大作业 5 Van der Pauw 法测量电阻率

张钰坤
2000011314
(C 语言实现)

2022 年 5 月 15 日

目录

1 解题思路

本题要求解如下的定解问题：

$$\text{方程：} \nabla^2\phi = 0, (x,y) \in \Omega \tag{1}$$

$$\text{定解条件：} \boldsymbol{n} \cdot \nabla\phi = 0, (x,y) \in \partial\Omega \tag{2}$$

$$\sigma d \int_C \boldsymbol{n} \cdot \nabla\phi dl = \pm I, \forall C \subseteq \Omega \tag{3}$$

1.1 无量纲化

选定

$$\tilde{\phi} = \phi/(1V)$$
$$\tilde{I} = \frac{I}{\sigma d(1V)}$$
$$\tilde{R} = \frac{\tilde{U}}{\tilde{I}}$$

而所有长度选定某个单位，比如说厘米。
于是定解问题化为

$$\text{方程：} \nabla^2\tilde{\phi} = 0, (\tilde{x},\tilde{y}) \in \Omega \tag{4}$$

$$\text{定解条件：} \tilde{\boldsymbol{n}} \cdot \tilde{\nabla}\tilde{\phi} = 0, (\tilde{x},\tilde{y}) \in \partial\Omega \tag{5}$$

$$\int_C \tilde{\boldsymbol{n}} \cdot \tilde{\nabla}\tilde{\phi} d\tilde{l} = \pm\tilde{I}, \forall C \subseteq \Omega \tag{6}$$

Van der Pauw 方程化为

$$\exp(-\pi\tilde{R}_{AB,DC}) + \exp(-\pi\tilde{R}_{AD,BC}) = 1 \tag{7}$$

1.2 简化条件

不失一般性，我们将物理量无量纲化的上标省略：

$$\text{方程：} \nabla^2\phi = 0, (x,y) \in \Omega \tag{8}$$

$$\text{定解条件：} \boldsymbol{n} \cdot \nabla\phi = 0, (x,y) \in \partial\Omega \tag{9}$$

$$\int_C \boldsymbol{n} \cdot \nabla\phi dl = \pm I, \forall C \subseteq \Omega \tag{10}$$

注意到第三个方程是对无穷个曲线的限制条件，不具有操作性。事实上，第三个方程可以用前两个条件简化。

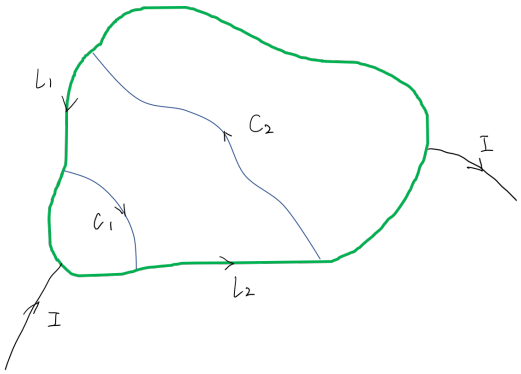

图 1: 条件简化示意图 1

如示意图 1 所示，对任意 Ω 内的曲线 C_1, C_2，设它们截出的边界为 l_1, l_2，且 l_1, l_2 不包含源点。那么有

$$\oint \boldsymbol{n} \cdot \nabla \phi dl = \begin{cases} \iint_\Omega \nabla^2 \phi dS = 0 \\ (\int_{C_1^-} + \int_{C_2^+} + \int_{l_1} + \int_{l_2}) \boldsymbol{n} \cdot \nabla \phi dl = (\int_{C_1^-} + \int_{C_2^+}) \boldsymbol{n} \cdot \nabla \phi dl \end{cases}$$

其中第一行的等号用到了方程，第二行等号用到了边界条件。

进而得到

$$\int_{C_1} \boldsymbol{n} \cdot \nabla \phi dl = \int_{C_2} \boldsymbol{n} \cdot \nabla \phi dl$$

根据上面的性质，方程三 $\int_C \boldsymbol{n} \cdot \nabla \phi dl = \pm I, \forall C \subseteq \Omega$ 可以简化为

$$\int_{C_{in}} \boldsymbol{n} \cdot \nabla \phi dl = I, \int_{C_{out}} \boldsymbol{n} \cdot \nabla \phi dl = -I$$

其中 C_{in}, C_{out} 是电极附近的一个选定的曲线，如示意图 2 所示。

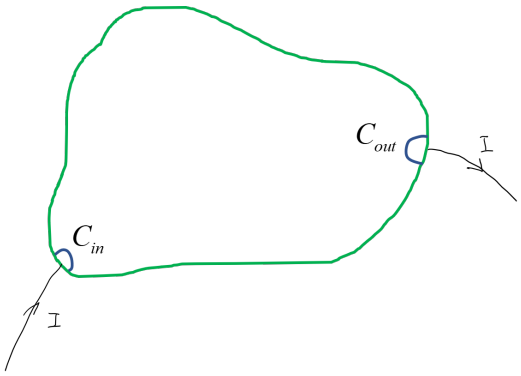

图 2: 条件简化示意图 2

1.3 方形网格离散化

根据上述讨论，我们确定了数值求解的定解问题：

$$\text{方程：} \nabla^2 \phi = 0, (x, y) \in \Omega \tag{11}$$

$$\text{定解条件：} \boldsymbol{n} \cdot \nabla \phi = 0, (x, y) \in \partial\Omega \tag{12}$$

$$\int_{C_{in}} \boldsymbol{n} \cdot \nabla \phi dl = I, \int_{C_{out}} \boldsymbol{n} \cdot \nabla \phi dl = -I \tag{13}$$

