求解 Poisson 方程五点差分格式的快速算法以及数值比较实验

林元莘

2024 年 4 月 2 日

摘要

在几周的学习中，我们学习了如何运用五点差分格式求解 Poisson 方程。并且通过矩阵函数的变换，引出了快速 DST 算法来求解该方程。本次大作业旨在利用 MATLAB，对各个五点差分格式求解方法进行对比，探究 DST 算法的高效性，通过误差分析，了解五点差分格式的收敛阶。并且通过一定的变换，将该方法推广至非齐次边界条件以及一般矩形域。

关键词： Poission 方程；五点差分格式；快速 DST

1 实验目的

1、了解并求解 Poisson 方程五点差分格式。

2、进行数值实验，比较各种求解算法的精度以及速度。

3、探究 DST 方法的收敛性。

2 实验问题

考虑如下带 Dirchlet 边界条件的 Poisson 方程：

$$\begin{cases} -\Delta u = f, & \text{in } \Omega \\ u = \alpha, & \text{on } \partial\Omega \end{cases}$$

其中 $\Omega = (0, a) \times (0, b)$

2.1 五点差分格式

将区域 Ω 沿 x 方向和 y 方向分别进行 $I+1$ 和 $J+1$ 等分，则 x 方向和 y 方向的网格步长分别 $h=\frac{a}{I+1}, k=\frac{b}{J+1}$，而目标点为 $x_i=ih, y_j=jk, 0\le i\le I+1, 0\le j\le J+1$.

考虑 Laplace 算子的五点差分格式：

$$\Delta u(x_i,y_j)\approx\Delta_h u_{ij}=\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{k^2}$$

则该 Poission 方程的五点差分格式给定如下：

$$\begin{cases}-\Delta_h u_{ij}=f_{ij}, & \text{in } \Omega\\ u_{ij}=\alpha_{ij}, & \text{on } \partial\Omega\end{cases}$$

2.1.1 齐次 Dirchlet 边界条件

此时 $a\equiv 0$，设 $a=b=1$，则等价于求解如下矩阵方程：

$$AU+UB=F$$

其中：

$$A=\frac{1}{h^2}\begin{bmatrix}2&-1&&&\\-1&2&-1&&\\&\ddots&\ddots&\ddots&\\&&-1&2&-1\\&&&-1&2\end{bmatrix},\quad B=\frac{1}{k^2}\begin{bmatrix}2&-1&&&\\-1&2&-1&&\\&\ddots&\ddots&\ddots&\\&&-1&2&-1\\&&&-1&2\end{bmatrix}$$

$$U=\begin{bmatrix}u_{11}&u_{12}&\dots&u_{1J}\\u_{21}&u_{22}&\dots&u_{2J}\\\vdots&\vdots&\cdots&\vdots\\u_{I1}&u_{I2}&\dots&u_{IJ}\end{bmatrix},\quad F=\begin{bmatrix}f_{11}&f_{12}&\dots&f_{1J}\\f_{21}&f_{22}&\dots&f_{2J}\\\vdots&\vdots&\cdots&\vdots\\f_{I1}&f_{I2}&\dots&f_{IJ}\end{bmatrix}$$

2.1.2 非齐次 Dirchlet 边界条件

非齐次 Dirchlet 边界条件时，$\alpha=\alpha(x,y)$。求解关键在于处理边界情况下的五点差分，我们举 $i=1,j=1$ 的例子

$$f(x_1,y_1)=-\Delta u(x_1,y_1)\approx-\Delta_h u_{1,1}=\frac{-u_{2,1}+2u_{1,1}-u_{0,1}}{h^2}+\frac{-u_{1,2}+2u_{1,1}-u_{1,0}}{k^2}$$

而此时，我们有 $u_{0,1}=\alpha_{0,1}, u_{1,0}=\alpha_{1,0}$。则有：

$$\frac{2u_{1,1}-u_{2,1}}{h^2}+\frac{2u_{1,1}-u_{1,2}}{k^2}=f(x_1,y_1)+\frac{1}{h^2}\alpha_{0,1}+\frac{1}{k^2}\alpha_{1,0}$$

