并行计算作业 2

李若泰
1601110032

2017 年 11 月 19 日

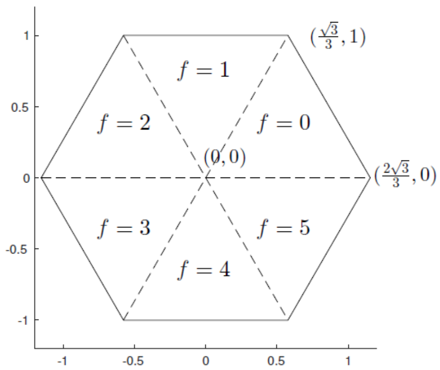
1 问题描述与分析

1.1 问题描述

考虑二维正六边形区域内的 $\Omega = [-\frac{2\sqrt{3}}{3}, \frac{2\sqrt{3}}{3}] \times [-1, 1]$ 内的泊松方程如下：

$$\begin{cases} -\Delta u = f, \ , \ x \in \Omega; \\ \frac{\partial u}{\partial n} = c, \ x \in \partial\Omega \end{cases} \tag{1}$$

其中，Ω 为正六边形，f 为分块常数，将正六边形区域 Ω 等分为六块，逆时针方向，从右往左分别取值为：$f = 0, 1, 2, 3, 4, 5$，$\frac{\partial u}{\partial n}$ 表示外法向导数，$\vec{n}$ 表示为边的单位外法向，c 为未知常数。如下图：



1.2 问题分析

对于上面问题，首先边界条件为 Neumman 边值，但是常数 c 并没有给出，需要求解。通过简单分析, 对于方程(1)，应用 Gause-Green 公式，得：

$$\begin{aligned} \int_{x\in\Omega} f \ dx &= \int_{x\in\Omega} -\Delta u \ dx \\ &= \int_{x\in\partial\Omega} -\frac{\partial u}{\partial n} \\ &= \int_{x\in\partial\Omega} -c \ dx \end{aligned} \tag{2}$$

由于 f 为分片常数，通过简单计算，得到 $\int_{x\in\Omega} f\ dx = 5\sqrt{3}$，而 $\int_{x\in\partial\Omega} dx = 4\sqrt{3}$，则 $c = -\frac{5}{4} = -1.25$，这样，得到了具体的 $Neumman$ 边值条件。

接下来，我们考虑网格的剖分，由于区域 Ω 不是规则的矩阵区域，对于矩阵剖分网格来说，比较难处理，因为选择均匀的矩阵剖分，会有点落在区域外面，而且，不能保证区域的边界上，总有剖分点存在。同时，在这种情况下，点的排列也相对比较困难。

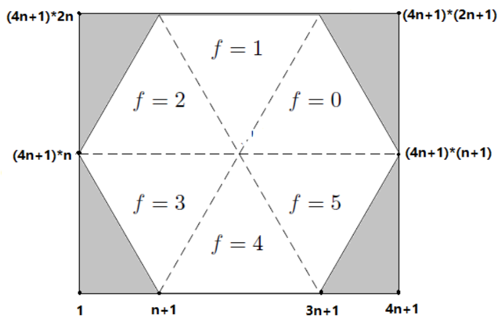
为了处理这种情况，同时使点的排列较为简单，我们引入虚拟节点，即对正六边形区域的四个角补上虚拟的节点，将区域补成一个标准的矩形区域，在这个矩形区域内做矩形网格剖分，就相对比较简单。

由于新的矩形区域，其长为：$l = \frac{4\sqrt{3}}{3}$，其宽度为：$w = 2$, 因此我们考虑的单位网格，长和宽不相同。假设在 y 方向，剖分网格数为 n，则单位网格竖直方向长度 $dy = \frac{2}{n}$，为了使在矩形内部与原正六边形重合部分都有单位网格点，结合图形的几何信息可知，单位网格水平方向长度满足：$\frac{dy}{dx} = \sqrt{3}$，即 $dx = \frac{dy}{\sqrt{3}} = \frac{2\sqrt{3}}{3n}$ 所以，水平方向上的网格数：$m = \frac{l}{dx} = 2n$。