本题欲采用有限差分的方法求解，并采用方形网格对求解区域离散化。

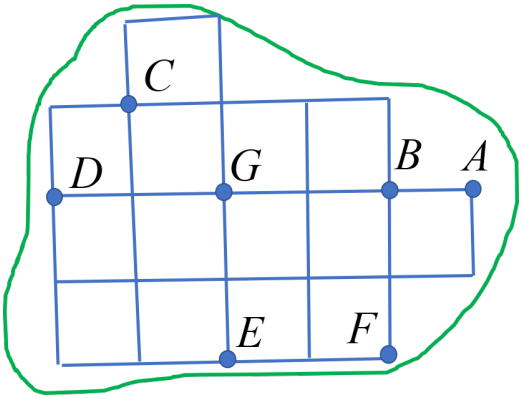

图 3: 方形网格离散化区域

如上图所示，经过离散化后，我们实际用网格区域代替了原来的区域。我们规定，当网格点上下左右都有格点的时候，认为该格点是内点；当网格上下左右有格点缺失的时候，认为该格点是边界点。例如，上图中 ADEF 是边界点，BCG 是内点。注意这里我们认为 BC 也是内点。看似当我们用网格区域代替了原来的区域时，BC 处在了边界上，但是由于它们常处在凹陷中，离真实物理边界较远，故可以把它们当作内点处理。

离散化之后给每个格点横纵指标 (i,j)，将每个格点的电势设为 ϕ_{ij}，进而我们可以分别对内点和边界点满足的方程应用有限差分。

1.3.1 内点

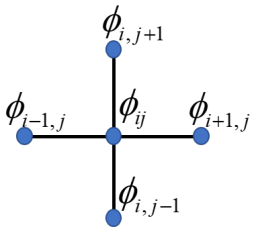

图 4: 内点

如图所示，内点满足拉普拉斯方程离散化为

$$\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i,j+1} - 4\phi_{i,j} = 0$$

1.3.2 边界点

1.3.2.1 无源 情况 1：格点上下左右只有一个点缺失，如图 3中的 DE，此时格点在网格的一条边上。

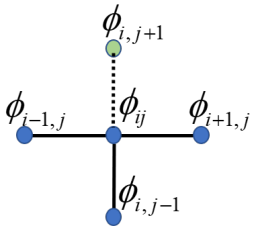

图 5: 无源边界点 1

由于边界点满足 $\boldsymbol{n} \cdot \nabla\phi = 0$，即电势法向导数为 0。我们可以关于直边界 $\phi_{i-1,j}\phi_{i,j}\phi_{i+1,j}$ 把内点 $\phi_{i,j-1}$ 对称出去，得到虚拟点 $\phi_{i,j+1}$，如图所示，再由 (i,j) 点法向导数为 0，得到 $\phi_{i,j+1} - \phi_{i,j-1} = 0$。由于补全了 (i,j)

点周围的格点，可以由拉普拉斯方程得到 $\phi_{i-1,j}+\phi_{i+1,j}+\phi_{i,j-1}+\phi_{i,j+1}-4\phi_{i,j}=0$，二者联立，就得到这种边界点的满足的离散化方程：

$$\phi_{i-1,j}+\phi_{i+1,j}+2\phi_{i,j-1}-4\phi_{i,j}=0$$

情况 2：格点上下左右有 2 个点缺失，如图 3中的 AF，此时格点在网格的某个角上。

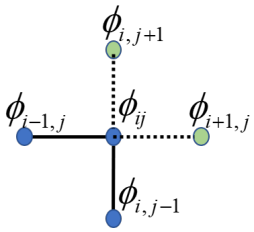

图 6: 无源边界点 2

由于边界点满足 $\boldsymbol{n}\cdot\nabla\phi=0$，即电势法向导数为 0。我们可以分别关于直边界 $\phi_{i-1,j}\phi_{i,j}$ 和 $\phi_{i,j}\phi_{i,j-1}$ 把内点 $\phi_{i-1,j}$ 和 $\phi_{i,j-1}$ 对称出去，得到虚拟点 $\phi_{i+1,j}$ 和 $\phi_{i,j+1}$，如图所示，再由 (i,j) 点法向导数为 0，得到 $\phi_{i,j+1}-\phi_{i,j-1}=0$ 和 _i+1,j- _i-1,j=0。由于补全了 (i,j) 点周围的格点，可以由拉普拉斯方程得到 $\phi_{i-1,j}+\phi_{i+1,j}+\phi_{i,j-1}+\phi_{i,j+1}-4\phi_{i,j}=0$，二者联立，就得到这种边界点的满足的离散化方程：

$$2\phi_{i-1,j}+2\phi_{i,j-1}-4\phi_{i,j}=0$$

我们不妨在取格点的时候保证，**边界格点只可能有 1 或 2 个上下左右格点缺失，即：边界格点要么在一条边上，要么在一个角上**。进而，所有边界点都可以分成上面两种情况（只差旋转一个角度，分析是类似的）。

