

Computational Electromagnetics

讲义

Liu Benyuan

*The Department of Biomedical Engineering, Fourth Military Medical University, ChangLe
West Road, 710032, Xi'an, P. R. China. byliu@fmmu.edu.cn*

March 8, 2016

目 录

课程设计	1
教学心得	2
专用名词	3
第 1 章 电磁场特性及数学模型	4
1.1 目的	4
1.2 内容预览	4
1.3 讲义内容	4
1.3.1 矢量分析的简短教程	4
1.3.2 Maxwell 公式	6
1.3.3 位函数	9
1.3.4 初始条件与边界条件	10
1.3.5 数值计算方法	13
1.3.6 电磁参数	17
1.4 习题	17
第 2 章 离散方程组的计算机解法	18
2.1 目的	18
第 3 章 数值积分法	19
3.1 目的	19
3.2 讲义内容	19
3.2.1 介绍	20
3.2.2 中值法	21
3.2.3 梯形法	22
3.2.4 辛普森法	23
3.2.5 高斯积分法	24
3.2.6 计算示例	27

3.2.7 结论	33
3.3 习题	35
3.4 上机实验	35
第 4 章 有限差分法	36
4.1 目的	36
4.2 有限差分法	36
4.2.1 从微分到差分	36
4.2.2 交错格点	37
4.2.3 二维 laplace 方程的差分形式	38
4.2.4 计算示例	39
4.2.5 迭代法	42
4.2.6 迭代法与矩阵求解的关系（选讲）	45
4.2.7 网格间距与数值误差（选讲）	48
4.3 时域有限差分法	50
4.3.1 理论要点回顾	51
4.3.2 特征值方法	52
4.3.3 频域方法	53
4.3.4 时域更新计算方法	54
4.3.5 数值稳定性与数值色散	59
4.3.6 蛙跳（leapfrog）更新计算方法	62
4.3.7 Yee 氏网格介绍	65
4.4 本章总结	67
4.4.1 有限差分法总结	67
4.4.2 有限时域差分法总结	67
4.5 习题	68
4.6 上机实验	70
第 5 章 加权余量法与伽辽金有限元	72
5.1 目的	72
5.1.1 回顾	72
5.1.2 授课框架	72

5.2 加权余量法	73
5.2.1 介绍	73
5.2.2 弱形式	74
5.2.3 边界条件	76
5.2.4 基函数	78
5.2.5 系数矩阵的计算：以元为中心的视角	80
5.2.6 系数矩阵的组装方法	82
5.2.7 计算边界条件	84
5.2.8 一维有限元编程步骤	85
5.2.9 计算示例	86
5.2.10 上机练习	89
5.3 二维有限元	91
5.3.1 弱形式与边界条件	91
5.3.2 基函数与有限元方程	93
5.3.3 系数矩阵的组装	94
5.3.4 系数矩阵的解析计算方法	96
5.3.5 系数矩阵的数值计算方法（选讲）	98
5.3.6 连接矩阵	98
5.3.7 实施步骤	98
5.3.8 计算示例	99
5.3.9 三维有限元方法（选讲）	102
5.4 上机练习	103
5.5 网格剖分（选讲）	103
5.5.1 函数输入	103
5.5.2 函数输出	104
5.5.3 实例讲解	105
第 6 章 边界元法	113
6.1 目的	113
6.1.1 简介	113
6.2 方法	114
6.2.1 静电场的积分形式和基本解	115

6.2.2 基本解与 Green 函数	116
6.2.3 BEM 的通用形式：直接计算边界积分方程	118
6.2.4 实施方法：有限元与加权余量法	118
6.2.5 实施步骤	121
6.3 计算示例	122
6.3.1 解析积分方法	123
6.3.2 2D 边界元法的通用计算程序	124
6.3.3 BEM 数值方法的奇异性	127
6.3.4 BEM 自适应边界元划分	128
6.3.5 数值积分方法	129
6.4 边界元背后的数学知识	130
6.4.1 微分算子示例	131
6.4.2 加权余量法与格林公式	132
6.4.3 边界积分方程	132
6.4.4 有限元与系数矩阵	134
6.4.5 系数矩阵元素的确定	136
6.5 总结	139
6.6 习题	140
第 7 章 电磁场逆问题	141
7.1 数学模型	141
7.1.1 逆问题实例及数学模型	141
7.1.2 电磁场数值计算中的逆问题	141
7.1.3 电阻抗断层成像简述	141
7.2 优化算法	141
7.2.1 引子	142
7.2.2 Newton (牛顿) 算法	145
7.2.3 正则化方法	152
7.2.4 Quasi–Newton (拟牛顿) 算法	156
7.2.5 实施方法及实例讲解	167
7.2.6 课后习题	171

7.3	Jacobian 矩阵	172
7.3.1	储备知识：范数与求导	172
7.3.2	Gauss–Newton（高斯牛顿）算法	175
7.3.3	有限元逆问题中的 Jacobian 矩阵	179
7.3.4	计算示例	188
7.3.5	课后习题	190
7.4	电阻抗断层成像	191
7.4.1	引子	191
7.4.2	计算流程	193
7.4.3	正问题	195
7.4.4	逆问题	201
7.4.5	成像算法	202
7.4.6	实际应用	205
7.4.7	上机指南	205
第 8 章	随机类全局优化算法	206
8.1	目的	206
8.2	引言	206
8.3	简介	209
8.4	起源, Metropolis Et. Al., 1953	210
8.4.1	蒙特卡洛方法	210
8.4.2	马尔可夫链	213
8.4.3	统计力学和玻尔兹曼（Bolzmann）分布	215
8.4.4	Metropolis 算法	217
8.4.5	Metropolis 算法与依辛（Ising）模型	220
8.4.6	依辛模型仿真	222
8.4.7	荣誉归属	223
8.4.8	插曲	223
8.5	模拟退火与组合优化: KIRKPATRICK ET AL., 1983.	225
8.5.1	电路设计与自旋玻璃	226
8.5.2	Kirkpatrick 等人之后	228
8.5.3	模拟退火算法的一个应用	229

8.6 尾声	229
8.7 结论	230
第 9 章 上机指南	232
9.1 目的	232

课程设计

给学生灌输信念，排除学习的心理障碍

理论 40 学时，实验 20 学时。先修：高等数学，线性代数。以学员为主体，以问题为中心，教员为主导。讨论，举例式教学。推进研究型教学。基本理论 20%，数值计算安排 47%，实验上机安排 33%。对正问题的有限元方法，逆问题的牛顿迭代法重点展开。能讲出来龙去脉，适当参考阅读相关材料，多做练习。教师要学会介绍这个领域中的一些新进展。

参考教材 [1–7]。仿照参考书籍 [1, 8] 的教学方法，

以例子教学开始引入，然后回归到倪老的经典著作 [2]，讲解一些深层次的、数学层面的东西，供学有余力的学生钻研。在授课过程中，讲解实例程序、可视化，然后再辅助对知识点的理解。在数值方法中，突出求解代数方程而不是微分或积分方程。

教学心得

1. 注意 PPT 中视频的节奏，不要太松散
2. 板书要字写好、安排好空间，最后留下来的应该都是精华
3. PPT 颜色的字体需要加深，字号要用大
4. 关注学生的反应、亲和力，课后再跟学生打成一片
5. 有自信，切记避免一个语句后的碎语言
6. 鼓励学生提问，而不是单纯的设置问题让学生发问
7. 教案签字需要一周以前准备完毕
8. 通过调动学生的积极性课程节奏可以方便的控制

专用名词

node (nodal) 节点

element 元; 有限元

gradient 梯度; gradient conforming 服从梯度分布的

cavity 空腔

extrapolation 外推法

第1章 电磁场特性及数学模型

刘本源 *bbyliu@fmmu.edu.cn*

学时 上机

4 0

1.1 目的

电磁场数值分析的任务和内容，电磁场的基本规律及其变形，定解条件等。

重点：Maxwell 方程组及其变形，电磁参数

难点：定解条件

1.2 内容预览

借鉴 CEM 教材 [1, 8] 的第一章引言。介绍电磁场起始、演变、现状。介绍 CEM 的最新应用，简要介绍 CEM 的主要工具及演示。再介绍一些最新的前沿内容，如 Science 上的无线信号 wireless 可视化、隐身斗篷、EIT、雷达等。

预先讲解了一部分矢量分析、矩阵运算的相关内容。随后以 PBL 的形式讲解了 CEM 中重要的数学概念：位函数。给出了 Maxwell 方程组，讲解了物理意义和实例。针对场矢量的微分形式，重点讲了 curl–curl 公式。并且介绍了 Maxwell 方程组的微分、积分形式，以及各个公式与数值计算方法之间的联系。帮助同学们建立一种全局观念，**每一个理论都不是死记硬背的，都与后续的数值方法紧密相连。**

1.3 讲义内容

1.3.1 矢量分析的简短教程

1. 梯度算子

梯度算子为 ∇f （注意 ∇ 和标量场 f 之间没有 \cdot ，其含义是不一样的）。标

量场 f 的梯度 [6, 7] 定义为

$$\nabla f = \frac{\partial f}{\partial x} \mathbf{n}_x + \frac{\partial f}{\partial y} \mathbf{n}_y + \frac{\partial f}{\partial z} \mathbf{n}_z \quad (1.1)$$

对于标量场 f , 其在 (x, y, z) 处只有函数值的大小, 我们可以利用等值面 (等值线) 来直观展示标量场和梯度 (例如地图上的等高线)。标量场 f 的梯度 ∇f 是一个矢量场, 在空间某一点上既有大小又有方向。梯度的方向垂直于等值面, 即为等值面的法向方向, 因而标量场的梯度指向为点 (x, y, z) 处变化率最大的方向。这一思路常用于优化问题的求解中。

2. 散度算子

矢量场 \mathbf{A} 既有大小又有方向

$$\mathbf{A} = \mathbf{A}(x, y, z) = A_x \mathbf{n}_x + A_y \mathbf{n}_y + A_z \mathbf{n}_z \quad (1.2)$$

矢量场 \mathbf{A} 的散度定义为单位时间内矢量场 \mathbf{A} 流出截面 S 的外向流量除以截面所围住的体积 ΔV , 写作 $\nabla \cdot \mathbf{A}$ 。按照该定义, 其数学形式为:

$$\nabla \cdot \mathbf{A} = \lim_{\Delta V \rightarrow 0} \frac{\oint_S \mathbf{A} dS}{\Delta V} = \frac{\partial A_x}{\partial x} + \frac{\partial A_y}{\partial y} + \frac{\partial A_z}{\partial z} \quad (1.3)$$

如果矢量场的散度为正, 等价于 ΔV 内有源 (Source); 若散度为负, 则等价于 ΔV 内存在漏 (Sink)。从式(1.3)也可看出, 矢量场的散度为一个标量场。

从式(1.3)中, 利用微积分的定义, 可直观的得到高斯定理 (Gauss's Theorem)

$$\oint_S \mathbf{A} dS = \iiint_V \nabla \cdot \mathbf{A} dV \quad (1.4)$$

通过高斯定理, 可将封闭体的面积分转换为等价的体积分, 反之亦然。

3. 旋度算子

矢量场 \mathbf{A} 的环量定义为: 矢量场绕着一闭合路径 L 的线积分; 而其旋度则为 \mathbf{A} 的环量除以闭合路径所围住的面积 ΔS , 其方向指向 ΔS 的法线方向 (右手法则)。矢量场 \mathbf{A} 的旋度 $\nabla \times \mathbf{A}$ 可写为:

$$\nabla \times \mathbf{A} = \lim_{\Delta S \rightarrow 0} \frac{\oint_L \mathbf{A} dL}{\Delta S} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ A_x & A_y & A_z \end{vmatrix} \quad (1.5)$$

矢量场的旋度仍就是一个矢量场。一个最直观的例子是, 重力场 (势场) 是无旋的, 这个例子很重要, 因为重力场是个梯度场, 而 $\nabla \times (\nabla f) = \mathbf{0}$ 。

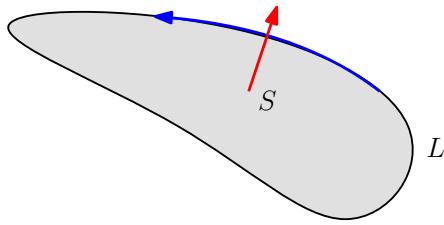


图 1.1 旋度方向的右手法则

从式(1.5)中，可直观的得到 Stokes 定理 (Stokes's Theorem)

$$\oint_L \mathbf{A} d\mathbf{L} = \iint_S \nabla \times \mathbf{A} d\mathbf{S} \quad (1.6)$$

式(1.6)的含义是：矢量场在任意闭合回路 L 上的环量等于以该环路为边界的曲面 S 上旋度的积分。

4. 二阶微商的公式

令 f 代表标量场， \mathbf{A} 为矢量场，我们可以得到以下的二阶微商公式 [6]

$$\nabla \times (\nabla f) = 0 \quad (1.7)$$

$$\nabla \cdot (\nabla \times \mathbf{A}) = 0 \quad (1.8)$$

$$\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla \cdot \nabla \mathbf{A} \quad (1.9)$$

其中 $\nabla \cdot \nabla$ 也写作 ∇^2 ，常被称作拉普拉斯算子 (Laplace Operator)。

例题 1.1： 将各个算子直接在直角坐标系下展开，试推导二阶微商公式。

式(1.7)表明，梯度场是无旋的，换言之，如果一个向量场 \mathbf{A} 的旋度为 $\nabla \cdot \mathbf{A} = \mathbf{0}$ ，那么这个向量场一定可以写成某个标量场 f 的梯度的形式。从式(1.8)可以看出，旋度场是无散的。也就是说一个旋度场的流量是守恒的，不存在源 (source) 和漏 (sink)。式(1.9)将会在后续推导 Helmholtz 波动方程，以及推导 curl–curl 公式中用到。并且更进一步的，将在时域有限差分方法中推导时域更新计算公式。

1.3.2 Maxwell 公式

在本书中我们主要考虑理想媒介，具备以下特性：线性 (linear)，各向同性 (isotropic)，无色散 (Non-dispersive)。在此类理想媒介中

$$\mathbf{B} = \mu \mathbf{H} \quad (1.10)$$

$$\mathbf{D} = \epsilon \mathbf{E} \quad (1.11)$$

$$\mathbf{J} = \sigma \mathbf{E} \quad (1.12)$$

其中 $\mu = \mu_0 \mu_r$ 是磁导率 (per-meability), $\epsilon = \epsilon_0 \epsilon_r$ 是介电常数 (per-mittivity), σ (也写作 γ) 是电导率。 $\mu_0 = 4\pi \times 10^{-1} H/m$, $\epsilon_0 \approx 8.854 \times 10^{-12} F/m$, $c = (\mu_0 \epsilon_0)^{-1/2} = 299\,792\,458 m/s$ 。 μ_r, ϵ_r 是相对磁导率和相对介电常数。在理想媒介下, 可以忽略电磁场中的磁化效应和极化效应。

各个电磁参量的单位和名称分别为:

1. \mathbf{B} (T for Tesla) 磁感应强度 (magnetic flux density)
2. \mathbf{H} (A/m) 是磁场强度 (magnetic field)
3. \mathbf{D} (C/m^2) 是电位移矢量 (electric displacement)
4. \mathbf{E} (V/m) 是电场强度 (electric field)
5. \mathbf{J} (A/m^3) 是电流密度, 而 ρ (C/m^3) 是电荷密度

Maxwell 方程组由四个公式构成, 分别是:

安培定律 (Ampère's law):

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \quad (1.13)$$

法拉第电磁感应定律 (Faraday's law):

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad (1.14)$$

泊松方程 (Poisson's equation):

$$\nabla \cdot \mathbf{D} = \rho \quad (1.15)$$

磁单极子¹ 通量条件 (the condition of solenoidal magnetic flux density):

$$\nabla \cdot \mathbf{B} = 0 \quad (1.16)$$

上面的四个公式就是 Maxwell 方程组。除此以外, 还有一个电荷守恒定律 (电荷的流动产生电流):

$$\nabla \cdot \mathbf{J} = -\frac{\partial \rho}{\partial t} \quad (1.17)$$

¹由于磁场是无散的, 人们据此提出假说: N 磁单极子和 S 磁单极子 (如果存在的话) 总是成对出现的。当前物理学中, 人们仍旧致力于寻找磁单极子——这一物理意义上存在的概念。

需要注意的是，Maxwell 公式之间并不是完全独立的。应用式(1.9)，

$$\nabla \cdot \nabla \times \mathbf{A} = 0 \quad (1.18)$$

1. 从式(1.13) Ampère's law 结合电荷守恒式(1.17)推出式(1.15) Poisson's law
2. 从式(1.14) Faraday's law 推出磁单极子磁通量公式 (1.16)

下面我们针对数值计算的不同应用，给出 Maxwell 方程组的不同形式。

1. 时谐电磁场

时谐 (time harmonic) 电磁场

$$\mathbf{E} = \mathbf{E}_r \exp^{j\omega t + j\phi_E} \quad (1.19)$$

$$\mathbf{B} = \mathbf{B}_r \exp^{j\omega t + j\phi_B} \quad (1.20)$$

然后利用

$$\frac{\partial \mathbf{A}}{\partial t} = j\omega \mathbf{A} \quad (1.21)$$

即可得到时谐条件下的 Maxwell 方程组。

2. 准静态场

在满足准静态条件时，

$$L \ll \lambda = cT \quad (1.22)$$

电磁波的时延可忽略不计。在这种情况下，某一瞬态的源，将决定某一瞬间的场的分布。准静态条件下又可分为磁准静态和电准静态两种。

$$\nabla \times \mathbf{H} = \mathbf{J} \quad (1.23)$$

$$\nabla \times \mathbf{E} = 0 \quad (1.24)$$

3. 静态场

当电量不随时间变化时，静止电荷所产生的静电场为

$$\begin{cases} \nabla \times \mathbf{E} = 0 \\ \nabla \cdot \mathbf{D} = \rho \end{cases} \quad (1.25)$$

而恒定电流产生的恒定磁场为

$$\begin{cases} \nabla \times \mathbf{H} = \mathbf{J} \\ \nabla \cdot \mathbf{B} = 0 \end{cases} \quad (1.26)$$

1.3.3 位函数

位函数，英文名为（Potential Function），或者称为势函数，是电磁场数值计算中一个重要的基础概念。结合 Maxwell 方程组和矢量场公式：

$$\nabla \times (\nabla f) = 0 \quad (1.27)$$

$$\nabla \cdot (\nabla \times \mathbf{A}) = 0 \quad (1.28)$$

$$\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla \cdot \nabla \mathbf{A} \quad (1.29)$$

能不能只用标量场的梯度来表示矢量，就像重力场用势能来表示一样。这样做的好处是标量场更容易计算。下面我们将发现，电磁场可以更准确、更方便的用位势函数来表示。

首先，引入几个定义：

1. 若矢量场在某范围内散度为 0，则称之为无散场（无源）。任何矢量场 \mathbf{A} 的旋度 $\nabla \times \mathbf{A}$ 都是无散场。
2. 若矢量场在某范围内旋度为 0，则称之为无旋场。任何标量场 φ 的梯度 $\nabla \varphi$ 都是无旋场。
3. 若矢量场在某范围内既无散也无旋，则称之为谐和场。

$$\nabla \times \mathbf{A} = 0 \rightarrow \mathbf{A} = \nabla \varphi \quad (1.30)$$

$$\nabla \cdot \mathbf{A} = 0 \quad (1.31)$$

谐和场满足 $\nabla \cdot \nabla \varphi = \nabla^2 \varphi = 0$ ，即 Laplace 方程。

4. 一般矢量场可分解为 $\mathbf{A} = \mathbf{A}_{grad} + \mathbf{A}_{curl}$ ，其中 \mathbf{A}_{grad} 为无旋场，而 \mathbf{A}_{curl} 为无散的有旋场。

例题 1.2: 首先，由于 \mathbf{B} 是无源的，即（迄今为止研究表明）不存在磁单极子。因此 \mathbf{B} 可写为矢量场的旋度的形式，

$$\mathbf{B} = \nabla \times \mathbf{A} \quad (1.32)$$

其中 \mathbf{A} 为任何一个矢量场。

例题 1.3: 同样，根据 Faraday's law 式(1.14)，电场 \mathbf{E} 可以写为，

$$\mathbf{E} = -\nabla \varphi - \frac{\partial \mathbf{A}}{\partial t} \quad (1.33)$$

根据 $\nabla \times (\nabla \varphi) = 0$, 我们可以方便的验证式(1.33)。

例题 1.4: 但是式(1.32)和(1.33)的结果是不唯一的, 如我们可以构造

$$\mathbf{A}' = \mathbf{A} + \nabla f \quad (1.34)$$

$$\varphi' = \varphi - \frac{\partial f}{\partial t} \quad (1.35)$$

针对这一问题, 我们引入洛伦兹规范对位函数进行约束

$$\nabla \cdot \mathbf{A} = -\mu\epsilon \frac{\partial \varphi}{\partial t} \quad (1.36)$$

式(1.32)和式(1.33)中的 \mathbf{A}, φ 被称作位函数。在电磁场数值求解中, 通过间接地计算 \mathbf{A}, φ , 可方便的得到 \mathbf{B} 和 \mathbf{E} 的数值解。

1.3.4 初始条件与边界条件

1. 初始条件

求解 $\mathbf{A}(\mathbf{r}, t)$ 的二阶偏微分方程, 需要给定

$$\mathbf{A}_{t=0} = g_1(\mathbf{r}) \quad (1.37)$$

$$\left. \frac{\partial \mathbf{A}}{\partial t} \right|_{t=0} = g_2(\mathbf{r}) \quad (1.38)$$

作为初始条件¹。

例题 1.5: 对于求解 $f''(t) = 6$, 通过两次积分可得, 函数 f 应满足 $f(t) = 3t^2 + \alpha t + \beta$ 。为了定解, 我们需指定 $\alpha = f'(0)$ 和 $\beta = f(0)$ 。

2. 边界条件

边界条件与坐标变量 \mathbf{r} 相联系。我们分以下两种情况。

第一类边界条件 (也称作 Dirichlet 条件):

$$\mathbf{A}|_{\mathbf{s}} = f_1(\mathbf{r}_b, t) \quad (1.39)$$

¹初始条件常跟迭代计算的数值方法相关。例如在时域有限差分方法中, 为了分别计算时域更新和空域更新, 分别需要给定 0 时刻的值和 0 时刻的一阶导

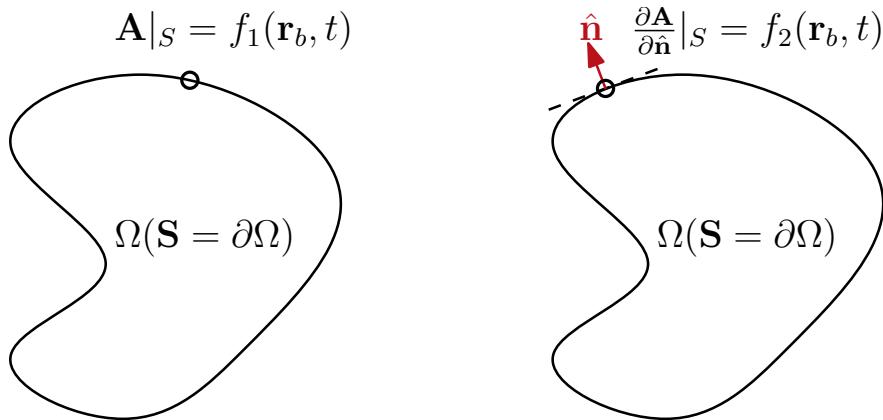


图 1.2 两种典型的边界条件。图 (a) 为第一类边界条件, 边界点上的函数值已知; 图 (b) 为第二类边界条件, 边界点上的法向导数已知。

第二类边界条件 (也称作 Neumann 条件):

$$\left. \frac{\partial \mathbf{A}}{\partial \hat{\mathbf{n}}} \right|_S = f_2(\mathbf{r}_b, t) \quad (1.40)$$

仅含初值条件的定解问题称为初值问题。没有初值条件而只有边界条件的定解问题为边值问题。由 Laplacian 方程构成的第一类、第二类边值问题, 常被称作 Dirichlet 问题和 Neumann 问题¹。

下面我们给一个例子, 利用 Maxwell 公式分析不同媒介分界面上的电磁场参数见的关系。本节内容对应于教材 1.7.3 的结论, 即正交 (**tangential**) 和切向 (**normal**) 分量所须满足的边界条件形式。

在准静态和某些静态边值问题中, 边界条件的形式通常为:

1. 给定 \mathbf{E} , \mathbf{H} 或者 \mathbf{A} 的切向分量 \mathbf{E}_t , \mathbf{H}_t , \mathbf{A}_t
2. 或者给定边界上场矢量 \mathbf{B} , \mathbf{D} 的法向分量 \mathbf{B}_n , \mathbf{D}_n

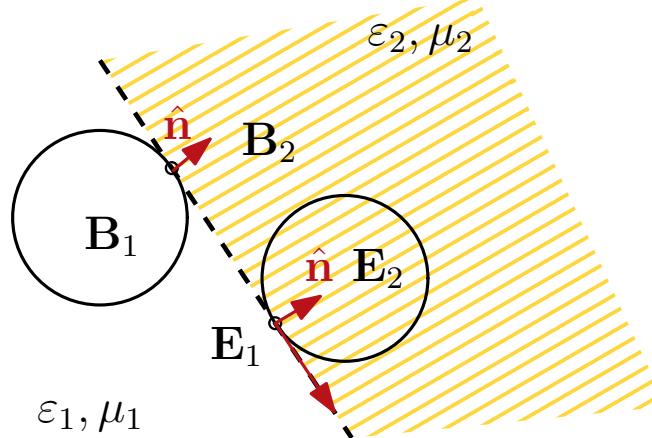
例题 1.6: 考虑静态场下, 两媒介 (ϵ_1, μ_1) 和 (ϵ_2, μ_2) , 由无限长边界分割:

首先, 根据 Maxwell 式(1.16) 和高斯定理 Gauss's Theorem

$$\nabla \cdot \mathbf{B} = 0 \quad (1.41)$$

$$\int_V \nabla \cdot \mathbf{B} dV = \oint_{\partial V} \mathbf{B} \cdot \hat{\mathbf{n}} dS \quad (1.42)$$

¹直观的, 当两类边界条件混合在一起时就被称作第三类边界条件“洛平问题”。本书中, 我们不讲解第三类——即混合了第一类和第二类边界条件的定解问题。



我们可得

$$\hat{n} \cdot (\mathbf{B}_1 - \mathbf{B}_2) = 0 \quad (1.43)$$

其中 \hat{n} 为垂直于交界面的、从媒介 1 指向媒介 2 的单位法向向量。

同理，利用 Maxwell 式(1.15)，

$$\hat{n} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = \rho_s \quad (1.44)$$

式(1.43)和式(1.44)的含义是：磁感应强度 \mathbf{B} 的法向分量是连续的，而电位移矢量 \mathbf{D} 的法向分量一般不连续，不连续值相当于界面上存在面电荷密度 ρ_s 。

随后，根据 Maxwell 式(1.14) 和斯托克斯定理 Stokes's Theorem

$$\nabla \times \mathbf{E} = 0 \quad (1.45)$$

$$\int_S (\nabla \times \mathbf{E}) dS = \oint_{\partial S} \hat{n} \times \mathbf{E} dl \quad (1.46)$$

我们可得

$$\hat{n} \times (\mathbf{E}_1 - \mathbf{E}_2) = 0 \quad (1.47)$$

以及

$$\hat{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{J}_s \quad (1.48)$$

式(1.47)和式(1.48)表明：电场强度 \mathbf{E} 的切向分量是连续的，而磁场强度 \mathbf{H} 的切向分量不连续，不连续值相当于界面上可能流过的自由电流密度 \mathbf{J}_s 。

同理，在准静态条件下，电流密度 \mathbf{J} 满足，

$$\hat{\mathbf{n}} \cdot (\mathbf{J}_1 - \mathbf{J}_2) = 0 \quad (1.49)$$

下面简单的讨论不同媒介下边界条件的特点。如果媒介 1 是良导体（PEC, Perfect electric conductor）¹，而良导体具备如下理想特性：

1. 良导体具备无穷的电导率（换言之，0 电阻率）
2. 良导体中电场强度为 $\mathbf{E} = \mathbf{0}$
3. 在有限频率下，且良导体中的磁通量为 $\mathbf{B} = \mathbf{0}$, \mathbf{B} 是不随时间变化的。作用在良导体上的任何外加电场都不会影响良导体的场结构。

根据这些 PEC 特性，从式(1.44)、式(1.47)、式(1.43) 和式(1.48)可得到

$$\begin{aligned}\hat{\mathbf{n}} \cdot \mathbf{D}_2 &= \rho_s & \hat{\mathbf{n}} \times \mathbf{E}_2 &= 0 \\ \hat{\mathbf{n}} \cdot \mathbf{B}_2 &= 0 & \hat{\mathbf{n}} \times \mathbf{H}_2 &= \mathbf{J}_s\end{aligned}$$

在电磁场数值计算中，另外一种理想假设是吸收边界条件（ABC, Absorbing boundary conditions）。ABC 中最为重要的是完美匹配层（PML, Perfectly matched layer）。在 PML 上，所有的功率输入都会被完全吸收。与之形成对比的是，在 PEC 上，所有的功率都会被反射。

1.3.5 数值计算方法

我们在本节将先前学习的知识汇总，并且给出以后各章 CEM 典型数值计算方法中需要用到的原理、基本思想和重点公式。我们力图讲解清楚，Maxwell 微分形式是怎么用的？频域时谐的形式又是怎样引入的？以及位函数、积分形式、边界条件、初始条件在数值计算中都是怎样使用的。

后续各个章节中，我们将会讲解 FDM、FDTD、FEM、MoM 等常见的电磁场数值计算方法。下面，我们将各个方法和本节已经学习的概念联系起来，讲一讲这些基本理论在数值方法中的作用。

1. 有限元方法（FEM）

¹PEC do not exist in nature, the concept is useful when electrical resistance is negligible compared to other effects. One example is electrical circuit diagrams, which carry the implicit assumption that the wires connecting the components have no resistance. In computational electromagnetics, PEC can be simulated faster given that finite conductivity is neglected.

首先，我们先讲一下 CEM 中的一个重要的电磁参数：电磁场能量。

$$-\oint_S (\mathbf{E} \times \mathbf{H}) dS = \frac{d}{dt} [(W_e + W_m) + P] \quad (1.50)$$

\mathbf{S} 为坡印廷矢量，定义为 $\mathbf{S} = \mathbf{E} \times \mathbf{H}$ 。式(1.50)的含义是：单位时间内流入闭合面 S 的能量等于电磁场能量的增加率和电磁能量的消耗率。

电场能量定义为：

$$W_e = \int_V \omega_e dV = \frac{1}{2} \int \epsilon |\mathbf{E}|^2 dV \quad (1.51)$$

其中 $\omega_e = (\mathbf{E} \cdot \mathbf{D})/2$.

磁场能量定义为：

$$W_m = \int_V \omega_m dV = \frac{1}{2} \int \frac{|\mathbf{B}|^2}{\mu} dV \quad (1.52)$$

其中 $\omega_m = (\mathbf{H} \cdot \mathbf{B})/2$.

有限元方法 (Finite Element Method, FEM)，其数学本质属于系统位能 φ 的泛函近似最小化方法，而泛函最小化本身就是变分原理的应用。针对静电场的数值计算，有限元方法中待极小化的泛函 $\mathbf{E}(\cdot)$ 即为封闭体积中的静电能：

$$\arg \min W_e = \frac{1}{2} \int_V \epsilon |\mathbf{E}|^2 dV \quad (1.53)$$

在位函数一节，我们讲解了可用位函数 φ 表示电场强度 $\mathbf{E} = -\nabla\varphi$ ，那么

$$\arg \min \frac{1}{2} \int_V \epsilon \left[\left(\frac{\partial \varphi}{\partial x} \right)^2 + \left(\frac{\partial \varphi}{\partial y} \right)^2 + \left(\frac{\partial \varphi}{\partial z} \right)^2 \right] dV \quad (1.54)$$

利用式(1.54)，可通过数值法先求 φ ，再计算 \mathbf{E} 。

2. 有限差分法 (FDM)

我们通过对 Ampère law 式(1.13) 求散度可得

$$\frac{\partial}{\partial t} \nabla \cdot \mathbf{D} + \nabla \cdot \mathbf{J} = 0 \quad (1.55)$$

结合电荷守恒定律式(1.17)

$$\frac{\partial \rho}{\partial t} + \mathbf{J} = 0 \quad (1.56)$$

我们就可以得到 Poisson's law

$$\nabla \cdot \mathbf{D} = \rho \quad (1.57)$$

有限差分方法的思想最为简洁：通过对待求区域进行网格划分，利用网格

差分近似微分，即可实现数值计算。如果：

1. 初始场满足 Poisson's law (初始条件)
2. 那么通过对 Ampère 的差分计算，结合电荷守恒定律 (差分数值计算)
3. 将会保证后续的数值计算结果满足 $\nabla \cdot \mathbf{D} = \rho$ (迭代循环)

同理，对于 Faraday's law 式(1.14)求散度，可得

$$0 = \frac{\partial \nabla \cdot \mathbf{B}}{\partial t} \quad (1.58)$$

也就是说在 Maxwell 方程组中，Poisson 公式 $\nabla \cdot \mathbf{D} = \rho_s$ 和磁单极子磁通量公式 $\nabla \cdot \mathbf{B} = 0$ 仅用来指定数值计算的初始条件，或者作为初始条件是否有效的判据；Ampère 定律和 Faraday 定律及其位函数的衍生形式是计算电磁场的核心。

3. 时域有限差分 (FDTD)

电磁场的时域变化 (time evolution) 可以仅用两个方程描述。我们把有关时域差分的部分放在等式一侧：

$$\epsilon \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} - \mathbf{J} \quad (1.59)$$

$$\mu \frac{\partial \mathbf{H}}{\partial t} = -\nabla \times \mathbf{E} \quad (1.60)$$

在时域有限差分 (Finite Difference Time Domain, FDTD) 中，我们就是用式(1.59)和式(1.60)计算下一步的 \mathbf{E} 和 \mathbf{H} (advance \mathbf{E} and \mathbf{H} in time)。而初始条件可由 $\nabla \cdot \mathbf{B} = 0$ 和 $\nabla \cdot \mathbf{D} = \rho_s$ 给出。

更进一步，上面两个式子可以合并成一个公式。通过对式 (1.59) 求 ∂t ，并将式 (1.60) 代入，可得

$$\epsilon \frac{\partial^2 \mathbf{E}}{\partial t^2} + \nabla \times \frac{1}{\mu} \nabla \times \mathbf{E} = -\frac{\partial \mathbf{J}}{\partial t} \quad (1.61)$$

上式也被称作 curl–curl 等式。需要指出的是，利用式(1.61)计算电磁场，需要给定电场 \mathbf{E} 及其时域导数 $\partial \mathbf{E} / \partial t$ 作为初始条件。

更进一步，在 FEM 中常在频域计算式(1.61)。式 (1.61) 的时谐形式为

$$-\epsilon \omega^2 \mathbf{E} + \nabla \times \frac{1}{\mu} \nabla \mathbf{E} = -j\omega \mathbf{J} \quad (1.62)$$

电磁场的数值计算多采用式 (1.59)，式 (1.60) 和式 (1.61) 的形式。为了便于理解 curl–curl 函数，下面给出一个一维的例子。

例题 1.7: 我们考虑一维 curl–curl 函数的特殊情形。对于信号 $\mathbf{E}(\mathbf{r}, t)$, 在一维简化下, 式(1.61)可写为 (用到 $c^2 = 1/(\epsilon\mu)$)

$$\frac{\partial^2 \mathbf{E}}{\partial t^2} = c^2 \frac{\partial^2 \mathbf{E}}{\partial z^2} \quad (1.63)$$

则对于信号 $\mathbf{E}(z, t) = \exp(j\omega t - jkz)$, 我们可得

$$\omega = ck \quad (1.64)$$

其中角频率 ω 为波数 (wavenumber) k 的线性函数。

相位速度 v_p (phase velocity) 定义为等相位面的速度

$$\frac{d}{dt}(\omega t - kz) = 0 \rightarrow \omega = kv_p \quad (1.65)$$

对于两个信号的合成:

$$E_A = \exp[j(\omega - \Delta\omega)t - j(k - \Delta k)z] \quad (1.66)$$

$$E_B = \exp[j(\omega + \Delta\omega)t - j(k + \Delta k)z] \quad (1.67)$$

$E_A + E_B$ 为一个载波 $\exp(j\omega t - jkz)$ 和一个包络 $2 \cos(t\Delta\omega - z\Delta k)$ 的叠加。包络的传播速度为 $\Delta\omega/\Delta k$, 当 $\Delta\omega$ 与 Δk 同时趋近于 0 时, 群速度 v_g 定义为

$$v_g = \frac{\delta\omega}{\delta k} \quad (1.68)$$

结合式(1.64), 式(1.65), 式(1.68), 我们可得:

$$v_p = v_g = c \quad (1.69)$$

理想媒介中所有波以相同的速度传播, 与波数 k 无关。这一点即为我们所考虑的理想媒介中无色散 (non-dispersive) 的特性。

但是, 在后续章节我们会讲到, 对式(1.61)的数值近似将会导致一种被称作数值色散 (numerical dispersion) 的现象。

4. 边界元方法 (BEM) 及矩量法 (MoM)

静态场中我们有

$$\nabla \times \mathbf{E} = 0 \quad (1.70)$$

将电场强度 \mathbf{E} 用位函数表示 $\mathbf{E} = -\nabla\varphi$, 代入 $\nabla \cdot \mathbf{D} = \rho$ 可得

$$\nabla \cdot (\epsilon \nabla \varphi) = -\rho \quad (1.71)$$

式(1.71)也被称作泊松方程。在三维场景中，其在自由空间下的解为

$$\varphi(\mathbf{r}) = \int \frac{\rho(\mathbf{r}') dV'}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \quad (1.72)$$

式(1.72)将在边界元方法（Boundary Element Method, BEM）中，用于构造边界元方程组。

最后需要注意的是，CEM 中的数值方法永远只是 Maxwell 方程组的近似，其具备一定（往往较大）的数值误差，其作用只是帮助我们获得电磁场分布（或其它类似的偏微分方程组场量分布）的直观理解。在实际应用中，CEM 的数值方法还存在以下几个问题：

1. 计算量大，极其耗费计算资源和存储资源
2. 实际应用中需要考虑复杂的三维结构
3. 在开区域中计算电磁场需要通过吸收边界条件加以简化

1.3.6 电磁参数

在电阻抗断层成像（Electrical Impedance Tomography, EIT）中，一个重要的电磁参数是电阻抗。复阻抗 Z 在电路原理中是一个常见的量，根据能量的定义式，我们可将复阻抗写为

$$Z = \frac{\mathbf{P}}{\mathbf{I}^2} = -\frac{1}{\mathbf{I}^2} \oint_S (\mathbf{E} \times \mathbf{H}) dS \quad (1.73)$$

而复阻抗由电阻 R 和电抗 X 构成 $Z = R + jX$ 。

1.4 习题

习题 1.1：根据 Ampère 定理式(1.13) 推导出 Poisson 公式(1.15)。

习题 1.2：推导出两媒介 $(\epsilon_1, \mu_1), (\epsilon_2, \mu_2)$ 分界面处的边界条件。

第2章 离散方程组的计算机解法

刘锐岗 *ruigang@fmmu.edu.cn*

学时 上机

4 6

2.1 目的

能够说出离散方程组的常用计算机解法的基本思想，计算过程及特点。

重点：高斯消去法

难点：广义代数特征值

第3章 数值积分法

刘本源 bbyliu@fmmu.edu.cn

“Numerical tools never give the exact answer”

学时	上机
2	0

3.1 目的

能够说出常用的数值积分求解方法和特点。

重点: 高斯求积法

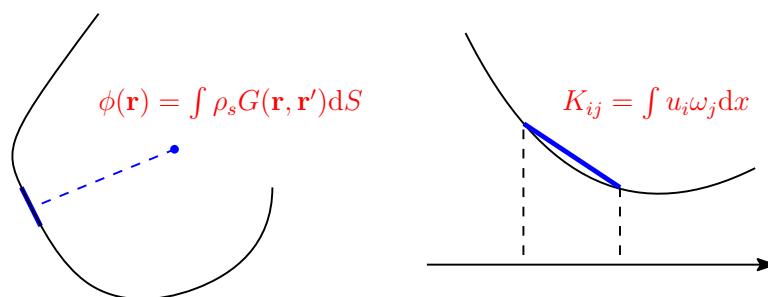
难点: 高斯求积法

3.2 讲义内容

1. 内容安排

重点讲解三种数值积分方法，分别是 (a) 中值法，(b) 辛普森积分法和 (c) 高斯积分法。课程的最后将给出一个二重积分的实例，讲解如何利用数值法计算二重积分，并分析数值积分的误差和收敛性。本节的主要内容来自教材 [2, 9]。

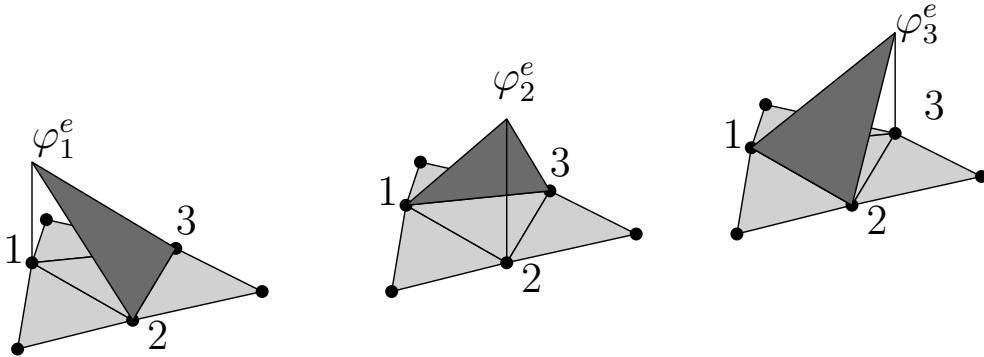
2. 课程应用



BEM : Boundary Element Method FEM : Finite Element Method

图 3.1 数值积分方法在电磁场数值计算中的应用

比如在有限元中，需要计算图中的基函数的积分值构建系数矩阵。



3. 解析积分与数值积分

在电磁场数值计算中，常需要计算给定区间的积分

$$I(f) = \int_a^b f(x)dx$$

若已知 $f(x)$ 的原函数 $F(x)$ ，则由 Newton-Leibniz 公式可得

$$I(f) = F(x)|_a^b = F(b) - F(a)$$

但是在工程应用及数值计算任务中：

1. $f(x)$ 的解析式根本不存在，只给出 $f(x)$ 在某些点的一些取值
2. $f(x)$ 的原函数 $F(x)$ 无法求出，如 $F(x)$ 不是初等函数，编程中运算复杂且计算量大
3. $f(x)$ 的表达式结构复杂，求原函数较困难

我们需要一种近似的、数值积分方法。

3.2.1 介绍

本节讲解针对 $\int_a^b f(x)dx$ 的数值积分方法。数值积分是一种近似方法，若区间 $[a, b]$ 可分为 n 个子区间，

$$a < \dots < x_{i-1} < x_i < \dots < b$$

则 $\int_a^b f(x)dx$ 可写为

$$\int_a^b f(x)dx \approx \sum_i^n \int_{x_{i-1}}^{x_i} g(x)dx \quad (3.1)$$

其中 $g(x)$ 选取为 $f(x)$ 的插值函数或者样条函数。

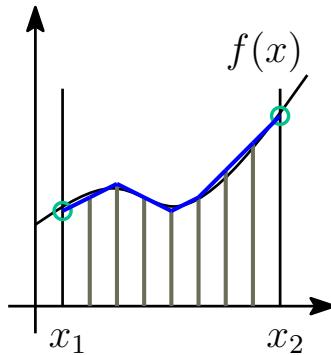


图 3.2 对积分区间进行细化，并用逼近（样条）函数去逼近原函数

3.2.2 中值法

中值法原理简单，即

$$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right) \quad (3.2)$$

其本质相当于对 $f(x)$ 函数的矩形逼近，其中逼近函数 $g(x) = f\left(\frac{a+b}{2}\right)$ 。很显然中值法的误差是比较大的。

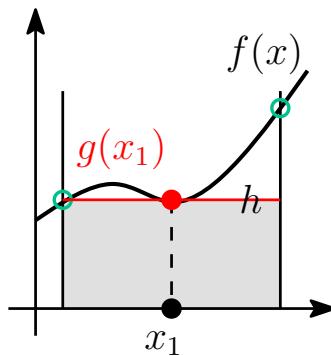


图 3.3 中值积分法示意

但是在数值计算中，可以通过将积分区间不断的细分进行逼近。实际应用中，中值法反而是一种简单、稳定、应用较为广泛的数值方法。这也跟我们在编程中所采取的原则一样¹。另外在同学的编程中，也希望能够遵循这一原则，合理的设计电磁场数值计算的子模块，时刻记着 UNIX 的编程哲学 “**提供锋利**

¹KISS, Keep it simple, stupid!

的小工具（子函数），把每一件事做好做到极致”。

3.2.3 梯形法

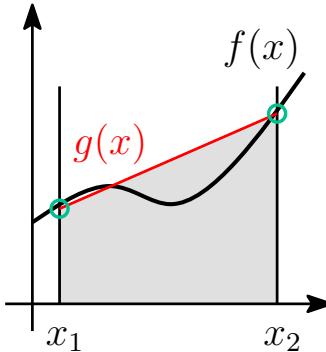


图 3.4 梯形积分方法示意

梯形法 (trapezoid) 即对函数采用梯形逼近，

$$\int_a^b f(x) dx \approx \frac{b-a}{2} [f(a) + f(b)] \quad (3.3)$$

其样条函数 $g(x)$ 为

$$g(x) = \frac{f(a) + f(b)}{2}$$

数值积分中的插值方法

这里我们简单讲解一下数值积分计算中插值的思想。插值方法是一种可以使数值积分精度逐步提升的一种递归方法。

设 h_k 为第 k 次数值积分中每个小梯形的高，令

$$S_k = \frac{h_k}{2} \sum_{i=0}^{n-1} [f(x_i) + f(x_{i+1})]$$

对梯形进行二等分， $h_{k+1} = h_k/2$ ，则

$$\begin{aligned} \int_a^b f(x) dx &\approx \frac{h_{k+1}}{2} \sum_{i=0}^{n-1} [f(x_i) + 2f(x_i + h_{k+1}) + f(x_{i+1})] \\ &\approx \frac{1}{2} S_k + h_{k+1} \sum_{i=0}^{n-1} f(x_i + h_{k+1}) \end{aligned}$$

其中 $f(x_i + h_{k+1})$ 为新中值点的函数值。通过这一计算方法，我们可以很方便的计算积分区间从 h_k 到 h_{k+1} 的数值积分值，并逐步的优化计算精度。

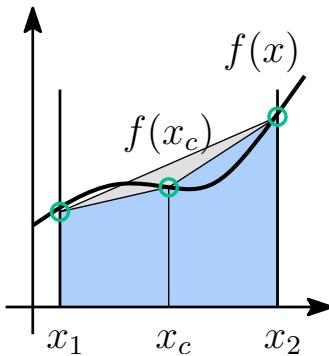


图 3.5 数值积分中的插值方法，以梯形积分方法为例

3.2.4 辛普森法

辛普森法的原理就是采用二次（样条）函数 $g(x) = ax^2 + bx + c$ 逼近 $f(x)$ ，参见图3.6。辛普森积分公式因而也被称作抛物线公式。

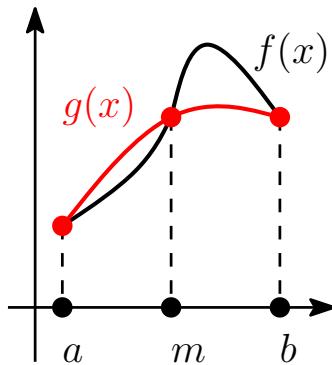


图 3.6 辛普森数值积分

我们首先给出辛普森积分的结论：将区间 $[a, b]$ 分为 $2N$ 份， $f(x)$ 在该区间上的积分可近似为

$$\int_a^b f(x)dx \approx \frac{h}{3} \sum_i^N [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] \quad (3.4)$$

文献 [9] 中，证明辛普森求积公式需要用勒让德多项式。这里，我们给出一个简单的证明方法¹。

例题 3.1：辛普森数值积分公式的推导。

¹推荐阅读 <http://www.intmath.com/integration/6-simpsons-rule.php>

首先，不失一般性，我们可在 $[-h, h]$ 上计算积分，

$$\int_{-h}^h g(x) = \frac{2}{3}ah^3 + 2ch = \frac{h}{3}[2ah^2 + 6c] \quad (3.5)$$

另外当我们用二次函数 $g(x)$ 逼近函数 $f(x)$ 时，设两个函数在 $-h, 0, h$ 三点重合，则有

$$\begin{aligned} ah^2 + bh + c &= f(h) \\ ah^2 - bh + c &= f(-h) \\ c &= f(0) \end{aligned}$$

解得 $2ah^2 = f(h) + f(-h) - 2f(0)$ 。进而可得

$$\int_{-h}^h g(x) = \frac{h}{3}[2ah^2 + 6c] = \frac{h}{3}[f(-h) + 4f(0) + f(h)] \quad (3.6)$$

3.2.5 高斯积分法

高斯积分法，又被称为高斯–勒让德 (Gauss–Legendre Integration) 积分，其原理如图3.7所示。回忆一下梯形法是采用 a 和 b 端点构成的梯形进行近似，这

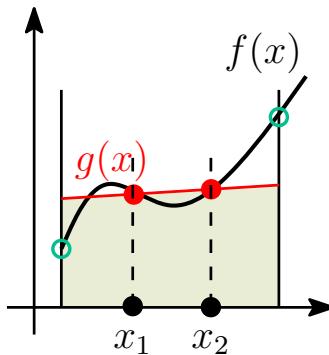


图 3.7 高斯积分法

种方法当曲线是凹函数的或者是凸函数的时，误差将会很大。

而高斯积分法的思想是一种割线 (abscissas) 法，在待积分函数 $f(x)$ 上寻找一条割线 $g(x)$ ，使得穿过割线的积分 $\int_a^b g(x)$ 逼近于原函数的积分值。割线法的关键在于寻找割线点和计算权系数。

下面以两点的高斯积分法为例讲解割点和权系数的计算方法。设待逼近函数 f 的积分区间是 $[-1, 1]$ ，我们选定任意两个点 $(x_1, f(x_1))$ 和 $(x_2, f(x_2))$ ，逼

近函数 $g(x)$ 可写为

$$g(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

高斯积分法假设割线下的面积可由下式计算：

$$\int_{-1}^1 g(x)dx = \frac{2x_2}{x_2 - x_1} f(x_1) - \frac{2x_1}{x_2 - x_1} f(x_2)$$

很明显，当 $x_1 = -1, x_2 = 1$ 时，Gauss 法就等价于我们的梯形积分法。

高斯积分法的出发点（思路）是，能否寻找一系列点 $\{x_i\}$ 和权系数 $\{\omega_i\}$ ，使得对于 $f(x)$ 的积分近似

$$\int_{-1}^1 f(x)dx \approx \sum_{i=1}^N \omega_i f(x_i) \quad (3.7)$$

具备较小的误差。

我们下面通过一个简单的例子来推导两点高斯积分法。

例题 3.2: 寻找割线点 x_1, x_2 以及权系数 ω_1, ω_2 ，使得 $N = 2$ 两点高斯积分

$$\int_{-1}^1 f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2)$$

对于任意三阶函数 $f(x) = ax^3 + bx^2 + cx + d$ 能得到精确解。确定权系数 ω_1, ω_2 和割线点 x_1, x_2 的值。

分析：由于积分是线性可加的，我们可以假设 ω_1, ω_2 和 x_1, x_2 对于每一阶都能得到精确解，即

$$\begin{aligned} f(x) = 1 : \quad & \int_{-1}^1 1dx = 2 = \omega_1 + \omega_2 \\ f(x) = x : \quad & \int_{-1}^1 xdx = 0 = \omega_1 x_1 + \omega_2 x_2 \\ f(x) = x^2 : \quad & \int_{-1}^1 x^2 dx = \frac{2}{3} = \omega_1 x_1^2 + \omega_2 x_2^2 \\ f(x) = x^3 : \quad & \int_{-1}^1 x^3 dx = 0 = \omega_1 x_1^3 + \omega_2 x_2^3 \end{aligned}$$

可以解得 $x_1^2 = x_2^2 = \frac{1}{3}$ ，不妨设 $x_1 < x_2$ ，所以 $x_1 = -\sqrt{1/3}, x_2 = \sqrt{1/3}$ 。

另一方面， $\omega_1 = \omega_2 = 1$ 。则

$$\int_{-1}^1 f(x)dx \approx f(-\sqrt{\frac{1}{3}}) + f(\sqrt{\frac{1}{3}})$$

与例题类似的方法，我们可以得到 $N = 3$ 三点的高斯积分公式

$$\int_{-1}^1 f(x)dx \approx \frac{5}{9}f(-\sqrt{\frac{3}{5}}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{\frac{3}{5}})$$

在本章的数值计算问题中，只需要掌握三点以下的高斯积分公式的推导思路即可。实际应用中对于三点以上的高斯积分，通常预先存储积分系数表。

在得到高斯积分公式后，我们关心的是这种逼近方法的误差是多少？这里，我们简单讨论一下高斯积分的误差特性。对于 $N = 2$ ，共有 4 个未知数，我们通过逼近 3 阶的 $f(x)$ 函数，来对方程组求解。两点高斯积分法的误差为 [9]

$$E_2(f) = \frac{f^{(4)}(c)}{135}$$

其中 $f^{(4)}$ 代表函数 f 的四阶导数。

对于 $N = 3$ 高斯积分，我们有 [9]

$$E_3(f) = \frac{f^{(6)}(c)}{15,750}$$

上述高斯积分误差的结论不需要同学们推导。下面给一个简单的例子。

例题 3.3：高斯积分法的数值误差分析。

证明三点高斯积分对于 $f(x) = 5x^4$ 可以得到精确解。由于 $f(x) = 5x^4$ 而 $f^{(6)}(x) = 0$ ，因而三点高斯积分的误差为 0。

另一方面，我们可通过求

$$\int_{-1}^1 f(x)dx = 2$$

积分的解析解，并对比高斯积分解，得到相同的结论。

高斯积分的系数 $\omega_{N,k}$ 和割点 $x_{N,k}$ 一般会预先存在内存中， $N = 2, 3, 4$ 的高斯系数见表 3.1，而 $N = 1$ 的高斯积分公式等价于中值法。更高阶的高斯积分系数、以及 C 和 MATLAB 的实现可参见 [10]。

需要注意的是，高斯积分需要积分限定在 $[-1, 1]$ 。对于一般的积分，积分

表 3.1 高斯积分的割点值与系数表

N	Abscissas $x_{N,k}$	Weights $\omega_{N,k}$	Truncation Error $E_N(f)$
1	0	2	midpoint method
2	$-\sqrt{\frac{1}{3}}$	1.0	$\frac{f^{(4)}(c)}{135}$
	$\sqrt{\frac{1}{3}}$	1.0	
3	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$	$\frac{f^{(6)}(c)}{15,750}$
	0	$\frac{8}{9}$	
4	± 0.8611363116	0.3478548451	$\frac{f^{(8)}(c)}{3,472,875}$
	± 0.3399810436	0.6521451549	

限是 $[a, b]$, 因而需要通过变量替换使得积分限平移到 $[-1, 1]$ 。令

$$t = \frac{a+b}{2} + \frac{b-a}{2}x, \quad dt = \frac{b-a}{2}dx \quad (3.8)$$

则

$$\int_a^b f(t)dt = \int_{-1}^1 f\left(\frac{a+b}{2} + \frac{b-a}{2}x\right) \frac{b-a}{2}dx \quad (3.9)$$

进而得到

$$\int_a^b f(t)dt = \frac{b-a}{2} \sum_{i=1}^N \omega_{N,k} f\left(\frac{a+b}{2} + \frac{b-a}{2}x_{N,k}\right) \quad (3.10)$$

或者, 换一种形式, 我们可写为如下形式方便记忆

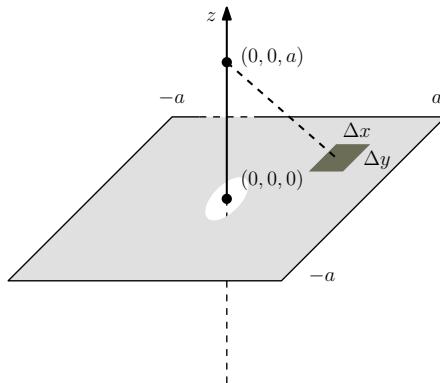
$$\int_{x-h/2}^{x+h/2} f(t)dt = \frac{h}{2} \sum_{i=1}^N \omega_{N,k} f\left(x + \frac{h}{2}x_{N,k}\right) \quad (3.11)$$

其中 x 是积分区间的中点, $x_{N,k}$ 是 N 点高斯积分的系数。

3.2.6 计算示例

我们通过一个简单的例子, 来直观的分析数值计算的误差。

例题 3.4: 计算均匀带电方板的对称轴上的静电势。方板的带电区域由 $-a < x < a$, $-a < y < a$, $z = 0$ 给出, 表面电荷密度 $\rho_s(x, y) = \rho_{s0}$ 为常量, 求对称轴上两点: $(0, 0, z)$ 和 $(0, 0, 0)$ 处的电势 φ 。



根据 Poisson 定理

$$\epsilon_0 \nabla \cdot \mathbf{E} = \rho_s$$

在静态场中有 $\nabla \times \mathbf{E} = 0$, 利用位函数表示可写为

$$\mathbf{E} = -\nabla \varphi$$

综合可得

$$\epsilon_0 \nabla^2 \varphi = -\rho_s$$

解上述泊松方程, 可得

$$\begin{aligned}\varphi(\mathbf{r}) &= \frac{\rho_{s0}}{4\pi\epsilon_0} \int_V \frac{1}{|\mathbf{r} - \mathbf{r}'|} dV \\ &= \frac{\rho_{s0}}{4\pi\epsilon_0} \iiint \frac{1}{\sqrt{(x-x')^2 + (y-y')^2 + (z-z')^2}} dx' dy' dz'\end{aligned}$$

在本例题中, 我们要求对称轴 z 上的电势

$$\varphi(0,0,z) = \frac{\rho_{s0}}{4\pi\epsilon_0} \int_{-a}^a dx' \int_{-a}^a \frac{1}{\sqrt{x'^2 + y'^2 + z^2}} dy' \quad (3.12)$$

利用积分的对称性可得

$$\varphi(0,0,z) = \frac{\rho_{s0}}{\pi\epsilon_0} \int_0^a dx' \int_0^a \frac{1}{\sqrt{x'^2 + y'^2 + z^2}} dy' \quad (3.13)$$

我们将利用中值法和辛普森法对式(3.13)求数值积分。

1. 数值积分法的网格划分与重积分的计算

我们将积分区域分成 n^2 份, 每份的大小为 $h = a/n$ 。数值积分法可以很方便地实现。

便的计算二重积分

$$I(f) = \iint_{\Omega} f(x, y) dx dy$$

只需先对 y 轴计算数值积分，再计算 x 轴积分即可。在编程实现中，此类重积分计算方法也被称为**乘积型求积公式**。

接下来以二重积分为例，我们分别考虑梯形法、辛普森法、高斯积分法的重积分计算方法。

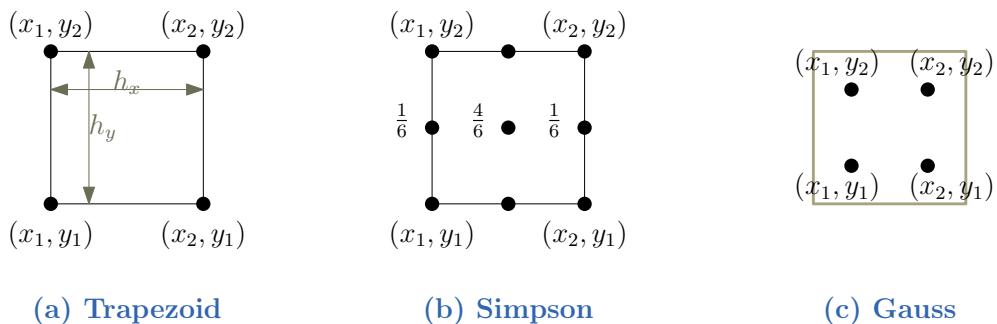
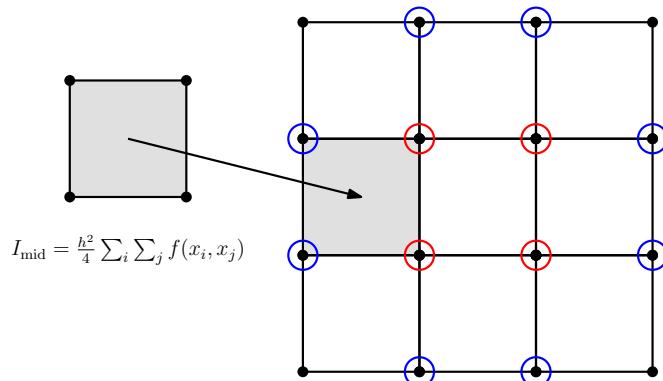


图 3.8 重积分的数值积分方法

对于梯形法，可首先求得 x 轴上的积分，随后计算 y 的数值积分即可。我们也可以写成

$$I = \frac{h_x h_y}{4} \sum_i \sum_j f(x_i, y_j)$$

其中 h_x 和 h_y 分别为梯形法在 x 轴和 y 轴的间距。其中位于角落的节点，每个函数值计算了 1 次；边上的节点计算了 2 次，而中心的节点则计算了 4 次。



而对于辛普森法，需要先固定 x 轴的三个点 $x_i, x_i + h/2, x_{i+1}$ ，分别对这

三个点上 y 轴的数据进行辛普森积分，然后再计算

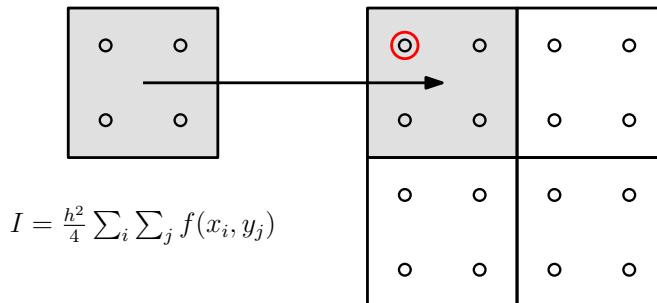
$$S_i = \frac{h}{6} [f(x_i, y, z) + 4f(x_i + h/2, y, z) + f(x_{i+1}, y, z)]$$

最后将 x 轴求和可得。具体实现可以参考 MATLAB 或 Python 例程。

最后对于高斯积分，则需要分别选择四个割点 (x_1, y_1) , (x_1, y_2) , (x_2, y_1) 和 (x_2, y_2) ，随后分别计算

$$I = \frac{h^2}{4} \sum_i \sum_j f(x_i, y_j)$$

高斯法的优势在于，其特点与中值积分方法类似，对于计算区域中的点只需要计算一次函数值即可。但是切记，高斯积分的积分限是 $[-1, 1]$ 。



下面，我们给出一个简单的例子。

例题 3.5: 用不同的数值积分方法计算

$$I(x, y) = \int_{-1}^1 \int_{-1}^1 x^2 y^2 dx dy$$

其中解析解为 $I = 4/9 = 0.4444$

1. 数值法的收敛性分析

再回到本例题。在误差分析部分，我们令 $a = 1$ ，并且绘制中值积分法和辛普森积分法对于不同 n 的积分结果。在此基础上，将结果外推到无穷小格点的情形，来得到尽可能精确的数值计算结果。

首先绘制中值法积分 I_{midp} ，辛普森数值积分 I_{simp} 与不同阶 h 的拟合图，见图3.9。从图(3.9)中可以发现中值法 I_{midp} 与 h^2 能拟合成一条直线，据此，我们推断中值积分法是二阶收敛的，即 $I_{midp}(h) = I_0 + I_2 h^2 + \dots$ 。也就是说

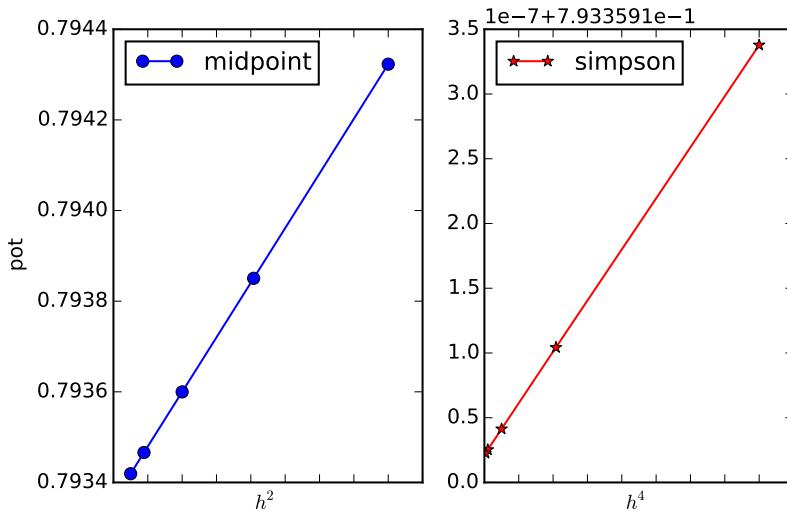


图 3.9 数值积分结果与不同阶 h 的关系。(left) 中值积分法, (right) 辛普森积分。

$I_2 h^2$ 是中值法积分泰勒展开式中的主要误差项, 对于较小的 h , 2 阶以上的高阶误差都可以忽略。从图 3.9 也可以看出, 辛普森数值积分是四阶 h^4 收敛的。

除了以上的两种分析方法外, 我们也可以直接计算数值算法的收敛率。假定数值积分结果是收敛的, 且满足

$$I(h) = I_0 + h^p$$

则由下式可估算收敛阶数 p

$$p = \ln \left[\frac{I(h_i) - I(h_{i+1})}{I(h_{i+1}) - I(h_{i+2})} \right] / \ln \left[\frac{h_i}{h_{i+1}} \right] \quad (3.14)$$

我们分别利用 $n = 5, 10, 20$ 这三点数据计算收敛率, 可得对于中值法 $p = 2.002$, 对于辛普森法 $p = 3.985$ 。这一结果表明在 $(0, 0, a)$ 处, 中值法和辛普森法分别是 2 阶和 4 阶收敛的。

另外, 我们可以通过数据拟合得到无穷小格点 h_0 的积分值。利用 polyfit 函数进行 h^2 和 I_{midp} 的二阶拟合, 可以得到 h_0 下的积分值为 0.79335912。同样, 对 h^4 和 I_{simp} 进行四阶拟合, 我们可得辛普森积分法 h_0 的外推值为 0.79335912。对称轴 $(0, 0, a)$ 处电势的 8 位精度解 0.79335912。利用数据外推(拟合), 我们可以仅用比较初级的数值方法就能得到较高的计算精度。例如, 采用最简单的中值法, 我们只用计算到 $n = 20$ 的网格格点数据, 就可通过对积分结果和 h^2 进行二阶拟合, 得到 8 位精度的数值结果。而实际每一个网格下的数

值计算精度只有 3 到 4 位。因而利用数据外推，可以极大的减少数值计算的网格数目和计算时间。同时，数据外推可以帮助我们评估数值积分算法的精度如何。例如采用辛普森数值积分法，我们无需数据外推，在较稀疏的格点上就能得到较为精确的结果。

3. 数值积分的奇异值

这里，我们要提醒大家高阶数值积分（相比较而言，最简单的中值积分方法却可方便的绕过这个问题，具备一定的稳定性）的方法可能存在奇异点，会导致积分结果发散。

下面我们来计算 $(0, 0, 0)$ 处的电势。在 $z = 0$ 下，积分元是存在奇异点的，而积分值却是收敛的。在这个问题上，辛普森数值积分需要计算 $(0, 0, 0)$ 处的值，因而结果是发散的。而中值法避开了奇异点，结果是收敛的，见图 3.10。可

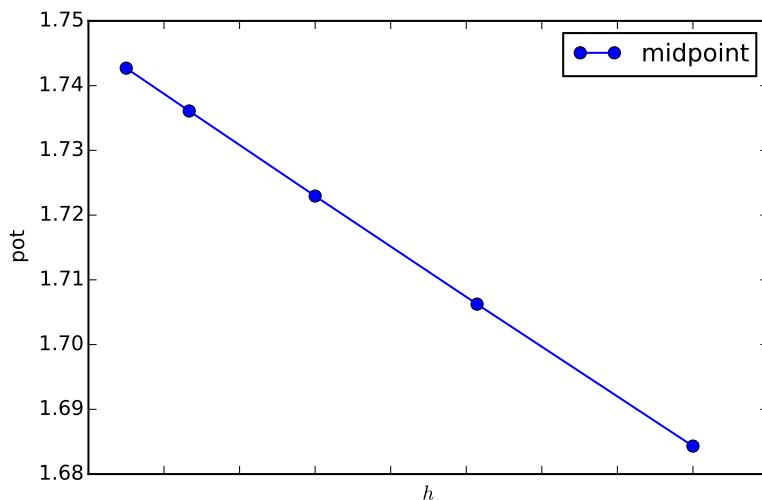


图 3.10 中值法计算 $(0, 0, 0)$ 处的电势

见在 $(0, 0, 0)$ 点处，中值法是一阶收敛的。我们同样可以通过数据拟合来计算无穷小格点 h_0 处的电势，一阶拟合结果是 1.762015，二阶是 1.762745，三阶是