最后，需要注意，为了使现在的矩形与原来正六边形端点重合的点，在剖分后的单位网格节点上，$\frac{1}{dy} = \frac{n}{2}$ 应该为整数，这样，n 需要被 2 能整除。因此，综合上面的分析，我们的网格剖分，如下;

- 水平方向网格数：$m = 4n$，竖直方向网格数：$k = 2n$，$n \in N$;
- 水平方向单位网格长度：$dx = \frac{\sqrt{3}}{3n}$，竖直方向单位网格长度：$\frac{1}{n}$;
- 节点总数:$m \times n = (4n + 1) \times (2n + 1)$，并且我们按从左到右，从下到上的顺序对节点进行标号，即从下边界开始，第一行从左忘右：$1, 2, 3, \cdots, 4n + 1$，进行标号；上面第二行，从左往右：$4n + 2, 4n + 3, \cdots, 8n + 2$, 进行标号；依次类推，对于第 k 行，从左往右：$(4n + 1) * \ k - 1 \ + 1, (4n + 1) * \ k - 1) + 2, \cdots, (4n + 1) * k$，$k = 1, 2, \cdots, 2n + 1$。

  所以，任意第 i 行 j 列的节点(i,j)其标号为：$(4n+1)*(i-1)+j, 1 \le i \le 2n+1$，$1 \le j \le 4n+1$，简单的图形表示，如下：



图中，阴影部分表示引入的虚拟节点。这样，对于边值条件的计算，网格的剖分以及节点的排列已经处理完成，下面对节点进行具体的差分处理。

2 差分格式

2.1 虚拟节点的处理

我们引入虚拟节点，使节点的排序变得方便，但是这样网格节点数目却增加了。因此，在差分处理时，我们根据排序信息，较容易的确定该点是引入的虚拟节点，还是在正六边形内部的或边界上，因为我们真正关系的点的信息，是在正六边形内部或边界上。

例如，对与第 k 行，$0 <= k < n$ 时，第 $k*(4n+1)+1$ 个节点，到 $k*(4n+1)+n-k$ 个节点，以及 $k*(4n+1)+3n+1+k$ 到 $(k+1)*(4n+1)$ 都是虚拟节点。这样，在节点差分时，不考虑这些节点，即不对这些节点做任何操作。

2.2 正六边形端点的处理

由于，题目所给的边界条件是 Neumman 边值，对于正六边形的端点，其不具有连续性，因此会存在两种可能的边值情况，因此我们做了如下约定：

- 对于下边界上两端点，其外法向方向与下边界的外法向方向相同，即满足下边界边界条件；
- 对于上边界上两端点，其外法相方向与上边界的外法向方向相同，即满足上边界边界条件；
- 对于左端点，其外法向方向，平行 x 轴向左，即指向 x 轴反方向。其边界条件满足：$\frac{\partial u_{i,j}}{\partial n} = (u_{i,j} - u_{i,j+1})/dx = c,\ i = n,\ j = 1$
- 对于右端点，其外法向方向，平行 x 轴向右，即指向 x 轴正方向。其边界条件满足：$\frac{\partial u_{i,j}}{\partial n} = (u_{i,j} - u_{i,j-1})/dx = c,\ i = n,\ j = 4n+1$。

2.3 正六边形边界点的差分

参考图形1.2，我们从 $f = 5$ 区域的边界开始，逆时针方向将边界标记为：$1, 2, 3, 4, 5, 6$。已知:

$$\frac{\partial u}{\partial n} = \nabla u \cdot \vec{n} = (\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}) \cdot (n_1, n_2) \tag{3}$$

