武汉大学数学与统计学院

微分方程数值解法实验报告

实验名称	Homework 1			实验时间	2020 年　3 月　15 日	
姓名	江金阳	班级	17 信计班	学号	2017301000090	成绩

1　问题一

1.1 理论推导

将网格取为 $x_i = ih,\ i=1,2,\dots,n,\ h=\frac{1}{n}$，并定义 $x_{i+\frac{1}{2}} = x_i + h/2$

对自伴微分方程 $(\beta(x)u')' - \gamma(x)u(x) = f(x)$ 的第一项在 x_i 处使用中心差分格式离散得到

$$\frac{\beta_{i+\frac{1}{2}}u'(x_{i+\frac{1}{2}}) - \beta_{i-\frac{1}{2}}u'(x_{i-\frac{1}{2}})}{h^2} - \gamma_i u(x_i) = f_i + E_i^1$$

其中 $\beta_{i+1/2} = \beta(x_{i+1/2}),\ \gamma_i = \gamma(x_i),\ f_i = f(x_i),\ E_i^1 = Ch^2$　再对 u' 分别在 $x_{i-1/2}$ 和 $x_{i+1/2}$ 处使用中心差分格式离散，得到

$$\frac{\beta_{i+\frac{1}{2}}\left(u(x_{i+1}) - u(x_i)\right) - \beta_{i-\frac{1}{2}}\left(u(x_i) - u(x_{i-1})\right)}{h^2} - \gamma_i u(x_i) = f_i + E_i^1 + E_i^2,\ i=1,2,\dots,n$$

令 $U_i \approx u(x_i)$ 得到相应的线性方程组为

$$\frac{\beta_{i+\frac{1}{2}}\left(U_{i+1} - U_i\right) - \beta_{i-\frac{1}{2}}\left(U_i - U_{i-1}\right)}{h^2} - \gamma_i U_i = f_i,\ i=1,2,\dots,n$$

整理得到

$$\frac{\beta_{i+\frac{1}{2}}U_{i+1} + \beta_{i-\frac{1}{2}}U_{i-1}}{h^2} - \left(\frac{\beta_{i+\frac{1}{2}} + \beta_{i-\frac{1}{2}}}{h^2} + \gamma_i\right)U_i = f_i,\ i=1,2,\dots,n$$

在 $x=h$ 处引入 Dirichlet 边界条件得到

$$-\left(\frac{\beta_{\frac{3}{2}} + \beta_{\frac{1}{2}}}{h^2} + \gamma_1\right)U_1 + \frac{\beta_{\frac{3}{2}}}{h^2}U_2 = f_1 - \frac{\beta_{\frac{1}{2}}}{h^2}u_a$$

对 $x=1$ 处的 Robin 边界条件引入鬼点，使用中心差分格式离散得到

$$aU_n + b\frac{U_{n+1} - U_{n-1}}{2h} = c \ \Rightarrow\ U_{n+1} = U_{n-1} + \frac{2h}{b}(c - aU_n)$$