推广至矩阵情形如下：

$$AU+UB=\widetilde{F}$$

其中，

$$\widetilde{F}=F+\frac{1}{h^2}\begin{bmatrix}\alpha_{0,1} & \alpha_{0,2} & \dots & \alpha_{0,J-1} & \alpha_{0,J}\\ 0 & 0 & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \dots & 0 & 0\\ \alpha_{I+1,1} & \alpha_{I+1,2} & \dots & \alpha_{I+1,J-1} & \alpha_{I+1,J}\end{bmatrix}_{I\times J}+\frac{1}{k^2}\begin{bmatrix}\alpha_{1,0} & 0 & \dots & 0 & \alpha_{1,J+1}\\ \alpha_{2,0} & 0 & \dots & 0 & \alpha_{2,J+1}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \alpha_{I-1,0} & 0 & \dots & 0 & \alpha_{I-1,J+1}\\ \alpha_{I,0} & 0 & \dots & 0 & \alpha_{I,J+1}\end{bmatrix}_{I\times J}$$

2.1.3 一般矩形区域 $\Omega=[0,a]\times[0,b]$

设函数 $v(x,y)=u(ax,by)$, 划分数仍为 $I+1, J+1$ 则满足

$$\begin{cases}-\frac{1}{a^2}v_{xx}-\frac{1}{b^2}v_{yy}=-\Delta u(ax,ay)=f(ax,by),\text{in } [0,1]\times[0,1]\\ v(x,y)=\alpha(ax,by),\text{on } \partial([0,1]\times[0,1])\end{cases}$$

得到新的五点差分格式：

$$-\frac{1}{a^2}v_{xx}-\frac{1}{b^2}v_{yy}\approx\frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{(ah)^2}+\frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{(bk)^2}$$

此时，矩阵方程 $AU+BU=\widetilde{F}$ 中，

$$A=\frac{1}{(ah)^2}\begin{bmatrix}2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1\\ & & & -1 & 2\end{bmatrix},\quad B=\frac{1}{(bk)^2}\begin{bmatrix}2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1\\ & & & -1 & 2\end{bmatrix}$$

$\widetilde{F}$ 中，$f_{ij}=f(ax_i,by_j)$, $\alpha_{ij}=\alpha(ax_i,by_j)$

值得注意的是，在 DST 算法中，特征值 $\lambda=\frac{4}{(ah)^2}[\sin^2\frac{\pi}{2(I+1)},\sin^2\frac{2\pi}{2(I+1)},\dots,\sin^2\frac{I\pi}{2(I+1)}]$, $\mu=\frac{4}{(bk)^2}[\sin^2\frac{\pi}{2(J+1)},\sin^2\frac{2\pi}{2(J+1)},\dots,\sin^2\frac{J\pi}{2(J+1)}]$。

2.2 快速 DST 算法

Algorithm 1 求解齐次边界条件 Poission 方程五点差分格式的快速 DST 方法

输入： 划分数 $I+1, J+1$，步长 $h=\frac{1}{I+1}, k=\frac{1}{J+1}$，函数 $f(x,y)$.

输出： 解函数矩阵 $U=[u_{i,j}]$

步骤一： 形成向量 $\lambda=\frac{4}{h^2}[\sin^2\frac{\pi}{2(I+1)},\sin^2\frac{2\pi}{2(I+1)},\ldots,\sin^2\frac{I\pi}{2(I+1)}]$;

形成向量 $\mu=\frac{4}{k^2}[\sin^2\frac{\pi}{2(J+1)},\sin^2\frac{2\pi}{2(J+1)},\ldots,\sin^2\frac{J\pi}{2(J+1)}]$;

形成正弦变换矩阵 $P=[sin(\frac{ij\pi}{I+1})]\in\mathbb{R}^{I\times I}, Q=[sin(\frac{ij\pi}{J+1})]\in\mathbb{R}^{J\times J}$;

形成函数矩阵 $F=[f(x_i,y_j)]\in\mathbb{R}^{I\times J}$;

步骤二： 计算 $V=PFQ$;

步骤三： 计算 $W=[w_{ij}]$, $w_{ij}=\frac{4hkv_{ij}}{\lambda_i+\mu_j}$;

