基于 FEM 与 FDM 求解各向异性 Poisson 方程的 Line Gauss-Seidel 迭代法

豆旭桉 1700013022 薛烨诚 1800010705 谢中林 1700016908
金则宇 1700017850 伍天一 1700010700 孔鼎问 1800010635

January 2020

1 引言

1.1 问题描述

本次作业主要关注定义在 $\Omega = (0,1) \times (0,1)$ 上的 Possion 方程

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} - \varepsilon \frac{\partial^2}{\partial y^2} = f(x,y), (x,y) \in \Omega \\ u(x,y) = 0, (x,y) \in \partial\Omega \end{cases} \tag{1}$$

其中取 $0 < \varepsilon < 1$,$\varepsilon \ll 1$.

要求对 N $= 2^5, 2^6, 2^7, 2^8$;$\varepsilon = 1, 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$, 对于定向与非定向网格，分别取 $\omega = 0, \pi/6, \pi/4$ 作为计算区域，设计方法进行计算，终止条件为 $\frac{\|r_k\|_2}{\|r_0\|_2} \leq 10^{-6}$.

可选择的方法有五点差分法和有限元方法，并结合 Line Gauss-Seidel 迭代。对于 $\omega = \pi/4$ 和 $\pi/6$，需要自行探讨设计有效的迭代格式。

要求列出表格比较数值解和真实解的误差，以及不同情况下所需的迭代次数。

1.2 算法概述

为解决上述问题我们共引入了四种算法,分别为 FEM、FDM30、FDM45 和 FDM 旋转算子法。第一种利用了有限元方法，后三种利用差分方法。

FEM 算法为有限元方法与线性高斯的结合，先计算得到刚度矩阵然后用线性高斯迭代求解，可以解决任意 ω 的问题。

FDM45 解决 $\omega = \pi/4$ 时问题的有效算法，该方法采用平行于坐标轴的均匀网格，逐层做线性高斯迭代。

FDM30 是对 $\omega = \pi/6$ 时的算法。$\omega = \pi/6$ 时，由于有理数和无理数的不可公度性，难以设计出格点落在边界上的均匀网格。为此我们选择牺牲格点落在边界上的便利，保留各向异性方向平行的网格结构，用平行于坐标轴的均匀网格覆盖住待求解区域, 保留落在正方形里面的节点，依靠泰勒展开来近似处理靠近网格的部分节点，在此基础上便可以按照线性高斯的原理对内部格点作更新。这个算法也容易推广到一般的 ω。

FDM 旋转算子法则希望通过考虑算子旋转之后的形式来求解 $\omega \neq 0$ 的问题。对于旋转后出现的交叉项，在五点差分基础上另外利用对角线位置的四个点进行离散，得到对应的分块三对角矩阵并用线性高斯迭代法求解。

1.3 小组分工

薛烨诚：计算 FEM 的刚度矩阵；撰写报告的引言部分和总结部分.

金则宇：FEM 程序、FDM45 程序，撰写报告第 2.1-2.4, 2.6 节、第 3 节部分.

伍天一：谱半径程序，算例设计，计算 FEM 刚度矩阵。

谢中林：FDM 程序、报告的第 5 节（表格除外）

孔鼎问: FEM 刚度矩阵程序与验证, 数据整理分析, 报告表格与配图

豆旭桉：FDM30 程序，报告第 2.5 节，第 4 节（表格和图6除外）.

2 FEM 算法

在本节中，我们来介绍我们引入的 FEM 算法。该算法利用有限元方法，并结合 Line Gauss-Seidel 迭代法，适用于任意旋转角度 ω 与任意 ε。我们将计算刚度矩阵、介绍算法、给出数值结果与分析。

2.1 刚度矩阵的显式形式

在本算法中，我们令网格跟随区域一起旋转，网格见图1。经过计算，我们可得刚度矩阵的显式形式如下：

$$M = \begin{pmatrix} A & B & & & \\ B^T & A & B & & \\ & \ddots & \ddots & \ddots & \\ & & B^T & A & B \\ & & & B^T & A \end{pmatrix}, \tag{2}$$

其中 A 与 B 为分块矩阵，形如：

$$A = \begin{pmatrix} \alpha & \beta & & & \\ \beta & \alpha & \beta & & \\ & \ddots & \ddots & \ddots & \\ & & \beta & \alpha & \beta \\ & & & \beta & \alpha \end{pmatrix}, \ B = \begin{pmatrix} \gamma & \delta & & & \\ & \gamma & \delta & & \\ & & \ddots & \ddots & \\ & & & \gamma & \delta \\ & & & & \gamma \end{pmatrix}, \tag{3}$$

其中

$$\begin{cases} \alpha = (2 + 2cs) + \varepsilon(2 - 2cs), \\ \beta = -(cs + c^2) - \varepsilon(s^2 - cs), \\ \gamma = -(s^2 + cs) - \varepsilon(c^2 - cs), \\ \delta = (1 - \varepsilon)cs, \\ c = \cos\omega, s = \sin\omega. \end{cases} \tag{4}$$

2.2 算法描述

注意到刚度矩阵 (2) 为分块三对角矩阵。我们考虑使用 Line Gauss-Seidel 迭代法，即分块 Gauss-Seidel 迭代法来求解。

2.3 数值结果

在本文中，我们考虑如下的数值算例：令真解为

$$\begin{aligned} u(x, y) = (y\cos\omega - x\sin\omega)(-y\cos\omega + x\sin\omega + 1) \\ (x\cos + y\sin\omega)(-x\cos\omega - y\sin\omega + 1); \end{aligned} \tag{5}$$

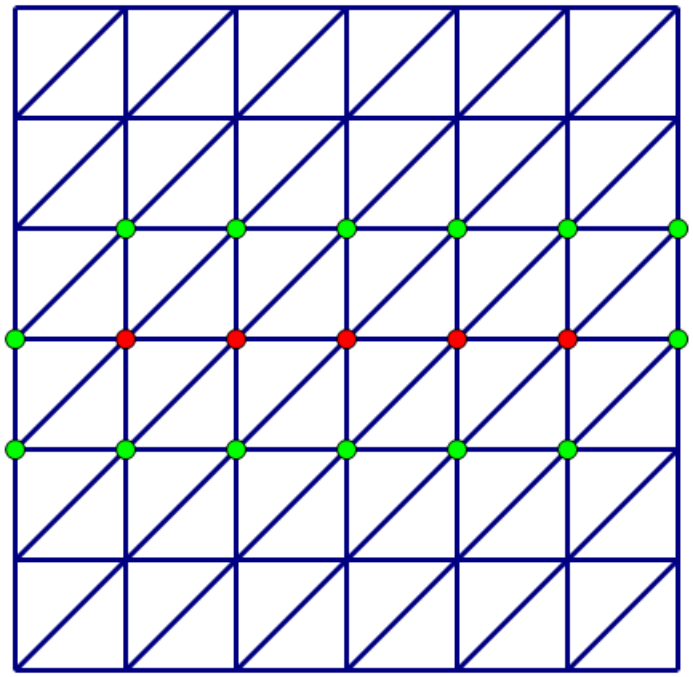

图 1: 有限元方法区域剖分情况

并对不同的 ω 与 ε 计算出相应的 $f(x,y)$. 此时区域由单位正方形绕原点旋转 ω 所得。我们对 $\mathrm{N} = 2^5, 2^6, 2^7, 2^8; \varepsilon = 1, 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ 进行求解。这里 N 指均匀网格的步长 $h = \frac{1}{N}$. 我们来记录迭代次数、时间、误差与残差. 结果见表1-6. 这里误差指的是 h 乘以 u 与真解之差作为向量的 L^2 范数，可以看成它们作为函数的 L^2 范数的数值积分。对于 FEM 算法的数值结果如下:

N	ε	iter	soltime	error
32	1e+00	716	1.90e-01	3.44e-08
	1e-02	21	7.05e-03	1.85e-08
	1e-04	3	1.09e-03	3.60e-08
	1e-06	2	1.13e-03	3.58e-10
	1e-08	1	3.12e-04	3.46e-08
64	1e+00	2857	1.96e+00	3.51e-08
	1e-02	63	4.07e-02	3.51e-08
	1e-04	5	4.10e-03	3.34e-09
	1e-06	2	1.62e-03	5.73e-09
	1e-08	2	1.61e-03	5.72e-13

表 1: $\omega = 0$

N	ε	iter	soltime	errh
128	1e+00	11419	2.86e+01	3.53e-08
	1e-02	233	5.88e-01	3.48e-08
	1e-04	8	2.35e-02	5.61e-09
	1e-06	3	9.87e-03	1.51e-10
	1e-08	2	5.60e-03	9.17e-12
256	1e+00	45668	2.35e+02	3.53e-08
	1e-02	909	4.66e+00	3.64e-08
	1e-04	15	7.65e-02	3.44e-08
	1e-06	3	1.60e-02	9.56e-09
	1e-08	2	1.12e-02	1.46e-10

表 2: $\omega = 0$

N	ε	iter	soltime	errh
32	1e+00	716	1.70e-01	3.44e-08
	1e-02	450	1.10e-01	7.73e-05
	1e-04	451	1.05e-01	7.95e-05
	1e-06	451	1.06e-01	7.95e-05
	1e-08	451	1.06e-01	7.95e-05
64	1e+00	2857	1.56e+00	3.50e-08
	1e-02	1789	9.97e-01	1.95e-05
	1e-04	1795	9.71e-01	2.01e-05
	1e-06	1795	1.01e+00	2.01e-05
	1e-08	1795	9.81e-01	2.01e-05

表 3: $\omega = \pi/6$

N	ε	iter	soltime	errh
128	1e+00	11419	2.46e+01	3.53e-08
	1e-02	7144	1.62e+01	4.94e-06
	1e-04	7177	1.64e+01	5.08e-06
	1e-06	7177	1.61e+01	5.08e-06
	1e-08	7177	1.60e+01	5.08e-06
256	1e+00	45668	2.28e+02	3.53e-08
	1e-02	28562	1.46e+02	1.27e-06
	1e-04	28719	1.53e+02	1.31e-06
	1e-06	28723	1.47e+02	1.31e-06
	1e-08	28723	1.43e+02	1.31e-06

表 4: $\omega = \pi/6$

N	ε	iter	soltime	errh
32	1e+00	716	1.70e-01	3.44e-08
	1e-02	977	2.07e-01	9.59e-05
	1e-04	1012	2.17e-01	9.93e-05
	1e-06	1012	2.14e-01	9.93e-05
	1e-08	1012	2.15e-01	9.93e-05
64	1e+00	2857	1.46e+00	3.50e-08
	1e-02	4023	2.06e+00	2.44e-05
	1e-04	4259	2.16e+00	2.54e-05
	1e-06	4262	2.18e+00	2.54e-05
	1e-08	4262	2.20e+00	2.54e-05

表 5: $\omega = \pi/4$

N	ε	iter	soltime	errh
128	1e+00	11419	1.70e-01	3.44e-08
	1e-02	16317	2.07e-01	9.59e-05
	1e-04	17684	2.17e-01	9.93e-05
	1e-06	17705	2.14e-01	9.93e-05
	1e-08	17705	2.15e-01	9.93e-05
256	1e+00	45668	2.25e+02	3.53e-08
	1e-02	65588	3.45e+02	1.60e-06
	1e-04	72546	3.64e+02	1.67e-06
	1e-06	72687	3.68e+02	1.67e-06
	1e-08	72688	3.71e+02	1.67e-06

表 6: $\omega = \pi/4$

2.4 收敛性的数值分析

在本小节中,我们来通过数值计算来分析 Line Gauss-Seidel 迭代法迭代矩阵 $(D_A - L_A)^{-1}U_A$ 的谱半径 ρ 与迭代次数的关系,其中 $A = D_A - L_A - U_A$, D_A, L_A, U_A 分别为分块矩阵 A 的块状对角部分，块状下三角部分与块状上三角部分。我们令 $N = 64$，实验结果见图2与图3. 由实验结果我们可以看

出，$-1/\log(\rho)$ 与迭代次数有很大的相关性，这与我们熟知的理论结果相符。

2.5 $\omega = 0$ 情形收敛性分析

经过计算，我们发现 $\omega = 0$ 情形时 FEM 与 FDM 对应的线性方程组相同。因此，本小节的收敛性分析同样适用于基于 FDM 的 Line Gauss-Seidel 迭代法的 $\omega = 0$ 情形。

设

$$T = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{pmatrix}_{n-1\times n-1}$$

$$A = I \bigotimes T + \varepsilon T \bigotimes I = \begin{pmatrix} D & -\varepsilon I & & & \\ -\varepsilon I & D & -\varepsilon I & & \\ & -\varepsilon I & D & -\varepsilon I & \\ & & \ddots & \ddots & \ddots \\ & & & -\varepsilon I & D \end{pmatrix}. \tag{6}$$

我们此时有特征向量的显式形式，设

$$x_k = (\sin(k\pi h), \sin(2k\pi h), ..., \sin((n-1)k\pi h))^\top$$
$$\lambda_k = 2(1 - \cos k\pi h),\ k = 1, ..., n-1.$$

则 A 的特征向量为:

$$x_{k_1,k_2} = x_{k_1} \bigotimes x_{k_2} = \begin{pmatrix} \sin(k_1\pi h)\sin(k_2\pi h) \\ \sin(k_1\pi h)\sin(2k_2\pi h) \\ ... \\ \sin(k_1\pi h)\sin((n-1)k_2\pi h) \\ \sin(2k_1\pi h)\sin(k_2\pi h) \\ ... \\ \sin((n-1)k_1\pi h)\sin((n-1)k_2\pi h) \end{pmatrix}, k_1, k_2 = 1, ..., n-1.$$