其中，∇u 为 u 的梯度，$\vec{n}$ 为单位外法向。则有：

- 对于 1 号边界，$\vec{n} = (\frac{\sqrt{3}}{2}, -\frac{1}{2})$，$\nabla u = (\frac{u_{i,j}-u_{i,j-1}}{dx}, \frac{u_{i+1,j}-u_{i,j}}{dy})$，因此，由(3)，其边界条件满足：
$$\frac{\sqrt{3}}{2}\frac{u_{i,j} - u_{i,j-1}}{dx} + \frac{1}{2}\frac{u_{i,j} - u_{i+1,j}}{dy} = c;\ 1 \le i \le n-1,\ j = 3*n+i$$
- 对于 2 号边界，$\vec{n} = (\frac{\sqrt{3}}{2}, \frac{1}{2})$，$\nabla u = (\frac{u_{i,j}-u_{i,j-1}}{dx}, \frac{u_{i,j}-u_{i-1,j}}{dy})$，因此，由(3)，其边界条件满足：
$$\frac{\sqrt{3}}{2}\frac{u_{i,j} - u_{i,j-1}}{dx} + \frac{1}{2}\frac{u_{i,j} - u_{i-1,j}}{dy} = c;\ n+1 \le i \le 2n-1,\ j = 5n-i;$$
- 对于 3 号边界，$\vec{n} = (0, 1)$，$\nabla u = (0, \frac{u_{i,j}-u_{i-1,j}}{dy})$，因此，由(3)，其边界条件满足：
$$\frac{u_{i,j} - u_{i-1,j}}{dy} = c;\ i = 2n,\ n+1 \le j \le 3n+1;$$

- 对于 4 号边界，$\vec{n}=(-\frac{\sqrt{3}}{2},\frac{1}{2})$，$\nabla u=(\frac{u_{i,j+1}-u_{i,j}}{dx},\frac{u_{i,j}-u_{i-1,j}}{dy})$，因此，由(3)，其边界条件满足：
$$\frac{\sqrt{3}}{2}\frac{u_{i,j}-u_{i,j+1}}{dx}+\frac{1}{2}\frac{u_{i,j}-u_{i-1,j}}{dy}=c;\ 1+n\le i\le 2n-1,\ j=i-n;$$
- 对于 5 号边界，$\vec{n}=(\frac{-\sqrt{3}}{2},-\frac{1}{2})$，$\nabla u=(\frac{u_{i,j+1}-u_{i,j}}{dx},\frac{u_{i+1,j}-u_{i,j}}{dy})$，因此，由(3)，其边界条件满足：
$$\frac{\sqrt{3}}{2}\frac{u_{i,j}-u_{i,j+1}}{dx}+\frac{1}{2}\frac{u_{i,j}-u_{i+1,j}}{dy}=c;\ 1\le i\le n-1,\ j=n-i;$$
- 对于 6 号边界，$\vec{n}=(0,-1)$，$\nabla u=(0,\frac{u_{i+1,j}-u_{i,j}}{dy})$，因此，由(3)，其边界条件满足：
$$\frac{u_{i,j}-u_{i+1,j}}{dy}=c;\ i=0,\ n+1\le j=\le 3n+1$$

2.4 内部节点的差分

在上面，我们已经对原来正六边形边界上的点（包括端点）做了差分处理，现在剩下的就只有正六边形内部的节点。由于我们的网格剖分是矩形，显然，最简单直接的办法是采用二阶中心差分，例如对 i 行 j 列的节点 (i,j)，其差分格式为：

$$\Delta u_{i,j}=\frac{\partial^2 u}{\partial x^2}|_{i,j}+\frac{\partial^2 u}{\partial y^2}|_{i,j}=\frac{u_{i,j+1}+u_{i,j-1}-2u_{i,j}}{dx^2}+\frac{u_{i+1,j}+u_{i-1,j}-2u_{i,j}}{dy^2}=f \tag{4}$$

其中，(i,j) 满足:

- 当 $0<i<n$ 时，$n-i<j<3n+i$;
- 当 $i=n$ 时，$1<j<4n+1$;
- 当 $n<i<2n$ 时，$i-n<j<5n-i$;

2.5 泊松方程右端项处理

对于泊松方程 $\Delta u=f$，由于 f 在正六边形内是分片常数，如图：1.2。我们从 $f=0$ 开始逆时针方向，将 f 标记为：$f0,f1,f2,f3,f4,f5$，其中 fi 表示，$fi=i,i=0,1,2,3,4,5$。

而在处理图1.2中所示的虚线部分的 f 的值时，我们选择 f 相同的划分方式式，从右端水平虚线开始，逆时针方向记为：$0,1,2,3,4,5$ 号，其中，每个标号，代表了在其虚线上的点处，f 的取值。因此，我们有如下划分：