再代入在 $x=1$ 处的离散格式得到

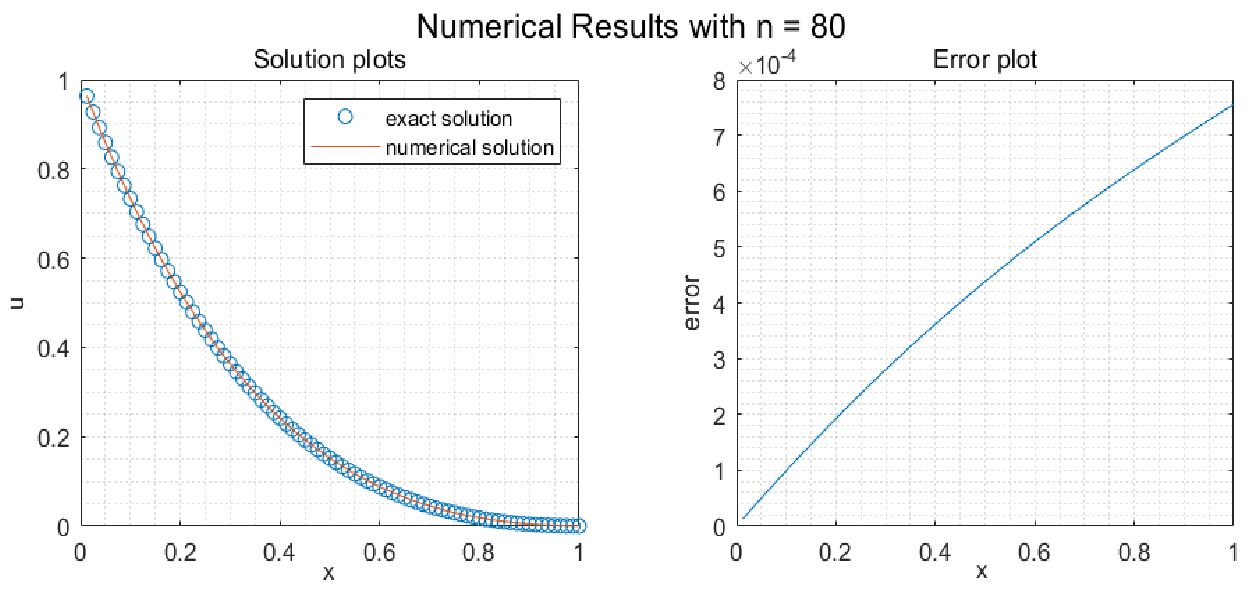
$$\frac{\beta_{n+\frac{1}{2}}+\beta_{n-\frac{1}{2}}}{h^2}U_{n-1}-\left[\frac{(1+2ha/b)\beta_{n+\frac{1}{2}}+\beta_{n-\frac{1}{2}}}{h^2}+\gamma_n\right]U_n=f_n-\frac{\beta_{n+\frac{1}{2}}}{h^2}\cdot\frac{2hc}{b}$$

将以上结果组装为矩阵形式如下

$$\begin{bmatrix} -\left(\frac{\beta_{\frac{3}{2}}+\beta_{\frac{1}{2}}}{h^2}+\gamma_1\right) & \frac{\beta_{\frac{3}{2}}}{h^2} & & & \\ \frac{\beta_{\frac{3}{2}}}{h^2} & -\left(\frac{\beta_{\frac{5}{2}}+\beta_{\frac{3}{2}}}{h^2}+\gamma_2\right) & \frac{\beta_{\frac{5}{2}}}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{\beta_{n-\frac{3}{2}}}{h^2} & -\left(\frac{\beta_{n-\frac{1}{2}}+\beta_{n-\frac{3}{2}}}{h^2}+\gamma_{n-1}\right) & \frac{\beta_{n-\frac{5}{2}}}{h^2} \\ & & & \frac{\beta_{n+\frac{1}{2}}+\beta_{n-\frac{1}{2}}}{h^2} & -\left[\frac{\left(1+\frac{2ha}{b}\right)\beta_{n+\frac{1}{2}}+\beta_{n-\frac{1}{2}}}{h^2}+\gamma_n\right] \end{bmatrix}\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{n-1} \\ U_n \end{bmatrix}=\begin{bmatrix} f_1-\frac{\beta_{\frac{1}{2}}}{h^2}u_a \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n-\frac{\beta_{n+\frac{1}{2}}}{h}\cdot\frac{2c}{b} \end{bmatrix}$$

1.2 数值结果

计算中取 $\beta(x)=1+x^2$，$\gamma(x)=x$，$a=2$，$b=-3$，真实解 $u(x)=e^{-x}(x-1)^2$，在网格 $n=80$ 下计算得到如下的数值结果和误差图像。