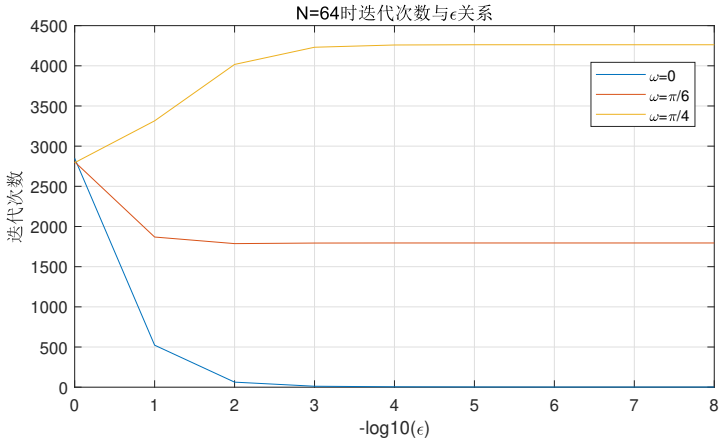

图 2: $N = 64$ 时迭代次数与 ε 关系

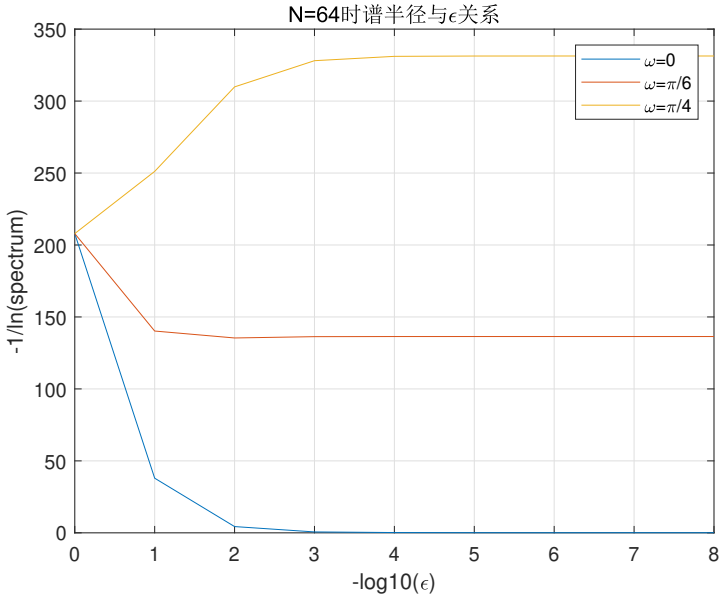

图 3: $N = 64$ 时 Line Gauss-Seidel 迭代矩阵的谱半径与 ε 关系

分析的关键是这些特征向量也是迭代法的特征向量:

Jacobi 迭代 设 $D_0 = \mathrm{diag}(A)$, 则 Jacobi 迭代矩阵 $D_0^{-1}(D_0 - A)$, 对某个特征向量 x_{k_1,k_2}, 有:

$$D_0^{-1}(D_0 - A)x_{k_1,k_2} = (1 - \frac{\lambda_{k_2} + \varepsilon\lambda_{k_1}}{2 + 2\varepsilon})x_{k_1,k_2}$$

x 前的因子在 $\lambda_{k_2} = \lambda_{k_1} = 2(1 - \cos\pi h)$ 时最大, 为 $\cos\pi h = \cos\frac{\pi}{N}$。

Line Jacobi 迭代 设 $D_1 = I \bigotimes D$, 则 Line Jacobi 迭代矩阵为 $M = D_1^{-1}(D_1 - A)$。对某个特征向量 $x = x_{k_1,k_2}$,

$$\begin{aligned} D_1 x_{k_1,k_2} &= (I \bigotimes T + 2\varepsilon I \bigotimes I)x_{k_1,k_2} \\ &= (I \bigotimes T)(x_{k_1} \bigotimes x_{k_2}) + 2\varepsilon x_{k_1,k_2} \\ &= (\lambda_{k_2} + 2\varepsilon)x_{k_1,k_2}. \end{aligned}$$

故

$$\begin{aligned} Mx = D_1^{-1}(D_1 - A)x &= x - (\lambda_{k_2} + 2\varepsilon)^{-1}(\lambda_{k_2} + \varepsilon\lambda_{k_1})x \\ &= \frac{\varepsilon(2 - \lambda_{k_1})}{\lambda_{k_2} + 2\varepsilon}x \\ &= \frac{\varepsilon}{\varepsilon + 1 - \cos\frac{k_2\pi}{N}} \cos\frac{k_1\pi}{N}. \end{aligned}$$

谱半径

$$\rho = \frac{\varepsilon}{\varepsilon + 1 - \cos\frac{\pi}{N}} \cos\frac{\pi}{N} < \cos\frac{\pi}{N}.$$

通过熟知的变换可以得到 GS 的特征值是 Jacobi 的平方。

子空间矫正的观点 Line Jacobi 可以看成对

$$V_i = \mathrm{span}\{e_{i,1}, ...e_{i,n-1}\}, i = 1, 2, ..., n - 1.$$

的并行子空间矫正。V_i 即第 i 行对应的子空间。

$\varepsilon \to 0$ 时, Line Jacobi 退化成求解 A_0 的直接法。

2.6 小结

在本节中，我们给出了 FEM 算法。从数值实验来看，该方法在 $\omega = 0$ 且 ε 较小时收敛很快，这一点在理论分析中也得到验证；当 $\omega = \pi/6$ 时，随着 ε 的减少，迭代次数只略有减少；当 $\omega = \pi/4$ 时，随着 ε 的减少，迭代次数甚至有所增加。以上实验结果与对 Line Gauss-Seidel 迭代矩阵的谱半径的数值结果相一致。

3 FDM45 算法

3.1 算法基本思想

在第 2 节中，我们可以看到当网格方向与区域边界平行、而不与各向异性的方向平行时，迭代次数较多。受此启发，在本节中，我们来介绍 FDM45 算法。该算法的思想是构造与坐标轴方向平行的网格，类似于 FEM 方法的 $\omega = 0$ 情形，我们认为在 ε 较小时具有较快的收敛速度。这一点在之后的数值实验中将得到验证。

在本节中，我们将专注于实现 $\omega = \pi/4$ 的情形。我们考虑在一个包括该区域的大正方形内部打均匀网格，其中大正方形的边界平行于坐标轴。利用 $\omega = \pi/4$ 的特殊性，我们可以使得一部分节点落在区域的边界上。在第 4 节中，我们推广这一想法并加上对边界的处理，则可以将此算法推广至任意旋转角度 ω 甚至任意有界区域上去。

3.2 算法介绍

记号声明 在FDM45 算法中,我们令网格不随区域一起旋转。利用 $\omega = \pi/4$ 的特殊性，我们设计如图4所示的与坐标轴平行的均匀网格。在数值解 u 的记号上，我们按照包括本区域的大正方形上的均匀网格进行编号，即此均匀网格上第 i 行第 j 列为 $u_{i,j}$. 我们取大正方形的均匀网格的离散数取为 $2N$. 此时，若节点未落入区域 Ω 内，我们将其值置为 0.

