=dk/four
            pkm2=((pk-dk)+(pkm1-dkm1))/(two-ten*gkm2)
            dkm2=-pkm2*gkm2
            p(km1)=pkm2
            If(km1.EQ.kstop) Goto 100
            p(km2)=pkm1
            If(km2.EQ.kstop) Goto 100
            k=km2
            Goto 90
          Endif
c
c--- numerov invrement for H unchanged
c
          dkm2=dkm1
          dkm1=-pk*gk
          pkm2=pkm1
          pkm1=pk
          Goto 90
        Endif
c
c--- end inward integration
c---------------------------------------
  100   Continue
        onedun=.TRUE.
c
c--- mach inwards integratrion to ontwards at KMATCH
c
        cmatch=pmatch/p(kmatch)
        Do 110 k=kstop,mesh
```

```
          p(k)=p(k)*cmatch
  110   Continue
        drvin=blatt(e,p(kstop),r(kstop),vradl(kstop),hsqo12)
        drvdif=drvin-drvout
c
c--- variational principle improvment for E
c
        Do 120 k=1,kmax
          fn(k)=p(k)*p(k)
  120   Continue
        Call integr(fn,r,kmax,ichg,a,1)
        xnorm=a(kmax)
        de=-p(kmatch)*drvdif/xnorm
        If(prnt) Write(6,5800) nst,ntries,e,kmatch,drvdif,de
        If(dabs(de/e).GE.thresh) Then
c
c--- E not converged: if DE/E .gt. % chnage of ETRIAL from
c--- E of last potential iter, reduce DE to that value
c
          If(dabs(de/e).GT.delast) de=de*dabs(delast*e/de)
c
c--- increment energy and solve again
c
          e=e+de
          Goto 30
        Endif
c
c--- energy is converged: normalize WFN and return
c
        elast(nst)=e
        xnorm=dsqrt(xnorm)
        Do 130 k=1,kmax
          p(k)=p(k)/xnorm
  130   Continue
        Return
      Endif
c
c--- error prints and stop
c
      Write(6,5000)
      Stop
  140 Write(6,5100)
      Stop
  150 Write(6,5200)
      Stop
  160 Write(6,5400)
      Stop
c
c
 5000 Format('  ntrymx exceeded in numerov--',
     *                     'E convergence too slow')
 5100 Format('  cannot find an exponential tail in numerov',
     * 'despite moving curvature CHG point back to R(40).',
     *                                     'check form of V.')
 5200 Format('  starting mesh is too large to accommodate',
     *                'core function integration by numerov')
```

```
 5300 Format(' warning--- GK<-1 for E=',1X,D14.7,
     *                                  ' and GK=',1X,D16.7/)
 5400 Format(' E has been changed to hunt for correct # of',
     *                        'nodes more time than allowed')
 5500 Format('  NST=',I3,'  after  NTRY=',I3,'  is having E',
     *                   'set non-variationally to ',1X,D20.7)
 5600 Format('  Nst=',I3,'  Trial #',I3,'  does not reach tail',
     *                               'E reset= VRADL(k)-K=',I4)
 5700 Format('  NSt=',I3,'  Trial #',I3,'  has NCROSS=',I3,
     *          ': E mult by factor to correct # radial nodes')
 5800 Format('  NST=',I3,'  Trial # ',I3,'  has E=',1X,D14.7,
     *       ' Kmatch=', I3,'   Drvdif=',D14.7,'  De=',D14.7)
      End
c
      Subroutine pstart(h,z,l,e,v,p1,p2)
c*****************************************************************
c
c      Begin integration of  radial equation  with  angular
c   momentum = L, near R=0, expands
c         P=R*(radial FN)=SUM<0,4>(A(j)*R**j)*R**(L+1)
c   and finds coefts A(J) by taking derivatives 0-3 of radial
c   equation at R=0;
c
c*****************************************************************
      Implicit Real*8 (a-h,o-z)
      Dimension v(3)
      Data s1,s3o2,s11o3,s2,s5o2,s3,s4,s6,s8,s12,s20 /1.D0,1.5D0,
     * 3.6666666666666D0,2.D0,2.5D0,3.D0,4.D0,6.D0,8.D0,12.D0,
     * 20.D0/
c
      xl=dfloat(l)
      zoh=z/h
c
c-- derivs of R*(v(i)-E) at R=0
c
      d0=-(z+z)
      d1=s3*(v(1)-v(2))+v(3)+s11o3*zoh-e
      d2=-(s5o2*v(1)-s4*v(2)+s3o2*v(3)+zoh+zoh)/h
      d3=((v(1)+v(3))/s2-v(2)+zoh/s3)/(h*h)
c
c--- expansion coefficients
c
      a1=-z/(xl+s1)
      a2=(d0*a1+d1)/(s4*xl+s6)
      a3=(d0*a2+d1*a1+d2)/(s6*xl+s12)
      a4=(d0*a3+d1*a2+d2*a1+d3)/(s8*xl+s20)
      lp1=l+1
c
c---evaluation of series expansion for P=R*(radial function
c
      p1=(s1+h*(a1+h*(a2+h*(a3+h*a4))))*h**(lp1)
      th=h+h
      p2=(s1+th*(a1+th*(a2+th*(a3+th*a4))))*th**(lp1)
      Return
      End
c
```