1.3 网格细化分析

1.3.1 理论分析

对 1.1 中推导的离散结果，当网格尺寸为 h 时，其在节点 x 处的误差有如下的表示

$$E(h)=\frac{\beta(x+\frac{h}{2})\frac{u(x+h)-u(x)}{h}-\beta(x-\frac{h}{2})\frac{u(x)-u(x-h)}{h}}{h}-(\beta(x)u'(x))'$$

将上式第一项各部分在 x 处 Taylor 展开有

$$\beta(x\pm\frac{h}{2})=\beta(x)\pm\frac{h}{2}\beta'(x)+\frac{h^2}{8}\beta''(x)+O(h^2)$$

$$\frac{u(x+h)-u(x)}{h}=u'(x)+\frac{h}{2}u''(x)+\frac{h^2}{6}u'''(x)+O(h^2)$$

$$\frac{u(x)-u(x-h)}{h}=u'(x)-\frac{h}{2}u''(x)+\frac{h^2}{6}u'''(x)+O(h^2)$$

将它们代入误差表达式可得

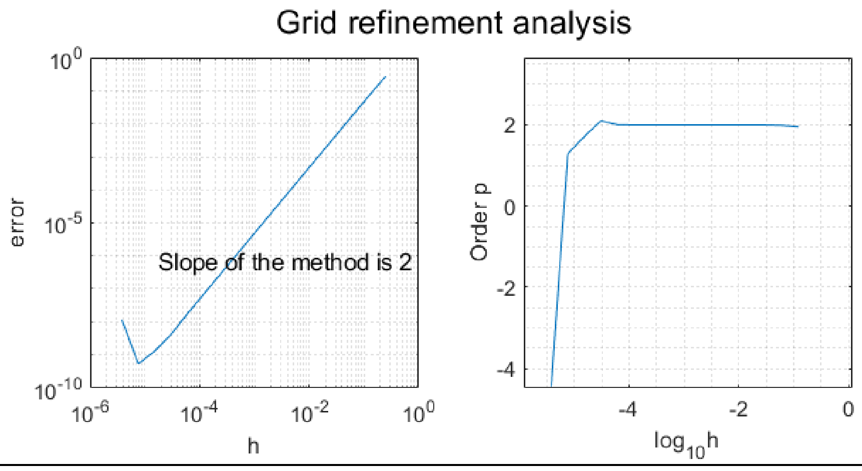
$$E(h)=\left[\frac{\beta(x)u^{(4)}(x)}{12}+\frac{\beta'(x)u'''(x)}{6}+\frac{\beta''(x)u''(x)}{8}+\frac{\beta'''(x)u'(x)}{24}\right]h^2+O(h^2)$$

故理论上格式应该是二阶的。

1.3.2 数值结果

将网格不断细化计算，并将误差向量分别取无穷范数得到各尺度网格下的全局误差 $\|E\|_\infty=\max_i\{e_i\}$.

又由于对格式精度的阶数有估计式 $p\approx\frac{\log(\|E_h\|/\|E_{h/2}\|)}{\log 2}$，最终计算得到下图中的数值结果。从左图的斜率和右图的 p 值可以观察到，此方法是二阶精度的。



1.4 附加问题

1.4.1 Robin 边界条件中$a=0$或$b=0$时代码的鲁棒性

① 当$a=0$时，Robin 边界条件退化为 Neumann 边界条件。不影响原代码的使用。

② 当$b=0$时，Robin 边界条件退化为 Dirichlet 边界条件。原离散格式中b在一些位置被用作分母，不能正常工作，此时需要将最后一个离散方程更换为$U_n=\dfrac{c}{a}$（见代码）.