1.3.2.2　有源　根据方程 3：$\int_{C_{in}}\boldsymbol{n}\cdot\nabla\phi dl=I$，可以得出 $\boldsymbol{n}\cdot\nabla\phi=\boldsymbol{j}$，即电势的法向导数等于电流密度。

情况 1：格点上下左右只有一个点缺失，此时电流加在网格的一条边上，如下图。

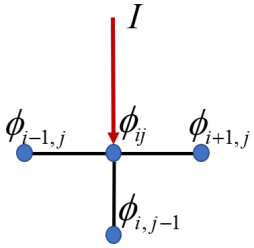

图 7: 有源边界点 1

设格点边长为 h，我们不妨设 $\phi_{i,j}\phi_{i,j-1}$ 中点的电流密度为 I/h，那么有

$$\frac{\phi_{i,j}-\phi_{i,j-1}}{h}=\frac{I}{h}$$

于是得到离散化方程

$$\phi_{i,j}-\phi_{i,j-1}=I$$

情况 2：格点上下左右只有 2 个点缺失，此时电流加在网格的一个角上，如下图。

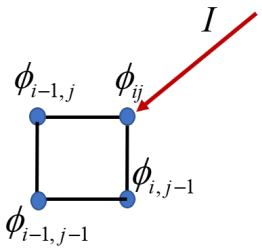

图 8: 有源边界点 2

设格点边长为 h，我们不妨设 $\phi_{i,j}\phi_{i-1,j-1}$ 中点的电流密度为 I/h，那么有

$$\frac{\phi_{i,j}-\phi_{i-1,j-1}}{\sqrt{2}h}=\frac{I}{\sqrt{2}h}$$

于是得到离散化方程

$$\phi_{i,j}-\phi_{i-1,j-1}=I$$

1.4 SOR 法/Gauss-Seidel 法求解线性方程组

根据前一节的分析，我们用方形网格离散化待求解区域，设出每个格点上的电势 ϕ_{ij}。经过讨论，这些格点分为三大类：内点，无源边界点，有源边界点。每种情况都可以给出以这些格点为中心的线性方程。因此，有多少个格点，就有多少个未知数，多少个线性方程，进而可以通过解线性方程组的方式，求出所有格点的电势值。

将 ϕ_{ij} 按一定顺序排列成一个向量 $\boldsymbol{x}$，线性方程组可以写为

$$\boldsymbol{Ax}=\boldsymbol{b}$$

注意到 A 是一个大型极端稀疏矩阵。因为，想要求解的电势十分精确的反应真实物理情景，有限差分只有在格子取得很小的情况下才能作为导数的近似，这会使得未知数 ϕ_{ij} 数量庞大。但是根据前节的讨论，每个格点对应的方程，之和自身和周围的 4 项相关，说明矩阵 A 中每行只有至多 5 个元素非 0。故，A 大型且稀疏。

于是我们采用 SOR 法/Gauss-Seidel 法求解。这会遇到两个问题：

问题 1：SOR 法/Gauss-Seidel 法需要矩阵对角元全部非 0。

问题 2：由于 A 规模巨大，难以储存；并且即便储存了，在迭代过程中也会遇到 0 和数的乘法这种不必要的运算。（下面的估算说明存储 A 的不值当：假设有一万个格点，那么矩阵 A 是 10000*10000 的，如果用二维 double 数组存储，共占据 $8\times10^8B=800MB$ 内存！并且这个矩阵元素中大部分都是 0，有多大部分呢？每行 10000 个元素最多有 5 个不是 0，那么有 9995/10000=99.95% 空间浪费掉了！）

1.4.1 解决 问题 1 的办法

解决 **问题 1** 的办法是：选定 ϕ_{ij} 在 $\boldsymbol{x}$ 中的排列次序，那么让 ϕ_{ij} 在 $\boldsymbol{x}$ 中由上到下的排列次序，与 (i,j) 满足的线性方程在 $\boldsymbol{A}$ 中由上到下的排列次序相同。由于 (i,j) 满足的线性方程一定含有 ϕ_{ij} 项，那么对角元一定非零。

比如说，$\phi_{1,2}$ 在 $\boldsymbol{x}$ 中是第 5 个，那么 A 的第 5 行就去记录以 (1,2) 点为中心的方程系数，以 (1,2) 是内点为例，去记录方程 $\phi_{0,2}+\phi_{2,2}+\phi_{1,1}+\phi_{1,3}-4\phi_{1,2}=0$ 的系数。