方程的离散 在本算法中，方程的离散我们将采用标准的中心差分格式，即

$$2(1+\varepsilon)u_{i,j} - u_{i,j-1} - u_{i,j+1} - \varepsilon(u_{i+1,j} + u_{i-1,j}) = f_{i,j}h^2,\ i,j = 2,3,\ldots,2N. \tag{7}$$

矩阵形式 我们注意到，按照之前的记号说明，数值解 u 中有大量元素均为 0. 我们关心 u 非零的部分，即在区域 Ω 内部的点，拉直成向量即为：

$$u = (u_{2,N+1}; u_{3,N}, \ldots, u_{3,N+2}; \ldots; u_{2N,N+1})^T. \tag{8}$$

写成矩阵形式，即为

$$\begin{pmatrix} A_1 & B_1 & & & & & \\ B_1^T & A_2 & \ddots & & & & \\ & \ddots & \ddots & B_{2N-2} & & & \\ & & B_{2N-2}^T & A_{2N-1} & \ddots & & \\ & & & \ddots & \ddots & B_2 & \\ & & & & B_2^T & A_2 & B_1 \\ & & & & & B_1^T & A_1 \end{pmatrix} u = f. \tag{9}$$

其中，

$$A_k = \begin{pmatrix} 2(1+\varepsilon) & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2(1+\varepsilon) \end{pmatrix}, B_k^T = \begin{pmatrix} 0 & \cdots & 0 \\ -\varepsilon & & \\ & \ddots & \\ & & \varepsilon \\ 0 & \cdots & 0 \end{pmatrix}, \tag{10}$$

Line Gauss-Seidel 迭代法 在 FDM45 算法中，我们令网格不随区域一起旋转。利用 $\omega = \pi/4$ 的特殊性，我们设计如图4所示的与坐标轴平行的均匀网格，每次 Line Gauss-Seidel 迭代按图中所示方法逐行更新，即依次对每行根据绿色节点来更新红色节点。

3.3 数值结果

我们的数值算例设定与第 2 节相同，令 $\omega = \pi/4$.

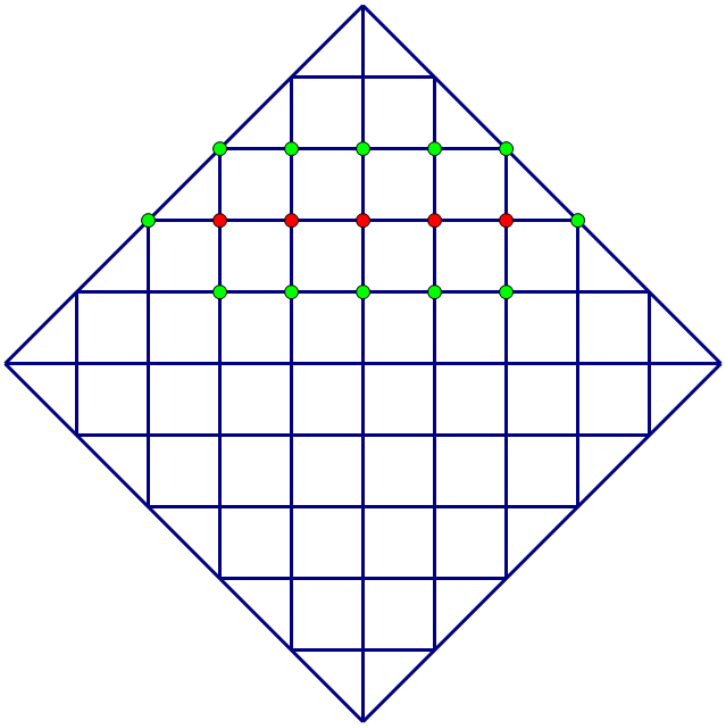

图 4: $N = 5$ 时 FDM45 网格与 Line Gauss-Seidel 迭代更新方式示意图

N	ε	iter	soltime	errh
32	1e+00	1399	1.63e+00	2.38e-02
	1e-02	46	4.24e-02	7.12e-04
	1e-04	4	4.28e-03	2.87e-05
	1e-06	2	2.75e-03	3.63e-05
	1e-08	2	2.10e-03	3.64e-05
64	1e+00	5581	1.44e+01	2.38e-02
	1e-02	168	5.42e-01	7.38e-04
	1e-04	7	2.28e-02	2.09e-06
	1e-06	2	4.31e-03	8.95e-06
	1e-08	2	8.58e-03	9.10e-06

表 7: FDM45

N	ε	iter	soltime	errh
128	1e+00	22307	1.17e+02	2.38e-02
	1e-02	655	3.01e+00	7.44e-04
	1e-04	13	6.38e-02	5.73e-06
	1e-06	3	1.37e-02	2.19e-06
	1e-08	2	8.62e-03	2.27e-06
256	1e+00	89201	1.42e+03	2.38e-02
	1e-02	2604	5.22e+01	7.46e-04
	1e-04	36	6.67e-01	7.40e-06
	1e-06	4	7.35e-02	4.84e-07
	1e-08	2	2.93e-02	5.66e-07

表 8: FDM45

从数值结果中可以看出随着 ε 的减小迭代法收敛速度大大加快, 且解的精确度提高. 这是由于 ε 取值小时问题本身性质好所致. 从结果看当 N 变化是, 求解代价大约在 $O(N^3)$ 量级.

3.4 小结

在本节中，我们给出了 FDM45 算法。从数值实验来看，该方法在 $\omega = \pi/4$ 且 ε 较小时收敛很快. 按照之前的分析，这依赖于我们的网格是平行于坐标轴的。我们将这个想法推而广之，即得到下一节关于一般旋转角度 ω、甚至一般有界区域的算法。

4 FDM30 算法 (均匀撒点法)

4.1 算法基本想法

我们已经知道, 网格方向和各向异性方向平行时, 线高斯有非常不错的表现。从大的类别来说, 均匀撒点法的动机是在 $\omega \neq 0$ 的情形下打和 x 轴（各向异性方向）平行的网格。

想在旋转后的正方形上构造平行于 x 轴的网格并不容易。除了 $\omega = \frac{\pi}{4}$ 这种几何关系特殊的情形外，对一般的 ω 似乎难以找到好的剖分方法。这本

质困难或许来源于 $\cos(\omega)$ 的无理性，其精神源头可以追溯到经典的几何剖分问题以及数论中对无理数的逼近。这种情况下一个精巧的网格，背后很可能对应着对 $\cos(\omega)$ 的一个分数逼近。

我们的动机是放弃寻找精巧地内嵌于区域的网格，力求保留 $\omega = 0$ 时网格的结构。我们的想法是把旋转后的正方形放到一个更大的均匀网格里面，然后以落入正方形内部的均匀网格格点作为格点，如图 5，这样，我们就得到了一个和 $\omega = 0$ 很像的均匀网格。

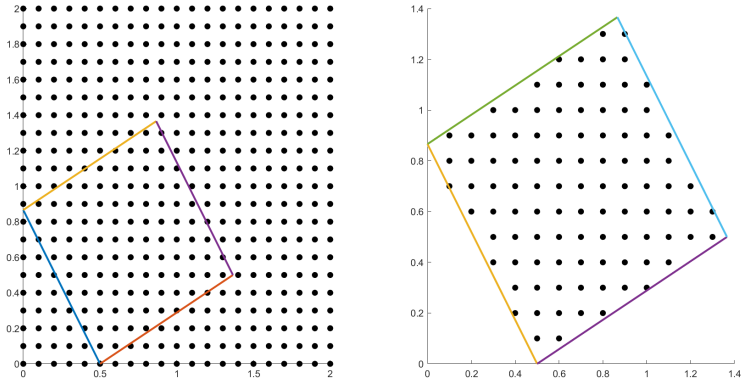

图 5: 均匀撒点法 (FDM30) 的想法，左：把区域放到大的均匀网格里面。右：保留落入正方形的顶点

进而我们希望使用五点差分进行计算，但边界需要额外处理，我们的选择是把网格线延长交到边界上，利用网格线和正方形边界的交点进行计算，想法来自泰勒展开。

我们来看一个一维的例子：设 $u(0)$ 是一个在边界附近的格点（实际上是格点处的函数值，为叙述方便，以格点称之），$u(h)$ 是 $u(0)$ 右边的点，在区域内部，而 $u(-\delta h)$ 是网格线与区域边界的交点。这里 $\delta < 1$。假设 u 足够好，我们有

$$u(h) = u(0) + hu'(0) + \frac{1}{2}u''(0)h^2 + o(h^2).$$
$$u(-\delta h) = u(0) - \delta hu'(0) + \frac{1}{2}u''(0)(\delta h)^2 + o(h^2).$$