③ 当$a=b=0$时，Robin 边界条件失效，不能考虑数值结果。

1.4.2 在$\beta u''+\beta' u'-\gamma u=f$中采用中心差分格式的优缺点

① 优点：在对流项数值结果不出现振荡的参数范围内，中心差分格式是二阶精度的。在相同的网格节点数下，采用中心差分格式的的数值结果要比采用一阶迎风格式的数值结果误差更小。

② 缺点：中心差分格式的系数矩阵A在$|\beta'|$较大时很可能不是对称的、负定的或对角占优的，此时中心差分格式得到的数值结果在实际的物理情景下很可能存在非物理振荡,如$|\beta'|\sim 1/h$时。

2 问题二

2.1 理论推导

将网格取为$t_i=ih,\ \ i=1,2,\ldots,n,\ \ h=\dfrac{2\pi}{n}$

对非线性微分方程$\dfrac{d^2\theta}{dt^2}+K\sin\theta=0$中的二次导数使用三点中心差分离散，得到的结果为

$$\frac{\Theta_{i+1}-2\Theta_i+\Theta_{i-1}}{h^2}+K\sin\Theta_i=0,\ \ i=1,2,\ldots,n-1$$

在$t=h$和$t=2\pi-h$处代入 Dirichlet 边界条件有

$$\frac{\Theta_2-2\Theta_1+\theta_1}{h^2}+K\sin\Theta_1=0,\ \ \frac{\theta_2-2\Theta_{n-1}+\Theta_{n-2}}{h^2}+K\sin\Theta_{n-1}=0$$

非线性方程组的 Jacobi 矩阵为

$$J\left(\Theta\right)=\begin{bmatrix} \frac{-2}{h^2}+K\cos\Theta_1 & \frac{1}{h^2} & & & \\ \frac{1}{h^2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{1}{h^2} \\ & & & \frac{1}{h^2} & \frac{-2}{h^2}+K\cos\Theta_{n-1} \end{bmatrix}$$

对线性微分方程 $\frac{d^2\theta}{dt^2}+K\theta=0$ 中的二次导数使用三点中心差分离散，得到的结果为

$$\frac{\Theta_{i+1}-2\Theta_i+\Theta_{i-1}}{h^2}+K\Theta_i=0,\ \ i=1,2,...,n-1$$

在 $t=h$ 和 $t=2\pi-h$ 处代入 Dirichlet 边界条件有

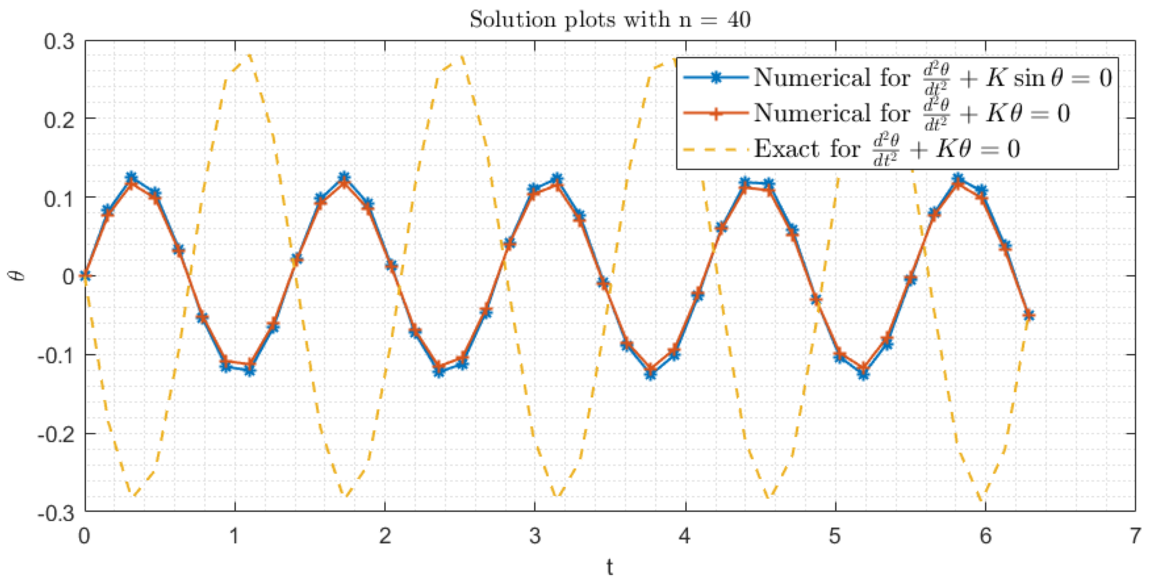
$$\frac{\Theta_2-2\Theta_1+\theta_1}{h^2}+K\Theta_1=0,\ \ \frac{\theta_2-2\Theta_{n-1}+\Theta_{n-2}}{h^2}+K\Theta_{n-1}=0$$