步骤四： 计算 $U=PWQ$

3 实验结果

3.1 齐次边界条件情形

3.1.1 实验函数

我们考虑使用函数 $u(x,y)=(1-x)(1-y)\sin(2\pi xy)$ 作为实验函数，其满足性质 $u(0,y)=u(1,y)=u(x,0)=u(x,1)\equiv 0$，图像如下：

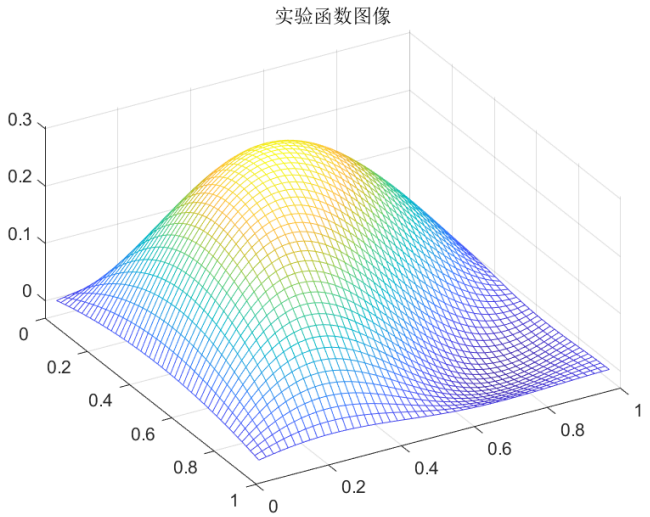


图 1: 实验函数

考虑其对应的 Dirchlet 边界条件的 Poission 方程求解问题：

$$\begin{cases} -\Delta u(x,y) = 4\pi^2(x^2+y^2)(1-x)(1-y)\sin(2\pi xy) + 4\pi(x+y-x^2-y^2)\cos(2\pi xy), & \text{in } \Omega \\ u(x,y) = 0, \quad \text{on } \partial\Omega \end{cases}$$

其中 $\Omega = (0,1) \times (0,1)$

3.1.2 五点差分格式算法对比

对方程使用五点差分格式，并使用快速 DST 算法、LU 分解算法（追赶法）、矩阵 Jacobi 迭代法、高斯-赛德尔迭代法、数列 Jacobi 迭代法进行求解，得到如下相同网格点数下计算时间与相对误差对比图：

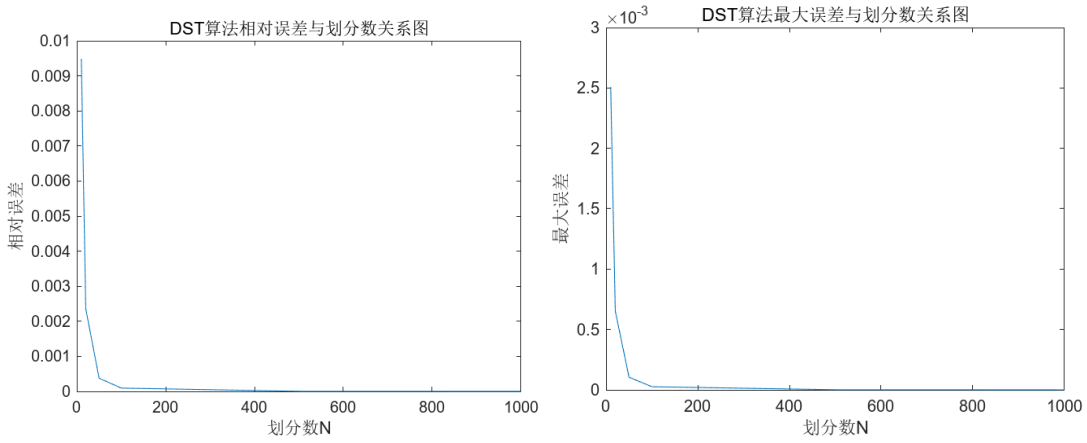
划分数	DST/s	分块 LU 分解/s	矩阵 Jacobi 迭代/s	Gauss-Seidel 迭代/s	Jacobi 迭代/s
10	0.000262	0.000260	0.000595	0.000317	0.004608
50	0.000400	0.002665	0.149696	0.048397	0.108123
100	0.001241	0.027567	1.885147	0.714911	1.322782
200	0.003254	0.279427	31.146315	15.629820	27.477940
300	0.012421	1.137324	250.671350	84.512536	152.722085