1.762748。而解析解是¹

$$\int_0^a dx' \int_0^a \frac{1}{\sqrt{x'^2 + y'^2}} dy' = 2 \ln(1 + \sqrt{2}) \approx 1.762747$$

即便存在奇异点的情况下，我们仍旧可利用中值法计算²，并通过二阶或者三阶外推，得到6位数的计算精度。

4. 实际应用中的注意事项

在实际应用中，我们很少用到、或者很难达到6位计算精度。而利用数值外推的方法，仅能适用于均匀网格的不断细分。当这一条件不满足时，数值积分的收敛性可能会出现振荡。

一个比较稳健的方法是通过不断细分网格的积分结果估计收敛阶数 h^p 。由于进行收敛阶数的估计用到3个积分结果，因此我们需要至少四个积分结果来验证收敛的阶数。一旦我们得到收敛阶数的估计 p ，我们就可以通过

$$I(h) = I_0 + I_p h^p \quad (3.15)$$

利用数据外推得到 h_0 的积分估计值。

另外针对具体情况可采用特殊的积分方法。例如，

例题 3.6: 计算下式积分

$$\int_{0.01}^1 \ln x dx$$

其中 $\int \ln x dx = x \ln x - x$ 。

3.2.7 结论

本次课程第一课时的讲授内容为：

¹将 y 写为 $y = x \tan \theta$ ，然后计算

$$\int_0^1 dx \int_0^{\frac{\pi}{4}} \frac{1}{\cos \theta} d\theta = 2 \ln \left[\frac{1}{\cos \theta} + \tan \theta \right] \Big|_0^{\pi/4}$$

²在后面我们会学习到 $\int \frac{1}{r}$ 是强奇异项，而 $\int \ln r$ 是弱奇异项这一概念。

1. 中值法:

$$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right)$$

2. 梯形法:

$$\int_a^b f(x)dx \approx \frac{b-a}{2} [f(a) + f(b)]$$

3. 辛普森法:

$$\int_a^b f(x)dx \approx \frac{b-a}{6} \sum_i^N [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})]$$

4. 高斯积分法:

$$\int_{-1}^1 f(x)dx \approx \sum_{i=1}^N \omega_i f(x_i)$$

5. 数值积分的插值方法 (以梯形法为例)

$$\int_a^b f(x)dx \approx \frac{1}{2}S_k + h_{k+1} \sum_{i=0}^{n-1} f(x_i + h_{k+1})$$

第二课时的讲授内容为:

1. 数值方法的计算结果取决于分辨率。例如，将积分区域划分为 h 大小的网格，则积分结果可以表达成 $I(h) = I_0 + I_p h^p$ 的形式，其中 I_0 是精确解， $I_p h^p$ 是主要误差项，而 p 是数值方法的收敛阶数。

2. 可通过

$$p = \ln \left[\frac{I(h_i) - I(h_{i+1})}{I(h_{i+1}) - I(h_{i+2})} \right] / \ln \left[\frac{h_i}{h_{i+1}} \right]$$

估计收敛阶数 p 。收敛阶数 p 的估算需用三个网格 h_i, h_{i+1}, h_{i+2} 下的数值积分值。在得到 p 后，需要更多的数值积分结果 h_k 来验证收敛阶数。

3. 数值积分误差分析的步骤是，(1) 计算不断细分网格的数值积分，(2) 计算收敛阶数 p ，(3) 利用数据外推得到无穷小网格 h_0 下的积分估计值。
4. 收敛的阶数不但取决于所采用的数值积分算法，同时也跟所解决问题的规则性有关。实际应用中，数值解的奇异性会降低收敛阶数 p 。

3.3 习题

习题 3.1：分别通过解析求积、 $N = 2$ 高斯积分、 $N = 3$ 高斯积分计算

$$\int_{-1}^1 \frac{1}{x+2} dx$$

3.4 上机实验

上机 3.1：(10 分) 编写 1 到 3 阶的高斯积分程序。并用该程序计算

$$\int_1^2 \frac{1}{x} dx$$

提示：手工计算精确解，并注意高斯数值积分的积分限。

上机 3.2：(10 分) 编写二维 $N = 2$ 的高斯积分程序，计算方形带点平板 $-1 < x < 1, -1 < y < 1, z = 0$ 对称轴 $z = 0$ 和 $z = 1$ 处的电势。绘图并分析收敛特性，解释实验结果。

上机 3.3：(10 分) 利用中值法计算积分

$$\int_0^1 x^{-\alpha} dx$$

计算 $\alpha = 0.5, \alpha = 0.8$ 下的收敛阶数 p ，并外推到 h_0 下的积分值。该积分的精确解是 $1/(1-\alpha)$ 。

第4章 有限差分法

刘本源 byl.liu@fmmu.edu.cn

学时 上机

2 0

4.1 目的

能够解释差分的定义，能够解释偏微分方程差分格式的构造方法和定解条件的离散化方法。

重点：差分格式的构造

难点：时域有限差分法

4.2 有限差分法

4.2.1 从微分到差分

电磁场理论中的 Maxwell 公式常以微分的形式给出，而一种最原始、最自然的方法就是对求解空间进行网格划分，采用有限差分（Finite Difference）近似微分实现数值计算。

对于一维函数 $f(x)$ ，可在 x 轴上选定 N 个格点（grids） x_1, x_2, \dots, x_N ，在格点上，待求解的函数 $f(x)$ 分别取值为 $f(x_1), f(x_2), \dots, f(x_N)$ 。当选用均匀格点时，有 $x_{n+i} = x_n + ih$ ，格点间距 $\Delta x = h$ ， i 通常为整数， h 也被称作格点单元大小。

将 f 泰勒展开可得

$$f(x + \delta) \approx f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{6} f'''(x) + \dots \quad (4.1)$$

令 $\delta = h$ 即可得到微分的差分构造

$$f'(x) + O(h) = \frac{f(x + h) - f(x)}{h}$$

根据式(4.1)得到的差分称作：两格点（two-cell）差分格式。两格点差分假设高

阶导 f'' 误差可忽略，因而其对于真实微分的近似误差阶数为 $O(h)$ 。

为了提高近似的精度，可通过对两个单元上的数据求差分，

$$f'(x) + O(h^2) = \frac{f(x+h) - f(x-h)}{2h} \quad (4.2)$$

但是这种方法不适用于波长较短的情形，尤其是 $\lambda < 4h$ （在后面的数值误差分析中将进行详细的讨论）。

4.2.2 交错格点

在有限差分方法中，并不使用两格点法，而多采用一种被称作交错格点（staggered grids）的构造方法，计算半格点（half grid）上的一阶差分

$$\begin{aligned} f(x+h) &\approx f\left(x+\frac{h}{2}\right) + \frac{h}{2}f'\left(x+\frac{h}{2}\right) + \frac{(h/2)^2}{2}f''\left(x+\frac{h}{2}\right) + O(h^2) \\ f(x) &\approx f\left(x-\frac{h}{2}\right) - \frac{h}{2}f'\left(x-\frac{h}{2}\right) + \frac{(h/2)^2}{2}f''\left(x-\frac{h}{2}\right) - O(h^2) \end{aligned}$$

两式相减可得

$$f'\left(x+\frac{h}{2}\right) + O(h^2) = \frac{f(x+h) - f(x)}{h} \quad (4.3)$$

重复调用式(4.3)，即可得交错格点上的二阶导数

$$f''(x) + O(h^2) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (4.4)$$

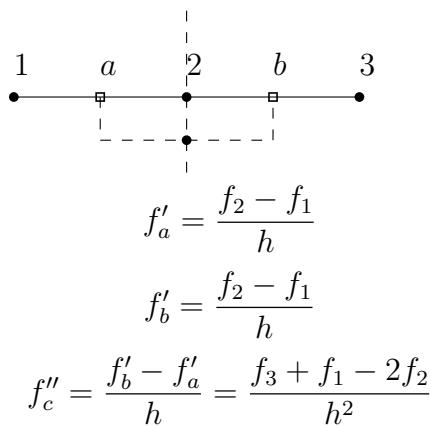


图 4.1 差分近似微分示意图。授课时，学生不能直观的理解半格点以及半格点近似二阶微分的思想，特绘图示之。

采用交错网格，式(4.3)–(4.4)的近似误差阶数为 $O(h^2)$ 。但该近似方法仅在数值解 f 是 sufficient regular（即 $f''(x)$, $f'''(x)$ 等高阶导数有界）时才有意

义。另外，式(4.4)具备局部化（local）、中心化（centered）的特性，每一个格点上的计算量仅仅与其邻近格点取值相关，因而适宜于大规模、规则区域的电磁场数值计算。

4.2.3 二维 laplace 方程的差分形式

对于 laplace 方程

$$\nabla^2 u = 0$$

在直角坐标系下，其二维微分为

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

我们将求解空间进行网格划分（五点差分格式），见图4.2。

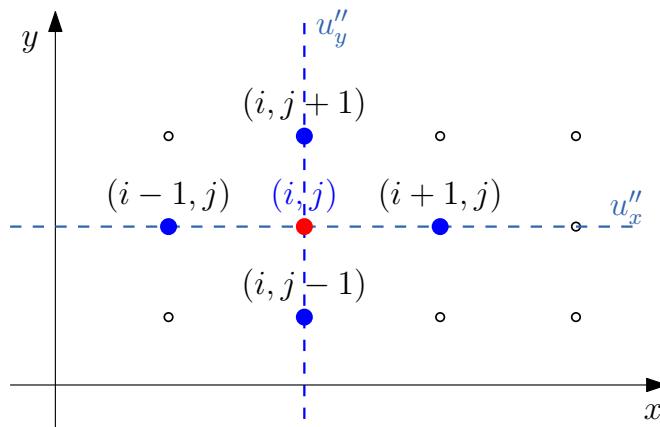


图 4.2 五点差分格式 (fix-point stencil)

利用交错格点构造方法，可得

$$u''_x = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

进而

$$\nabla^2 u(i, j) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} \quad (4.5)$$

求解 laplace 方程 $\nabla^2 u = 0$ ，从第 (n) 步到第 $(n+1)$ 步可迭代计算

$$u_{i,j}^{(n+1)} = \frac{1}{4}(u_{i-1,j}^{(n)} + u_{i+1,j}^{(n)} + u_{i,j+1}^{(n)} + u_{i,j-1}^{(n)}) \quad (4.6)$$

4.2.4 计算示例

例题 4.1：计算图 4.3 中位势 u 的分布。

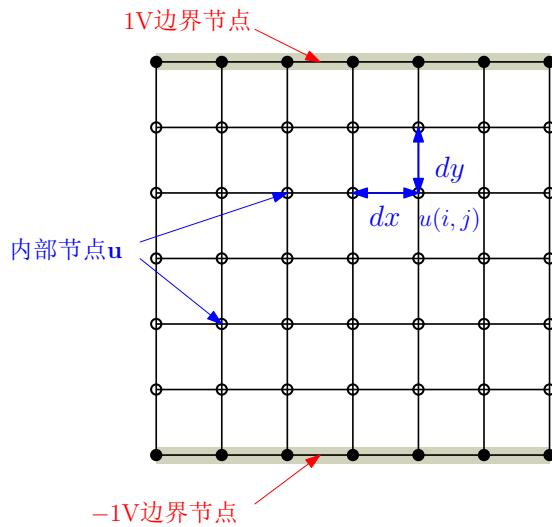


图 4.3 FDM 例题一

一个典型的 FDM 计算程序包括初始条件、边界条件、收敛条件三个部分。而算法核心则为 $\nabla^2 u = 0$ 的迭代更新计算。需要注意的是，对于线性方程组的求解，既可用矩阵的方法直接求解，也可用迭代求解的方法进行计算。

```
def py_update(u, dx2, dy2):
    """ Take a time step using straight Python loops
    and update 'u' in place. """
    nx, ny = u.shape
    dnr_inv = 0.5 / (dx2 + dy2)

    for i in range(1, nx - 1):
        for j in range(1, ny - 1):
            u[i, j] = ((u[i - 1, j] + u[i + 1, j]) * dy2 +
                        (u[i, j - 1] + u[i, j + 1]) * dx2) *
                        dnr_inv
```

主函数包括两部分：初始化网格上的位函数 u 以及设定边界条件：

```
def calc(N=200, Niter=50, func=py_update, dx=0.1, dy=0.1):
    dx2 = dx * dx
```

```

dy2 = dy * dy
u = np.zeros([N, N])

u[0] = 1           # boundary( 1V)
u[N - 1] = -1    # boundary(-1V)

for i in range(Niter):
    func(u, dx2, dy2)
return u

```

数值计算结果见图4.4。

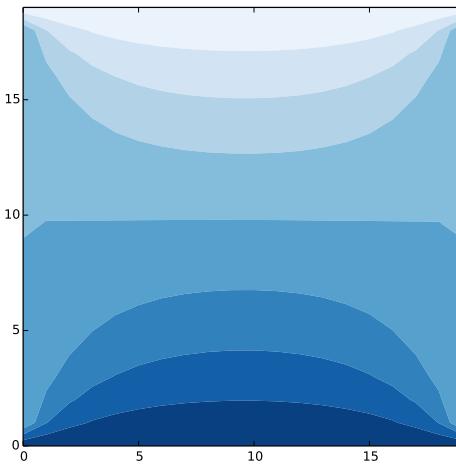


图 4.4 FDM 例题一的数值计算结果

例题 4.2: 计算同轴传输线的电容。同轴矩形传输线的几何形状如图4.5所示，包含一个截面为 $a \times b$ 的矩形内导，以及一个与它同轴放置的外波导，外波导的截面为 $c \times d$ 。

在内导和外导之间的真空区域，静电势 φ 满足二维的拉普拉斯方程

$$\nabla^2 \varphi = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0 \quad (4.7)$$

而在导体上，位势 φ 为常量。定义内导的位势为 φ_1 ，外导的位势为 φ_2 。

更进一步，在有限差分方法中，常假设 FDM 方法的计算区域刚好位于正

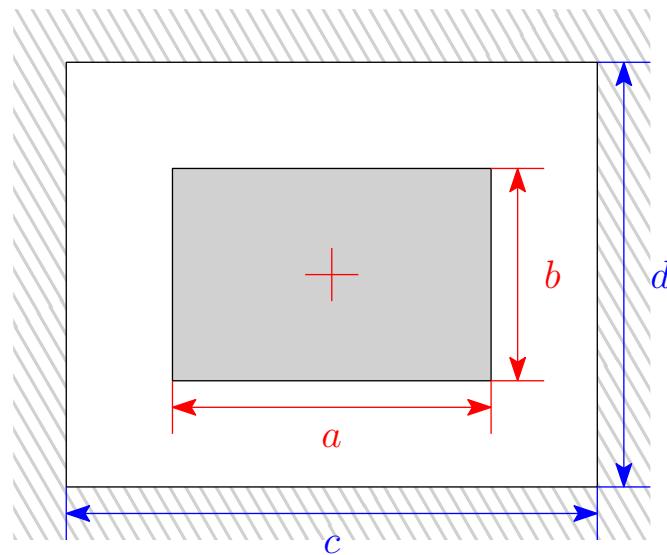


图 4.5 同轴传输线的几何形状

方形的网格格点上。(理论上可使用任何矩形格点、或者非均匀的格点。但是在有限差分法中，常使用均匀的、正方形的格点。实际应用中，多采用有限元法 FEM 解决非均匀、不规则边界的电磁场数值计算问题。)

在均匀方形格点上，我们有

$$x_i = i h, \quad i = \dots, -1, 0, 1, 2, \dots,$$

$$y_j = j h, \quad j = \dots, -1, 0, 1, 2, \dots,$$

定义格点 (i, j) 上的电势 f 为待求解变量

$$f_{i,j} = \varphi(x_i, y_j)$$

离散化后的拉普拉斯 (laplace) 方程为

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = \frac{f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2} = 0 \quad (4.8)$$

对于边界条件，令外导的电势 $\varphi_2 = 0V$ (对所有位于边界上的格点 (x_i, y_j) 都有 $f_{i,j} = 0$)，令内导的电势 $\varphi_1 = 1V$ 。我们将通过这一例题，讲解如何利用数值方法计算电磁参量 C 。由于作用在电容上的电压 $V = 1V$ ，因而单位长度上的电容 C 就为 $C = Q/V = Q$ ，其中 Q 为单位距离上的电荷量。

4.2.5 迭代法

为了求解离散 laplace 等式(4.8)，我们采用 Jacobi 迭代法求解线性方程组。这种方法简单、高效、易于实现，不需要存储线性方程组，只需存储数值解，因而可以解决大规模的数值计算问题。当然，同学们也可以用矩阵求解的方法进行计算。

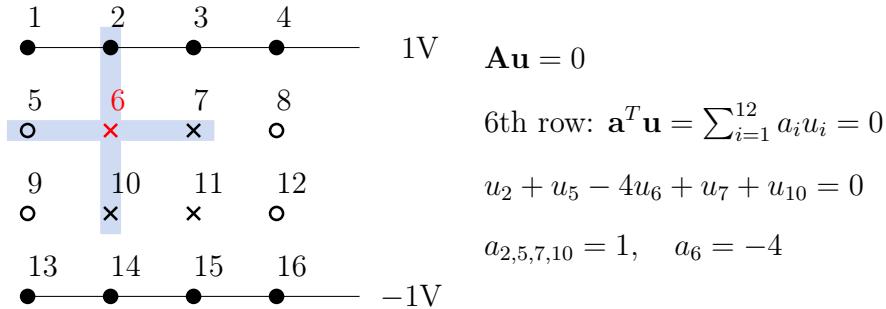


图 4.6 FDM 方程组求解的矩阵形式。授课时，学生不能直观的理解 FDM 方程的矩阵形式，特绘图示之。

应用迭代法，需要给定

1. 初始条件：给出所有内部格点一个初始值，如 $f_{i,j} = 0$
2. 边界条件：位于边界上的格点在迭代过程中固定为预先设定的值
3. 收敛条件：当我们的迭代解满足式(4.8)时退出

1. 基本原理

Jacobi 迭代法将式(4.8)写为

$$f_{i,j} = \frac{1}{4} (f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1}) \quad (4.9)$$

也就是说对于每一个格点，其电势都是周围四个格点的平均值。Jacobi 迭代法基于此设定下一次的迭代值

$$f_{i,j}^{(n+1)} = \frac{1}{4} (f_{i-1,j}^{(n)} + f_{i+1,j}^{(n)} + f_{i,j-1}^{(n)} + f_{i,j+1}^{(n)}) \quad (4.10)$$

式(4.10)的收敛很慢，而且需要开辟两个独立的存储区分别存储 $f^{(n)}$ 与 f^{n+1} ，需要较多的存储资源。

Gauss-Seidel 迭代法与 Jacobi 迭代法类似，设 f 的更新是按照 i, j 递增的方式进行，则已经更新的数据，如 $f_{i-1,j}$ 和 $f_{i,j-1}$ ，覆盖掉原先的存储数据，在

下一次迭代计算 $f_{i,j}$ 时可直接使用，即

$$f_{i,j}^{(n+1)} = \frac{1}{4} \left(f_{i-1,j}^{(n+1)} + f_{i+1,j}^{(n)} + f_{i,j-1}^{(n+1)} + f_{i,j+1}^{(n)} \right) \quad (4.11)$$

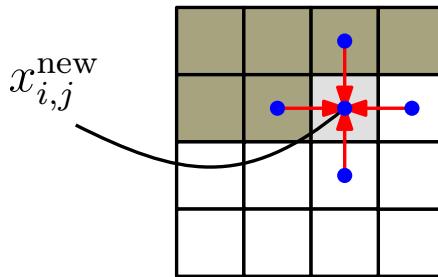


图 4.7 Gauss–Seidel 迭代方法，覆盖已更新的存储区

另外一种改进算法被称作松弛 (over-relaxation) 算法，

$$f_{i,j}^{(n+1)} = f_{i,j}^{(n)} + R \left[\frac{f_{i-1,j}^{(n+1)} + f_{i+1,j}^{(n)} + f_{i,j-1}^{(n+1)} + f_{i,j+1}^{(n)}}{4} - f_{i,j}^{(n)} \right] \quad (4.12)$$

松弛因子 $R > 1$ 可增加算法的收敛速度。但是基于数值稳定性考虑， R 一般小于 2。针对 laplace 方程，松弛算法的最优值 R 与 FDM 方法在某一维的格点数目 N 相关，即

$$R_{opt} = 2 - c/N$$

其中 c 是与 N 无关的、与待求解区域几何形状相关的一个变量。

2. 计算步骤

现在已经具备了计算 2D 拉普拉斯方程的所有条件，下面是具体实现步骤

1. **生成网格**。导体的边界刚好置于格点上。对于本例题，可以利用对称性，减少待求解的未知数，只计算右上部 $1/4$ 的网格
2. **边界条件**。在外导的格点上令 $f = 0$ ，在内导的格点上有 $f = V = 1$
3. **计算区域**。建立一个数组来标记格点是否位于计算区域内。区域内的格点由 laplace 方程计算电势值
4. **求解算法**。利用 Gauss–Seidel 算法迭代计算格点上的电势值
5. **电磁参量**。单位长度的电容值为 $C = Q/V = Q$ 。因此我们只需计算内导的电荷数 Q 即可， Q 可由高斯定理计算

$$Q = \epsilon_0 \int_S \nabla \cdot E dS = \epsilon_0 \oint_L E \cdot \hat{n} dl = -\epsilon_0 \oint_L \frac{\partial \varphi}{\partial n} dl$$

其中积分区域 L 设定为环绕内导的环线

6. **退出条件。**如果电容 C 的变化率在两次迭代中变动很小，则停止迭代
7. **数据外推。**计算完成后，继续细分网格重新计算步骤 1–6。最后，通过数据外推得到 h_0 的数值计算结果

3. 算法实现

在这个例题中，我们设定 $a = b = 1\text{cm}$, $c = d = 2\text{cm}$ 。首先我们根据式(4.12)编写松弛迭代算法。这里为了利用平面的对称性，我们只对右上角 $1/4$ 的区域进行计算，如图4.8所示。

- Gauss–Seidel update
- Symmetric boundary
- ✗ Fixed Boundary Conditions

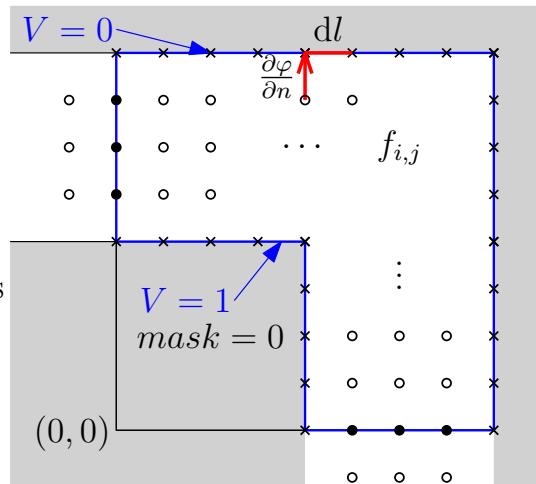


图 4.8 同轴传输线电容的 FDM 计算程序编程示意

在 Gauss–Seidel 迭代计算中，我们通过 mask 变量将区域划分为计算区域和内导体区域（不参与迭代更新计算）。同时在边界点的计算上，利用对称性，例如对于左边边界上的点有 $f_{i-1,j} = f_{i+1,j}$ ；而对于右下边界上的点，有 $f_{i,j-1} = f_{i,j+1}$ 。

在计算电荷量——这一电磁参量时，我们有（ L 取外导边界）

$$\oint_L \frac{\partial \varphi}{\partial n} dl = \sum_i \frac{\partial \varphi}{\partial n} h = \sum_{j=m,i} \frac{\varphi_{i,j} - \varphi_{i,j+1}}{h} \cdot h + \sum_{i=n,j} \frac{\varphi_{i,j} - \varphi_{i+1,j}}{h} \cdot h$$

由于在外导体上 $V = 0$ ，则 $\varphi_{i,m+1} = 0$, $\varphi_{n+1,j} = 0$ ，那么

$$\oint_L \frac{\partial \varphi}{\partial n} dl = \sum_{j=m,i} \varphi_{i,j} + \sum_{i=n,j} \varphi_{i,j}$$

实际计算时，为了避免重复计算 $1/4$ 计算区域两端上的格点，我们可以在边界

格点上取 h 线段中点的平均值近似一阶差分，即

$$\oint_L \frac{\partial \varphi}{\partial n} dl = \sum_{j=m,i} \frac{\varphi_{i,j} + \varphi_{i+1,j}}{2} + \sum_{i=n,j} \frac{\varphi_{i,j} + \varphi_{i,j+1}}{2}$$

程序中定收敛精度为 10^{-1} ，松弛因子 $R = 1.9$ ，仿真结果见图4.9。

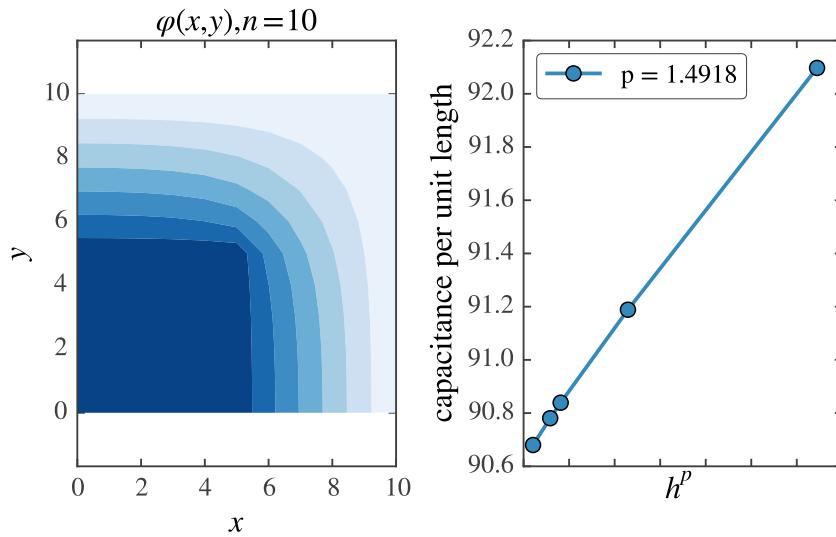


图 4.9 2D 同轴传输线电容的 FDM 计算结果

在图4.9中，数值解的收敛阶数 $O(h^p)$ 近似为 $p = 1.5$ ，达不到 $O(h^2)$ 的收敛特性。事实上，交错格点 $O(h^2)$ 这一理想收敛特性，只有在待求解的函数 f 足够规则（sufficient regular）时才能得到。

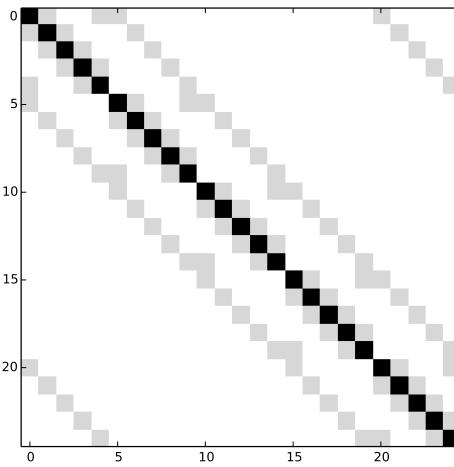
另一方面，从图4.9中可以看出，在内导角的位置上，电势近似与 r 的幂次方成反比。因而在内导体的凹角位置附近，存在数值计算的奇异点。数值奇异区域影响有限差分方法的收敛性。最后，假定收敛阶数 $p = 1.5$ ，利用该拟合值进行数据外推，可得 h_0 （即网格间距 $h \rightarrow 0$ ）下电容的估计值为 $C_0 = 90.65\text{pF/m}$ 。

4.2.6 迭代法与矩阵求解的关系（选讲）

我们将简单的解释下为什么 Jacobi 和 Gauss-Seidel 迭代法可以得到线性方程组的解 [11]。首先，FDM 方法可写作等效的线性方程组形式

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

其中矩阵 \mathbf{A} 见图4.10。在中心节点上（对角线上的元素）系数为 -4 ，对于上下

图 4.10 5×5 大小的 FDM 等效矩阵

左右的邻近节点系数为 1。矩阵 \mathbf{A} 是严格对角占优的，即 $|a_{ii}| \geq \sum_{j \neq i} |a_{i,j}|$ 。在 Laplace 方程中， $|a_{ii}| = 4 = \sum_{j \neq i} |a_{i,j}|$ 。

迭代法每次近似的求一个解，如利用 $\mathbf{a}_i^T \mathbf{x} = b_i$ 近似的计算 x_i

$$x_i = \frac{b_i - \sum_{j,j \neq i} a_{ij} x_j}{a_{ii}}$$

Jacobi 迭代法使用所有的 $x_j^{(n)}$ 数据计算 $x_i^{(n+1)}$ ，而 Gauss-Seidel 算法则对所有的 $(j < i)$ 使用已经计算出来的结果 $x_{j,j < i}^{(n+1)}$ 。因而对于大型矩阵，尤其是稀疏矩阵，迭代法比直接求解法具有较小的计算量。

我们希望迭代法得到的解序列 $\{\mathbf{x}^{(k)}\}$ 能够逼近真实解 $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ ，也就是说两者最终能够得到近似的数值计算结果。假设在从 k 到 $k + 1$ 步的迭代过程中，每一次迭代都可减少余量（Residual） $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ ，则

$$\Delta\mathbf{x} = \mathbf{x}^* - \mathbf{x}^{(k)} = \mathbf{A}^{-1}\mathbf{r}$$

如果能够精确的计算到 $\Delta\mathbf{x}$ ，则可一步从 $\mathbf{x}^{(k)}$ 得到 \mathbf{x}^* ，见图 4.11。

实际应用中，矩阵 \mathbf{A} 是一个大型的稀疏矩阵，直接计算 \mathbf{A}^{-1} 是很困难的。为此，可暂时令 $\Delta\mathbf{x}' \approx \mathbf{M}^{-1}\mathbf{r}$ ，其中 \mathbf{M} 是一个非奇异的、便于方程组求解的构造矩阵，设 $\{\mathbf{x}^{(k)}\}$ 能够收敛到 \mathbf{x}^* ，则有 $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}' = \mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{r}$ 即

$$\mathbf{x}^{(k+1)} = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{A})\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{b} \quad (4.13)$$

$$= \mathbf{G}\mathbf{x}^{(k)} + \mathbf{g} \quad (4.14)$$

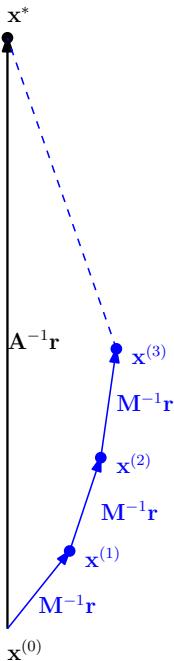


图 4.11 Jacobi 迭代法与矩阵求解

其中 $\mathbf{G} = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{A})$ 被称作迭代矩阵, $\mathbf{g} = \mathbf{M}^{-1}\mathbf{b}$ 。

1. Jacobi 迭代方法

在构造矩阵的基础上, 我们分析 Jacobi 和 Gauss–Seidel 迭代法。

首先, 将 \mathbf{A} 写为 $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$, 其中 \mathbf{D} 是对角矩阵, \mathbf{L} 是严格的下三角矩阵, \mathbf{U} 是严格的上三角矩阵。那么 Jacobi 迭代等价为: 令 $\mathbf{M} = \mathbf{D}$,

$$\mathbf{x}^{(k)} = \mathbf{D}^{-1}(\mathbf{U} + \mathbf{L})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b} \quad (4.15)$$

这里要求 $D_{ii} \neq 0$ 。在此基础上, 我们可方便的写成 Jacobi 迭代的形式

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n \quad (4.16)$$

2. 高斯迭代方法

Gauss–Seidel 迭代方法令 $\mathbf{M} = \mathbf{D} - \mathbf{L}$, 则

$$\mathbf{x}^{(k+1)} = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U} \mathbf{x}^{(k)} + (\mathbf{D} - \mathbf{L})^{-1} \mathbf{b} \quad (4.17)$$

为了得到迭代式，我们将式(4.17)写为

$$\mathbf{D}\mathbf{x}^{(k+1)} - \mathbf{L}\mathbf{x}^{(k+1)} = \mathbf{U}\mathbf{x}^{(k)} + \mathbf{b} \quad (4.18)$$

则高斯迭代为

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n \quad (4.19)$$

最后，迭代法求解线性方程组是收敛的 [12]，当且仅当 \mathbf{A} 满足：

1. \mathbf{A} 是正定的，
2. \mathbf{A} 严格对角占优 (Strict Diagonally Dominant)，即 $|a_{ii}| \geq \sum_{j \neq i} |a_{i,j}|$ 。

4.2.7 网格间距与数值误差 (选讲)

在编写 FDM 电磁场数值计算程序时，同学们需要掌握网格间距 h 的设定方法，并且理解数值计算的误差（也被称作数值色散 Numerical Dispersion）。同时牢记，任何数值计算方法都是对物理中偏微分方程的一种近似，因而都存在（有些情形下较大）数值误差。

对于函数 f ，定义归一化一阶微分 D_x 和二阶微分 D_{xx} 为

$$D_x = f'/f,$$

$$D_{xx} = f''/f$$

则对于任意一个复正弦波 $f(x) = \exp(jkx)$ ，微分的解析解分别为 $D_x = jk$ 、 $D_{xx} = -k^2$ 。

若采用二格点 (two-cell) 差分近似微分，有

$$D_x = \frac{f(x_i + h) - f(x_i - h)}{2hf(x_i)} = \frac{j}{h} \sin kh$$

利用泰勒展开，可得到波束 k 的两格点数值解 $k_{\text{num}}^{\text{two-cell}}$ 为

$$k_{\text{num}}^{\text{two-cell}} = \frac{\sin kh}{h} = k \left(1 - \frac{k^2 h^2}{6} + \dots \right) \quad (4.20)$$

由于 kh 的定义域是 $[-\pi, \pi]$ ，当实际 $kh = \pi$ 时，数值方法的波数 $k_{\text{num}}^{\text{two-cell}} h$ 是 0，数值计算的结果会产生较大的偏差。因为当 $kh = \pi$ 时，间隔两个格点的差分对应于波峰减去波峰，波谷减去波谷。

另一方面，当 $\pi/2 < kh < \pi$ 时，波速 (group velocity) $\partial k_{\text{num}} / \partial k < 0$ ，其物理含义是波速是负的，也就是说数值计算得到的包络将沿相反方向传播，这也是不符合常理的。我们定义由数值方法导致的误差为数值色散 (Numerical Dispersion)。

若采用交错格点 (staggered grids)，根据差分的定义

$$D_x = \frac{f(x_i + h) - f(x_i)}{hf(x_i + h/2)} = \frac{2j}{h} \sin \frac{kh}{2}$$

则 k 的数值解为

$$k_{\text{num}}^{\text{staggered}} = \frac{2}{h} \sin \frac{kh}{2} = k \left(1 - \frac{k^2 h^2}{24} + \dots \right) \quad (4.21)$$

当 $kh = \pi$ 时，采用 staggered grids 的数值计算结果是 $k_{\text{num}}^{\text{staggered}} h = 2$ 。虽然相比于真实值 π 有一定 (较大) 的数值误差，但是该误差是单调变化 (递增) 的，而且采用 staggered grids 不会产生负的波速 (同学们可试着计算 $\partial k_{\text{num}} / \partial k$)。

对于二阶导数 D_{xx} 。

$$D_{xx} = \frac{\exp(jkh) - 2 + \exp(-jkh)}{h^2} = -\frac{4}{h^2} \sin^2 \frac{kh}{2}$$

可得交错格点下

$$k_{\text{num}}^2 = \frac{4}{h^2} \sin^2 \frac{kh}{2} = k^2 \left(1 - \frac{k^2 h^2}{12} + \dots \right) \quad (4.22)$$

对于真实解 $D_{xx} = -k^2$ ，当 $kh = \pi$ 时，数值解仅为 $D_{xx} = -4/h^2$ 。此时，数值解与真实解的误差最小。同时我们也可看出，采用交错格点的数值解能够单调递增的逼近真实解 π^2/h^2 。

在数值计算程序中，为了保证与真实波数 (频率) 1% 的精确度，利用泰勒展开式(4.22)的前两项逼近，需 D_{xx} 的精度在 2% 以内，即 $k^2 h^2 < 0.24$ ，则

$$\frac{1}{\omega h} \geq \frac{2\pi}{\sqrt{0.24}} \approx 4\pi$$

其物理意义为，每一个波长 $1/\omega$ 内，FDM 方法至少需要包括 13 个格点。

例题 4.3: 对于 $f_s = 900 \text{ MHz}$ 的 GSM 波段，其波长为 $\lambda = 33\text{cm}$ ，则在 $5m$ 的空间内，每一维度的格点数目为 $13 \times 5 / 0.33 \approx 200$ 。三维空间 (立方体) 的网格数目将达到 8×10^6 。

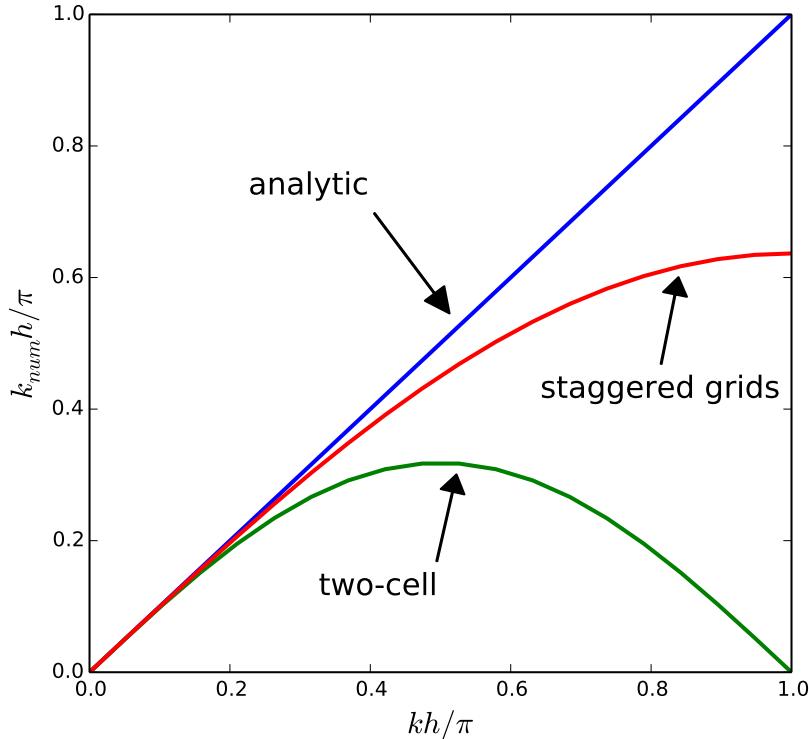


图 4.12 FDM 方法的数值误差。令 $f = \exp(jkx)$, 通过计算一阶微分可得 $k = -jff'/f$, 图中分别给出了两格点法和交错格点法的数值误差。注意到当 $kh \rightarrow \pi$ 时, 两格点法具有较大的数值误差。

4.3 时域有限差分法

时域有限差分方法 (FDTD, Finite Difference Time Domain) 最早由 K. S. Yee 于 1966 年提出, FDTD 方法与 FEM 方法进行对比 (互有优劣):

1. FDTD(Finite Difference Time Domain Method) 支持显式的时域 (explicit time advancing) 更新计算; 而 FEM 方法只能 implicit 的更新计算。
2. 无需存储系数矩阵 (或 Stiffness matrix, 在 FEM 章节中将讲解), 存储量小, 适宜求解大型的数值计算问题; FEM 需要存储和计算 Stiffness 系数矩阵, 对于大型问题计算量大, 耗时多。
3. 缺点在于仅适用于直角坐标系、形状规则的计算用例, 对于边界条件的处

理不佳；FEM 方法适用于不规则边界数值计算问题。

FDTD 适用于求解数值计算问题中特征尺寸长度与波长 λ 可比拟的计算场景。除此以外，FDTD 方法还可以对时域计算的结果进行傅里叶变换，得到特征频率。而进行傅里叶变换可以仅使用有限个时域计算结果，便于分析数值计算问题的特性。

接下来我们首先回顾一下 FDTD 方法中用到的基本电磁场理论，随后讲解一维 Helmholtz 方程的特征值（Eigen）、频域（Frequency）和时域（Time）的数值分析方法，以及二维 Helmholtz 的计算实例，最后介绍 Maxwell 方程的 leapfrog 更新计算方法，并且直观的给出三维 Yee 氏网格的计算方法和物理意义。

4.3.1 理论要点回顾

本章中重点讲两个理论要点。首先是从 Maxwell 公式到一阶 $\mathbf{E} - \mathbf{H}$ 偏微分方程，我们分别从安培定理（Ampère's law）和法拉利定理（Faraday's law）展开。

Ampère's law:

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$$

Faraday's law:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

将时域微分部分放在一边：

$$\epsilon \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} - \mathbf{J} \quad (4.23)$$

$$\mu \frac{\partial \mathbf{H}}{\partial t} = -\nabla \times \mathbf{E} \quad (4.24)$$

另外一个知识点是 Curl-curl 方程，频域表示即为大名鼎鼎的 Helmholtz 方程。同样可从 Maxwell 方程中得到（只要在方程两边求 $\partial/\partial t$ ，再将 H 替代 E 即可）

$$\epsilon \frac{\partial^2 \mathbf{E}}{\partial t^2} + \nabla \times \frac{1}{\mu} \nabla \times \mathbf{E} = -\frac{\partial \mathbf{J}}{\partial t} \quad (4.25)$$

本章将针对 curl-curl 等式的一维情形进行教学（同学们需要巩固 $\nabla \times$ 旋度算子的代数形式，同时 $c = 1/\sqrt{\epsilon\mu}$ ）：

$$\frac{\partial^2 \mathbf{E}}{\partial t^2} = c^2 \frac{\partial^2 \mathbf{E}}{\partial z^2} \quad (4.26)$$

采用频域近似 $\partial/\partial t = j\omega$ 可得

$$-\omega^2 E(z) = c^2 \frac{\partial^2}{\partial z^2} E(z) \quad (4.27)$$

时谐形式即为一维 Helmholtz 方程

$$\frac{d^2 f}{dx^2} = -k^2 f, \quad 0 < x < a, \quad f(0) = f(a) = 0 \quad (4.28)$$

其中 $f(0)$ 和 $f(a)$ 为边界条件。

4.3.2 特征值方法

这部分先讲的原因是，在特征值（Eigen）分解分析中，还未涉及时域更新公式，因而可以直接利用 FDM 方法得到，知识体系具备一定的连续性。

根据 1D Helmholtz 方程(4.28)，我们将区间划分为 N 个格点，格点间隔设定为 h ，则其差分形式为

$$\frac{f_{i-1} - 2f_i + f_{i+1}}{h^2} = -k^2 f_i, \quad i = 1, 2, \dots, N-1 \quad (4.29)$$

给定边界条件 $f_0 = f_N = 0$ ，上式可写作矩阵的形式

$$A\mathbf{f} = \lambda \mathbf{f}$$

A 的形式很简单，在有限差分方法中已经介绍过了， $\lambda = -k^2$ 叫做矩阵 A 的特征值。

那么 1D Helmholtz 方程的解析解是什么呢？

$$f = A \cos kx + B \sin kx$$

套用边界条件可解得 $\sin ka = 0$ （注意这里 a 为区间长度），即

$$k_m = \frac{m\pi}{a}, \quad m \text{ an integer}$$

通过对矩阵 A 进行特征值分解即可求得 Helmholtz 方程的数值波数 k 。

```
def HFD1D(a, N):
    """
    a : length of interval
    N : number of grids
    return : eigen values
    """
    h = float(a)/N
```

```

A = np.zeros((N-1,N-1))
d = -2/h**2
s = 1/h**2

for i in range(N-1):
    A[i,i] = d

for i in range(N-2):
    A[i,i+1] = s
    A[i+1,i] = s

lam,v = np.linalg.eig(A)
k = np.sqrt(np.sort(-lam))

return k

```

表 4.1 1D Helmholtz 方程的特征值

	$m = 1$	$m = 2$
$N = 5$	0.98363164	1.87097857
$N = 10$	0.99589274	1.96726329
$N = 20$	0.99897223	1.99178547
$N = 100$	0.99995888	1.99967103
$N = 1000$	0.99998972	1.99991775

同学们可进行外推计算，验证解的正确性。

4.3.3 频域方法

例题 4.4：计算二维 Helmholtz 方程

$$\nabla^2 E + \left(\frac{k}{n}\right)^2 E = f$$

其中 $k = 2\pi/\lambda$ ，而 n 为电磁波的折射率 (refraction index)， f 为激励源所在的位置。我们计算图4.13中划定区域的电磁场的能量分布。在该例题中，设空气中电磁波的折射率 $n_{\text{air}} = 1.0$ ，墙壁的折射率 $n_{\text{wall}} = 2.55 - 0.01j$ 。激励源位于

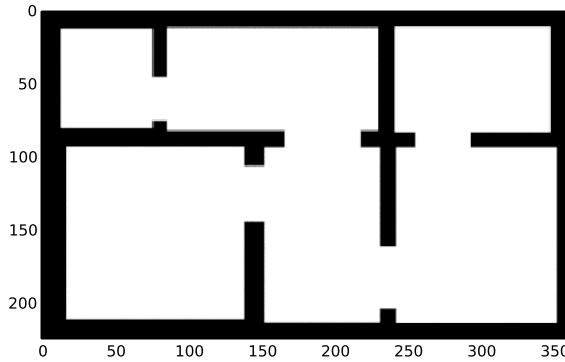


图 4.13 2D Helmholtz 方程计算区域

坐标 (80, 160) 位置处，其辐射能量恒为 1。

对于二维 Helmholtz 方程，我们有

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} + \left(\frac{k}{n}\right)^2 E = f$$

即

$$\frac{E|_{p+1,q} + E|_{p-1,q} + E|_{p,q+1} + E|_{p,q+1} - 4E|_{p,q}}{h^2} + \left(\frac{k}{n}\right)^2 E|_{p,q} = f \quad (4.30)$$

可写为矩阵的形式

$$\mathbf{A}\mathbf{E} = \mathbf{f}$$

因而只需求解这一线性方程组，即可得到 2D Helmholtz 方程的频域解，见图 4.14。并且通过数值计算得到 E ，可绘制电磁波的能量分布。

4.3.4 时域更新计算方法

在一维 curl-curl 方程中，有

$$\frac{\partial^2 \mathbf{E}}{\partial t^2} = c^2 \frac{\partial^2 \mathbf{E}}{\partial x^2} \quad (4.31)$$

不失一般性，我们忽略 c^2 项，将上式写作类似一维 Helmholtz 方程的形式

$$\frac{\partial^2 f}{\partial t^2} = \frac{\partial^2 f}{\partial x^2} \quad (4.32)$$

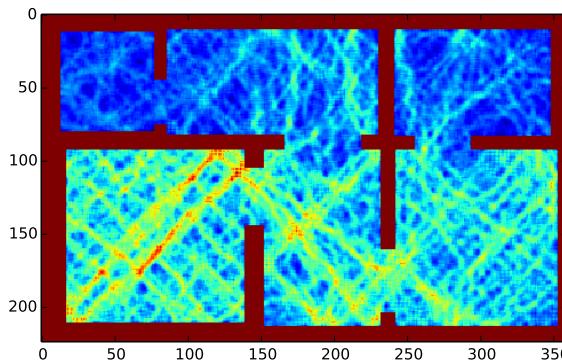


图 4.14 2D Helmholtz 方程数值计算结果

定义 L 为差分线性算子 $L = \partial^2/\partial x^2$ (线性算子可用矩阵 A 表示), 则

$$L[f] = \frac{\partial^2 f}{\partial t^2}, \quad \text{时域} \quad (4.33)$$

$$L[f] = -\omega^2 f, \quad \text{频域} \quad (4.34)$$

对于时域微分 $\partial^2 f / \partial t^2$, 进一步利用差分近似可得

$$\frac{f^{(n+1)} - 2f^{(n)} + f^{(n-1)}}{(\Delta t)^2} = L[f^{(n)}]$$

进而可得显式 (explicit) 时域更新公式

$$f^{(n+1)} = 2f^{(n)} - f^{(n-1)} + (\Delta t)^2 L[f^{(n)}] \quad (4.35)$$

其中

$$L[f^{(n)}] = \frac{f_{i+1}^{(n)} - 2f_i^{(n)} + f_{i-1}^{(n)}}{(\Delta x)^2} \quad (4.36)$$

为了进行时域计算, 我们有

$$f_i^{(n+1)} = 2f_i^{(n)} - f_i^{(n-1)} + \left(\frac{\Delta t}{\Delta x}\right)^2 \left(f_{i+1}^{(n)} - 2f_i^{(n)} + f_{i-1}^{(n)}\right) \quad (4.37)$$

式(4.37)给出的显式的时域更新方法也叫做蛙跳 (leapfrog) 计算, 在 Yee 氏网格一节将会详细讲解。

FDTD 方法的另一个应用是频域特性分析。利用式(4.37)进行时域更新, 计算得到 N 点数据并进行傅里叶变换, 即可得电磁场数值问题的频率特性。

例题 4.5: 考虑 Helmholtz 方程

$$\frac{\partial^2 f}{\partial t^2} = \frac{\partial^2 f}{\partial x^2}, \quad 0 < x < a, \quad f(0, t) = f(a, t) = 0$$

其特征频率为

$$\omega_m = \frac{m\pi}{a}, \quad m = 1, 2, \dots$$

我们用时域更新公式

$$f_i^{(n+1)} = 2f_i^{(n)} - f_i^{(n-1)} + \left(\frac{\Delta t}{\Delta x}\right)^2 \left(f_{i+1}^{(n)} + f_{i-1}^{(n)} - 2f_i^{(n)}\right)$$

进行数值计算并记录 $a/5$ 位置处和 $a/2$ 位置处的仿真波形，随后利用傅里叶变换分析这两点的频谱特性。

```
import numpy as np

def Wave1D(a, time, nx):
    """
    Input:
        a      = the length of the interval
        time  = the total time interval for the simulation
        nx    = the number of grids intervals in (0,a)
    Output:
        omega = the angular frequencies
        s1    = the complex FFT transform of data at x = a/5
        s2    = the complex FFT transform of data at x = a/2
    """
    # initialize two time samples
    f0      = np.random.randn(nx+1)
    f0[0]   = 0
    f0[nx]  = 0

    f1      = np.random.randn(nx+1)
    f1[0]   = 0
    f1[nx]  = 0

    dx      = float(a) / nx
    d2tmax = 1.9*dx
    ntime  = int(np.round(float(time)/d2tmax + 1))
```

```

dt      = float(time) / (2*ntime)

A       = np.zeros((nx+1, nx+1))
for i in range(1,nx):
    A[i,i]   = 2*(1 - (dt/dx)**2)
    A[i,i+1] = (dt/dx)**2
    A[i,i-1] = (dt/dx)**2

# every time means two time steps 'dt'
sig1 = np.zeros(2*ntime)
sig2 = np.zeros(2*ntime)
for itime in range(ntime):
    f0          = np.dot(A, f1) - f0
    sig1[2*itime-1] = f0[int(round(nx/5))]
    sig2[2*itime-1] = f0[int(round(nx/2))]
    f1          = np.dot(A, f0) - f1
    sig1[2*itime]   = f1[int(round(nx/5))]
    sig2[2*itime]   = f1[int(round(nx/2))]

# compute discrete FFT
spectr1 = np.fft.fft(sig1, n=2*ntime)
spectr2 = np.fft.fft(sig2, n=2*ntime)

# record only the first half of the outputs
s1 = spectr1[:ntime]
s2 = spectr2[:ntime]

# frequency vector for use with s1, s2
omega = (2*np.pi/time) * np.linspace(0, ntime-1, ntime)

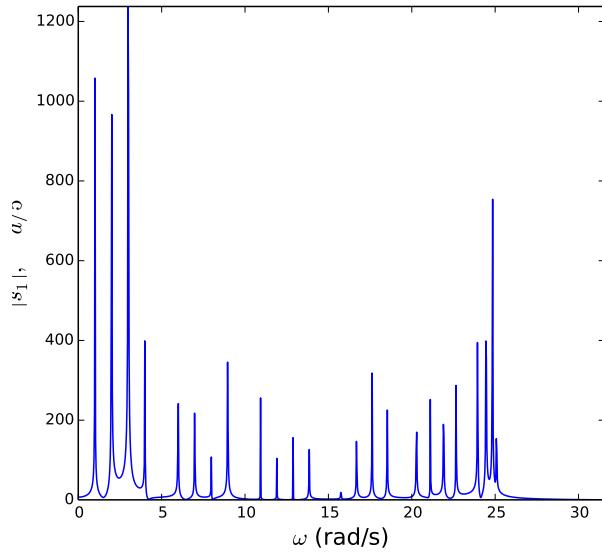
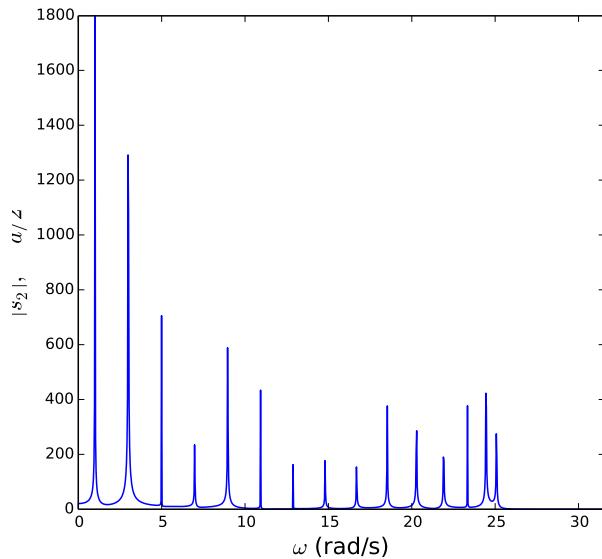
return omega, s1, s2

```

其计算结果可分别见图4.15与图4.16。

图4.16与图4.15中，特征频率所在的点非常接近整数格点 $m\pi/a$ 。而特别需要注意的是图4.16中，没有出现偶数的特征频率；在图4.15中，每隔4个特征频率即消失一个特征频率。这一点跟 FDTD 频率分析的特定位置是相关的，在这些位置上，电磁波不具备相应的特征模式。

这个例子中，也展示了 FDTD 频率分析方法的优势：即我们只需运行一定

图 4.15 FDTD 频域仿真结果 ($a/5$)图 4.16 FDTD 频域仿真结果 ($a/2$)

(一般都很短) 时间的 FDTD 仿真, 通过傅里叶变换即可得到系统的特征频率(也被称作模式)信息。

4.3.5 数值稳定性与数值色散

FDTD 方法中时域更新步长 Δt 应该怎么选择呢？接下来我们给出 FDTD 方法的稳定性分析（设时域网格间隔 Δt 、空域网格大小 h ）。

我们令 $f^{(n)} = f_w \rho^n$ ，为了保证时域更新是稳定的，需要 $|\rho| \leq 1$ （想想看为什么，可考虑 $n \rightarrow \infty$ ）。则利用时域更新表达式和频域表达式，可得

$$f_w \rho^{(n+1)} = 2f_w \rho^{(n)} - f_w \rho^{(n-1)} - \omega^2 (\Delta t)^2 f_w \rho^{(n)}$$

进一步

$$\rho^2 - [2 - \omega^2 (\Delta t)^2] \rho + 1 = 0 \quad (4.38)$$

式(4.38)中，当 $(\omega \Delta t)^2 > 4$ 时，方程有两个实根 ρ_1, ρ_2 且乘积为 $\rho_1 \rho_2 = 1$ ，则必有一个根 $\rho > 1$ ，导致时域更新的结果发散。当 $(\omega \Delta t)^2 < 4$ ，有两个虚根且共轭对称 $\rho_1 = \rho_2^*$ ，两个根的模 $|\rho|$ 刚好位于单位圆上，此时数值计算收敛。

进而，FDTD 方法的收敛条件为（对所有的 ω 收敛，即对最大的收敛）

$$\Delta t \leq \frac{2}{|\omega_{\max}|} \quad (4.39)$$

而在有限差分 $L[f]$ 中，最大的 k_{\max} 为 $k_{\max}^2 = 4/h^2$ 。类似的可令 $f = \exp(j\omega t)$ ，则有限差分计算中，最大的频率为 $\omega_{\max} = \sqrt{k_{\max}} = 2/h$ 。为了保证时域更新计算的稳定性，要求（注意此时方程中假定 $c^2 = 1$ ）

$$\Delta t \leq h$$

下面我们再以一维 curl-curl 方程为例分析数值稳定性和数值色散。首先，一维中网格格点（t-z）的划分见图4.17。

在图4.17中，定义 $|_r$ 为 z 轴的格点坐标， $|^n$ 为时域坐标，则有 $E|_r^n = E(r\Delta z, n\Delta t)$ 。利用 staggered grids，可写出一维差分计算公式

$$\frac{E_r^{n+1} - 2E_r^n + E_r^{n-1}}{(\Delta t)^2} = c^2 \frac{E_{r+1}^n - 2E_r^n + E_{r-1}^n}{(\Delta z)^2} \quad (4.40)$$

与之前的分析类似，可得到 $E|_r^{n+1}$ 的显式更新公式

$$E|_r^{n+1} = 2E_r^n - E_r^{n-1} + \left(\frac{c\Delta t}{\Delta z} \right)^2 (E_{r+1}^n - 2E_r^n + E_{r-1}^n) \quad (4.41)$$

式(4.41)中，需要两个时刻的 E 作为初始条件。而算法实现中，为了得到数值结果，需要给出 E 和 $\partial E / \partial t$ 在 z 和 $t = 0$ 的初始值。

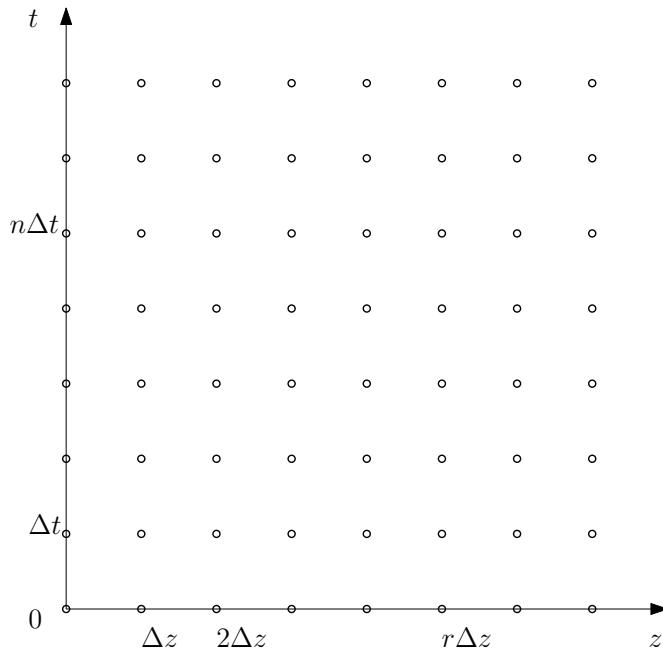


图 4.17 1D wave 公式的网格格点

令 $E_r^n = \exp(j\omega n\Delta t - jkr\Delta z)$, 代入式(4.40)中可得

$$\left(\frac{e^{j\omega\Delta t/2} - e^{-j\omega\Delta t/2}}{2j} \right)^2 = \left(\frac{c\Delta t}{\Delta z} \right)^2 \left(\frac{e^{jk\Delta z/2} - e^{-jk\Delta z/2}}{2j} \right)^2$$

开平方可得

$$\sin \frac{\omega\Delta t}{2} = \pm \frac{c\Delta t}{\Delta z} \sin \frac{k\Delta z}{2} \quad (4.42)$$

当且仅当 $c\Delta t = \Delta z$ 时, 数值计算的结果 $\omega = k$, 否则 ω 仅仅是在数值计算中近似的等于波数 k 。也就是说, 随着 FDTD 方法时域的更新计算, 电磁波中不同的频率成份将以不同的速度传播。这一点被称作数值色散 (numerical dispersion)。而 $c\Delta t = \Delta z$ 则被称作 magic time step。

令 $R = c\Delta t/\Delta z$, 若 $R > 1$, 则 $|\sin \omega\Delta t/2| > c\Delta t/\Delta z = R$, 即时域更新得到复数 (complex valued) 的数值结果, 且数值解不稳定 (发散)。从另一个角度看其含义为, 时域更新步长小于真实波传播的速度, 进而用显式的时域更新计算无法得到收敛的数值结果。若 $R < 1$, 则数值计算的结果是稳定的, 但是和真实情形下波的传播模式间存在 (一定的) 数值色散。

例题 4.6: 我们分别仿真矩形波和高斯包络的 1D FDTD 时域计算实例。

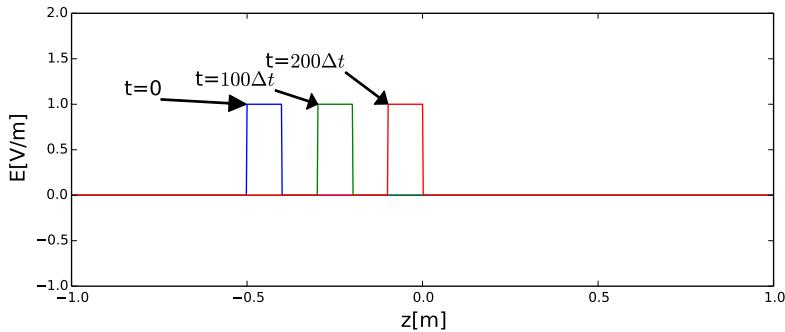


图 4.18 矩形波的 FDTD 数值稳定性。 $R = c\Delta t/\Delta z = 1.00$ ，图中在 magic time step 设置下，矩形波的传导没有失真。

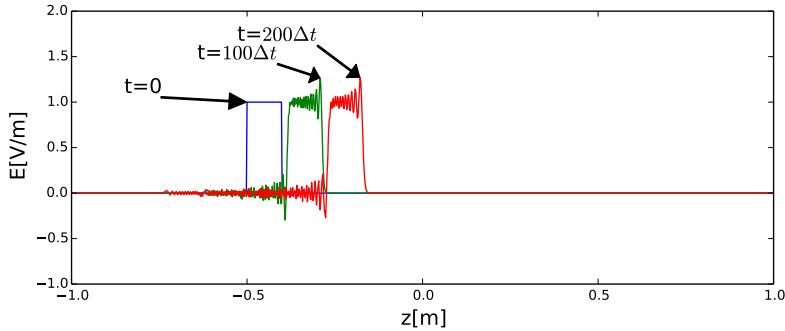


图 4.19 矩形波的 FDTD 数值稳定性。 $R = c\Delta t/\Delta z = 1/\sqrt{3} \approx 0.58$ ，可以看到矩形波的传播有明显的数值色散，同时波的包络也随之而改变。

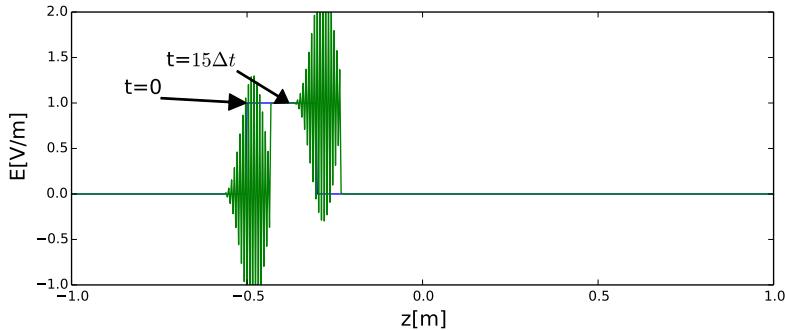


图 4.20 矩形波的 FDTD 数值稳定性。 $R = c\Delta t/\Delta z = 1.01$ ，此时的数值解是不稳定的，波的幅度随着仿真时间的增加而急剧变大。

矩形波由于包含了过多的频率成分，因而不能被有限的网格很好近似。与之对应的是，高斯包络 (Gaussian pulse) 却可以用网格近似，见图4.21。即便在 $R = 0.58$ 的设置下，随着波的传播数值色散很小。但是如果一个高斯宽度 $1/e$ 内采样点较少（即包含较多或较丰富的频率成分），则 FDTD 法仍会产生较大的数值色散现象，见图4.22。

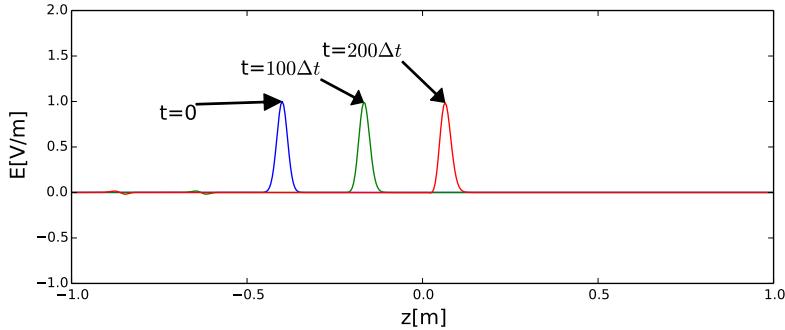


图 4.21 高斯包络的 FDTD 数值稳定性。在高斯包络 $1/e$ 的宽度内的采样点数为 12，即便对于 $R = c\Delta t/\Delta z = 0.58$ 数值色散也很小。

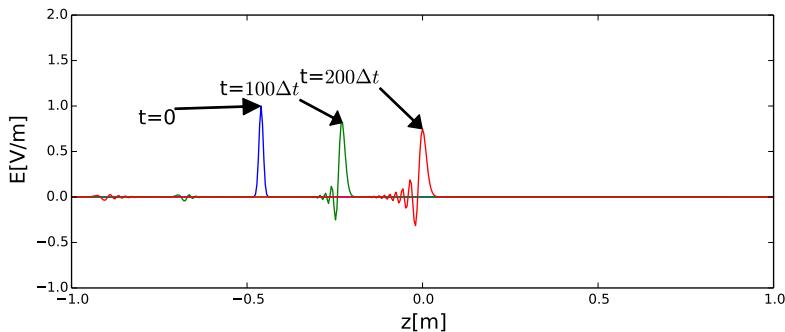


图 4.22 高斯包络的 FDTD 数值稳定性。在高斯包络 $1/e$ 的宽度内的采样点数为 6，由于采样点较少，所以包含了较多的频率成分，对于 $R = c\Delta t/\Delta z = 0.58$ 有较大数值色散。

4.3.6 蛙跳 (leapfrog) 更新计算方法

在 Maxwell 方程中，需要求解电场 E 的二阶微分方程（也被称作 curl-curl 方程）。另一方面，该方程也可用 E 和 H 的一阶偏微分方程描述。在自由空间中 ($\mathbf{J} = 0$)，可分别根据 Ampère's law 和 Faraday's law 得到相应的计算公式。

以三维空间为例，对于 Amère's law 有

$$\epsilon \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} = \begin{vmatrix} i & j & k \\ \partial/\partial x & \partial/\partial y & \partial/\partial z \\ H_x & H_y & H_z \end{vmatrix}$$

即

$$\epsilon \frac{\partial E_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \quad (4.43)$$

$$\epsilon \frac{\partial E_y}{\partial t} = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \quad (4.44)$$

$$\epsilon \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \quad (4.45)$$

共计六个未知参量，三个方程组。另外三个方程组可由 Faraday's law

$$\mu \frac{\partial \mathbf{H}}{\partial t} = -\nabla \times \mathbf{E}$$

得到

$$\mu \frac{\partial H_x}{\partial t} = \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \quad (4.46)$$

$$\mu \frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \quad (4.47)$$

$$\mu \frac{\partial H_z}{\partial t} = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \quad (4.48)$$

FDTD 方法本质上基于以上六个方程组（实际上是一个四维 $\{x, y, z, t\}$ 方程组）。基于 staggered grids 的差分形式，FDTD 的更新计算可具备中心性、局部性 (local and centered) 的特点。

我们以一维简化的例子来帮助同学们理解，讲解为什么 staggered grids 对于 FDTD 方法是极其重要和关键的。假设电磁场沿着 z 方向传播，传播过程中，在其余方向 (x, y) 上都为常量。假定电场 E 的方向指向 x 轴，而 H 则指向 y 轴 (http://en.wikipedia.org/wiki/Circular_polarization)，则

$$\epsilon \frac{\partial E_x}{\partial t} = -\frac{\partial H_y}{\partial z} \quad (4.49)$$

$$\mu \frac{\partial H_y}{\partial t} = -\frac{\partial E_x}{\partial z} \quad (4.50)$$