线性方程组的 Jacobi 矩阵为

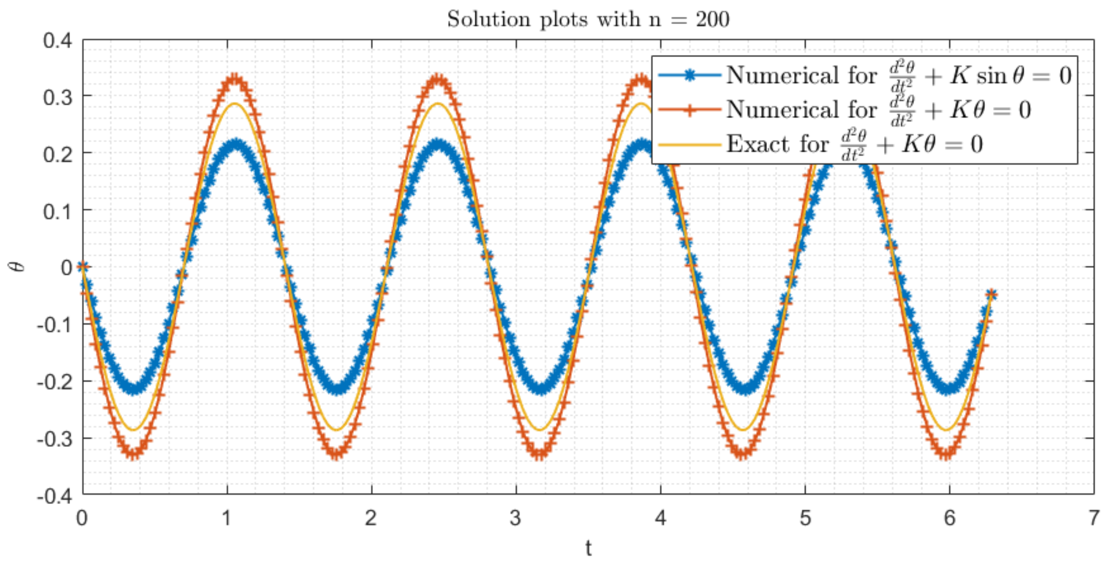
$$J\left(\Theta\right)=\begin{bmatrix} \frac{-2}{h^2}+K & \frac{1}{h^2} & & & \\ \frac{1}{h^2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{1}{h^2} \\ & & & \frac{1}{h^2} & \frac{-2}{h^2}+K \end{bmatrix}$$

2.2 数值结果

计算中将参数取为 $K=20,\ \theta_1=0,\ \theta_2=-0.05$，使用牛顿迭代法求解方程组，当网格取 $n=40$ 时有如下的数值结果，其中非线性方程 $\frac{d^2\theta}{dt^2}+K\sin\theta=0$ 与线性方程 $\frac{d^2\theta}{dt^2}+K\theta=0$ 的数值解结果非常接近，但都与 dsolve 命令直接求得的线性方程解析解结果相差半个相位，且振幅也不同。



经检验，两方程的求解中牛顿迭代法都是达到收敛准则停机的，故猜测误差是格式带来的，由于网格尺度过大导致结果与解析解相差较大，再次取网格为$n=200$得到如下结果，两方程的数值结果均与线性方程的解析结果相近，符合预期。



2.3 $\frac{d^2\theta}{dt^2}+K\sin\theta=0$和$\frac{d^2\theta}{dt^2}+K\theta=0$的数值结果比较