表 1: 不同算法的划分数-计算用时对比

显然，快速 DST 在多网格点计算上，速度远远优于其他算法。

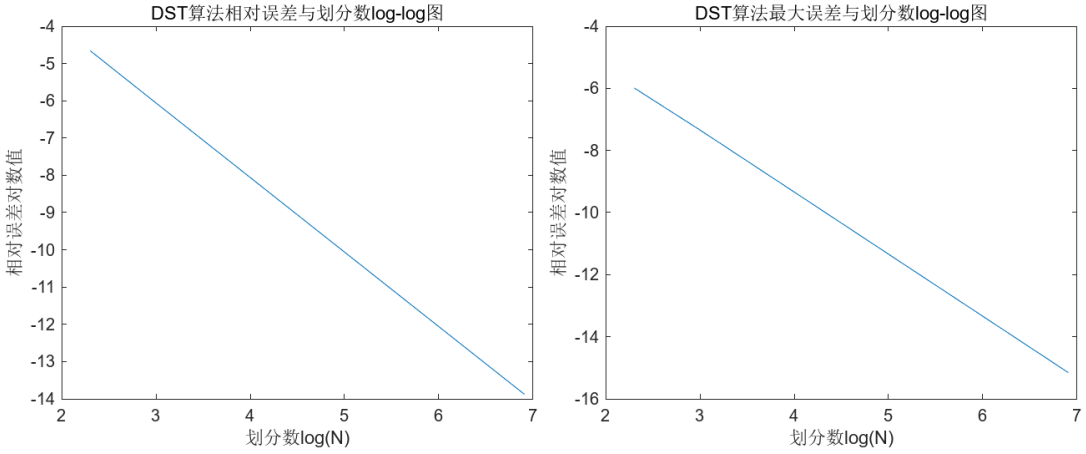
3.1.3 快速 DST 算法误差收敛分析

使用快速 DST 算法求解非齐次 Dirchlet 边界条件 Poission 方程，误差与划分数的对比图如下：



(a) (b)

显然误差下降速度很快，我们考考虑使用 log-log 图观察收敛阶



(a) (b)

对 log-log 图做线性回归, 可以得到 $k_1 = -2.001298506975118, k_2 = -1.993979792790152$ 可以得到快速 DST 算法的收敛阶为 $O(N^{-2})$

3.2 非齐次边界条件情形

3.2.1 实验函数

我们考虑使用函数 $u(x, y) = xye^{x^2+y^2}$ 作为实验函数，其对应的 Dirchlet 边界条件的 Poission 方程求解问题为:

$$\begin{cases} -\Delta u(x, y) = -4xy(3 + x^2 + y^2)e^{x^2+y^2}, & \text{in } \Omega \\ u(x, y) = xye^{x^2+y^2}, & \text{on } \partial\Omega \end{cases}$$