我们之前分析过，在 staggered grids 上可以较好的用差分近似一阶微分。即信号取值在整数格点上，一阶差分在半格点上计算得到。这一点对于时间和空

间都是适用的。如果我们设定将 E_x 在时间空间中都置于整数格点上，那么 H_y 需放于半格点上，见图4.23

$$n = t/\Delta t$$

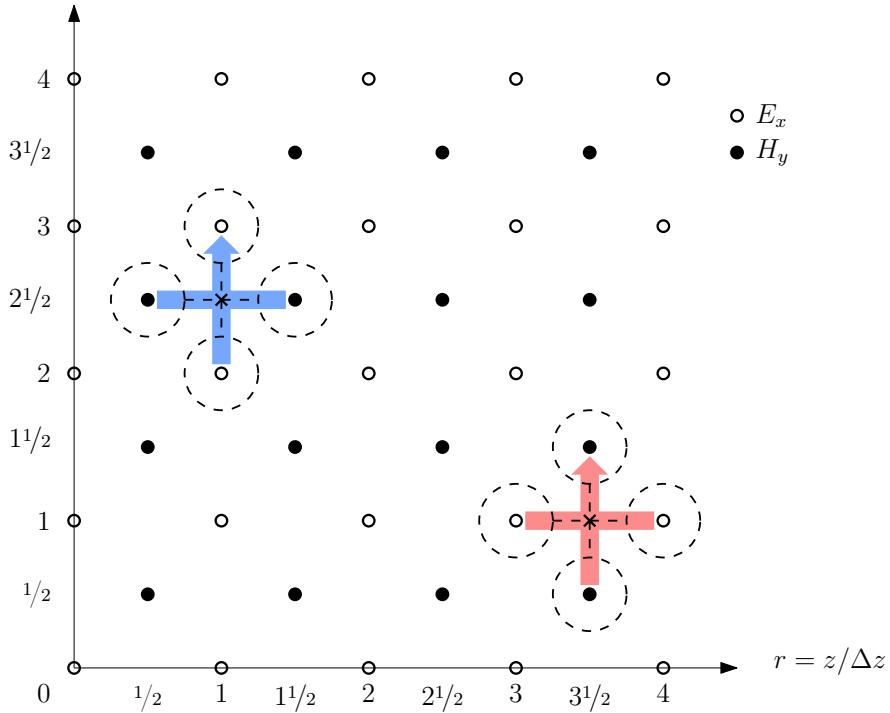


图 4.23 EH 一阶差分的蛙跳更新示意图

图4.23中，令 $|_r$ 代表 z 轴上的刻度， $|^n$ 代表时间轴上的刻度，则 $f|_r^n \equiv f(r\Delta z, n\Delta t)$ 。则（图中第一个十字）为

$$\frac{E_x|_r^{n+1} - E_x|_r^n}{\Delta t} = -\frac{1}{\epsilon} \frac{H_y|_{r+\frac{1}{2}}^{n+\frac{1}{2}} - H_y|_{r-\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta z} \quad (4.51)$$

同理（图中的第二个十字）

$$\frac{H_y|_{r+\frac{1}{2}}^{n+\frac{1}{2}} - H_y|_{r+\frac{1}{2}}^{n-\frac{1}{2}}}{\Delta t} = -\frac{1}{\mu} \frac{E_x|_{r+1}^n - E_x|_r^n}{\Delta z} \quad (4.52)$$

在进行计算时，需要设定 E_x 和 H_y 在初始时刻的值 $E_x^{(0)}$ 和 $H_y^{(0)}$ 。对于媒介中可变的 ϵ 和 μ ，需要记住 $\epsilon = \epsilon(z_r)$ 在整数格点上取值，而 $\mu = \mu(z_{r+\frac{1}{2}})$ 在半格点上取值。在这一过程中，所有参量的更新都满足中心化（center）、局部化（local）的特性。

另外根据式(4.51)和式(4.52)也可得到一维的波动方程，即

$$\begin{aligned} \frac{E_x|_r^{n+1} - 2E_x|_r^n + E_x|_r^{n-1}}{(\Delta t)^2} &= \frac{1}{\Delta t} \left(\frac{E_x|_r^{(n+1)} - E_x|_r^n}{\Delta t} - \frac{E_x|_r^n - E_x|_r^{(n-1)}}{\Delta t} \right) \\ &= \frac{1}{\epsilon\mu} \frac{1}{\Delta z} \left(\frac{E_x|_{r+1}^n - E_x|_r^n}{\Delta z} - \frac{E_x|_r^n - E_x|_{r-1}^n}{\Delta z} \right) \\ &= c^2 \frac{E_x|_{r+1}^n - 2E_x|_r^n + E_x|_{r-1}^n}{(\Delta z)^2} \end{aligned} \quad (4.53)$$

进而我们可以得出，采用 EH 一阶差分进行更新计算，同样需要满足 $\Delta t \leq \Delta z/c$ 这个 FDTD 方法的数值稳定条件。

定义 4.3.1: CFL 稳定性条件 (Courant–Friedrichs–Lowy condition) 为： $c\Delta t/h < 1/\sqrt{n}$ ，其中 n 为计算维度。对于三维情形 $c\Delta t < h/\sqrt{3}$ ，一般取 $c\Delta t < h/2$

若不使用 staggered grids，则式(4.53)中就会包含 $E_x|_{r+2}^n$ 和 $E_x|_{r-2}^n$ ，即网格中奇数编号和偶数编号是解耦的，两者的更新相互独立，不利于数值计算。

4.3.7 Yee 氏网格介绍

Yee 氏网格见图4.24。

图4.24中，电场 \mathbf{E} 在整数时间间隔上进行更新计算， \mathbf{H} 则在半时间格点上进行更新计算。三个坐标轴 $x - y - z$ 的网格划分分别为 $\Delta x = \Delta y = \Delta z = h$ 。在图4.24中，电场 E_x 在 x 轴上取半格点，而在 yz 轴上取整数格点。另一方面， H_x 则在 x 轴上取整数格点，在 yz 轴上取半格点。同理可以得到 E_y , E_z 以及 H_y , H_z 的格点设置。

令 $|_{p,q,r}$ 代表三轴的网格格点坐标，令 $|^n$ 代表时间坐标，则有 $f|_{p,q,r}^n \equiv f(p\Delta x, q\Delta y, r\Delta z, n\Delta t)$ 。我们可得到三维坐标系中 EH 的更新计算公式。

以 E_x 的更新计算为例 (见图4.25)。可得，

$$\begin{aligned} \epsilon \frac{E_x|_{p+\frac{1}{2},q,r}^{n+1} - E_x|_{p+\frac{1}{2},q,r}^n}{\Delta t} \\ = \frac{H_z|_{p+\frac{1}{2},q+\frac{1}{2},r}^{n+\frac{1}{2}} - H_z|_{p+\frac{1}{2},q-\frac{1}{2},r}^{n+\frac{1}{2}}}{\Delta y} - \frac{H_y|_{p+\frac{1}{2},q,r+\frac{1}{2}}^{n+\frac{1}{2}} - H_y|_{p+\frac{1}{2},q,r-\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta z} \end{aligned} \quad (4.54)$$

用右手螺旋定则来近似 $\nabla \times$ ，同理可得 E_y , E_z 及 H_x , H_y , H_z 的更新公式。

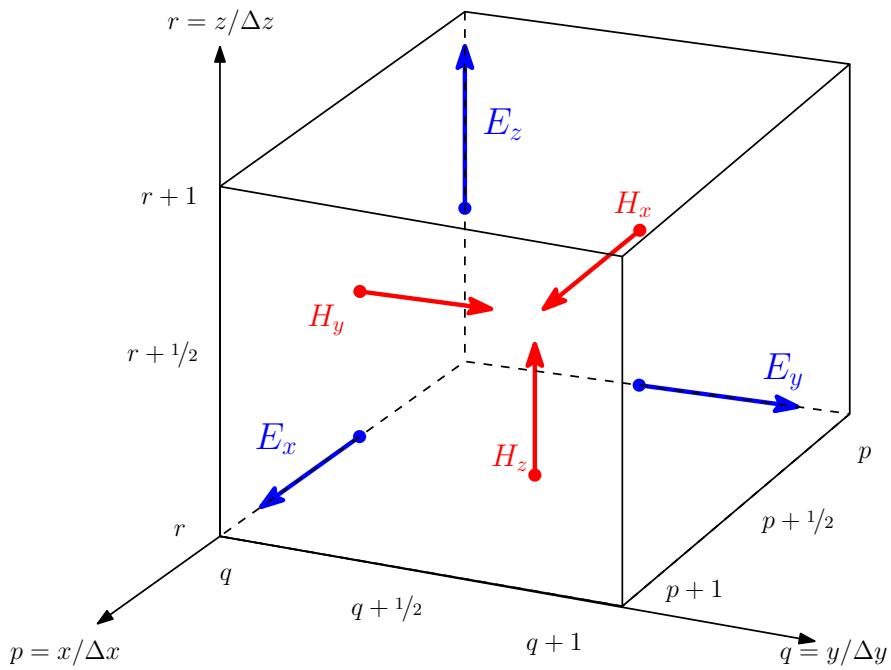


图 4.24 Yee 式单位网格

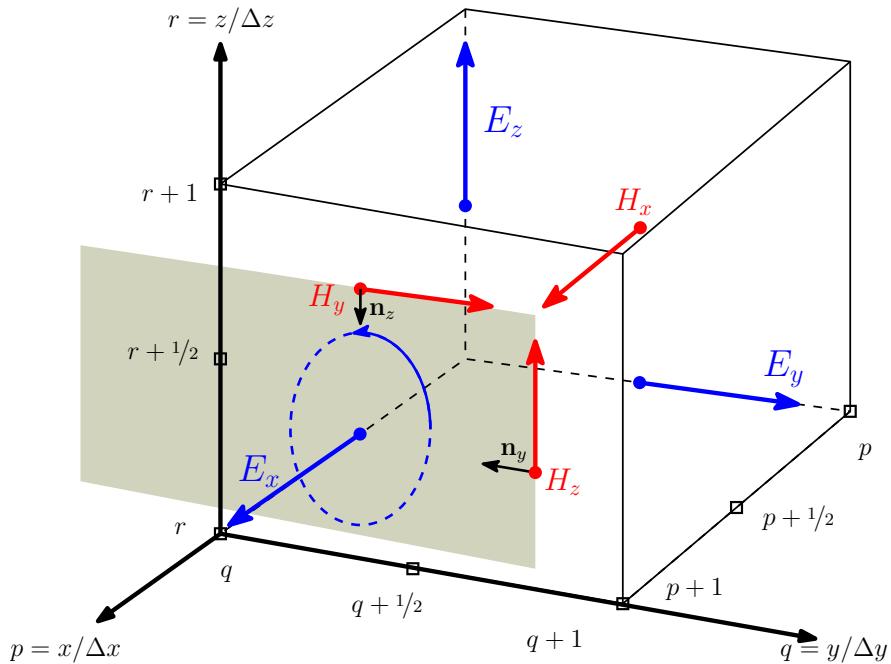


图 4.25 Yee 氏网格更新计算示意图

最后，在三维网格中 FDTD 数值计算的稳定条件是

$$c \Delta t \leq \left[\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2} \right]^{-1/2} \quad (4.55)$$

对于 $\Delta x = \Delta y = \Delta z = h$ 有

$$c\Delta t \leq h/\sqrt{3} \quad (4.56)$$

实际实现中，一般取 $c\Delta t \leq h/2$ ，即时域网格间隔 Δt 一般为电磁波传输一个空间网格 h 所用时间的一半。在三维空间中，也有一维中的 magic time step：当且仅当 $|k_x h| = |k_y h| = |k_z h|$ ，同时需满足 $c\Delta t = h/\sqrt{3}$ 。

4.4 本章总结

4.4.1 有限差分法总结

1. 微分用差分近似。对于一个间距为 h 的均匀网格而言， $x_{n+i} = x_n + ih$
2. 利用均匀网格可以分别计算微分以及差分。对于间隔两个格点的网格，

$$f'_i \approx \frac{f_{i+1} - f_{i-1}}{2h}$$

对于交错网格 (staggered grid)，

$$f'_{i+1/2} \approx \frac{f_{i+1} - f_i}{h}$$

而交错网格下二阶差分为，

$$f''_i \approx \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}$$

3. 拉普拉斯方程的差分近似

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \approx \frac{f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2}$$

4. FDM 方法具有数值色散现象。对于复指数函数 $f = \exp(jkx)$ ，交错格点的数值近似分别为

$$f'/f = \frac{2j}{h} \sin\left(\frac{kh}{2}\right)$$

$$f''/f = -\frac{4}{h^2} \sin^2\left(\frac{kh}{2}\right)$$

4.4.2 有限时域差分法总结

1. 对于一维的 Helmholtz 方程，FDTD 方法可得到显式的时域更新计算式

$$f_i^{(n+1)} = 2f_i^{(n)} - f_i^{(n-1)} + \left(\frac{\Delta t}{\Delta x}\right)^2 \left(f_{i+1}^{(n)} - 2f_i^{(n)} + f_{i-1}^{(n)}\right)$$

2. FDTD 方法中时域更新的步长 Δt 与数值计算的稳定性可通过分析

$$\rho^2 - [2 - \omega^2(\Delta t)^2]\rho + 1 = 0$$

对于 $|\rho| = 1$ 要求 $\Delta t \leq h$ 。

3. 根据 Faraday's law 和 Ampère's law, 可以分别通过在时间和空间维度上绘制交错网格进行数值计算。对于 z 方向传播的一维情形来说

$$\frac{\partial E_x}{\partial z} = -\mu \frac{\partial H_y}{\partial t}, \quad -\frac{\partial H_y}{\partial z} = \epsilon \frac{\partial E_x}{\partial y},$$

其中 $E_x = E_x(r, n)$ 及 $H_y = H_y(r + \frac{1}{2}, n + \frac{1}{2})$ 。与之对应的二阶波动方程 $\partial^2 E / \partial t^2 = c^2 \partial^2 E / \partial x^2$ 可由上式两个一阶方程得到。

4. 在三维 Yee 氏网格中：

$$\begin{aligned} &E_x|_{p+\frac{1}{2}, q, r}^n, \quad E_y|_{p, q+\frac{1}{2}, r}^n, \quad E_z|_{p, q, r+\frac{1}{2}}^n, \\ &H_x|_{p, q+\frac{1}{2}, r+\frac{1}{2}}^{n+\frac{1}{2}}, \quad H_y|_{p+\frac{1}{2}, q, r+\frac{1}{2}}^{n+\frac{1}{2}}, \quad H_z|_{p+\frac{1}{2}, q+\frac{1}{2}, r}^{n+\frac{1}{2}} \end{aligned}$$

电场置于与其方向相同的坐标轴的半格点上。磁场则位于其正交平面的中心位置。需理解记忆 Yee 元胞的结构和性质。

5. 稳定性条件 (Courant Condition) 为 $c\Delta t/h < 1/\sqrt{n}$, 其中 n 为计算的维度。对于三维情形, $c\Delta t < h/\sqrt{3}$, 一般取 $c\Delta t < h/2$ 。

4.5 习题

习题 4.1: 若数值计算时 k_{num} 需要低于 5% 的误差, 则网格间距 h 应该怎样设置? 试讨论两格点法和交错格点法下 D_{xx} 的近似性能。

习题 4.2: 介绍交错格点 (staggered grids) 方法, 绘制五格点差分构造。写出一维微分 $f'' = 0$ 和二维微分方程 $\nabla^2 f = 0$ 的交错格点差分构造格式。

习题 4.3: 简述基于复指数函数分析 FDM 方法数值色散的步骤和基本结论, 讨论数值方法 D_x 和 D_{xx} 与微分真值的关系。

习题 4.4: 电容也可用 $C = 2W/V^2$ 来计算, 其中 W 是电场能量, V 是电位差。将 W 写作 φ 的函数, 并讨论如何用有限差分计算 W 。

习题 4.5: 将二维 laplace 迭代式

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = \frac{f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2} = 0$$

写成矩阵的格式。考虑图 4.8 中边界条件和对称边界，将 $f_{i,j}$ 的迭代计算改写成矩阵形式。

习题 4.6: 我们通过对 f 进行泰勒展开，可得二阶导数的逼近形式为

$$f''(x) + O(h^2) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

进而得到二维情形下，离散 laplace 方程为

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = \frac{f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2} = 0$$

试推导三维情形下的离散拉普拉斯方程，并绘制格点示意图。

习题 4.7: 对于一维的 helmholtz 方程

$$\frac{\partial^2 f}{\partial t^2} = \frac{\partial^2 f}{\partial t^2}, \quad f(0, t) = f(1, t) = 0$$

可写为矩阵的形式

$$Af = kf$$

对于 5×1 的格点写出矩阵 A ，并解释差分矩阵 A 的特征值与 k 的关系。

习题 4.8: 试写出 1D Helmholtz 方程，给出数值计算中的时域更新计算式。并针对一维问题，给出 FDTD 方法时域更新计算的编程思路。

习题 4.9: 列举出 FDTD 方法的优点和缺点。

习题 4.10: 绘制三维的 Yee 氏网格。分别说明 E_x, E_y, E_z 和 H_x, H_y, H_z 的格点位置，并写出 H_x, H_y 和 H_z 的更新公式（注意： H 和 E 的空间坐标和时间坐标是否位于半格点上）。

4.6 上机实验

上机 4.1: (10 分) 在同轴传输线电容计算例题中, 我们令 $c = d = 3a = 3b$ 。

修改电容的计算程序, 用线性矩阵求解的形式 $A\mathbf{f} = \mathbf{b}$ 重新编写程序, 其中 \mathbf{f} 用来存储外导和内导之间格点上的电势值。利用二维可视化的方式展示计算得到的 \mathbf{f} , 同时利用数据外推计算 h_0 下的电容值。

上机 4.2: (10 分) 利用 Jacobi 和 Gauss–Seidel 迭代法计算边长为 a 的正方形区域中的电势 φ 。其中边界条件为 $\varphi(x, 0) = \varphi(0, y) = 0$ 和 $\varphi(x, a) = \varphi_0 \cdot (x/a), \varphi(a, y) = \varphi_0 \cdot (y/a)$, 其中 φ_0 为常量。分析各个算法的收敛率。在此基础上, 实现松弛迭代算法, 分析松弛因子 R 和收敛率的关系。该问题的解析解为 $\varphi(x, y) = \varphi_0 \cdot (xy/a^2)$ 。

上机 4.3: (20 分) 利用有限差分法计算同轴圆柱形传输线的电容值。内导半径为 a , 外导半径为 b , 设内导电位 $V = 1V$ 。本题电容的解析解为 $2\pi\epsilon_0/\ln(b/a)$, 可以利用对称性在轴上解算得到。通过计算机编写 FDM 程序, 可视化展示计算所得的 \mathbf{f} , 并分析计算误差与格点大小 h 的关系。另外, 分析在这种边界失配的情况下, 应该怎样计算边界上的差分, 并得到电容值? 能否通过数据外推得到 h_0 下的电容值?

提示: 在利用有限差分计算时, 需要处理边界不在格点上的情形。在这种情况下, 我们需要对差分运算进行一定的近似。例如图 4.26。其中 1, 2 点为不在格点上的边界值。而 3, 4 为内部格点。将中心点在

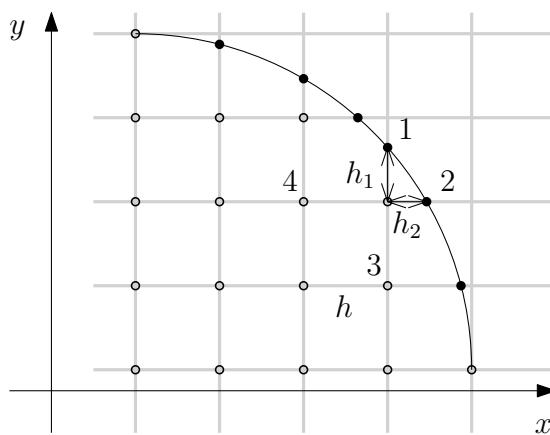


图 4.26 边界条件的差分离散化

y 轴方向上泰勒展开可得

$$\begin{aligned}f(y + h_1) &= f(y) + h_1 f'(y) + \frac{h_1^2}{2} f''(y) + \frac{h_1^3}{6} f'''(y) + \dots \\f(y - h) &= f(y) - h f'(y) + \frac{h^2}{2} f''(y) - \frac{h^3}{6} f'''(y) + \dots\end{aligned}$$

两式分别乘以 h, h_1 。相加并忽略 f''' 高次项，可得

$$\frac{1}{2} f''(y) = \frac{f(y + h_1)}{h(h + h_1)} + \frac{f(y - h)}{h_1(h + h_1)} - \frac{f(x, y)}{hh_1} \quad (4.57)$$

同理可求得 $f''(x)$

$$\frac{1}{2} f''(x) = \frac{f(x + h_2)}{h(h + h_2)} + \frac{f(x - h)}{h_2(h + h_2)} - \frac{f(x, y)}{hh_2} \quad (4.58)$$

解 laplace 等式 $\nabla^2 f = 0$ 即可得迭代求解的等式。

第5章 加权余量法与伽辽金有限元

刘本源 *bbyliu@fmmu.edu.cn*

学时 上机

2 0

5.1 目的

能理解加权余量法的概念和思路，能够写出加权余量有限元方程组。

重点：加权余量法

难点：伽辽金有限元法

5.1.1 回顾

上一节课我们讲了有限差分方法 (FDM)。有限差分法的核心思想有两个：其一是用差分近似微分；其二是交错格点 (Staggered Grids, 也被称作半格点)。

$$f''_x \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

时域有限差分的原理与之类似，即对时间微计算差分；并可得到显式的时域更新公式，计算从时刻 $n-1$ 、 n 到下一时刻 $n+1$ 的电磁场数值解

$$f''_t = L[f] \quad \rightarrow \quad \frac{f(t-\Delta t) - 2f(t) + f(t+\Delta t)}{(\Delta t)^2} = L[f]$$

差分法实施的注意事项有两个：其一差分网格间距 h 与电磁场频率的约束关系；其二时域网格间距 Δt 和空域网格间距 h 的 CFL 条件 $c\Delta t \leq h/\sqrt{3}$ ，实际中一般取 $c\Delta t \leq h/2$ 。

5.1.2 授课框架

本章将利用加权余量法，推导并学习一种简单的、易于理解的有限元方法。本章的授课核心为加权余量 (Methods of Weighted Residuals, MWR) 的思想，利用分部积分和高斯积分转化，得到微分方程的弱形式，并在此基础上推导伽辽金 (Galerkin) 有限元。通过详尽的一维及二维有限元计算示例，给学生们讲

解清楚 Galerkin 有限元的实施步骤，为接下来的上机实验储备理论知识和应用技能。

授课方法：简洁、用图例直观数学思想、循序渐进；授课时需以板书为主，适时总结回顾，讲解计算示例。

5.2 加权余量法

5.2.1 介绍

在有限差分法中，我们讲解了如何利用差分近似微分，求解规则场域中的电磁场分布。在一维情形下，一个重要的二阶偏微分方程即亥姆霍兹（Helmholtz）方程：

$$\frac{\partial^2 f}{\partial x^2} = -k^2 f$$

在本章，我们同样通过研究这一常见的二阶微分方程，其通用形式为

$$-\frac{d}{dx} \left(\alpha \frac{df}{dx} \right) + \beta f = \rho, \quad a < x < b \quad (5.1)$$

$$f(a) = f_a, \quad (5.2)$$

$$f(b) = f_b \quad (5.3)$$

其中 $f = f(x)$ 是待求函数，媒介特性（例如电导率，介电常数等）分别由 $\alpha = \alpha(x)$ 和 $\beta = \beta(x)$ 给出，（激励）源特性由 $\rho = \rho(x)$ 给出。在实际应用中 α 、 β 、 ρ 可为固定值或为 x 的函数。

为了方便起见，令式(5.1)中 $\beta = 0$ ，则 Helmholtz 方程可写为

$$-\frac{d}{dx} \left(\alpha \frac{df}{dx} \right) = \rho, \quad a < x < b \quad (5.4)$$

上式也称作一维泊松（Poisson's Equation）方程。

在一维场景下，有限元进行数值求解的思想即为函数的分段逼近，见图5.1。在图5.1中，我们将定义域 $\Omega = (a, b)$ 所在的区间¹ 分为有限个（finite number）互不重叠的元（element），每个元的端点称作节点（node），节点的坐标用 x_i 表示。例如，对于第 e 个有限元，其节点为 x_e 和 x_{e+1} ，而 Ω_e 则代表第 e 个有限元的定义域 $[x_e, x_{e+1}]$ 。有限元的个数用 n_e 表示，在一维情形下，节点数 n

¹注意这里用到了一些数学符号，例如 Ω 、 $\partial\Omega$ 等，可能学生会看着头大。这里着重讲解其思想，并给学生解释，在科学的研究中，合理的使用数学符号清晰易懂、易于加深理解。

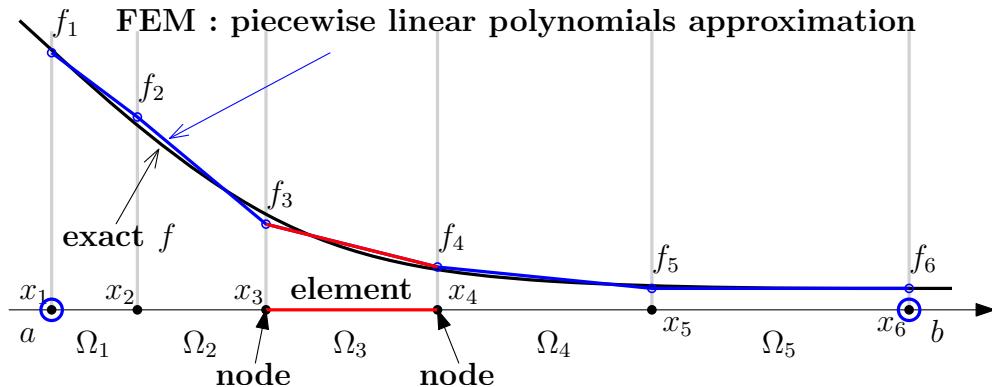


图 5.1 一维有限元：利用分段连续 C^0 函数逼近偏微分方程的解。

与有限元数 n_e 的关系是 $n = n_e + 1$ 。为了方便处理边界条件，我们定义 Γ 表示区间 Ω 的边界。在一维问题中，边界 Γ 仅包含 a 和 b 两个点；而积分区间 Ω 是开区间，不包含边界 Γ 上的点。

对比思考：回忆利用差分方法求解 Helmholtz 方程的相关知识。利用交错网格构建差分格式，可以方便的计算一维微分方程。同时由于一维的微分问题是规则的，因而差分法也可以方便的处理边界条件。与有限差分法不同，有限元的思想为：求解一维偏微分方程时，我们通过求函数 f 在节点 i 上的取值 f_i ，利用分段（在每个元上）逼近的方式，得到方程(5.4)的近似解。

5.2.2 弱形式

式(5.4)在有限元中常被称作强形式 (strong form)，而 FEM 方法（或伽辽金有限元方法）需要用到偏微分方程的弱形式 (weak form)。

推导弱形式需要用到加权余量法 (Methods of Weighted Residuals, MWR)。该方法起源于 30 年前，相比于变分法，MWR 易于理解且所需数学知识较少。在国外已经被越来越多的教材采用，用作 FE 的教学和科学的研究。

加权余量法的基本思想可见图 5.2。首先，在偏微分方程(5.4)中，定义余量（或称残差） r 为

$$r = -\frac{d}{dx} \alpha \frac{df}{dx}$$

其中 $r = r(x)$ ，即余量 r 为 x 的函数。

在后文中，不失一般性，我们用 f 代表数值解。那么，如果我们得到精确的数值解 f ，则余量 $r = 0$ 。但是大多数情况下，余量 $r(x)$ 不可能处处为零

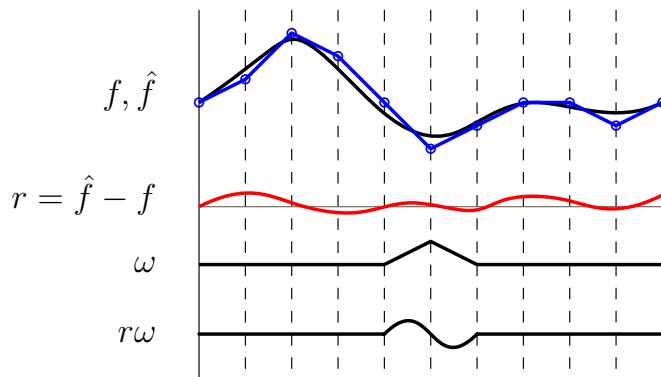


图 5.2 加权余量法示意图。 f 为原函数， \hat{f} 为 FE 数值解，余量 r 定义为 $\hat{f} - f$ 。我们选取一权函数 ω ，使得余量在加权积分上为 0，即 $\int r\omega = 0$ 。

(即 $r(x) = 0, a < x < b$)。

要点：这时，我们不妨换一种思路，令残差 r 在区域 $[a, b]$ 上的加权积分为零。也就是说寻找与残差正交的权函数 ω ，使得残差在加权平均（积分）的意义上为零。这一方法即为加权余量法。

首先，任意选定一个权函数 ω^1 ，将余量 $r(x)$ 乘以权函数 ω ，在区间 $[a, b]$ 上计算积分并令加权余量为 0

$$\int_a^b r(x)\omega dx = \int_a^b \left[-\frac{d}{dx}\alpha \frac{df}{dx} - \rho \right] \omega dx = 0 \quad (5.5)$$

式(5.5)定义为偏微分方程的加权余量。

要点：对于 n 个未知节点，如果我们能够碰巧找到 n 个权函数 ω_i ，那么便可以构成 n 个方程组，从中解出 $f = [f_1, \dots, f_n]$ 。

偏微分方程

$$-\frac{d}{dx}\alpha \frac{df}{dx} = \rho$$

常被称作强形式（Strong Form），因为该式需要 f 的二阶微分 $f''(x)$ 存在并且分段连续。在加权余量中，引进了积分和权函数的巧妙之处在于，我们可以分别用分部积分和高斯积分降低函数的阶数、转化积分区间。

仔细观察式(5.5)，首先利用分部积分公式

$$(u'w)' = u''w + u'w'$$

¹ 权函数也被称为检验函数（Testing function）。其含义为不断测试不同的权函数，选择一个恰当的权函数使其与余量正交。这一过程同样蕴含着泛函或能量最小化的思想。

调换次序 $-u''w = u'w' - (u'w)'$, 加权余量式(5.5)可写为

$$\int_a^b \left[-\frac{d}{dx} \alpha \frac{df}{dx} - \rho \right] \omega dx = 0, \quad (5.6)$$

$$\int_a^b \left[\alpha \frac{df}{dx} \frac{d\omega}{dx} - \frac{d}{dx} \left(\alpha \omega \frac{df}{dx} \right) - \rho \right] \omega dx = 0, \quad (5.7)$$

更进一步, 对于 $(u'w)'$, 利用高斯积分公式化简: 在一维中, 一阶导数的线积分可转化为端点的边界积分,

$$\int_a^b \frac{d}{dx} \left(\alpha \omega \frac{df}{dx} \right) dx = \int_{\Gamma} \alpha \omega \frac{df}{dx} \hat{n}_x d\Gamma$$

则

$$\int_a^b \left(\alpha \frac{df}{dx} \frac{d\omega}{dx} - \omega \rho \right) dx = \int_{\Gamma} \alpha \omega \frac{df}{dx} \hat{n}_x d\Gamma \quad (5.8)$$

其中式(5.8)的右边被称为边界积分, \hat{n}_x 为 x 轴由区间内向外的单位法向向量。在一维情形下, Γ 上的左端点 a 点 $\hat{n}_x = -1$, 右端点 b 点 $\hat{n}_x = 1$ 。

式(5.8)即被称作 FEM 的弱形式 (**weak form**)。相比于 FEM 的强形式(5.4), 弱形式(5.8)降低了待求函数 f 的微分阶数, 只包含 f 和 ω 的一阶导数, 这样我们就可以从分段连续的函数空间¹ 中寻找 f 和 w 。而在强形式(5.4)中, 要求 f 必须是二阶连续可导, 即 $f \in C^2(\Omega)$ 。在 FEM 中, 数值解 f 也被称作 trial function, 而权函数 ω 也被称作 test function。

要点: FEM 的思路为, 寻找分段连续的数值解 f , 选择权函数 ω , 使得数值计算的结果满足弱形式(5.8)。

下面, 在讲怎样计算 (或者简而言之: 构造) f 和选择 ω 前, 我们先讲一下弱形式中是怎样方便的处理边界条件。

5.2.3 边界条件

FEM 弱形式(5.8)的右边称为边界条件,

$$\int_{\Gamma} \alpha \omega \frac{df}{dx} \hat{n}_x d\Gamma \quad (5.9)$$

¹分段连续的函数属于 $C^0(\Omega)$ 空间。按照定义, 一个函数 u 位于 $C^k(\Omega)$ 空间表明函数 u 的 0 到 k 阶导数都是连续的。

对于第一类边界条件 (Dirichlet), 边界点上的函数值已知

$$f(a) = f_a, \quad f(b) = f_b$$

这样, 在式(5.8)中, 待求节点 f_i 中就可以消去 $f(a)$ 和 $f(b)$ 两个节点, 进而减小方程组的数目 (从 n 减少到 $n - 2$)。在边界点 a, b 上, 由于函数值已知且为常数, 为了使权函数 ω 与余量正交, 就要求检验函数 ω 必须为 $\omega(a) = 0$ 和 $\omega(b) = 0$ 。另外, 我们也可以从变分的角度进行理解。检验函数 ω 近似为 f 的一阶变分 (函数的导数, $\omega \sim \delta f$)。这样, 如果 f 在点 i 上的值已知 (为常数), 则第 i 点的一阶变分就为 0。其物理含义为: 当节点上的函数值已知时, 其不确定性为 0。

对于第二类边界条件 (Neumann), 边界点 a, b 上函数值的导数已知

$$\alpha \frac{df}{dx} \Big|_{x=a} = q_a, \quad \alpha \frac{df}{dx} \Big|_{x=b} = q_b,$$

这时, 我们只要将 q_a 和 q_b 替换式(5.8)中, 边界条件所在位置的元素即可。这里需要强调一下边界条件中的单位通量: \hat{n}_x 。在先前推导中, 我们利用高斯积分公式得到了边界条件。在一维情形中, x 轴方向上 a 点的单位通量 $\hat{n}_x = -1$, b 点的单位通量为 $\hat{n}_x = 1$ 。对于二维情形, 可参见图5.3。

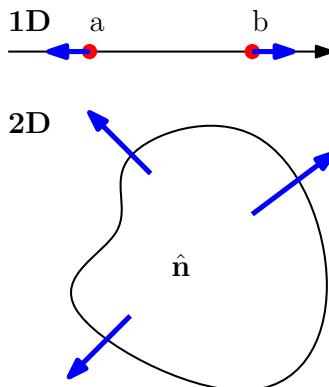


图 5.3 一维及二维单位通量

若已知 a 点和 b 点的导数值分别为 q_a 和 q_b , 则

$$\int_{\Gamma} \alpha \omega \frac{df}{dx} \hat{n}_x d\Gamma = q_b - (-q_a) = q_b + q_a$$

在用式(5.8)处理第二类边界条件时, 不要忘记了媒介特性 $\alpha(x)$ 和 \hat{n}_x 。

利用弱形式(5.8)的优点是: 我们不但可以降低待求函数 f 的阶数, 同时也

可以方便的集成边界条件。对于一维情形，边界 Γ 只包含两个端点，Neumann 边界条件不需要计算式(5.9)的积分，只需求和即可。对于二维情形，边界 Γ 是区域的围线，在三维下则是区域的边界面。弱形式的得出，是 FEM 法相对于先前学习的有限差分 FDM、时域有限差分 FDTD 的明显优势。

5.2.4 基函数

在得到弱形式之后，还不能直接用于数值计算，因为还缺少了最重要的一环：有限元。在一维情形下，就是用有限多个线段上的连续函数逼近数值解。我们不妨这样想，利用有限多个函数逼近原函数，反过来就是将原函数在有限多个（基）函数上展开。

在傅立叶变换中，我们就已经学习了函数 f 在傅立叶基（正弦函数作为基函数）上展开的方法。有限元的思想与之类似，将函数 f 在元的基函数（Nodal basis）上展开，从数学上描述，就是将函数 f 表示成基函数的线性组合形式

$$f(x) = \sum_{j=1}^n f_j \varphi_j(x) \quad (5.10)$$

那么有限元中的基函数是什么样子的呢？一种常用的线性基函数为帐篷函数（hat function）或草帽函数。在每个节点上，定义节点的基函数（node basis function）为 $\varphi_i(x)$ ，其函数形状如图5.4所示。

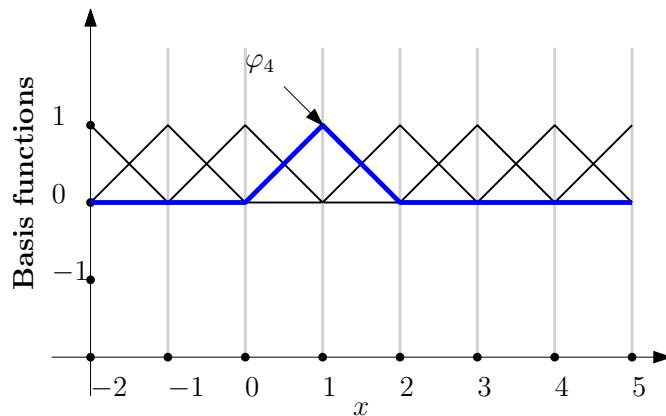


图 5.4 一维线性有限元。基函数 $\varphi_4(x)$ 被着重标出。

该函数在每一个区间上都是线性的、局部化的：只在节点 i 处取值为 1，在其它节点为 0。节点的基函数也被称作“帐篷函数（tent function）”。由于基函数 φ_i 的局部化特性（由于 $\varphi_j(x)$ 仅在 x_j 上为 1，在其他节点 $x_i, i \neq j$ 上为 0。

φ_i 类似于 Kronecker delta δ_j 函数在连续区间的数值近似)，我们有 $f(x_i) = f_i$ 。即 $f(x)$ 在基函数上展开的系数 f_j 就是 f 在该节点上的取值。

n 个未知节点包含 n 个待求解变量 f_i ，为了能够求解线性方程组，我们需要选择 n 个权（检验）函数 ω_i 。一般而言权函数的选择是很随意的，只要满足加权余量公式即可现有选函数的设计准则包括：

- **点匹配法：** ω_i 选取为区间中点的函数值， $\omega_i = \delta(x - x_{i+h_e/2})$
- **最小二乘法：** ω_i 选取为余量 r
- **伽辽金法：** ω_i 选取为基函数 φ_i

接下来，我们重点介绍伽辽金（Galerkin）有限元。

Galerkin 有限元法设定检验函数 ω 与基函数 φ 相同，即

$$\omega_i = \varphi_i(x) \quad (5.11)$$

将基函数展开式(5.10)与伽辽金有限元式(5.11)代入式(5.8)中，可得

$$\int_a^b \left[\alpha \varphi'_i \sum_j^n f_j \varphi'_j - \varphi_i \rho \right] dx = \int_{\Gamma} \alpha \varphi_i \frac{df}{dx} \hat{n}_x d\Gamma$$

交换积分和求和的顺序，我们可得

$$\sum_j^n f_j \int_a^b \alpha \varphi'_i \varphi'_j dx = \int_a^b (\varphi_i \rho) dx + \int_{\Gamma} \alpha \varphi_i \frac{df}{dx} \hat{n}_x d\Gamma \quad (5.12)$$

在式(5.12)可写为方程组的形式：每选择一个检验函数 $\omega_i = \varphi_i$ ，就可以生成一个方程式 $\mathbf{K}_i^T \mathbf{f} = s_i + B_i$ ，其中 K_{ij} , s_i 和 B_i 分别为

$$K_{ij} = \int_a^b \alpha \varphi'_i \varphi'_j dx \quad (5.13)$$

$$B_i = \int_{\Gamma} \alpha \varphi_i \frac{df}{dx} \hat{n}_x d\Gamma \quad (5.14)$$

$$s_i = \int_a^b (\varphi_i \rho) dx \quad (5.15)$$

为了帮助学生们记忆，需掌握 $\sum a_i b_i = \mathbf{a}^T \mathbf{b}$ 。

通过选择 n 个权函数 ω_i 就可构成 n 个方程组，进而可将式(5.12)拼接起来，写成矩阵的形式

$$\mathbf{K}\mathbf{f} = \mathbf{s} + \mathbf{B} \quad (5.16)$$

其中 \mathbf{K} 被称作系数矩阵（也被称作 Stiffness Matrix¹）， \mathbf{f} 为待求节点上的势函数值（也被称作 Load Vector）， \mathbf{s} 是源向量， \mathbf{B} 是边界向量。

5.2.5 系数矩阵的计算：以元为中心的视角

在 FEM 方法中，我们可以分别计算在每个元 Ω_e 上的系数矩阵，然后求和得到式(5.13)的积分

$$K_{ij} = \int_a^b \alpha \varphi_i' \varphi_j' dx = \sum_e^{n_e} K^e$$

K^e 定义为

$$K^e = \int_{x_e}^{x_{e+1}} \alpha \varphi_i' \varphi_j' dx$$

其中 x_e 和 x_{e+1} 为第 e 个有限元的端点。

由于基函数 φ 的局部化特性， \mathbf{K} 矩阵具备

1. 稀疏性， K_{ij} 只在邻近 $|i - j| \leq 1$ 的元上非零，其它位置上的积分为 0
2. 对称性，即 $K_{ij} = K_{ji}$

5.2.5.1 主元

更进一步，我们发现对于非零的 K^e ，其积分形式仅仅包含三种样式，见图5.5。从每一个面元的角度看去，当采用线性基函数（帐篷函数）时，一维伽

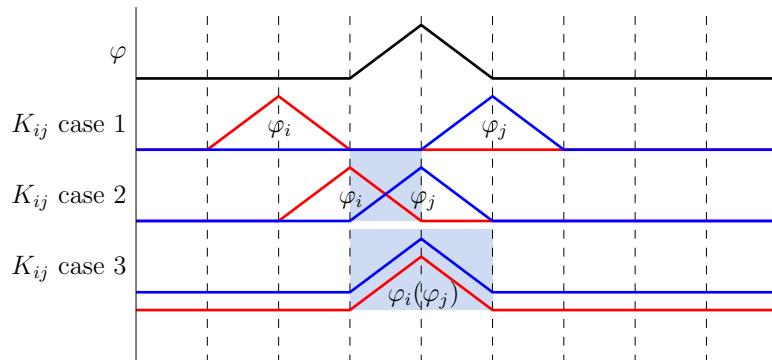


图 5.5 系数矩阵积分的典型样式

¹Stiffness Matrix 这一名称最早来源于力学中的有限元分析，但是在流体以及热学分析中，以至在电磁场数值计算中，都会用这一称呼。Stiffness Matrix 反映 Ω 在外界“力”作用下的内部抗 (Resistance) 力。

辽金有限元的积分仅仅包含三种类型，见图5.6。因而，我们可以将每个面元上

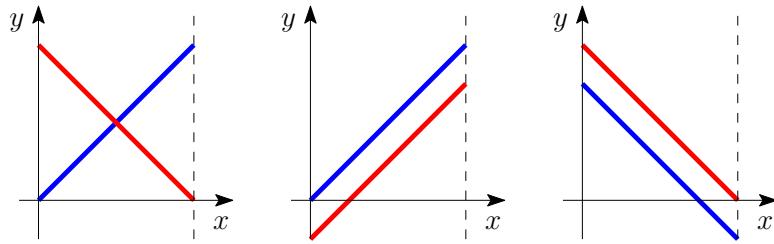


图 5.6 每个面元上的积分样式

的积分转化为主元积分的形式。定义一维有限元中的主元为 S_1 和 S_2 两个函数，其定义域为 $\Omega_0 = (-1, 1)$ ，见图5.7。每一个 K^e 都可转化为两个主元组合的形

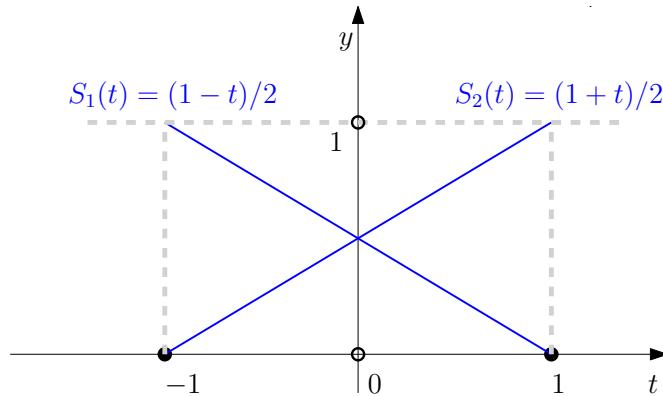


图 5.7 一维有限元中的主元

式，如 $\int \alpha S'_1 S'_1$, $\int \alpha S'_1 S'_2$ 和 $\int \alpha S'_2 S'_2$ 。通过积分限变换（平移、缩放），令

$$x = \frac{h^e}{2}t + \frac{x_e + x_{e+1}}{2} \quad (5.17)$$

其中 $h^e = x_{e+1} - x_e$ 为第 e 个有限元的长度。

我们可以将 K^e 的积分，表达成主元上积分。定义 Jacobi 系数 J_e 为

$$J_e = \frac{dx}{dt} = \frac{h^e}{2}$$

则

$$K^e = \int_{x_e}^{x_{e+1}} \left(\alpha \frac{d\varphi_i}{dt} \frac{dt}{dx} \cdot \frac{d\varphi_j}{dt} \frac{dt}{dx} \right) \frac{dx}{dt} dt \quad (5.18)$$

$$= \int_{-1}^1 \left(\alpha \frac{d\varphi_i}{dt} \frac{1}{J_e} \cdot \frac{d\varphi_j}{dt} \frac{1}{J_e} \right) J_e dt \quad (5.19)$$

由于主元函数只有两种类型 S_1 和 S_2 ，我们可将 K^e 的计算式写作

$$K_{IJ}^e = \int_{-1}^1 \left(\alpha \frac{dS_I}{dt} \frac{1}{J_e} \cdot \frac{dS_J}{dt} \frac{1}{J_e} \right) J_e dt, \quad I, J \in \{1, 2\} \quad (5.20)$$

式(5.20)可以通过数值积分法（高斯积分或中值法）计算得到。

而对于 s^e ，我们同样有

$$s_I^e = \int_{-1}^1 \rho S_I J_e dt, \quad I \in \{1, 2\} \quad (5.21)$$

在一维情形下，边界 Γ 仅包含两个端点，因此边界条件 \mathbf{B} 可以直接得到。为了方便后面扩展到二维以及三维，我们将 B_I^e 写为

$$B_I^e = \int_{\Gamma \cap \Gamma_e} \alpha \varphi_i \frac{df}{dx} \hat{n}_x d\Gamma_e \quad (5.22)$$

其中 Γ_e 为第 e 个元的边界。式(5.22)只有在面元 e 的端点包含边界点（二维为线，三维为面）时才不为零，在内部元上都为 0。

需要注意的是，若 α 、 β 、 ρ 为 x 的函数，我们需要通过式(5.17)对 α 、 β 、 ρ 进行积分限变换后，才能在主元下计算 \mathbf{K} 、 \mathbf{s} 和 \mathbf{B} 。

主元积分计算系数矩阵的思想有两个好处：其一可以适用于计算高阶基函数，其二具备模块化的优势，易于编程实现。但是当我们仅仅采用线性基函数时（例如本章所用的帐篷函数），采用解析积分可以很方便的计算 K^e 。这一点将在后面二维 FEM 中详细讲解。

5.2.6 系数矩阵的组装方法

系数矩阵计算的典型思路为：局部计算，全局组装。理解了组装（Assemble）的思想后，FEM 的程序框架就会逐步显现出来。下面，以图5.8为例。

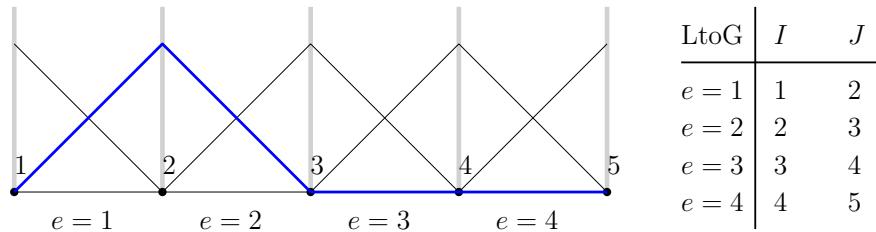


图 5.8 LtoG 映射矩阵示例

图5.8包含 4 个元和 5 个节点。对于每一个元 e ，我们首先计算（局部的）

\mathbf{K}^e 和 \mathbf{s}^e

$$\mathbf{K}^e = \begin{bmatrix} K_{11}^e & K_{12}^e \\ K_{21}^e & K_{22}^e \end{bmatrix}, \quad \mathbf{s}^e = \begin{bmatrix} s_1^e \\ s_2^e \end{bmatrix} \quad (5.23)$$

随后，我们要组装成全局的 \mathbf{K} 和 \mathbf{s} 。而组装的过程，需要用到一个被称作 Local to Global 的连接矩阵。图 5.8 中，LtoG 为，

$$\text{LtoG} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \end{bmatrix} \quad (5.24)$$

其中 LtoG 矩阵的每一行对应第 e 个有限元，里面的值为有限元 e 的全局节点编号，以小写字母 i, j 编号；局部节点以 I, J 大写字母编号。例如 $\text{LtoG}(e=2, I=2) = 3$ 。

在通过式(5.20)计算得到 K_{IJ}^e 后，我们要将他放在 K_{ij} 位置上。对于一维情形，一个直观的例子可见图 5.9。也就是说，有限元的连接矩阵描述了面元系

$$K_{33} = \int \alpha \varphi'_3 \varphi'_3 = \int \alpha S'_2 S'_2 + \int \alpha S'_1 S'_1 = K_{22}^{e=2} + K_{11}^{e=3}$$

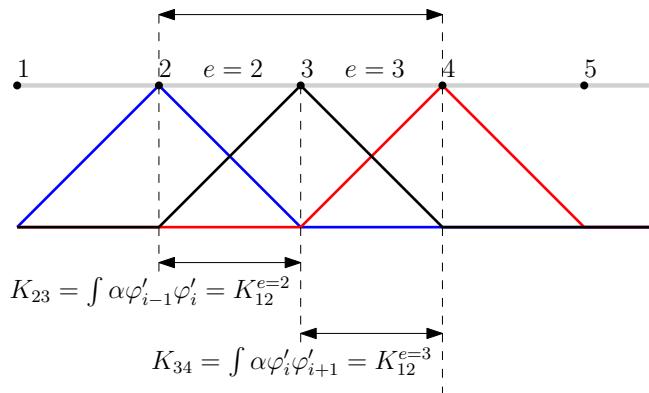


图 5.9 利用连接矩阵组装系数矩阵示意图

数矩阵 \mathbf{K}^e 对于全局系数矩阵 \mathbf{K} 的贡献方法。以面元 $e = 2$ 为例，其系数矩阵 \mathbf{K}^2 的四个元素，需要分别贡献到（用编程语言描述就是累加）全局系数矩阵 \mathbf{K} 的 $(2, 2)$, $(2, 3)$, $(3, 2)$ 和 $(3, 3)$ 位置处。这一过程在有限元中被称作组装 (Assemble)，可以方便的用 LtoG 矩阵实现。

系数矩阵的组装过程用数学形式表示即

$$K_{ij} = K_{ij} + K_{IJ}^e, \quad i = \text{LtoG}(e, I), \quad j = \text{LtoG}(e, J) \quad (5.25)$$

同理对于 s_I^e , 我们需要置于 s_i 上

$$s_i = s_i + s_I^e, \quad i = \text{LtoG}(e, I) \quad (5.26)$$

而所有这一组装过程都是局部计算 \mathbf{K}^e , 全局组装 (利用 LtoG 矩阵)。

我们再回到图 5.8 的例子, 利用 LtoG 矩阵, 可写出方程 $\mathbf{K}\mathbf{f} = \mathbf{s} + \mathbf{B}$ 为

$$\begin{bmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 \\ K_{21}^1 & K_{22}^1 + K_{11}^2 & K_{12}^2 & 0 & 0 \\ 0 & K_{21}^2 & K_{22}^2 + K_{11}^3 & K_{12}^3 & 0 \\ 0 & 0 & K_{21}^3 & K_{22}^3 + K_{11}^4 & K_{12}^4 \\ 0 & 0 & 0 & K_{21}^4 & K_{22}^4 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} = \begin{bmatrix} s_1^1 \\ s_2^1 + s_1^2 \\ s_2^2 + s_1^3 \\ s_2^3 + s_1^4 \\ s_2^4 \end{bmatrix} + \begin{bmatrix} B_1^1 \\ 0 \\ 0 \\ 0 \\ B_2^4 \end{bmatrix}$$

对于边界节点 (内部面元上 $\Gamma \cap \partial\Gamma = \emptyset$), 我们有

$$\mathbf{B} = \begin{bmatrix} B_1^1 \\ B_2^1 + B_1^2 \\ B_2^2 + B_1^3 \\ B_2^3 + B_1^4 \\ B_2^4 \end{bmatrix} = \begin{bmatrix} B_1^1 \\ 0 \\ 0 \\ 0 \\ B_2^4 \end{bmatrix} \quad (5.27)$$

这里再次强调一点: 一维 FEM 的计算过程分为两步, 首先根据面元 e 的节点坐标 x_e, x_{e+1} 计算 \mathbf{K}^e ; 随后利用 LtoG 矩阵进行组装得到 \mathbf{K} 。也就是说, 全部的计算过程需要两个输入参数: 节点坐标矩阵和节点连接矩阵。这一概念将在二维中非结构化网格部分进行加强。

5.2.7 计算边界条件

对于第一类边界条件, 假定我们已知左边端点的函数值 f_1 , 则方程组可消去 f_1 所在的行和列, 将矩阵 $\mathbf{K}\mathbf{f} = \mathbf{s} + \mathbf{B}$ 写为

$$\begin{bmatrix} K_{22}^1 + K_{11}^2 & K_{12}^2 & 0 & 0 \\ K_{21}^2 & K_{22}^2 + K_{11}^3 & K_{12}^3 & 0 \\ 0 & K_{21}^3 & K_{22}^3 + K_{11}^4 & K_{12}^4 \\ 0 & 0 & K_{21}^4 & K_{22}^4 \end{bmatrix} \begin{bmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} = \begin{bmatrix} s_2 - f_1 K_{21} \\ s_3 - f_1 K_{31} \\ s_4 - f_1 K_{41} \\ s_5 - f_1 K_{51} \end{bmatrix}$$

对于第二类边界条件，假定我们右边端点的 f'_n 已知，则我们可通过式(5.22)计算 B_2^4 的值，将矩阵 $\mathbf{K}\mathbf{f} = \mathbf{s} + \mathbf{B}$ 写为

$$\begin{bmatrix} K_{22}^1 + K_{11}^2 & K_{12}^2 & 0 & 0 \\ K_{21}^2 & K_{22}^2 + K_{11}^3 & K_{12}^3 & 0 \\ 0 & K_{21}^3 & K_{22}^3 + K_{11}^4 & K_{12}^4 \\ 0 & 0 & K_{21}^4 & K_{22}^4 \end{bmatrix} \begin{bmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} = \begin{bmatrix} s_2 \\ s_3 \\ s_4 \\ s_5 + B_2^4 \end{bmatrix}$$

通过求解线性方程组即可得到节点上的函数值 f_i ，然后利用

$$f = \sum_i^n f_i \varphi_i$$

得到有限元的数值解。

5.2.8 一维有限元编程步骤

经过前面的分析，我们已经大致的得到了有限元计算的思想。有限元本质是构建系数矩阵 \mathbf{K} 、源向量 \mathbf{s} 和边界向量 \mathbf{B} ，通过解方程得到 \mathbf{f} 。

有限元编程的步骤（以一维有限元为例）为：

1. 读取网格表（mesh）的节点坐标 x_i ，建立节点连接矩阵 LtoG。对于一维情形，连接矩阵 LtoG 是一个 $n_e \times 2$ 的矩阵
2. 选择基函数 $S_I, I = 1, 2$ 。当使用一维线性基函数近似时，主元上的基函数为： $t \in \Omega_0 = [-1, 1]$, $S_1 = (1-t)/2$, $S_2 = (1+t)/2$ 。计算 $S'_1 = -1/2$ 和 $S'_2 = 1/2$ （常数）
3. 建立映射，将 Ω_e 映射到 Ω_0 上，即 $x = \frac{h^e}{2}t + \frac{x_e + x_{e+1}}{2}$ ，其中 h^e 为元 e 的长度。计算 Jacobi 矩阵，在一维情形下有 $J_e = dx/dt = h^e/2$
4. 计算主元上的积分（可利用高斯数值积分法或中值法）

$$K_{IJ}^e = \int_{\Omega_0} \alpha(J_e t + \frac{x_e + x_{e+1}}{2}) \frac{S'_I}{J_e} \frac{S'_J}{J_e} J_e dt, \quad I, J \in \{1, 2\}$$

$$s_I^e = \int_{\Omega_0} \rho(J_e t + \frac{x_e + x_{e+1}}{2}) S_I J_e dt, \quad I \in \{1, 2\}$$

5. 利用链接矩阵 LtoG 组装 $\mathbf{K} = \sum_e \mathbf{K}^e$, $\mathbf{s} = \sum_e \mathbf{s}^e$ 。其中 \sum_e 是组装算子，不是求和。组装过程为

$$\begin{cases} K_{ij} = K_{ij} + K_{IJ} \\ s_i = s_i + s_I \end{cases} \quad \begin{cases} i = \text{LtoG}(e, I) \\ j = \text{LtoG}(e, J) \end{cases}$$

6. 计算边界条件 B_I^e

$$B_I^e = \int_{\Gamma \cap \Gamma_e} \alpha(x) \varphi_i \frac{df}{dx} \hat{n}_x d\Gamma_e$$

7. 依照第一类或第二类边界条件修改矩阵 K , s 和 B

8. 求解线性方程组得 $f = \{f_1, \dots, f_n\}$, 得到数值解 $f = \sum_i^n f_i \varphi_i$

注意: 当采用线性基函数时, 利用解析计算积分可能效率更高、更易编程。但是主元法模块性强, 且可方便的扩展到高阶基函数和高维情形。

5.2.9 计算示例

例题 5.1: 利用有限元方法计算

$$-\frac{d^2 f}{dx^2} = 1, \quad f(0) = 0, f(1) = 0$$

提示: 函数的解析解为 $f = -x^2/2 + x/2$ 。

首先在区间 $\Omega = (0, 1)$ 上划分有限元, 本例中生成 $n_e = 5$ 个有限元, 包含 $n = n_e + 1$ 个节点; 随后建立节点坐标 x_i :

$$x_i = [0, 0.2, \dots, 0.8, 1]$$

以及连接矩阵

$$\text{LtoG} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \end{bmatrix}$$

本例中 K_{IJ}^e 和 s_I^e 可以直接按照

$$K_{IJ}^e = \int_{\Omega_0} S'_I S'_J \frac{1}{J_e} dt$$

$$s_I^e = \int_{\Omega_0} S_I J_e dt$$

计算。由于主元的基 $S_1 = (1-t)/2$ 和 $S_2 = (1+t)/2$ 是线性的, 我们可以直接中值法计算, 其中 $J_e = h^e/2 = 0.1$, 积分区间为 $[-1, 1]$ 。

由于所有元的大小是相等的, 且媒介和源的特性 $(\alpha(x), \beta(x), \rho(x))$ 为常

量，不随 x 变化。因而我们可以只计算出一个主元上的 \mathbf{K}^e 和 \mathbf{s}^e

$$\mathbf{K}^e = \begin{pmatrix} 5 & -5 \\ -5 & 5 \end{pmatrix} \quad \mathbf{s}^e = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$$

然后通过组装 \sum_e 可得，得到 \mathbf{K} 和 \mathbf{s}

$$\mathbf{K} = \begin{pmatrix} 5 & -5 & 0 & 0 & 0 & 0 \\ -5 & 5+5 & -5 & 0 & 0 & 0 \\ 0 & -5 & 5+5 & -5 & 0 & 0 \\ 0 & 0 & -5 & 5+5 & -5 & 0 \\ 0 & 0 & 0 & -5 & 5+5 & -5 \\ 0 & 0 & 0 & 0 & -5 & 5 \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.1 \end{pmatrix}$$

下一步就是处理边界条件。由于 $f(0)$ 和 $f(1)$ 是已知的，则

$$\mathbf{s} = \mathbf{s} - f(0)\mathbf{K}_{.1} - f(1)\mathbf{K}_{.6}$$

其中 $\mathbf{K}_{.1}$ 和 $\mathbf{K}_{.6}$ 分别是 \mathbf{K} 的第 1 列和第 6 列，随后解得 $\{f_2, f_3, f_4, f_5\}$ 。

$$\begin{pmatrix} 5+5 & -5 & 0 & 0 \\ -5 & 5+5 & -5 & 0 \\ 0 & -5 & 5+5 & -5 \\ 0 & 0 & -5 & 5+5 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}$$

可得 $\mathbf{f} = [0.08, 0.12, 0.12, 0.08]$ 。解析解和有限元法的对比见图 5.10。

例题 5.2: 利用有限元方法计算

$$-\frac{d^2 f}{dx^2} = 1, \quad f(0) = 0, f'(1) = -1$$

提示：函数的解析解为 $f = -x^2/2$ 。

本例题的解题思路与上一个例题一致，所不同的是在 $x = 1$ 点边界上变成了 Neumann 边界条件。我们可以直接修改 \mathbf{s} 为

$$s_6 = s_6 + f'(1)$$

即可。本例题的数值解和解析解见图 5.11。

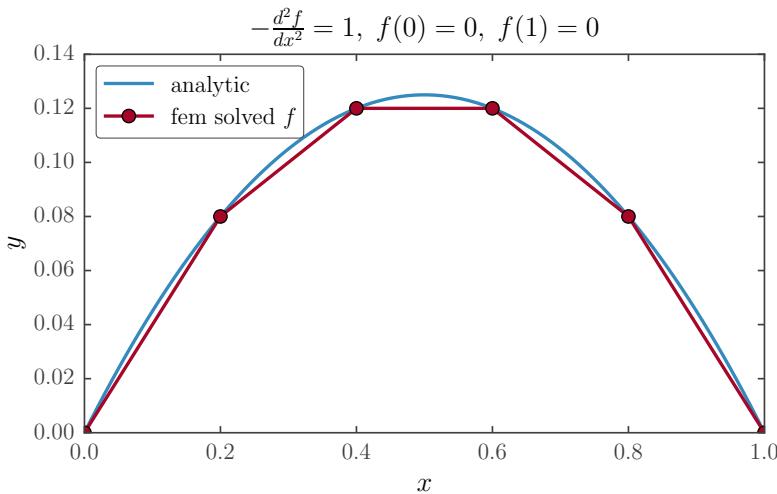


图 5.10 方程 $-\frac{d^2f}{dx^2} = 1, f(0) = 0, f(1) = 0$ 的解析解和有限元数值解。

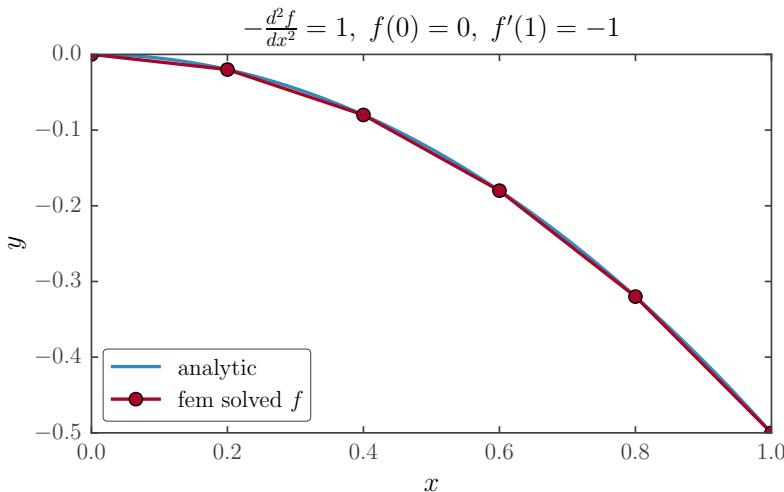


图 5.11 方程 $-\frac{d^2f}{dx^2} = 1, f(0) = 0, f'(1) = -1$ 的有限元数值解和解析解。

例题 5.3: 利用有限元方法计算

$$-\frac{d}{dx} \left(k\alpha(x) \frac{df}{dx} \right) = \rho(x)$$

其中 $k = 1, \alpha(x) = x^2(1+x), \rho(x) = -2x$ 。边界条件为 $f(0) = 0, f'(1) = 0.5$ 。

提示：函数 f 的解析解为 $\ln(x+1)$ 。

我们首先写出 K_{IJ}^e 和 s_I^e 的计算式

$$K_{IJ}^e = \int_{\Omega_0} k\alpha(J_e t + \frac{x_e + x_{e+1}}{2}) S'_I S'_J \frac{1}{J_e} dt$$

$$s_I^e = \int_{\Omega_0} \rho(J_e t + \frac{x_e + x_{e+1}}{2}) S_I J_e dt$$

在本例题中，由于媒介特性 $\alpha(x)$ 和源特性 $\rho(x)$ 是 x 的函数。因此在移动到主元上时，需要进行坐标变换（其中 $J_e = h_e/2$ ）

$$x = J_e t + \frac{x_e + x_{e+1}}{2}$$

另外，在计算边界条件 B_i 时

$$B_i = \int_{\Gamma} \alpha(x) \varphi_i \frac{df}{dx} \hat{n}_x d\Gamma$$

对于第一类边界条件，若 $f(0)$ 已知，则对于第一个节点 $i = 1$ 有 $\varphi_1 = 0$ ，进而 $B_1 = 0$ 。而对于第二类边界条件，若 $f'(1)$ 已知，在 $i = n$ 节点上有 $\varphi_n = 1$ ，进而 $B_n = \alpha(1)f'(1)$ 。将 \mathbf{K} , \mathbf{s} 和 \mathbf{B} 算出后，即可求解得到 \mathbf{f} ，见图5.12。

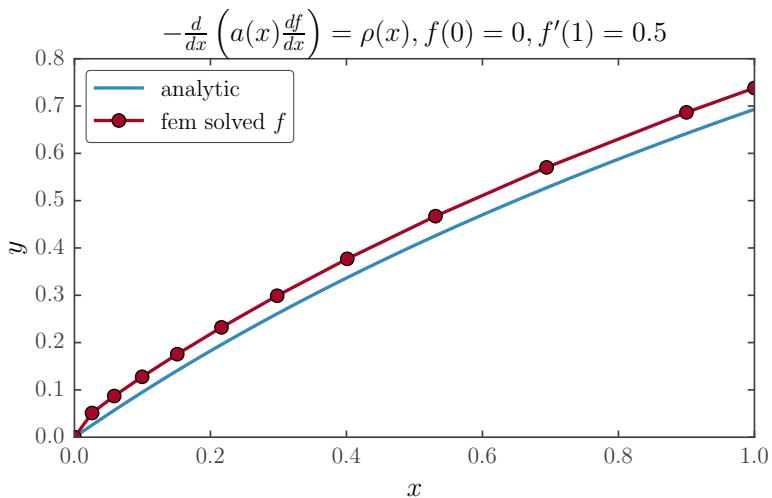


图 5.12 方程 $-\frac{d}{dx} \left(k \alpha(x) \frac{df}{dx} \right) = \rho(x)$ 的有限元数值解和解析解。

图5.12中需要注意的是，我们划分网格（mesh）时，可将 x 轴按照对数坐标划分，在靠近 0 端点处网格较密，在靠近 1 端点处网格较为稀疏。另外，从求解得到的 f_i 中也可以看到，有限元法与解析解存在一定的误差。

5.2.10 上机练习

上机 5.1: (5 分) 编程实现一维的有限元算法，计算

$$-\frac{d^2 f}{dx^2} = 1, \quad f'(0) = 1, \quad f(1) = 0.5$$

提示：函数的解析解为 $f = -x^2/2 + x$ 。需要注意的是，在处理边界条件时，不要忽略边界积分式(5.22)中的 \hat{n}_x 。

上机 5.2: (10 分) 在一维场景中，我们先前学习的有限差分法也是一个很有效的数值求解方法。请编写 FDM 程序求解本节的三个例题。其中节点上的一阶和二阶差分可用下式计算

$$\begin{aligned}\frac{df}{dx} &= \frac{f_{n+1} - f_{n-1}}{2h} \\ \frac{d^2f}{dx^2} &= \frac{f_{n+1} - 2f_n + f_{n-1}}{h^2}\end{aligned}$$

FDM 法可以方便的处理第一类边界条件，只需对 f_1 或者 f_n 赋值即可。对于第二类边界条件，若边界点 n 或点 1 的一阶导数已知，则

$$\begin{aligned}f''_n &= \frac{f_{n-1} - 2f_n + f_{n+1}}{h^2} \\ &= \frac{2f_{n-1} - 2f_n - (f_{n-1} - f_{n+1})}{h^2} \\ &= \frac{2f_{n-1} - 2f_n}{h^2} - \frac{2f'_n}{h}\end{aligned}$$