- 当 $0<i<n$，$n-i<j<3n+i$ 时为内部节点，其中，若 $n-i<j<n+i$，$f=f3$, 若 $3n-i\le j<3n+i$，$f=f5$，否则 $f=f4$;
- 当 $i=n$，$1<j<4n+1$ 时为中心水平虚线上点，其中，若 $1<j\le 2n+1$，$f=f3$，否则 $f=f0$;
- 当 $n<i<2n$，$i-n<j<5n-i$ 时，也为内部节点，其中，若 $i-n<j\le 3n-i$，$f=f2$，若 $3n-i<j\le n+i$，$f=f1$，否则 $f=f0$;

这样，对于任意内部节点 (i,j)，我们便可以知道，其对应的泊松方程右端项，f 的取值。

2.6 化简差分格式

由于我们选取的矩形网格剖分，因此单位网格的长和宽满足：$\frac{dy}{dx} = \sqrt{3}$ 和 $\frac{dy^2}{dx^2}$ 这样，对于上面正六边形关于边界点（包括端点）和内部点的差分格式，我们统一的将左端式子中分母的单位网格尺寸（dy 和 dx）放到等式的右端项中，这样便可以得到相对简单的差分表达式：

- 对于左端点：$u_{i,j} - u_{i,j+1} = dx * c,\ i = n, j = 1$; 对于右端点：$u_{i,j} - u_{i,j-1} = dx * c,\ i = n, j = 4n + 1$;
- 对于 1 号边界点：$4u_{i,j} - 3u_{i,j-1} - u_{i+1,j} = 2dy * c;\ 1 \le i \le n - 1,\ j = 3 * n + i$;
- 对于 2 号边界点：$4u_{i,j} - 3u_{i,j-1} - u_{i-1,j} = 2dy * c;\ n + 1 \le i \le 2n - 1,\ j = 5n - i$;
- 对于 3 号边界点：$u_{i,j} - u_{i-1,j} = dy * c,\ i = 2n, n + 1 \le j \le 3n + 1$;
- 对于 4 号边界点：$4u_{i,j} - 3u_{i,j+1} - u_{i-1,j} = 2dy * c,\ n + 1 \le i \le 2n - 1,\ j = i - n$;
- 对于 5 号边界点：$4u_{i,j} - 3u_{i,j+1-u_{i+1,j}} = 2dy * c,\ 1 \le i \le n - 1,\ j = n - i$;
- 对于 6 号边界点：$u_{i,j} - u_{i+1,j} = dy * c,\ i = 0,\ n + 1 \le j \le 3n + 1$;
- 对于内部节点：$8u_{i,j} - 3u_{i,j-1} - 3u_{i,j+1} - u_{i-1,j} - u_{i+1,j} = dy^2 * f$，其中，当 $0 < i < n$ 时，$n - i < j < 3n + i$；当 $i = n$ 时，$1 < j < 4n + 1$，当 $n < i < 2n$ 时，$i - n < j < 5n - i$。而关于 f 的取值，在上面右端项的处理中，已经明确的给出。

3 系数矩阵和右端项的存储

在上面，我们已经给出了具体的差分格式，即对于任意 (i,j) 点，能很容易也清楚的知道其满足的差分方程。因此，综合上面的信息，我们便可以将带 Neumman 边值条件的泊松方程，通过差分近似的方式，将求解原方程转化为求解线性方程组：$A0 * x = b$。其中，A0 对应通过差分格式得到的系数矩阵，x 对应为未知的 u，b 对应为右端项 f。

由于网格剖分一共有 $m * n$ 个节点，则对应的系数矩阵 A0 应该有 $m * n$ 行和 $m * n$ 列，x 和 b 也为 $m * n$ 的列向量。其中，x 和 b 的第 i 行，表示第 i 个节点的值和对应的右端项 f 的值。因此，相应的 A 的第 i 行表示第 i 个节点的差分方程的系数。$1 \le i <\le m * n$

观察上面的差分方程，可以发现，对于系数矩阵 A0 的每一行，最多只有 5 个非 0 元素。这样，为了节省内存，我们将系数采取稀疏存储的方式，用矩阵 A 表示，其中 A 为 $m * n$ 行，5 列的矩阵。需要注意的是，对于 A 的任意第 i 行，当 i 属于内部节点时：