由数值结果图像观察发现，非线性方程与线性方程都表现出类似三角函数的周期特点，两者具有相同的相位，但由于第二项中$\sin\theta\le\theta$，$\theta\in[0,2\pi]$，非线性方程解的振幅略小于线性方程。

3 问题三

3.1 理论推导

将网格取为 $x_i = a + ih_x,\ \ i = 1,2,\dots,n,\ \ h_x = \dfrac{b-a}{n};\ \ y_i = c + jh_y,\ \ j = 1,2,\dots,m,\ \ h_y = \dfrac{d-c}{m}$

对 $u_{xx} + p(x,y)u_{yy} + r(x,y)u(x,y) = f(x,y)$ 中的二次偏导数使用三点中心差分离散得到

$$\frac{u(x_{i+1},y_j) - 2u(x_i,y_j) + u(x_{i-1},y_j)}{h_x^2} + p_{i,j}\frac{u(x_i,y_{j+1}) - 2u(x_i,y_j) + u(x_i,y_{j-1})}{h_y^2} + r_{i,j}u(x_i,y_j) = f_{i,j} + T_{i,j}$$

其中 $p_{i,j} = p(x_i,y_j),\ \ r_{i,j} = r(x_i,y_j),\ \ f_{i,j} = f(x_i,y_j),\ \ T_{i,j} \sim O(h_x^2 + h_y^2)$

令 $U_{i,j} \approx u(x_i,y_j)$ 得到相应的线性方程组为

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h_x^2} + p_{i,j}\frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h_y^2} + r_{i,j}U_{i,j} = f_{i,j},\ \ i = 1,2,\dots,n,\ \ j = 1,2,\dots,m-1$$

整理后得到

$$\frac{U_{i+1,j} + U_{i-1,j}}{h_x^2} + p_{i,j}\frac{U_{i,j+1} + U_{i,j-1}}{h_y^2} + \left(\frac{-2}{h_x^2} + \frac{-2p_{i,j}}{h_y^2} + r_{i,j}\right)U_{i,j} = f_{i,j},\ \ i = 1,2,\dots,n,\ \ j = 1,2,\dots,m-1$$

将 $x = b$ 处的 Neumann 边界条件记为 $u_x(b,y) = \varphi(y)$，并在此处引入鬼点 $U_{n+1,j}$ 得到

$$\frac{U_{n+1,j} - U_{n-1,j}}{2h_x} = \varphi_j \ \Rightarrow\ U_{n+1,j} = 2h_x\varphi_j + U_{n-1,j},\ \ j = 1,2,\dots,m-1$$

将上式代入 $i = n$ 处的 $m-1$ 个方程有

$$\frac{2U_{n-1,j}}{h_x^2} + p_{n,j}\frac{U_{n,j+1} + U_{n,j-1}}{h_y^2} + \left(\frac{-2}{h_x^2} + \frac{-2p_{n,j}}{h_y^2} + r_{n,j}\right)U_{n,j} = f_{n,j} - \frac{2\varphi_j}{h_x},\ \ j = 1,2,\dots,m-1$$

3.2 网格细化分析

在网格 $n = 16,\ \ n = 32,\ \ n = 64$ 下分别使用 Gauss-Seidel 迭代法和 SOR 迭代法进行计算，为避免迭代法的停机准则对数值结果精度的影响，将迭代法的停机准则调整为 $tol = 10^{-8}$，将所得的误差矩阵取无穷范数有如下表的结果

n	Global Error	
	Guass-Seidel	SOR
16	0.00290990	0.00291020
32	0.00072947	0.00073055
64	0.00017836	0.00018220