同理对于起始点 f''_1

$$f''_1 = \frac{2f_2 - 2f_1}{h^2} - \frac{2f'_1}{h}\hat{n}_x$$

其中定义：在区间的两个端点 a 和 b 上 $\hat{n}_a = -1$, $\hat{n}_b = 1$ 。同学们对比这个表达式与有限元法中计算 B_i 的边界条件式(5.22)的关系。

讨论有限元法与有限差分法的优劣。试回答：有限差分法可以集成媒介特性 $\alpha(x)$ 和源特性 $\rho(x)$ 么？你的编程结果又是怎样的。

上机 5.3: (20 分) 我们通过分部积分法得到了有限元的弱形式(5.8)，这样避免了二阶微分，同时方便我们在 $C^0(\Omega)$ 空间上寻找 φ_i 和 ω_i ，而不是在 $C^2(\Omega)$ 上直接寻找 f 。

试着直接利用限元的强形式(5.4)构建系数矩阵 K_{IJ}^e 。并试着用得到的系数矩阵求解本节中的例题，在这种情况下，你能否得到数值解？如果不行的话，遇到了什么问题？

5.3 二维有限元

有限元法的优势在于能够对复杂形状建模。而对复杂物体建模，需要用到无结构化网格（unstructured mesh）。在不同维度下，一些基本的网格形状可见图5.13。一维网格由线条构成，而在二维，我们多用三角形（triangles）网格。

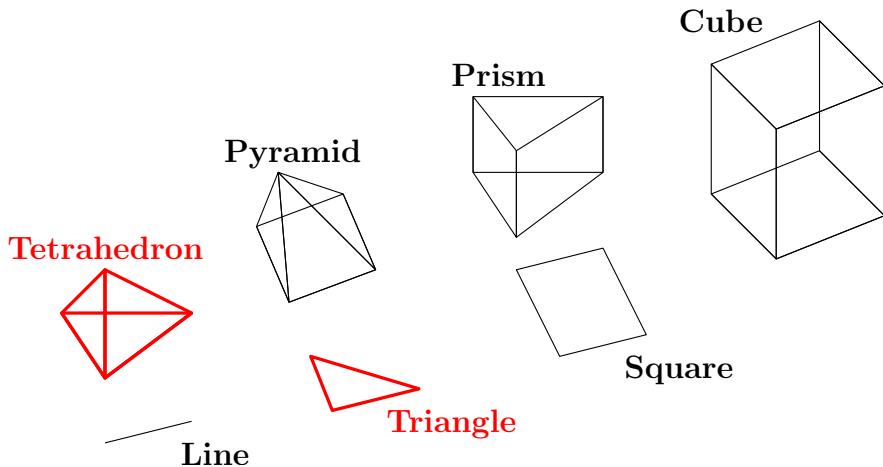


图 5.13 基本元的形状：一维下是直线，二维是三角和方形，三维是四面体（Tetrahedron），三棱柱（Prism），金字塔（Pyramid）和立方体（Cube）。

5.3.1 弱形式与边界条件

在二维情形下，令 f 表示该区域内的电势，满足

$$-\nabla \cdot (\alpha \nabla f) + \beta f = s, \quad \text{in } S, \quad (5.28)$$

$$f = p, \quad \text{in } L_1, \quad (5.29)$$

$$\hat{n} \cdot (\alpha \nabla f) = q, \quad \text{in } L_2 \quad (5.30)$$

其中区域 S 的左边界为 L_1 ，右下边界为 L_2 。在这两个边界上，分别包含了式(5.29)中的 Dirichlet 边界条件和式(5.30)中 Neumann 边界条件。

在静电场数值计算中，基于式(5.28)–(5.30)可以有很多应用。例如，我们考虑图5.14中的电场数值计算的问题：求解区域 S 上，从左边边界到右下边界之间的电阻值。在这种情况下，式(5.28)–(5.30)可以进一步简化。其中， α 等价于媒介的电导率 σ ， $\beta = 0$ 。另外，我们假设区域 S 内没有自由电荷，则 $s = 0$ 。更进一步，我们在左边的边界区域上加 10V 的电压，即得到 Dirichlet 边界条件式(5.29)中 $f = 10V$ 。我们再令右下边界接地，则该处电势 $f = 0V$ 。除此以

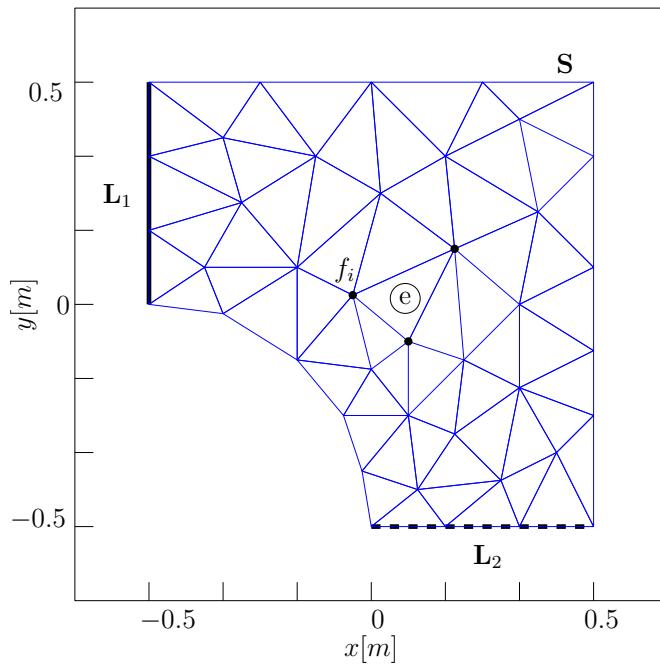


图 5.14 二维导电平板。区域 S 被自动划分成了三角形的网格，边界 L_1 和 L_2 上的边界条件由(5.29)和式(5.30)给出。

外，令图5.14中除 L_1 、 L_2 外的边界都为绝缘体，没有电荷从此类边界上流入或者流出，即 Neumann 边界条件式(5.30)中 $\hat{\mathbf{n}} \cdot (\nabla f) = 0$ 。

接下来，我们利用加权余量法推导二维有限元的弱形式。将式(5.28)两边乘以 ω_i ，并在 S 上积分

$$\int_S \omega_i [-\nabla \cdot (\alpha \nabla f) + \beta f] dS = \int_S \omega_i s dS$$

利用分部积分公式

$$\nabla \cdot [\omega_i (\alpha \nabla f)] = \nabla \omega_i \cdot \alpha \nabla f + \omega_i \nabla \cdot (\alpha \nabla f)$$

以及高斯积分公式

$$\int_S \nabla \cdot \mathbf{F} dS = \int_{L_1+L_2} \hat{\mathbf{n}} \cdot \mathbf{F} dl$$

其中 $\mathbf{F} = \omega_i \alpha \nabla f$ ，可得

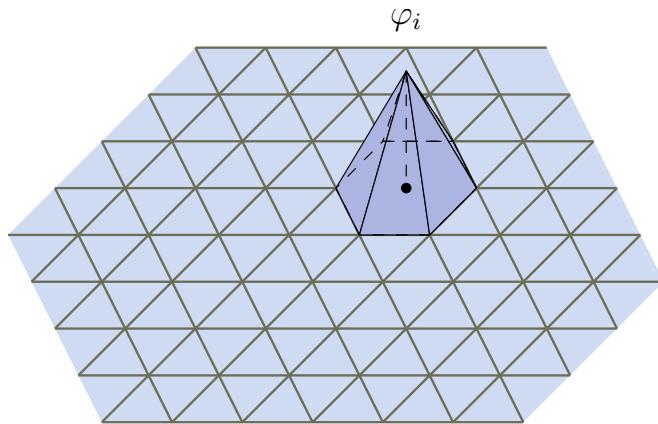
$$\int_S (\alpha \nabla \omega_i \cdot \nabla f + \beta f) dS = \int_S \omega_i s dS + \int_{L_2} \omega_i q dl \quad (5.31)$$

其中在 L_1 上的积分由于边界值 f_{L_1} 已知，检验函数 $\omega_{L_1} = 0$ ，边界积分中不包含 L_1 部分。在 L_2 边界上，已知 $\hat{\mathbf{n}} \cdot \alpha \nabla f$ ，即 $\alpha \nabla f$ 的通过围线 L_2 向外的通

量。式(5.31)即为二维情形下的弱形式。

5.3.2 基函数与有限元方程

在二维情形下，我们设节点数为 n ，面元数为 n_e 。第 i 个节点的坐标用 \mathbf{r}_i 表示，其中 $i = 1, \dots, n$ 。我们令 $\varphi_i(\mathbf{r})$ 代表第 i 个节点的线性基函数。 $\varphi_i(\mathbf{r})$ 函数具备局部化的特性，在 \mathbf{r}_i 处为 $\varphi_i(\mathbf{r}_i) = 1$ ，而在其它节点 $j \neq i$ 上为 $\varphi_i(\mathbf{r}_j) = 0$ 。有限元和其节点上的基函数统称为节点面元（nodal elements）。



我们将待求函数 f 在基函数上展开

$$f = \sum_i^n f_i \varphi_i(\mathbf{r}) \quad (5.32)$$

将式(5.32)代入式(5.31)中，并采用伽辽金有限元 $\omega_i = \varphi_i$ ，可得线性方程组 $\mathbf{K}\mathbf{f} = \mathbf{b}$ ，其中

$$K_{ij} = \int_S (\alpha \nabla \varphi_i \cdot \nabla \varphi_j + \beta \varphi_i \varphi_j) dS \quad (5.33)$$

$$b_i = \int_S \varphi_i s dS + \int_{L_2} \varphi_i q dI \quad (5.34)$$

其中 j 要遍历所有节点，而 i 则只在函数值未知的节点上计算（即不包含 Dirichlet 边界条件 L_1 上的节点）。

$$\left(\begin{array}{c|c} \mathbf{K}_e & \mathbf{K}_n \end{array} \right) \begin{bmatrix} \mathbf{f}_e \\ \mathbf{f}_n \end{bmatrix} = \mathbf{K}_e \mathbf{f}_e + \mathbf{K}_n \mathbf{f}_n = \mathbf{b}$$

其中 \mathbf{f}_e 为 Dirichlet 边界条件上的函数节点， \mathbf{f}_n 为未知节点， \mathbf{K}_n 是方阵，最终需要从 $\mathbf{K}_n \mathbf{f}_n = \mathbf{b} - \mathbf{K}_e \mathbf{f}_e$ 中求解 \mathbf{f}_n 。

在本小节的最后，我们讲下如何在有限元的框架下，计算电磁参数，求解图5.14中的阻抗。假定导电板的厚度为 h ，则

- 通过电流密度表达式 $\mathbf{J} = -\sigma \nabla f$ ，首先计算通过选定横截面的电流，

$$I = \int_{z=0}^h \int_{x=0}^{0.5} \sigma \frac{\partial f}{\partial y} \Big|_{y=-0.5} dx$$

再利用 $R = U/I$ 计算得到电阻。

- 首先计算功率损耗，

$$P = \int_V \mathbf{J} \cdot \mathbf{E} dV = \int_V \sigma |\nabla f|^2 dV = h \mathbf{f}^T \mathbf{K} \mathbf{f} = h \mathbf{f}^T \mathbf{b}$$

其中我们用到了，

$$h \int_S \sigma |\nabla f|^2 = h \sum_i f_i \sum_j \left(\int_S \sigma \nabla \varphi_i \cdot \nabla \varphi_j dS \right) f_j$$

再利用 $R = U^2/P$ 计算电阻。

很显然，利用有限元法可以方便的计算功率，进而得到电阻值。

5.3.3 系数矩阵的组装

在计算 K_{ij} 时，我们可以首先计算面元 e 上的 K_{ij}^e ，

$$K_{ij}^e = \int_{S^e} \nabla \varphi_i \cdot \nabla \varphi_j dS$$

随后得到

$$K_{ij} = \sum_e K_{ij}^e = \sum_e \int_{S^e} \nabla \varphi_i \cdot \nabla \varphi_j dS \quad (5.35)$$

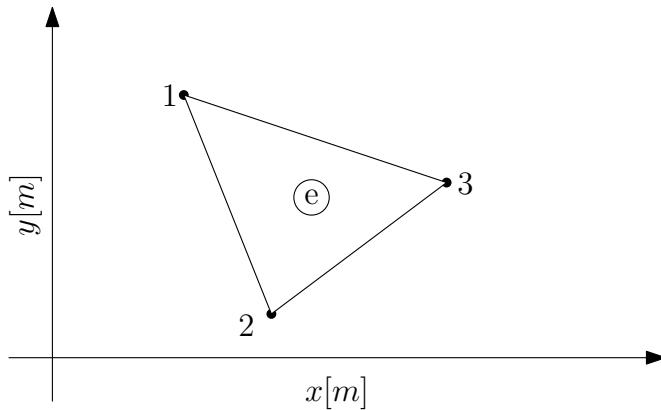
其中 n_e 为总面元数。

在实现 FEM 算法时，为了程序的通用性，我们多采用数值积分方法，并引入主元和坐标旋转简化计算过程。但是，对于静电场数值计算问题

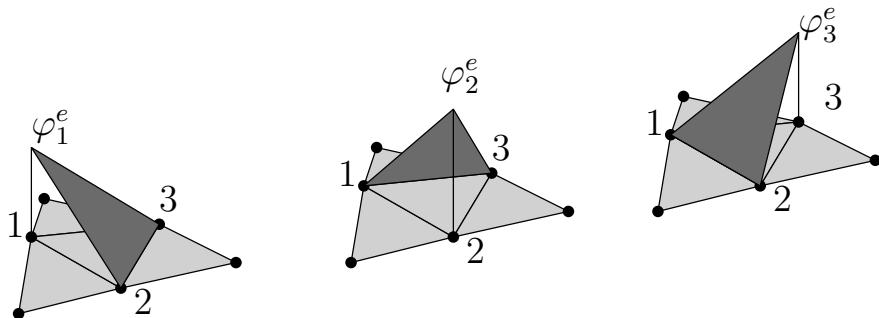
$$-\nabla \cdot (\alpha \nabla f) = 0$$

我们可采用线性基函数，直观、简洁的得到面元 e 上积分的解析解。下面，我们将推导系数矩阵解析计算的公式。通过研究 $-\nabla^2 f = 0$ 这类简单的偏微分方程，也能够便于我们学习二维有限元系数矩阵 \mathbf{K} 的组装步骤。

首先，我们单独考虑一个面元 e ，见图5.15。在局部坐标系下，面元 e 的三

图 5.15 面元 e 的局部坐标系。

个端点的坐标分别为 \mathbf{r}_1^e , \mathbf{r}_2^e 和 \mathbf{r}_3^e 。我们用 $\varphi_i^e(x, y)$ 表示面元 e 节点上的基函数, 其中上标表示基函数所处的面元 e , 下标表示对应的局部节点序号 $i = 1, 2, 3$ 。对于面元中的所有三个节点 1, 2, 3, 都可关联一个基函数, 其形状如图 5.16 所示。需要注意的是, 节点 i 的全局基函数 φ_i 由以节点 i 为端点的所有面元上的基函数构成。

图 5.16 三角形面元 e 的三个基函数, 与 e 共用相邻边界的面元也绘于图中。

在每一个面元 e 上, 基函数 φ_i^e 具备线性

$$\varphi_i^e(x, y) = a_i^e + b_i^e x + c_i^e y \quad (5.36)$$

及局部化的特点

$$\varphi_i^e(x_i^e, y_i^e) = 1, \quad \varphi_i^e(x_j^e, y_j^e) = 0, \forall i \neq j \quad (5.37)$$

下一步, 我们推导面元 e 上基函数 $\varphi_i^e(x, y)$ 的解析表达式。首先, 面元 e 中的任意一点 $\mathbf{r} = (x, y)$ 都可表示为矢量 $\mathbf{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}}$ 。任选 e 中的一点 \mathbf{r} , 都可将 e 分成三个小三角形, 见图 5.17。其中, A_i^e 代表面元 e 中的节点 i 对向位

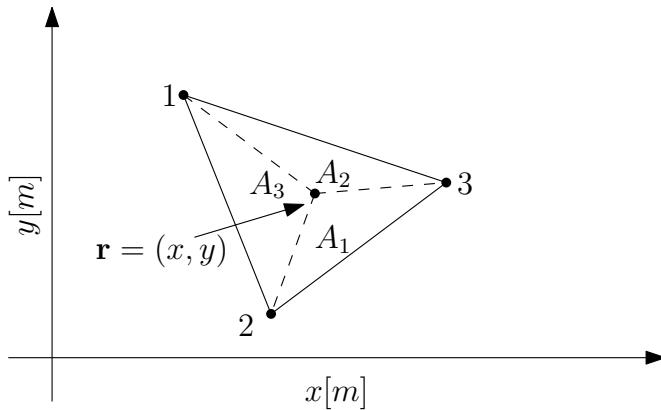


图 5.17 构建 $\varphi_i^e(x, y)$ 的分区样式。

置的子三角形。面元 e 的总面积用 A_{tot}^e 表示, 有 $A_{tot}^e = A_1^e + A_2^e + A_3^e$ 。

5.3.4 系数矩阵的解析计算方法

节点上的基函数 $\varphi_i^e(\mathbf{r})$ 可以利用面积坐标表示¹, 即

$$\varphi_i^e(\mathbf{r}) = \frac{A_i^e}{A_{tot}^e} \quad (5.38)$$

式中 A_i^e 的面积随着 \mathbf{r} 位置的不同而发生变化。很容易验证式(5.38)满足基函数线性(5.36)和局部化(5.37)的特性。比如对于 φ_1^e , 当 $\mathbf{r} = \mathbf{r}_1^e$ 时, $\varphi_1^e = 1$ 。

更进一步, 面积 A_i^e 可以写为向量叉乘的形式

$$\begin{aligned} A_1^e &= \frac{1}{2} \hat{\mathbf{z}} \cdot (\mathbf{r}_3^e - \mathbf{r}_2^e) \times (\mathbf{r} - \mathbf{r}_2^e) \\ A_2^e &= \frac{1}{2} \hat{\mathbf{z}} \cdot (\mathbf{r}_1^e - \mathbf{r}_3^e) \times (\mathbf{r} - \mathbf{r}_3^e) \\ A_3^e &= \frac{1}{2} \hat{\mathbf{z}} \cdot (\mathbf{r}_2^e - \mathbf{r}_1^e) \times (\mathbf{r} - \mathbf{r}_1^e) \end{aligned}$$

或者统一写为

$$A_i^e = \frac{1}{2} (\mathbf{r} - \mathbf{r}_{i+1}^e) \cdot \hat{\mathbf{z}} \times \mathbf{s}_i \quad (5.39)$$

其中

$$\mathbf{s}_i = \mathbf{r}_{i-1} - \mathbf{r}_{i+1} \quad (5.40)$$

为逆时针方向上三角形第 i 条边所代表的向量。

¹这时 φ_i^e 也被称作 simplex 坐标系或者重心坐标系 (barycentric coordinates)。

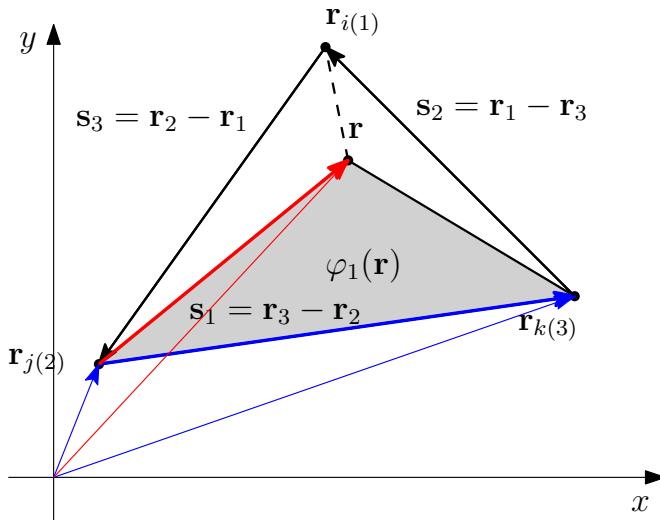


图 5.18 重心坐标与面元的解析计算公式

总面积 A_{tot}^e 可表示为

$$A_{tot}^e = \frac{1}{2} \hat{z} \cdot s_2 \times s_3 \quad (5.41)$$

对于梯度 $\nabla \varphi_i^e$, 可写为

$$\nabla \varphi_i^e = \frac{\hat{z} \times s_i}{2A_{tot}^e} \quad (5.42)$$

显然, 由于面元 e 的边长是固定的, 因而 $\nabla \varphi_i^e$ 是一个恒定的向量场 (正如一维情形下基函数的导数 φ' 是一个常量一样)。面元 e 上的系数矩阵 K_{ij}^e 为

$$K_{ij}^e = \int_{S^e} \nabla \varphi_i^e \cdot \nabla \varphi_j^e dS = \frac{s_i \cdot s_j}{4A_{tot}^e} \quad (5.43)$$

需要注意的是, 我们现在使用的是面元 e 的局部节点序号 $i, j = 1, 2, 3$, 最后, 我们需要将局部序号映射到系数矩阵 \mathbf{K} 的节点全局编号上。

在利用 MATLAB 或 Python 实现时, 面元 e 的三个节点坐标 (x, y) 可用一个 2×3 的矩阵表示, 对应于三角形面元的三个节点; 另外, 三个节点两两计算 ($\int_{S^e} \nabla \varphi_i^e \cdot \nabla \varphi_j^e dS$), 会生成一个 3×3 的局部系数矩阵 \mathbf{K}^e 。在程序编写时, 我们可设计一个简单的函数, 输入为任意面元 e 的坐标矩阵, 输出面元 e 对应的系数矩阵 \mathbf{K}^e 。最后, 我们将 \mathbf{K}^e 组装到全局系数矩阵 \mathbf{K} 上。

5.3.5 系数矩阵的数值计算方法 (选讲)

实际编程时, 对于 K_{ij}^e 和 b_i^e ,

$$K_{ij} = \int_S (\alpha \nabla \varphi_i \cdot \nabla \varphi_j + \beta \varphi_i \varphi_j) dS \quad (5.44)$$

$$b_i = \int_S \varphi_i s dS + \int_{L_2} \varphi_i q dl \quad (5.45)$$

也可 (在现有商用数值计算程序中, 多采用主元方法) 采用主元的方法: 将待计算的面元 e 平移旋转变换到主元之后, 利用高斯数值积分计算。利用主元方法, 对于面元上的电导率 α 及源分布函数 s 同样需要进行坐标变换, 进行数值计算。

5.3.6 连接矩阵

在二维有限元中, 我们需要一种高效、直观的数据结构, 用来存储和处理非结构化网格。以图 5.19 为例, 图中的网格包含 6 个节点和 4 个元。我们分别

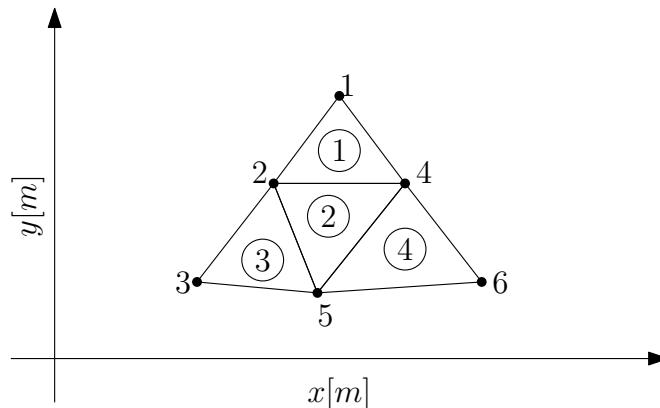


图 5.19 二维非结构化网格。节点全局编号标记于节点旁, 面元编号置于三角形内。

建立两个矩阵用来描述网格信息:

- $no2xy$, 是一个 $2 \times n$ 的矩阵, 用来存储每一个节点的 (x, y) 坐标
- $el2no$, 是一个 $3 \times n_e$ 的矩阵, 用来存储每一个面元 e 对应的三个节点编号, 节点编号以逆时针的顺序存放。 $el2no$ 即组装矩阵 LtoG。

对于图 5.19 中所示的例子, $no2xy$ 可见表 5.1, $el2no$ 可见表 5.2。

5.3.7 实施步骤

二维有限元的实现步骤可以总结为三步

表 5.1 给定节点的全局编号, no2xy 存储节点的 (x, y) 坐标

Node	1	2	3	4	5	6
x	0.0	-0.5	-0.8	0.6	0.0	1.0
y	1.0	0.5	0.0	0.4	-0.2	-0.1

表 5.2 给定面元的全局编号, el2no 存储面元所对应的节点序号

Element	1	2	3	4
Node 1	1	4	3	5
Node 2	2	2	5	6
Node 3	4	5	2	4

1. 网格化：将二维区域划分成由三角面元构成的网格
2. 组装系数矩阵：利用数值积分法组装生成 \mathbf{K} 和 \mathbf{b}
3. 边界条件：计算边界积分并按照边界条件修改矩阵 \mathbf{K} 和 \mathbf{b}

需要注意的是，网格划分（mesh generation）本身就是一个热门的研究领域，我们不要求学生自己编写网格划分算法，但是要求会编程生成圆形区域的网格，同时会使用网格生成软件生成复杂形状的网格。大多网格生成软件，使用自己固有的输入输出格式。同学们基于先前讲到的利用 $no2xy$ 和 $el2no$ 矩阵描述非结构化网格结构的思想，可以针对性的编写单独的格式转化函数，将网格生成工具输出的特定网格文件，转化成我们自己的程序需要的方式。

5.3.8 计算示例

我们将通过一个实例，来展示 FEM 法的计算步骤，

1. 给定面元 e 的节点坐标矩阵 $no2xy$ ，计算局部系数矩阵 \mathbf{K}_{IJ}
2. 根据面元 e 的组成矩阵 $el2no$ ，将局部系数矩阵 \mathbf{K}_{IJ} 组装到全局系数矩阵 \mathbf{K}_{ij} 对应的位置上，即 $IJ \rightarrow ij$
3. 根据边界条件求解 f_i ，将电势的数值计算结果可视化的展现出来
4. 根据 \mathbf{K} 和 \mathbf{f} 计算静电场的能量，求出电磁参数

例题 5.4：计算同轴矩形波导之间的电容 C 。其中 $a = b = 1\text{cm}$, $c = d = 2\text{cm}$ 。内部矩形波导上的电压为 1V，外部矩形传输线的电压为 0V，内外波导之间的

介质为真空。

分析：本例题为静电场的数值计算，其数学形式为 laplace 方程：

$$-\nabla^2 f = 0, \quad \text{in } S \quad (5.46)$$

$$f = 1, \quad \text{in } L_{in} \quad (5.47)$$

$$f = 0, \quad \text{in } L_{ext} \quad (5.48)$$

其中 S 为波导之间的真空区域， L_{in} 为内波导的边界， L_{ext} 为外波导的边界。

利用式(5.46)，可得 $\mathbf{b} = 0$ 及

$$K_{ij}^e = \int_{S^e} \nabla \varphi_i^e \cdot \nabla \varphi_j^e dS \quad (5.49)$$

在选取线性节点基函数（Nodal Basis）时，给定 I, J 的节点坐标，我们可以直接用式(5.42)解析计算 K_{ij}^e ，

$$K_{IJ}^e = \frac{\mathbf{s}_I \cdot \mathbf{s}_J}{4A_{tot}^e} \quad (5.50)$$

其中 I, J 是 $el2no$ 中的局部坐标。接下来，利用矩阵 $el2no$ ，提取第 e 个面元 I, J 位置所对应的节点坐标 i, j ，并将计算结果加到系数矩阵对应的 i, j 位置上。

在得到系数矩阵 \mathbf{K} 后，我们需要将它分为 \mathbf{K}_e 和 \mathbf{K}_n ，其中 \mathbf{K}_e 对应于已知边界条件的节点， \mathbf{K}_n 对应于未知格点。通过求解 $\mathbf{K}_n \mathbf{f}_n = -\mathbf{K}_e \mathbf{f}_e$ ，我们可得到内部格点的电势 \mathbf{f}_n 。

最后，为了计算电容 C ，我们首先计算能量 W ，利用 $C = 2W/U^2$ 间接地得到电容值。其中 W 为每单位长度的静电势能， U 是内外导之间的电压差。能量 W 由下式计算，

$$\begin{aligned} W &= \frac{1}{2} \int_S \mathbf{E} \cdot \mathbf{D} dS = \frac{1}{2} \int_S \epsilon_0 |\nabla f|^2 dS \\ &= \frac{\epsilon_0}{2} \sum_i \sum_j f_i \left(\int_S \nabla \varphi_i \cdot \nabla \varphi_j dS \right) f_j \\ &= \frac{\epsilon_0}{2} \mathbf{f}^T \mathbf{K} \mathbf{f} \end{aligned}$$

具体的实现同学们可以参照代码 `fem2d.py`。我们将计算结果可视化，见图5.20和图5.21。其中用到了 `matplotlib` 的 `tripcolor` 以及 `mplplot3d` 的 `plot_trisurf` 画图函数。我们也可以用 `tricontour` 或者 `tricontourf` 来绘制

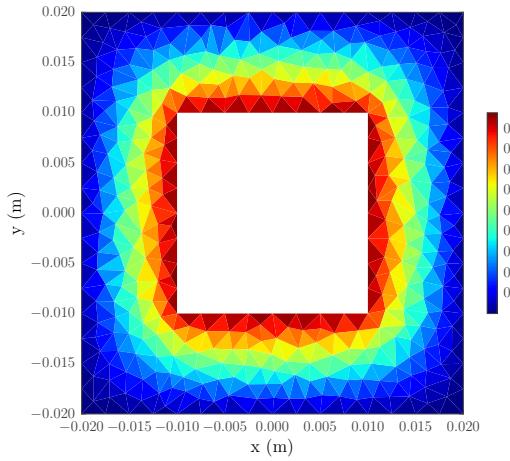


图 5.20 同轴矩形传输线电容的 FEM 数值计算结果，采用 tripcolor 可视化。

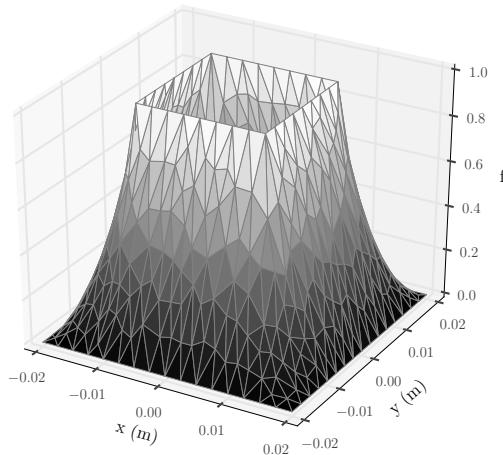


图 5.21 同轴矩形传输线电容的 FEM 数值计算结果，采用 trisurf 可视化。

等值线。这样，在二维有限元计算中，当计算得到节点上的函数值 f 之后，可方便的调用 python 的绘图函数将结果绘制出来。

针对本例题，我们还需要计算电容 C 。

- 采用均匀划分，设定所有面元的最大面积为 0.025。生成的网格中包含节点 423 个，面元 750 个，电容计算结果 $C = 91.68866 \text{ pF/m}$
- 面元起始最大面积为 0.05，仅在内导四个角点处细化网格。共包含节点 214 个，面元 363 个，电容计算结果为 $C = 91.57589 \text{ pF/m}$

可见，通过在形状不连续处对网格进行细化，不仅可以减小网格规模，节省计算量。在优化策略选择得当的情况下，可以得到比均匀细化更优的结果。

5.3.9 三维有限元方法（选讲）

在有了二维有限元基础后，三维有限元可以看作是二维的扩展。本节简要的介绍下三维有限元计算中的关键部分：系数矩阵 K_{ij} 的计算。

例题 5.5 (三维 Tetrahedron 上的有限元计算): 在推导二维有限元系数矩阵的解析计算中，我们用到重心坐标系方便直观的表示了节点 i 上的测试函数 $\varphi_i(\mathbf{r})$ ，然后推导了得到了 $\nabla \varphi_i(\mathbf{r})$ 。这一思想可以很方便的推广到三维的形式。

我们考虑三维有限元划分中的四面体，见图 5.22。四面体的体积 V 为，

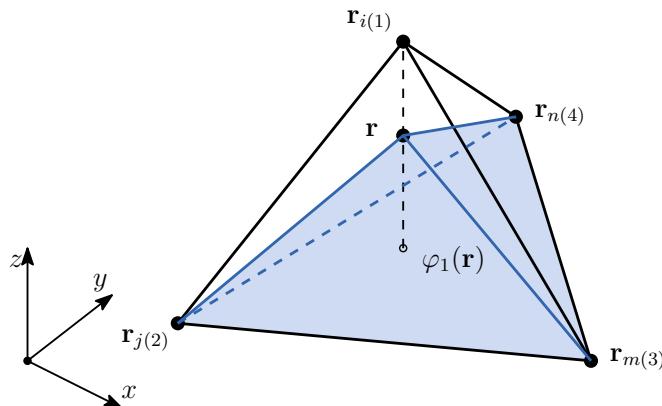


图 5.22 三维中的有限元四面体 (Tetrahedron)。四面体的四个顶点我们用 $\mathbf{r}_{i(1)}, \mathbf{r}_{j(2)}, \mathbf{r}_{m(3)}, \mathbf{r}_{n(4)}$ 表示，其中 (i, j, m, n) 表示节点的全局编号， $(1, 2, 3, 4)$ 为节点的局部编号。

$$V = \frac{1}{3!} \begin{vmatrix} r_x^1 & r_y^1 & r_z^1 & 1 \\ r_x^2 & r_y^2 & r_z^2 & 1 \\ r_x^3 & r_y^3 & r_z^3 & 1 \\ r_x^4 & r_y^4 & r_z^4 & 1 \end{vmatrix} \quad (5.51)$$

对于节点 1 而言，其基函数 $\varphi_1(\mathbf{r})$ 可以写成以 $(2, 3, 4)$ 为底、 \mathbf{r} 为顶点的四面体体积，

$$\varphi_1(\mathbf{r}) = \frac{1}{3V} \left(\frac{1}{2} \mathbf{s}_{23} \times \mathbf{s}_{24} \right) \cdot (\mathbf{r} - \mathbf{r}_2) \quad (5.52)$$

而 $\nabla\varphi_1(\mathbf{r})$ 为,

$$\nabla\varphi_1(\mathbf{r}) = \frac{1}{3!V} (\mathbf{s}_{23} \times \mathbf{s}_{24}) = \frac{\mathbf{A}_1}{3!V} \quad (5.53)$$

其中 $\mathbf{s}_{23} = \mathbf{r}_3 - \mathbf{r}_2$, $\mathbf{s}_{24} = \mathbf{r}_4 - \mathbf{r}_2$ 。

为了表示的方便, 我们定义节点 i 对向的三角面元 \mathbf{A}_i 为

$$\mathbf{A}_i = \mathbf{s}_{i,i+1} \times \mathbf{s}_{i,i+2} \quad (5.54)$$

则局部系数矩阵 \mathbf{K}_e 可由下式计算,

$$K_{ij} = \int_{V_e} \sigma_e \nabla\varphi_i \cdot \nabla\varphi_j dV = \frac{1}{3!} \cdot \frac{\mathbf{A}_i \cdot \mathbf{A}_j}{V} \quad (5.55)$$

5.4 上机练习

5.5 网格剖分 (选讲)

本节我们将讲解如何用 `meshpy` 生成无结构化网格。`meshpy` 是 Andreas Klöckner 开发的 mesh 工具包, 可以从 <https://github.com/inducer/meshpy/> 下载。MeshPy 可以方便的调用 Triangle 和 tetgen 生成二维及三维的网格, 同时还具备强大的网格自动优化功能。

注意, 我们可以用很多工具生成网格¹。但是, 不要沉溺于工具的使用, 而将重点放在网格输出的格式运用、网格优化、以至有限元的理解使用上。

5.5.1 函数输入

MeshPy 调用 Triangle[13] (作者在代码主页上声明, Triangle 软件的第一个字母 T 必须大写) 生成二维三角形网格。Triangle 具备优异的性能, 并且能够按照需要 (需指定 `refine` 函数) 对生成的网格进行细分。**MeshPy** 接口简洁, 主要步骤见图 5.23。

1. `nodes`: 我们在 $x - y$ 坐标轴上生成 n 个节点, 第 i 个节点的位置由坐标 (x_i, y_i) 表示。按照生成节点的顺序, **MeshPy** 依次对每个节点进行编号 $i = 1, 2, \dots, n$

¹请参考 <http://people.sc.fsu.edu/~jburkardt/>, 在他的 presentation 目录下面, 有很多介绍有限元和网格划分的文档。尤其是, MATLAB 下很棒的网格划分软件 `distmesh` 就是这个作者编写的, 而且我们也可以用另外一个 MATLAB 工具 `mesh2d` 编写自动优化的程序。

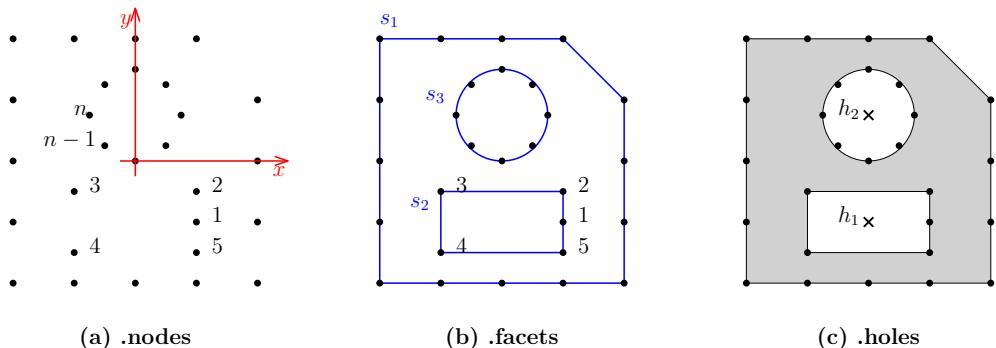


图 5.23 MeshPy 生成二维网格的流程图。

2. facets: 根据节点编号生成连接多个节点的封闭曲线 s_i 。例如，对于图5.23 (b) 中的曲线 s_2 ，其格式为

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 1 \end{bmatrix}$$

3. `holes`: 我们指定孔洞的坐标, 如图中的 h_1 和 h_2 。MeshPy 会自动根据围线确定孔的区域, 而剩下的区域则被设定为网格的生成区域。如图 5.23(c) 中灰色的区域所示。需要注意的是, 洞的位置不能位于曲线上, 否则 MeshPy 会随机的设定曲线的任意一侧为打孔区域。

除此之外，**MeshPy** 的一个重要功能是面元的优化（refinement）。我们可以（1）通过指定优化函数 `refinement_func` 来进行自适应的优化，或（2）限定全局面元的面积（area）来进行约束，或（3）通过在图形中划分不同的区域（regions）分别进行特定的约束。总体来说，区域的描述（节点、线型、孔洞）代码相对简单；而面元的约束是此类工具的难点。

5.5.2 函数输出

MeshPy 的输出包括

1. `mesh.points` 节点的编号和对应的节点坐标 (x, y)
 2. `mesh.elements` 面元的编号和三个节点的全局编号（逆时针方向）
 3. `mesh.point_markers` 节点的标记符，0 为内部节点， $n(n > 0)$ 为 facets 的标记符，可以通过 `facet_markers` 属性进行更改

4. `mesh.element_attributes` 三角面元的属性。只有当使用了 regions 约束，并且在 build 中打开了属性开关后才有效

5.5.3 实例讲解

例题 5.6: 生成矩形同轴电缆截面的三角网格，其中 $a = b = 1$, $c = d = 2$

我们先编写一个子函数，自动按照点的顺序生成头尾相连的连线。这个函数的功能即根据节点的编号（当前节点为顺序编号），生成闭合的曲线。

```
def round_trip_connect(start, end):
    return [(i, i+1) for i in range(start, end)] + [(end, start)]
```

下面，我们分别生成内外导上的节点（nodes）和连线（facets），并指定孔洞（holes）的坐标。将这些信息附在 MeshInfo 上后，我们直接调用 triangle.build 即可生成网格。

```
def main():
    points = [(1, 0), (1, 1), (-1, 1), (-1, -1), (1, -1), (1, 0)]
    facets = round_trip_connect(0, len(points)-1)

    outer_start = len(points)
    points.extend(
        [(2, 0), (2, 2), (-2, 2), (-2, -2), (2, -2), (2, 0)])
    facets.extend(round_trip_connect(outer_start, len(points)-1))

    # triangle info
    info = triangle.MeshInfo()
    info.set_points(points)
    info.set_holes([(0, 0)])
    info.set_facets(facets)

    # build
    mesh = triangle.build(info)

    # extract mesh .nodes .elements .markers
```

```
mesh_points = np.array(mesh.points)
mesh_tris = np.array(mesh.elements)
mesh_attr = np.array(mesh.point_markers)
```

在这些代码中，我们可以直接从 `mesh.points`，`mesh.elements` 中提取节点的位置和面元的结构信息。在 Python 中，可调用 `triplot` 绘制网格图，见图 5.24。

```
import matplotlib.pyplot as plt
plt.triplot(mesh_points[:, 0], mesh_points[:, 1], mesh_tris)
```

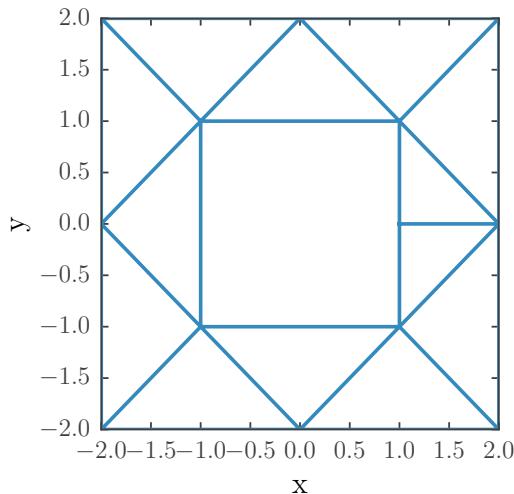


图 5.24 同轴电缆的网格剖分图

在这个例子中，我们展示了 `meshpy.triangle` 最基本的用法，生成同轴矩形波导横截面的三角网格。从图 5.24 中可以很明显的看出，三角面元过大，且网格稀疏，在这种网格下很难用有限元法计算得到较好的数值结果。而对于网格细分，直观的，我们有以下两种思路

1. 编写生成内导和外导节点的函数，增加 facets 上的节点密度¹
2. 编写优化函数，自动对面元的面积进行优化

下面，我们重点讲解 `meshpy.triangle` 的自动优化方法。

例题 5.7：生成同轴矩形传输线截面的网格，要求生成的网格面积小于 0.1

¹当我们生成一个不包含孔的网格时，可以仅仅指定连接边界的点。也就是说，facets 可以不包含所有的 points。`meshpy.triangle` 会自动依据边界围成的所有节点，进行网格划分。

我们首先讲解面元优化函数的编写方法。`meshpy.triangle` 的优化函数包含两个输入参数，第一个是 3×2 的矩阵，代表当前面元 e 上所有三个节点的 (x, y) 坐标；第二个是当前面元 e 的面积。优化函数的输出是一个 Bool 变量，用来表示当前节点是否需要进行进一步的细分。在下面这个例子中，仅仅针对所有面元面积优化，暂时不需要面元 e 的坐标信息 `tri_points`。

```
def refinement_func(tri_points, area):
    max_area=0.1
    return bool(area>max_area);
```

在调用 `triangle.build` 模块时，我们只需指定优化函数即可

```
mesh = triangle.build(info, refinement_func=refinement_func)
```

生成的结果可见图5.25。图5.25和图5.24对比，网格更加密了。

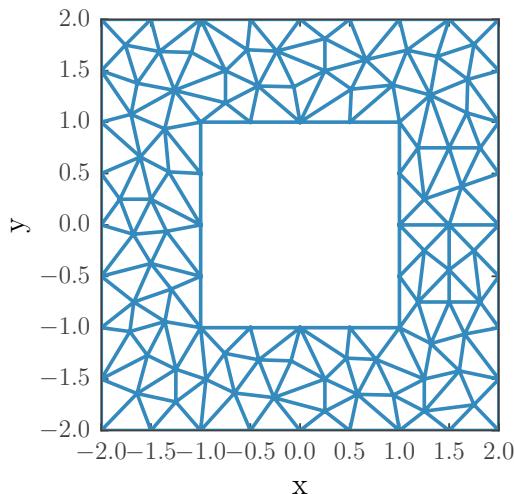


图 5.25 同轴电缆的网格剖分，优化面元面积 $a \leq 0.1$

但是结合我们先前学习的数值计算结果，在内导体的四个角点外部¹，电势 φ 的解具有较大的奇异性。进而在这些位置附近，我们希望能够加入更为严格的约束；而在电磁场解较为规则的区域，可加入较为宽松的约束。

¹一般来说在形状的不连续处，电磁场的数值计算结果往往具备较大的奇异性。这一特性也是一类自适应网格大小调节算法所使用的，即利用曲面的曲率进行自适应网格大小调整。

例题 5.8: 生成同轴矩形传输线横截面的网格，在内导的四个角点附近，实现更为细分（面元面积更小）的网格

内导体四个角点的坐标分别为 $(1, 1)$, $(-1, 1)$, $(-1, -1)$ 和 $(1, -1)$ 。我们可以首先计算每一个面元的中心点坐标（或面元内部任意一点坐标，例如取三个节点 x 轴、 y 轴坐标的平均值），随后计算每一个面元的中心坐标到内导上所有角点的最近距离，最后令面积约束值 a 随着距离的增加而逐步松弛。在下面的代码中，我们令面积约束值 a 正比于面元到角点的距离。

```
def refinement_func(tri_points, area):
    center_tri = np.sum(np.array(tri_points), axis=0)/3.
    max_area = 0.005 + lp.norm(np.abs(center_tri)-1.0) * 0.05
    return bool(area > max_area);
```

生成的网格可参见图5.26。在内导的四个角点附近，我们生成了面积较小、密

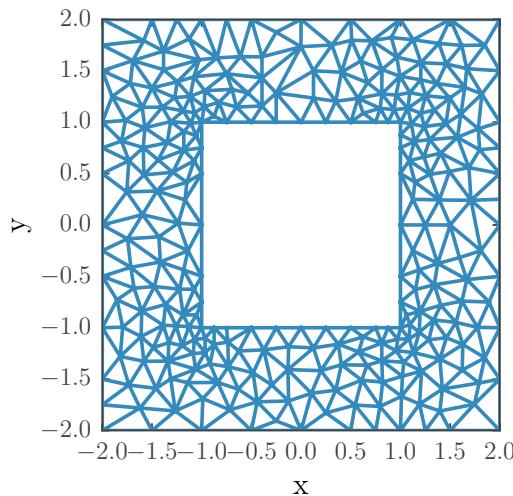


图 5.26 同轴电缆的网格剖分，在内导角点附近优化网格

度较高的网格。这样在编写有限元计算程式时，便于我们在这些角点位置附近，得到精度较高的数值结果。

再来看一个例子，这里，我们将要讲解怎样添加区域（regions）面积约束。

例题 5.9: 对一个半径为 $r = 3$ 圆形区域进行网格划分。其中，圆形内部嵌了一个边长为 $a = 1$ 的正方形。要求：生成的面元面积小于 0.1，且正方形内的面元面积小于 0.05

我们只要用 Python 生成特定形状边界上的节点，就可以制作任意的网格。这里，对于圆形上的节点，我们可以

```
circ_start = len(points)
points.extend(
    (3 * np.cos(angle), 3 * np.sin(angle))
    for angle in np.linspace(0, 2*np.pi, 30, endpoint=False))
```

我们对区域的面积进行约束，设面元面积小于 0.1。同时，如果面元的中心点位于正方形内¹，我们就进一步的约束面元面积为 0.05。

```
def needs_refinement_square(tri_points, area):
    points = [(1, 0), (1, 1), (-1, 1), (-1, -1), (1, -1), (1, 0)]
    polygon = Path(points)
    center_tri = np.sum(np.array(tri_points), axis=0)/3.
    if area>0.1:
        return True
    elif (area > 0.05) and polygon.contains_point(center_tri):
        return True
    else:
        return False
```

除了手动编写面积约束代码之外，我们也可以用 **MeshPy** 的区域约束语言描述。这种方法更加方便，而且可以根据区域的 ID 对区域内的面元进行编号，方便我们在以后的程序中添加区域的电导率等特征参数。例如，我们只需要制定区域（facets）中的内点、区域编号、以及区域内的面积约束即可。

¹同学们除了利用现有代码之外，还可以研究一下怎样自己编写“Point in Polygon”的实现。这里需要一些计算几何学的知识，请参考以下两个链接：

<http://www.toptal.com/python/computational-geometry-in-python-from-theory-to-implementation>

<http://www.heikkitoivonen.net/blog/2009/01/26/point-in-polygon-in-python/>

```

info.regions.resize(1)
# points [x,y] in region, + region number, + regional area constraints
info.regions[0] = ([0,0] + [1,0.05])
mesh = triangle.build(info, volume_constraints=True, max_volume=0.1)

```

生成的网格见图5.27。

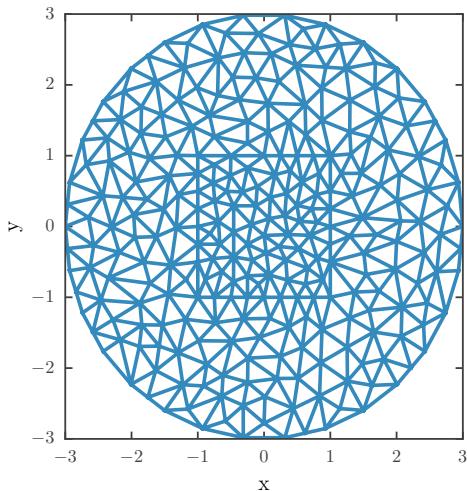


图 5.27 由圆形和方形构成区域的网格自动生成

可见，在 **MeshPy** 中，只要我们能够通过编程 (1) 生成节点坐标，(2) 连接成线，(3) 编写面元优化函数，就可以利用 **Meshpy** 实现任意二维图形的网格剖分。

最后，在实际应用中，我们需要对不同的 facets/points 进行标记。比如在图5.26中，内导的边界上命名为边界 Int，外导的边界上命名为 Ext。这样，我们就可以对不同的 label 上的节点赋予不同的边界条件。

例题 5.10：生成矩形同轴电缆截面的三角网格，其中 $a = b = 1$, $c = d = 2$ 。将内导边界、外导边界、内部节点分别用不同编号标记

我们可以通过指定 `triangle.set_facets` 命令中的 `facet_markers` 参数来实现这个功能。针对图5.26中的例子，我们可以加入

```

markers = [2,2,2,2,2,2]
markers.extend([3,3,3,3,3,3])
info.set_facets(facets, facet_markers=markers)

```

也就是说，内导体边上的所有节点（包含根据面积约束优化后，生成的所有节点）都被赋值为 2，而外导体上的所有节点则被赋值为 3。在 Python 中，我们可以编写一个字典 dict 结构体，来标记不同的边界类型。

本例题中，我们可以通过

```
mesh_attr = np.array(mesh.point_markers)
print mesh_attr
```

查看节点的属性。这里，节点的属性是一个由 [0, 2, 3] 构成一维数组，见图 5.28。在得到 `mesh.point_markers` 之后，我们就可方便的针对不同边界条件上的节点，引入对应的边界条件。

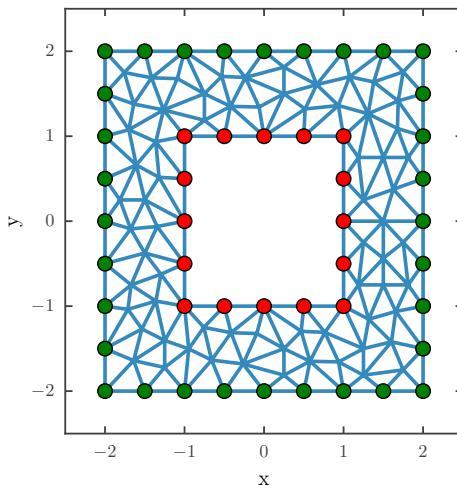


图 5.28 利用 MeshPy 自动标记边界节点。内部导体上的节点用红色表示，外部导体上的节点用绿色表示。

最后，我们总结一些 `meshpy.triangle` 的使用技巧

- 除了编写优化函数，也可通过 `max_volume` 参数来约束面元的面积
- `triangle.build` 有一个参数是 `min_angle`，其作用是约束生成的所有三角形面元中，三个顶角的最小角度不能小于 `min_angle`。在 Triangle 软件中的调用命令开关是 `-q` (Quality Meshing)，默认值 20° 。在 20.7° 以下，算法能确保收敛；对于过大的角度，如 34° 度以上，算法可能无法收敛
- 使用 `mesh.point_markers` 来标记不同类型的边界节点，

- 打开 `triangle.build` 中的 `attributes=True` 属性，划分好区域，并设置区域（Region）的 ID；就可以使用 `element_attributes` 查看生成面元的种类，对应的属于哪一个区域。

同学们在以后的编程中，需要反复运用这些技巧，优化自己的计算程序，进而得到较好的网格剖分和数值计算效果。

第6章 边界元法

刘本源 *bbyliu@fmmu.edu.cn*

学时 上机

2 0

6.1 目的

给学生要讲的简单直观。能够解释边界元法的主要特点，能够解释边界元的计算步骤。

重点：边界元法的思想及求解步骤

难点：边界元方程中系数矩阵的确定

本章主旨：**数学的方法必需以物理概念为依托。**

6.1.1 简介

边界元方法（Boundary Element Method, BEM）是近 30 年来发展形成的一种数值计算方法，提到 BEM，同样不得不提的就是大名鼎鼎的冯康先生，同学们可以下来自己搜索冯康的事迹在手机上读一读。

边界元法相比于其他数值计算方法（以有限元为例），其优点是：

1. 降低问题求解的空间维数。

有限元（FEM）需对全部计算空间进行离散化，而边界元（BEM）只需计算边界积分方程，从而可降低电磁场数值计算问题求解的维数。例如三维问题可以利用边界表面积分降维成二维的面积分问题、二维数值计算问题则可降维成一维的线积分问题。由于待求解的未知参量仅仅包括了边界节点，方程维数降低可减少数据存储和计算量。

2. 易于处理开域问题。

相对于有限元方法，边界元更能方便的计算开域的电磁场问题。对于开域问题，FEM 需要对开域空间进行离散化构建有限元，元（Element）的数目极多

且计算量大。而 BEM 只需对计算边界进行离散化即可。这也是冯康先生基于天线计算的实际需求，开发 BEM 方法的主要动力。

3. 计算精度高。

BEM 可以直接、精确的求解源（例如电荷）的分布，进而，场域中任意一点的场分布将可通过场源作用的线性叠加方式获得，无需再进行微分计算。同时边界元只对边界离散，误差仅仅来源于边界，因而计算精度可优于对场域离散化的有限元方法。本节课程中，我们将通过 BEM 计算示例的讲解，来直观的分析 BEM 方法的数值精度。

但是，BEM 不是完美的，其缺点为：

1. 系数矩阵的计算和存储耗时较多。

系数矩阵需要经数值积分处理，因而系数矩阵的建立需要较多计算时间。

2. 不易处理多种媒质共存的问题。

3. 积分方程的奇异性。

由于 BEM 方法中积分方程具备奇异性特点，在边界元数值计算程序编写前，需要对微分方程解的分布有大致的了解。进而，在进行边界元的划分时，避开存在奇异积分的区域。

6.2 方法

我们之前研究了静电场的微分方程，即 laplace 方程和 poisson 方程。在接下来，我们将从积分的角度描述电磁场理论，研究静电场和 Maxwell 方程组的积分形式，并且利用积分数值方法求解电磁场。在数学应用中，此类方法又被称作边界元方法（Boundary Element Method, BEM）。¹

本次课程的主要内容安排如下

1. 首先回顾的是 Maxwell 方程的积分形式
2. 给出微分方程的基本解和 Green 公式辅助理解边界元的思想
3. 讲解边界元的实施方法。牢记核心思想：为了求解微分方程，可首先计算边界积分方程的解，随后通过边界上电荷的分布计算区域内的场分布

¹此类积分的方法也被称作矩量法（Method of Moments, MoM），其原因我们稍后会讲。

4. 最后讲解一个二维边界元的编程示例，讨论 BEM 方法实施的全过程，方便同学们理解边界元的主要思想和求解步骤，并通过对数值误差和奇异性的讨论加深对电磁场中数值计算的理解

6.2.1 静电场的积分形式和基本解

在静电场中，根据 Poisson 公式，场内的电势 ϕ 与导体内部的自由电荷之间的关系为

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_0} \quad (6.1)$$

这个是 Poisson 公式的微分形式。

在自由空间中，Poisson 公式中的静电势 ϕ 可由自由空间中，所有静电荷 $q = \rho_v dV$ 的电势 $\phi(\mathbf{r}) = q/4\pi\epsilon_0|\mathbf{r} - \mathbf{r}'|$ 叠加而成，即

$$\phi(\mathbf{r}) = \int_V \frac{\rho(\mathbf{r}') dV'}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \quad (6.2)$$

那么，正向的去考虑，若已知电荷分布，我们就可计算得到场内任意一点的电势 ϕ ；反过来想，如果 ϕ 已知，那么上式即为电荷密度分布 ρ 的积分方程。

式(6.2)中还蕴含着一个 BEM 方法中的重要思想：即线性算子的叠加原理。其描述为，如果激励源是连续分量，那么它所产生的效应可以表示为无穷多个分量所产生效应的叠加。

这一问题很适用于我们在 FDM 方法中讲解的电容计算的例子。在那个例子中，表面的电势分布是已知的，在外导上 $\phi = \phi_{\text{spec}} = 0$ ，而在内导上 $\phi = \phi_{\text{spec}} = 1$ ，电荷仅仅分布在内导的表面。那么，除了用 FDM 求解 laplace 方程，我们也可以求解积分方程得到电荷的分布（注意此时积分为面积分）：

$$\int_S \frac{\rho_s(\mathbf{r}') dS'}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} = \phi_{\text{spec}}(\mathbf{r}) \quad (6.3)$$

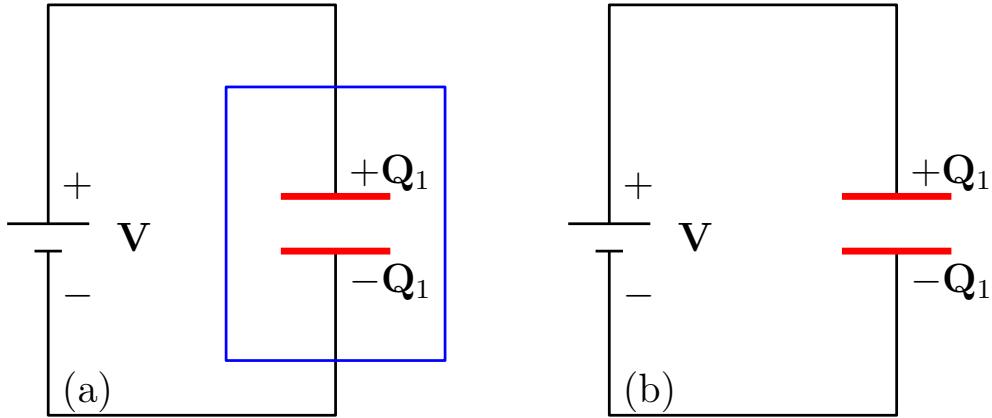
注意到三维的边界是面、而二维的边界是围线。因而在 2D 电磁场计算中，面积分转化为线积分，则我们可以使用线电荷（注意不是点电荷了）的电势分布作为积分项，

$$\int_S -\frac{\rho_l(\mathbf{r}') \ln |\mathbf{r} - \mathbf{r}'|}{2\pi\epsilon_0} dl' = \phi_{\text{spec}}(\mathbf{r}) \quad (6.4)$$

这样，我们就不经意间得到了静电场中 BEM 方法经典的积分公式。但是，我们还是需要以更数学化的方式，例如 Green 公式进行推导。类似的方法将被用于

推导 Maxwell 方程组的积分方程。

积分方法的好处是可以处理开放空间的数值计算问题。例如在 FDM 方法



需要在计算区域上附加一个封闭的导电框，即需要边界是固定的。而 BEM 则无需如此，可以方便的计算任意开放空间的电磁场问题。

6.2.2 基本解与 Green 函数

我们定义 $G(\mathbf{r}, \mathbf{r}')$ 为格林函数，代表位于 \mathbf{r}' 处的源，在 \mathbf{r} 处产生的场。在静电场中，格林函数就代表了位于 \mathbf{r}' 处的点电荷在 \mathbf{r} 处的电势。在三维空间中，格林函数为

$$G(\mathbf{r}, \mathbf{r}') = \frac{1}{4\pi\epsilon_0|\mathbf{r} - \mathbf{r}'|} \quad (6.5)$$

我们接下来将给同学们讲解，如何从 Poisson's equation 中得到格林公式。也就是说，格林函数是 Poisson 方程的基本解 (Fundamental solution)。基本解这一概念将为接下来，计算时谐场的积分方程做准备。

在三维空间中，点电荷满足 Poisson 方程

$$-\epsilon_0\nabla^2\varphi(\mathbf{r}) = \delta^3(\mathbf{r} - \mathbf{r}') \quad (6.6)$$

其物理意义为，对于包含点电荷的区域，波松方程不为 0；而对于不包含点电荷的区域，波松方程为 0。其中 $\delta^3(\mathbf{r} - \mathbf{r}')$ 代表三维中的狄拉克 (dirac delta) 函数，代表一个点电荷位于 \mathbf{r}' 处。其含义就不多说了，对于 $\mathbf{r}' = \mathbf{r}$ ，dirac 函数取值为无穷大，对于 $\mathbf{r} \neq \mathbf{r}'$ ，dirac 函数为零。那么我们对于 dirac 函数计算空域的积分，可得其空间积分为 $\int_V \delta^3(\mathbf{r} - \mathbf{r}') dV = 1$ 。

上式的基本解 (Fundamental Solution) 即为格林函数, 即

$$-\epsilon_0 \nabla_r^2 G(\mathbf{r}, \mathbf{r}') = \delta^3(\mathbf{r} - \mathbf{r}') \quad (6.7)$$

下标 r 代表仅对 \mathbf{r} 取 ∇ 计算。下面我们给出最简单的推导。

例题 6.1: (补充知识) 球坐标系下的 laplace 算子

$$\nabla^2 f = \frac{1}{r^2} \frac{\partial}{\partial r} r^2 \frac{\partial f}{\partial r} + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \sin \theta \frac{\partial f}{\partial \theta} + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 f}{\partial \phi^2}$$

根据对称性, 静电势仅与电荷的距离 $R = |\mathbf{r} - \mathbf{r}'|$ 相关, 因而, 除了在 $R = 0$ 处, G 都满足

$$-\frac{\epsilon_0}{R^2} \frac{d}{dR} R^2 \frac{dG}{dR} = 0, \quad R > 0 \quad (6.8)$$

式(6.8)的两个解分别为 $G_1 = a_1$ 和 $G_2 = a_2/R$ 。我们不关心 $G_1 = a_1$ 的解, 因为它不能生成电场。而对于 $G_2 = a_2/R$, 我们需要确定常量 a_2 。

a_2 可以通过对式(6.7)在源 \mathbf{r}' 为圆心、半径为 R_0 的球体上积分得到。其物理意义为, 我们划定一个封闭的曲面包围电荷, 积分即为计算通过曲面的电位移矢量 $\mathbf{D} = \epsilon \mathbf{E}$ 的通量。利用高斯定理, 积分式左边可写为

$$\begin{aligned} -\epsilon_0 \int_{R < R_0} \nabla \cdot \nabla G dV &= -\epsilon_0 \oint_{R=R_0} \nabla G \cdot \hat{\mathbf{n}} dS \\ &= -\epsilon_0 \left(-\frac{a_2}{R_0^2} \right) \cdot 4\pi R_0^2 = 4\pi \epsilon_0 a_2 \end{aligned} \quad (6.9)$$

因而其积分式需要等于式(6.7)右边的积分, 还记得右边的积分是多少么? 1。可得 $a_2 = 1/4\pi\epsilon_0$, 那么三维情形下, 静电场的基本解 (Green 函数) 为

$$G(\mathbf{r}, \mathbf{r}') = \frac{1}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \quad (6.10)$$

严格意义上说, 这里得到的 Green 函数仅对自由空间、且无边界的静电场问题有效。

在导体中, 电荷仅存在于导体表面, 则场内任意一点的电势可按下式计算

$$\phi(\mathbf{r}) = \int_{\text{Conductors}} G(\mathbf{r}, \mathbf{r}') \rho_s(\mathbf{r}') dS' \quad (6.11)$$

这两类等式 (微分和积分方程) 分别被称作, Poisson 公式 (6.1) 和 Coulomb 公式(6.2)或式(6.11), 两式在本质上是等价的。

为了证明这一点，我们对式(6.11)加入 laplace 算子，然后利用 $\nabla_r^2 G(\mathbf{r}, \mathbf{r}') = -\frac{1}{\epsilon_0} \delta^3(\mathbf{r} - \mathbf{r}')$ 可得

$$\begin{aligned}\nabla_r^2 \int G(\mathbf{r}, \mathbf{r}') \rho(\mathbf{r}') dV' &= \int [\nabla_r^2 G(\mathbf{r}, \mathbf{r}')] \rho(\mathbf{r}') dV' \\ &= -\frac{1}{\epsilon_0} \int \delta^3(\mathbf{r} - \mathbf{r}') \rho(\mathbf{r}') dV' \\ &= -\frac{\rho(\mathbf{r})}{\epsilon_0}\end{aligned}$$

即 Coulomb 积分式(6.11)或式(6.2)等价为 Poisson 公式。

6.2.3 BEM 的通用形式：直接计算边界积分方程

先前的内容中，我们将静电场的计算写成了积分等式的形式。接下来，我们将该方法扩展到更为通用的问题中。

对于一个微分方程

$$\mathbf{Df} = \mathbf{s} \quad (6.12)$$

其中 \mathbf{D} 是一个微分 (differential) 算子， \mathbf{f} 是场强， \mathbf{s} 是源的分布。令 $G(\mathbf{r}, \mathbf{r}')$ 位于 \mathbf{r} 位于 \mathbf{r}' 的源在 \mathbf{r} 处的场强，即 $G(\mathbf{r}, \mathbf{r}')$ 满足

$$\mathbf{D}G(\mathbf{r}, \mathbf{r}') = \delta^3(\mathbf{r} - \mathbf{r}') \quad (6.13)$$

根据线性系统的叠加原理 (principle of superposition)，我们可将微分形式写为积分形式

$$f(\mathbf{r}) = \int G(\mathbf{r}, \mathbf{r}') s(\mathbf{r}') dV' \quad (6.14)$$

通过替换可知，式(6.14)为式(6.12)的解。

这一通用形式适用于已知电荷分布、且电荷位于某一特定区域（例如导体表面）的计算，同时也适用于计算开区域的情形。这也是积分方法相对于差分方法的一个明显优势。

6.2.4 实施方法：有限元与加权余量法

我们可以仿照伽辽金有限元的思想：通过离散化边界元并且利用加权余量的思想，构造并求解 BEM 积分方程组。下面简要的叙述 BEM 的实施方法和计算过程。

6.2.4.1 基函数

每个有限元上的电荷分布可以用基函数 $s_k(\mathbf{r})$ 表示，第 k 个元上的电荷可写作

$$\rho_k = a_k s_k(\mathbf{r})$$

总电荷量为 $\rho_s = \sum_{k=1}^N \rho_k$ 。

在 BEM 方法发展早期，人们倾向于选择全局的基函数（及不具备局部化特征的基函数），然后通过对电磁场数值解的分析来寻找精确地、尽可能少的（有时甚至仅为一个）基函数。现在，常用的做法是将电荷所在区域划分成小的、局部的元，随后使用局部化的基函数（还记得伽辽金有限元中基函数的样子么？）进行求解。局部化基函数这一思路具备较强的通用性。

为了表示的方便，我们定义一个基函数 $s_k(\mathbf{r})$ 的电势为

$$\phi_k(\mathbf{r}) = \int G(\mathbf{r}, \mathbf{r}') s_k(\mathbf{r}') dS' \quad (6.15)$$

随后，由局部基函数近似计算得到的电势为

$$\bar{\phi}(\mathbf{r}) = \int \sum_{k=1}^N a_k \phi_k(\mathbf{r}) \quad (6.16)$$

6.2.4.2 实施方法

我们希望计算得到的 $\bar{\phi}$ 满足数值计算问题的边界条件 $\bar{\phi} = \phi_{\text{spec}}$ ，即可以最小化余量 $\mathbf{r} = \sum_k a_k \phi_k - \phi_{\text{spec}}$ 。现有两种方法最小化余量 \mathbf{r}

- 点匹配。这一方法也被称作配点法（collocation）或 Nystrom 方法。选择与基函数数目相同的一组测试点（testing points） $\mathbf{r}_j, j = 1, 2, \dots, N$ ，随后令

$$\bar{\phi}(\mathbf{r}_j) = \phi_{\text{spec}}(\mathbf{r}_j), \quad j = 1, 2, \dots, N \quad (6.17)$$