其中 $\Omega = (0,1) \times (0,1)$，图像如下

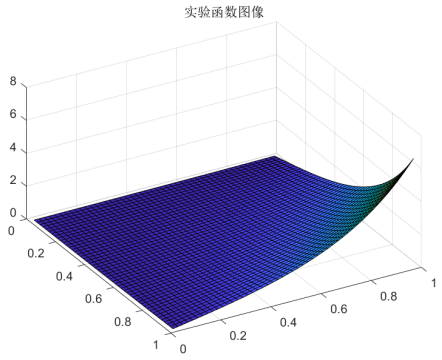


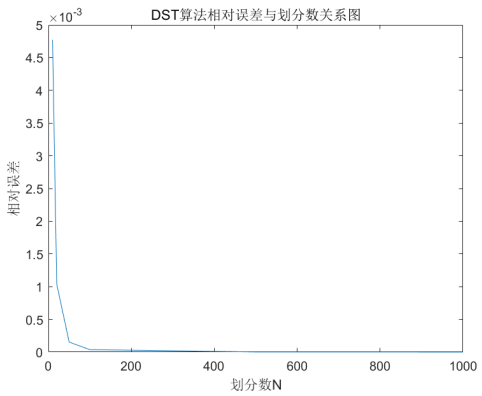
图 4: 实验函数

3.2.2 快速 DST 算法误差收敛分析

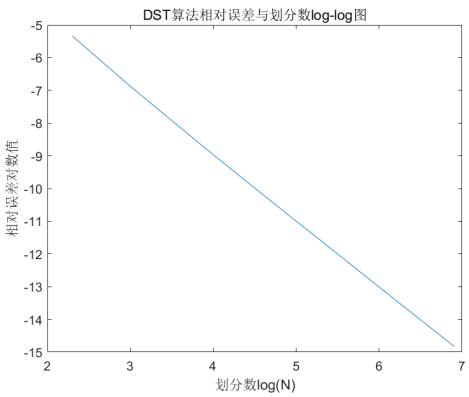
划分数	误差	用时/s
10	0.004772453938975	0.000342
20	0.001040832266306	0.000314
50	1.528864234884704e-04	0.000508
100	3.712796738389100e-05	0.000763
500	1.450758862575780e-06	0.035667
1000	3.616259034571202e-07	0.244182

表 2: 快速 DST 算法的划分数-误差-计算用时对比

同样的，画出划分数-相对误差比较图以及其 log-log 图



(a)



(b)

收敛阶仍然为 $O(N^{-2})$

3.2.3 一般矩形边界条件情形

3.2.4 实验函数

仍然使用 **3.2.1** 中的实验函数，即

$$\begin{cases} -\Delta u(x,y) = -4xy(3+x^2+y^2)e^{x^2+y^2}, & \text{in } \Omega \\ u(x,y) = xye^{x^2+y^2}, & \text{on } \partial\Omega \end{cases}$$

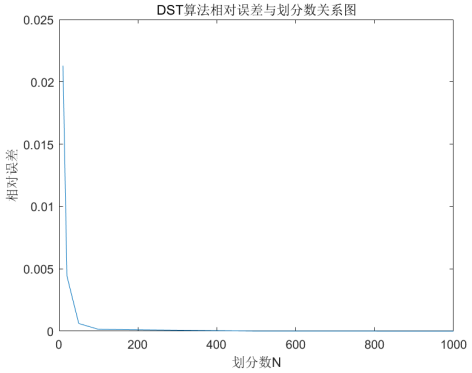
其中 $\Omega = (0,a)\times(0,b)$ 此时，我们令 $a=3, b=5$ 进行实验。

3.2.5 快速 DST 算法误差收敛分析

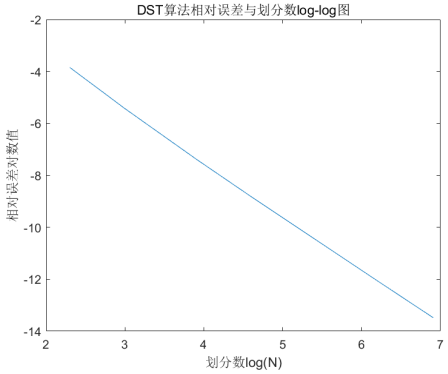
划分数	误差	用时/s
10	0.021292169995370	0.000369
20	0.004416724699001	0.000303
50	6.141542791538716e-04	0.000606
100	1.460330877354940e-04	0.000739
500	5.606570385507676e-06	0.029691
1000	1.394399612941948e-06	0.256950

表 3: 快速 DST 算法的划分数-误差-计算用时对比

这是由于原函数的特性，函数值变化幅度较大，所以误差较前面的结果稍大一些。但通过画出划分数-相对误差对比图以及 log-log 图可以得知，误差收敛阶仍然为 $O(N^{-2})$:



(a)



(b)

4 结论

- 五点差分格式的解法多种多样，其中，快速 DST 算法是远远快于其他算法。
- 五点差分格式收敛速度较快，收敛阶为 $O(N^{-2})$。
- 通过代数变换，可以将单位正方形域齐次 Dirchlet 边界条件的 Poission 方程求解方法，推广至非齐次边界条件以及一般矩形域，并且不影响收敛速度。