故

$$\delta u(h) + u(-\delta h) - (1+\delta)u(0) = \frac{\delta+\delta^2}{2}u''(0)h^2 + o(h^2).$$

进一步我们假设 $u(-\delta h)$ 作为边值是 0，我们有

$$\delta u(h) - (1+\delta)u(0) \approx \frac{\delta+\delta^2}{2}u''(0)h^2$$

因此我们认为

$$-u''(0) \approx \frac{1}{h^2}(\frac{2}{\delta}u(0) - \frac{2}{1+\delta}u(h)). \tag{11}$$

是一个不错的近似关系。

实践中我们正是利用这一关系处理在边界上的顶点，具体做法参见下节的细节描述中的 (13),(14)。

4.2 算法细节描述

我们遵循大的均匀网格对顶点的编号,即,第 i 行从左到右是 $u_{i,1}, u_{i,2}, ...u_{i,n}, ...$, 第 j 列从下到上是 $u_{1,j}, .., u_{n,j}, ...$。也就是说，即使格点没落在正方形内部，我们也保存它的编号，并把 u 在此格点处的值置零。这样做有实现和叙述上的方便。

我们按照原来网格的编号，即，第 i 行从左到右是 $u_{i,1}, u_{i,2}, ...u_{i,n}, ...$, 第 j 列从下到上是 $u_{1,j}, .., u_{n,j}, ...$。即使格点没落在正方形内部，我们也保存它的编号，并把 u 在此格点处的值置零。这样做有实现和叙述上的方便。

我们设第 $2, 3, 4..., N_y$ 行有格点落入正方形，第 i 行最左边的格点是 (i, L_i)，最右边的格点是 (i, R_i)。即第 i 行从左到右是落入正方形的点是 $(i, L_i), (i, L_i+1), ...(i, R_i)$。

方程的离散

对于内部格点 (i, j)(即上下左右都在正方形内部)，我们可以用经典的等步长中心差分来逼近 x, y 方向的二阶导数，得到如下的离散格式：

$$(2+2\varepsilon)u_{i,j} - u_{i,j+1} - u_{i,j-1} - \varepsilon(u_{i+1,j} + u_{i-1,j}) = f_{i,j}h^2. \ (i,j)\text{在内部.} \tag{12}$$

这里 h 是均匀网格的步长。

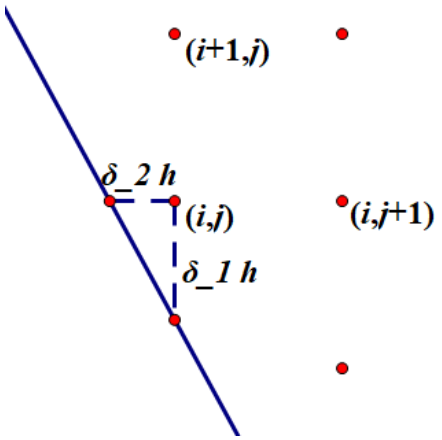

图 6: 格点靠近边界的情况

对于靠近边界的格点，我们采用类似 (11) 的方式，以图 6 中的 (i,j) 为例（按上述记号的话 $j = L_i$），它的左边和下边没有格点。我们把网格线向下延长，交正方形于 $(i-\delta_1, j)$，这里 $i-\delta_1$ 的含义是 (i,j) 到交点的距离是 $\delta_1 h$, δ_1 可以由十分简单的计算得出。(当然我们这里有 $\delta_1 < 1$, 不然它的下方就应该有格点了)

由于 $u_{i-\delta_1,j}$ 作为边值是 0，应用上小节泰勒展开的结果 (11), 我们有对 y 方向 2 阶导数的近似：

$$-\frac{\partial^2 u}{\partial y^2}|_{(i,j)} \approx \frac{1}{h^2}(\frac{2}{\delta_1}u_{i,j} - \frac{2}{1+\delta_1}u_{i+1,j}). \tag{13}$$

类似地，我们把网格线向左延长交正方形于 $(i, j-\delta_2)$, 同样，这里 $j-\delta_2$ 的代表 (i,j) 到交点的距离是 $\delta_2 h$, 应用 (11), 我们得到对 x 方向 2 阶导数的近似：

$$-\frac{\partial^2 u}{\partial x^2}|_{(i,j)} \approx \frac{1}{h^2}(\frac{2}{\delta_2}u_{i,j} - \frac{2}{1+\delta_2}u_{i,j+1}). \tag{14}$$

把两个逼近 (13)、(14) 带到方程里去，我们就得到了 (i,j) 这点的离散格式

$$(\frac{2}{\delta_2} + \varepsilon\frac{2}{\delta_1})u_{i,j} - \frac{2}{1+\delta_2}u_{i,j+1} - \frac{2}{1+\delta_1}u_{i+1,j} = f_{i,j}h^2. \tag{15}$$

其他靠近边界的格点也可以这样处理。结合“内点”时的 (12) 我们就得到全部格点的离散方程。

关于落单顶点的注记 [1]细心的读者可能会注意到，我们讨论的都是每个格点水平、垂直方向上至少有一个相邻格点的情况，因此可以应用 (11)。会不会出现一个格点左右都没有相邻的格点的情况呢? 事实上，在网格步长很小的时候，正方形最上端可能会有一个落单的格点，我们实践中的处理是认为这个格点就是在边界上的点，该格点处的解始终取边值 0。

矩阵形式 如果我们把我们关心的 u(即不恒为 0 的部分) 拉直成向量:

$$u = (u_{2,L_2}, u_{2,L_2+1}, ...u_{2,R_2}, ..., u_{N_y,L_{N_y}}, u_{N_y,L_{N_y}+1}..., u_{N_y,R_{N_y}})^\top. \tag{16}$$

那我们可以写出上述格式 (12)(15) 的矩阵形式

$$\begin{pmatrix} A_2 & B_2^{up} & & & & & \\ B_3^{down} & A_3 & B_3^{up} & & & & \\ & B_4^{down} & A_4 & B_4^{up} & & & \\ & & B_5^{down} & A_5 & B_5^{up} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & B_{N_y}^{down} & A_{N_y} \end{pmatrix} u = f. \tag{17}$$

设 $l_i = R_i - L_i + 1$ 是第 i 行在内部的格点数目。则这里 A_i 是 $l_i \times l_i$ 矩阵 $(i = 2, ..N_y)$，B_i^{down} 是 $l_i \times l_{i-1}$ 矩阵 $(i = 3, ..N_y)$,B_i^{up} 是 $l_i \times l_{i+1}$ 矩阵 $(i = 2, ..N_y - 1)$。

线高斯 我们实际中不会去直接求解矩阵方程 (17), 而是用线高斯的想法逐行更新，图 7 画出了一步典型的线高斯更新，红色为更新的点，绿色为用到的点, 蓝色是其他格点。