为了使得配点法正常工作，配点的位置应当选择在对应基函数作用范围内。若这一点无法满足，则存在某一基函数无法被观测到的情况，导致较大的数值误差。实际中，我们往往选择分段平滑的基函数，并且将配点置于基函数的中心位置，见图6.1。

○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○

图 6.1 2D BEM 方法求解静电场数值问题的基函数和配点。电荷密度可以表示为分段平滑的形式，配点（○）可置于每一个面元的中心处。

- 加权余量。选择权函数 $\omega_j, j = 1, 2, \dots, N$ ，使得

$$\int_{\text{conductor}} \omega_j(\mathbf{r}) [\bar{\phi}(\mathbf{r}) - \phi_{\text{spec}}(\mathbf{r})] dS = 0 \quad (6.18)$$

这里利用伽辽金 (Galerkin) 有限元可令 $\omega_j(\mathbf{r}) = s_j(\mathbf{r})$ 。这里如果我们使用传统的、全局基函数，那么伽辽金有限元等价为最小化电势的失配量。这也是为什么在电磁场数值计算中，BEM 方法也常常被称作矩量法 MoM 的原因。而配点法，则等价为将测试函数选择为 delta 函数 $\omega_j(\mathbf{r}) = \delta(\mathbf{r} - \mathbf{r}_j)$ 。

式(6.17)和式(6.18)中的积分常用数值的方法进行计算。但是，Green 函数在 $\mathbf{r} = \mathbf{r}'$ 处存在奇异值，需要特别的注意。

这两种方法，配点法和加权余量法，都可得到一组 $N \times N$ 的方程组

$$\sum_{k=1}^N A_{jk} a_k = b_j, \quad j = 1, 2, \dots, N,$$

$$A_{jk} = \int \omega_j(\mathbf{r}) \phi_k(\mathbf{r}) dS = \int dS \omega_j(\mathbf{r}) \int dS' G(\mathbf{r}, \mathbf{r}') s_k(\mathbf{r}') \quad (6.19)$$

$$b_j = \int \omega_j(\mathbf{r}) \phi_{\text{spec}}(\mathbf{r}) dS \quad (6.20)$$

对于共轭的 Poisson 方程，Green 函数是对称的，即 $G(\mathbf{r}, \mathbf{r}') = G(\mathbf{r}', \mathbf{r})$ 。这

里，如果采用伽辽金有限元推导边界元的方程组，那么矩阵 A 也会是对称的，即 $A_{jk} = A_{kj}$ 。

这里再强调一下 BEM 方法的解题思路：即利用配点法或者加权余量法，首先构造（数值或解析的方法计算）出 A ，利用边界条件得到 b 。通过计算系数分布 a 可得边界电荷的分布，对于第 k 个边界元，有 $\rho_k = a_k s_k(\mathbf{r})$ 。最后，利用电荷分布和格林公式去计算任何（开放）区域的电势分布

$$\begin{aligned}\phi_k(\mathbf{r}) &= \int G(\mathbf{r}, \mathbf{r}') s_k(\mathbf{r}') dS' \\ \bar{\phi}(\mathbf{r}) &= \int \sum_{k=1}^N a_k \phi_k(\mathbf{r})\end{aligned}$$

并且在这一数值计算过程中，我们还可计算电磁参量。

6.2.5 实施步骤

边界元数值计算的实施步骤如下：

1. **有限元：**将边界 S （在二维计算问题中，边界为曲线 L ）离散成一系列的边界元，在每个单元上，都假定位数函数和电荷密度为常数。注意：我们仅仅针对最简单的情形，不去讨论更为复杂的线性、二次或高次的基函数（或称作内插函数）
2. **计算方法：**选择 BEM 数值计算实施方法：配点法（推荐）及伽辽金法。
写出系数矩阵 A_{jk} 和 b_j 的边界积分形式
3. **系数矩阵：**基于边界积分方程，按照边界节点参数计算系数矩阵 A 和 b 。在计算边界积分时，既可以采用解析计算的方法，也可以采用我们先前学习过的数值积分方法进行计算
4. **方程求解：**按照给定的边界条件确定边界元方程。求解线性方程组得到电荷密度分布 \mathbf{r}
5. **反演位势函数：**同样基于电势函数 ϕ 的积分方程，在离散解的基础上可得到场域内任何一点的位函数与场量解
6. **电磁参量：**根据数值计算结果，计算电磁参数

接下来我们将针对 2D BEM 数值计算问题，给出一个直观简单的计算实例。三维边界元方法思路相同，但由于此时离散的边界元为平面或曲面，处理和数值计算过程较为复杂，不展开讨论。

6.3 计算示例

我们讲解一个简单的计算实例，来展示 BEM 方法优点，以及如何计算边界方程中的系数矩阵 A 与 b 。

例题 6.2：计算等长平行放置的两导电板之间单位长度的电容 C 。见图 6.2。

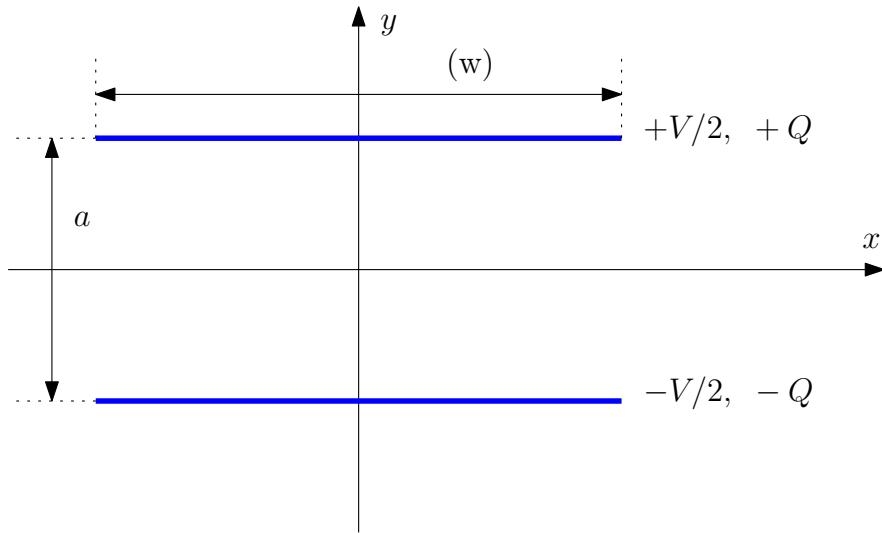


图 6.2 Capacitor 的截面形状

我们设线电荷密度（注意不是面电荷或体电荷咯）为 ρ_l （单位是库伦 / 米），则在 $\mathbf{r}' = (x', y')$ 处的电势为

$$\phi = -\frac{\rho_l}{2\pi\epsilon_0 r_0} \ln |\mathbf{r} - \mathbf{r}'| \quad (6.21)$$

其中 r_0 是一常量。那么对于平行放置的导电板，可得

$$\begin{aligned} \phi(x, y) = & -\frac{1}{2\pi\epsilon_0} \int_{-w/2}^{w/2} \rho_s(x', \frac{a}{2}) \ln \sqrt{(x - x')^2 + \left(y - \frac{a}{2}\right)^2} dx' \\ & -\frac{1}{2\pi\epsilon_0} \int_{-w/2}^{w/2} \rho_s(x', -\frac{a}{2}) \ln \sqrt{(x - x')^2 + \left(y + \frac{a}{2}\right)^2} dx' \end{aligned} \quad (6.22)$$

这一问题具备对称性，其中左右是对称

$$\rho_s(x', a/2) = \rho_s(-x', a/2)$$

而上下是反对称的

$$\rho_s(x', -a/2) = -\rho_s(x', a/2)$$

6.3.1 解析积分方法

我们将电容板划分为元 $x' \in [x_i, x_{i+1}]$ 并且使用分段平滑的基函数来近似电荷密度。测试方法选取为配点法，且配点的位置选择在每一个元的中心位置 $x_{\text{test},i} = x_{i+\frac{1}{2}} = \frac{1}{2}(x_i + x_{i+1})$ 。这种方法可以得到与基函数较好的耦合性，即在每一个基函数上都是恒定的，且对应的测试点上的值也为恒定。与之对应的是，若我们将测试点选择在端点上，那么当相邻元的电荷密度极性相同时，测试点就能很好的反映这一电势变化。因为当两个相邻面元电荷量相同时，配点法 $x_{i+\frac{1}{2}} = \frac{1}{2}(x_i + x_{i+1})$ 会在端点位置上互相抵消。

为了得到分段恒定电荷分布的电势，我们就可以用积分方法进行计算。当观测点位于面元上时，核函数（格林函数）的奇异性导致积分值不稳定。但是当采用分段恒定的基函数时，我们可得到积分的解析解

$$\begin{aligned} I(x_s, x_e, d) &= -\frac{1}{2\pi\epsilon_0} \int_{x_s}^{x_e} \ln \sqrt{x^2 + d^2} dx \\ &= -\frac{1}{2\pi\epsilon_0} \left[\frac{1}{2}x \ln(x^2 + d^2) - x + d \arctan(x/d) \right]_{x_s}^{x_e} \end{aligned} \quad (6.23)$$

其中下标 s 和 e 分别代表积分限的 start 和 end。我们用解析积分法计算系数矩阵。

同时，如果利用问题的对称性，我们还可进一步减少计算量。在这个例题中，上下是反对称的，而左右则是对称的，那么

$$\begin{aligned} \phi(x, y) &= \sum_{i=0}^{N-1} \rho_{i+\frac{1}{2}} [I(x_i - x, x_{i+1} - x, y - a/2) \\ &\quad + I(-x_{i+1} + x, -x_i + x, y - a/2) \\ &\quad - I(x_i - x, x_{i+1} - x, y + a/2) \\ &\quad - I(-x_{i+1} + x, -x_i + x, y + a/2)] \end{aligned} \quad (6.24)$$

通过在上 $1/4$ 平板区域中选择测试点 $x_{i+\frac{1}{2}}, i = 0, 1, \dots, N-1$ ，我们就可得到

BEM 的线性方程组

$$\mathbf{A}\mathbf{r} = \mathbf{v}$$

其中

$$\begin{aligned} A_{ij} = & I(x_j - x_{i+\frac{1}{2}}, x_{j+1} - x_{i+\frac{1}{2}}, 0) \\ & + I(-x_{j+1} + x_{i+\frac{1}{2}}, -x_j + x_{i+\frac{1}{2}}, 0) \\ & - I(x_j - x_{i+\frac{1}{2}}, x_{j+1} - x_{i+\frac{1}{2}}, a) \\ & - I(-x_{j+1} + x_{i+\frac{1}{2}}, -x_{j+1} + x_{i+\frac{1}{2}}, a) \end{aligned}$$

而 v 为上导电板的电势 $V/2$ 。求解这一线性方程组，即可得到每一面元的电荷密度 \mathbf{r} 。

6.3.2 2D 边界元法的通用计算程序

在上面这个例子中，我们讲解了如何计算一个规则的、简单的 BEM 问题。同样的，边界元方法可以方便的扩展到更为通用的数值计算问题上。

图6.3给出了一段面元和一个观测点（observation point），其中观测点位于面元延长线的法线方向距离 d 处。面元对于该观测点的电势贡献为

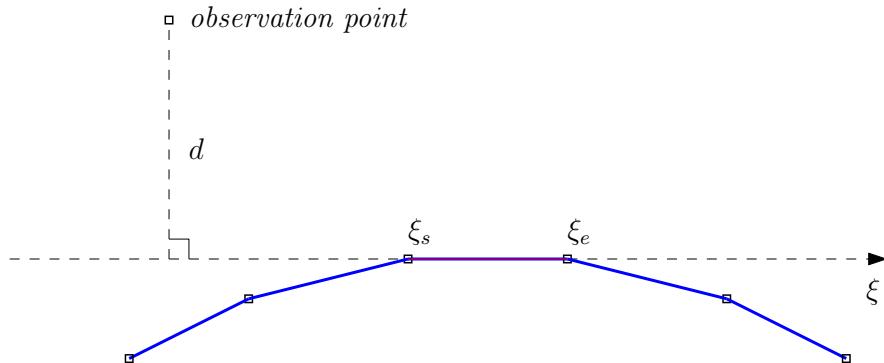


图 6.3 复杂形状的 2D BEM，图中坐标 ξ 旋转后与第 i 个面元对齐。

$$-\frac{\rho_s}{2\pi\epsilon_0} \int_{\xi_s}^{\xi_e} \ln \sqrt{x^2 + d^2} dx = -\frac{\rho_s}{2\pi\epsilon_0} \left[\frac{1}{2} x \ln(x^2 + d^2) - x + d \arctan(x/d) \right]_{\xi_s}^{\xi_e}$$

2D BEM 编程的关键在于如何计算局部坐标系的 ξ_s 和 ξ_e ，以及第 i 个面元的观测点到第 j 个面元的法向距离 d 。见图6.4。原理很简单，即

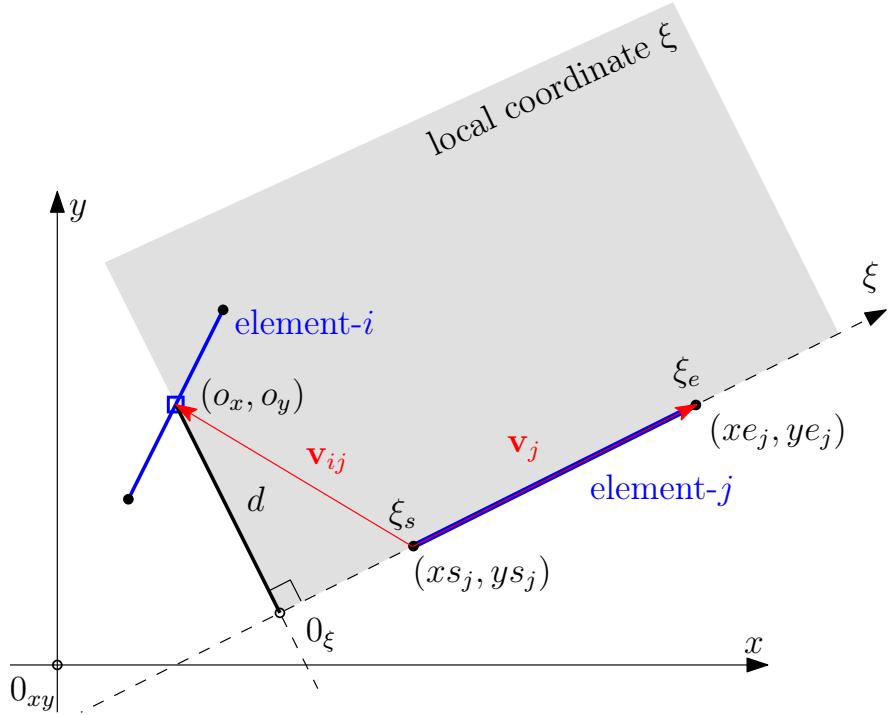


图 6.4 2D 通用 BEM 计算程序实施步骤。首先利用 xy 坐标系计算局部坐标系下的 ξ_s 、 ξ_e 、 d ，随后即可利用有限元积分方程组装系数矩阵 A_{ij} ，最后利用 $\mathbf{r} = \mathbf{A} \setminus \mathbf{v}$ 求解线性方程组，计算面元上的电荷密度 \mathbf{r} 。那么，密度乘以面元长度 (re) 即为面元所带电荷量 Q_e 。

$$h = \|\mathbf{v}_j\| \quad (6.25)$$

$$s = \frac{\langle \mathbf{v}_{ij}, \mathbf{v}_j \rangle}{\|\mathbf{v}_j\|^2} \quad (6.26)$$

$$\xi_s = -sh \quad (6.27)$$

$$\xi_e = \xi_s + h = (1 - s)h \quad (6.28)$$

$$d = \sqrt{\|\mathbf{v}_{ij}\| - (sh)^2} \quad (6.29)$$

其中 h 为面元 j 的长度， s 为向量 \mathbf{v}_{ij} 在 \mathbf{v}_j 定义的局部坐标系中的单位投影。那么利用 h 和 s 即可分别得到局部坐标系 ξ 中积分限的起始坐标 ξ_s 和终点坐标 ξ_e ，同时也可以方便的计算得到 d 。

接下来，我们将用这种解析计算的方法生成系数矩阵，用来求解二维的边界元问题。每一个带电荷的元 (element) 都用 xs 和 ys 来表示起始点的坐标，用 xe 和 ye 表示终止点的坐标，而 ϕ 代表电势。注意到在程序中没有假设数值计算中的几何形状信息：程序对于任何二维 BEM 数值计算都是适用的。

此处插入子程序：MoM2D

利用 MoM2D 即可得到每一个面元上的电荷量，随后我们可以

1. 积分得到单位长度上的电容；

$$Q = \int_{\text{Conductors}} \rho_s(\mathbf{r}') dS'$$

2. 分别由面元点电荷计算开区域中的电势分布

$$\phi(\mathbf{r}) = \int_{\text{Conductors}} G(\mathbf{r}, \mathbf{r}') \rho_s(\mathbf{r}') dS'$$

我们再回过头研究本章中的例子。我们设上导体板的电压为 0.5V，下导体板的电压为 -0.5V，那么两板间的电容就是上板的电荷数。数值计算的方法很简单：

此处插入主程序：Demo

实验结果见图6.5。

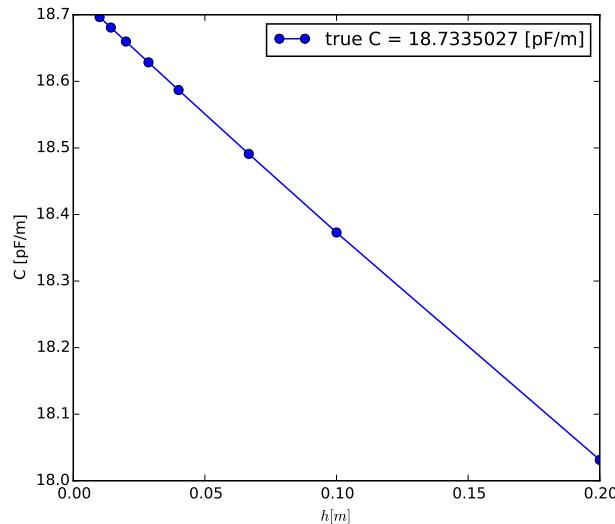


图 6.5 例题中 $a = 1$ 及 $w = 1$ 、均匀边界元、解析积分计算得到的 BEM 数值计算结果

其中真值为 $C = 18.7335027 \text{ pF/m}$ 。为了达到小于 1% 的数值误差，至少需要的边界元数目为 $n = 50$ 。利用数据拟合进行外推可得：2 阶拟合值为 $C_{h_0} = 18.733500 \text{ pF/m}$ ，而 3 阶拟合为 $C_{h_0} = 18.733503 \text{ pF/m}$ 。

6.3.3 BEM 数值方法的奇异性

我们将上导电板的电荷分布 ($n = 20$) 绘于图6.6。可以明显的看出，BEM

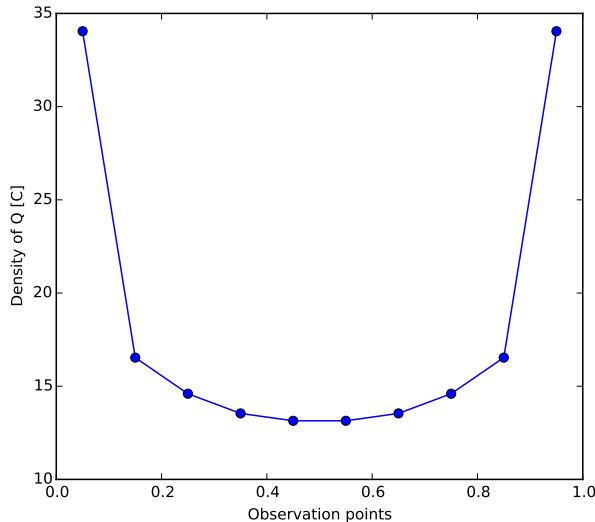


图 6.6 上导电板的电荷分布。其中 $n = 20$ ，电容误差为 1.92%。

计算出的电荷分布在导电板端点附近有较大奇异性 (singular)，存在幅度跳变。这一奇异性可以很方便的用解析的方法进行分析和解释。

我们考虑开放区域中一个角点的静电场数值计算问题，即图6.7中所示的 2D 角附近的区域。

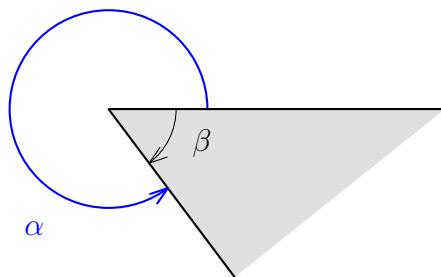


图 6.7 2D BEM 方法的奇异性分析

假设导体的内角 $\beta < 180^\circ$ ，在真空区域中电势的包角 (subtends an angle) $\alpha = 360^\circ - \beta > \pi$ 。在圆柱坐标系下，laplace 算子 ∇^2 为

$$\nabla^2 f = \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial f}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 f}{\partial \theta^2} + \frac{\partial^2 f}{\partial z^2}$$

而 2D 平面区域等价为 z 轴无限长，则 laplace 方程为

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial f}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 f}{\partial \theta^2} = 0, \quad 0 < \theta < \alpha \quad (6.30)$$

其中对于 $\theta = 0$ 或 $\theta = \alpha$, 有 $\phi = 0$ 。

我们可以用分离变量法 (separation of variables) 得到解析解。令 $\phi(r, \theta) = f(r)g(\theta)$, 代入式(6.30)中, 并且在等式两边乘以 $r^2/(f(r)g(\theta))$, 可得

$$\frac{r(rf'(r))'}{f(r)} = \frac{g''(\theta)}{g(\theta)}$$

由于左边仅仅包含 r , 而右边仅包含 θ , 因而左右两边的结果必定为常数, 我们令其等于 p^2 。那么可得 $g(\theta) = a \sin p\theta + b \cos p\theta$ 以及 $f(r) = cr^p + dr^{-p}$ 。若 $p > 0$, 为了使得 $r \rightarrow 0$ 时电势 ϕ 的解是有界的, 需令 $d = 0$ 。则我们可得电势分布的解析解为

$$\phi = (a \sin p\theta + b \cos p\theta)r^p$$

下一步我们就需要确定 p 的取值。利用边界条件, 当 $\theta = 0$ 时 $\phi = 0$, 可得 $b = 0$; 当 $\theta = \alpha$ 时 $\phi = 0$, 可得 $p\alpha = n\pi$ 其中 n 为整数。那么最低阶的解 ($n = 1$) 为

$$\phi = r^p \sin p\theta, \quad p = \pi/\alpha$$

对于开区域的角 α , 若 $\alpha > \pi$, 则 p 的最小值小于 1。那么根据 $E = -\nabla\phi$, 电场 E_r 或者 E_θ 就会随 $r^{(-1+\pi/\alpha)}$ 变化。那么可以明显的看出, 当在角部进行数值计算且 $\alpha > \pi$ 时, 随着 r 的减小, 电场 E 的取值近似为无穷大, 进而导致数值结果的奇异性。

在本例题中, $\alpha = 2\pi$, 那么 $E_\theta \propto r^{-1/2}$, 这也就意味着电荷量将随着距离 r 以 $r^{-1/2}$ 的趋势变化。同学们可以类比下避雷针开心开心就行。

6.3.4 BEM 自适应边界元划分

在电荷分布中, 我们可以看出导电板中心部分电荷量少, 而在边界上电荷量较大。也就是说, 我们应当在电荷密度大的区域划分较为密的元, 而在电荷密度低的区域划分较为稀疏的元。

除此以外, 还可以通过迭代计算的方法, 首先采用均匀网格划分, 随后根据 BEM 计算得到的电荷量进行面元的自适应 (adaptivity) 调整, 电荷量大的面

元进行细分，而电荷量小的面元进行合并，进而大致的保证每个面元所带电荷量是近似一样的。同学们可以尝试自己编写 / 或修改此类计算程序，在本课程中不讲解。

在自适应中关键的部分是需要理解数值计算的奇异区域，并且在奇异区域进行自适应的元调整。这一点需要同学们了解所计算区域的几何形状，以及奇异性相关知识。在这一例子中，我们可设定在端点区域进行较密的划分。2D BEM 程序计算结果见图6.8。采用自适应边界元的计算结果为

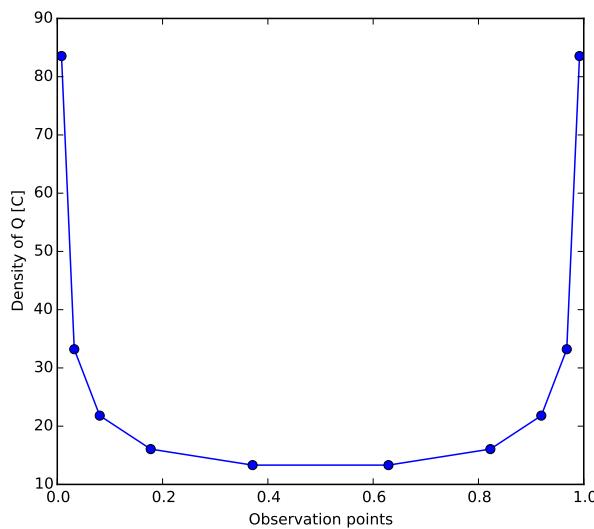


图 6.8 自适应边界元划分的计算结果。其中边界元在端点处间隔小、密度大，在导电平板的中心区域间隔较大。

$C = 18.668327 \text{ pF/M}$ ，精度可达到 0.348%。采用自适应的边界元可以具备较少的计算量和较高的精度，但是自适应边界元的划分不可避免的会存在一个问题：即数据无法方便的进行外推计算。而采用均匀边界元方法，则可方便的外推数据，逼近真实结果。

6.3.5 数值积分方法

我们在 BEM 程序中使用了解析计算线电荷势函数积分的方法。在前面的学习内容中，我们掌握了数值积分这一方法。同样类似于有限元中的内容，我们可以对边界元系数矩阵的计算采用数值积分的方法。但是在 BEM 中，需要谨慎的处理积分的奇异性。在二维边界元问题中，即处理对数函数的奇异性 (logarithmic singularity)。

定义 x 为边界元的中点，我们可以

- 中值积分法 (Midpoint integration):

$$\int_{x-h/2}^{x+h/2} f(x')dx' \approx hf(x)$$

这一数值积分法对于自观测 (self observation)，即观测点位于当前面元上时，具有较大的数值误差。

- 梯形法 (Trapezoidal rule):

$$\int_{x-h/2}^{x+h/2} f(x')dx' \approx \frac{1}{2}h [f(x-h/2) + f(x+h/2)]$$

虽然该方法具备 $O(h^2)$ 的收敛性，但是对于 $f(x) = \ln x$ 的数值积分计算有较大的误差。

- 高斯积分 (Gaussian integration):

$$\int_{x-h/2}^{x+h/2} f(x')dx' \approx \frac{1}{2}h [f(x_1) + f(x_2)]$$

其中 $x_{1,2} = x \pm (h/2)/\sqrt{3}$ 。高斯方法具备 $O(h^4)$ 的收敛性，但是同样的，对于 $f(x) = \ln x$ 的积分有较大的误差。

- 针对对数函数的特殊数值积分方法:

$$\int_{x-h/2}^{x+h/2} f(x')dx' \approx \frac{1}{2}h [f(x_1) + f(x_2)], \quad x_{1,2} = x \pm (h/2)/e$$

对于一般的函数，其收敛性为 $O(h^2)$ 。而对于 $f(x) = \ln x$ 该方法却可得到精确的积分结果，其中 $e = 2.7182818284$ 。

采用自适应边界元 $n = 20$ ，利用数值积分方法得到的电容值为 $C = 18.800158\text{pF/m}$ ，精度为 0.355%。而采用均匀边界元，解析积分法和数值积分法所得电容值可见图6.9。从图中可以看出，数值积分法并不能完全等于解析积分法，同样存在着数值误差。且数值积分法对于端点位置处 Green 函数的积分同样存在着奇异性，影响数值计算精度。

6.4 边界元背后的数学知识

本节内容参考了 Tara LaForce[14] 课程的内容。

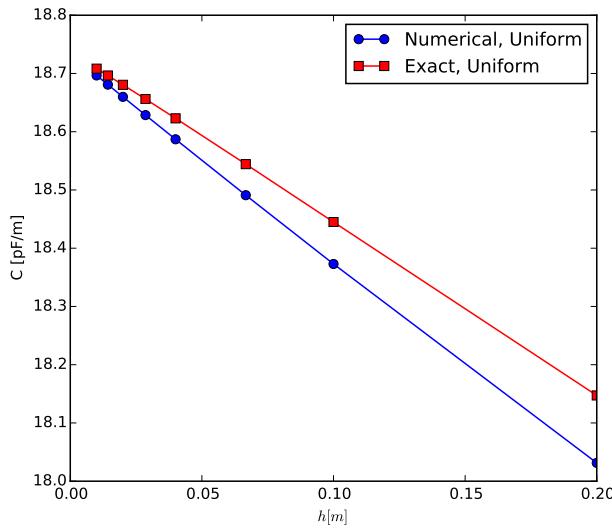


图 6.9 采用均匀边界元，解析积分法与数值积分法计算所得电容值

6.4.1 微分算子示例

我们令微分方程 $Df = s$ 中的微分算子 $D = \frac{d^2}{dx^2} + 1$ ，设区间为 $[0, 1]$ ，则 $D(f) = \frac{d^2 f}{dx^2} + f$ 。假设存在任意一个测试函数 ω ，有

$$\begin{aligned} \int_0^1 D(f)\omega dx &= \int_0^1 \frac{d^2 f}{dx^2} \omega + f\omega dx \\ &= \frac{df}{dx}\omega|_0^1 - \int_0^1 \frac{df}{dx} \frac{d\omega}{dx} dx + \int_0^1 f\omega dx \\ &= \frac{df}{dx}\omega|_0^1 - \frac{d\omega}{dx} f|_0^1 + \int_0^1 \frac{d^2 \omega}{dx^2} f dx + \int_0^1 f\omega dx \\ &= \text{boundary terms} + \int_{\Omega} f D(\omega) d\Omega \end{aligned}$$

由于 ω 测试函数可任意选取，则我们令 ω 和 $d\omega/dx$ 在边界 $x = 0$ 和 $x = 1$ 上为 0，则上式中边界项为 0，可写作

$$\int_0^1 D(f)\omega d\Omega = \int_0^1 f D(\omega) d\Omega$$

同时该式也意味着微分算子 D 是自伴随算子 (self-adjoint operator)。

6.4.2 加权余量法与格林公式

我们考虑 Laplace 方程 $\nabla^2 u = 0$, 采用加权余量法, 可得

$$\int_{\Omega} \nabla^2 u \omega d\Omega = 0$$

需要注意的是, 对于泊松方程 $\nabla^2 u = \rho_s$, 推导过程是类似的。BEM 方法可以适用求解任何具备基本解的微分方程组。

一维的分部积分公式 $(u' \omega)' = u'' \omega + u' \omega'$ 在更高维的空间被称作 Green-Gauss 定理, 仿照一维情形我们可得

$$\int_{\Omega} \nabla^2 u \omega d\Omega = \int_{\partial\Omega} \frac{\partial u}{\partial n} \omega dS - \int_{\Omega} \nabla u \nabla \omega d\Omega \quad (6.31)$$

继续对第二部分应用分部积分公式可得

$$\int_{\Omega} \nabla^2 u \omega d\Omega = \int_{\partial\Omega} \frac{\partial u}{\partial n} \omega dS - \int_{\partial\Omega} \frac{\partial \omega}{\partial n} u dS + \int_{\Omega} u \nabla^2 \omega d\Omega \quad (6.32)$$

式(6.32)被称作格林第二公式, 亦称为格林定理。

6.4.3 边界积分方程

再次回想前面章节的内容, 我们得到 Green 函数是微分方程的基本解 (我们忽略了 ϵ_0 项)

$$\nabla_r^2 G(\mathbf{r}, \mathbf{r}') = -\delta^3(\mathbf{r} - \mathbf{r}')$$

在 BEM 方法中, 可将测试函数 (test function, 也被称作权函数 Weighting function) 选为格林函数, 则

$$\int_{\Omega} u \nabla^2 \omega d\Omega = \int_{\Omega} -u \delta(\mathbf{r} - \mathbf{r}') d\Omega = -u(\mathbf{r}) \quad (6.33)$$

其中 $\mathbf{r} \in \Omega$ 且 $\mathbf{r} \notin \partial\Omega$ 。

利用加权余量法, 可得边界积分方程 (Boundary Integral Equation, BIE)

$$u(\mathbf{r}) + \int_{\partial\Omega} u \frac{\partial \omega}{\partial n} dS = \int_{\partial\Omega} \omega \frac{\partial u}{\partial n} dS \quad (6.34)$$

式(6.34)说明了我们可以通过计算边界 $\partial\Omega$ 上的 u 和 ω , 即可得到区域 Ω 内 u 的分布。但是为了求解 BEM 方程, 必须知道边界上的 u 或 $\partial u / \partial n$ (即电荷 q)。但对于 $u(\mathbf{r})$ 的取值, 需要分情况讨论。

1. 当源点 $\mathbf{r}' \notin \Omega$ 时。

此时由于 $\int_{\Omega} \delta(\mathbf{r} - \mathbf{r}') = 0$, 则式(6.34)中 $u(\mathbf{r}) = 0$ 。

2. 当源点 $\mathbf{r}' \in \partial\Omega$ 时。

见图6.10。其中当源点 \mathbf{r}' 位于边界上时, 由于要分别计算 $\partial\omega/\partial n \propto 1/r$ 以及

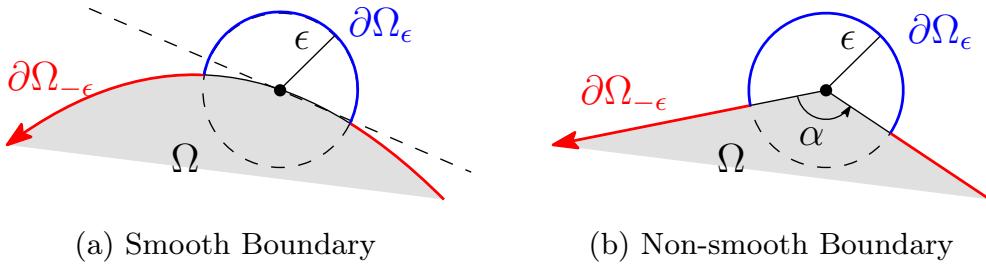


图 6.10 边界积分方程组中 $\mathbf{r}' \in \partial\Omega$ 的情况

$\omega \propto \ln 1/r$ 的积分, 这两个分别是严格奇异的以及弱奇异 (weakly singular) 的, 无法通过直接积分计算得到。因此我们可采用柯西主值积分 (cauchy principal value) 方法, 即将 Ω 扩展成 Ω' 。 Ω' 包含了源点 \mathbf{r}' 上一个半径为 ϵ 的圆, 而 $\partial\Omega'$ 包含了由 $\partial\Omega_\epsilon$ 和 $\partial\Omega_{-\epsilon}$ 构成的新的边界。此时由于源点 \mathbf{r}' 位于新的积分区域 Ω' 内, 式(6.34)可写为

$$u(\mathbf{r}) + \int_{\partial\Omega_\epsilon \cup \partial\Omega_{-\epsilon}} u \frac{\partial\omega}{\partial n} dS = \int_{\partial\Omega_\epsilon \cup \partial\Omega_{-\epsilon}} \omega \frac{\partial u}{\partial n} dS$$

我们计算 $\epsilon \rightarrow 0$ 的主值积分。

式(6.34)中的积分项可分为四部分。首先,

$$\begin{aligned} \int_{\partial\Omega_\epsilon} u \frac{\partial\omega}{\partial n} dS &= \int_{\partial\Omega_\epsilon} u \frac{\partial}{\partial n} \left(-\frac{1}{2\pi} \ln r \right) dS = \int_{\partial\Omega_\epsilon} \frac{-u}{2\pi r} dS \\ &= \frac{-1}{2\pi r} \int_0^{2\pi-\alpha} u d\theta = -u(\mathbf{r}) \frac{(2\pi-\alpha)\epsilon}{2\pi r} = -u(\mathbf{r}) \left(1 - \frac{\alpha}{2\pi} \right) \end{aligned}$$

其中 $r = \epsilon$ 。当 $\epsilon \rightarrow 0$ 时, 对于平滑的边界有 $\alpha = \pi$, 而对于不平滑的边界, α 设定为导体的内角。

另一方面

$$\lim_{\epsilon \rightarrow 0} \int_{\partial\Omega_\epsilon} \omega \frac{\partial u}{\partial n} dS = \lim_{\epsilon \rightarrow 0} \frac{-\ln r}{2\pi} \frac{\partial u(\mathbf{r})}{\partial n} \alpha \epsilon = 0$$

而对于 $\partial\Omega_{-\epsilon}$, 有

$$\lim_{\epsilon \rightarrow 0} \int_{\partial\Omega_{-\epsilon}} \omega \frac{\partial u}{\partial n} dS = \int_{\partial\Omega} \omega \frac{\partial u}{\partial n} dS$$

$$\lim_{\epsilon \rightarrow 0} \int_{\partial\Omega_{-\epsilon}} u \frac{\partial \omega}{\partial n} dS = \int_{\partial\Omega} u \frac{\partial \omega}{\partial n} dS$$

则式(6.34)可写作

$$\left(\frac{\alpha}{2\pi}\right)u(\mathbf{r}) + \int_{\partial\Omega} \frac{\partial \omega}{\partial n} u dS = \int_{\partial\Omega} \frac{\partial u}{\partial n} \omega dS \quad (6.35)$$

考虑所有情况，式(6.34)可写作

$$c(\mathbf{r})u(\mathbf{r}) + \int_{\partial\Omega} \frac{\partial \omega}{\partial n} u dS = \int_{\partial\Omega} \frac{\partial u}{\partial n} \omega dS \quad (6.36)$$

其中

$$c(\mathbf{r}) = \begin{cases} 1 & \mathbf{r} \in \Omega \\ \frac{\alpha}{2\pi} & \mathbf{r} \in \partial\Omega \\ 0 & \mathbf{r} \notin \Omega \end{cases} \quad (6.37)$$

6.4.4 有限元与系数矩阵

我们以 2D 情形为例。BEM 方法基于有限元的思想，将 2D 区域划分为 N 个边界元，每一个边界元用 Γ_i 表示。我们假定

- 边界条件将给出 u_i 或者 $q_i = \frac{\partial u}{\partial n}$ ，采用边界元方法，当给出其中一个时，另一个可通过边界元方程解出
- 我们假定常数节点，即在边界元上 u 和 q 为常量
- 每个边界元的节点都位于元的中心处。当然我们也可类似有限元中的方法，选择线性基函数（两个节点），或者二次函数（三个节点），但是选择单节点是最简单、也是最易于理解的
- 我们假设边界是平滑的，即式(6.36)中 $c(\mathbf{r}) = 1/2$ ，同时令 $u_i = u(\mathbf{r})$

BEM 积分方程式(6.36)可写为

$$\begin{aligned} \frac{1}{2}u_i + \int_{\partial\Omega} \frac{\partial \omega}{\partial n} u dS &= \int_{\partial\Omega} \frac{\partial u}{\partial n} \omega dS \\ \frac{1}{2}u_i + \sum_{j=1}^N \left[\int_{\Gamma_j} u q^* d\Gamma \right] &= \sum_{j=1}^N \left[\int_{\Gamma_j} w q d\Gamma \right] \end{aligned} \quad (6.38)$$

其中 $q^* = \partial\omega/\partial n$ 。由于在边界元上 u_j 和 q_j 为常量，则

$$\frac{1}{2}u_i + \sum_{j=1}^N u_j \left[\int_{\Gamma_j} q^* d\Gamma \right] = \sum_{j=1}^N q_j \left[\int_{\Gamma_j} w d\Gamma \right] \quad (6.39)$$

则我们需要计算以下两个积分

$$\hat{H}_{ij} = \int_{\Gamma_j} q^* d\Gamma, \quad G_{ij} = \int_{\Gamma_j} w d\Gamma \quad (6.40)$$

为了简化表示，我们定义

$$H_{ij} = \begin{cases} \hat{H}_{ij}, & i \neq j \\ \hat{H}_{ij} + 0.5, & i = j \end{cases} \quad (6.41)$$

则我们可得到一组方程组

$$\sum_{j=1}^N u_j H_{ij} = \sum_{j=1}^N q_j G_{ij} \quad (6.42)$$

对于所有的边界元 $i = 1, 2, \dots, N$ 可得到 BEM 的矩阵形式

$$\mathbf{H}\mathbf{u} = \mathbf{G}\mathbf{q} \quad (6.43)$$

当边界条件是混合了 u 、 q 时，需要通过矩阵变换（或称作 shuffling）将未知参数移动到方程式的一边进行求解。例如，假定对于前 $N/2$ 个边界元电压是已知的，而对于后 $N/2$ 边界元 $\partial u / \partial n$ 是已知的，即 q 已知。我们将 BIE 写作

$$\mathbf{H}_1 \mathbf{u}_1 + \mathbf{H}_2 \mathbf{u}_2 = \mathbf{G}_1 \mathbf{q}_1 + \mathbf{G}_2 \mathbf{q}_2 \quad (6.44)$$

调换顺序可得

$$\mathbf{G}_1 \mathbf{q}_1 - \mathbf{H}_2 \mathbf{u}_2 = -\mathbf{H}_1 \mathbf{u}_1 + \mathbf{G}_2 \mathbf{q}_2 \quad (6.45)$$

其中 \mathbf{u}_1 、 \mathbf{q}_2 是已知的边界条件，而 \mathbf{q}_1 、 \mathbf{u}_2 是代求边界元参数，即

$$\begin{pmatrix} \mathbf{G}_1 & -\mathbf{H}_2 \end{pmatrix} \begin{pmatrix} \mathbf{q}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} -\mathbf{H}_1 & \mathbf{G}_2 \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{q}_2 \end{pmatrix} \quad (6.46)$$

可转化为 $\mathbf{Ax} = \mathbf{f}$ 的形式。

在解得边界上的 u 、 q 之后，我们可直接利用 Coulomb 公式

$$\phi(\mathbf{r}) = \int_{\text{Conductors}} G(\mathbf{r}, \mathbf{r}') \rho_s(\mathbf{r}') dS'$$

或者用 Green 公式

$$u_i = \int_{\Gamma} \frac{\partial u}{\partial n} \omega dS - \int_{\Gamma} u \frac{\partial \omega}{\partial n} dS$$

后者可采用矩阵的形式计算

$$u_i = \sum_{j=1}^N q_j G_{ij} - \sum_{j=1}^N u_j \hat{H}_{ij} \quad (6.47)$$

BEM 方法的方便之处在于，利用格林公式，我们可以计算电磁场在任意方向的通量（flux）

$$q_{ix} = \frac{\partial u}{\partial x} = \int_{\Gamma} \left(\frac{\partial \omega}{\partial x} \right)_i q d\Gamma - \int_{\Gamma} u \left(\frac{\partial q^*}{\partial x} \right)_i d\Gamma \quad (6.48)$$

其中计算过程中不包含 q 和 u 的导数。将上式写为有限元的格式，

$$q_{ix} = \sum_{j=1}^N q_j \int_{\Gamma_j} \left(\frac{\partial \omega}{\partial x} \right) d\Gamma - \sum_{j=1}^N u_j \int_{\Gamma_j} \left(\frac{\partial q^*}{\partial x} \right) d\Gamma \quad (6.49)$$

其中 $\partial q^*/\partial x$ 是混合型导数（ ω 对 x 和 n 分别求偏导）。同样 BEM 方法的缺点也很明显，在计算得到边界参数 u 或者 $\partial u/\partial n$ 之后，计算空间任意节点的分布需重新进行积分计算生成对应系数向量，运算量较大。

6.4.5 系数矩阵元素的确定

在之前的例题中，我们已知边界上的电势，求解边界元上的电荷分布。同时在 2D BEM 中采用了配点法构造积分方程。

当 ω 为基本解时，即线电荷的电势分布

$$\omega(r) = \frac{-1}{2\pi\epsilon_0} \ln r$$

则

$$G_{ij} = \int_{\Gamma_j} \omega d\Gamma$$

可用解析计算的方法得到。需要注意的是 G_{ij} 与本章我们利用 Green 函数得到的 BEM 方程组中的 A_{ij} 是一致的。

对于 \mathbf{H} ，我们需要计算 ω 在法线方向 n 上的偏导数，对于 $i = j$ 时

$$H_{ii} = \frac{1}{2} + \hat{H}_{ii} = \frac{1}{2} + \int_{\Gamma_j} \frac{\partial \omega}{\partial n} d\Gamma = \frac{1}{2} + 0$$

其中由于观测点在自身边界元上，法向导数在边界元的切线方向上分量为 0。

同样的，我们也可以解析的计算 H_{ij} 。回想一下之前的程序中，我们定义了观测点 i 到面元 Γ_j 的法向距离为 d ，那么

$$\begin{aligned}\hat{H}_{ij} &= \int_{\Gamma_j} \frac{\partial \omega}{\partial n} d\Gamma = \frac{-1}{2\pi\epsilon_0} \int_{\Gamma_j} \frac{\partial \ln \sqrt{x^2 + d^2}}{\partial d} dx \\ &= \frac{-1}{2\pi\epsilon_0} \int_{x_s}^{x_e} \frac{d}{x^2 + d^2} dx\end{aligned}\quad (6.50)$$

那么当 $i = j$ 时 $d = 0$ ，即 $\hat{H}_{ij} = 0$ 。而 $i \neq j$ 时，

$$\hat{H}_{ij} = \frac{-1}{2\pi\epsilon_0} \left[\arctan \frac{x}{d} \right]_{x_s}^{x_e} \quad (6.51)$$

当然，除了解析计算的方法，我们也可以使用数值积分的方法计算 \hat{H}_{ij} 。

针对前面的计算电容的例子，由于在端点处边界元不连续，我们令 $c_i = 0.99$ 。 $N = 20$ 个边界元设置下的数值计算结果为 $C = 18.473866\text{pF/m}$ ，误差为 1.38%。

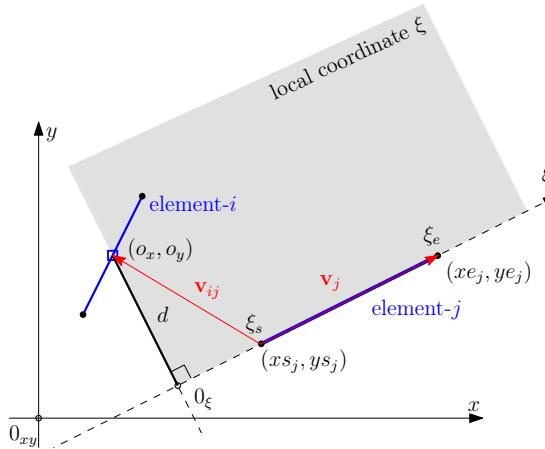
Recipe for implementing the BIE.

$$\omega(r) = \frac{-1}{2\pi\epsilon_0} \ln r$$

1. Exact G_{ij}

$$G_{ij} = \int_{\Gamma_j} \omega d\Gamma$$

$$\begin{aligned} G_{ij} &= -\frac{1}{2\pi\epsilon_0} \int_{x_s}^{x_e} \ln \sqrt{x^2 + d^2} dx \\ &= -\frac{1}{2\pi\epsilon_0} \left[\frac{1}{2} x \ln(x^2 + d^2) - x + d \arctan(x/d) \right]_{x_s}^{x_e} \end{aligned} \quad (6.52)$$



2. Exact H_{ij}

$$\begin{aligned} \hat{H}_{ij} &= \int_{\Gamma_j} \frac{\partial \omega}{\partial n} d\Gamma = \frac{-1}{2\pi\epsilon_0} \int_{\Gamma_j} \frac{\partial \ln \sqrt{x^2 + d^2}}{\partial d} dx \\ &= \frac{-1}{2\pi\epsilon_0} \int_{x_s}^{x_e} \frac{d}{x^2 + d^2} dx \end{aligned} \quad (6.53)$$

$$\hat{H}_{ij} = \frac{-1}{2\pi\epsilon_0} \left[\arctan \frac{x}{d} \right]_{x_s}^{x_e} \quad (6.54)$$

$$H_{ii} = \hat{H}_{ii} + c_i \quad (6.55)$$

6.5 总结

本次课程需要同学重点掌握以下两个知识：

1. 积分方程的构造

对于差分方程 $Df = s$, 其中 D 为差分算子, f 为待求场量, s 为源。格林函数 (Green's function) $G(\mathbf{r}, \mathbf{r}')$ 满足 $D_r G(\mathbf{r}, \mathbf{r}') = \delta(\mathbf{r} - \mathbf{r}')$, 其中 D_r 代表对 \mathbf{r} 进行差分计算。给定格林函数, 差分方程可以写作积分方程的形式

$$f(\mathbf{r}) = \int G(\mathbf{r}, \mathbf{r}') s(\mathbf{r}') dV'$$

对于泊松等式 $-\epsilon_0 \nabla^2 \phi = \rho$, 我们有

- 三维中的格林函数 (基本解) 为 $G(\mathbf{r}, \mathbf{r}') = |\mathbf{r} - \mathbf{r}'|/(4\pi\epsilon_0)$, 对应于点、面电荷的电势分布;
- 二维中的格林函数 (基本解) 为 $G(\mathbf{r}, \mathbf{r}') = -1/(2\pi\epsilon_0) \ln |\mathbf{r} - \mathbf{r}'|$, 对应于线电荷的电势分布。

2. BEM 的系数矩阵构造和程序编写方法

BEM 方法采用有限元的思想, 将导体电荷分布写成元上的基函数, 任意一点电势为

$$\phi_k(\mathbf{r}) = \int G(\mathbf{r}, \mathbf{r}') s_k(\mathbf{r}') dS' \quad \rightarrow \quad \bar{\phi}(\mathbf{r}) = \int \sum_{k=1}^N a_k \phi_k(\mathbf{r})$$

其中 $s_k(\mathbf{r})$ 为源 (电荷) 分布的基函数。为了求解 a_k , 我们需要选择同基函数 (或有限元数目) 一样多的测试函数 $\omega_k(\mathbf{r})$, 然后通过加权余量法

$$\int_{\text{conductor}} \omega_k(\mathbf{r}) [f - f_{\text{specified}}] dS = 0$$

得到线性方程组系数矩阵的计算方法

$$\sum_{k=1}^N A_{jk} a_k = b_j, \quad j = 1, 2, \dots, N$$

$$A_{jk} = \int \omega_j(\mathbf{r}) f_k(\mathbf{r}) dS = \int dS \omega_j(\mathbf{r}) \int dS' G(\mathbf{r}, \mathbf{r}') s_k(\mathbf{r}')$$

$$b_j = \int \omega_j(\mathbf{r}) f_{\text{spec}}(\mathbf{r}) dS$$

最小化格林函数 $G(\mathbf{r}, \mathbf{r}')$ 与 $f - f_{\text{specified}}$ 的积分。

编程实现时，我们有两个选择测试函数（权函数）的方法

- 配点法 (collocation)。令 $\omega_k(\mathbf{r}) = \delta^2(\mathbf{r} - \mathbf{r}_k)$ ，即在 $\mathbf{r} = \mathbf{r}_k$ 处计算电场；在二维 BEM 问题中， A_{jk} 变为 $A_{jk} = \int_{\xi_s}^{\xi_e} dS' G(\mathbf{r}, \mathbf{r}')$, $b_j = f_{\text{spec}}(\mathbf{r}_j)$
- 伽辽金法 (Galerkin's method)。即令 $\omega_k(\mathbf{r}) = s_k(\mathbf{r})$

在 BEM 方法中，积分项常常可被分为具备奇异积分项和常规积分项。对于奇异积分项我们往往可得到解析的定积分表达式，而对于常规积分项我们则可使用数值积分法进行计算。

6.6 习题

习题 6.1: 写出静电场中的积分方程和微分方程，并从数值计算的角度，描述两者的优劣。

习题 6.2: 推导三维自由空间中的 Green 函数（针对点电荷，给出 laplace 算子的球坐标形式，并求解积分）

习题 6.3: 描述 BEM 方法中的配点法和加权余量法。针对 BEM 数值计算中的线性方程组 $A\mathbf{f} = \mathbf{b}$ 的构造，对比两种方法，写出 BIE 实施步骤，并且给出配点法和伽辽金有限元法中 A_{jk} 和 b_j 的计算公式。

习题 6.4: 通过阅读 2D BEM 计算程序，讨论配点法和采用分段平滑的基函数对于程序编写的便易之处。

习题 6.5: 推导并讨论空间中包角为 α 的带电导体电势分布的奇异性。

第7章 电磁场逆问题

刘本源 *byl.liu@fmmu.edu.cn*

学时 上机

8 6

目的：能够解释电磁场逆问题的概念和特点，能够解释电磁场逆问题求解中涉及的常见问题和对策。

重点：电阻抗断层成像逆问题的求解方法

难点：电阻抗断层成像逆问题的求解方法

本章内容安排四个单元，分别是：

1. 数学模型。
2. 优化算法。(需要储备矩阵求导、范数定义知识)
3. Jacobian 矩阵。
4. 电阻抗断层成像。

7.1 数学模型

7.1.1 逆问题实例及数学模型

修订：仅保留蝙蝠和 CT 例子，随后讲解电磁场数值计算的逆问题，第二课时讲反投影法。与等位线的概念对应起来，非线性反投影与 CT 的线性对比起来。

7.1.2 电磁场数值计算中的逆问题

7.1.3 电阻抗断层成像简述

7.2 优化算法

We call two problems inverses of one another if the formulation of each involves all or part of the solution of the other. Often, for historical reasons,

one of the two problems has been studied extensively for some time, while the other is newer and not so well understood. In such cases, the former problem is called the direct problem, while the latter is called the inverse problem.

J. B. Keller. Inverse problems. 1976.

本节讲解 Newton 算法，主题是“Numerical Methods for Inverse Problems”。用最通俗的语句讲解牛顿迭代法，而对于拟牛顿法，则以附录的形式给出，供学有余力的同学阅读。每一部分的内容包括：算法是“怎么来的”、“是什么”以及“怎么用”，以程序直观显示结果，更易上手。

主要内容中，拟牛顿算法一节参考了 peghoty 的博客 [15]，Nocedal 的论文 [16]，BFGS 算法、L-BFGS 算法的 Wiki 页面等；推导过程用到了 Sherman–Morrison 公式及 Woodbury 矩阵求逆公式的 Wiki 页面；有关牛顿优化算法也参考了其他高校的讲义，例如 EE236C[17]，RIT 的教程 [18]，宾州大学 Griffin 的讲义“Numerical Optimization”[19] 等。

7.2.1 引子

常见的偏微分方程可写为，

$$D(\mathbf{x}, \alpha) = \mathbf{y}$$

其中 \mathbf{x} 是场量， α 是场域的参数。例如，在电磁场数值计算中，laplace 方程（静电场方程）为

$$-\nabla \cdot (\epsilon \nabla f) = 0$$

式中节点电势 f 对应于 \mathbf{x} ，而介电常数 ϵ 对应于 α 。电磁场数值方法即将理论变成可以数值计算的程序。

先前学习的数值解法，例如有限差分法、有限元法，主要目的是求解电磁场数值计算中的正问题。其特点是场域 Ω 以及场域参数 ϵ 已知，求电场或磁场的能量分布。其数学表达式往往可写为矩阵的形式，例如对于有限差分法，有

$$\mathbf{A}\mathbf{f} = \mathbf{b}$$

其中在二维情形中， \mathbf{A} 是五点差分格式构成的矩阵。对于有限元法，有

$$\mathbf{K}\mathbf{f} = \mathbf{b}$$

其中 \mathbf{K} 是有限元系数矩阵。后续的讨论将以有限元法为主。

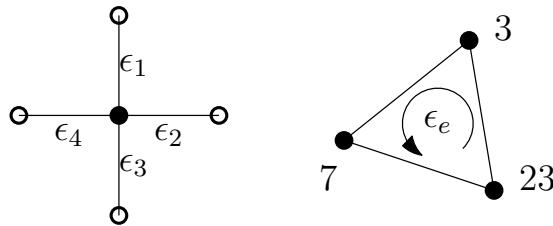


图 7.1 当考虑介质特性时的 FDM 和 FEM 数值方法

若已知区域中介电特性（物质的属性）分布¹，那么利用本节课所学的数值方法，电磁场的正问题即为：给定区域 Ω 和其上介电常数的分布 ϵ ，计算电势。这里 \mathbf{K} 为 ϵ 的参数，即

$$\mathbf{K}(\epsilon) \mathbf{f} = \mathbf{b}$$

而与之相对的是，电磁场数值计算中的反问题（逆问题）为：给定部分观测 \mathbf{f}^* ，求区域 Ω 上的介电常数（或电导率）分布。此时，边界条件 \mathbf{b} 是激励， \mathbf{f} 是响应， $\mathbf{K}(\sigma)$ 是模型， σ 是待估计参量。

下面着重将一下部分观测的概念。我们可以想象一下，在二维情形下（或者更科幻一些，在二维世界中生活的人只能观察到物体的边界特征，而三维世界的人只能观察到物体面的特性），能够在边界 $\partial\Omega$ 上方便的获得测量，在 Ω 内部获得测量较难或不切实际（需要将探针伸到平面内）。若有限元中面元的个数为 n_e ，节点个数为 n_n ，而实际观测获得的边界观测数目为 n_b ，一般有 $n_b \ll n_n$ 。因而给定部分观测 $\mathbf{f}^* \in \mathbb{R}^{n_b}$ ，求解面元上的 $\sigma \in \mathbb{R}^{n_e}$ 是一个病态的、欠定的问题。也就是说存在无穷多个 σ 的组合，都能符合 \mathbf{f}^* 的观测结果。

理解欠定性最好的例子是求解方程组

$$2x + y = 1$$

该方程写成矩阵的形式 $A\mathbf{f} = \mathbf{b}$ ，其中 A ， \mathbf{f} 和 \mathbf{b} 分别为

$$A = \begin{bmatrix} 2 & 1 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

为一个欠定矩阵。方程的一个解是 $(1/3, 1/3)$ ，而另外一个解是 $(1/2, 0)$ 。前者

¹ 我们此处简化了具体问题。事实上电阻抗成像针对电导率 σ 进行估计

的解符合最小二乘原则，而后的解则是稀疏的。基于对解特性的理解，可以帮助我们添加约束，减小欠定问题求解的不确定性。

例题 7.1：附讲：向量范数的定义。

$$\|\mathbf{x}\|_2^2 \triangleq \mathbf{x}^T \mathbf{x} = \sum_i x_i^2$$

$$\|\mathbf{x}\|_1 \triangleq \sum_i |x_i|$$

例如在二维中，向量 $\mathbf{x} = \{x_1, x_2\}$ 可写为 $\mathbf{x} = \{x, y\}$ ，那么

$$\|\mathbf{x}\|_2 = \sqrt{x^2 + y^2}, \quad \|\mathbf{x}\|_1 = |x| + |y|$$

针对上一例题的优化，我们还可以从范数的几何图像上寻求思路。

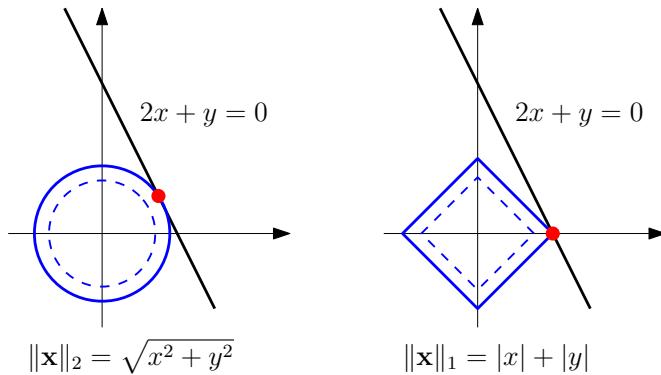


图 7.2 向量范数的几何图示及优化示意

那么电磁场数值计算中逆问题可描述为

1. 已知物理原理 (Maxwell 方程组)，可用有限元方法计算正问题：

$$\mathbf{K}(\epsilon) \mathbf{f} = \mathbf{b}$$

2. 获得部分观测 \mathbf{f}^*
3. 添加约束 (正则化)，引入解的先验知识
4. 寻找 ϵ ，使得迭代计算得到的 \mathbf{f} 与 \mathbf{f}^* 之间满足 (最小化)

$$\min \|\mathbf{f} - \mathbf{f}^*\|_2^2$$

求解最小化问题，需要学习优化算法；而求解欠定方程，则需要添加约束。

接下来，我们分别讲解这两方面的内容。

7.2.2 Newton (牛顿) 算法

7.2.2.1 问题引出

考虑以下无约束最小化问题

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (7.1)$$

其中 $\mathbf{x} = (x_1, x_2, \dots, x_N)^T \in \mathbb{R}^N$ 。不妨假设 $f : \mathbb{R}^N \rightarrow \mathbb{R}$ 为一个凸函数¹，且两阶连续可微。此外记式(7.1)的极小值为 \mathbf{x}^* 。

约束信息可以方便的添加进优化的过程中。例如，在欠定矩阵

$$\mathbf{A}\mathbf{x} = \mathbf{y}$$

求解中，我们可以对 \mathbf{x} 添加各种类型的先验信息。例如在压缩感知 (Compressive Sensing) 中，可以添加稀疏性约束，此时最小化问题变为

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda \|\mathbf{x}\|_0$$

其中 $\|\mathbf{x}\|_0$ 为 \mathbf{x} 中非零元素的数目， λ 为一罚参数，用来在数据拟合误差 $f(x)$ 和稀疏性 $\|\mathbf{x}\|_0$ 之间进行权衡。一般直接优化 $\|\mathbf{x}\|_0$ 是比较困难的，我们可以转而优化 (proximal approximation) $\|\mathbf{x}\|_1$ ，即

$$\|\mathbf{x}\|_1 \triangleq \sum_i |x_i|$$

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda \|\mathbf{x}\|_1$$

类似的，我们可以添加更多的种类的，诸如平滑性、分块性约束。添加约束 (正则化) 的作用在于减小解的不确定性，增加优化算法的抗噪能力 (鲁棒性)。

7.2.2.2 函数的极值与梯度法

在函数的极值点上，其导数为零

$$f'(x) = 0$$

¹ 这里需要注意的是，凸函数定义为

$$f(x_c) < \frac{f(x_1) + f(x_2)}{2}, \quad x_c = \frac{x_1 + x_2}{2} \quad \forall x_1, x_2$$

凸函数使得针对 f 的优化有全局最小值。

反之未必成立，例如 $f'(x) = 0$ 的点可能为二维函数的鞍点（Saddle Point）。

例题 7.2：例如，对于函数 $z = x^2 - y^2$ ，在 $(0, 0)$ 处即为函数的鞍点。而在该点上， $z' = 0$ 。你能够绘制出 z 的图像么？

在求极小值 $\min f(x)$ 问题中，直接求解 $f'(x) = 0$ 是比较困难的，常用迭代方法进行计算。梯度法是最直观的迭代算法：任选一点 x_k ，迭代计算

$$x_{k+1} = x_k - r f'(x_k)$$

其中 r 是可控步长参数。梯度法的原理很简单：我们只关心迭代的方向（梯度方向），在迭代的终点，也就是极值点上，函数的导数为 0。

试思考，怎样用梯度法求解二次函数的极值 $\min f(x)$ 呢？只需将求导 f' 变为梯度 ∇f 即可

$$x_{k+1} = x_k - r \nabla f$$

7.2.2.3 牛顿迭代法的一维情形

我们从同学们最为熟悉的牛顿拉夫逊（Newton–Raphson）求根公式展开。这里讲解的 Newton 迭代优化算法与利用 Newton–Raphson 求函数 $f(x) = 0$ 的根是有差别的。但恰恰相反，从 Newton–Raphson 求根公式可以更好的理解 Newton 优化方法。

为了求函数 $f(x) = 0$ 的根，Newton–Raphson 方法首先任选一点 x_k ，计算该点 $(x_k, f(x_k))$ 的斜率

$$f'(x_k) = \frac{f(x_k) - 0}{x_k - x_{k+1}}$$

那么通过迭代的计算

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (7.2)$$

可得序列 $\{x_k\}$ ，序列的极限即为 $f(x) = 0$ 的根 x^* （每次迭代的结果都更靠近 $f(x) = 0$ 的根）。若 $f(x) = 0$ 有不同多个根，则收敛的结果跟初始值 x_0 的选取有关。

类比的去思考，针对优化（最小化）问题，我们需要求的是导数为零的点，

也就是 f' 的根

$$f'(x) = 0$$

那么，直接将牛顿拉夫逊方法移植而来，即可得 Newton 优化算法

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

其中 $f' = 0$ 的点也被称为驻点 (Stationary Point)。当待优化的代价函数为凸函数时，该点即为函数的极小值。同学们可对比式(7.6)和式(7.2)体会两者的差别。

那么，Newton 法的思路就可以很直观的理解了。牛顿法的基本思想是：在当前点 (x_k) 附近对函数 $f(x)$ 做二阶泰勒展开 (近似)，寻找极小点的下一个估计值。

$$\varphi(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2 \quad (7.3)$$

由极值条件可知， $\varphi(x)$ 的极值点应满足

$$\varphi'(x) = 0 \quad (7.4)$$

忽略函数 2 阶以上导数 (即令高阶导为 0)，等价于在 $f(x_k)$ 附近取线性近似，可得

$$f'(x_k) + f''(x_k)(x - x_k) = 0 \quad (7.5)$$

进而求得

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \quad (7.6)$$

通过迭代计算，产生序列 $\{x_k\}$ ，序列的极限即逼近极值点 x^* 。

下面给一个简单的例子。

例题 7.3：求解 $f(x) = x^2 + 2x - 3$ 的极小值点。很明显有 $x^* = -1$ 。

取任意一点，例如 $x_0 = 4$ ，计算 $f'(x_0) = 2x_0 + 2 = 10$ ， $f''(x_0) = 2$ ，随后

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} = x_0 - \frac{10}{2} = -1$$

需要注意的是，由于函数是二次项，那么我们只需一步即可迭代计算得到函数 f 的极值点。这是因为当函数是二次项时，在任意一点附近的二阶泰勒展

开不是近似，而就是函数本身。

另外，如果我们要计算函数的根 $f(x) = 0$ 。那么随机选一点 $x_0 = 2$,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = x_0 - \frac{5}{6} = 1\frac{1}{6}$$

继续迭代计算

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = x_1 - \frac{1}{6} \cdot \frac{25}{26} = 1\frac{1}{256}$$

迭代的序列 $\{x_k\}$ 即逼近于函数 $f(x)$ 的一个根 $x^* = 1$ 。

7.2.2.4 几何图示

梯度法：梯度法是利用 Δx 在一次导数下的逼近

$$x_{k+1} = x_k - r f'(x_k)$$

其中 r 是一个控制迭代步长 Δx 的参量。

牛顿法：牛顿法不仅利用了梯度，同时考虑了梯度的变化量（二阶导数）的信息

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

两者的差别可以从图7.3看出¹。

另外，我们给出一个牛顿法和梯度法对比的直观解释，参见图7.4。从函数逼近的角度来看：梯度下降法相当与用一个线（平面）去拟合当前局部曲线，而牛顿法是用一个二次曲线（二次曲面）去拟合当前位置的局部曲线。

7.2.2.5 高维情形

针对实际优化问题，我们不仅仅需要求解一维函数的优化，更多的情形下还需要对很多变量（多变量函数 $f(\mathbf{x})$ ）进行优化。

类似的，对于 $N > 1$ 的情形，函数 $f(\mathbf{x})$ 的二阶泰勒展开为：

$$\varphi(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k) \cdot (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T \cdot \nabla^2 f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) \quad (7.7)$$

¹参见http://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

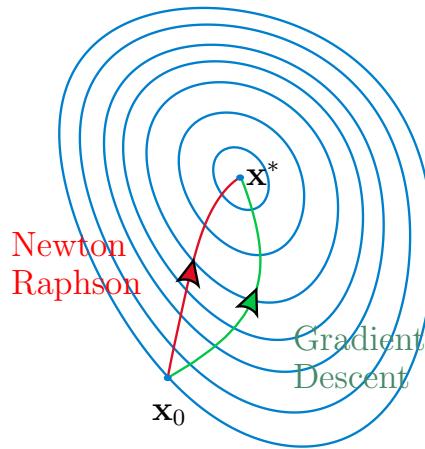


图 7.3 牛顿法和梯度（下降）法寻找函数的极小值的示意图。梯度法在每一步，都依照当前点梯度的方向进行下降。牛顿法则考虑了梯度的变化量信息，能够更为快速的到达极值点。