1.4.2 解决 问题 2 的办法

解决 **问题 2** 的办法是：不储存矩阵，而把每一步迭代写成函数，这样既不用储存矩阵，又不用计算许多 0 和数的乘法。

参考 SOR 算法 [1],

计

$$A = \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha_n} \end{pmatrix}$$

SOR 算法的每次迭代可以写为

$$\begin{cases} x_i^{(k+1)} = x_i^{(k)} + \Delta x_i, \\ \Delta x_i = \omega(b_i - \boldsymbol{\alpha}_i^\top \boldsymbol{x})/A(i;i), \\ i = 1, 2, \cdots, n; k = 1, 2, \cdots; \omega\text{为松弛因子} \end{cases}$$

为简便起见可以让 A 对角元都为-4，然后可以写一个函数计算 $\boldsymbol{\alpha}_i^\top \boldsymbol{x}$。

Algorithm 1 AmX 函数，计算 $\boldsymbol{\alpha}_i^\top \boldsymbol{x}$

Input: 行数 m，待求解向量 x;

Output: 内积 $\boldsymbol{\alpha}_m^\top \boldsymbol{x}$;

通过 m 的值确定这一行系数对应的方程中心点 (i,j)。（因为要选定一定顺序排列 $\phi_{i,j}$，而且系数矩阵 A 每行的排列与这个顺序相同，见“解决问题 1 的办法”）;
if (i,j) 是有源边界点（通电流的点） **then**
    按照有源边界点方程，用这些系数 * 对应 x 中的分量再相加，返回这个值。
else
    (i,j) 是无源边界点或内点；
    由于无源边界点和内点上的方程本质都是拉普拉斯方程（见“方形网格离散化”），因而二者可以统一，并通过一些判断条件，把想要求的值算出来，下面实现这一点。
    设一个变量 s 临时储存待求内积;s=0;
    因为 (i,j) 项对应的系数一定是-4，所以 s+=$-4\phi_{i,j}$
    if (i+1,j) 在界外 **then**
        (i-1,j) 一定在界内，并且由于对称性，(i-1,j) 的系数一定是 2，故 s+=$2\phi_{i-1,j}$
    end if
    if (i-1,j) 在界外 **then**
        (i+1,j) 一定在界内，并且由于对称性，(i+1,j) 的系数一定是 2，故 s+=$2\phi_{i+1,j}$
    end if
    if (i,j+1) 在界外 **then**
        (i,j-1) 一定在界内，并且由于对称性，(i,j-1) 的系数一定是 2，故 s+=$2\phi_{i,j-1}$
    end if
    if (i,j-1) 在界外 **then**
        (i,j+1) 一定在界内，并且由于对称性，(i,j+1) 的系数一定是 2，故 s+=$2\phi_{i,j+1}$
    end if
    if (i+1,j) 不在界外且 $\phi_{i+1,j}$ 没有在之前计入过 s **then**
        (i-1,j) 的系数一定是 1，故 s+=$\phi_{i-1,j}$
    end if
    if (i-1,j) 不在界外且 $\phi_{i-1,j}$ 没有在之前计入过 s **then**
        (i+1,j) 的系数一定是 1，故 s+=$\phi_{i+1,j}$
    end if
    if (i,j+1) 不在界外且 $\phi_{i,j+1}$ 没有在之前计入过 s **then**
        (i,j-1) 的系数一定是 1，故 s+=$\phi_{i,j-1}$
    end if
    if (i,j-1) 不在界外且 $\phi_{i,j-1}$ 没有在之前计入过 s **then**
        (i,j+1) 的系数一定是 1，故 s+=$\phi_{i,j+1}$
    end if
    return s
end if

值得注意的是，对于不同形状的导体，不同的电极接入位置，电极方程都略有不同，但好处是内点和无源边界点的处理是兼容的，可以参考源代码中 (double) AmX(int, vector*) 函数。

1.5 遍历格点并寄存位置

在上节算法的实现中，我们认为 $\phi_{i,j}$ 通过一定顺序排列好了，并且我们需要经常通过 $\phi_{i,j}$ 的排列序号得到位置坐标 (i,j)，和通过位置坐标 (i,j) 确定排列序号。这说明我们需要记录这种映射关系。另外，对于不同形状

的导体，如何构造格点是一个问题。我们将在本节中解决这两个问题。

1.5.1 第一问方形导体

方形导体的格子可以直接划分，举个简单的例子，取格子边长 0.5，划分 3*2 的长方形，那么将划出 (7*5=35) 个格子，选定如下的排列方式。

$$\begin{array}{ccccccc} 28 & 29 & 30 & 31 & 32 & 33 & 34 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 \\ 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array}$$

那么容易定出格点位置和序号的函数关系。

一般的，取格子边长 h，划分 3*2 的长方形，可以计算出 NX=3/h,NY=2/h。于是划分出 (NX+1)*(NY+1) 个格子，选定和例子一样的排列方式。

$$\begin{array}{ccc} NY*(NX+1) & \cdots & NY*(NX+1)+NX \\ \vdots & & \vdots \\ 0 & \cdots & NX \end{array}$$