[1]初次阅读时可以略过

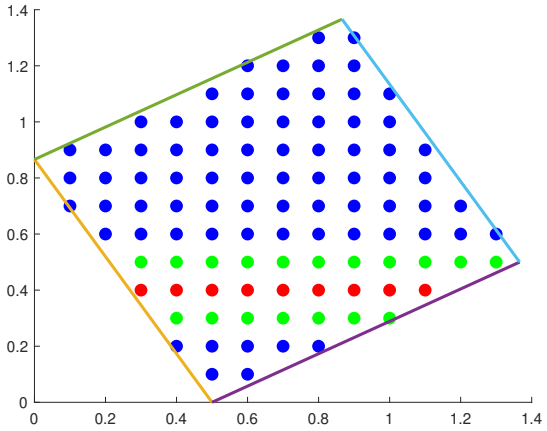

图 7: 一步典型的线高斯更新，红色为更新的点，绿色为用到的点, 蓝色是其他格点

结合 (12),(15) 我们可以写出更新第 i 行时需要求解的线性代数方程 $Ax = b$, 这里

$$A = \begin{pmatrix} \frac{2}{\delta_i^L} + \varepsilon\frac{2}{\delta^1_{i,L_i}} & -\frac{2}{1+\delta_i^L} & & & & & \\ -1 & 2+\varepsilon\frac{2}{\delta^1_{i,L_i+1}} & -1 & & & & \\ & -1 & 2+\varepsilon\frac{2}{\delta^1_{i,L_i+2}} & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2+\varepsilon\frac{2}{\delta^1_{i,R_i-1}} & -1 & \\ & & & & -\frac{2}{1+\delta_i^R} & \frac{2}{\delta_i^R} + \varepsilon\frac{2}{\delta^1_{i,R_i}} \end{pmatrix}, \tag{18}$$

$$x = \begin{pmatrix} u_{i,L_i} \\ u_{i,L_i+1} \\ \vdots \\ u_{i,R_i} \end{pmatrix}, b = \begin{pmatrix} h^2 f_{i,L_i} + \frac{2}{\delta^{down}_{i,L_i}} u_{i-1,L_i} + \frac{2}{\delta^{up}_{i,L_i}} u_{i+1,L_i} \\ h^2 f_{i,L_i+1} + \frac{2}{\delta^{down}_{i,L_i+1}} u_{i-1,L_i+1} + \frac{2}{\delta^{up}_{i,L_i+1}} u_{i+1,L_i+1} \\ \vdots \\ h^2 f_{i,R_i} + \frac{2}{\delta^{down}_{i,R_i}} u_{i-1,R_i} + \frac{2}{\delta^{up}_{i,R_i}} u_{i+1,R_i}. \end{pmatrix} \tag{19}$$

这里出现了好多个 δ 它们的含义和 (15) 中类似，具体解释如下:

1. δ_i^L, δ_i^R: 左、右格点到边界 (网格和边界交点) 的距离与网格步长 h 的比值。

2. $\delta_{i,j}^1$: 如果 (i,j) 上方或者下方没有格点，则是它到边界 (网格和边界交点) 的距离与网格步长 h 的比值，否则是 1。

3. $\delta_{i,j}^{up}$ 如果 (i,j) 下方没有格点，则是 $\delta_{i,j}^1$，否则是 1。因为这是乘在上方格点的系数，故用 up。

4. $\delta_{i,j}^{down}$ 如果 (i,j) 上方没有格点，则是 $\delta_{i,j}^1$，否则是 1。因为这是乘在下方格点的系数，故用 down。

可见，大部分的 δ 其实是 1。注意到我们把外部格点处的值都设为 0，因此 (19)b 的部分不必考虑 L_i, L_{i+1} 的大小关系。

4.3 数值结果及分析

我们来计算 $\omega = \frac{\pi}{6}$ 的情况。计算区域 Ω 如图 (5)

$$\Omega = \{(x,y) : 0 \le \sqrt{3}x + y - \frac{\sqrt{3}}{2} \le 2,\ 0 \le x - \sqrt{3}y + \frac{3}{2} \le 2\}.$$

真解为

$$u = (\sqrt{3}x + y - \frac{\sqrt{3}}{2})(\sqrt{3}x + y - \frac{\sqrt{3}}{2} - 2)(x - \sqrt{3}y + \frac{3}{2})(x - \sqrt{3}y - \frac{1}{2}) \quad (20)$$

我们通过 matlab 符号微分，将 $(-\frac{\partial^2 u}{\partial x^2} - \varepsilon\frac{\partial^2 u}{\partial y^2})$ 作用于 (20) 计算出外力，然后使用 FDM30(均匀撒点法) 进行求解。

我们对 N $= 2^5, 2^6, 2^7, 2^8$;$\varepsilon = 1, 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ 进行求解。这里 N 指均匀网格的步长 $h = \frac{1}{N}$. 迭代次数、时间、误差、残差如表 9 与表10。这里误差指的是 h 乘以 u 与真解之差作为向量的 L^2 范数，可以看成它们作为函数的 L^2 范数的数值积分。

N	eps	iter	soltime	errh	res
32	1e+00	716	1.16e+00	4.52e-04	3.46e-07
	1e-02	23	2.11e-01	5.06e-04	2.02e-07
	1e-04	4	1.64e-01	5.73e-06	3.53e-07
	1e-06	2	1.65e-01	2.19e-06	2.12e-09
	1e-08	1	4.74e-01	2.27e-06	1.70e-07
64	1e+00	2856	1.24e+01	2.44e-04	1.74e-07
	1e-02	76	4.74e-01	1.30e-04	9.71e-08
	1e-04	5	1.67e-01	1.32e-04	2.02e-08
	1e-06	2	1.89e-01	1.32e-04	1.69e-08
	1e-08	2	1.60e-01	1.32e-04	1.70e-12

表 9: FDM30 数值结果 1

N	eps	iter	soltime	errh	res
128	1e+00	716	1.79e+02	2.89e-05	8.74e-08
	1e-02	23	4.41e+00	3.25e-05	5.48e-08
	1e-04	4	3.20e-01	3.32e-05	2.94e-08
	1e-06	2	2.13e-01	3.36e-05	2.88e-10
	1e-08	1	1.90e-01	3.36e-05	1.36e-11
256	1e+00	45665	3.29e+03	6.84e-06	4.37e-08
	1e-02	1115	8.39e+01	7.56e-06	2.85e-08
	1e-04	18	1.68e+00	8.02e-06	1.54e-08
	1e-06	3	6.32e-01	8.20e-06	9.04e-09
	1e-08	2	4.53e-01	8.48e-06	1.09e-10