```
      Subroutine vgener(z,exfact,facs,ichg,mesh,nsts,nspins,
     *    watson,radion,zion,ratio,convg)
c*****************************************************************
c     generate new atomic potential UNEW(k) from charge density
c  information. if CONVG=.true., also calculates total energy
c  under the HFS hamilton
c
c*****************************************************************
      Implicit Real*8 (a-h,o-z)
      Logical convg,watson
      Dimension dumcom(2084),fn(521),a(521)
      Dimension r(521),p(521),rhosp(521,2),rho(521)
      Dimension ucoul(521),uexch(521,2),unew(521,2),v(521,2)
      Dimension e(30),nspin(30),kmx(30),ocup(30)
      Dimension ichg(12),eexch(2)
      Dimension fwk(521),rwk(521),awk(521)
      Common dumcom,r,v,rhosp,rho,e,ocup,kmx,nspin,p
      Equivalence(dumcom(1),uexch(1,1),unew(1,1))
      Equivalence(dumcom(1043),fn(1))
      Equivalence(dumcom(1564),a(1))
      Data zero,one,two,three,six /0.D0,1.D0,2.D0,3.D0,6.D0/
      Data third,s3o4 /0.33333333333333D0,0.75D0/
      Data pi4 /12.56637061435916D0/

c
c---generate new potential U=R*V (rho,rhosp in form (R*P)**2)
c
      kmax=0
      Do 10 nst=1,nsts
        kmax=max0(kmax,kmx(nst))
   10 Continue
      kmxm1=kmax-1
      mesp=mesh-1
      Do 20 k=1,1563
        dumcom(k)=zero
   20 Continue
c
c--- exchange potential
c
      Do 40 is=1,nspins
        Do 30 k=1,kmax
          uexch(k,is)=-six*exfact*(three*facs*r(k)*rhosp(k,is)
     *     /(two*pi4*pi4))**third
   30   Continue
   40 Continue
      If(convg) Then
c
c--- if potential converged, find E-E exchange energy
c
        eexch(1)=zero
        eexch(2)=zero
        Do 80 is=1,nspins
          Do 50 k=2,kmax
            fn(k)=s3o4*rhosp(k,is)*uexch(k,is)/r(k)
   50     Continue
          Do 60 kk=1,520
```

```
            fwk(kk)=fn(kk+1)
            rwk(kk)=r(kk+1)
            awk(kk)=a(kk+1)
   60     Continue
          Call integr(fwk,rwk,kmxm1,ichg,awk,1)
          Do 70 kk=2,521
            fn(kk)=fwk(kk-1)
            r(kk)=rwk(kk-1)
            a(kk)=awk(kk-1)
   70     Continue