- 矩阵 A 的第一列，A(i,1) 对应于第 i 个节点前第 i-m 个节点的系数；
- 矩阵 A 的第二列，A(i,2) 对应于第 i 个节点前第 i-1 个节点的系数；
- 矩阵 A 的第三列，A(i,3) 对应于第 i 个节点本身的系数；
- 矩阵 A 的第四列，A(i,4) 对应于第 i 个节点后第 i+1 个节点的系数；
- 矩阵 A 的第五列，A(i,5) 对应于第 i 个节点后第 i+m 个节点的系数；

即我们将矩阵 A 任意第 i 行每列的系数，与第 i 个节点周围空间位置的信息相联系，每一列都对应于第 i 个节点周围节点的空间位置。这样，在处理矩阵乘向量时，遵从通常矩阵乘向量的规则，即对应位置对应相乘即可，例如：原来线性方程组 A0*x=b 的第 i 行，等价于

$$A_{i,1} * x_{i-m} + A_{i,2} * x_{i-1} + A_{i,3} * x_i + A_{i,4} * x_{i+1} + A_{i,5} * x_{i+m} = b_i$$

而当 i 不是内部节点时，存在两种情况，i 节点在正六边形边界上，此时矩阵 A0 第 i 行的非 0 系数，小于 5，本质上也满足上面的对应关系，但需要特别注意和处理。例如：当对于第 i 个节点，若 $i-m$，$i-1$，$i+1$，$i+m$ 任意之一对应的节点位置，在正六边形之外时，我们令矩阵 A 第 i 行相对应的列系数为 0。这样，在处理矩阵 A 乘向量 x 时，也满足上面的对应式：

$$A_{i,1} * x_{i-m} + A_{i,2} * x_{i-1} + A_{i,3} * x_i + A_{i,4} * x_{i+1} + A_{i,5} * x_{i+m} = b_i$$

而若 (i-m) 节点不在正六边形内，相应项:$A(i,1) * x(i-m) = 0$，即在第 i 行的方程中不存在该项。综上所述，采取稀疏存储以及矩阵 A 任意 i 行 j 列，与第 i 个节点，周围节点空间位置相对应的关系，使的存储空间减小，且保留了原来的矩阵 A0 有用的信息。因为，原来线性方程组 $A0 * x = b$ 的第 i 行，等于现在 A 的第 i 行与 x 对应位置相乘。而当第 i 节点不在正六边形内部时，相应的矩阵 A0 的第 i 行系数全部为 0，此时我们也设 A 的第 i 行系数为 0，这样在处理矩阵 A 与 x 对应位置相乘时，引入的虚拟节点并没有实际参与计算。

将上面所述的格式整理成矩阵形式，矩阵如下：

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i,1} & a_{i,2} & a_{i,3} & a_{i,4} & a_{i,5} \\ a_{i+1,1} & a_{i+1,2} & a_{i+1,3} & a_{i+1,4} & a_{i+1,5} \\ a_{i+2,1} & a_{i+2,2} & a_{i+2,3} & a_{i+2,4} & a_{i+2,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m*n,1} & a_{m*n,2} & a_{m*n,3} & a_{m*n,4} & a_{m*n,5} \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{m*n-1} \\ b_{m*n} \end{bmatrix}$$

其中，对于任意 i 节点：

- 若不在正六边形上，则 $a_{i,1} = a_{i,2} = a_{i,3} = a_{i,4} = a_{i,5} = 0$，
- 若其为正六边形左端点，$a_{i,3} = 1, a_{i,4} = -1$，若为右端点，$a_{i,3} = 1, a_{i,2} = -1$，矩阵第 i 行其余值为 0，
- 若其在正六边形下边界上，$a_{i,3} = 1, a_{i,5} = -1$，矩阵第 i 行其余的值为 0，
- 若其在正六边形上边界上，$a_{i,3} = 1, a_{i,1} = -1$, 矩阵第 i 行其余值为 0，
- 若其在正六边形左下边界上，$a_{i,3} = 4, a_{i,4} = -3, a_{i,5} = -1$，矩阵第 i 行其余的值为 0，
- 若其在正六边形左上边界上，$a_{i,3} = 4, a_{i,4} = -3\ a_{i,1} = -1$, 矩阵第 i 行其余值为 0，
- 若其在正六边形右下边界上，$a_{i,3} = 4, a_{i,2} = -3\ a_{i,5} = -1$，矩阵第 i 行其余的值为 0，