可以计算出位置 (i,j)(i=0,1,...,NX;j=0,1,...,NY) 与序号 l(l=0,1,...,(NX+1)*NY+NX) 的函数关系：

$$\begin{aligned} l &= i + j(NX+1) \\ i &= l\%(NX+1) \\ j &= [l/(NX+1)] \end{aligned}$$

1.5.2 具有中心对称和轴对称的曲线边界

第二、三、四问中的导体形状都是具有中心对称和轴对称性质的，我们不妨把曲线的对称中心放在原点，只在第一象限内划分格子，进而可以对称到其余象限中。

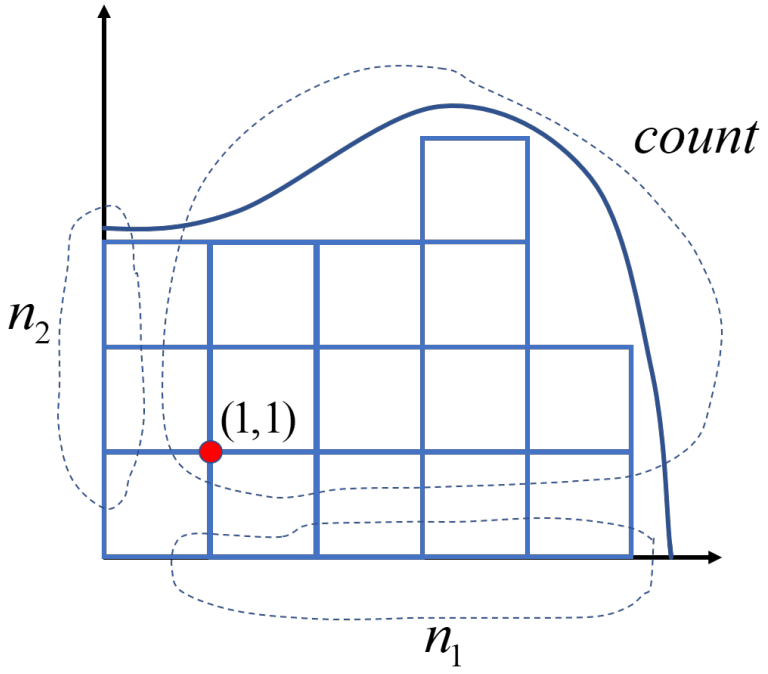

图 9: 构造格点示意图

如上图所示，我们只需要数出上图中三个虚线框中各有多少个格点 (count 个，n1 个，n2 个)，就可以得到总格点数 4*count+2*(n1+n2)+1。

怎样去数格点呢？我们通过逐行搜索的方式遍历第一象限内所有格点。起点设在 (1,1)，每次向右踏一步，直到走到了曲线外，我们就回到 i=1 但是纵指标增加了 1，在往上一行继续搜索，直到一整行都没有点在曲线内为止。

在这个搜索过程中我们可以得到许多离散后的网格信息，不仅限于图中的 count,n1,n2，我们还可以得到横向最大格数，纵向最大格数，曲线与 y=x 交点的近似格点位置（第四问），曲边中点近似格点位置（第二问）等等，详情见每一问的源代码。

我们通过遍历确定了格点，接下来就可以按照一定顺序给格点编号，并用数组记录三个数（序号 l，横指标 i，纵指标 j）的关系。

这三个数组分别是：

ltoi[], 一维数组，记录每个 l 对应的 i；

ltoj[]，一维数组，记录每个 l 对应的 j；

ijtol[][]，二维数组，记录 (i,j) 位置的序号 l，相当于是把整个格点图画了下来，由于边界不规则而二维数组是标准的矩形，所以一些数组元素表示导体外的点，可以置为-1，另外由于 i,j 可能取负数，但数组元素的序号非负，因此可以做适当的平移加以储存，这里不过多赘述，详情见源代码。