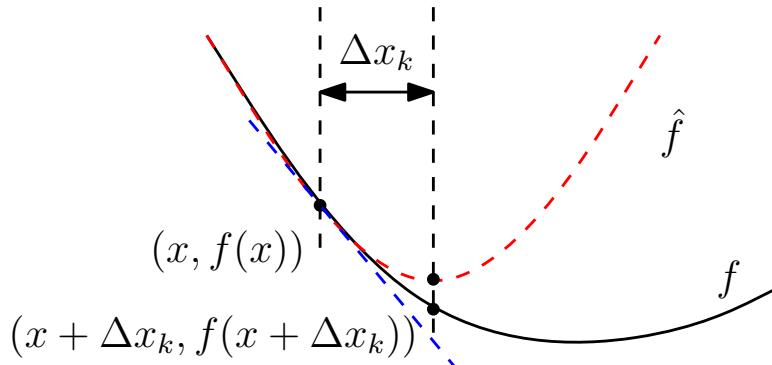


图 7.4 从对函数的二次逼近上理解牛顿迭代法。图中 f 是待优化的函数， \hat{f} 是在 x 点处的二阶逼近函数， Δx_k 是迭代方向，第 $k+1$ 时刻的迭代位置为 $x_{k+1} = x + \Delta x_k$ 。如果待优化的函数本身就是二阶的，那么牛顿法可以只用 1 步到达极值点。

其中 ∇f 为函数 f 的梯度向量， $\nabla^2 f$ 为 f 的海森 (Hessian Matrix) 矩阵。

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix} \quad \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix} \quad (7.8)$$

我们记 (\mathbf{g} 表示 gradient, \mathbf{H} 表示 Hessian)

$$\mathbf{g} = \nabla f(\mathbf{x}_k) \quad (7.9)$$

$$\mathbf{H} = \nabla^2 f(\mathbf{x}_k) \quad (7.10)$$

若对于函数 f , \mathbf{H} 满足

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}, \quad \forall i, j \quad (7.11)$$

即 \mathbf{H} 为对称矩阵。

极值条件要求 \mathbf{x}_{k+1} 为 $\varphi(\mathbf{x})$ 的驻点, 即

$$\nabla \varphi(\mathbf{x}) = 0 \quad (7.12)$$

可得 (对式(7.7)两边同时求梯度)

$$\mathbf{g}_k + \mathbf{H}_k(\mathbf{x} - \mathbf{x}_k) = 0 \quad (7.13)$$

若 \mathbf{H}_k 非奇异 (矩阵可求逆), 则

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_k^{-1} \mathbf{g}_k \quad (7.14)$$

我们定义 $\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$ 为牛顿方向。一个完整的牛顿迭代法需要制定初值 \mathbf{x}_0 和收敛条件 ϵ , 当梯度的模值 (绝对值) $\|\mathbf{g}_k\| < \epsilon$ 时退出算法。

例题 7.4: 计算 (Rosenbrock function)

$$f(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

在 $(x, y) = (1, 1)$ 点处的梯度向量 \mathbf{g} 和 Hessian 矩阵 \mathbf{H} 。

解:

$$\mathbf{g} = \begin{bmatrix} \frac{\partial f}{\partial x} = -2(1 - x) - 400x(y - x^2) \\ \frac{\partial f}{\partial y} = 200(y - x^2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} = 2 - 400y + 1200x^2 & \frac{\partial^2 f}{\partial x \partial y} = -400x \\ \frac{\partial^2 f}{\partial y \partial x} = -400x & \frac{\partial^2 f}{\partial y^2} = 200 \end{bmatrix} = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$$

例题 7.5: 计算下式函数

$$f(x, y) = x^2 - y^2$$

在 $(x, y) = (0, 0)$ 点处的梯度向量 \mathbf{g} 和 Hessian 矩阵 \mathbf{H} 。

解：

$$\mathbf{g} = \begin{bmatrix} \frac{\partial f}{\partial x} = 2x \\ \frac{\partial f}{\partial y} = -2y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} = 2 & \frac{\partial^2 f}{\partial x \partial y} = 0 \\ \frac{\partial^2 f}{\partial y \partial x} = 0 & \frac{\partial^2 f}{\partial y^2} = -2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

从这两个例题中，你能判断所计算的点是否是函数的极值点么？你的判断准则又是什么？

7.2.2.6 Hessian 矩阵

牛顿法具备较好的收敛特性。例如，牛顿法对于二次函数只需要一次迭代即可以到达极值点 \mathbf{x}^* 。这是因为对于二次函数 f ，其二阶泰勒展开式不是近似而是其本身，此时 **Hessian** 退化成一个常数矩阵。因此牛顿法是一种具有二次收敛特性的算法。但实际使用中，牛顿法的症结就在于 Hessian 矩阵 \mathbf{H} 。

问题一、Hessian 矩阵计算量大。

计算 Hessian 矩阵的算法复杂度为 $O(N^2)$ 。

问题二、当 \mathbf{H}_k 无法求逆。

我们可以通过迭代法，近似求解线性代数方程组 $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ ，来间接的求牛顿方向 \mathbf{d}_k 。

问题三、Hessian 矩阵的限定条件。

这里，我们要讲一下牛顿法中的 Hessian 矩阵有什么限定条件。最优化（寻找极小值） $\min f(\mathbf{x})$ 到第 k 步，需要保证沿 \mathbf{d}_k 方向上，函数值下降（类比斜率乘以 x 轴的步长），即

$$\mathbf{g}_k^T \mathbf{d}_k < 0$$

而牛顿方向可写为 $\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$ 或 $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ ，则

$$\mathbf{d}_k^T \mathbf{H}_k^T \mathbf{d}_k > 0$$

这一点要求所使用的 Hessian 矩阵， \mathbf{H}_k 是正定的。而 f 在远离极值点处，不一定能够满足这一条件，将会导致目标函数不降反升。

7.2.2.7 阻尼牛顿法

原始的牛顿法可能存在 $f(\mathbf{x}_{k+1}) > f(\mathbf{x}_k)$ 的情况，不能保证函数值稳定的下降，甚至可能造成迭代序列 $\{\mathbf{x}_k\}$ 发散而导致计算失败。

这里我们可以想象一下拉长后，受重力作用下弹簧的自由震荡。弹簧重力与弹簧自身弹力平衡的点类似于我们所搜寻的极小值；在震荡的过程中，振幅逐渐降低，最后将收敛到该极值点上。

阻尼牛顿法在计算得到 $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ 后，通过

$$\lambda_k = \arg \min_{\lambda} f(\mathbf{x}_k + \lambda \mathbf{d}_k) \quad (7.15)$$

即选择一个系数 λ ，使得迭代后的函数值 $f(\mathbf{x}_{k+1})$ 最小。计算得到 λ_k 后，进行可变步长因子的迭代

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k \quad (7.16)$$

针对 λ_k 参数的选取方法，又被称作线搜索（line search）算法。在后续讲解的 Quasi–Newton（拟牛顿）算法中，常会用到线搜索以及更通用的非精确（Inexact line search）线搜索算法。

7.2.2.8 牛顿法小结

牛顿法利用目标的一阶偏导数和一阶偏导数的变化率的特性，能够更全面的确定合适的搜索方向加快收敛，具备二阶收敛速度。而牛顿法的缺点为：

1. 目标函数 f 必须具备一阶、二阶偏导数，同时 \mathbf{H} 矩阵必须正定
2. 需要计算并存储 \mathbf{H} 矩阵及其逆，复杂度以 $O(N^2)$ 增加

7.2.3 正则化方法

前面，我们讲解了由于逆问题的欠定性，求解优化问题需要对解的形式加入约束。而约束信息，可以方便的加入到优化算法中。

针对无约束的优化，牛顿法可用于求解

$$\min_{\mathbf{x}} f(\mathbf{x})$$

这类最优化问题，其迭代计算式为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$$

在 $D(x) = y$ 这类模型中，首先对算子 D 和观测离散化，得到线性的、矩阵向量相乘形式 $\mathbf{A}x = \mathbf{y}$ 。那么，

- 能否利用牛顿法求解线性方程组 $\mathbf{A}x = \mathbf{y}$ 呢？
- 如何在求解方程组过程中加入约束信息呢？

7.2.3.1 最小二乘 (Least square) 方法

最小二乘法最小化数据的拟合误差。即令 $f(x) = \|\mathbf{A}x - \mathbf{y}\|^2$ ，最小化

$$\min_{\mathbf{x}} \|\mathbf{A}x - \mathbf{y}\|^2$$

那么给定任意初始值 x_k ，我们可分别计算

$$\begin{aligned} f'(x_k) &= 2\mathbf{A}^T(\mathbf{A}x_k - \mathbf{y}) \\ f''(x_k) &= 2\mathbf{A}^T\mathbf{A} \end{aligned}$$

进而可得

$$x_{k+1} = x_k - (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\mathbf{A}x_k - \mathbf{y})$$

当 $x_k = 0$ 时，单步迭代更新为 $x_1 = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} = \mathbf{A}^\dagger\mathbf{y}$ ，其中 \mathbf{A}^\dagger 为 \mathbf{A} 的伪逆，也被称作 Moore–Penrose 伪逆。

由于最小二乘中 $f(x)$ 是二次项，则任选初始点，单步迭代即可得到该式的极小值，即

$$x^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} = \mathbf{A}^\dagger\mathbf{y}$$

若 \mathbf{A} 是行满秩（欠定）的，右除伪逆写作 $\mathbf{A}^\dagger = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$ 。有关左除伪逆和右除伪逆，可见

$$\begin{aligned} \mathbf{A} \left[\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} \right] &= \mathbf{I} \\ \left[(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \right] \mathbf{A} &= \mathbf{I} \end{aligned}$$

例题 7.6: 右除伪逆公式的推导。

解：当 \mathbf{A} 为欠定矩阵时，需最小化

$$\text{minimize } \mathbf{x}^T \mathbf{x}$$

subject to $\mathbf{A}\mathbf{x} = \mathbf{y}$

采用 Lagrange Multipliers 方法，

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{x}^T \mathbf{x} + \boldsymbol{\lambda} (\mathbf{A}\mathbf{x} - \mathbf{y})$$

分别计算 $\nabla_{\mathbf{x}} L$ 和 $\nabla_{\boldsymbol{\lambda}} L$ 可得

$$\nabla_{\mathbf{x}} L = 2\mathbf{x} + \mathbf{A}^T \boldsymbol{\lambda} = 0$$

$$\nabla_{\boldsymbol{\lambda}} L = \mathbf{A}\mathbf{x} - \mathbf{y} = 0$$

第一个公式可得 $\mathbf{x} = -\mathbf{A}^T \boldsymbol{\lambda} / 2$ ，代入第二式可得

$$\boldsymbol{\lambda} = -2(\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{y}$$

进而

$$\mathbf{x} = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{y}$$

7.2.3.2 Tikhonov 正则化

Andrey Nikolayevich Tikhonov (中: 吉洪诺夫, October 30, 1906 – October 7, 1993) was a Soviet and Russian mathematician known for important contributions to topology, functional analysis, mathematical physics, and ill-posed problems.

当矩阵 \mathbf{A} 是欠定矩阵时，矩阵求逆 $(\mathbf{A}^T \mathbf{A})^{-1}$ 是病态的，求解欠定问题需要对 $f(x)$ 加入约束。在优化问题中常用的正则化（约束）方法为 Tikhonov 正则化，其形式为

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \alpha \|\mathbf{x}\|^2$$

其中 $\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$ 为数据拟合误差 (misfit)， $\|\mathbf{x}\|^2$ 被称作约束项， α 为罚参数 (penalty)。

更通用的约束形式是对 $\|\mathbf{x}\|^2$ 项加入一个算子 \mathbf{L} ，可写为

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \alpha \|\mathbf{L}\mathbf{x}\|^2$$

\mathbf{L} 可以是一阶差分算子，此时 Tikhonov 正则化方法可用于重构平滑的信号（即对信号的平滑性加入约束）。

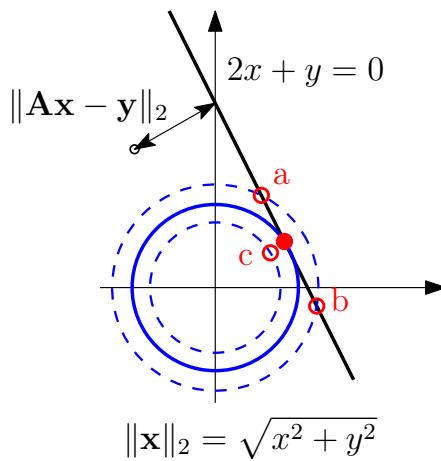


图 7.5 Tikhonov 优化的几何解释

我们方便的用牛顿优化算法得到 Tikhonov 的迭代计算公式

$$\begin{aligned}\frac{1}{2}f'(\mathbf{x}_k) &= \mathbf{A}^T(\mathbf{A}\mathbf{x}_k - \mathbf{y}) + \alpha\mathbf{L}^T\mathbf{L}\mathbf{x} \\ \frac{1}{2}f''(\mathbf{x}_k) &= \mathbf{A}^T\mathbf{A} + \alpha\mathbf{L}^T\mathbf{L}\end{aligned}$$

当初始值 $\mathbf{x}_0 = 0$ 时，单步迭代计算为（思考当 $\mathbf{L} = \mathbf{I}$ 时）

$$\mathbf{x}_1 = (\mathbf{A}^T\mathbf{A} + \alpha\mathbf{L}^T\mathbf{L})^{-1}\mathbf{A}^T\mathbf{y}$$

另一方面，在最小二乘法的基础上，我们可以用堆叠（Stacked form）法来推导 Tikhonov 正则化的更新公式。在最小二乘问题中，

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$$

其迭代更新公式为 $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$ 。Tikhonov 正则化问题可以写为类似的最小二乘形式，定义

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{y} \\ \sqrt{\alpha}\mathbf{L}\mathbf{x} &= \mathbf{0}\end{aligned}$$

上式可合并写为堆叠形式（stacked form），即

$$\begin{bmatrix} \mathbf{A} \\ \sqrt{\alpha}\mathbf{L} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}$$

或者

$$\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{y}}$$

约束参数 α 可以方便的包含在矩阵 $\tilde{\mathbf{A}}$ 中。

那么类似的，其最小二乘解为

$$\begin{aligned}\mathbf{x} &= (\tilde{\mathbf{A}}^T \tilde{\mathbf{A}})^{-1} \tilde{\mathbf{A}}^T \tilde{\mathbf{y}} \\ &= (\mathbf{A}^T \mathbf{A} + \alpha \mathbf{L}^T \mathbf{L})^{-1} \mathbf{A}^T \mathbf{y}\end{aligned}$$

其中用到了

$$\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}^T & \sqrt{\alpha} \mathbf{L}^T \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \sqrt{\alpha} \mathbf{L} \end{bmatrix} = \mathbf{A}^T \mathbf{A} + \alpha \mathbf{L}^T \mathbf{L}$$

及

$$\tilde{\mathbf{A}}^T \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{A}^T & \sqrt{\alpha} \mathbf{L}^T \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} = \mathbf{A}^T \mathbf{y}$$

7.2.3.3 罚参数 α 的选取方法

7.2.4 Quasi–Newton (拟牛顿) 算法

牛顿法的主要缺陷都是由 \mathbf{H} 矩阵导致的，要么是很难计算得到（计算量大），或者是求逆是病态的。拟牛顿法，即 Quasi–Newton，或称为准牛顿法，其基本思想是：不用二阶偏导数，转而去构造出近似 Hessian 矩阵（或者 Hessian 矩阵的逆）的正定对称矩阵。而 Hessian 矩阵代表了牛顿法中的搜索方向（或称为“牛顿方向”）。

在拟牛顿法框架下，不同的 \mathbf{H} 矩阵构造（或迭代更新）方法就产生了不同的拟牛顿法。当前主要的拟牛顿法有 DFP 方法、BFGS 方法和 L-BFGS 方法，同学们不要被它们的名字吓坏，下面将一步一步理解他们的原理和内容。

首先，我们要简单讲一下为什么拟牛顿法能够得到较优解。前面我们讲过牛顿法要收敛，Hessian 矩阵需要满足

$$\mathbf{d}_k^T \mathbf{H}_k^T \mathbf{d}_k > 0$$

即正定性。而对于一般的优化问题，在远离极小值点附近， \mathbf{H}_k 的正定性无法得到满足，进而影响算法的性能。拟牛顿算法可以逐步逼近 Hessian 矩阵，只要在逼近过程中保证每一次近似结果的正定性，那么就可以使得目标函数值沿 \mathbf{d}_k 方向降下去。最终在极值点附近，构造出来的矩阵就很像 Hessian 矩阵了。

拟牛顿算法的思想很简单，

1. 牛顿法在第 k 次迭代时, 根据 \mathbf{x}_k , 计算 \mathbf{g}_k 和 \mathbf{H}_k
2. 拟牛顿法根据 $(k, k+1)$ 时刻的 \mathbf{x} 和 \mathbf{g} 近似的构造出 \mathbf{H}_{k+1} 进行迭代

拟牛顿算法的关键就是对 Hessian 矩阵进行近似。首先需要知道的是, 我们不能随便的近似, 逼近的二阶矩阵需要满足一定的条件。在当前文献中, 这一条件被称作拟牛顿条件。我们将待优化的函数 f 在 \mathbf{x}_{k+1} 处二阶泰勒展开

$$\begin{aligned} f(\mathbf{x}) \approx & f(\mathbf{x}_{k+1}) + \mathbf{g}_{k+1} \cdot (\mathbf{x} - \mathbf{x}_{k+1}) \\ & + \frac{1}{2} (\mathbf{x} - \mathbf{x}_{k+1})^T \mathbf{H}_{k+1} (\mathbf{x} - \mathbf{x}_{k+1}) \end{aligned} \quad (7.17)$$

对式(7.17)两边同时作用一个梯度算子 ∇ , 并取 $\mathbf{x} = \mathbf{x}_k$ 可得

$$\mathbf{g}_{k+1} - \mathbf{g}_k \approx \mathbf{H}_{k+1} (\mathbf{x}_{k+1} - \mathbf{x}_k) \quad (7.18)$$

引入 (注意脚标 $k, k+1$)

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k, \quad \mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k \quad (7.19)$$

则式(7.18)可写为

$$\mathbf{y}_k \approx \mathbf{H}_{k+1} \mathbf{s}_k \quad (7.20)$$

或者

$$\mathbf{s}_k \approx \mathbf{H}_{k+1}^{-1} \mathbf{y}_k \quad (7.21)$$

根据式(7.20)和式(7.21), 拟牛顿条件为:

$$\mathbf{B}_{k+1} \approx \mathbf{H}_{k+1} \quad \rightarrow \quad \mathbf{y}_k = \mathbf{B}_{k+1} \mathbf{s}_k \quad (7.22)$$

$$\mathbf{D}_{k+1} \approx \mathbf{H}_{k+1}^{-1}, \quad \rightarrow \quad \mathbf{s}_k = \mathbf{D}_{k+1} \mathbf{y}_k \quad (7.23)$$

寻找 (或逼近) \mathbf{B}_k 或 \mathbf{D}_k 的不同方法就构成了不同的拟牛顿法。

7.2.4.1 DFP 算法

最早的拟牛顿法是以 William C. Davidon, Roger Fletcher, Michael J. D. Powell 三人的名字的首字母命名的。它最初由 Davidon 于 1959 年提出, 后经 Fletcher 和 Powell 加以完善和发展。

该算法的核心是, 通过迭代的方法, 对 $\mathbf{D}_{k+1} \approx \mathbf{H}_{k+1}^{-1}$ 进行近似。DFP 方法假定 \mathbf{D}_k 的迭代更新式为

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \Delta \mathbf{D}_k, \quad k = 0, 1, 2, \dots \quad (7.24)$$

其中 \mathbf{D}_0 可以选择为单位矩阵 \mathbf{I} 。那么矫正矩阵 $\Delta \mathbf{D}_k$ 应该怎样构造呢？DFP 采用待定法，即首先将 $\Delta \mathbf{D}_k$ 待定为某种形式，再用拟牛顿条件式(7.23)来推导。

DFP 算法中的一个小技巧是将 $\Delta \mathbf{D}_k$ 待定为 (tricky)

$$\Delta \mathbf{D}_k = \alpha \mathbf{u} \mathbf{u}^T + \beta \mathbf{v} \mathbf{v}^T \quad (7.25)$$

其中 α, β 为待定系数， $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$ 为待定向量。这种待定方法能够保证矩阵 $\Delta \mathbf{D}_k$ 的对称性。这里需要注意的是 $\alpha \mathbf{u} \mathbf{u}^T$ ($\beta \mathbf{v} \mathbf{v}^T$) 分别是一个秩为 1 的矩阵，则 $\Delta \mathbf{D}_k$ 可以被看作一个 2 秩的更新矩阵 (Correction Matrix)。

将式(7.25)代入式(7.23)中可得

$$\mathbf{s}_k = \mathbf{D}_k \mathbf{y}_k + \alpha \mathbf{u} \mathbf{u}^T \mathbf{y}_k + \beta \mathbf{v} \mathbf{v}^T \mathbf{y}_k \quad (7.26)$$

$$= \mathbf{D}_k \mathbf{y}_k + \mathbf{u}(\alpha \mathbf{u}^T \mathbf{y}_k) + \mathbf{v}(\beta \mathbf{v}^T \mathbf{y}_k) \quad (7.27)$$

由于 $\alpha, \beta, \mathbf{u}, \mathbf{v}$ 是任意选定的，不失一般性，我们可以令

$$\left\{ \begin{array}{l} \alpha \mathbf{u}^T \mathbf{y}_k = 1 \\ \beta \mathbf{v}^T \mathbf{y}_k = -1 \\ \mathbf{u} = \mathbf{s}_k \\ \mathbf{v} = \mathbf{D}_k \mathbf{y}_k \end{array} \right. \quad (7.28)$$

可得

$$\alpha = \frac{1}{\mathbf{s}_k^T \mathbf{y}_k} \quad (7.29)$$

$$\beta = -\frac{1}{\mathbf{y}_k^T \mathbf{D}_k \mathbf{y}_k} \quad (7.30)$$

那么 $\Delta \mathbf{D}_k = \alpha \mathbf{u}_k \mathbf{u}_k^T + \beta \mathbf{v}_k \mathbf{v}_k^T$ 为

$$\Delta \mathbf{D}_k = \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k} - \frac{\mathbf{D}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{D}_k}{\mathbf{y}_k^T \mathbf{D}_k \mathbf{y}_k} \quad (7.31)$$

在得到了 \mathbf{H}^{-1} 的近似计算公式之后，DFP 算法剩余的部分和典型的阻尼牛顿法类似。我们同样可以采用 λ_k 的一维搜索，唯一不同的是添加了 \mathbf{D}_{k+1} 的近似计算，以及 $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \lambda_k \mathbf{d}_k$ 。

7.2.4.2 BFGS 算法

BFGS 算法是以其发明者 Broyden, Fletcher, Goldfarb 和 Shanno 四个人的名字的首字母命名的。与 DFP 算法相比, BFGS 算法性能更佳。目前, 它已经成为求解无约束非线性优化问题最常用的方法之一。

BFGS 算法的推导过程与 DFP 类似, 只不过是采用了准牛顿条件式(7.22), 即对 Hessian 矩阵进行近似 $\mathbf{B}_{k+1} \approx \mathbf{H}_{k+1}$ 。BFGS 算法的推导过程可以借鉴 DFP 算法类似的思路, 将 \mathbf{B}_{k+1} 写为

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \Delta \mathbf{B}_k \quad (7.32)$$

推导过程与 DFP 算法近似,

$$\Delta \mathbf{B}_k = \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \quad (7.33)$$

其结论 (与 DFP 中的 $\Delta \mathbf{D}_k$ 对比) 是仅仅互换了 \mathbf{s}_k , \mathbf{y}_k 的位置

$$\begin{cases} \mathbf{u} = \mathbf{y}_k \\ \mathbf{v} = \mathbf{B}_k \mathbf{s}_k \\ \alpha = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \\ \beta = -\frac{1}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \end{cases} \quad (7.34)$$

有关这一点 (互易性), 我们也可以从拟牛顿法限定条件式(7.22)与式(7.23)的对称性上得出。

PBL: 学生可尝试推导 BFGS 的更新矩阵 $\Delta \mathbf{B}_k$, 体会对称性。

但是牛顿法的关键是计算 \mathbf{d}_k , 利用 DFP 方法近似得到的 \mathbf{D}_k 可以直接用来计算 $\mathbf{d}_k = -\mathbf{D}_k \mathbf{g}_k$, 而 BFGS 方法面临的一个问题是计算 $\mathbf{d}_k = -\mathbf{B}_k^{-1} \mathbf{g}_k$ 。我们知道矩阵的求逆是及其耗费计算资源的, 虽然可以通过求解 $\mathbf{B}_k \mathbf{d}_k = -\mathbf{g}_k$ 来得到 \mathbf{d}_k , 仍旧没有 DFP 的矩阵向量相乘更为直接。

这里就要讲一下 BFGS 的另外一个简化步骤了。根据 Sherman–Morrison 公

式¹

$$(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1} \mathbf{u}}, \quad 1 + \mathbf{v}^T A^{-1} \mathbf{u} \neq 0 \quad (7.35)$$

下面给出具体的推导过程（需要用到两次 Sherman–Morrison 公式）

$$\begin{aligned} \mathbf{B}_{k+1} &= (\mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}) - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \\ &= \mathbf{C} + \beta(\mathbf{B}_k \mathbf{s}_k)(\mathbf{B}_k \mathbf{s}_k)^T \end{aligned}$$

那么（第一遍利用 Sherman–Morrison 矩阵求逆等式）

$$\mathbf{B}_{k+1}^{-1} = \mathbf{C}^{-1} - \frac{\mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k \mathbf{C}^{-1}}{1/\beta + \mathbf{s}_k^T \mathbf{B}_k \mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k}, \quad \beta = -\frac{1}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \quad (7.36)$$

同时（第二遍利用 Sherman–Morrison 矩阵求逆等式）

$$\mathbf{C}^{-1} = \mathbf{B}_k^{-1} - \frac{\mathbf{B}_k^{-1} \mathbf{y} \mathbf{y}^T \mathbf{B}_k^{-1}}{1/\alpha + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k}, \quad \alpha = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \quad (7.37)$$

首先，可以得到式(7.36)中的分母为一个标量：

$$\kappa = -\frac{\mathbf{s}_k^T \mathbf{y}_k \mathbf{y}_k^T \mathbf{s}_k}{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k}$$

而 $\mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k$ 和 $(\mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k)^T / \kappa$ 为

$$\begin{aligned} \mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k &= \mathbf{s}_k - \frac{\mathbf{B}_k^{-1} \mathbf{y}_k \mathbf{y}_k^T \mathbf{s}_k}{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k} \\ \frac{(\mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k)^T}{\kappa} &= \frac{\mathbf{s}_k^T}{\kappa} - \frac{\mathbf{y}_k^T \mathbf{B}_k^{-1}}{\mathbf{s}_k^T \mathbf{y}_k} \end{aligned}$$

结合以上分析，将式(7.37)代入式(7.36)，可得

$$\mathbf{B}_{k+1}^{-1} = \mathbf{B}_k^{-1} - \frac{1}{\kappa} \mathbf{s}_k \mathbf{s}_k^T - \frac{1}{\mathbf{s}_k^T \mathbf{y}_k} \left(\mathbf{B}_k^{-1} \mathbf{y}_k \mathbf{s}_k^T + \mathbf{s}_k \mathbf{y}_k^T \mathbf{B}_k^{-1} \right) \quad (7.38)$$

其中用到了 $\mathbf{y}_k^T \mathbf{s}_k = \mathbf{s}_k^T \mathbf{y}_k$ ，以及 $(\mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k)(\mathbf{C}^{-1} \mathbf{B}_k \mathbf{s}_k)^T / \kappa$ 的展开因子中可将式(7.37)右边第二项对消掉。

¹Sherman–Morrison 公式适用于求 $A + \mathbf{u}\mathbf{v}^t$ 此类包含低秩 correction matrix 的逆。其中 $\Delta A = \mathbf{u}\mathbf{v}^T$ 被称作 perturbation 或者 rank-1 更新，利用 Sherman–Morrison 矩阵求逆等式，可以将 $N \times N$ 的矩阵求逆转换为标量和矩阵的运算，大大减小计算量。

在 BFGS 算法中，我们将式(7.38)中的 \mathbf{B}_k^{-1} 用 \mathbf{D}_k 代替，

$$\begin{aligned}\mathbf{D}_{k+1} &= \mathbf{D}_k - \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \left(\mathbf{D}_k \mathbf{y}_k \mathbf{s}_k^T + \mathbf{s}_k \mathbf{y}_k^T \mathbf{D}_k \right) \\ &\quad + \frac{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{D}_k \mathbf{y}_k}{(\mathbf{y}_k^T \mathbf{s}_k)^2} (\mathbf{s}_k \mathbf{s}_k^T)\end{aligned}\tag{7.39}$$

其中 $\frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$, $\frac{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{D}_k \mathbf{y}_k}{(\mathbf{y}_k^T \mathbf{s}_k)^2}$ 都为常量。在用 BFGS 方法得到 $\mathbf{D}_{k+1} \approx \mathbf{H}_{k+1}^{-1}$ 后，搜索方向 $\mathbf{d}_{k+1} = -\mathbf{D}_{k+1} \mathbf{g}_{k+1}$ 。

例题 7.7: 已知 Sherman–Morrison–Woodbury 矩阵求逆等式，

$$(\mathbf{A} + \mathbf{U} \mathbf{C} \mathbf{V})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{V} \mathbf{A}^{-1} \mathbf{U})^{-1} \mathbf{V} \mathbf{A}^{-1}\tag{7.40}$$

式(7.40)可以将 k 秩的矩阵求逆，转换为 $r < k$ 的矩阵求逆运算。利用 Woodbury 等式可以加速一些矩阵的求逆，比如，在本节 BFGS 算法推导中， \mathbf{B}_k 的更新矩阵（Correction Matrix）是一个 2 秩的矩阵，

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \alpha \mathbf{u} \mathbf{u}^T + \beta \mathbf{v} \mathbf{v}^T$$

接下来，我们将利用 Woodbury 等式(7.40)来推导 BFGS 更新式(7.38)。

首先记

$$\mathbf{U} = [\mathbf{u} \ \mathbf{v}] \in \mathbb{R}^{N \times 2}, \quad \mathbf{C} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}, \quad \mathbf{V} = \mathbf{U}^T\tag{7.41}$$

其中（注意，在 BFGS 中的 $\mathbf{u}, \mathbf{v}, \alpha, \beta$ 跟 DFP 中相比是对调了 $\mathbf{y}_k, \mathbf{s}_k$ 的，即在 BFGS 中 $\mathbf{u}_k = \mathbf{y}_k$, $\mathbf{v}_k = \mathbf{B}_k \mathbf{s}_k$ ）

那么，根据 Woodbury 等式(7.40)，我们可得

$$\mathbf{B}_{k+1}^{-1} = \mathbf{B}_k^{-1} - \mathbf{B}_k^{-1} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{U}^T \mathbf{B}_k^{-1} \mathbf{U})^{-1} \mathbf{U}^T \mathbf{B}_k^{-1}\tag{7.42}$$

首先，

$$(\mathbf{C}^{-1} + \mathbf{U}^T \mathbf{B}_k^{-1} \mathbf{U})^{-1} = \begin{bmatrix} \frac{1}{\alpha} + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k & \mathbf{y}_k^T \mathbf{s}_k \\ \mathbf{s}_k^T \mathbf{y}_k & 0 \end{bmatrix}^{-1}\tag{7.43}$$

$$= \begin{bmatrix} 0 & -\frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \\ -\frac{1}{\mathbf{y}_k^T \mathbf{s}_k} & \frac{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k}{(\mathbf{y}_k^T \mathbf{s}_k)^2} \end{bmatrix}\tag{7.44}$$

进而可以方便的得到

$$\begin{aligned} \mathbf{B}_{k+1}^{-1} &= \mathbf{B}_k^{-1} - \frac{\mathbf{B}_k^{-1} \mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbf{s}_k \mathbf{y}_k^T \mathbf{B}_k^{-1}}{\mathbf{y}_k^T \mathbf{s}_k} \\ &\quad + (\mathbf{s}_k \mathbf{s}_k^T) \frac{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k}{(\mathbf{y}_k^T \mathbf{s}_k)^2} \end{aligned} \quad (7.45)$$

将 \mathbf{B}_k^{-1} 用 \mathbf{D}_k 替代即可得 BFGS 更新式(7.39)。

BFGS 算法的特点是不要求解矩阵的逆，也不要求解线性方程组，仅需要矩阵向量运算即可完成。BFGS 算法和 DFP 算法的不同之处仅在于 \mathbf{D}_{k+1} 的迭代公式。

7.2.4.3 非精确一维搜索算法

另外，感兴趣的同学可以了解一下 λ_k 的一维搜索算法。在 DFP 算法中，我们采用的是精确搜索，即寻找

$$\lambda_k = \arg \min_{\lambda} f(\mathbf{x}_k + \lambda \mathbf{d}_k) \quad (7.46)$$

除此之外，还有像 Wolfe 搜索，Armijo 搜索以及满足 Goldstein 条件的非精确搜索。带非精确搜索的拟牛顿法的研究是从 1976 年 Powell 的工作开始的，他证明了带 Wolfe 搜索的 BFGS 算法的全局收敛性和超线性收敛性。

下面以 Wolfe 搜索 [18] 为例。令 $c_1 \in (0, 1)$, $c_2 \in (c_1, 1)$, Wolfe 搜索寻找 λ_k ，满足 条件一 (Sufficient decrease)：

$$f(\mathbf{x}_k + \lambda_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \lambda_k \mathbf{d}_k^T \mathbf{g}_k \quad (7.47)$$

条件二 (Curvature Condition)：

$$\mathbf{d}_k^T \mathbf{g}(\mathbf{x}_k + \lambda_k \mathbf{d}_k) \geq c_2 \mathbf{d}_k^T \mathbf{g}_k \quad (7.48)$$

那么这两个条件应该怎样去理解呢？见图7.6。条件一 (式(7.47)) 保证了 f 能够在 $\mathbf{x}_k + \lambda_k \mathbf{d}_k$ 方向上，随着 λ_k 的增大，减少足够大的值。式(7.47)右边是一个 λ_k 的线性函数 $l(\lambda_k)$ ，需满足 $\varphi(\lambda_k) \leq l(\lambda_k)$ 。实际应用中， c_1 一般取很小的值 $c_1 = 10^{-4}$ 。条件二 (式(7.48)) 被称作 Curvature Condition。式(7.48)左边 $\mathbf{d}_k \mathbf{g}(\mathbf{x} + \lambda_k \mathbf{d}_k)$ 是 $\varphi(\lambda_k)$ 的导数。所以，式(7.48)的含义是 (注意，求极小值时斜率是负的)， $\varphi'(\lambda_k)$ 要大于起始点处的导数 $\varphi'(0)$ 乘以一个系数 c_2 。如果 $\varphi'(\lambda_k) < c_2 \varphi'(0)$ ，说明我们可以继续往前移动 (增大 λ_k) 来进一步的减小 f

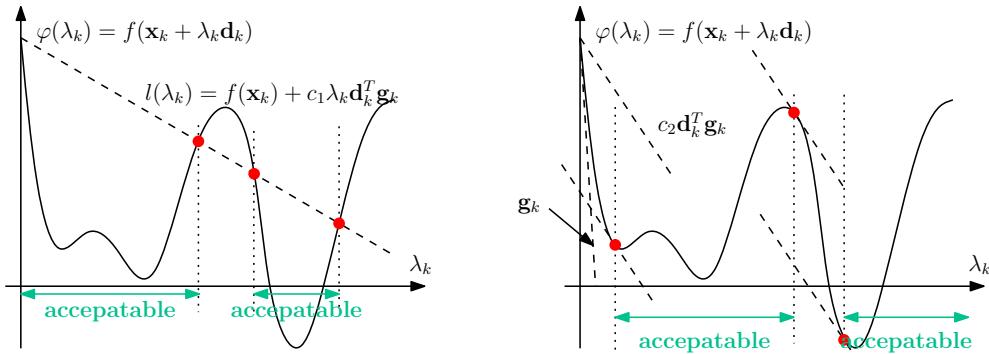


图 7.6 Wolfe 条件。(left) Sufficient decrease condition (right) The curvature condition.

的值。换言之，如果 $\varphi'(\lambda_k)$ 不那么“负”了，甚至变动到正值，说明 f 在这个方向上的减小受阻。此时，我们就可以停止搜索了。 c_2 的典型值一般取 $c_2 = 0.9$ ，而 d_k 正是由拟牛顿法求出的极值方向。

那么，根据图 7.6 的解释， $(0, 1)$ 的界限将很好理解。现在还有一个问题，为什么需要 $c_1 < c_2$ 呢？从图 7.6 可以看出，曲线 $l(\lambda_k)$ 的变化率是 $c_1 \mathbf{d}_k^T \mathbf{g}_k$ ，而式(7.48)的期望斜率（称作 desired slope，或停止斜率）是 $c_2 \mathbf{d}_k^T \mathbf{g}_k$ 。若 $c_1 = c_2$ ，两曲线重合。若 $c_1 > c_2$ ，则存在这样的可能¹，即便我们可以通过拟牛顿法求得函数下降的方向 \mathbf{d}_k ，却无法通过 Wolfe 搜索到合适的 λ_k 。

通过上面的描述，我们可写出 Wolfe 搜索的流程 [20]

7.2.4.4 L-BFGS 算法

在 BFGS 算法中，仍旧需要用到一个 $N \times N$ 的矩阵 \mathbf{D}_k 。当 N 很大的时候，存储和计算这个矩阵都要消耗计算机资源。例如当 $N = 10^5$ 时，假设对系数矩阵采用 double 型数据（8 字节 Byte），所需存储空间为

$$\frac{10^5 \times 10^5 \times 8}{2^{10} \times 2^{10} \times 2^{10}} = 74.5(\text{GB})$$

虽然可以利用矩阵的对称性降低一半的存储空间，但内存占用仍就是惊人的。那么，能否对 BFGS 算法改造，减少迭代过程中所需的内存开销呢？L-BFGS[16] (Limited-memory BFGS 或 Limited-storage BFGS) 就是这样一种算法。其原理是，不用存储 \mathbf{D}_k ，而是存储计算过程中的向量序列 $\{s_k\}$, $\{y_k\}$ 。当需要计算 $\mathbf{d}_k = -\mathbf{D}_k \mathbf{g}_k$ 时，利用向量序列的计算来代替直接的矩阵向量相乘。更进一步，L-BFGS 算法中，向量序列也不用都存；只需每次计算 \mathbf{D}_k ，使用最新的 m 个

¹例如 f 是凸函数，此时式(7.47)的有效区间在曲线左侧，式(7.48)的有效区间在右侧

Algorithm 7.1 二分法 (Bisection Method) 实现 Wolfe 搜索

Input: 选定 $0 < c_1 < c_2 < 1$, 令 $\alpha = 0, \lambda = 1, \beta = +\infty$

```

1: loop
2:   if  $(f(\mathbf{x}_k + \lambda_k \mathbf{d}_k) > f(\mathbf{x}_k) + c_1 \lambda_k \mathbf{d}_k^T \mathbf{g}_k)$  then
3:      $\beta = \lambda$ 
4:      $\lambda = \frac{1}{2}(\alpha + \beta)$ 
5:   else if  $(\mathbf{d}_k^T \mathbf{g}(\mathbf{x}_k + \lambda_k \mathbf{d}_k) < c_2 \mathbf{d}_k^T \mathbf{g}_k)$  then
6:      $\alpha = \lambda$ 
7:     if  $\beta = +\infty$  then
8:        $\lambda = 2\alpha$ 
9:     else
10:       $\lambda = \frac{1}{2}(\alpha + \beta)$ 
11:    end if
12:  else
13:    STOP
14:  end if
15: end loop

```

$\{\mathbf{s}_k\}$ 和 $\{\mathbf{y}_k\}$ 即可。

这里我们定义, 丢弃第 k 个更新矩阵 (Correction Matrix) 等价于 $\mathbf{V}_k = \mathbf{I}$ 以及 $\rho_k \mathbf{s}_k \mathbf{s}_k^T = \mathbf{0}$, 即 $\Delta \mathbf{D}_k = \mathbf{0}$ 。

首先根据式(7.39)可得。

$$\mathbf{D}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) \mathbf{D}_k \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \quad (7.49)$$

$$\text{令 } \mathbf{V}_k = \mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}, \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$$

$$\mathbf{D}_{k+1} = \mathbf{V}_k^T \mathbf{D}_k \mathbf{V}_k + \rho_k \mathbf{s}_k \mathbf{s}_k^T \quad (7.50)$$

式(7.50)在文献 [16] 中被称为 BFGS 的乘积 (Prod) 形式, 而式(7.39)被称为 BFGS 的求和 (Sum) 形式。

下面推导 L-BFGS 的计算式。首先, 我们有

$$\begin{aligned}
\mathbf{D}_{k+1} &= \mathbf{V}_k^T \mathbf{D}_k \mathbf{V}_k + \rho_k \mathbf{s}_k \mathbf{s}_k^T, \\
&= (\mathbf{V}_k^T \mathbf{V}_{k-1}^T) \mathbf{D}_{k-1} (\mathbf{V}_{k-1} \mathbf{V}_k) + (\mathbf{V}_k^T) \rho_{k-1} \mathbf{s}_{k-1} \mathbf{s}_{k-1}^T (\mathbf{V}_k) + \rho_k \mathbf{s}_k \mathbf{s}_k^T
\end{aligned}$$

即（先不考虑 m 的长度限制，使用所有的历史序列来计算 \mathbf{D}_{k+1} ）

$$\begin{aligned}
 \mathbf{D}_{k+1} = & (\mathbf{V}_k^T \cdots \mathbf{V}_0^T) \mathbf{D}_0 (\mathbf{V}_0 \cdots \mathbf{V}_k) \\
 & + (\mathbf{V}_k^T \cdots \mathbf{V}_1^T) \rho_0 \mathbf{s}_0 \mathbf{s}_0^T (\mathbf{V}_1 \cdots \mathbf{V}_k) \\
 & + (\mathbf{V}_k^T \cdots \mathbf{V}_2^T) \rho_1 \mathbf{s}_1 \mathbf{s}_1^T (\mathbf{V}_2 \cdots \mathbf{V}_k) \\
 & + \cdots \\
 & + (\mathbf{V}_k^T) \rho_{k-1} \mathbf{s}_{k-1} \mathbf{s}_{k-1}^T (\mathbf{V}_k) \\
 & + \rho_k \mathbf{s}_k \mathbf{s}_k^T
 \end{aligned} \tag{7.51}$$

在文献 [16] 中，式(7.51)被称作 special BFGS matrices。而我们最终需要计算得到

$$\mathbf{d}_{k+1} = -\mathbf{D}_{k+1} \mathbf{g}_{k+1} \tag{7.52}$$

首先，定义

$$\mathbf{q}_i = \left(\prod_{j=i}^k \mathbf{V}_j \right) \mathbf{g}_{k+1}, \quad i = 0, \dots, k \tag{7.53}$$

$$\alpha_i = \rho_i \mathbf{s}_i^T \mathbf{q}_{i+1} \tag{7.54}$$

将式(7.51)代入 $\mathbf{D}_{k+1} \mathbf{g}_{k+1}$ 中，

$$\begin{aligned}
 \mathbf{D}_{k+1} \mathbf{g}_{k+1} = & (\mathbf{V}_k^T \cdots \mathbf{V}_0^T) \mathbf{D}_0 \mathbf{q}_0 \\
 & + (\mathbf{V}_k^T \cdots \mathbf{V}_1^T) \mathbf{s}_0 \alpha_0 \\
 & + (\mathbf{V}_k^T \cdots \mathbf{V}_2^T) \mathbf{s}_1 \alpha_1 \\
 & + \cdots \\
 & + (\mathbf{V}_k^T) \mathbf{s}_{k-1} \alpha_{k-1} \\
 & + \mathbf{s}_k \alpha_k
 \end{aligned} \tag{7.55}$$

其中 $\mathbf{a}_k = \rho_k \mathbf{s}_k^T \mathbf{g}_{k+1}$ 。则 L-BFGS 的计算示意图 7.7 所示

若已知 $(\mathbf{q}_{i+1}, \alpha_i)$ ，则 \mathbf{q}_i 为（后向计算 $k, k-1, \dots, 0$ ）

$$\begin{aligned}
 \mathbf{q}_i &= \mathbf{V}_i \mathbf{q}_{i+1} \\
 &= \left(\mathbf{I} - \rho_i \mathbf{y}_i \mathbf{s}_i^T \right) \mathbf{q}_{i+1} \\
 &= \mathbf{q}_{i+1} - \alpha_i \mathbf{y}_i
 \end{aligned}$$

则可得

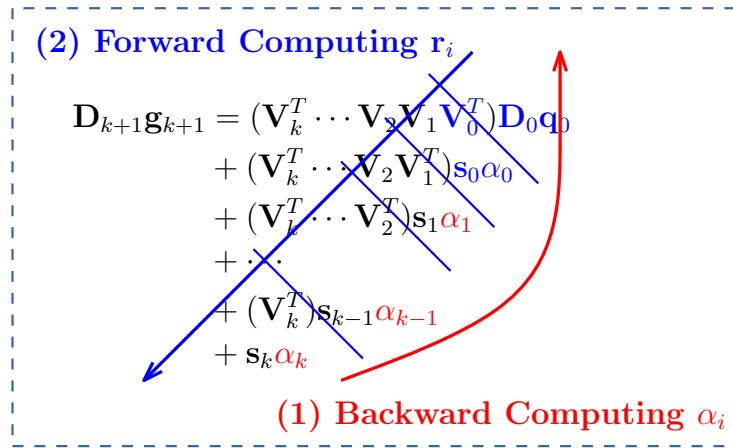


图 7.7 L-BFGS 迭代计算 $\mathbf{d}_{k+1} = -\mathbf{D}_{k+1}\mathbf{g}_{k+1}$ 示意图。首先，我们后向 (backward) 计算系数 $\{\alpha_k, \dots, \alpha_0\}$ ，并储存计算的系数向量 $\{\alpha_i\}$ 与 \mathbf{q}_0 。随后，我们前向 (forward) 计算系数 \mathbf{r}_i ，并且输出计算结果 $\mathbf{d}_{k+1} = -\mathbf{r}_{k+1}$

算法 7.2.1 (L-BFGS 的后向系数更新): 给定 $\alpha_k = \rho_k \mathbf{s}_k^T \mathbf{g}_{k+1}$ 与 $\mathbf{q}_{k+1} = \mathbf{g}_{k+1}$ ，对于 $i \in [k, \dots, 0]$ ，迭代计算：

$$\alpha_i = \rho_i \mathbf{s}_i^T \mathbf{q}_{i+1} \quad (7.56)$$

$$\mathbf{q}_i = \mathbf{q}_{i+1} - \alpha_i \mathbf{y}_i \quad (7.57)$$

序列 $\{\alpha_i\}$ ， $\mathbf{q}_0 = (\mathbf{V}_0 \cdots \mathbf{V}_k) \mathbf{g}_{k+1}$ 需存储下来供前向迭代算法使用。

另一方面，观察式(7.55)，可令

$$\begin{aligned} \mathbf{r}_{i+1} &= \mathbf{V}_i^T \mathbf{r}_i + \mathbf{s}_i \alpha_i \\ &= (\mathbf{I} - \rho_i \mathbf{s}_i \mathbf{y}_i^T) \mathbf{r}_i + \mathbf{s}_i \alpha_i \\ &= \mathbf{r}_i + \mathbf{s}_i (\alpha - \rho_i \mathbf{y}_i^T \mathbf{r}_i) \end{aligned}$$

定义 $\beta_i = \rho_i \mathbf{y}_i^T \mathbf{r}_i$ ，则

算法 7.2.2 (L-BFGS 的前向更新): 给定 $\mathbf{r}_0 = \mathbf{D}_0 \mathbf{q}_0$ ，对于 $i \in [0, \dots, k]$ 迭代计算

$$\beta_i = \rho_i \mathbf{y}_i^T \mathbf{r}_i \quad (7.58)$$

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{s}_i (\alpha - \beta) \quad (7.59)$$

最终计算得到 $\mathbf{d}_{k+1} = -\mathbf{r}_{k+1}$ 。

若只存储最新的 m 组数据，而丢弃之前的 \mathbf{D}_k 更新矩阵，即

$$\begin{aligned} V_i &= \mathbf{I}, \quad \forall i < m \\ \rho_i s_i s_i^T &= \mathbf{0}, \quad \forall i < m \end{aligned}$$

式(7.51)可写为

$$\begin{aligned} \mathbf{D}_{k+1} &= (V_k^T \cdots V_{k-m+1}^T) \mathbf{D}_0 (V_{k-m+1} \cdots V_k) \\ &\quad + (V_k^T \cdots V_{k-m+2}^T) \rho_{k-m+1} s_{k-m+1} s_{k-m+1}^T (V_{k-m+2} \cdots V_k) \\ &\quad + (V_k^T \cdots V_{k-m+3}^T) \rho_{k-m+2} s_{k-m+2} s_{k-m+2}^T (V_{k-m+3} \cdots V_k) \\ &\quad + \cdots \\ &\quad + (V_k^T) \rho_{k-1} s_{k-1} s_{k-1}^T (V_k) \\ &\quad + \rho_k s_k s_k^T \end{aligned} \tag{7.60}$$

同学们可在此基础上，根据 L-BFGS 的前向、后向计算公式，推导出有限存储（深度为 m ）下的 L-BFGS 算法。

7.2.5 实施方法及实例讲解

7.2.5.1 实施方法

这里再次强调下，牛顿法和拟牛顿法的初衷是解决无约束的最优化问题

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{7.61}$$

而无论牛顿法或者拟牛顿法，都需要给定收敛条件 ϵ ，当

$$\|\mathbf{g}_k\| < \epsilon \tag{7.62}$$

的时候退出算法。

(1) 牛顿法的实现

牛顿法需要给定迭代的初值 \mathbf{x}_0 ，并且需要显式的给出 \mathbf{g}_k 和 \mathbf{H}_k ，或者给出 $g(\mathbf{x})$ 和 $H(\mathbf{x})$ 的函数调用。以最基本的阻尼牛顿法为例，阻尼因子 λ_k 通过一维最优搜索得到。该算法的实现流程为。

(2) DFP 拟牛顿法的实现

DFP 拟牛顿法通过迭代更新计算 \mathbf{D}_k ，因此在牛顿法的基本框架下，还需要指定 \mathbf{D}_0 。另外 DFP 法的 λ_k 可以采用多种非精确搜索方法得到。下面以简单的一维最优搜索为例。

Algorithm 7.2 阻尼牛顿法

Input: 给定初值 \mathbf{x}_0 和收敛精度 ϵ , 令 $k = 0$

- 1: **repeat**
- 2: 计算 \mathbf{g}_k 和 \mathbf{H}_k
- 3: 计算搜索方向 $\mathbf{d}_k = -\mathbf{H}_k \mathbf{g}_k$
- 4: 一维搜索 $\lambda_k = \arg \min_{\lambda} f(\mathbf{x}_k + \lambda \mathbf{d}_k)$
- 5: $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k$
- 6: $k = k + 1$
- 7: **until** $\|\mathbf{g}_k\| < \epsilon$

Algorithm 7.3 DFP 拟牛顿法

Input: 给定初值 \mathbf{x}_0 和精度阈值 ϵ , 令 $\mathbf{D}_0 = \mathbf{I}$, $k = 0$

- 1: **repeat**
- 2: 确定搜索方向 $\mathbf{d}_k = -\mathbf{D}_k \mathbf{g}_k$
- 3: 一维搜索 λ_k , 令 $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k$, $\mathbf{s}_k = \lambda_k \mathbf{d}_k$
- 4: 计算 $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$
- 5: 计算 $\mathbf{D}_{k+1} = \mathbf{D}_k + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k} - \frac{\mathbf{D}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{D}_k}{\mathbf{y}_k^T \mathbf{D}_k \mathbf{y}_k}$
- 6: $k = k + 1$
- 7: **until** $\|\mathbf{g}_{k+1}\| < \epsilon$

(3) BFGS 拟牛顿法的实现

BFGS 算法与 DFP 法类似, 区别只在于 \mathbf{D}_k 的更新公式。这里需要着重强调的是, BFGS 算法用到了矩阵求逆等式, 将 $N \times N$ 矩阵求逆运算转换为标量的求逆, 优化了计算资源。这里, 我们仅仅列出 BFGS 算法和 DFP 算法的不同之处。

Algorithm 7.4 BFGS 拟牛顿法

- 1: **repeat**
- 2: 可利用 Wolfe 搜索得到 λ_k
- 3: 计算 $\mathbf{D}_{k+1} = \mathbf{D}_k - \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} (\mathbf{D}_k \mathbf{y}_k \mathbf{s}_k^T + \mathbf{s}_k \mathbf{y}_k^T \mathbf{D}_k) + \frac{\mathbf{y}_k^T \mathbf{s}_k + \mathbf{y}_k^T \mathbf{D}_k \mathbf{y}_k}{(\mathbf{y}_k^T \mathbf{s}_k)^2} (\mathbf{s}_k \mathbf{s}_k^T)$
- 4: **until** $\|\mathbf{g}_{k+1}\| < \epsilon$

(4) 存储优化的 L-BFGS 实现

L-BFGS 算法是存储优化 (有限存储条件下) 的 BFGS 算法, 跟 BFGS 算法对比, 区别仅在于 \mathbf{d}_k 的计算方法。这里设定 L-BFGS 算法的存储长度为 L 。同

时, 为了算法描述方便, 我们假定内存中的 s_i 和 y_i 都统一的以 $[0, \dots, L - 1]$ 进行编号 (相当于一个先入先出 FIFO 系统), 例如

$$\begin{array}{ccccccccc} \cdots & (k-L+1) & (k-L+2) & \cdots & (k-1) & k & \cdots \\ & 0 & & & 1 & & \cdots & (L-2) & (L-1) \end{array}$$

那么, L-BFGS 算法的核心为通过后向计算和前向更新, 得到 d_k 。

Algorithm 7.5 L-BFGS 拟牛顿法

Input: 存储深度 L , $q_L = g_k$

Output: 后向计算 α_i , q_0

- 1: **for** $i = [L - 1, L - 2, \dots, 0]$ **do**
- 2: $\alpha_i = \rho_i s_i^T q_i$
- 3: $q_i = q_{i+1} - \alpha_i y_i$
- 4: **end for**

Input: 后向计算的系数 $\{\alpha_i\}$ 及 q_0

Output: 前向计算 r_L

- 5: $r_0 = D_0 q_0$
 - 6: **for** $i = [0, 1, \dots, L - 1]$ **do**
 - 7: $\beta_i = \rho_i y_i^T r_i$
 - 8: $r_{i+1} = r_i + s_i (\alpha_i - \beta_i)$
 - 9: **end for**
 - 10: 输出 $d_k = -r_L$
-

在给出了以上各个算法的实施框架之后, 编写牛顿法或拟牛顿法的计算程式就简单的多了。编程时, 我们可以将阻尼系数 λ_k 的一维搜索 (非精确线搜索算法)、 d_k 的计算公式编写成子函数的形式, 这样可以方便的从最基本的牛顿法, 逐步更新到 DFP、BFGS 以至 L-BFGS 算法。

需要注意的是, 在编程中, 不要被细节所打败 (Don't bury in the details)。直接用 C 语言实现, 需要处理好内存分配 (alloc) 和释放 (free), 对程序编写能力提出较高的要求。我们可以从高阶语言开始, 如从 MATLAB、到 Python、再到 C++。这里强烈推荐使用 Eigen 库¹ 在 C++ 下编写矩阵向量运算的高效程序。

¹ 见 <http://eigen.tuxfamily.org/>

7.2.5.2 计算实例讲解

例题 7.8: 寻找以下这个函数的极小值

$$f(\mathbf{x}) = e^{x_1-1} + e^{-x_2+1} + (x_1 - x_2)^2 \quad (7.63)$$

Iteration 1:

初始化

$$\mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

令 $\mathbf{D}_0 = \mathbf{I}$ 单位矩阵, 则 \mathbf{g}_0 为

$$\mathbf{g}_0 = \begin{pmatrix} e^{x_1-1} + 2(x_1 - x_2)|_{\mathbf{x}_0} = 0.3679 \\ -e^{x_2+1} - 2(x_1 - x_2)|_{\mathbf{x}_0} = -2.7183 \end{pmatrix}$$

及 $\mathbf{d}_0 = -\mathbf{D}_0 \mathbf{g}_0$ 。

假定 Wolfe 搜索得到 $\lambda_0 = 1$, 接下来需计算 \mathbf{x}_1

$$\mathbf{x}_1 = \mathbf{x}_0 + \lambda_0 \mathbf{d}_0 = \begin{pmatrix} -0.3679 \\ 2.7183 \end{pmatrix}$$

和

$$\mathbf{g}_1 = \begin{pmatrix} -5.9178 \\ 5.9930 \end{pmatrix}$$

以及 $\mathbf{s}_0 = \lambda_0 \mathbf{d}_0$ 和

$$\mathbf{y}_0 = \mathbf{g}_1 - \mathbf{g}_0 = \begin{pmatrix} -6.2856 \\ 8.7113 \end{pmatrix}$$

最后可得

$$\begin{aligned} \mathbf{D}_1 &= \mathbf{D}_0 - \frac{1}{\mathbf{y}_0^T \mathbf{s}_0} (\mathbf{D}_0 \mathbf{y}_0 \mathbf{s}_0^T + \mathbf{s}_0 \mathbf{y}_0^T \mathbf{D}_0) + \frac{\mathbf{y}_0^T \mathbf{s}_0 + \mathbf{y}_0^T \mathbf{D}_0 \mathbf{y}_0}{(\mathbf{y}_0^T \mathbf{s}_0)^2} (\mathbf{s}_0 \mathbf{s}_0^T) \\ &= \begin{pmatrix} 0.8504 & 0.5714 \\ 0.5714 & 0.7243 \end{pmatrix} \end{aligned}$$

同学们可以自己编程, 实现本例题的最小程序。

7.2.5.3 编程实例讲解

在数学最优化问题中，Rosenbrock 函数是一个用来测试最优化算法性能的非凸函数，由 Howard Harry Rosenbrock 在 1960 年提出。该函数的形状见下图。该函数也被称作 Rosenbrock 山谷函数，或 Rosenbrock 香蕉函数。

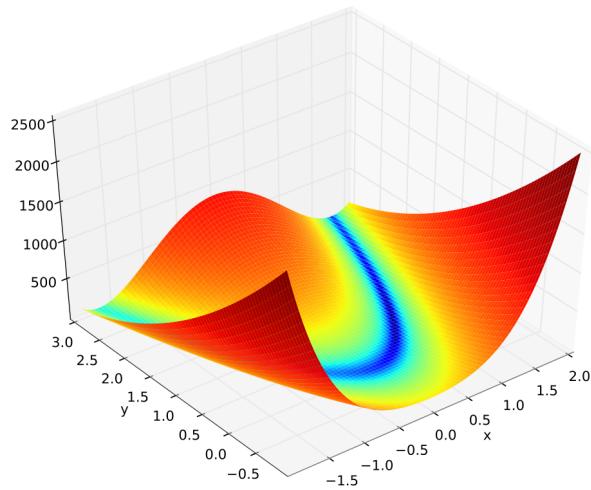


图 7.8 Rosenbrock 函数示意图

Rosenbrock 函数定义如下：

$$f(x, y) = (1 - x^2) + 100(y - x^2)^2$$

该函数的每个等高线大致呈抛物线形，其全局极小值就在山谷中（香蕉形山谷）。但是，由于山谷中函数值的变化不大，寻找全局最小值相当困难。直接分析代价函数 $f(x)$ 可发现，其全局最小值位于点 $(x, y) = (1, 1)$ 处，函数值为 $f(x, y) = 0$ 。有时 Rosenbrock 函数的第二项系数不同，但是不影响全局最小值的位置。

7.2.6 课后习题

7.2.6.1 习题

习题 7.1：证明牛顿法对于二次函数只需要一次迭代即可收敛到极值点 \mathbf{x}^* 。

习题 7.2：证明拟牛顿法中 BFGS 更新后矩阵 $\mathbf{D}_{k+1} = \mathbf{D}_k + \Delta\mathbf{D}_k$ 的正定性。提示，可以（1）从 BFGS 的乘积（prod）形式出发或（2）从 L-BFGS 的 Special BFGS Matrices 中寻找。也可以直接根据定义展开并证明

$\mathbf{d}_k \mathbf{g}_k < 0$ 。需要注意的是，对于凸函数 f ，我们有 $\mathbf{y}^T \mathbf{s} > 0$ 。

7.2.6.2 上机实验

上机 7.1：查阅 Python 工具包 Scipy 中 bfgs 优化方法的调用接口，会运行 minimize 函数中的 rosen 实例。

上机 7.2：牛顿法（拟牛顿法）也可以用来解决下面这个优化问题

$$\min \mathbf{c}^T \mathbf{x} - \sum_{i=1}^m \log(b_i - \mathbf{a}_i^T \mathbf{x})$$

编写牛顿法，DFP 法，BFGS 法，绘制出 $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ 和随着迭代步数 k 变化的曲线。并解释实验结果，试分析迭代收敛次数，单步迭代时间等结果及其成因。

7.3 Jacobian 矩阵

在上一节中，我们讲解了牛顿优化方法。该方法针对无约束的最小化问题

$$\min_{\mathbf{x}} f(\mathbf{x})$$

通过迭代计算

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_k^{-1} \mathbf{g}_k$$

进行求解，其中 \mathbf{H}_k 和 \mathbf{g}_k 分别是多元函数 $f(\mathbf{x})$ 的 Hessian 矩阵和梯度向量。

实际应用中 Hessian 矩阵是很难求解的，除了拟牛顿算法之外，针对一类特殊的非线性最小二乘问题，还使用高斯牛顿（Gauss–Newton）算法进行求解。

7.3.1 储备知识：范数与求导

令 \mathbf{x} 、 \mathbf{y} 分别代表长度为 n 、 m 的向量（列向量）

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

而 \mathbf{y} 的每一个元素，都可写为 \mathbf{x} 的函数

$$y_i = y_i(\mathbf{x})$$

当 $n = 1$ 时， x 为标量；当 $m = 1$ 时， y 为标量。

当 \mathbf{y} 为标量、 \mathbf{x} 为向量时，

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{pmatrix} \quad (7.64)$$

最基本的求导公式（向量 - 向量求导）定义为

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \quad (7.65)$$

其中 $\partial \mathbf{y} / \partial \mathbf{x}$ 中的第 i 列为 $\partial y_i / \partial \mathbf{x}$ ，第 j 行为 $\partial y / \partial x_j$ 。

需要注意的是，当 \mathbf{x} 为向量时，我们定义

$$\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^T = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^T}$$

当 \mathbf{y} 为向量、 \mathbf{x} 为标量时，

$$\frac{\partial \mathbf{y}}{\partial x} = \left(\frac{\partial \mathbf{y}}{\partial x} \quad \frac{\partial \mathbf{y}}{\partial x} \quad \cdots \quad \frac{\partial \mathbf{y}}{\partial x} \right) \quad (7.66)$$

例题 7.9: Hessian 矩阵定义为

$$\frac{\partial^2 y}{\partial \mathbf{x} \partial \mathbf{x}^T}$$

其中 y 是标量， \mathbf{x} 是向量。试用求导的基本定义写出 Hessian 矩阵的表达式。

解：首先 Hessian 矩阵等价为计算

$$\mathbf{z} = \frac{\partial y}{\partial \mathbf{x}}, \quad \mathbf{H} = \frac{\partial \mathbf{z}}{\partial \mathbf{x}^T}$$

则 \mathbf{H} 中的每一行 (由于是计算 $\partial \mathbf{x}^T$) 为 $\partial z_i / \partial \mathbf{x}$, 而 $z_i = \partial y / \partial x_i$, 那么

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 y}{\partial x_1^2} & \frac{\partial^2 y}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 y}{\partial x_1 \partial x_N} \\ \frac{\partial^2 y}{\partial x_2 \partial x_1} & \frac{\partial^2 y}{\partial x_2^2} & \cdots & \frac{\partial^2 y}{\partial x_2 \partial x_N} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial^2 y}{\partial x_N \partial x_1} & \frac{\partial^2 y}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 y}{\partial x_N^2} \end{pmatrix}$$

例题 7.10: 直接用求导定义, 计算

$$\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}}$$

解: 设 $\mathbf{y} = \mathbf{A}\mathbf{x}$, 那么

$$y_j = \mathbf{A}_j^T \mathbf{x} = \sum_i A_{ji} x_i$$

其中 \mathbf{A}_j 为 \mathbf{A} 的第 j 行。那么求导矩阵的第 ij 个元素为

$$\left[\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} \right]_{ij} = \frac{\partial y_j}{\partial x_i} = A_{ji}$$

直接用定义可得: \mathbf{A}^T

例题 7.11: 计算

$$\frac{\partial \mathbf{x}^T \mathbf{A}\mathbf{x}}{\partial \mathbf{x}}$$

解: 首先 $\mathbf{x}^T \mathbf{A}\mathbf{x}$ 为一个标量, 所以求导结果应为一个向量。不失一般性, 我们将 \mathbf{A} 写为

$$\mathbf{A} = \mathbf{B}^T \mathbf{C}$$

那么用分部求导可得

$$\frac{\partial (\mathbf{B}\mathbf{x})^T \mathbf{C}\mathbf{x}}{\partial \mathbf{x}} = \mathbf{B}^T \mathbf{C}\mathbf{x} + (\mathbf{x}^T \mathbf{B}^T \mathbf{C})^T = \mathbf{A}\mathbf{x} + \mathbf{A}^T \mathbf{x}$$

当 \mathbf{A} 为对称矩阵时

$$\frac{\partial \mathbf{x}^T \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}$$

例题 7.12: 令矩阵 A 的每一项都为 x 的函数, 求

$$\frac{\partial A^{-1}}{\partial x}$$

解: 根据 $AA^{-1} = I$, 可得

$$A \frac{\partial A^{-1}}{\partial x} + \frac{\partial A}{\partial x} A^{-1} = \frac{\partial I}{\partial x} = 0$$

则

$$\frac{\partial A^{-1}}{\partial x} = A^{-1} \frac{\partial A}{\partial x} A^{-1}$$

接下来我们引入 Jacobian 矩阵的定义, 需用到向量对向量求导的定义。对于多元函数 $r \in \mathbb{R}^M$, 每一项都为向量 $x \in \mathbb{R}^N$ 的函数, 即

$$r_i = f_i(x)$$

定义 r 的 Jacobian 矩阵 $J_r \in \mathbb{R}^{M \times N}$ 为¹

$$J_r = \frac{\partial r}{\partial x^T} = \begin{pmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_N} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \cdots & \frac{\partial r_2}{\partial x_N} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial r_M}{\partial x_1} & \frac{\partial r_M}{\partial x_2} & \cdots & \frac{\partial r_M}{\partial x_N} \end{pmatrix}$$

跟向量 - 向量求导相比, 有

$$J = \frac{\partial r}{\partial x^T}, \quad J^T = \frac{\partial r}{\partial x}$$

除此以外, Jacobian 矩阵的每一行、每一列为

1. Jacobian 的第 ij 个元素: $J_{ij} = \frac{\partial r_i}{\partial x_j}$
2. Jacobian 的第 i 行: $J_{i \cdot} = \frac{\partial r_i}{\partial x}$
3. Jacobian 的第 i 列: $J_{\cdot i} = \frac{\partial r}{\partial x_i}$

在计算有限元逆问题的 Jacobian 矩阵时会用到上述的定义。

7.3.2 Gauss–Newton (高斯牛顿) 算法

¹在文献中, Jacobian 矩阵也被称作扰动 (perturbation matrix) 矩阵

7.3.2.1 数学模型

Gaussian–Newton 算法可用来寻找非线性最小二乘 (sum of squares) 此类问题的极小值。对于函数 $y = f(x)$, 若已知真实观测 y 和一个非线性算子 f 。我们定义 r 为真实观测和数值解之间的差, 也被称作余量 (Residuals) :

$$r = y - f(x)$$

典型的优化目标即使得迭代解的误差最小, 也就是希望最小化余量, 即

$$\min_x \|r\|_2^2, \quad \|r\|_2^2 = \sum_i^M r_i^2(x)$$

或

$$\min_x \|y - f(x)\|_2^2 \tag{7.67}$$

通过拟合最小均方误差, 我们预期求得信号 x 的估计。针对这一模型, 可方便的通过 Guass–Newton 迭代求得非线性函数的极小值。

一个典型的例子就是电磁场的数值计算模型。例如, 课程中所学的有限元方法就可看作是一个典型的算子, 节点上的电势可写为

$$v = K^{-1}(\sigma)b$$

那么电磁场中的最小化问题就是

$$\min_x \|v^* - K^{-1}(\sigma)b\|_2^2$$

这个模型就是后面将要讲到的电磁场逆问题的数学模型。

7.3.2.2 方法推导

Newton 迭代法是寻找

$$x_{k+1} = x_k - H^{-1}g \tag{7.68}$$

其中 g 为 Gradient 向量, H 为 Hessian 矩阵。为了最小化:

$$\min_x \sum_i^M r_i^2(x)$$

可分别计算 Gradient 和 Hessian 矩阵,

$$g_j = 2 \sum_{i=1}^M r_i \frac{\partial r_i}{\partial x_j} \quad (7.69)$$

$$H_{jk} = 2 \sum_{i=1}^M \left(\frac{\partial r_i}{\partial x_j} \frac{\partial r_i}{\partial x_k} + r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k} \right) \quad (7.70)$$