- 若其在正六边形右上边界上，$a_{i,3} = 4, a_{i,2} = -3\ a_{i,1} = -1$, 矩阵第 i 行其余值为 0，
- 若第 i 节点在正六边形内部，相应的 $a_{i,1} = -1, a_{i,2} = -3, a_{i,3} = 8, a_{i,4} = -3, a_{i,5} = -1$

4 并行处理

在并行求解线性方程组 $Ax = b$ 时，我们先观察上面的矩阵 A 的系数，发现对于矩阵 A，其任意第 i 行，行和为 0。这样，取一个 $m * n$ 维列向量 e，$e = (1, 1, \cdots, 1, 1)^T$，可以得到，$A * e = 0 * e$。则说明，$\lambda = 0$ 为矩阵 A 的一个特征值，对应特征向量为 e，说明矩阵 A 是奇异的，因此解不唯一。其实，直接分析原方程(1)也可以知道，假设 u 为方程(1)的解，则 $u + C$ 也是该方程的解，其中 C 为任意常数。由定理也可知，Neumman 边值的泊松方程，解不唯一。

因此，在使用迭代法求解时，为了能保证原来解的性质，同时又避免解不唯一，我们修改了矩阵 A 任意 i 行对角元的值，当 i 在正六边形区域内时，使 i 行为严格对角占优的，这样系数矩阵 A 不存在 0 特征值的情况，即矩阵 A 非奇异。

我们选择经典的 Jacobi 迭代法求解线性方程组，Jacobi 迭代主要思想为，对于任意第 k 步迭代，$k \geq 0$，有如下关系:

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

其中，x_k 为第 k 次迭代的值，D 为 A 的对角元组成的对角阵，L 为 A 的严格下三角矩阵，U 为 A 的严格上三角矩阵, 即 $A = D + L + U$。则对于第 k 次迭代的任意 i 行，有如下关系：

当节点 i 不在正六边形区域内时，$x_i^{k+1} = x_i^k = 0$，$k \geq 0$;

当节点 i 在正六边形区域内部时，

$$x_i^{k+1} = (-(A_{i,1} * x_{i-m}^k + A_{i,2} * x_{i-1}^k + A_{i,4} * x_{i+1}^k + A_{i,5} * x_{i+m}^k) + b_i)/A_{i,3}$$

- 因此，我们的并行思想是：

  * 假设我们的进程数为 size，则将 x 等分给每个进程去求解，即每次迭代中一个进程需要求解的 x 数量为：$m * n/k$；
  * 对于任意第 i 号进程，$0 \leq i \leq size - 1$，其初始位置和终止位置定义为：
    $begin(i) = m * n * i/size, end(i) = m * n * (i + 1)/size$，初始位置为 begin(i)，而终止位置为 end(i)-1；
  * 因此在信息传递时，由 Jacobi 迭代关系可知，满足如下：

```
if (i > 0)
MPI_Send(&U[begin[i]], m, MPI_DOUBLE, i - 1, 0, MPI_COMM_WORLD);
if (rank < size - 1)
MPI_Send(&U[end[i] -m ], m, MPI_DOUBLE, i + 1, 0, MPI_COMM_WORLD);
if (i < size - 1)
MPI_Recv(&U[begin[i+1]], m, MPI_DOUBLE, i + 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
if (i > 0)
MPI_Recv(&U[end[i-1] - m], m, MPI_DOUBLE, i - 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

即对于第 i 号进程，若 $0 \le i \le size$，则 i 号进程向 i-1 号进程传从 x(begin(i)) 到 x(begin(i)+m-1) 的值；向 i+1 号进程传，从 x(end(i)-m) 到 x(end(i)-1) 的值; 同时，接收 i-1 号进程传来的从 x(end(i-1)-m) 到 x(end(i-1)-1) 的值，以及 i+1 号进程传来的从 x(begin(i+1)) 到 x(begin(i+1)+m-1) 的值。

这样，在每次迭代后，进行一次信息传递，能保证任意第 i 号进程，能获得其进行下一次迭代所需要的所有值。

* 每个进程独立计算迭代一次的误差，我们的误差定义为前后两步 x 的差的二范数。即对于第 i 号进程，其误差计算为：

$$error(i) = \sqrt{\sum_{j=begin(i)}^{j=end(i)-1} (x^{k+1}(j) - x^k(j))^2}$$

然后将其传送给 0 号进程，由 0 号进程负责计算前后两次迭代，总的误差：

$$Error = \sum_{i=0}^{size-1} error(i)$$

之后将总的误差 Error，广播给所有进程，由每个进程独立判断，迭代是否终止。

* 判断迭代终止的条件为：$Error < eps$ 且 $k < Nmax$，其中 eps 为误差精度，Nmax 为最大迭代次数。

* 当迭代终止后，第 i 个进程，$i \neq 0$ 将 x(begin(i)) 到 x(end(i)-1) 传送给 0 号进程，由 0 号进程负责接收其他进程传递的 x 的部分信息，构成完整的解 x。如下：