我们采用如下顺序编号：

1、原点；

2、x 轴：每一次先后标记正负半轴对称位置的两个值，绝对值从小到大；

3、y 轴：每一次先后标记正负半轴对称位置的两个值，绝对值从小到大；

4、象限内的点：按照数格点时的遍历方式，在第一象限遍历，每遍历到一个点，根据对称性依次把四个象限对称位置的点全部标记。

这样我们就完成了三个数组的初始化，进而我们可以随时取用 l 与 (i,j) 的对应信息。

我们可以把 ijtol[][] 打印出来，欣赏一下格点化的成果。

这里为了演示方便，步长取得比较大。

第二问，步长 0.1

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	290	43	289	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	286	41	285	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	282	278	39	277	281	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	274	270	266	37	265	269	273	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	262	258	254	250	35	249	253	257	261	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	246	242	238	234	33	233	237	241	245	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	230	226	222	218	214	31	213	217	221	225	229	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	210	206	202	198	194	29	193	197	201	205	209	-1	-1	-1	-1	-1
-1	-1	-1	-1	190	186	182	178	174	170	27	169	173	177	181	185	189	-1	-1	-1	-1
-1	-1	-1	166	162	158	154	150	146	142	25	141	145	149	153	157	161	165	-1	-1	-1
-1	-1	-1	138	134	130	126	122	118	114	23	113	117	121	125	129	133	137	-1	-1	-1
-1	-1	110	106	102	98	94	90	86	82	21	81	85	89	93	97	101	105	109	-1	-1
-1	78	74	70	66	62	58	54	50	46	19	45	49	53	57	61	65	69	73	77	-1
-1	18	16	14	12	10	8	6	4	2	0	1	3	5	7	9	11	13	15	17	-1
-1	79	75	71	67	63	59	55	51	47	20	48	52	56	60	64	68	72	76	80	-1
-1	-1	111	107	103	99	95	91	87	83	22	84	88	92	96	100	104	108	112	-1	-1
-1	-1	-1	139	135	131	127	123	119	115	24	116	120	124	128	132	136	140	-1	-1	-1
-1	-1	-1	167	163	159	155	151	147	143	26	144	148	152	156	160	164	168	-1	-1	-1
-1	-1	-1	-1	191	187	183	179	175	171	28	172	176	180	184	188	192	-1	-1	-1	-1
-1	-1	-1	-1	-1	211	207	203	199	195	30	196	200	204	208	212	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	231	227	223	219	215	32	216	220	224	228	232	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	247	243	239	235	34	236	240	244	248	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	263	259	255	251	36	252	256	260	264	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	275	271	267	38	268	272	276	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	283	279	40	280	284	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	287	42	288	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	291	44	292	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