Gauss–Newton 算法的技巧是忽略 H_{jk} 的第二项。这么做的原因是第二项为二阶导乘以余量 r_i , 在优化的过程中余量都比较小, 而且在极值点 (优化算法的终点) 附近, 余量的期望值是 0, 进而

$$r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k} \ll \frac{\partial r_i}{\partial x_j} \frac{\partial r_i}{\partial x_k} \quad (7.71)$$

那么,

$$H_{jk} = 2 \sum_{i=1}^M \frac{\partial r_i}{\partial x_j} \frac{\partial r_i}{\partial x_k} = 2 \sum_{i=1}^M J_{ij} J_{ik} \quad (7.72)$$

其中 $J_{ij} = \frac{\partial r_i}{\partial x_j}$ 为 Jacobian 矩阵 \mathbf{J}_r 的第 ij 项。进而

$$g_j = 2 \sum_{i=1}^M r_i \frac{\partial r_i}{\partial x_j} = 2[\mathbf{J}_r]_j^T \mathbf{r} \quad (7.73)$$

$$H_{jk} = 2 \sum_{i=1}^M \frac{\partial r_i}{\partial x_j} \frac{\partial r_i}{\partial x_k} = 2[\mathbf{J}_r]_j^T [\mathbf{J}_r]_k \quad (7.74)$$

其中 $[\mathbf{J}_r]_i$ 代表 Jacobian 矩阵 \mathbf{J}_r 的第 i 列 (列向量), 可得

$$\mathbf{g} = 2\mathbf{J}_r^T \mathbf{r} \quad (7.75)$$

$$\mathbf{H} \approx 2\mathbf{J}_r^T \mathbf{J}_r \quad (7.76)$$

代入 Newton 迭代公式(7.68)中, 可得

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}_r^T \mathbf{J}_r)^{-1} \mathbf{J}_r^T \mathbf{r} \quad (7.77)$$

利用式(7.77), 只需计算 \mathbf{J}_r , 即可迭代的求得该非线性问题的极小值。

7.3.2.3 正则化 Gauss–Newton 法

类似的, 我们可以将 Gauss–Newton 法的代价函数修改为

$$\min_{\mathbf{x}} \|\mathbf{r}\|_2^2 + \alpha \|\mathbf{x}\|_2^2$$

或者

$$\min_{\mathbf{x}} \|\mathbf{r}\|_2^2 + \alpha \|\mathbf{Lx}\|_2^2$$

前者为标准的 Tikhonov 正则化，后者为通用的 Tikhonov 正则化方法。

在正则化约束下，Gauss–Newton 法的迭代计算公式为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{J}_r^T \mathbf{J}_r + \alpha \mathbf{L}^T \mathbf{L} \right)^{-1} \mathbf{J}_r^T \mathbf{r}$$

加入正则化约束 $\alpha \mathbf{L}^T \mathbf{L}$ ，可以避免矩阵求逆的病态性。

7.3.2.4 优劣对比

Gauss–Newton 法（后续简称 **gn** 算法）与 Newton 法的区别在于，

1. **gn** 算法（也被称作 Levenberg–Marquardt (**lm**) 算法）只能寻找非线性最小二乘问题 (nonlinear least squares problems)；Newton 算法适用于任何函数的极小值问题；
2. **gn** 算法不需要计算二阶导数，只需计算一阶导数 (Jacobian 矩阵) 对其近似即可；而 Newton 算法需要计算代价函数的 Hessian 矩阵；
3. **gn** 算法由于对二阶导引入了近似条件，不能保证算法收敛；而 Newton 算法要求 Hessian 矩阵满足正定性，即可收敛。

7.3.2.5 计算示例

例题 7.13：我们再回到优化 Rosenbrock 函数上来

$$\min_{(x,y)} f(x,y) \quad f(x,y) = 100(y - x^2)^2 + (1 - x)^2$$

提示：牛顿法针对下式进行优化

$$\min_{\mathbf{x}} f(\mathbf{x})$$

而高斯牛顿法仅用于求解非线性最小二乘问题

$$\min_{\mathbf{x}} \|\mathbf{r}\|_2^2, \quad \|\mathbf{r}\|_2^2 = \mathbf{r}^T \mathbf{r}$$

关键在于将本例题中的 $f(x,y)$ 写成向量 \mathbf{r} 内积的形式。

解：我们将 $f(x,y)$ 写成非线性最小二乘的形式

$$\mathbf{r} = \begin{bmatrix} 10(y - x^2) \\ (1 - x) \end{bmatrix}$$

那么 Jacobian 矩阵为

$$\mathbf{J}_r = \begin{pmatrix} \frac{\partial r_1}{\partial x} & \frac{\partial r_1}{\partial y} \\ \frac{\partial r_2}{\partial x} & \frac{\partial r_2}{\partial y} \end{pmatrix} = \begin{pmatrix} -20x & 10 \\ -1 & 0 \end{pmatrix}$$

用牛顿法计算得到的 Hessian 矩阵为

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{pmatrix}$$

用高斯牛顿法近似得到的 Hessian 矩阵为

$$\mathbf{H} = 2\mathbf{J}^T \mathbf{J} = \begin{pmatrix} 800x^2 - 2 & -400x \\ -400x & 200 \end{pmatrix}$$

在极值点 $(1, 1)$ 处，两者得到的 Hessian 矩阵相等。

7.3.3 有限元逆问题中的 Jacobian 矩阵

7.3.3.1 逆问题的数学模型

电磁场数值计算的逆问题即：已知某一激励下的观测向量 \mathbf{v} ，求区域 Ω 上的电导率 σ 的分布。我们可以将这一逆问题，转化为最小化

$$\boldsymbol{\sigma}^* = \arg \min \|\mathbf{v} - f(\boldsymbol{\sigma}, \Omega)\|_2^2$$

其中 Ω 代表区域形状及网格结构， f 为将电磁场参数 $(\boldsymbol{\sigma}, \Omega)$ 正向映射到测量 \mathbf{v} 的非线性投影算子。没错， f 就是电磁场正问题的数值计算模型。

以有限元方法为例，

$$\boldsymbol{\sigma}^* = \arg \min \|\mathbf{v} - \mathbf{K}(\boldsymbol{\sigma}, \Omega)^{-1} \mathbf{b}\|_2^2$$

其中 \mathbf{v} 是观测向量， \mathbf{b} 是已知的边界激励条件。

求解非线性最小化问题，就可以用到本节课讲到的高斯牛顿算法。那么，只需计算余量

$$\mathbf{r} = \mathbf{v} - \mathbf{K}(\boldsymbol{\sigma}, \Omega)^{-1} \mathbf{b}$$

的 Jacobian 矩阵 \mathbf{J} 。

更进一步，假定在求解过程中，区域形状 Ω （或者有限元网格结构误差）

导致的扰动量很少¹，则

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \boldsymbol{\sigma}} = \mathbf{K}(\boldsymbol{\sigma})^{-1} \frac{\partial \mathbf{K}(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}} \mathbf{K}(\boldsymbol{\sigma})^{-1} \mathbf{b}$$

这里 $\mathbf{K}(\boldsymbol{\sigma})^{-1} \mathbf{b}$ 就是有限元正问题计算方法。类比于欧姆定律 $V = RI$ ：当边界条件 \mathbf{b} 为电流，计算节点电势 \mathbf{v} ， $\mathbf{K}(\boldsymbol{\sigma})^{-1}$ 也就被称为节点阻抗矩阵。

下面我们重点讲有限元模型下 Jacobian 矩阵的计算，尤其是

$$\frac{\partial \mathbf{K}(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}}$$

为了便于理解，我们将仅仅考虑如何计算第 i 个面元电导率的扰动量

$$\frac{\partial \mathbf{K}(\boldsymbol{\sigma})}{\partial \sigma_i}$$

7.3.3.2 有限元系数矩阵的计算方法（回顾）

在计算 Jacobian 矩阵之前，我们先回顾下系数矩阵 \mathbf{K} 的计算步骤。计算 K_{ij} 时，可以分别计算每一个元 e 上的 \mathbf{K}^e ，然后通过叠加的形式得到

$$K_{ij} = \sum_e^{n_E} \int_{\Omega^e} \sigma_e \cdot (\nabla \varphi_i \cdot \nabla \varphi_j) d\Omega = \sum_e^{n_E} K_{IJ}^e \quad (7.78)$$

其中 n_E 为总面元数，其中每个面元 e 上的 K_{IJ}^e 为，

$$K_{IJ}^e = \int_{\Omega^e} \sigma_e \cdot (\nabla \varphi_I \cdot \nabla \varphi_J) d\Omega$$

随后，将 K_{IJ}^e 组装（assemble）进最终的系数矩阵 K_{ij} 中即可。对于三角形面元， \mathbf{K}_{IJ}^e 的大小为 3×3 。

当采用二维中线性的基函数时，我们可以用重心坐标系，解析的计算出 \mathbf{K}_{IJ}^e 。首先，节点上的基函数 $\varphi_i^e(\mathbf{r})$ 可以利用面积坐标表示²，即

$$\varphi_i^e(\mathbf{r}) = \frac{A_i^e}{A_{tot}^e}$$

而面积 A_i^e 可以写为

$$A_i^e = \frac{1}{2} (\mathbf{r} - \mathbf{r}_{i+1}^e) \cdot \hat{\mathbf{z}} \times \mathbf{s}_i$$

¹实际上，有关区域形状，或者更具体的说是电极位移的扰动，是当前研究的一个重点问题。关于电极位移 \mathbf{r} 扰动矩阵的计算，可参考 [21]

²这时 φ_i^e 也被称作 simplex 坐标或者重心坐标（barycentric coordinates）。

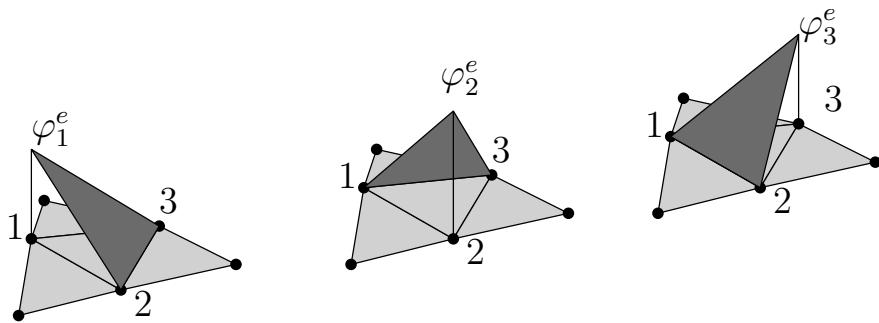
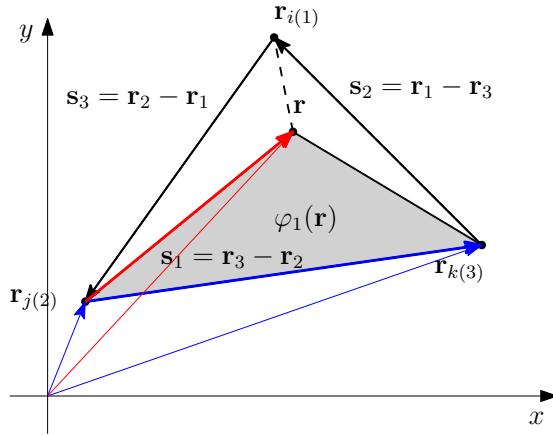
图 7.9 三角形面元 e 的三个基函数，与 e 共用相邻边界的面元也绘于图中。

图 7.10 重心坐标与面元的解析计算公式

其中

$$\mathbf{s}_i = \mathbf{r}_{i-1} - \mathbf{r}_{i+1}$$

为逆时针方向上三角形第 i 条边所代表的向量。

另一方面，总面积 A_{tot}^e 可表示为

$$A_{tot}^e = \frac{1}{2} \hat{\mathbf{z}} \cdot \mathbf{s}_2 \times \mathbf{s}_3$$

则我们可以将 $\nabla \varphi_i^e$ 写为

$$\nabla \varphi_i^e = \frac{\hat{\mathbf{z}} \times \mathbf{s}_i}{2A_{tot}^e}$$

面元 e 上的系数矩阵 K_{IJ}^e 为

$$K_{IJ}^e = \int_{\Omega^e} \sigma_e \cdot \nabla \varphi_i^e \cdot \nabla \varphi_j^e d\Omega = \sigma_e \cdot \frac{\mathbf{s}_I \cdot \mathbf{s}_J}{4A_{tot}^e} \quad (7.79)$$

在 FEM 方法中，面元 e 上系数矩阵 \mathbf{K}_{IJ}^e 的计算方法为

$$\mathbf{s}_i = \mathbf{r}_{i-1} - \mathbf{r}_{i+1} \quad (7.80)$$

$$\nabla \varphi_i^e = \frac{\hat{\mathbf{z}} \times \mathbf{s}_i}{2A_{tot}^e} \quad (7.81)$$

$$A_{tot}^e = \frac{1}{2} \hat{\mathbf{z}} \cdot \mathbf{s}_2 \times \mathbf{s}_3 \quad (7.82)$$

$$K_{IJ}^e = \int_{S^e} \sigma_e \nabla \varphi_I^e \cdot \nabla \varphi_J^e dS = \sigma_e \frac{\mathbf{s}_I \cdot \mathbf{s}_J}{4A_{tot}^e} \quad (7.83)$$

其中 $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ 为面元 e 三个顶点的坐标，依照逆时针方向排序。式(7.83)的计算方法也被称作系数矩阵的解析计算方法。

例题 7.14: 将 FEM 中系数矩阵的解析（基于重心坐标系）计算过程描述成矩阵的形式。

我们首先分析面元 e 的局部坐标系，见图 7.11。首先，对于 \mathbf{r}_1 上的基函数

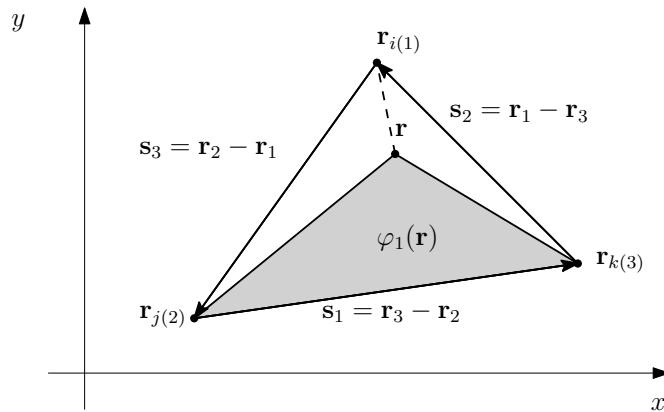


图 7.11 面元 e 的局部坐标系，其中 i, j, k 代表全局节点编号， $1, 2, 3$ 为局部坐标系下的节点编号。节点 $\mathbf{r}_{i(1)}$ 上线性检验函数 $\varphi_i(\mathbf{r})$ （灰色高亮部分）可用重心坐标（barycentric coordinates）表示。

$\varphi_1(\mathbf{r})$ ，有 ($\hat{\mathbf{z}} \times \mathbf{s}$ 即相当于将向量 \mathbf{s} 逆时针旋转 $\pi/2$)

$$\varphi_1(\mathbf{r}) = \frac{(\mathbf{r} - \mathbf{r}_2) \cdot \hat{\mathbf{z}} \times \mathbf{s}_1}{2A_{tot}}$$

则

$$\nabla \varphi_1^e = \frac{\hat{\mathbf{z}} \times \mathbf{s}_1}{2A_{tot}^e}$$

进而

$$S_{ij} = \int_{S_e} \nabla \varphi_i \cdot \nabla \varphi_j dS = \frac{\mathbf{s}_i \cdot \mathbf{s}_j}{4A_{tot}}$$

而对于三角形面积 A_{tot}^e , 我们有

$$\begin{aligned} 2A_{tot}^e &= \hat{z} \cdot \mathbf{s}_2 \times \mathbf{s}_3 = \hat{z} \cdot \mathbf{s}_3 \times \mathbf{s}_1 \\ &= \hat{z} \cdot (\mathbf{r}_2 - \mathbf{r}_1) \times (\mathbf{r}_3 - \mathbf{r}_2) \\ &= \hat{z} \cdot (\mathbf{r}_2 \times \mathbf{r}_3 - \mathbf{r}_1 \times \mathbf{r}_3 + \mathbf{r}_1 \times \mathbf{r}_2) \\ &= \begin{vmatrix} 1 & r_x^1 & r_y^1 \\ 1 & r_x^2 & r_y^2 \\ 1 & r_x^3 & r_y^3 \end{vmatrix} \end{aligned}$$

定义矩阵 \mathbf{A} 和 \mathbf{B} 分别为

$$\mathbf{A} = \begin{pmatrix} 1 & r_x^1 & r_y^1 \\ 1 & r_x^2 & r_y^2 \\ 1 & r_x^3 & r_y^3 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} s_x^1 & s_y^1 \\ s_x^2 & s_y^2 \\ s_x^3 & s_y^3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_x^1 & r_y^1 \\ r_x^2 & r_y^2 \\ r_x^3 & r_y^3 \end{pmatrix}$$

则

$$\mathbf{S}_e = \frac{\mathbf{B}\mathbf{B}^T}{4A_{tot}^e} = \frac{\mathbf{B}\mathbf{B}^T}{2|\mathbf{A}|} \quad (7.84)$$

实际中, 我们既可以利用式(7.83), 用解析计算的方法逐步的得到 \mathbf{S}_e , 也可以采用式(7.84), 用矩阵形式直接计算。

K_{IJ}^e 中的 I 和 J 代表面元 e 中的局部坐标系, 如 $I, J = 1, 2, 3$ 。计算得到面元系数矩阵后, 我们需要根据面元描述矩阵 $el2no$, 将局部坐标 I, J 映射到全局坐标 i, j 上。这一步骤也被称作组装。

组装的核心思想就是将计算得到的面元系数矩阵 K_{IJ}^e 加到全局矩阵 K_{ij} 对应的位置上去

$$K_{ij} = \sum_e^{n_E} K_{IJ}^e$$

下面首先看一个例子。

例题 7.15: 对于下表中的例子

表 7.1 节点连接矩阵示例

	1	2	3
el2no	1	4	5

在计算得到 \mathbf{K}_{IJ}^e 后，将其组装到 \mathbf{K}_{ij}^e 的过程可见下图。

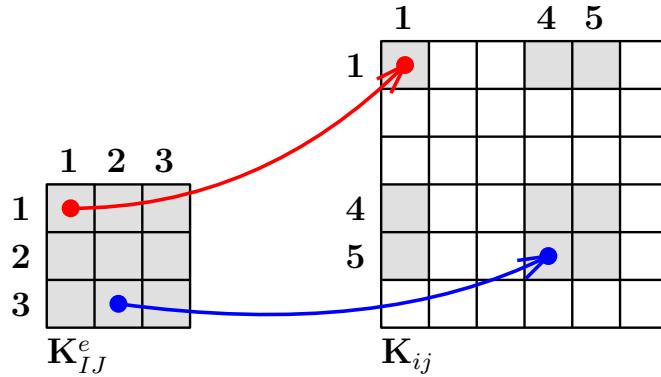


图 7.12 系数矩阵的组装示意图

7.3.3.3 扰动矩阵计算方法的组装视角

从系数矩阵组装的视角，可以方便的计算扰动矩阵。在计算

$$\frac{\partial \mathbf{K}}{\partial \sigma_i}$$

从组装示意图可以看出， \mathbf{K} 跟第 i 个面元电导率有关的只有 \mathbf{K}_{IJ}^i 。也就是说，在 $\partial \mathbf{K} / \partial \sigma_i$ 中，只有 \mathbf{K}_{IJ}^i 的贡献部分非零，其余面元的贡献全为 0，另一方面

$$K_{IJ}^i = \int_{\Omega^i} \sigma_i \cdot (\nabla \varphi_I \cdot \nabla \varphi_J) d\Omega$$

则

$$\frac{\partial \mathbf{K}_{IJ}^i}{\partial \sigma_i} = \int_{\Omega^i} (\nabla \varphi_I \cdot \nabla \varphi_J) d\Omega$$

针对 σ_i 扰动矩阵的计算可以简化为

1. 令 $i = i + 1$
2. 设 $\sigma_i = 1$ ，计算该面元系数矩阵 \mathbf{K}_{IJ}^i
3. 生成全零矩阵 \mathbf{K} ，利用 el2no 矩阵将 \mathbf{K}_{IJ}^i 组装到 \mathbf{K} 上
4. 所得 \mathbf{K} 即 $\partial \mathbf{K} / \partial \sigma_i$ ，跳转到 1

7.3.3.4 扰动矩阵计算方法的矩阵视角

组装的过程，可以方便的（数学意义上）用矩阵的形式表示

例题 7.16：左乘行搬移，右乘列搬移。

对于例题 el2no = (1, 4, 5)，我们构造矩阵 \mathbf{C}_e 为

$$\mathbf{C}_e = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

那么 (\mathbf{C}_e^T 中，每一个非零行中 1 的位置可选定右侧矩阵的对应行向量，并将其搬到新矩阵的当前行上)

$$\mathbf{C}_e^T \mathbf{K} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} = \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \\ 0 & 0 & 0 \end{pmatrix}$$

继续计算 $(\mathbf{C}_e^T \mathbf{K}) \mathbf{C}_e$ 可得 (\mathbf{C}_e 中，每一个非零列中 1 的位置可选定左侧矩阵的对应列向量，并将其搬到新矩阵的当前列上)

$$\begin{pmatrix} K_{11} & K_{12} & K_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} K_{11} & 0 & 0 & K_{12} & K_{13} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ K_{21} & 0 & 0 & K_{22} & K_{23} & 0 \\ K_{31} & 0 & 0 & K_{32} & K_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

利用 el2no 中的第 e 行，构造连接矩阵 \mathbf{C}_e ，则组装过程可写为

$$\mathbf{K} = \mathbf{K} + \mathbf{C}_e^T \mathbf{K}_{IJ}^e \mathbf{C}_e$$

更进一步，局部系数矩阵 \mathbf{K}_{IJ}^e 可以写为两部分

$$\mathbf{K}_{IJ}^e = \sigma_e \int \nabla \varphi_I \nabla \varphi_J = \mathbf{D}^e(\sigma) \mathbf{S}^e(\mathbf{r})$$

跟坐标相关的 $\mathbf{S}^e(\mathbf{r})$ 和跟电导率相关的 $\mathbf{D}^e(\sigma)$ 。当面元电导率为常数时, \mathbf{D}^e 为

$$\mathbf{D}^e = \sigma_e \mathbf{I}$$

其中 \mathbf{I} 为单位矩阵。则

$$\mathbf{K} = \mathbf{K} + \mathbf{C}_e^T \mathbf{S}^e(\mathbf{r}) \mathbf{D}^e(\sigma) \mathbf{C}_e = \mathbf{K} + \sigma_e \mathbf{C}_e^T \mathbf{S}^e(\mathbf{r}) \mathbf{C}_e$$

面元连接矩阵 \mathbf{C}_e 可辅助系数矩阵的组装过程。总系数矩阵的计算方法为

$$\mathbf{K} = \sum_e^{n_E} \sigma_e \mathbf{C}_e^T \mathbf{S}^e(\mathbf{r}) \mathbf{C}_e \quad (7.85)$$

例题 7.17: 总连接矩阵的意义以及系数矩阵计算的分块形式。

总连接矩阵 \mathbf{C} 由面元连接矩阵 \mathbf{C}_e 对叠而成, 它具备一些有用(简单)的性质。例如, 若连接矩阵 \mathbf{C} 的第 j 列 \mathbf{c}_j 中的 1 分别位于第 1 行、第 8 行和第 17 行, 其含义为

1. 模 3: 该节点 j 分别属于第 1 号面元、第 2 号面元和第 5 号面元,
2. 取余: 该节点在这三个面元中局部坐标分别为 1、2 和 2

系数 Stiffness 矩阵 \mathbf{K} 可以写为

$$\mathbf{K} = \mathbf{C}^T \mathbf{S}(\mathbf{r}) \mathbf{D}(\sigma) \mathbf{C} \quad (7.86)$$

其中 $\mathbf{S}(\mathbf{r}) \in \mathbb{R}^{3n_E \times 3n_E}$ 为局部面元系数矩阵构成的分块对角矩阵

$$\mathbf{S}(\mathbf{r}) = \begin{pmatrix} \mathbf{S}^1 & 0 & \cdots & 0 \\ 0 & \mathbf{S}^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{S}^{n_E} \end{pmatrix}$$

每一块 \mathbf{S}_i 对应于第 i 个面元的系数矩阵(除去 σ 部分的积分)。 $\sigma \in \mathbb{C}^{n_E}$ 为面元上阻抗值, $\mathbf{D}(\sigma) = \text{diag}(\sigma) \otimes \mathbf{I}_3$ 。

我们先来分析第 i 个面元电导率的绕动(很小的变动)对观测的影响, 余量 \mathbf{r} 可写为 $\mathbf{r} = \mathbf{v} - \mathbf{K}^{-1} \mathbf{b}$, 则 Jacobian 矩阵的第 i 列为

$$\mathbf{J}_i = \frac{\partial \mathbf{r}}{\partial \sigma_i} = -\frac{\partial}{\partial \sigma_i} (\mathbf{K}^{-1} \mathbf{b}) = \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \sigma_i} \mathbf{K}^{-1} \mathbf{b} = \mathbf{R} \frac{\partial \mathbf{K}}{\partial \sigma_i} \mathbf{v} \quad (7.87)$$

其中

$$\mathbf{R} \triangleq \mathbf{K}^{-1}$$

$$\mathbf{v} = \mathbf{K}^{-1}\mathbf{b}$$

\mathbf{R} 被称作节点（电极）阻抗矩阵， \mathbf{v} 为有限元法计算得到的节点电势。根据公式(7.85)，我们可得

$$\frac{\partial \mathbf{K}}{\partial \sigma_i} = \mathbf{C}_i^T \mathbf{S}_i \mathbf{C}_i \quad (7.88)$$

其中 $\mathbf{S}_i \in \mathbb{R}^{3 \times 3}$ 为第 i 个面元的系数矩阵， $\mathbf{C}_i \in \mathbb{R}^{3 \times n_P}$ 为第 i 个面元的连接矩阵。将式(7.88)代入式(7.87)中，即得 Jacobian 矩阵第 i 列的计算公式。

$$\mathbf{J}_i = \frac{\partial \mathbf{r}}{\partial \sigma_i} = \mathbf{R}(\mathbf{C}_i^T \mathbf{S}_i \mathbf{C}_i)\mathbf{v} \quad (7.89)$$

需要注意的是，计算 $\partial \mathbf{r} / \partial \sigma_i$ 只是得到 Jacobian 矩阵的一列，最后需要拼接为

$$\mathbf{J} = \left[\frac{\partial \mathbf{r}}{\partial \sigma_1}, \frac{\partial \mathbf{r}}{\partial \sigma_2}, \dots, \frac{\partial \mathbf{r}}{\partial \sigma_{n_E}} \right]$$

进行迭代计算。

7.3.3.5 Jacobian 矩阵计算流程图

Jacobian 矩阵计算流程图见图 7.13。图中展示的是一次激励下，Jacobian 矩

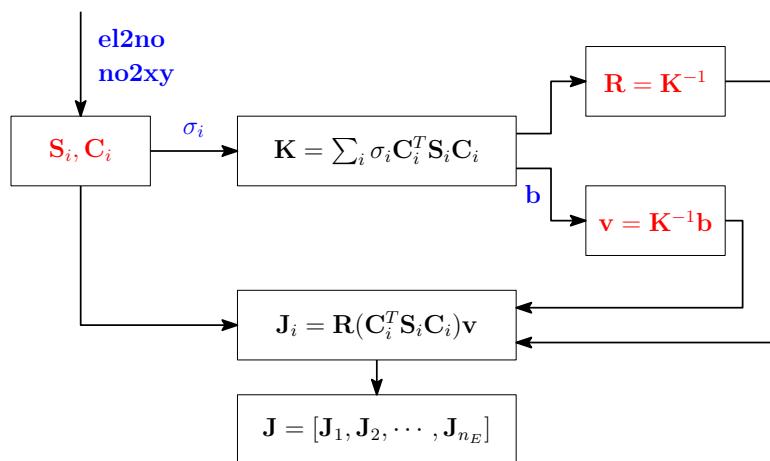


图 7.13 Jacobian 矩阵计算流程图

阵 \mathbf{J} 的计算步骤。蓝色的变量代表输入参数，红色的变量代表需存储的数据。

7.3.4 计算示例

逆问题，以及后面将要讲到的电阻抗断层成像 EIT，其计算量的瓶颈在 Jacobian 矩阵的计算。我们下面来看一个简单的例子，通过这个例子

- 理解系数矩阵（Stiffness Matrix） \mathbf{K} 的组装步骤，
- 理解电极阻抗矩阵 \mathbf{R} 、边界条件 \mathbf{b} 以及节点电势 \mathbf{v} 的计算步骤，
- 学习 Jacobian 矩阵 \mathbf{J} 的计算过程，

进而对后面将要学到的 EIT 正问题、逆问题有一个初步的认识。

例题 7.18：计算图7.14中的 Jacobian 矩阵。

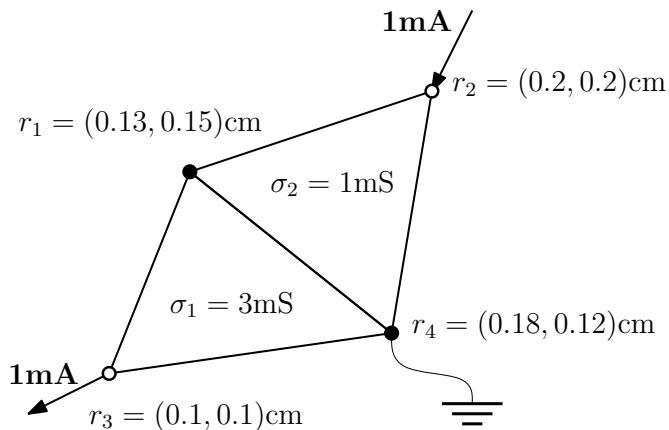


图 7.14 最简电磁场逆问题。这个例子包括 2 个电极（注入电流节点处），4 个节点。其中 4 号节点 r_4 设定为参考节点（电势设为 0，并不是物理接地，接地的话会有电流流出）。

在这个例子中，两个面元 σ_1, σ_2 的电导率分别为 $\sigma_1 = 3\text{mS}$, $\sigma_2 = 1\text{mS}$ 。给定节点位置坐标和节点编号后，可以方便的写出有限元结构化矩阵 $no2xy$ 和 $el2no$ 。

$$no2xy = \begin{pmatrix} 0.13 & 0.15 \\ 0.2 & 0.2 \\ 0.1 & 0.1 \\ 0.18 & 0.12 \end{pmatrix} \text{cm}, \quad el2no = \begin{pmatrix} 1 & 3 & 4 \\ 1 & 4 & 2 \end{pmatrix}$$

利用之前有限元课程中的 CmpElMtx，我们可以利用 **no2xy** 计算 S_1 和 S_2 ，

$$S_1 = \begin{pmatrix} 1 & -0.5 & -0.5 \\ -0.5 & 0.5 & 0 \\ -0.5 & 0 & 0.5 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 0.739 & -0.586 & -0.152 \\ -0.586 & 0.804 & -0.217 \\ -0.152 & -0.217 & 0.369 \end{pmatrix}$$

系数矩阵 \mathbf{K} 可由下式计算得到

$$\mathbf{K} = \sigma_1 \mathbf{C}_1^T \mathbf{S}_1 \mathbf{C}_1 + \sigma_2 \mathbf{C}_2^T \mathbf{S}_2 \mathbf{C}_2$$

连接矩阵 \mathbf{C} 可由 *el2no* 得出，

$$\mathbf{C}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{C}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

最后，设定边界条件为

$$\mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} \text{ mA}$$

需要注意的是，在未设置参考节点时，系数矩阵 \mathbf{K} 是奇异的。同学们可以打印 \mathbf{K} 的逆，或者计算 \mathbf{K} 的条件数 (rcond)。

为了解决奇异性，我们可设定节点 4 (实际中可任选节点) 为参考节点，等价于 Dirichlet 边界条件 $V_4 = 0V$ 。那么，我们 (1) 可以删除 \mathbf{K} 等式的第四行，同时移除 \mathbf{K} 的第 4 列，或者 (2) 令 \mathbf{K} 的第 4 行，第 4 列为 0，同时令 $K_{44} = 1$ 。这两种方法都等价为设该节点为参考节点，消除 \mathbf{K} 求逆的奇异性。本例题中，我们采用第二个策略。

通过求解 $\mathbf{K}^{-1} \mathbf{b}$ 我们可以得到节点上的电压

$$\mathbf{v} = (-0.27 \quad 2.59 \quad -0.93 \quad 0)^T \text{ V}$$

以及电极阻抗 (此时设测量电极位于激励电极上，即在 2 号和 3 号电极上测量

电势), 即 $\mathbf{R} = [\mathbf{K}^{-1}]_{row2, row3}$

$$\mathbf{R} = \begin{pmatrix} 189 & 2783 & 189 & 0 \\ 459 & 189 & 1126 & 0 \end{pmatrix} \text{ Ohm}$$

最后, 我们来计算电导率变化的扰动矩阵, 即 Jacobian 矩阵。首先计算第一个面元电导率 σ_1 的扰动对于观测的影响 $\partial \mathbf{r} / \partial \sigma_1$ 。

$$\frac{\partial \mathbf{r}}{\partial \sigma_1} = \mathbf{R}(\mathbf{C}_i^T \mathbf{S}_i \mathbf{C}_i) \mathbf{v}$$

得到

$$\frac{\partial \mathbf{r}}{\partial \sigma_1} = \mathbf{R} \mathbf{C}_1^T \mathbf{S}_1 \mathbf{C}_1 \mathbf{V} = \begin{pmatrix} -25 \\ -284 \end{pmatrix}$$

对于 $\partial \mathbf{r} / \partial \sigma_2$ 可类似得出。

7.3.5 课后习题

7.3.5.1 上机实验

上机 7.3: 编写高斯牛顿迭代算法: gn(), 算法更新公式为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}_r^T \mathbf{J}_r + \alpha \mathbf{I})^{-1} \mathbf{J}_r^T \mathbf{r}$$

请类比本章例题所讲内容, 设计函数的输入与输出接口。

上机 7.4: 在所给程序 (Simple EIT) 的基础上, 编写代码, 分别计算

1. 计算 Jacobian 矩阵。首先继续计算

$$\frac{\partial \mathbf{r}}{\partial \sigma_2}$$

随后将其拼接为一个 Jacobian 矩阵 \mathbf{J}

2. 设定初始电导率为 $\sigma_0 = \{1, 1\}$, 计算电势分布 (电磁场正问题)
3. 假定节点 2、3 的观测电势分别为 2.59V、-0.93V, 利用该观测向量, 编写 gn 算法, 迭代计算

$$\sigma_1 = \sigma_0 - (\mathbf{J}_r^T \mathbf{J}_r + \alpha \mathbf{I})^{-1} \mathbf{J}_r^T \mathbf{r}$$

试着仅进行 4 步迭代, 计算并打印出 σ_k

4. 你能从数值计算结果中得到怎样的结论?

7.4 电阻抗断层成像

7.4.1 引子

7.4.1.1 最简电磁场逆问题的求解

我们再次回到上节课的例子。这里假定真实的面元阻抗是未知的，在2号（3号）节点上注入（流出）恒定的电流，并且测量2、3节点的电势值。它们分别为 $V_2 = 2.59V$, $V_3 = -0.93V$ 。跟例题对比可知，实际观测值有2的精度。随后同学们会学习到，由于电阻抗断层成像逆问题的病态性，对测量数据的精度提出了较高的要求。

现在，我们要用上一节课所讲内容——高斯牛顿法——求解最小化问题，估计面元阻抗。使用牛顿法或者高斯牛顿这一类迭代优化算法，需要指定待估计参数的初值，设

$$\sigma_0 = [1, 1]$$

除此以外，gn 算法还需计算 Jacobian 矩阵。回忆一下，Jacobian 矩阵的第 i 列可写为

$$\mathbf{J}_i = \frac{\partial \mathbf{r}}{\partial \sigma_i} = \mathbf{R}(\mathbf{C}_i^T \mathbf{S}_i \mathbf{C}_i) \mathbf{u}$$

式中 \mathbf{R} 和 \mathbf{u} 都是跟当前的 σ_k 相关的。也就是说，在每一次迭代计算得到新的 σ_k 后，都需要重新计算系数矩阵和节点电势。这也就意味着进行多少次迭代，就要算多少遍有限元正问题。这些计算量也就是 EIT 问题的瓶颈。

程序的编写流程可见图7.15。实际的程序相比于先前的代码只需很少的改动。我们设定迭代次数为5，正则化参数 $\alpha = 0.1$ （根据具体问题的不同，正则化参数的选取也不同），每次迭代中，阻抗估计值见表7.2。从表中可以看出，迭代4次以后，估计得到的值与真值之间的差距很小。同学们可以试着更改观测电势的值（误差），或者选择一组新的初始阻抗，又或者设定一个新的正则化参数 α （例如 $\alpha = 0.0001$ ），看结果有何不同。

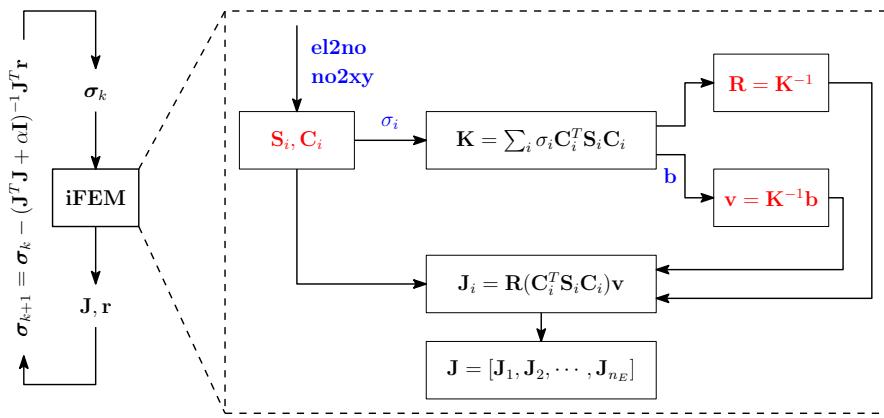


图 7.15 最简电磁场逆问题程序编写流程

表 7.2 高斯牛顿法迭代求解最简电磁场逆问题的数值结果

	σ_1	σ_2
iter = 0	2.0	2.0
iter = 1	2.6110	0.3586
iter = 2	2.8818	0.5880
iter = 3	2.9690	0.8298
iter = 4	2.9964	0.9711
iter = 5	3.0087	1.0005
真实值	3.0	1.0

7.4.1.2 手工上色的火星图像

在本课程中，所有数值计算的结果都要可视化。事实上，工程领域中最早的可视化都是由人工完成的。一个最经典的例子是水手 4 号回传的火星图像。

图 7.16 是水手 4 号 (Mariner 4) 掠过火星时传递过来的人类第一张地外行星的近距离特写图。当时接收的图像数据以整数的形式打印在纸带上，喷气推进实验室 (Jet Propulsion Laboratory) 的科学家们 (Richard Grumm 及他的团队) 通过手工上色的方式绘制出了这幅图片。更为巧合的是，科学家们采用了褐色色调的彩笔，采用这种原始的可视化方法，第一次绘制出了火星的色彩。

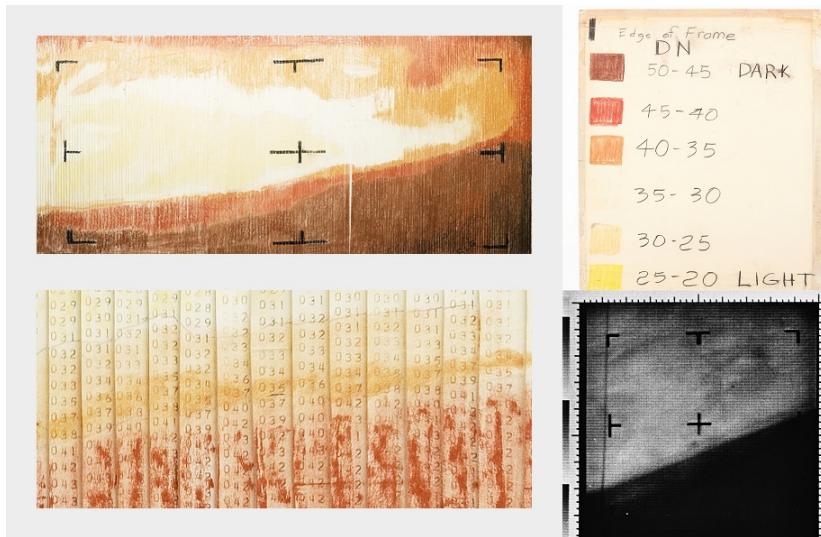


图 7.16 水手 4 号太空探测器于 1965 年 7 月 15 日“路过”火星时拍摄的照片。图片由计算机转成整数打印成纸带，喷气推进实验室的一群科学家迫不及待（当时的电脑速度还很慢）的用手工上色的方式绘制出了这幅图片。

7.4.2 计算流程

7.4.2.1 基本定义

在二维 (2D) 情形中 (文献 [21] 考虑了二维及三维的情形, 本课程我们只针对 2D 问题展开), 电磁场数值计算问题可见图 7.17。

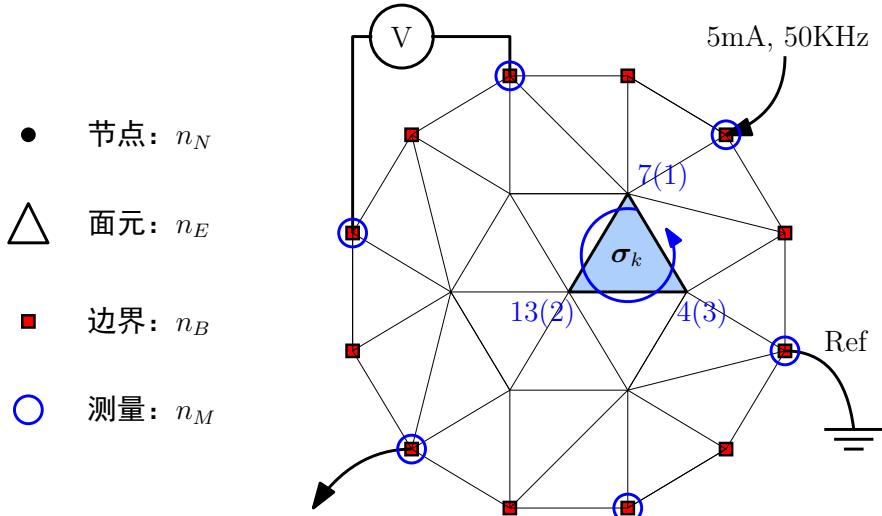


图 7.17 电磁场数值计算问题。其中第 k 个面元用灰色高亮, 其介电常数用 σ_k 表示。其节点坐标绘制于三个顶点处, 括号内的数字为该面元局部节点编号

设节点数为 n_N (Node), 面元数为 n_E (Element), 面元介电常数为定义在区域 Ω 中的一个标量 σ 。也就是说, 假定在每一个面元中, 介电常数是恒定的。在区域 Ω 的边界 $\partial\Omega$ 上, 共有 n_B (Boundary) 个有限元节点, 其中置有 n_M (Measurement) 个测量电极¹。

受制于器件水平制约, 针对每次激励 (边界条件 \mathbf{b}), 我们只能最多获取 n_M 个测量²。而一个中等密度剖分的圆形区域, 其面元数目一般可达 $200 \sim 400$ 。那么对于逆问题, 有

$$n_M \ll n_E$$

因而, 单单一次测量的成像质量和精度都是有限的。解决办法就是多次重复激励。因此, 在最简 EIT 流程图上, 我们只需加入多个激励即可 $\{\mathbf{b}_i\} = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M\}$ 。最终 EIT 的计算流程见图 7.18。程序的主模块为 iFEM, 其内

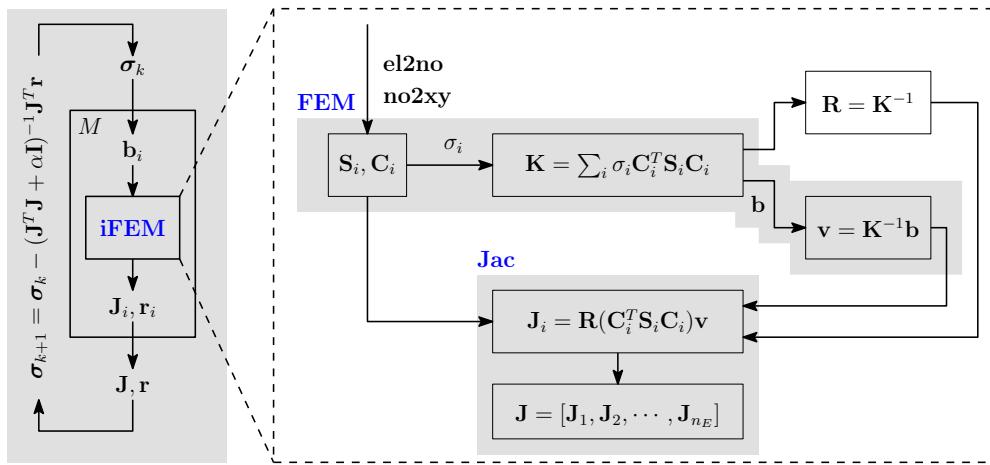


图 7.18 EIT 程序编写流程

部主要包括两部分, 其一是有限元正问题计算程序 **FEM**; 另外一个是 (每次激励下) Jacobian 矩阵的计算程序 **Jac**。

这里需要强调的是, 同学们一定要对有限元中各个元素的数目有清晰的认识, 尤其是 n_N 、 n_E 、 n_M , 以及激励的次数 M 。在后面的数值计算程序中, 将反复讲到各个变量 (矩阵、向量) 的大小。

¹每个电极可以用点电极 (single node) 模型或者全电极 (CEM, Complete Electrode Model) 模型来描述, 在全电极模型下, 电极可与区域之间的多个节点相连接。

²只有当假设电极为复合电极时, 才能够获取 n_M 个测量数据。也就是说对于任何一个电极, 即能够当作激励电极, 又能被用于测量电压。

7.4.3 正问题

7.4.3.1 基本理论：Maxwell 方程组和边界条件

电阻抗断层成像研究的是无电荷存在区域的电磁场数值计算问题，且在激励过程中忽略感应磁场和感应电流的影响。那么 EIT 所研究的问题就是典型的静电场数值计算。在静电场（electro-static）中，有

$$-\nabla \cdot (\sigma \nabla u) = 0 \quad \text{in } \Omega \in \mathbb{R}^2 \quad (7.90)$$

$$\sigma \frac{\partial u}{\partial \hat{n}} = g \quad \text{on } \partial\Omega \quad (7.91)$$

$$\int_{\partial\Omega} u \, ds = 0 \quad (7.92)$$

其中 $g \in H_0^{-1/2}(\partial\Omega)$ 是激励电流（current flux）， $u \in H^1(\Omega)$ 是电势的位函数， $\sigma \in L^\infty(\Omega, \mathbb{C}_c)$ 是电导率。

例题 7.19：（选学）在有限元分析、电阻抗成像文献中，常会见到以下几个数学符号 $H^{-1/2}(\partial\Omega)$ ， $H^1(\Omega)$ 以及 $L^\infty(\Omega, \mathbb{C}_c)$ 。文献中，这些空间分别被定义为

$$\begin{aligned} \mathbb{C}_c &\triangleq \{z \in \mathbb{C} : \operatorname{Re}(z) > c > 0\} \\ H^{-1/2}(\partial\Omega) &\triangleq \left\{ g \in H^{-1/2}(\partial\Omega) : \int_{\partial\Omega} g \, ds = 0 \right\} \\ H^1(\Omega) &\triangleq \left\{ \varphi \in H^1(\Omega) : \int_{\partial\Omega} \varphi \, ds = 0 \right\} \end{aligned}$$

简言之，Sobolev 空间 $H^k(\Omega)$ 由 k 阶平方可积（square integrable L^2 ）的函数构成。在高维空间中，Sobolev 空间比连续函数空间 $C^0(\Omega)$ 的约束更少。在阅读文献中，为了便于理解，我们可以将所有的 Sobolev 空间替换为由连续函数构成的 C^0 空间。

下面考虑该问题的弱形式，随意选定测试函数 ω ，利用加权余量法可得

$$\int_{\Omega} \sigma \nabla u \cdot \nabla \omega \, dx = \int_{\partial\Omega} g \omega \, ds \quad (7.93)$$

那么，给定面元电导率 σ 的分布与边界激励 g ，我们可以唯一的确定节点上的电势 u 。

7.4.3.2 有限元方法

有限元是求解电磁场方程的一种数值（近似）方法。除了有限元方法，我们还可以采用有限差分法或者边界元法求解电磁场数值计算问题。实际应用中，由于有限元方法的边界适应性强，一般的 EIT 问题多采用 FEM 进行计算。

首先，在加权余量法得到的弱形式基础上

$$\int_{\Omega} \sigma \nabla \omega_i \cdot \nabla u d\Omega = \int_{\partial\Omega} \omega_i g dl$$

我们令 $\varphi_i(\mathbf{r})$ 代表第 i 个节点的线性基函数。基函数 $\varphi_i(\mathbf{r})$ 具备局部化的特性：只有在 \mathbf{r}_i 处为 $\varphi_i(\mathbf{r}_i) = 1$ ，而在其它节点 $j \neq i$ 上为 $\varphi_i(\mathbf{r}_j) = 0$ 。有限元和其节点上的基函数统称为节点面元（nodal elements）。

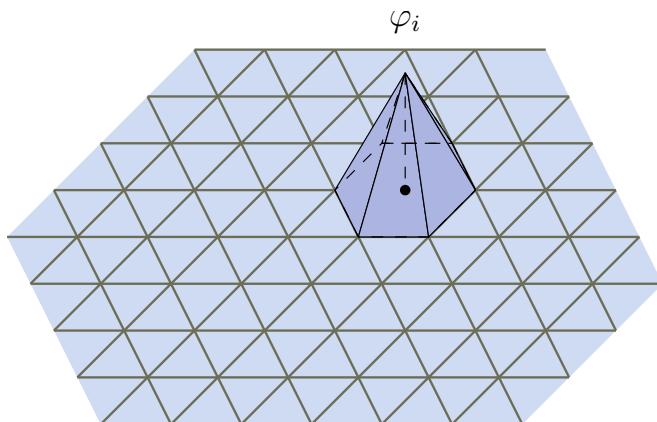


图 7.19 有限元中的节点基函数。基函数具备局部化的特性，仅仅在节点 i 上为 1，在其它节点上都为 0。

接下来，我们将待求函数（节点电势） \mathbf{u} 在基函数上展开

$$\mathbf{u} = \sum_i^n u_i \varphi_i(\mathbf{r})$$

将该展开式代入弱形式，并且利用伽辽金有限元（即令测试函数等同于基函数 $\omega_i = \varphi_i$ ），得到有限元方程组 $\mathbf{K}\mathbf{u} = \mathbf{b}$ ，其中

$$K_{ij} = \int_{\Omega} \sigma (\nabla \varphi_i \cdot \nabla \varphi_j) d\Omega \quad (7.94)$$

$$b_i = \int_{\partial\Omega} \varphi_i g dl \quad (7.95)$$

其中系数矩阵 \mathbf{K} 是一个与电导率 σ 及面元结构相关的系数矩阵，可写作

$\mathbf{K}(\sigma, \Omega)$ 。

在系数矩阵的计算中，无需求解 Dirichlet 边界条件 L_1 上的节点（此类边界节点上电势已知），有限元方程可改写为

$$\left(\begin{array}{c|c} \mathbf{K}_e & \mathbf{K}_n \end{array} \right) \begin{pmatrix} \mathbf{f}_e \\ \mathbf{f}_n \end{pmatrix} = \mathbf{K}_e \mathbf{f}_e + \mathbf{K}_n \mathbf{f}_n = \mathbf{b}$$

其中 \mathbf{f}_e 为 Dirichlet 边界条件下已知函数值的节点， \mathbf{f}_n 为未知节点， \mathbf{K}_n 是方阵。最终我们需要从 $\mathbf{K}_n \mathbf{f}_n = \mathbf{b} - \mathbf{K}_e \mathbf{f}_e$ 中求解 \mathbf{f}_n 。假定 EIT 应用中不存在 Dirichlet 边界条件，那么有限元方程可简写为

$$\mathbf{K}(\sigma, \Omega) \mathbf{f} = \mathbf{b}$$

7.4.3.3 网格剖分

网格剖分是一个专业性较强的领域，按照难易程度可分为

1. (易) 标准几何形状的网格剖分算法。主要要求同学们掌握圆形区域的分层网格生成方法；
2. (中) 基本几何形状的网格剖分算法。对于较为复杂的区域，可采用现有的网格生成工具，例如 `distmesh`、`meshpy` 等，进行网格生成。这两个工具可支持不同的编程语言，例如 C/C++、Python、MATLAB 等；
3. (难) 真实区域的网格剖分算法。对于实际采集得到的 CT/MRI 图像，可以用专业的工具进行网格剖分，例如 `COMSOL` 等。

实际上，网格剖分的难易程度是相对的。对于工具使用熟练的同学，利用 `COMSOL` 进行网格剖分可能会更加便捷。

但无论使用何种工具，为了有限元程序编写的方便，我们需要一种高效、直观的数据结构，用来存储和处理非结构化网格。例如在二维场景中，以图7.20为例，图中的网格包含 6 个节点和 4 个元。我们可分别建立两个矩阵用来描述网格信息：

- $no2xy$ ，是一个 $2 \times n_N$ 的矩阵，用来存储每一个节点的 (x, y) 坐标
- $el2no$ ，是一个 $(3 + 1) \times n_E$ 的矩阵，用来存储每一个面元 e 对应的三个节点编号和介电常数，节点编号以逆时针的顺序存放。

对于图7.20中所示的例子， $no2xy$ 可见表7.3， $el2no$ 可见表7.4。

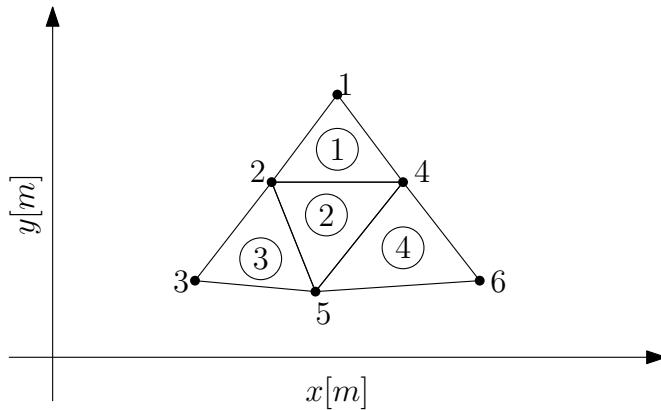


图 7.20 二维非结构化网格。节点全局编号标记于节点旁，面元编号置于三角形内。

表 7.3 给定节点的全局编号，no2xy 存储节点的 (x, y) 坐标

Node	1	2	3	4	5	6
x	0.0	-0.5	-0.8	0.6	0.0	1.0
y	1.0	0.5	0.0	0.4	-0.2	-0.1

表 7.4 给定面元的全局编号，el2no 存储面元所对应的节点序号

Element	1	2	3	4
Node 1	1	4	3	5
Node 2	2	2	5	6
Node 3	4	5	2	4
σ	1	$2 + 2j$	1	1

7.4.3.4 系数矩阵的计算

有限元系数矩阵可以写成每个面元上系数矩阵 \mathbf{K}^e 的组合。而 \mathbf{K}^e (3×3 矩阵) 中 K_{IJ}^e 的计算方法为

$$K_{IJ}^e = \sigma_e \frac{\mathbf{s}_I \cdot \mathbf{s}_J}{4A_{tot}^e}$$

其中 \mathbf{s}_I 、 \mathbf{s}_J 为三角形面元的两条边向量， A_{tot}^e 为面元 e 的面积。这样，我们仅仅需要三角形面元三个顶点的坐标，即可计算得到 \mathbf{K}^e 。

在得到 \mathbf{K}^e 后，有两种方法将其组装到 \mathbf{K} 上。设 $el2no = \{1, 4, 5\}$ 为当前面

元的连接方式，那么

1. 系数组装方法。组装方法本质上是个 `for` 循环

```
for I, J ∈ {1, 2, 3}
```

$$K_{ij} += K_{IJ}^e, \quad i = \text{el2no}(I), j = \text{el2no}(J)$$

2. 连接矩阵方法。设 C_e 为面元 e 的连接矩阵，则

$$\mathbf{K} = \sum_e^{n_E} \sigma_e C_e^T S^e(\mathbf{r}) C_e$$

这里需要注意的是，在求 \mathbf{K}^{-1} 时，为了避免矩阵的奇异性，需选定电势参考节点。即设某一节点的电势为 0，随后所有电势按照该点进行归一化。归一化的方法为将 \mathbf{K} 中该节点（假设节点序号为 i ）所在行和列置零，同时令 $K_{ii} = 1$ 。

7.4.3.5 观测数据的获取

在真实的电阻抗 (EIT: Electrical Impedance Tomography) 成像系统中，观测数据的获取流程为：任意选择两电极作为激励电极。¹ 在每一种激励模式下，我们通过测量两电极 p, q 之间的电压差，构成第 m 个观测量，

$$v_m = \phi_p - \phi_q$$

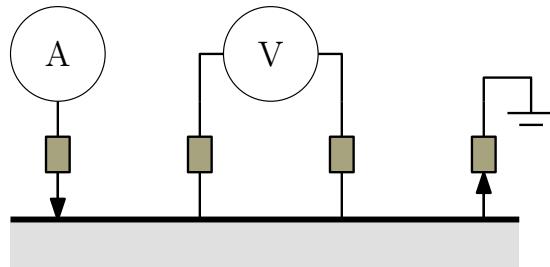


图 7.21 四电极法测量模式

在计算 EIT 正问题时，我们用 A, B 表示电流激励电极 (current/active electrodes)，用 M, N 表示电压测量电极 (voltage/passive electrodes)。通过这种四电极法测量边界上 M, N 节点对之间的电势差，可以较好的抑制接触阻抗对于

¹ 这里还有参考电极的概念，在边界 $\partial\Omega$ 上任选一节点（可以不是电极所在的节点）设为参考电极，令参考电极上电势为 0（即 0 电位），等效为接地，但是没有电流流出。电极 p 与参考电极之间的电势差为 ϕ_p 。

测量精度的影响，适用于边界阻抗率（如皮肤、颅骨）等变化较大的情形。在正问题仿真中，我们设定所有电极为理想的点电极模型（即假定电极置于边界上有限元的节点上）。

对于 n_M 个测量电极（简单电极），在

1. 邻近激励模式。共有 $n_M(n_M - 3)$ 个有效电压观测数据
2. 对向激励模式。共有 $n_M(n_M - 4)$ 个有效电压观测数据

当前研究中，不同激励模式对于不同应用效果也不同。例如在颅脑 EIT 监控中，多采用对向激励，以便更好的反映脑内部的变化信息；在胸腹部 EIT 监控中，则多采用邻近激励模式，检测随着呼吸肺部容积的变化。并且，激励模式不仅仅局限于以上两种，事实上可以采用更为灵活多变的激励模式。同学们可以试着探讨下不同激励模式对于成像效果的影响。

7.4.3.6 激励模式与观测向量

前面讲到，为了改善逆问题的病态特性，我们可以通过切换 M 次激励，测量不同激励下的电极电势。我们假定 EIT 工作于电流激励模式下，设定正激励在 i ，负激励在 j ，那么对于第 k 次激励，边界条件 $\mathbf{b}_k \in \mathbb{R}^{n_N}$ 为

$$[\mathbf{b}_k]_i = 1, \quad [\mathbf{b}_k]_j = -1$$

对于每次激励，正问题计算得到的节点电势为

$$\mathbf{u}_k = \mathbf{K}^{-1} \mathbf{b}_k$$

设算子 T 的功能为提取出测量电极（ n_M 个）所在的行。例如，假设电极位于节点 [1, 4, 11, 19] 处，则 T 就可提取 \mathbf{u} 中的对应行的元素构成新的向量。而对于全电极模型，算子 T 则提取所有与电极相连接的节点所在的行。

电极节点上的测量数据为

$$\mathbf{v}'_k = T[\mathbf{u}_k]$$

同理，节点阻抗矩阵定义为 $\mathbf{R} = \mathbf{K}^{-1}$ 。为了减小计算量，我们可以仅仅提取 \mathbf{R} 中电极节点所在的行即可。最终得到电极阻抗矩阵 $\mathbf{R}' \in \mathbb{R}^{n_M \times n_P}$ 为

$$\mathbf{R} = \mathbf{K}^{-1}$$

$$\mathbf{R}' = T[\mathbf{R}]$$

电极阻抗矩阵的含义与电阻的定义类似，即

$$\mathbf{v} = \mathbf{K}^{-1} \mathbf{b} \sim V = RI$$

所有激励（边界条件）可以存储在一个向量中

$$\mathbf{b} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M]$$

若有 M 次激励，需要将所有的观测电势拼接成一个大的向量

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}'_1 \\ \mathbf{v}'_2 \\ \vdots \\ \mathbf{v}'_M \end{bmatrix}$$

在仿真中，可以不考虑四电极测量模型，即无需计算两两电极之间的电势差。需要注意的是，若考虑四电极模型，则在计算电极阻抗矩阵时，也需要将对应电极所在的行向量求差，构成符合模型的差分电极阻抗矩阵。

7.4.3.7 编程讲解：2D EIT 正问题仿真

在电阻抗成像中，为了便于同学们编程，我们可以假定为电流激励、电压测量模式，通过 FEM 仿真，计算不同激励下，节点上的电势分布。

7.4.4 逆问题

上节课我们讲解了电磁场逆问题中 Jacobian 矩阵的计算方法，这里简要回顾一下重要知识点。电磁场逆问题通过求解最小化

$$\boldsymbol{\sigma}^* = \arg \min \|\mathbf{v} - f(\boldsymbol{\sigma}, \Omega)\|_2^2$$

获得 $\boldsymbol{\sigma}$ 的估计。其中 f 为将 $(\boldsymbol{\sigma}, \Omega)$ 映射到测量电压 \mathbf{v} 的非线性投影算子，在有限元数值模型中，

$$\mathbf{v} = f(\boldsymbol{\sigma}, \Omega) = \mathbf{K}(\boldsymbol{\sigma})^{-1} \mathbf{b}$$

因而最小化问题可写为

$$\boldsymbol{\sigma}^* = \arg \min \|\mathbf{v} - \mathbf{K}(\boldsymbol{\sigma})^{-1} \mathbf{b}\|_2^2$$

而逆问题的求解核心就是计算 Jacobian 矩阵。

7.4.4.1 扰动 (Jacobian) 矩阵

第 k 次激励下, Jacobian 矩阵的计算方法:

$$[\mathbf{J}_k]_i = \frac{\partial \mathbf{r}_k}{\partial \sigma_i} = \mathbf{R}(\mathbf{C}_i^T \mathbf{S}_i \mathbf{C}_i) \mathbf{v} \quad (7.96)$$

$$\mathbf{J}_k = \left[\frac{\partial \mathbf{r}_k}{\partial \sigma_1}, \frac{\partial \mathbf{r}_k}{\partial \sigma_2}, \dots, \frac{\partial \mathbf{r}_k}{\partial \sigma_{n_E}} \right] \quad (7.97)$$

其中 $\mathbf{J}_k \in \mathbb{R}^{n_N \times n_E}$ 。我们若仅提取电极阻抗矩阵, 即

$$\mathbf{R}' = T[\mathbf{R}]$$

进行计算, 则 \mathbf{J} 的大小为 $n_M \times n_E$ 。若采用邻近激励、四电极测量方法, 则 \mathbf{J} 的大小为 $(n_M - 3) \times n_E$ 。

M 次激励下的 Jacobian 矩阵 $\mathbf{J} \in \mathbb{R}^{M n_M \times n_N}$ 由每一种激励模式的矩阵 \mathbf{J}_i 纵向拼接而成

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_{n_E} \end{pmatrix} \quad (7.98)$$

7.4.4.2 逆问题求解算法

我们利用 Gaussian–Newton 算法求解 EIT 的逆问题。设定初始面元电导率 $\sigma_0 \in \mathbb{C}^{n_E \times 1}$, 通过迭代求解 $\sigma_k \rightarrow \sigma_{k+1}$,

$$\sigma_{k+1} = \sigma_k - \left[\mathbf{J}^T \mathbf{J} + \alpha \mathbf{L}^T \mathbf{L} \right]^{-1} \mathbf{J}^T \mathbf{r} \quad (7.99)$$

Jacobian 矩阵 $\mathbf{J} = f'(\sigma_k)$ 也可以看作映射 f 在 $\sigma = \sigma_k$ 时的 Frechet 导数。 \mathbf{v} 是不同激励模式下的合成测量矩阵 (向量), $\mathbf{L}^T \mathbf{L}$ 可以选取为 $\mathbf{J}^T \mathbf{J}$ 的对角元素, 也可以令 \mathbf{L} 为单位矩阵 $\mathbf{L} = \mathbf{I}_{n_E}$ 。针对正则化参数 α , 实际编程中, 我们可以选择一个逐步减小的值 $\alpha = \{\alpha_1, \dots, \alpha_k\}$, 渐进的重构 σ_k 。

7.4.5 成像算法

7.4.5.1 EIT 静态成像算法

静态成像 (static imaging) 算法, 即在 t 时刻, 通过优化算法求解面元上电导率 σ 的绝对分布。

静态 EIT 成像可利用式(7.99)进行计算

$$\begin{aligned}\mathbf{r} &= \mathbf{v} - f(\boldsymbol{\sigma}) \\ \boldsymbol{\sigma}_{k+1} &= \boldsymbol{\sigma}_k - [\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{r}\end{aligned}$$

静态 EIT 成像的步骤分为：首先，设定面元电导率 $\boldsymbol{\sigma}$ 的初始值（一般选取均匀背景阻抗）；随后，迭代求解 $\boldsymbol{\sigma}_{k+1}$ 。在每一步迭代时，都需要根据当前 $\boldsymbol{\sigma}_k$ 重新计算扰动矩阵 \mathbf{J} 和残差 \mathbf{r} 。若包含 M 次激励，则在每次迭代中，需重新计算 M 个正问题 $f(\boldsymbol{\sigma}_k)$ 。

静态成像的优点是能得到区域 Ω 上绝对的阻抗分布。而缺点是

1. 成像结果敏感于边界电压的测量精度；
2. 成像质量受边界形状影响，实际应用中边界的真是形状难以精确获取；
3. 需要较多的迭代次数，每一步都要计算多次有限元正问题，计算量大，不利于实时成像。

7.4.5.2 EIT 动态成像算法

EIT 动态成像，也被称作时域差分（temporal differential）成像。即在 $t, t+1$ 时刻，通过两帧的观测数据 $\mathbf{v}^{(t)}$ 和 $\mathbf{v}^{(t+1)}$ 的差值，求电导率 $\boldsymbol{\sigma}$ 在两时刻的变化量 $\delta_t \boldsymbol{\sigma}$ 。

在静态成像算法的基础上，我们很容易得到 EIT 动态成像算法。假定在两时刻 $t, t+1$ ，区域 Ω 上的电导率分布分别为 $\boldsymbol{\sigma}^{(t)}$ 和 $\boldsymbol{\sigma}^{(t+1)}$ ，两时刻采用相同的激励模式，电压测量值分别为 $\mathbf{v}^{(t)}$ 和 $\mathbf{v}^{(t+1)}$ 。我们设定相同的迭代初始值 $\boldsymbol{\sigma}_0$ ，根据静态成像公式(7.99)，有

$$\begin{aligned}\boldsymbol{\sigma}_1^{(t)} &= \boldsymbol{\sigma}_0 - [\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I}]^{-1} \mathbf{J}^T [\mathbf{v}^{(t)} - f(\boldsymbol{\sigma}_0)] \\ \boldsymbol{\sigma}_1^{(t+1)} &= \boldsymbol{\sigma}_0 - [\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I}]^{-1} \mathbf{J}^T [\mathbf{v}^{(t+1)} - f(\boldsymbol{\sigma}_0)]\end{aligned}$$

两式相减，可得

$$\boldsymbol{\sigma}^{(t+1)} - \boldsymbol{\sigma}^{(t)} = -[\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I}]^{-1} \mathbf{J}^T [\mathbf{v}^{(t+1)} - \mathbf{v}^{(t)}] \quad (7.100)$$

其中 \mathbf{J} 为初始状态——均匀背景阻抗 $\boldsymbol{\sigma}_0$ 下计算得到的扰动矩阵。

需要注意的是，尽管 \mathbf{J} 可由程序离线计算得到。但是，由于扰动矩阵 \mathbf{J} 需要精确的匹配实际应用中的电极位置和边界形状，这一条件往往难以满足。为此，我们引入归一化（定标）步骤：通过采集均匀背景下的实测电压值（参考

帧) $\mathbf{v}_0 = f(\sigma_0)$, 随后令 $\mathbf{J}' = \text{diag}(\mathbf{v}_0^{-1})\mathbf{J}$ 对扰动矩阵进行归一化处理。

动态 EIT 成像的过程为: 首先计算均匀背景下的扰动矩阵 \mathbf{J} , 根据式

$$\mathbf{H} = [\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I}]^{-1} \mathbf{J}^T \quad (7.101)$$

计算矩阵 \mathbf{H} , 并储存在内存中(动态成像过程只需计算一次)。需要注意的是, 在式(7.101)中, \mathbf{H} 等价为扰动矩阵 \mathbf{J} 的正则化伪逆 \mathbf{J}^\dagger 。随后根据 t 和 $t+1$ 时刻的电压测量数据的差, 计算电导率的变化量

$$\delta_t \sigma = \sigma^{(t+1)} - \sigma^{(t)} = -\mathbf{H} (\mathbf{v}^{(t+1)} - \mathbf{v}^{(t)}) \quad (7.102)$$

7.4.5.3 EIT 准静态(多频)成像算法

EIT 准静态(Quasi-static)成像算法, 也被称作频域差分成像, 即在 t 时刻, 利用 m 和 n 两个频点(不同激励频率)之间的观测数据 $\mathbf{v}^{(m)}$ 和 $\mathbf{v}^{(n)}$, 求两频点下区域电导率的变化量 $\delta_f \sigma$ 。

与动态成像类似, 我们可以方便的写出频差准静态成像算法的计算公式

$$\delta_f \sigma = \sigma^{(m)} - \sigma^{(n)} = -\mathbf{H} (\mathbf{v}^{(m)} - \mathbf{v}^{(n)}) \quad (7.103)$$

7.4.5.4 EIT 成像方式对比

三种成像方式的对比可见图7.22。

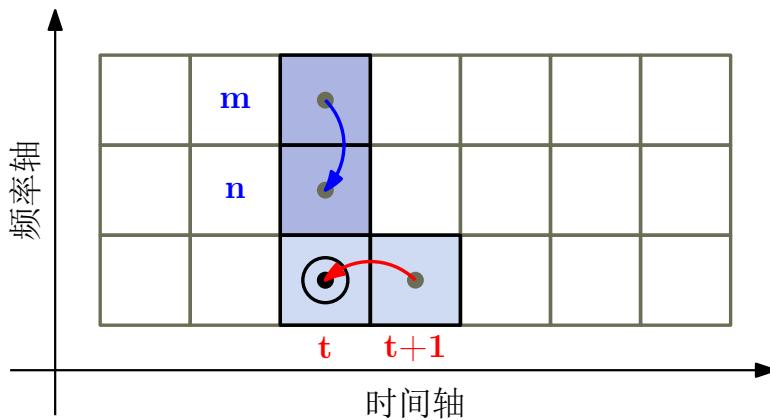


图 7.22 EIT 三种成像方式对比

对于电阻抗成像技术, 人们希望她能够具备和 CT/MRI 类似的功能, 能够对区域内部组织特性的绝对分布成像。也就是期望能够应用静态成像技术。但是实际应用中由于物理设备的种种限制(例如人体界面的边界形状、从 3D 到

2D 的简化、电极接触阻抗), 以及实际应用的限制 (例如在颅脑应用中, 颅骨及脑脊液的阻抗严重制约了内部区域的成像精度) 等, 静态成像算法研究虽多, 但实际应用较少。当前动态成像算法是研究和应用的热点。动态成像已经在脑卒中患者的床旁实时监控、人体 (肺) 呼吸动态监测中得到了具体应用。随着器件的更新和设备的研发, 频差准静态成像设备也逐步发展起来。频差成像算法能够提供更多的信息量 (频域), 将极大的推进 EIT 设备在医疗领域的应用。可以预见, 在后续的研究中, 时频信息的利用和信号处理算法将在 EIT 设备中发挥举足轻重的地位。

7.4.6 实际应用

7.4.7 上机指南

我们再来分析一个更为实际的例子¹。通过本样例程序

- 理解实际 EIT 中正问题的计算步骤 (以有限元法为例);
- 计算 EIT 中的 Jacobian (扰动) 矩阵 J ;
- 利用 Gauss–Newton 算法迭代求解 EIT 逆问题。

¹参考 Andreas Helfrich–Schkarbanenko [22] 的 EIT2D 程序结构。或者参考教研室提供的 pyEIT 程序代码 (Python)。

第 8 章 随机类全局优化算法

刘本源 *bbyliu@fmmu.edu.cn*

学时 上机

2 0

8.1 目的

能够解释常用随机类全局优化算法的类型和特点

重点：各种优化算法的基本原理

难点：各种优化算法的基本原理

课程设计如下：（重点讲解 MCMC 优化算法和模拟淬火算法，同学们只需要知道基本原理、思路、会用这两类算法就行）。在回顾中，重点讲解何为优化、优化的目标函数、主要方法。随后，从全局最优化这一**全局、随机**概念讲起，给出针对单变量函数的优化方法：MCMC 方法（Boltzman 分布）以及模拟淬火算法。最后我们进入正文，开始从历史故事讲起，细致的讲解 MCMC 方法的来龙去脉。根据授课情况，给出 Rosenbrock 函数、甚至 FEM 逆问题的计算实例。

注意控制节奏、不要急。这部分内容抽象，但历史趣味盎然，综合了概率论、随机过程（马尔科夫过程）、优化、采样等多个理论，讲得好了对开阔学生们的思路极为有益。

8.2 引言

随机优化算法解决的问题是什么呢？

对于任意一个优化问题，需求解

$$\min_x f(x), \quad a < x < b$$

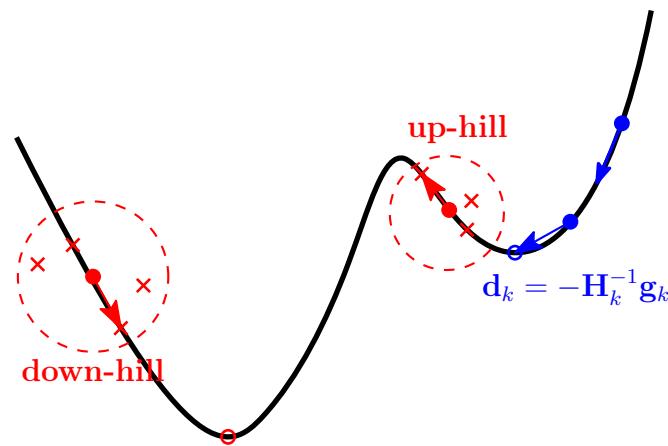
最直观的（暴力求解，brute force）思路就是，我们可以在自变量 x 的区间 $[a, b]$ 上不停的生成随机的点 x_i 。那么仿真 T 次之后，提取 $f(x_i)$ 最小的那个就行。

问题来了：

1. 如何保证这一优化过程（或者函数 f ）是收敛的呢？也就是说当 $f(x)$ 存在极小值，最多经过 T 迭代就可以找到极小点；
2. 如何处理维度灾难（Curse of dimensionality）呢？例如，当 $x \in \mathbb{R}^N$ 时，若对每一维度下的区间随机选择 P 个点，那么总粒子数目将达到 P^N 。

马尔科夫链—蒙特卡罗算法（Markov Chain Monte Carlo, MCMC）就是针对以上问题的、有效的随机优化算法。其特点是与暴力算法的随机——选择相比，采用了 Random — Move — Select。

全局优化算法——顾名思义——就是寻找待优化函数全局极小值的优化算法。该体系下的算法往往为随机算法。与牛顿优化这类确定性算法对比，其区别在于：在每次迭代时，牛顿算法在牛顿方向 $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ 上寻找极小值；而全局优化算法则在待搜寻区间内，随机生成函数的迭代点，进行取舍。全局优化算法的一个重要特性是上山性（up-hill）。



全局随机优化算法中的代表算法是马尔科夫链—蒙特卡洛（Markov Chain Monte Carlo, MCMC）算法，上山性（Up-hill）使得该算法为脱颖而出的随机类优化算法。在 SIAM News 中，MCMC 算法被评为 20 世纪十大算法之一。

同学们感兴趣可以看（希望 21 世纪的十大算法有你们一份）：

1. **1946:** John von Neumann [fon Noy-mon], Stan Ulam, and Nick Metropolis, all at the Los Alamos Scientific Laboratory, cook up the Metropolis algorithm, also known as the **Monte Carlo method**.
2. 1947: George Dantzig, at the RAND Corporation, creates the **simplex method** for

linear programming.