表 10: FDM30 数值结果 2

为了更加直观，我们画出不同 ε 下误差和 N 的关系，均取以 2 为底对数：

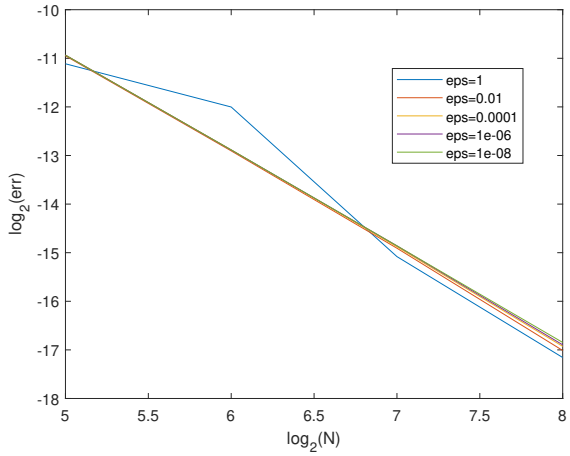

图 8: 不同 ε 下误差和 N 的关系，均取以 2 为底对数

可见，我们的格式达到了 2 阶精度。再画出不同 N 时所用时间和 ε 之间关系，均取以 10 为底对数

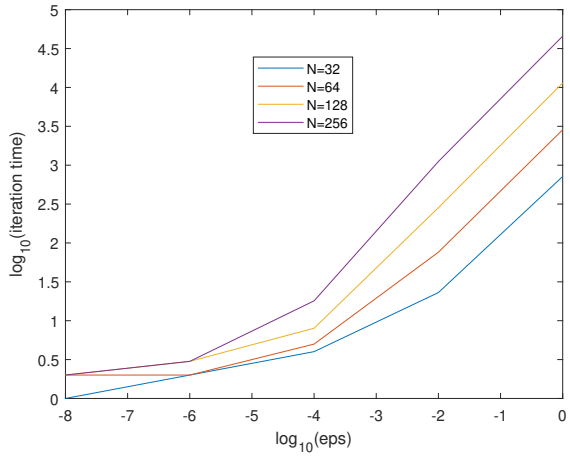

图 9: 不同 N 时所用时间和 ε 之间关系，均取以 10 为底对数

可见，和 $\omega = 0$ 时的线高斯、FDM45 一样，在 ε 小的时候我们的算法迅速。

4.4 讨论

算法的数值表现不错，但是目前缺少理论分析。感觉有两方面的理论分析可以做，一是初等几何上的，分析到底有多少顶点在边界，弄清楚孤立顶点是不是只可能在顶点出现等。另一方面则是分析截断误差、以及离散线性代数方程 (17) 的性质。

尽管我们只算了 $\omega = \frac{\pi}{6}$ 的例子，但本算法不难推广到一般的 ω。甚至有推广到一般区域的潜力。这也需要进一步的研究。

5 FDM 旋转算子法

我们希望将有限差分方法运用于旋转后的区域，且网格跟随旋转. 为此，考虑算子旋转后的形式.

5.1 算子旋转后的形式

将坐标轴逆时针旋转 ω，一点在此标架下的坐标 (x', y') 与原坐标 (x, y) 有如下关系：

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

由链式法则：

$$\frac{\partial}{\partial x} = \cos(\omega)\frac{\partial}{\partial x'} - \sin(\omega)\frac{\partial}{\partial y'}$$

故：

$$\frac{\partial^2}{\partial x^2} = \cos^2(\omega)\frac{\partial^2}{\partial x'^2} - 2\cos(\omega)\sin(\omega)\frac{\partial^2}{\partial x'\partial y'} + \sin^2(\omega)\frac{\partial^2}{\partial y'^2}$$

同理

$$\frac{\partial^2}{\partial y^2} = \sin^2(\omega)\frac{\partial^2}{\partial x'^2} + 2\cos(\omega)\sin(\omega)\frac{\partial^2}{\partial x'\partial y'} + \cos^2(\omega)\frac{\partial^2}{\partial y'^2}$$

于是

$$\begin{aligned} &\frac{\partial^2}{\partial x^2} + \varepsilon\frac{\partial^2}{\partial y^2} \\ &= (\cos^2(\omega)+\varepsilon\sin^2(\omega))\frac{\partial^2}{\partial x'^2} - 2(1-\varepsilon)\cos(\omega)\sin(\omega)\frac{\partial^2}{\partial x'\partial y'} + (\sin^2(\omega)+\varepsilon\cos^2(\omega))\frac{\partial^2}{\partial y'^2} \end{aligned}$$

5.2 九点差分的引入

这启发我们用旋转后均匀网格上的差分来表示原来的算子 $\frac{\partial^2}{\partial x^2}+\varepsilon\frac{\partial^2}{\partial y^2}$，由于交叉项的存在，我们在五点差分的基础上需要额外四个点，以内部点 $u_{i,j}$ 为例，此时

$$\begin{aligned}\frac{\partial^2}{\partial x'^2}&=(u_{i+1,j}+u_{i-1,j}-2u_{i,j})/h^2\\ \frac{\partial^2}{\partial y'^2}&=(u_{i,j+1}+u_{i,j+1}-2u_{i,j})/h^2\\ \frac{\partial^2}{\partial x'\partial y'}&=(u_{i+1,j+1}+u_{i-1,j-1}-u_{i-1,j+1}-u_{i+1,j-1})/h^2\end{aligned}$$

此时我们有矩阵

$$M=\begin{pmatrix}A & B & & & \\ B^\top & A & B & & \\ & \ddots & \ddots & \ddots & \\ & & B^\top & A & B\\ & & & B^\top & A\end{pmatrix},\tag{21}$$

其中 A 与 B 为分块矩阵，形如：

$$A=\begin{pmatrix}\alpha & \beta & & & \\ \beta & \alpha & \beta & & \\ & \ddots & \ddots & \ddots & \\ & & \beta & \alpha & \beta\\ & & & \beta & \alpha\end{pmatrix},\ B=\begin{pmatrix}\gamma & \delta & & & \\ -\delta & \gamma & \delta & & \\ & & \ddots & \ddots & \\ & & -\delta & \gamma & \delta\\ & & & -\delta & \gamma\end{pmatrix},\tag{22}$$

其中

$$\begin{cases}\alpha=2(1+\varepsilon),\\ \beta=-c^2-\varepsilon s^2,\\ \gamma=-s^2-\varepsilon c^2,\\ \delta=(1-\varepsilon)cs/2,\\ c=\cos\omega, s=\sin\omega.\end{cases}\tag{23}$$

5.3 数值结果

我们所使用的算例依旧为Equation 5，由矩阵分块三对角的特性，我们用线高斯来解此方程. 其收敛性理论见subsection 2.5. 同时我们在Figure 10与Figure 11中

N	ε	iter	soltime	errh
128	1e+00	11385	3.42e+01	1.66e-02
	1e-02	233	4.35e-01	3.51e-04
	1e-04	8	1.48e-02	3.67e-06
	1e-06	3	5.47e-03	3.69e-08
	1e-08	2	3.66e-03	3.76e-10
256	1e+00	45526	2.74e+02	1.66e-02
	1e-02	909	3.16e+00	3.51e-04
	1e-04	15	6.11e-02	3.70e-06
	1e-06	3	1.18e-02	4.58e-08
	1e-08	2	8.34e-03	5.06e-10

表 12: FDM: $\omega = 0$

展示了 N=64 时谱半径和迭代次数随 ε 的变化，他们呈现出与 FEM 算法相似的特征.