图 10: 第二问格点划分示意

第三问，步长 0.1

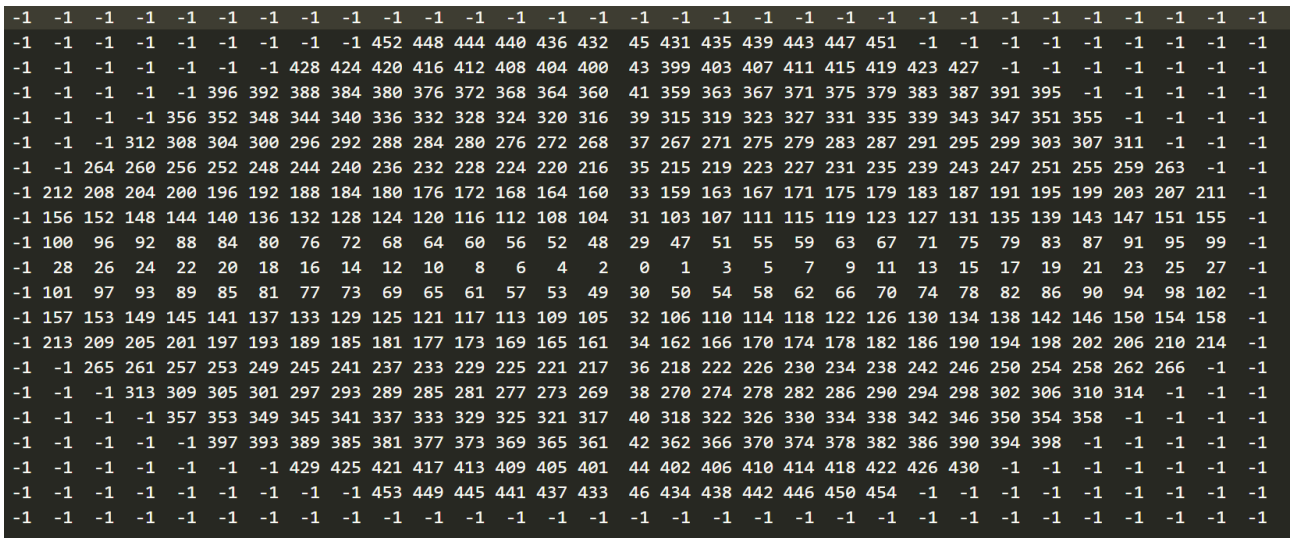

图 11: 第三问格点划分示意

第四问，步长 0.3

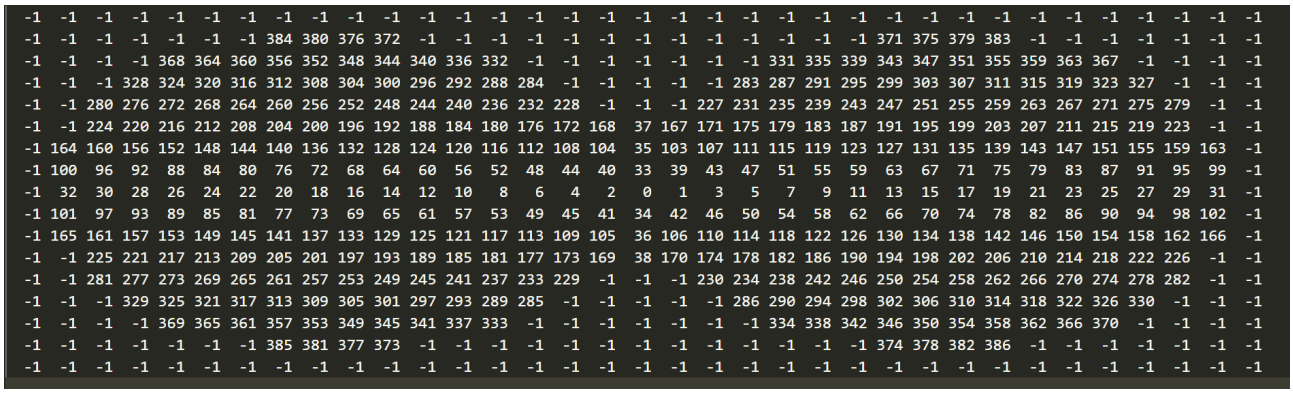

图 12: 第四问格点划分示意

2 源代码

为方便计算取 I=1

所有 C 源文件在 code 文件夹，输出 txt 文本 (打印出绘制等高线图的原始数据，即打印出 ijtol[][] 矩阵，其中为方便 mathematica 作图，-1 用复数-1+I 代替) 在 data 文件夹。对应关系如下：

	code	data	对应下节中的图
第一问	”q1.c”	”q1_NX=300.txt”	Q1(1)
	”q1_inverse.c”	”q1_NX=300_inverse.txt”	Q1(2)
第二问	”q2.c”	”q2_H=0.01.txt”	Q2(1)
	”q2_inverse.c”	”q2_H=0.01_inverse.txt”	Q2(2)
第三问	”q3.c”	”q3_H=0.01.txt”	Q3(1)
	”q3_inverse.c”	”q3_H=0.01_inverse.txt”	Q3(2)
第四问	”q4.c”	”q4.txt”	Q4(1)
	”q4_inverse.c”	”q4_inverse.txt”	Q4(2)

注意：在 C 编译环境可以运行上述代码，只是每个代码中读写的 txt 文件需要重新更改路径!!

另外，有关松弛因子的选取，这里只是数值估算出的。

具体方式是，选取一个大一点的步长做测试，这时矩阵维数较小，可以快速出结果。由于改变步长只是对矩阵进行了放缩，可以认为矩阵的性质大致不变，因而可以对小矩阵测试不同的松弛因子，输出迭代次数，然后选取次数最小的松弛因子作为正式的松弛因子。代码中所示松弛因子的就是这种数值计算的近似估计，效果是显著的，运行时间可以缩短几倍。

3 结果展示

数据代码和对应的图关系见上节对应表格，这里用来绘制图片的 mathematica 文件”plot.nb” 也在附件中。

3.1 第一问

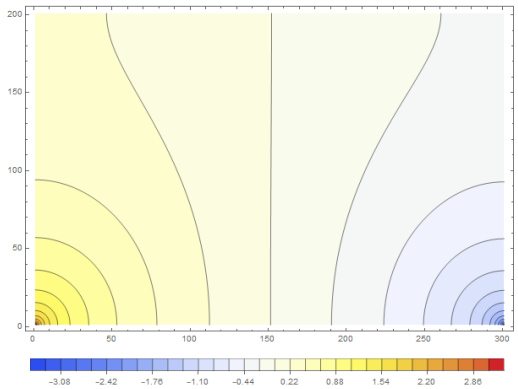

图 13: Q1(1)

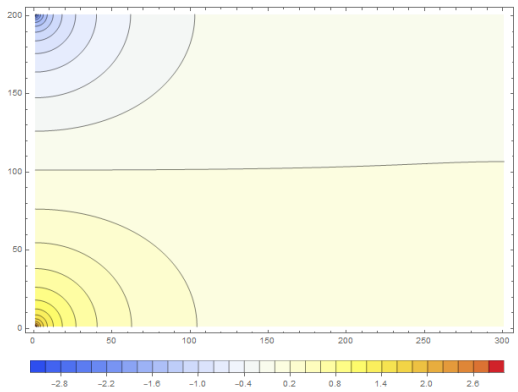

图 14: Q1(2)

$$R_1 = (0.248355 - (-0.242514))/1$$
$$R_2 = (0.017833 - (-0.014927))/1$$
$$E^{-\pi R_1} + E^{-\pi R_2} = 1.11613$$

3.2 第二问

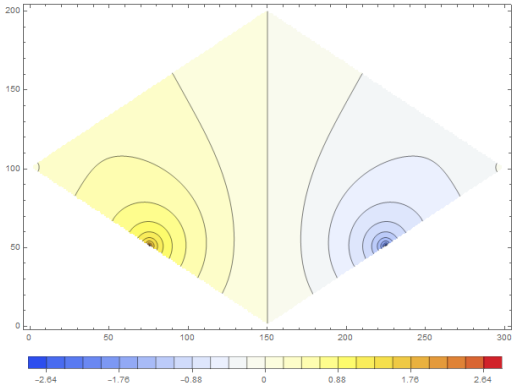

图 15: Q2(1)