5 附录

A 附录 A：MATLAB 代码

在这个附录中，我们提供了完成该大作业所用的代码。

Fun_f.m

```
function y=Funf(x,y)
y=+4*pi*pi*(x.^2+y.^2).*(1-x).*(1-y).*sin(2*pi*x.*y)+...
4*pi*(x+y-x.^2-y.^2).*cos(2*pi*x.*y);
```

test1.m

```
format long


for N=[10,50,100,200,300]
tic
U_dst=DSTPS(N);
toc
tic
U_lu = BlockLU(N);
toc


tic
U_I = IPS(N);
toc
```

```
tic
U_GS = PSGS(N);
toc
U_GS = U_GS(2:N,2:N);
tic
U_J = PSJacobi(N);
toc
U_J = U_J(2:N,2:N);

h=1/N;x=h:h:1-h;tes
[X,Y]=meshgrid(x,x);
Uexact=(1-X).*(1-Y).*sin(2*pi*X.*Y)
E_dst=norm(U_dst-Uexact,'fro')/norm(Uexact,'fro')
EM_dst =max(max(U_dst -Uexact))
E_lu = norm(U_lu-Uexact,'fro')/norm(Uexact,'fro')
E_I= norm(U_I-Uexact,'fro')/norm(Uexact,'fro')
E_GS = norm(U_GS-Uexact,'fro')/norm(Uexact,'fro')
E_J= norm(U_J-Uexact,'fro')/norm(Uexact,'fro')
end
```

test2.m

```
format long
E_dst = zeros(1,6)
EM_dst = zeros(1,6)
N=[10,20,50,100,500,1000]
for i = 1:6
tic
U_dst=DSTPS(N(i));
toc
h=1/N(i);x=h:h:1-h;
```

```
[X,Y]=meshgrid(x,x);
Uexact=(1-X).*(1-Y).*sin(2*pi*X.*Y);
E_dst(i)=norm(U_dst-Uexact,'fro')/norm(Uexact,'fro')
EM_dst(i) =max(max(abs(U_dst -Uexact)))
end

h=1/50;x=h:h:1-h;
[X,Y]=meshgrid(x,x);
Uexact=(1-X).*(1-Y).*sin(2*pi*X.*Y);
mesh(X,Y,Uexact)
view(60,45)
title('实验函数图像')

plot(N,EM_dst)
title('DST算法最大误差与划分数关系图')
xlabel('划分数N')
ylabel('最大误差')

plot(N,E_dst)
title('DST算法相对误差与划分数关系图')
xlabel('划分数N')
ylabel('相对误差')
plot(log(N),log(EM_dst))
title('DST算法最大误差与划分数log-log图')
xlabel('划分数log(N)')
ylabel('相对误差对数值')
plot(log(N),log(E_dst))
title('DST算法相对误差与划分数log-log图')
xlabel('划分数log(N)')
ylabel('相对误差对数值')
```

```
p = polyfit(log(N),log(E_dst),1)
pM = polyfit(log(N),log(EM_dst),1)
```

Funf_q2.m

```
function z=Funf_q2(x,y)
    z= -4*x.*y.*(3+x.^2+y.^2).*exp(x.^2+y.^2);
```

Bound.m

```
function z = Bound(x,y)
z = x.*y.*exp(x.^2+y.^2);
```

DSTPS_q2.m

```
function U = DSTPS_q2(N)

% 快速正弦变换求解Poisson方程,求解区域[0,1]^2
% 输入:划分数 N
% 输出:数值解 U(矩阵形式)

h=1/N;
x=h*(1:N-1);
[X,Y]=meshgrid(x,x);
Boundry = zeros(N-1,N-1);
Boundry(1,:) = N^2*Bound(0,x);
Boundry(N-1,:) = N^2*Bound(1,x);
Boundry(:,1) = N^2*Bound(x',0) + Boundry(:,1);
Boundry(:,N-1) = N^2*Bound(x',1) + Boundry(:,N-1);
F=Funf_q2(X,Y);
F=F'+Boundry;
lambda=sin(pi*x/2);
lambda=4*lambda.*lambda;

T=dst(dst(F)')';
```

```
V=zeros(N-1);
for i=1:N-1
    for j=1:N-1
        V(i,j)=T(i,j)/(lambda(i)+lambda(j));
    end
end
V=4*h^4*V;

U=dst(dst(V)')';
```