N	ε	iter	soltime	errh
32	1e+00	714	2.12e-01	1.66e-02
	1e-02	21	6.10e-03	3.51e-04
	1e-04	3	1.70e-03	3.67e-06
	1e-06	2	5.92e-04	3.67e-08
	1e-08	1	8.42e-04	3.49e-08
64	1e+00	2849	2.25e+02	1.66e-02
	1e-02	63	3.45e+02	3.51e-04
	1e-04	5	3.64e+02	3.66e-06
	1e-06	2	3.68e+02	4.20e-08
	1e-08	2	3.71e+02	3.67e-10

表 11: FDM: $\omega = 0$

N	ε	iter	soltime	errh
32	1e+00	704	1.29e-01	1.68e-02
	1e-02	450	9.98e-02	4.41e-04
	1e-04	451	8.20e-02	4.57e-06
	1e-06	451	8.80e-02	9.94e-08
	1e-08	451	8.05e-02	5.63e-08
64	1e+00	2805	1.01e+00	1.68e-02
	1e-02	1788	6.18e-01	4.45e-04
	1e-04	1797	5.62e-01	4.61e-06
	1e-06	1797	5.25e-01	1.02e-07
	1e-08	1797	5.64e-01	5.92e-08

表 13: FDM: $\omega = \pi/6$

N	ε	iter	soltime	errh
128	1e+00	11205	1.08e+01	1.68e-02
	1e-02	7138	8.70e+00	4.46e-04
	1e-04	7181	9.92e+00	4.63e-06
	1e-06	7182	8.32e+00	1.03e-07
	1e-08	7182	7.84e+00	6.05e-08
256	1e+00	44797	1.51e+02	1.68e-02
	1e-02	28536	1.07e+02	4.46e-04
	1e-04	28731	1.11e+02	4.64e-06
	1e-06	28736	1.08e+02	1.04e-07
	1e-08	28736	1.09e+02	6.10e-08

表 14: FDM: $\omega = \pi/6$

N	ε	iter	soltime	errh
32	1e+00	701	6.55e-02	1.68e-02
	1e-02	994	9.90e-02	5.15e-04
	1e-04	1041	1.01e-01	5.42e-06
	1e-06	1042	1.05e-01	1.23e-07
	1e-08	1042	1.04e-01	7.13e-08
64	1e+00	2794	7.23e-01	1.68e-02
	1e-02	4055	1.15e+00	5.23e-04
	1e-04	4350	1.24e+00	5.55e-06
	1e-06	4354	1.23e+00	1.30e-07
	1e-08	4354	1.23e+00	7.87e-08

表 15: FDM: $\omega = \pi/4$

N	ε	iter	soltime	errh
128	1e+00	11159	1.03e+01	1.68e-02
	1e-02	16350	1.85e+01	5.26e-04
	1e-04	17939	2.14e+01	5.60e-06
	1e-06	17967	2.07e+01	1.34e-07
	1e-08	17968	2.07e+01	8.38e-08
256	1e+00	44611	1.51e+02	1.68e-02
	1e-02	65560	2.64e+02	5.27e-04
	1e-04	73205	2.99e+02	5.62e-06
	1e-06	73382	2.89e+02	1.35e-07
	1e-08	73384	2.83e+02	8.75e-08

表 16: FDM: $\omega = \pi/4$

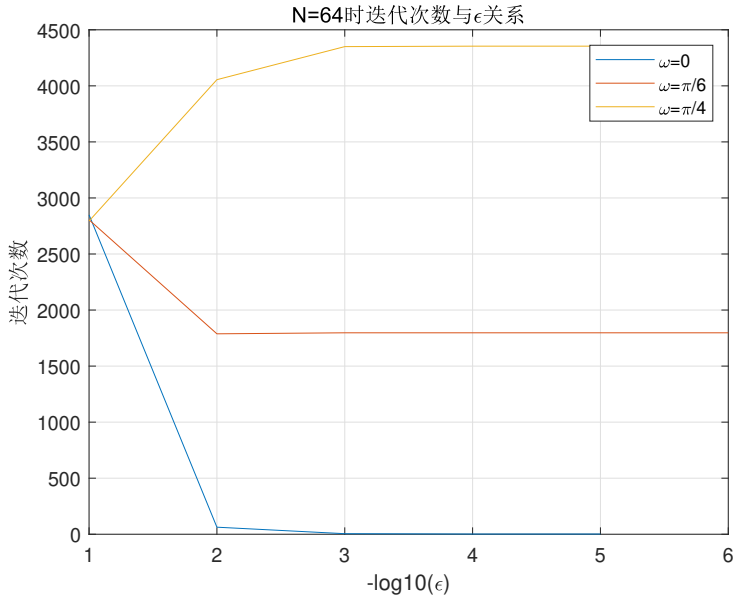

图 10: $N = 64$ 时迭代次数与 ε 关系

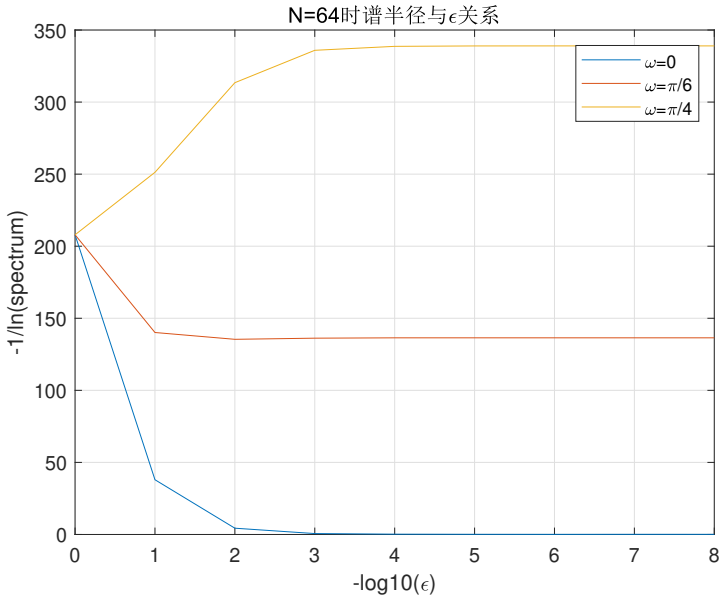

图 11: $N = 64$ 时 Line Gauss-Seidel 迭代矩阵的谱半径与 ε 关系

6 总结

纵向对比几种方法及其数值结果，我们可以观察到这样的现象：当网格纵横方向与坐标轴平行，即 Line Gauss-Seidel 迭代的“线”在 Possion 方程的各向异性方向上时，对于较小的 ε 算法表现出很优越的性能，往往只需迭代两三次就能得到很准确的近似解，如 FDM30、FDM45 算法以及 FEN、FDM 旋转算子法求解 $\omega = 0$ 的问题。反之，当网格方向与坐标轴有夹角时，算法收敛速度受 ε 影响很小，在 ε 很小时仍需上千次迭代。

由此可见选取能够与方程各向异性方向对应的网格，结合线性高斯迭代，对于各向异性较明显,$\varepsilon \ll 1$ 的方程效果显著。这是因为在如此选取的网格和线性高斯迭代下，每一步的更新大部分地落在误差所在子空间里，使得算法表现出优越的性能。

FEM 方法和 FDM 旋转算子法选取的是与旋转后的区域边界平行而不是与各向异性方向平行的网格，两种方法能够轻易地推广到一般的 ω，但对于 $\varepsilon \ll 1$ 的情形两种方法并不十分合适。FDM45 算法利用了 $\omega = \pi/4$ 时平行于坐标轴的网格格点可以落在边界上的特点，比较具有特殊性。FDM30 算法则以 $\omega = \pi/6$ 为例解决了格点不能落在边界上时的问题，需要对边界处的格点进行较为细致的处理，但具有推广到一般的 ω 乃至一般的区域上的发展潜力。