3. **1950:** Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysis at the National Bureau of Standards, initiate the development of **Krylov subspace iteration methods**. $Kx_{i+1} = Kx_i + b - Ax_i$, the remainder $r_i = b - Ax_i$.
4. 1951: Alston Householder of Oak Ridge National Laboratory formalizes the **de-compositional** approach to matrix computations.
5. **1957:** John Backus leads a team at IBM in developing the **Fortran optimizing compiler**.
6. **1959–61:** J.G.F. Francis of Ferranti Ltd., London, finds a stable method for computing eigenvalues, known as the **QR algorithm**.
7. **1962:** Tony Hoare of Elliott Brothers, Ltd., London, presents **Quicksort**.
8. **1965:** James Cooley of the IBM T.J. Watson Research Center and John Tukey of Princeton University and AT&T Bell Laboratories unveil the **fast Fourier transform (FFT)**.
9. 1977: Helaman Ferguson and Rodney Forcade of Brigham Young University advance an **integer relation detection algorithm**.
10. 1987: Leslie Greengard and Vladimir Rokhlin of Yale University invent the **fast multipole algorithm** (N-body simulations).

下面来看 Metropolis Hastings (MH) 算法和模拟退火 (Simulated Annealing, SA) 算法的主要架构，希望给同学们一个初步的认识。

首先，我们将一些算法中的变量按照其最原始的物理含义进行定义

- 自变量 x 代表系统的状态 (state)
- $f(x)$ 代表系统的能量，这里能量不分正负 (物理中能量是正值)。在优化问题中，能量总是越低越好
- T 是系统的温度， kT (k 为波尔兹曼常量) 定义为能量单位

那么 MCMC 算法 (下面这个算法称作 Metropolis–Hastings 算法) 的迭代步骤为：

1. $x' = x + \text{random change}$
2. if $\exp -\frac{f(x') - f(x)}{kT} > R(0; 1)$ then $x = x'$
3. goto 1

其核心思想为，在自然界中，物理系统总是倾向于收敛在能量最低的状态上。

那么在最初迭代的几步之后，系统的状态 x 会在极小值附近震荡，而震荡的幅度则正比于系统的温度 T 。较高的温度可以避免状态 x 卡（聚集）在局部极小值点，而较低的温度（系统活跃度低）下 x 则聚集于（clustering）全局极小值附近。MCMC 算法的关键在于允许偶尔的 uphill，这一特性可以使状态跳出局部极小值，避免优化算法局部收敛。

与之类似的，模拟退火算法（SA）在 Metropolis-Hastings 算法基础上得出，其迭代步骤为

1. decrease T
2. $x' = x + \text{random change}$
3. if $\exp \frac{f(x) - f(x')}{kT} > R(0; 1)$ then $x = x'$
4. goto 1

与 MH 相比，SA 算法仅仅多了一个递减的系统温度 T 。

我们可以这样联想，在温度较高时（例如冶金），系统的分子（状态 x ）处于无序模式，因而可能存在于能量 $f(x)$ 上的任意一点，随着温度的逐步降低，系统会趋向于能量最低的状态。这一思想与实际中的**淬火概念（图）**是类似的，因而 MCMC 算法也被称作最为自然（Natural）的算法。

接下来我们从 MCMC 算法的历史开始讲解。它的历史伴随着计算机的发展，并且是计算机模拟技术第一次在最优化问题中展露头角，极其精彩。

8.3 简介

马尔科夫链蒙特卡洛（MCMC）算法实际上是相关算法——Metropolis-Hastings、模拟退火、以及吉布斯采样——的集合。MCMC 算法起源于物理学，而今其最前沿的分支已成了计算统计学的核心。该算法在物理学、统计学、应用数学及计算机科学中应用广泛。在它的难以置信的潜能被发掘前，算法的发现者就知道他们或许找到了：

… 一种通用的方法，适用于快速电子计算机，去计算那些由相互作用的分子所构成的物质的属性。

MCMC 算法起源于一篇开创性的论文。这一原创的思想需要地点、人物与灵感的完美组合。**地点**是二战结束后不久的洛斯阿拉莫斯 (Los Alamos)。**人物**有大家所熟悉的——冯诺依曼、乌拉姆、泰勒以及其他一些大家不是特别熟悉的。

灵感是随机理论和采样可以解决传统解析算法无法处理的物理问题。当然还有最后一样必不可少的东西：计算机。

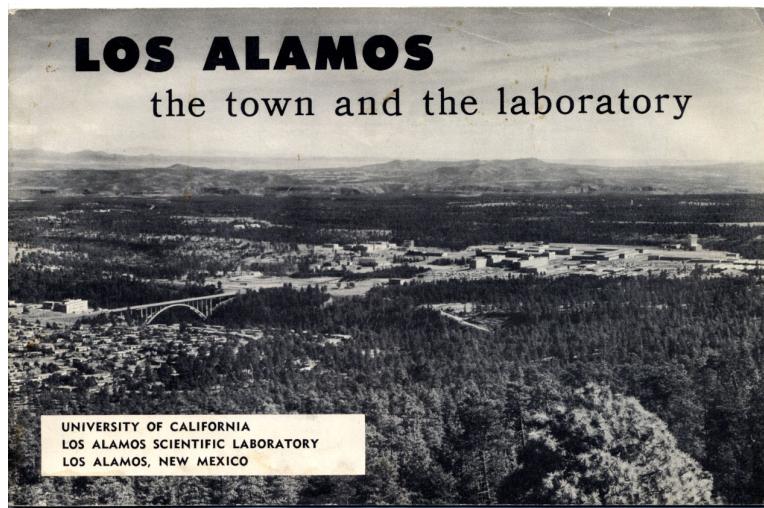
MCMC 方法的演变由来自不同领域的学者共同推进。在每一个关键的时间节点，都有一篇决定性的论文标志着该算法被拓展并进入了一个新的领域。我们的故事将顺着这些论文的时间次序展开：

1. *Equations of State Calculations by Fast Computing Machines*, 1953, by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller, 首次提出 Metropolis 算法
2. *Optimization by Simulated Annealing*, 1983, by Kirkpatrick, Gelatt, and Vecchi, 在应用数学问题中引入了模拟退火算法
3. *Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images*, 1984, by Geman and Geman, 得到吉布斯采样 (Gibbs Sampling)
4. *Sampling-Based Approaches to Calculating Marginal Densities*, 1990, by Gelfand and Smith, 将 MCMC 方法引入统计学。

这些论文的影响力以及 MCMC 方法在现代应用数学中的地位难以估量。以上四篇论文总共被索引将近 **80,000 次 (2015 年 Google Scholar, 87421)**，而原始的 Metropolis 算法被称为 20 世纪十篇最重要的算法之一。

8.4 起源, Metropolis Et. Al., 1953

8.4.1 蒙特卡洛方法



二战结束后不久，洛斯阿拉莫斯就成了美国那个年代应用数学与理论物理

研究的中心。此时这里所承担的是核能武器的研发（如 Manhattan Project），在这一国运项目中，催生了当前我们广泛使用的最优化控制理论、优化算法中的大部分工作。

研究中遇到的一个难题是估计大量（如 10^{23} ）原子核的行为。在这一过程中，人们已经完全掌握决定它们的行为的物理定律——如热力学、统计力学、量子力学——而这些理论本质上都是概率的，但是在实际应用中却又极其复杂，以致于传统的解析方法无法进行细致深入的分析。在这种情况下，人们自然而然的想到了一种新的解决思路：与其寻找解析解，不如去仿真整个系统的行为，进而用仿真结果估计期待的答案（例如系统的状态）。但是如何进行仿真是当时的难点。

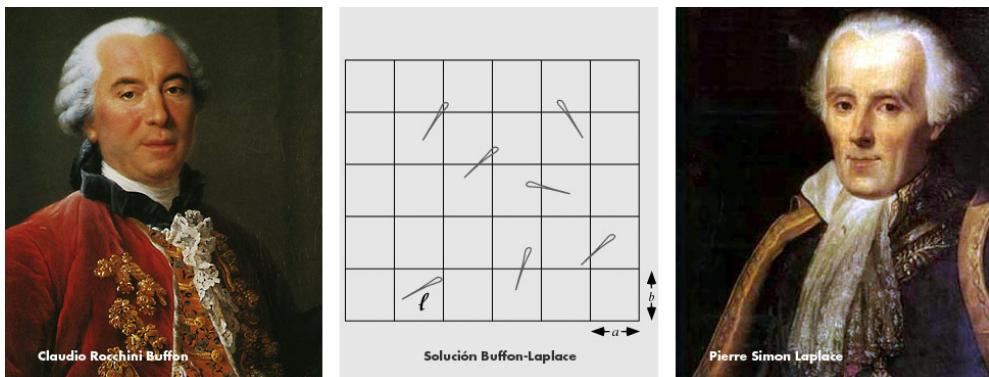
与现在截然相反的是（现在我们可能想到的第一个解题思路不是解析解，而是仿真），40 年代末以前，没有任何设备能够快速准确的进行大规模的、随机的仿真。当二战结束时，这一情形得以改观。洛斯阿拉莫斯的研究人员有了一种新研发出来的计算设备——位于宾西法尼亚大学的 ENIAC（Electronic Numerical Integrator And Computer，电子数值积分计算机）。



早在计算机发明之前，人们就已懂得使用随机仿真。18 世纪的布冯丢针实验（有待考证）就是利用“手动”仿真去估计¹感兴趣的确定性变量——圆周率 π 。另一个有趣的，有详细文献的仿真实验是 kelvin 爵士的 1901 年的论文 *Nineteenth Century Clouds Over the Dynamic Theory of Heat and Light*²，其中他描述了一种通过精心设计的（人为）一组卡片生成仿真数据，用以估计速度分布的方

¹设针长度为 l ，平行线间距为 t ，总共丢针 n 次，有 h 次针与线相交，则 $\pi \approx (2ln)/(th)$

²这篇文章除了描述蒙特卡洛仿真之外，还在现代物理学的发展历史中具有重要地位。Kelvin 爵士指出 20 世纪物理学的关键问题，即“紫外灾变 (Ultraviolet Catastrophe)”与关于“异常”光速的迈克尔逊 – 莫雷实验。



法。也有迹象表明在早在 30 年代，费米在进行核裂变的早期研究中，使用过类似蒙特卡洛仿真的方法¹

而在洛斯阿拉莫斯，随机仿真的引入要归功于乌拉姆（Stanislaw Ulam）。1946 年的乌拉姆因病不得不待在床上休养身体。为了打发时间，他玩起了坎菲尔德纸牌。这种纸牌游戏胜率较低，但是一旦洗完牌后结果就确定了。他玩着玩着就想，规则是固定的，但是如何能够确定这种纸牌游戏的胜率。显然，这种情形没法解析计算。但是他想到，可以利用 ENIAC 模拟洗牌，之后利用游戏规则计算胜负。那么，只需要重复这一计算过程多次，就可以获得关于这个游戏胜率的经验估计。但是显然，用洛斯阿拉莫斯宝贵的、新兴的计算机资源分析纸牌游戏显然行不通。但是乌拉姆发现这种思路同样适用于实际中更为重要的情形。于是，他将这个想法告诉了他的朋友冯诺依曼。

1947 年，冯诺依曼和他的同事致力于估计裂变设备（如原子弹）中中子的扩散速率和倍率。根据乌拉姆的建议，冯诺依曼提出一个简单的设想：利用计算机，创建相对来说较多的“虚拟（virtual）”中子，然后用随机仿真，模拟它们在裂变材料中的演变过程。仿真结束后，只需计算剩余的中子数量，即可得到裂变速率。在现在看来，仿真规模是相当小的：仅仅生成 100 个中子，且每个中子仅设定 100 次碰撞。在 ENIAC 上，仿真大概需要耗费 5 个计算机时。即便如此，该方法的实用性立马显现出来。从那以后，随机仿真——不久便被称作蒙特卡洛方法——成为统计物理学中的一种重要技术。

¹1955 年，诺贝尔终身成就奖得主赛格雷回忆：

… 通过结合直观经验和蒙特卡洛计算，再加上正统的物理理论，费米获取了关于慢中子行为极不寻常的认识。

但是蒙特卡洛方法这个名字却是 Nicholas Metropolis 命名的

… 我建议对这种统计学方法使用一个显而易见的名字——这个建议的出发点是，乌拉姆的叔叔常常向他的亲戚借钱，只是为了能去蒙特卡洛（赌博）。于是，这个名字于是便被沿用下来。

这个方法最初出现在乌拉姆和 Metropolis 于 1949 年合作发表的一篇论文上：

我们指出，现代计算机将极其适用于执行本文所描述的方法。实际使用中，只需提供一组参量值描述粒子特性，比如，可使用一组数据卡片（当时计算机是打卡编程的）。… 该机器可以仿真生成随机数列，尽管看上去很奇怪，但是确实是可行的。事实上，它足以生成 $[0, 1]$ 区间内均匀分布的随机数…¹

在我们现在看来，很难认识到将仿真法作为解析法之外的另一种替代方法这一思想的革命性。但是在那个年代，几乎很少有数学家或物理学家能够使用电子计算机，当然就更别提仿真了。

除了从均匀分布中生成采样序列之外，之后还出现了针对其它分布的采样算法。对于常用的标准分布（如正态分布），通过对均匀分布的数学变换便可得到。对更为广义的分布，尤其是由物理（贝叶斯）模型所导出的，则需更为复杂的技巧。在仿真算法的早期，同样也是由冯诺依曼提出的一种被称作取舍采样的算法被广为使用。但是这些方法还远远不够通用，也不适用于高维概率分布。与之相对的，MCMC 算法则克服了这类限制。

再次强调一下，算法研究的初衷是从某一个概率密度分布 p 中获得采样。同学们需要注意的是，如果在优化算法中，能够获得概率密度的采样；那么反过来利用采样近似概率密度分布，我们就能够实现优化算法。MCMC 算法的关键在于使用了马尔可夫链。下面，我们将在简要回顾一下马尔科夫链这个概念。

8.4.2 马尔可夫链

此处讲重点，学生很难直接接触并掌握 MC 理论。

最为直观的，马尔科夫链定义了一种转移概率，而这个转移概率仅仅与上一个状态相关，而与更早时刻的状态无关。

¹那个时代典型的随机数生成方法是“取中法”。令 r_k 表示 n 位随机数，取平方并取出中间 n 位作为 r_{k+1} ，依次迭代。而线性同余法在那不久之后由 lehmer 等人发展起来。

给定一个有限状态（也被称作配置， Configuration）空间 $\mathbb{S} = \{1, 2, \dots, N\}$ 下，马尔可夫链是由序列随机变量定义的一个随机过程，对 $X_i \in \mathbb{S}$, for $i = 1, 2, \dots, N$ 有：

$$\text{Prob}(X_{k+1}|X_1, \dots, X_k) = \text{Prob}(X_{k+1}|X_k)$$

也就是说，处于 $k + 1$ 时刻的概率只与 k 时刻的状态有关。我们只考虑转移概率与时刻 k 无关的马尔可夫链（即非时变马尔可夫链）。由此可得一个 $N \times N$ 大小的转移矩阵 $\mathbf{P} = (\mathbf{p}_{ij})$ ，定义为：

$$\mathbf{p}_{ij} = \text{Prob}(X_{k+1} = j | X_k = i).$$

也就是说， \mathbf{P} 矩阵 K 次方 (i, j) 位置处的概率即为从状态 i 到状态 j 的 K 步转移概率。并且，注意到对于 $i = 1, 2, \dots, N$ ，有（归一化）

$$\sum_{j=1}^N \mathbf{p}_{ij} = 1.$$

在实际应用中，我们希望马尔可夫链具备以下两个重要特性：

- 不可约 (*irreducible*)：对于所有状态 i, j ， K 步转移概率 $(\mathbf{P}^K)_{ij} \neq 0$
- 非周期 (*aperiodic*)：对于所有状态 i, j ， $\text{gcd}\{K : (\mathbf{P}^K)_{ij} > 0\} = 1$

一个不可约、非周期的马尔可夫链在状态空间 \mathbb{S} (π_i 为状态 i 的概率) 上必定具备唯一分布 $\pi = (\pi_1, \pi_2, \dots, \pi_N)$ ，且满足

$$\pi = \pi \mathbf{P}$$

那么，我们就说满足以上两个特性的马尔可夫链在分布 π 上是稳定的，或者称 π 为马尔可夫链的稳定分布。

MCMC 方法基于下面这个结论：

已知 π 是不可约、非周期马尔可夫链的稳定分布，那么我们可以利用马尔可夫链 \mathbf{P} 获得稳定分布 π 的采样

接下来就水到渠成了，为了获取稳定分布 π 的采样，可随机选择 $s_1 \in \mathbb{S}$ ，对于任意 $k > 1$ ，如果 $s_{k-1} = i$ ，则按照概率 \mathbf{p}_{ij} 选择 $s_k = j$ 。那么，生成的序列 s_1, s_2, \dots 当 $M \rightarrow \infty$ 时，下式可以概率 1 (一定) 得到。

$$\frac{|\{k : k \leq M \text{ and } s_k = j\}|}{M} \rightarrow \pi_j \quad (8.1)$$

序列的任意足够大、有限的部分都可近似作为稳定分布 π 的采样。人们通常舍弃序列的前 m 个元素，而只使用序列的“尾部”

$$s_{m+1}, s_{m+2}, \dots, s_M$$

作为稳定分布的采样。

无论以何种方式获得的 π 的采样，我们都可以从采样中估计稳定分布 π 的属性。比如，令 f 为状态空间 \mathbb{S} 上的任一实变函数，我们想要估计函数的期望值

$$E[f] = \sum_{i=1}^N f(i)\pi_i$$

其中 π_i 是 $f(i)$ 的概率密度。那么，我们可以随意选择 π 的一个采样 s_1, s_2, \dots, s_M ，由遍历定理可得

$$\frac{1}{M} \sum_{i=1}^M f(s_i) \rightarrow E[f] \quad (8.2)$$

在 $M \rightarrow \infty$ 时按 $O(M^{1/2})$ 收敛。

在已知不可约、非周期马尔可夫链的转移概率矩阵 \mathbf{P} 后，如何从中获得稳定分布 π 的采样最多只能算一个课后练习题。

我们最为感兴趣的是它的反问题：给定分布 π 和一状态空间 $\mathbb{S} = \{1, 2, \dots, N\}$ ，寻找 π 上稳定分布的不可约、非周期的马尔可夫链。答案就是 Metropolis 算法。

8.4.3 统计力学和玻尔兹曼（Boltzmann）分布

统计力学是物理学中的一门重要分支，主要研究系统中大量的、相互作用的粒子的平均行为，Metropolis 算法的出发点是为了获取统计力学中的玻尔兹曼分布的特性参数。首先，我们简单阐述下玻尔兹曼分布的几个关键思想。

粒子所处的状态可以认为是配置空间 Ω 的一个配置 ω 。对于一阶函数 $f(x)$ 而言， Ω 为定义域，而 ω 为自变量 x 。配置空间可以是有限的或无限的、离散的或连续的。比如，如果我们考虑 N 个相互作用的粒子，每一个粒子可由它在三维空间中所处的位置坐标和速度矢量描述。在这种情况下 Ω 就是一个无穷的、6 自由度连续空间 \mathbb{R}^{6N} 的一个子集。另外，从 Ω 也可取出一个有界子集 Λ ，由空间中整数坐标的网格构成。在整数坐标的网格中的每个位置上我们都

可赋一个值，如 ± 1 ，用来表征该处粒子是否存在，或者表征粒子的指向（甚至可代表例子的“自旋方向”）。如果 Λ 空间的大小为 $|\Lambda| = N$ ，则由于每个配置处存在 ± 1 两种可能，则配置空间由 2^N 种赋值方式构成。

接下来，作用于配置空间 Ω 上的物理规则由能量函数 $E : \Omega \rightarrow \mathbb{R}^+$ 描述。我们用 $E(\omega)$ 代表配置 ω 的能量。对于之前连续配置空间 Ω 的情况，能量为粒子的势能。而对于离散配置空间 Λ ，能量则表征所有相邻粒子之间的相互作用。后者正如依辛（Ising）模型一样。这一点我们马上会讲到。

统计物理学的一个基本原理是“自然总是在寻找能量最低的配置方式”。例如，房间中随机分布的分子即受到这一原理的制约。基本不可能存在的配置（比如所有分子都挤在房间的一角）具备较高能量，因而仅仅以极小概率存在。常见的配置（比如分子各向同性的均匀分布于屋子中）具备较低的能量和极高的概率，而概率足够大以至于基本上是唯一能被观察到的配置方式。

对于处在均衡状态下的系统，某一配置 ω 的（相对）频率由其玻尔兹曼权值给出：

$$e^{-E(\omega)/kT} \quad (8.3)$$

其中 T 是温度值， k 是玻尔兹曼常量。

对于任一配置 $\omega \in \Omega$ ，其概率密度可用玻尔兹曼概率 $Boltz(\omega)$ 表示

$$Boltz(\omega) = \frac{e^{-E(\omega)/kT}}{Z} \quad (8.4)$$

分母 Z

$$Z = \sum_{\omega' \in \Omega} e^{-E(\omega')/kT}$$

被称作配分函数。在实际应用中，配分函数 Z 无论从解析上还是数值计算上都是难以处理的。这一难题直接导致了统计力学中极度缺乏解析解和闭式解。

能量与概率之间的关系可以导出很多有趣的物理量。比如，系统的总能量 $\langle E \rangle$ 即为所有配置能量函数 $E(\omega)$ 的期望值：

$$\langle E \rangle = \sum_{\omega \in \Omega} E(\omega) Boltz(\omega) = \frac{\sum_{\omega \in \Omega} E(\omega) e^{-E(\omega)/kT}}{Z} \quad (8.5)$$

其他类似的物理量定义方式与之类似。但是不可避免的，每一个参量的定义都需要计算配分函数 Z 。

式(8.5)可以直接用蒙特卡洛采样近似，从 Ω 中均匀的生成 $\omega_1, \omega_2, \dots, \omega_M$ ，分别估计式(8.5)中的分子和分母，可得

$$\langle E \rangle \approx \frac{\sum_{i=1}^M E(\omega_i) e^{-E(\omega_i)/kT}}{\sum_{i=1}^M e^{-E(\omega_i)/kT}} = \sum_{i=1}^M E(\omega_i) \cdot \left(\frac{e^{-E(\omega_i)/kT}}{\sum_{i=1}^M e^{-E(\omega_i)/kT}} \right)$$

Metropolis 等人知晓从配置空间中均匀采样的缺点，提出了另外一种方法。

这种（原有的随机采样）方法不可行… 因为，我们极有可能选择了一种配置 ω ，而其概率 $\exp(-E/kT)$ 却相当小；也就是说即选择了一种具备较小权值的配置。我们提出的方法可称之为修正的蒙特卡罗方法：与其随机采样生成配置，再乘以概率 $\exp(-E/kT)$ 加权，我们不如直接按照概率 $\exp(-E/kT)$ 生成配置，并均匀加权。

换言之，更好的方案就是直接从 Ω 中采样，使得每一个配置 ω 都以概率 $Boltz(\omega)$ 生成。如果可行的话，那么对于任意采样 $\omega_1, \omega_2, \dots, \omega_M$ 有

$$\frac{1}{M} \sum_{i=1}^M E(\omega_i) \rightarrow \langle E \rangle$$

由先前所述，上式以 $O(M^{-1/2})$ 收敛，并且无需计算配分函数 Z 。尽管思路简洁有力，但是难点在于如何从玻尔兹曼分布中生成采样序列。

8.4.4 Metropolis 算法

Metropolis 算法的天才之处就在于得到了易于计算的、在**玻尔兹曼分布**上稳定的马尔可夫链。那么，随后只要利用该马尔可夫链，就可以很容易获得玻尔兹曼分布的采样。

该构建算法只需要知道玻尔兹曼权值，即式(8.3)，而非玻尔兹曼概率式(8.4)，这样就可以避免讨厌的配分函数 Z 。为了欣赏 Metropolis 算法的思路和推导过程，我们这里将重新复现 Metropolis 等人 1953 年论文中的工作。

Metropolis 算法包括一个很大的、但有限的配置空间 Ω ，一个能量函数 E 以及恒定的温度 T 。令 $\tilde{\Omega}$ 为 Ω 的重采样， $\tilde{\Omega}$ 可大于 Ω 。通过添加和删除某些配置，我们期望 $\tilde{\Omega}$ 能够变成（或趋近于）玻尔兹曼分布的采样。

假定重采样空间的大小 $|\tilde{\Omega}| = \tilde{N}$ ，令 N_ω 为 $\tilde{\Omega}$ 中任意配置 ω 出现的次数。

配置符合玻尔兹曼分布这一概念等价于

$$\frac{N_\omega}{\tilde{N}} \propto e^{-E(\omega)/kT}$$

或者等价的，对任意两个配置 ω 与 ω' ，有

$$\frac{N_{\omega'}}{N_\omega} = \frac{e^{-E(\omega')/kT}}{e^{-E(\omega)/kT}} = e^{-\Delta E/kT} \quad (8.6)$$

其中 $\Delta E \triangleq E(\omega') - E(\omega)$ 。注意这个比值是与配分函数 Z 无关的。

为了从随机的能量分布 E 中获得理想的玻尔兹曼分布，想像一下从 Ω 到 $\tilde{\Omega}$ 的所有配置上应用（尚待发现的）马尔可夫链：首先，选择一个 Ω 上不可约、非周期的马尔可夫链（请求转移过程），令 $P_{\omega,\omega'}$ 作为从配置 ω 到配置 ω' 的转移概率。同样，假定转移矩阵 \mathbf{P} 是对称的， $P_{\omega,\omega'} = P_{\omega',\omega}$ 。

对于 $E(\omega) \leq E(\omega')$ 的配置 ω 与 ω' 。我们设定：系统允许所有从高能量 $E(\omega')$ 配置到低能量 $E(\omega)$ 配置的转移请求。这一事件发生的次数是

$$P_{\omega',\omega} N_{\omega'}$$

其本质是一个最速下降算法的随机版本：任意“下山（down-hill）”的转移都是允许的。

为了能够达到均衡状态（自然总是能够搜寻到系统的全局极小点），我们必须偶尔允许从低能量 $E(\omega)$ 到高能量 $E(\omega')$ 的“上山（up-hill）”转移。偶尔的含义是：以概率 $\text{Prob}(\omega \rightarrow \omega')$ 进行这种转移。那么，请求进行该“上山”转移的次数是 $P_{\omega,\omega'} N_\omega$ 。因此，总共的转移次数是

$$P_{\omega,\omega'} N_\omega \text{Prob}(\omega \rightarrow \omega')$$

由于 $P_{\omega,\omega'} = P_{\omega',\omega}$ ，则从能量 $E(\omega)$ 到 $E(\omega')$ 的净流量是

$$P_{\omega,\omega'} [N_{\omega'} - N_\omega \text{Prob}(\omega \rightarrow \omega')] \quad (8.7)$$

假设若式(8.6)满足， $\tilde{\Omega}$ 中的能量分布完全符合玻尔兹曼分布，则式(8.7)中的净流量必定为 0。其含义就是物理学中所说的细致平衡。这一约束这意味着那个偶尔的“上山”概率必为

$$\text{Prob}(\omega \rightarrow \omega') = e^{-\Delta E/kT}$$

偶尔允许“上山（up-hill）”转移这一想法即是 Metropolis 算法的神来之笔。

这一进程驱动着能量随机的向着玻尔兹曼分布迁移。假定具备较高能量

$E(\omega')$ 配置的数目多于较低能量 $E(\omega)$ 配置数目，即

$$\frac{N_{\omega'}}{N_{\omega}} > e^{-\Delta E/kT}.$$

则式(8.7)中的净流量为正。那么，其含义是将会有相对更多的从较低能量 $E(\omega)$ 到较高能量 $E(\omega')$ 的转移发生。因此， $\tilde{\Omega}$ 中的能量分布将朝着玻尔兹曼分布（平衡方向）移动。重复这一进程足够多次，就会生成能量分布趋近玻尔兹曼分布的一组配置 ω 。

基于这一论断以及物理学家的直觉，Metropolis 等人认为他们的算法一定能够生成符合玻尔兹曼分布的采样序列。但是在实际应用中需要重点考虑的问题，尤其是对收敛率的理解，可能还要再等上一段时间¹。

我们现在正式给出 Metropolis 算法。假定已选择好请求转移（马尔科夫转移概率密度矩阵）过程。从配置 $\omega \in \Omega$ 到配置 ω^* 的转移可由以下步骤得出

- **Step 1.** 按照请求转移过程²选择 ω'
- **Step 2A.** 如果 $E(\omega') \leq E(\omega)$ ，或者等价的， $Boltz(\omega') \geq Boltz(\omega)$ ，则令 $\omega^* = \omega'$ 。换言之，总是向低能量（高概率）配置迁移
- **Step 2B.** 如果 $E(\omega') > E(\omega)$ ，或者等价的， $Boltz(\omega') < Boltz(\omega)$ ，则以下式概率取 $\omega^* = \omega'$ （即依概率 $Prob = e^{-\Delta E/kT}$ 允许 $\omega \rightarrow \omega'$ 的迁移）

$$\frac{Boltz(\omega')}{Boltz(\omega)} = e^{-\Delta E/kT} > R(0; 1) \quad (8.8)$$

否则取 $\omega^* = \omega$

下面是 Metropolis 算法中几个需要注意的地方：

- 该过程定义了配置空间 Ω 上的一不可约、非周期的马尔可夫链
- 式(8.8)的比值对于 Metropolis 算法的可计算性极其重要，因为它避免了计算配分函数 Z
- 链中的每一步都易于计算，或者至少和请求转移函数一样，如 $E(\omega)$ 及最为重要的， $\Delta E = E(\omega') - E(\omega)$ 。在许多情形中， ΔE 都极其易于计算，且通常都与配置空间大小 $|\Omega|$ 无关
- 由 Metropolis 算法定义的马尔可夫链，在实现时无需知晓全部转移矩阵

¹Metropolis 等人知道随机算法的收敛率在当时是一个公开的问题：“我们的算法并没有给出，经过多久仿真多久才能达到期望分布”。

²请求转移过程很重要，要尽量选择局部化的 ω' ，或者方便计算 $E(\omega')$ 的配置

最初 Metropolis 算法文献中研究的应用是分析硬球模型。该简单模型由非重叠的分子（如气体）构成。尽管硬球模型比较简单，他却是历史上许多物理学家想法的源泉。Metropolis 等人将他们的新算法应用于由 224 个圆盘构成的二维模型上，使系统从任意给定状态转移到趋近均衡态。仿真结论振奋人心：他们从仿真中估计得到的物理量与传统解析方法获得的值相当吻合。最棒的是，计算时间合理，没有出现维度灾难。在当时的仿真中，描述硬球模型信息曲线上的任意一点（每一条曲线都有几百个点），仅需要洛斯阿拉莫斯的 MANIAC (Mathematical Analyzer, Numerical Integrator, and Computer) 计算机 4 或 5 个小时左右的计算时间。

8.4.5 Metropolis 算法与依辛 (Ising) 模型

Metropolis 算法一个直观的应用是二维依辛模型。依辛模型无论在物理学还是应用数学中都被深入研究过。除了验证 Metropolis 算法的有效性之外，依辛模型还在 Geman and Geman 有关图像重建工作中起着重要的作用。

依辛模型可以看作一个简单的铁磁模型，其中所有元素都蕴含了 (a) 邻近单元取向对齐或者 (b) 与外界磁场取向对齐这两种趋势。二维依辛模型定义在 N 个节点的有界平面网格上。每一个节点，都用 ± 1 表征该节点的“旋向”。我们定义 $\omega = (\omega_1, \omega_2, \dots, \omega_N)$ 为依辛模型的一组配置，其中 ω_i 表示第 i 个节点的旋向，因此总配置空间大小为 $|\Omega| = 2^N$ 。

在依辛模型中，一组配置 ω 的能量定义¹ 为

$$E_{Ising}(\omega) = -J \sum_{\langle i, j \rangle} \omega_i \omega_j - H \sum_{i=1}^N \omega_i \quad (8.9)$$

其中 $J > 0$ 表示邻近单元的吸引力， $H > 0$ 代表外界磁场， $\langle i, j \rangle$ 表示节点 i 和 j 是近邻节点，也就是两者共享水平或者垂直边界。在本章中，我们假定没有外界磁场作用（即 $H = 0$ ）以及令 $J = 1$ ，那么

$$E_{Ising}(\omega) = - \sum_{\langle i, j \rangle} \omega_i \omega_j$$

依辛模型引起统计物理学家持久兴趣的一个原因是它具备相变过程。相变即为当某一变量越过临界值时物质的某一属性发生的突变。最熟悉的例子即水

¹加深能量概念的认识，因为后续所有的随机类全局优化算法都是在跟能量打交道

在结冰或者沸腾时的相变；在这种情形下，水的密度在温度越过临界值 $T_c = 0$ （或者 $T_c = 100$ ）时发生突变。

依辛模型的一个重要的相变特征为磁化。对于给定的配置 ω ，定义

$$M(\omega) = \sum_{i=1}^N \omega_i$$

代表温度 T 下系统的磁化值 $\langle M \rangle_T$ ，即为 $M(\omega)$ 的期望值：

$$\begin{aligned} \langle M \rangle_T &= \sum_{\omega \in \Omega} M(\omega) \text{Boltz}(\omega) \\ &= \frac{1}{Z} \sum_{\omega \in \Omega} M(\omega) e^{-E_{ising}(\omega)/kT} \end{aligned} \quad (8.10)$$

在较高温度下，各个状态基本上是均匀分布，因此 $\langle M \rangle_T$ 近似为 0；值得注意的是，此时各个节点之间互不相关。当温度降低时，就会发生自发磁化：即存在一临界温度 T_c ，低于该温度值，节点间仅存在远距离的相互作用。统计物理学中一个著名的结论是 Osager 对于二维依辛模型临界温度值的精确计算：

$$kT_c/J = \frac{2}{\ln(1 + \sqrt{2})} \approx 2.269$$

我们可用 Metropolis 算法直观显示 N 节点依辛模型磁化的相变过程。为了实现 **Step 1**，我们需要选择一个从配置 ω 到 ω' 的请求转移过程。一个最简单的办法是均匀随机的从 $1, 2, \dots, N$ 中选择一节点 i ，对于节点 i ，以概率 $1/2$ 翻转配置 ω_i 到他的相反状态；若否则保持当前值。对于所有的 $j \neq i$ 有 $\omega'_j = \omega_j$ ，也即只有一个节点 ω_i 受到影响。这一配置元素的请求转移过程是不可约的、非周期的以及对称的。

对于 **Step 2**，我们必须决定是否保留状态 ω' ，依据为能量的变化量：

$$\Delta E = E_{ising}(\omega') - E_{ising}(\omega) \quad (8.11)$$

$$= (\omega'_i - \omega_i) \sum_{\langle i,j \rangle} \omega_j \quad (8.12)$$

其中求和是对第 i 节点最邻近的四个节点进行的。因此 ΔE 只取决于节点 i 的邻近四个节点的旋向，因此更新节点的计算量很小并且与节点数目 N 无关。这一与局部结构的相关性，称之为局部特性，是 Metropolis 算法（MCMC 算法）共有的常见特性。

MCMC 算法中另一个常见的——难缠的——问题是算法收敛性。一般很难估计逼近目标分布所需要的算法迭代次数。除此之外，马尔可夫链的一个难以避免的问题是采样序列间的相关性。这意味着遍历整个配置空间需要很长很长时间，尤其是在临界温度附近，现象会更加有趣¹。有关依辛模型在临界温度附近以至远离临界温度的收敛特性可参见 Diaconis 的综述。

8.4.6 依辛模型仿真

图 8.1 显示了一个 100×100 大小的依辛网格的两幅截图：黑色的表示表示 +1 的旋向，白色的表示 -1 的旋向。左边的二维网格低于临界温度，中间的近似为临界温度值，右边的则处在相变临界温度以上。在每一种情形，我们运行 Metropolis 算法足够长的时间，使生成的序列能够代表玻尔兹曼分布的一个采样。最右边的图中，可以明显看出相距一定距离的节点之间的自旋几乎毫不相关。左边图像中，相距较远节点的自旋之间则具备明显的相关性。这一本质差别意味着依辛模型处于两个明显的、截然不同相位中。

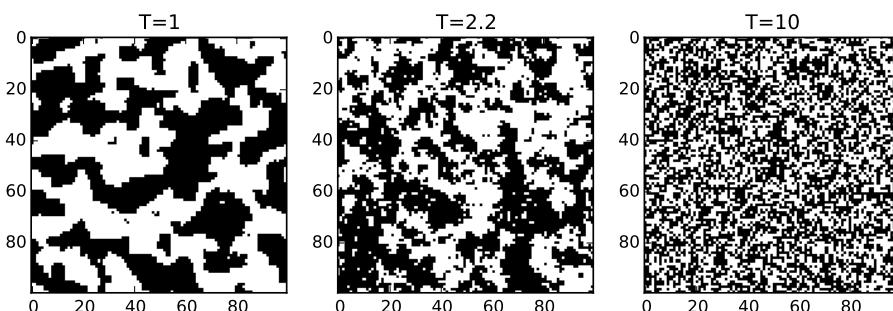


图 8.1 Metropolis 算法仿真 — 100×100 大小的依辛模型。左边的图片低于临界温度 $kT_c/J \approx 1$ ，节点之间具备明显的远距离相关性。中间的图片温度取临界温度附近 $kT_c/J \approx 2.2$ ，右边的图片大于临界温度 $kT_c/J \approx 10$ ，节点之间不具备显著的远距相关性。

¹Swendsen-Wang 算法针对第二个问题提出了一个有效、优雅的解决方法。他们这个版本的 Metropolis 算法用相似自旋的方式更新整个簇，因此加快了遍历整个配置空间所需的时间，即便是在靠近甚至低于临界温度时。Swendsen-Wang 算法有趣的地方是引入了“关联”变量，这一思想也出现在 Geman 和 Geman 有关图像重建的工作以及贝叶斯分层模型中。

8.4.7 荣誉归属

那么谁是 Metropolis 算法真正的发明人呢？Metropolis 最早的文献有五个合作作者（还包括两对夫妻组合），因此任何对于“所有权”的声明都很难定夺。随着时间的流逝，答案很可能永远无法知晓。

有关这一话题的最终声明可能来自 Marshall Rosenbluth，他是那五个合作作者的最后一个幸存者。在 2003 年去世前的几个月，他简单的回答了这个问题。在一个庆祝 Metropolis 算法诞生 50 周年的会议上，他说：“Metropolis 除了提供上机时间外，没在算法的研究中发挥任何作用”。Rosenbluth 接着说使用蒙特卡洛方法的这一思路来自他和乌拉姆、冯诺依曼的讨论，并且泰勒也提出过有用的建议，把一般的取平均替换为加入了玻尔兹曼分布权重的平均。同时，他还说他和他的妻子以及另外一位合作者阿丽亚纳，完成了大部分重要的工作。

与此同时，有迹象表明加利福尼亚理工大学和劳伦斯利沃莫实验室的 Bernie Alder, Stan Frankel, S.G.Lewinson 与 J.G.Kirkwood 在 40 年代独立的提出了几乎相同的想法。当被问及时，Alder 对他们的角色相当明确：

我猜可能是我们首先在加利福尼亚理工（首先）做到的。得到那个
算法并不难，顺便说一下，我认为它是最强大的算法之一。… 事
实是，我们从未发表我们的方法——因为你不能发表任何你们老板从
不相信的东西！

缺乏老板的支持并不是 Alder 和他的同事唯一缺乏的。另外一个条件他们也同样不具备，那就是便利的使用及实现这一算法所不可或缺的工具：计算机。

有趣的是，在 1947 年，Frankel 和 Metropolis 合作完成了第一篇演示如何用计算机实现数值（并非蒙特卡洛）积分的应用文章。毫无疑问，在那个时代，这些人之间的存在密切的交流并相互影响，因而我们现在称之为 Metropolis 的算法，这一荣誉应当被更多人共享。

8.4.8 插曲

在 1950s 到 1980s，大多数对于 Metropolis 算法的兴趣都来自物理学领域。一个例外是 Hammersley 和 Handscomb 的 1964 年的经典蒙特卡洛算法。这一漂亮的——同时也是相关的——著作代表了那个时代蒙特卡洛算法的最前沿，同时也包括了一篇 MCMC 方法应用于统计力学的综述文章。

那个时代的物理学家们忙于研究通用的 Metropolis 算法，而大多数算法都应用于类似于依辛模型的自旋模型。这些文章中的第一篇来自 Glauber¹ 1963 年提出的“热泳”。Glauber 在算法中提出对节点顺序操作。在第 i 个节点，旋向 ω_i 由局部玻尔兹曼权值决定：

$$\text{Prob}(\omega_i = s) = e^{-(s \sum_{\langle i,j \rangle} \omega_j)/kT}$$

其中能量的求和与通常一样是对节点 i 的所有邻近节点进行。有趣的是，Glauber 的出发点是理解自旋模型时域相关（非均衡）动力学的属性，而不是去开发一种新的计算工具。十年以后，Flinn 描述了一种称作“自旋交换”的算法，从数值计算方面研究依辛模型的相变问题。Creutz 在 1979 年展示了如何将单节点更新应用于 $SU(2)$ (special unitary group of degree 2) 的规范场论。

另一个算法通用化工作出现在 1965 年，Barker 引入了另一种 Metropolis 构建算法，得出了以玻尔兹曼分布为稳定分布的新的马尔可夫链。这些变体的层出不穷引出了这样一个问题：究竟有多少类 Metropolis 算法？在这些之中又有哪些是最优的？

这个问题的答案出现在 1970s，由统计学家 W. K. Hastings 完成。他可能是第一个预见到 Metropolis 算法可能是一个 (perhaps, the) 通用的采样算法。在一次采访中，Hastings 回忆到他是如何碰到这个算法的：

[化学家们] 使用 Metropolis 算法用来分析势场中的粒子所构成系统的平均能量。每一个粒子有六个自由度，那么由 100 个粒子构成的系统包括 600 维。当我得知他们用马尔可夫链那么方便的从高维分布中生成采样序列时，我便意识到这种方法对于统计学家是多么的重要。因此，我用全部的时间来研究者个方法和它的变体，最终导致了 1970 年的那篇论文。

这篇 1970 年的论文就是 *Monte Carlo Sampling Methods using Markov Chains and Their Applications*，其中，Hastings 将 Metropolis 算法提炼为最基本的数学框架。他也证明了使用 Metropolis 算法可从一些标准分布中生成随机采样序列，同时也包含了从正交矩阵群中采样这种情形。但是当时他并没有发现这篇论文的重要

¹Glauber 同样因对相干光学的量子理论做出的贡献而为众人所知，他因此分享了 2005 年的诺贝尔奖。

性。只有等到 20 年后，经由 Gelfand and Smith 所发表的论文，才使得这一思想被大众接受。因而在统计学领域，Metropolis 算法常被称作 Metropolis–Hastings 算法。

让我们简单的了解一下 Hastings 对 Metropolis 算法的通用化所做的工作。假定对分布 π 采样，任选一在状态空间 \mathbb{S} 上类似 Metropolis 算法的请求转移过程 $\mathbf{Q} = (\mathbf{q}_{ij})$ ，与原始的 Metropolis 算法不同，该（请求转移矩阵）无须对称。定义转移矩阵 $\mathbf{P} = (\mathbf{p}_{ij})$ 为

$$\mathbf{p}_{ij} = \begin{cases} \mathbf{q}_{ij} \alpha_{ij} & \text{if } i \neq j \\ 1 - \sum_{k \neq i} \mathbf{p}_{ik} & \text{if } i = j \end{cases} \quad (8.13)$$

其中 α_{ij} 由下式给出

$$\alpha_{ij} = \frac{s_{ij}}{1 + \frac{\pi_i}{\pi_j} \frac{q_{ij}}{q_{ji}}}$$

s_{ij} 的取值不作限定，只要满足 (i) 对所有 i, j ，有 $s_{ij} = s_{ji}$ ，(ii) $\alpha_{ij} \in [0, 1]$ 。那么对于任意符合这类要求的 s_{ij} ，很容易验证 π 是 \mathbf{P} 的稳定分布。而对于对称的 \mathbf{Q} ，随意选择 s_{ij} 即变为原始的 Metropolis 算法。

对于给定分布 π ，选择不同的 s_{ij} 将生成截然不同的多个 Metropolis 算法，每一个都能生成 π 上稳定的马尔可夫链。但为什么只有原始的 Metropolis(–Hastings) 算法延续至今呢？Hastings 的学生，P. H. Peskun 给出了原因。Peskun 提出对于所有 s_{ij} 的取值，由式(8.2)式所给出方差的估计中，具备渐进极小值的那个 s_{ij} 就是原始 Metropolis 算法。不管是因为幸运还是直觉，更不知是不是自然的选择，最初的那个马尔可夫链蒙特卡洛方法的例子被证明是最优的。

8.5 模拟退火与组合优化: KIRKPATRICK ET AL., 1983.

尽管在统计物理学中 Metropolis 算法很是流行，并且 Hastings 也预见了它能够成为一种通用采样算法的潜力。但是在 1980 年以前，圈外的人很少了解该算法。这种情况随着 Scott Kirkpatrick, C. D. Gelatt, M. P. Vecchi 的 1983 年的 *Optimization by Simulated Annealing* 这篇论文的出现而发生改变。几乎是在同一时期，捷克数学家 V. Cerny 于在 1985 年发表了论文 *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*，独立的得到了类似的方法。但是 Kirkpatrick 等人的工作更为人所知，其原因是：他们更为深入的发

展了模拟淬火的数学原理，并且将它应用到了一系列问题中。这里，我们更多关注于他们的工作，但是 Cerny 的论文也有它的意义与地位，应当为更多的人所了解。

Kirkpatrick 等人是 IBM 托马斯沃森研究中心的成员。在那个年代，他们正在研究组合优化相关的问题。这类问题为确定性最优化问题，Metropolis 算法表现得出人意料的有效。

组合优化问题主要有两个特点：

1. 选择一代价函数，藉以寻找全局最小值
2. 一离散（常为有限，但巨大）的最小值搜寻空间。实际上，获得全局最小值的近似就是可以接受的最好结果了

典型的一个组合优化问题就是旅行销售员问题。这个问题是给定地图上随机分布的节点（城市），搜寻遍历所有节点的最小化距离。在这个问题中，搜寻空间包括可能的旅行路线，而代价函数是旅行距离。与其它组合优化问题类似，TSP 问题（也可称为决策问题）是 NP 不完全的（**确定是 NP 不完全么**）。

Kirkpatrick 和论文的其它作者曾接受过统计物理学的培训，他们很自然的将代价函数类比为能量函数。在了解到自然总是能有效的寻找到最低能量配置这一特征后，他们就设想利用 Metropolis 算法去选择搜寻空间中的低能量配置。但是，他们发现难点在于，如何合理设置温度参数 T ，这个参数在组合优化中没有一个明显的等价物理量。对于较大的 T ，Metropolis 只会生成一个均匀分布，这与随机搜寻无异。而对于较小的 T ，Metropolis 算法极易陷于局部极小值，而与全局最小值相差甚远。下面，我们将重现 Kirkpatrick 等人最初的证明，看看他是如何处理这个难题的。

8.5.1 电路设计与自旋玻璃

Kirkpatrick 等人最初研究的问题并不是 TSP，而是如何有效的在集成芯片上布置电路元件（比如三极管）。位于同一块芯片上的电路元件之间的通信方便，而不同芯片电路元件之间的通信则会有额外的代价。我们的目标是如何分配电路元件，使得在通信代价最小的情况下，每个芯片上具备大体相当的元件数目。

为了从数学的角度重新分析这个问题，假定要在两块芯片上布 N 个元件，

一个配置 ω 由以下 N 元向量表示:

$$\omega = (\omega_1, \omega_2, \dots, \omega_N)$$

其中 $\omega_i = \pm 1$ 表示第 i 个元件是位于哪块芯片上, 而 a_{ij} 表示元件 i 和 j 之间的信号线(连接线)数目。

Kirkpatrick 等定义芯片间的通信代价为

$$\sum_{i \geq j} \frac{a_{ij}}{4} (\omega_i - \omega_j)^2 \quad (8.14)$$

而两块芯片之间的电路数目不同导致的不均衡代价可表示为:

$$\lambda \left(\sum_i \omega_i \right)^2 \quad (8.15)$$

其中 $\lambda \geq 0$ 表示不均衡“惩罚”。将式(8.14)展开带入式(8.15), 注意到所有的 ω_i^2 都为 1, 那么去掉所有常数项可得代价函数

$$C(\omega) = \sum_{i \geq j} \left(\lambda - \frac{a_{ij}}{2} \right) \omega_i \omega_j \quad (8.16)$$

80 年代初期, IBM 的研究者开发了多种算法来(或近似)最小化 $C(\omega)$ 。但是随着三极管数目 N 从几百增至上千(乃至更多), 这些方法都变得越来越不适用。正如 Kirkpatrick 回忆,

先前的算法很神秘, 如果你仔细查看这些算法, 他们包含着解决一个接一个相互矛盾的问题。我们知道我们可以做得更好(来自 Scott Kirkpatrick 个人通信)

幸运的是, Kirkpatrick 等人知道在统计力学中存在一个与式(8.16)类似的模型, 这个模型就是自旋玻璃。

自旋玻璃与依辛模型类似, 只不过能量函数稍有差别

$$E(\omega) = \sum_{i \geq j} (U - U_{ij}) \omega_i \omega_j.$$

与式(8.16)类似这一点是显而易见的。 U_{ij} 表示的是相邻节点间的局部吸引力(铁磁力), 这些力与用 U 表征的远距抗力(抗铁磁力)对抗。自旋玻璃模型被称作不稳定的是因为他们不具备同时满足吸力与抗力要求的配置。结果是, 低能量的基态无非是绝对对称。

对于自旋玻璃（**总感觉名字怪怪的**）模型以及其它的不稳定性系统，Kirkpatrick 知道必须仔细考虑 Metropolis 算法的使用方法，以便得到低温下的基态。如果系统骤停，即降温过于迅速，则它会停在任何一个非基态的局部极小值上。一个较好的方法是淬火，也就是缓慢的降低温度，使系统逐渐变化到基态。基于这一直观的观察，Kirkpatrick 等人提出了模拟淬火算法。

对于任一组合优化问题，以代价函数替换能量函数，并使用一组由参量 $\{x_{ij}\}$ 定义的配置，我们可直接使用 Metropolis 算法生成在某些（合理）温度下一系列配置的采样，温度与代价函数一样也是可控变量。模拟退火的过程包含首先在高温下“融化”待优化的系统，然后缓慢的逐步降低温度直到系统“冷却”并且没有更进一步的变化发生。在每一个温度，仿真都需要进行足够长时间，以便系统能够缓慢演变到稳定状态。退火策略可以认为是由以下两部分构成，降温的一系列温度值以及在每一温度为了能够达到均衡态所重新分配 $\{x_{ij}\}$ 的次数。

Kirkpatrick 等人将这个算法应用到几个电路设计中的实际问题，同时也以 TSP 为例展示了该算法的有效性。结果令人印象深刻——模拟淬火成功了。几乎是在同时，Černý 将他的模拟淬火仿真算法应用于 TSP，得到了相似的结果。

8.5.2 Kirkpatrick 等人之后

从 1983 年起，模拟退火算法就成了应用数学家工具箱中的一件标准工具。而它所能解决的问题也在不断增长，某种程度上，模拟淬火算法既可以应用于离散问题，也可用于连续问题。它几乎被用在了所有应用数学的领域中，包括运筹学、生物学、经济学以及电子工程。组合优化领域充斥着仅能解决特定问题——甚至特定问题的特殊情形——有效的方法，但是大多方法仅仅是为了应对手头上特定问题所定制的。模拟淬火的流行，在于其有效性和易于实现性的结合：只要给定代价函数与请求转移过程，几乎任何问题都能用上模拟退火算法。

可能是由于缺乏稳健的数学框架，模拟淬火算法费了一些时间才被应用数学社区接受。第一个模拟淬火算法的深入分析出现在 1989 年，由 Johnson, Aragon, McGeoth, Schevon 的一系列论文构成。

模拟淬火算法的计算效率取决于请求转移过程与能量函数之间的关系。一个好的请求转移过程应当尽可能小的改变能量函数，即使得 ΔE 易于计算，通常计算量应当与问题规模无关，具备局部化的特性。在最初电路设计的问题中，请求转移过程为随机选择一块电路并移到另一块芯片上。因而能量函数变化量的计算量与问题规模无关。这一高效的、局部性的优点在 Geman and Geman 的工作中也得到了论证，他们借鉴了 Metropolis 算法与模拟退火算法中的一些思想，用来解决图像重构中的问题。

8.5.3 模拟退火算法的一个应用

模拟退火算法的一个应用是销售员旅行问题 (Traveling salesman problem, TSP)，从中可以看出局部变动思想的重要性，以及如何计算能量函数。在这个问题中，配置 ω 定义为一个包含 n 个节点的旅行路线，用 $(1, 2, \dots, n)$ 表示。

组合优化的关键在于如何选择有效的请求转移过程。一个简单的请求转移过程为随机的选择两个节点 $1 \leq i \leq j \leq n$ ，并逆转 (reverse) 两者之间的路线。即如果有

$$\omega = (a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{j-1}, a_j, a_{j+1}, \dots, a_n)$$

逆转节点 i, j 之间的路线，可得

$$\omega = (a_1, \dots, a_{i-1}, a_j, a_{j-1}, \dots, a_{i+1}, a_i, a_{j+1}, \dots, a_n)$$

在逆转两节点之后，旅行距离（能量）的变化为

$$\Delta E = (|a_{i-1} - a_j| + |a_i - a_{j+1}|) - (|a_{i-1} - a_i| + |a_j - a_{j+1}|).$$

图8.2所示为模拟退火算法应用于 TSP 的实现仿真，其中 TSP 由单位平面内随机放置的 100 个节点构成。

8.6 尾声

90 年代中期以来，MCMC 方法几乎被用于自然与社会科学的每一领域，同时也吸引了数学家、尤其是统计学家的注意。MCMC 方法被创造性用于计算机科学、生物学（尤其是基因学）、化学、物理学、心理学、神经科学、以及经济学、政治学、社会学等几乎所有人们可以想象到的领域中，甚至在纯数学中——例如，从对称群中采样——MCMC 方法一样出色。

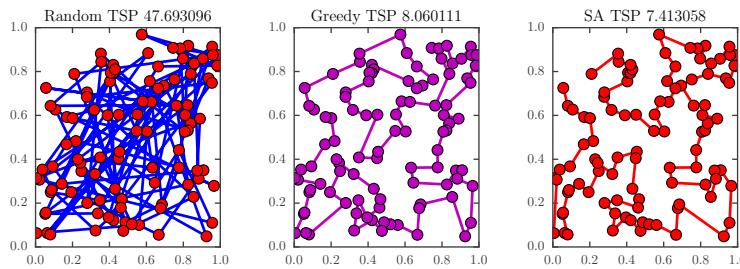


图 8.2 模拟退火算法应用于由 100 个随机节点构成的 TSP 问题。左边的图像为初始条件 ($T = 4.0$) 下的配置, 总距离为 47.69。右边的配置为总距离接近全局最小值 7.40 ($T = 0.002$)。

随着应用领域的扩张, 人们也在不断的试图理解 MCMC 的收敛性。Tierney 于 1994 年发表的文献 *Markov Chains for Exploring Posterior Distributions*, 在统计学社区中对于理解 MCMC 方法的收敛性有一定的影响力。在该文中, 他对 MCMC 方法, 也包括吉布斯采样、Metropolis–Hastings 算法以及其他一些混合方法 (混合方法中某些步骤用了吉布斯采样, 某些步骤使用了 Metropolis–Hastings 算法), 提供了鲁棒的理论框架。Tierney 讨论了算法中方方面面的问题, 包括 Metropolis–Hastings 算法中采用不同类型转移函数对算法的影响, 他同时提供马尔科夫链遍历性质的一些较强的结论, 可以帮助算法使用者提前确定马尔可夫链的运行长度。几乎是在同时, Rosenthal 等人提出了一种方法 (常被称作最小化法) 研究了为了达到满意的性能所需运行的最少迭代次数, 给出了明确、先验的对数界限。

在统计学之外的应用中 (这些应用往往占大多数), 对于收敛性的讨论就较为少见了。正如 Diaconis 提到:

我相信, 随便从科学中选定任一领域, 无论从硬科学还是到社会学, 都会发现针对这一领域、专门定制的 MCMC 算法。但是, 我注意这些应用都基本上没有附带任何形式的、有用的有关运行时间分析。

8.7 结论

我们来归纳一下马尔可夫链蒙特卡洛方法 (MCMC) 的演变过程。它来自新墨西哥的沙漠中, 因一群人、一个问题、一台机器的幸运交汇而诞生。它随着计算机的发展不断成长, 对数学和自然科学产生了巨大的影响。虽然对于它为什么能如此有效、以及预测需要多长时间才可收敛仍存在疑问, 但即便如此, 它能工作而且做的比其他 (确定性) 算法都好。

在观测到模拟退火算法（SA）可有效的求解旅行销售员问题之后，可能正如 V. Čerňy 所说

在先前的例子（TSP）中，我们的算法出人意料的有效。这或许可以认为，由于我们的算法模拟了大自然寻求复杂系统均衡态的方法，而大自然总能把自己的工作做的有效出色。

第9章 上机指南

刘本源 *byliu@fmmu.edu.cn*

学时 上机

2 0

9.1 目的

讲解 Python 编程关键 (python in a nutshell), 以及边界元生成库 meshpy 的安装和使用。

重点: Python 编程语言

难点: Python 编程及网格划分

参 考 文 献

- [1] Bondeson A, Rylander T, Ingelström P. Computational Electromagnetics [M]. Springer, 2005.
- [2] 倪光正, 杨仕友, 邱捷. 工程电磁场数值计算 [M]. 2nd ed. 机械工业出版社, 2012.
- [3] 颜威利, 徐桂芝. 生物医学电磁场数值分析 [M]. 2nd ed. 机械工业出版社, 2007.
- [4] 金建铭. 电磁场有限元方法 [M]. 2nd ed. 西安电子科技大学出版社, 2001.
- [5] Holder D S. Electrical Impedance Tomography : Methods, History and Applications [M]. Institute of Physics, 2005.
- [6] Guru B S, Hiziroglu H R. 电磁场与电磁波 [M]. 2nd ed. 机械工业出版社, 2002.
- [7] 赵凯华, 陈熙谋. 电磁学 [M]. 高等教育出版社, 2006.
- [8] Rylander T, Ingelström P, Bondeson A. Computational Electromagnetics [M]. Springer, 2013.
- [9] Mathews J H, Fink K D. Numerical Methods Using MATLAB [M/OL]. Prentice Hall inc., 2004. <http://mathfaculty.fullerton.edu/mathews/>.
- [10] Holodorodko P. Numerical Integration [EB/OL]. <http://www.holoborodko.com/pavel/numerical-methods/numerical-integration/>.
- [11] Pan J. Jacobi and Gauss–Seidel Iteration Methods [EB/OL]. http://www.math.ecnu.edu.cn/~jypan/Teaching/ParaComp/lect_Jacobi_GS.pdf.
- [12] Pasciak J E. Gauss–Seidel [EB/OL]. <http://www.math.tamu.edu/~pasciak/classes/639/1310.pdf>.
- [13] Shewchuk J R. Delaunay Refinement Algorithms for Triangular Mesh Generation [J/OL]. Computational Geometry: Theory and Applications. 2002, 22 (1–3): 21–74. <http://www.cs.cmu.edu/~quake/triangle.html>.
- [14] LaForce T. PE281 Boundary Element Method Course Notes [EB]. 2006. <http://web.stanford.edu/class/energy281/BoundaryElementMethod.pdf>.

- [15] peghoty. 牛顿法与拟牛顿法学习笔记 [EB/OL]. <http://blog.csdn.net/itplus/article/details/21896453>.
- [16] Nocedal J. Updating quasi-Newton matrices with limited storage [J]. *Math. Comp.* 1980, 35: 773–782.
- [17] EE236C. Quasi-Newton methods [EB/OL]. <http://www.seas.ucla.edu/~vandenbe/236C/lectures/qnewton.pdf>.
- [18] Domke J, Wang L. Discovery Proj 2 – Unconstrained Optimization [EB/OL]. 2012. <http://phd.gccis.rit.edu/discovery/proj2/>.
- [19] Griffin C. Numerical Optimization: Penn State Math 555 Lecture Notes [EB/OL]. 2012. <http://www.personal.psu.edu/cxg286/Math555.pdf>.
- [20] Line Search Methods [EB/OL]. <https://www.math.washington.edu/~burke/crs/408/notes/nlp/line.pdf>.
- [21] Gómez-Laberge C, Adler A. Direct EIT Jacobian calculations for conductivity change and electrode movement [J]. *Physiological Measurement*. 2008, 29 (6): 89–S99.
- [22] Helfrich-Schkarbanenko A, Kreutzmann T. MATLAB software for 2D Electrical Impedance Tomography [EB/OL]. 2013. <http://www.math.kit.edu/iag1>.