          eexch(is)=a(kmax)
   80   Continue
      Endif
c
c--coulomb potential (assuming sperical symmetry)
c
      ucoul(1)=zero
      Do 90 kk=1,520
        fwk(kk)=rho(kk+1)
        rwk(kk)=r(kk+1)
        awk(kk)=ucoul(kk+1)
   90 Continue
      Call integr(fwk,rwk,mesp,ichg,awk,1)
      Do 100 kk=2,521
        rho(kk)=fwk(kk-1)
        r(kk)=rwk(kk-1)
        ucoul(kk)=awk(kk-1)
  100 Continue

c      call integr(rho(2),r(2),nesp,ichg,ucoul(2),1)
      rho(1)=zero
      Do 110 k=2,kmax
        rho(k)=rho(k)/r(k)
  110 Continue
      Do 120 kk=1,520
        fwk(kk)=rho(kk+1)
        rwk(kk)=r(kk+1)
        awk(kk)=fn(kk+1)
  120 Continue
      Call integr(fwk,rwk,kmxm1,ichg,awk,1)
      Do 130 kk=2,521
        rho(kk)=fwk(kk-1)
        r(kk)=rwk(kk-1)
        fn(kk)=awk(kk-1)
  130 Continue
      Do 140 k=1,kmax
        fn(k)=fn(kmax)-fn(k)
  140 Continue
c
c--- UCOUL set equal electron-electron coulomb potential
c
      Do 150 k=1,mesh
        ucoul(k)=ucoul(k)+r(k)*fn(k)
  150 Continue
      If(convg) Then
```

```
c
c --- if potential converged, find N-E,& E-E coulomb energies
c
        enucel=-two*z*fn(1)
        Do 160 k=1,kmax
          fn(k)=rho(k)*ucoul(k)
  160   Continue
        Do 170 kk=1,520
          fwk(kk)=fn(kk+1)
          rwk(kk)=r(kk+1)
          awk(kk)=a(kk+1)
  170   Continue
        Call integr(fwk,rwk,kmxm1,ichg,awk,1)
        Do 180 kk=2,521
          fn(kk)=fwk(kk-1)
          r(kk)=rwk(kk-1)
          a(kk)=awk(kk-1)
  180   Continue
        eecoul=a(kmax)
      Endif
c
c---new coulomb potential (nuclear+electronic)
c
      Do 190 k=1,mesh
        ucoul(k)=two*(ucoul(k)-z)
        If(watson) Then
c
c-- watson sphere contribution, if any
c
          xion=zion*ratio
          If(r(k).LT.radion) Then
            ucoul(k)=ucoul(k)+two*xion*r(k)/radion
          Else
            ucoul(k)=ucoul(k)+two*xion
          Endif
        Endif
  190 Continue
c
c---sum coulomb and exange terms
c
      Do 210 is=1,nspins
        Do 200 k=1,mesh
          unew(k,is)=ucoul(k)+uexch(k,is)
  200   Continue
  210 Continue
      If(.NOT.convg) Then
        Return
      Endif
c
c--- Total Energy Calculation
c
      ekinet=zero
c
c--- Kinetic energy from integration (E-V) over orbital densities
c
      Do 220 nst=1,nsts
```

```
        ekinet=ekinet+ocup(nst)*e(nst)
  220 Continue
      Do 260 is=1,nspins
        Do 230 k=2,kmax
          fn(k)=v(k,is)*rhosp(k,is)
  230   Continue
        Do 240 kk=1,520
          fwk(kk)=fn(kk+1)
          rwk(kk)=r(kk+1)
          awk(kk)=a(kk+1)
  240   Continue
        Call integr(fwk,rwk,kmxm1,ichg,awk,1)
        Do 250 kk=2,521
          fn(kk)=fwk(kk-1)
          r(kk)=rwk(kk-1)
          a(kk)=awk(kk-1)
  250   Continue
        ekinet=ekinet-a(kmax)
  260 Continue
      ewats=zero
      If(watson) Then
c
c--- watson sphere contibution if any
c
        xion=zion*ratio
        Do 270 k=2,kmax
          If(r(k).LT.radion) Then
            fn(k)=two*xion*r(k)/radion
          Else
            fn(k)=two*xion
          Endif
  270   Continue
        Do 280 k=2,kmax
          fn(k)=fn(k)*rho(k)
  280   Continue
        Do 290 kk=1,520
          fwk(kk)=fn(kk+1)
          rwk(kk)=r(kk+1)
          awk(kk)=a(kk+1)
  290   Continue
        Call integr(fwk,rwk,kmxm1,ichg,awk,1)
        Do 300 kk=2,521
          fn(kk)=fwk(kk-1)
          r(kk)=rwk(kk-1)
          a(kk)=awk(kk-1)
  300   Continue
        ewats=a(kmax)
      Endif
c
c---SUM and Print Total energy
c
      etotal=ekinet+eecoul+eexch(1)+eexch(2)+enucel+ewats
      virial=-etotal/ekinet+one
      Write(6,5100)
      Write(6,5000) ekinet,enucel,eecoul,eexch(1),eexch(2),ewats,
     *    etotal,virial
```