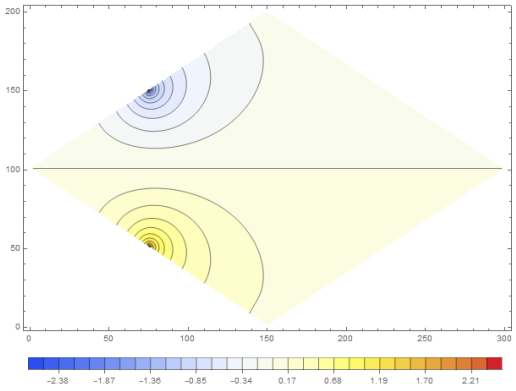

图 16: Q2(2)

$$R_1 = (0.293313 - (-0.293313))/1$$
$$R_2 = (-(-0.033495) + 0.033495)/1$$
$$E^{-\pi R_1} + E^{-\pi R_2} = 0.968566$$

3.3 第三问

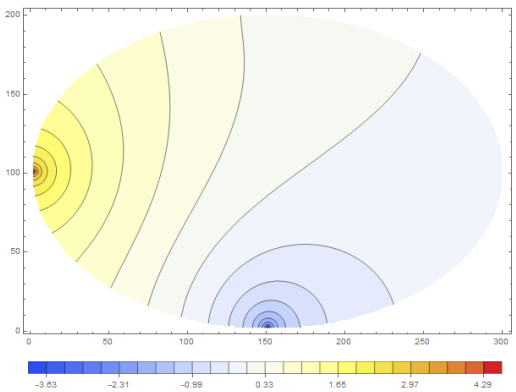

图 17: Q3(1)

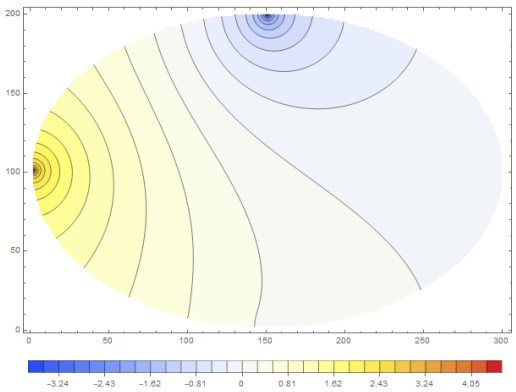

图 18: Q3(2)

$$R_1 = (0.129034 - (-0.048891))/1$$
$$R_2 = (0.129034 - (-0.048891))/1$$
$$E^{-\pi R_1} + E^{-\pi R_2} = 1.1436$$

3.4 第四问

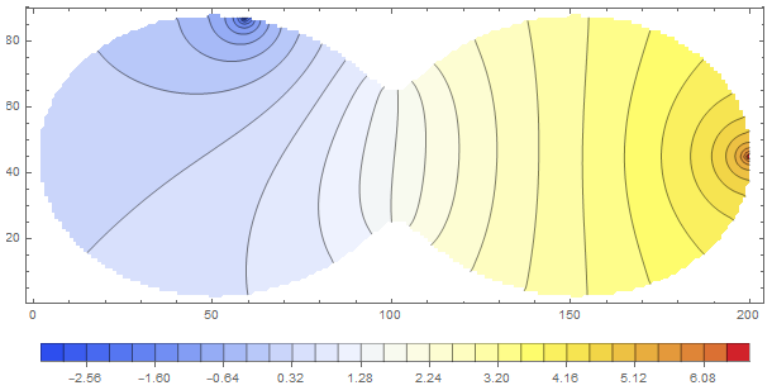

图 19: Q4(1)

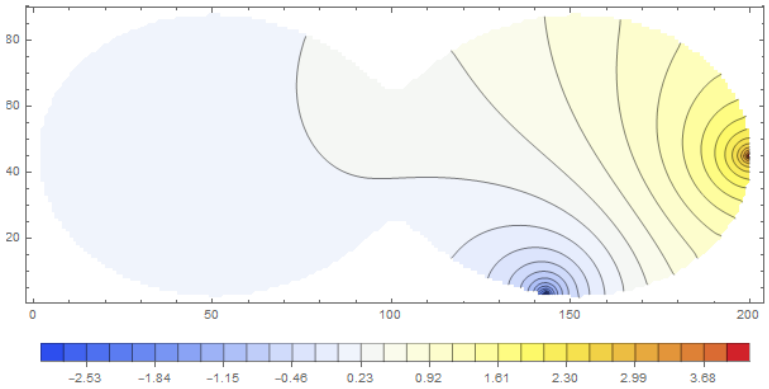

图 20: Q4(2)

$$R_1 = (2.985171 - 1.656232)/1$$
$$R_2 = (0.215642 - 0.197188)/1$$
$$E^{-\pi R_1} + E^{-\pi R_2} = 0.959049$$

参考文献

[1] 李庆扬，王能超，易大义：数值分析，194 页，2008 年 12 月第五版