test_q2.m

```
format long
E_dst = zeros(1,6)
EM_dst = zeros(1,6)
N=[10,20,50,100,500,1000]
for i = 1:6
tic
U_dst=DSTPS_q2(N(i))
toc
h=1/N(i);x=h:h:1-h;
[X,Y]=meshgrid(x,x);
Uexact=X.*Y.*exp(X.^2+Y.^2)
E_dst(i)=norm(U_dst-Uexact,'fro')/norm(Uexact,'fro')
EM_dst(i) =max(max(abs(U_dst -Uexact)))
end
h=1/50;x=h:h:1-h;
[X,Y]=meshgrid(x,x);
Uexact=X.*Y.*exp(X.^2+Y.^2);
surf(X,Y,Uexact)
view(60,45)
title('实验函数图像')
```

```
plot(N,EM_dst)
title('DST算法最大误差与划分数关系图')
xlabel('划分数N')
ylabel('最大误差')

plot(N,E_dst)
title('DST算法相对误差与划分数关系图')
xlabel('划分数N')
ylabel('相对误差')
plot(log(N),log(EM_dst))
title('DST算法最大误差与划分数log-log图')
xlabel('划分数log(N)')
ylabel('相对误差对数值')
plot(log(N),log(E_dst))
title('DST算法相对误差与划分数log-log图')
xlabel('划分数log(N)')
ylabel('相对误差对数值')

p = polyfit(log(N),log(E_dst),1)
pM = polyfit(log(N),log(EM_dst),1)
```

DSTPS_q3.m

```
function U=DSTPS_q3(N1,N2,a,b)
% 快速正弦变换求解Poisson方程,求解区域[0,1]^2
% 输入:x方向划分数 N1,y方向划分数 N2
% 输出:数值解 U(矩阵形式)

h1=1/N1;
h2=1/N2;
x=h1*(1:N1-1);
y=h2*(1:N2-1);
```

```
[X,Y]=meshgrid(x,y);
Boundry = zeros(N1-1,N2-1);
Boundry(1,:) = (N1/a)^2*Bound(0,b*y);
Boundry(N1-1,:) = (N1/a)^2*Bound(a,b*y);
Boundry(:,1) = (N2/b)^2*Bound((a*x)',0) + Boundry(:,1);
Boundry(:,N2-1) = (N2/b)^2*Bound((a*x)',b) + Boundry(:,N2-1);
F=Funf_q2(a*X,b*Y);F=F'+Boundry ;
lambda=sin(pi*x/2);
lambda=4*lambda.*lambda/(a*h1)/(a*h1);
mu=sin(pi*y/2);
mu=4*mu.*mu/(b*h2)/(b*h2);

T=dst(dst(F)')';

V=zeros(N1-1,N2-1);
for i=1:N1-1
    for j=1:N2-1
        V(i,j)=T(i,j)/(lambda(i)+mu(j));
    end
end
V=4*h1*h2*V;

U=dst(dst(V)')';
```

test_q3.m

```
format long
E_dst = zeros(1,6)
EM_dst = zeros(1,6)
N=[10,20,50,100,500,1000]
for i = 1:6
tic
```

```
U_dst=DSTPS_q3(N(i),N(i),2,1/2)
toc
h=2/N(i);x=h:h:2-h;
k = 1/2/N(i);y=k:k:1/2-k
[X,Y]=meshgrid(x,y);
Uexact=X.*Y.*exp(X.^2+Y.^2)
E_dst(i)=norm(U_dst-Uexact','fro')/norm(Uexact','fro')
EM_dst(i) =max(max(abs(U_dst -Uexact')))
U_dst-Uexact'
end

plot(N,EM_dst)
title('DST算法最大误差与划分数关系图')
xlabel('划分数N')
ylabel('最大误差')

plot(N,E_dst)
title('DST算法相对误差与划分数关系图')
xlabel('划分数N')
ylabel('相对误差')
plot(log(N),log(EM_dst))
title('DST算法最大误差与划分数log-log图')
xlabel('划分数log(N)')
ylabel('相对误差对数值')
plot(log(N),log(E_dst))
title('DST算法相对误差与划分数log-log图')
xlabel('划分数log(N)')
ylabel('相对误差对数值')
```