```
if(i!=0)
MPI_Send(&U[begin[i]], end[i] - begin[i], MPI_DOUBLE,0,0,MPI_COMM_WORLD);
if(i==0)
for(int i = 1; i < size; i++)
MPI_Recv(&U[begin[i]], end[i] - begin[i], MPI_DOUBLE, i,0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

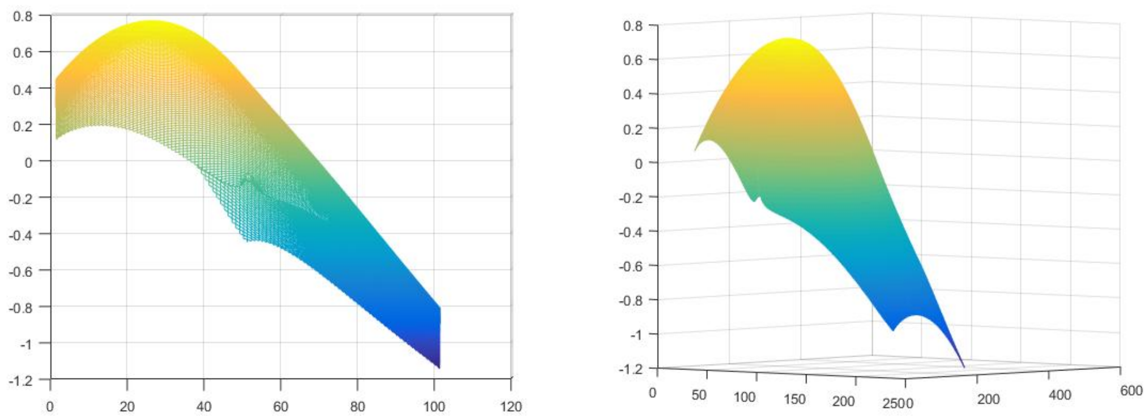
5 数值结果

我们取 $k = 100$ 时的情况，此时，横向网格数 $m = 401$，纵向网格数 $n = 201$，最大迭代次数为：$Kmax = 10^4$ 时;

进程数	误差	时间	相对加速比	效率
1	2.10784e-5	9.91s	1	1
2	2.10784e-51	5.25s	1.8876	0.9438
3	2.10784e-5	4.22s	2.3483	0.7827
4	2.10784-e	3.47s	2.8724	0.7181
5	2.10784e-5	3.99s	2.4837	0.4967
8	2.10784e-5	6.28s	1.5780	0.1972

从中可以看出，当进程数小于 4 的时候，相对于一个进程的加速比基本在 2 以上，且并行效率也保持在 0.7 以上，处于相对合理的范围内。但是，当进程数大于 5 之后，并行效率就大大降低，这是因为我用于计算的电脑最多只有 4 个核，当满核操作的时候，效率会大大降低，一般保证效率时，只需要用到电脑 $\frac{1}{2}$ 的核数。

下面将画出 $N = 100$ 和 $N = 50$ 时的计算结果图，此时的误差为：$Error = 1.5395e-5$



(a) $N = 50$ 时的计算结果

(b) $N = 100$ 时的计算结果

从上图5可以看出，这两幅图形状相似，只是从不同的角度观察的。发现当网格加密时，得到的结果更光滑，但是这样会增大计算量。以上两幅图都是 4 个进程时，取不同的网格数得到的结果。当取其他进程数时，得到的结果与上面结果相似，就不再一一列举了。