```
      Return
 5000 Format(//T20,'****   ATOMIC ENERGIES    ****'
     * /T20,36('*')
     *  //T20,'Kinetic Energy=',1X,D21.7
     *   /T20,'Nuc-El Coulomb=',1X,D21.7
     *   /T20,'El--El  Energy=',1X,D21.7
     *   /T20,'Spin up Exchg =',1X,D21.7
     *   /T20,'Spin Dn Exchg =',1X,D21.7
     *   /T20,'Watson Sphere =',1X,D21.7
     * ///T20,'Total  Energy =',1X,D21.7
     *   /T20,'Virial ratio  =',1X,D21.7
     * ///)
 5100 Format('1')
      End
```

7.8 Sturm-Liouville 问题

Sturm-Liouville 问题包含了一大类常微分方程的本征值问题，其一般形式为

$$(p(x)y'(x))' + q(x)y(x) = s(x)$$

一大类物理相关的方程，如勒让德方程，贝塞尔方程等，都是这个方程的特殊形式。一种常见的特殊情形是$s(x) = 0$, 而$q(x)$中含有参数λ, 在给定的边界条件下，只有λ取特定数字时，方程才有解，这些特定值称为方程的本征值。

Sturm-Liouville 方程可以利用Numerov方法的推广来构造高精度格式。利用中心差分，

$$y'_i = \frac{y_{i+1} - y_{i-1}}{2h} - \frac{h^2 y_i^{(3)}}{6} + O(h^4)$$

$$y'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{h^2 y^{(4)}}{12} + O(h^4)$$

于是

$$p'_i \frac{y_{i+1} - y_{i-1}}{2h} + p_i \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = (p_i y'_i)' + \frac{h^2}{12}\left(p_i y_i^{(4)} + 2p'_i y_i^{(3)}\right) + O(h^4)$$

对LS方程求导两次，解出y的三阶和四阶导数，并把其中的二阶导数和一阶导数代之以差分形式，经过比较繁琐的代数运算，可以得到

$$c_{i+1} y_{i+1} = c_i y_i - c_{i-1} y_{i-1} + d_i + O(h^6)$$

其中

$$\begin{aligned} c_{i+1} =& 24p_i + 12hp'_i + 2h^2 q_i + 6h^2 p''_i - 4h^2 (p'_i)^2/p_i \\ &+ h^3 p_i^{(3)} + 2h^3 q'_i - h^3 p'_i q_i/p_i - h^3 p'_i p''_i/p_i \end{aligned}$$

$$\begin{aligned} c_{i-1} =& 24p_i - 12hp_i' + 2h^2q_i + 6h^2p_i'' - 4h^2(p_i')^2/p_i \\ & - h^3p_i^{(3)} - 2h^3q_i' + h^3p_i'q_i/p_i + h^3p_i'p_i''/p_i \end{aligned}$$

$$\begin{aligned} c_i =& 48p_i - 20h^2q_i + 12h^2p_i'' - 8h^2(p_i')^2/p_i \\ & + 2h^4q_i'' + 2h^4p_i'q_i'/p_i \end{aligned}$$

$$d_i = 24h^2s_i + 2h^4s_i'' - 2h^4p_i's_i'/p_i$$

当$p(x) = 1$时，就是我们前面处理过的方程。