使用如下公式对方法的精度阶数进行计算

$$p \approx \frac{\log(\|E_h\| / \|E_{h/2}\|)}{\log 2}$$

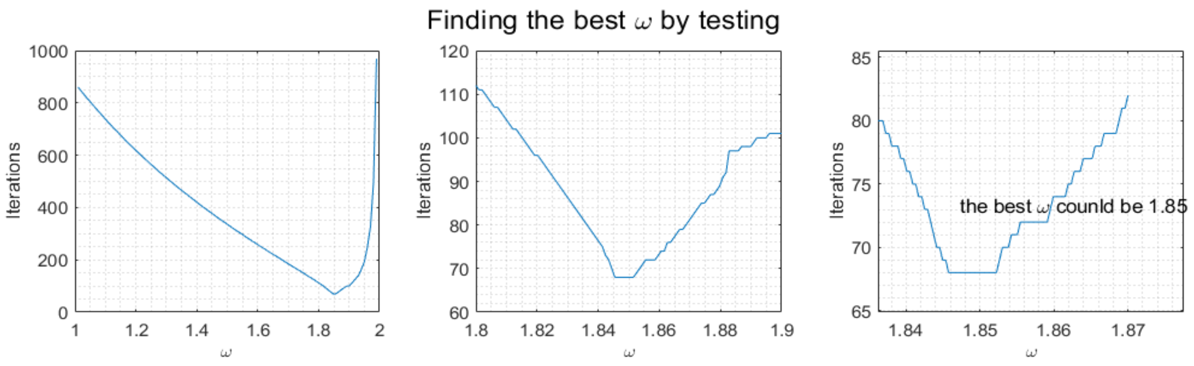
得到结果为

$$Gauss-Seidel: \frac{\log(\|E_{1/16}\| / \|E_{1/32}\|)}{\log 2} = 1.9960, \quad \frac{\log(\|E_{1/32}\| / \|E_{1/64}\|)}{\log 2} = 2.0321$$

$$SOR: \frac{\log(\|E_{1/16}\| / \|E_{1/32}\|)}{\log 2} = 1.9941, \quad \frac{\log(\|E_{1/32}\| / \|E_{1/64}\|)}{\log 2} = 2.0035$$

可见此方法表现出的结果是二阶精度的。

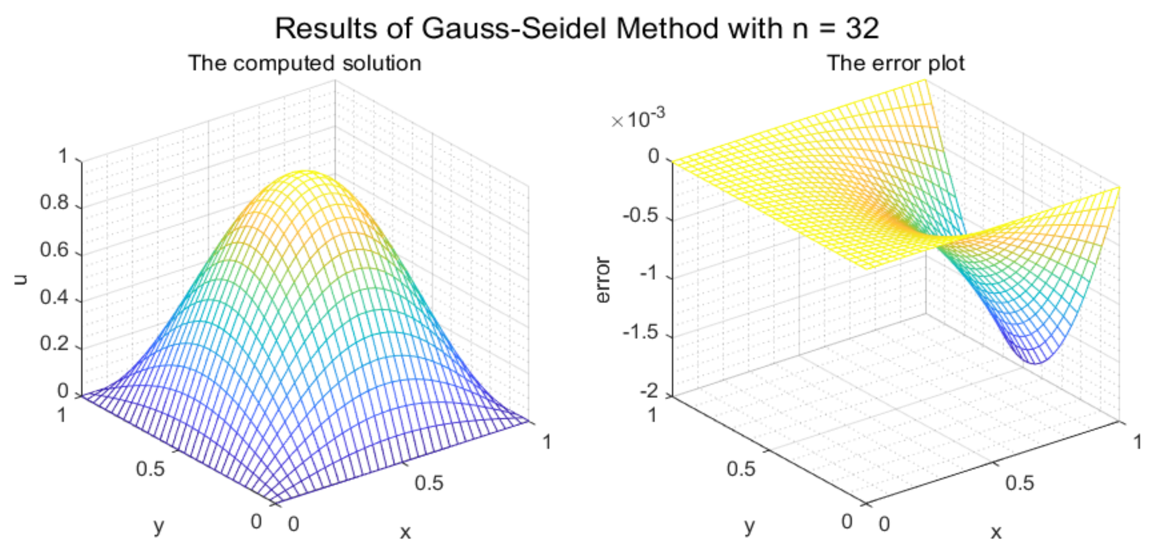
3.3 最佳松弛因子ω的确定

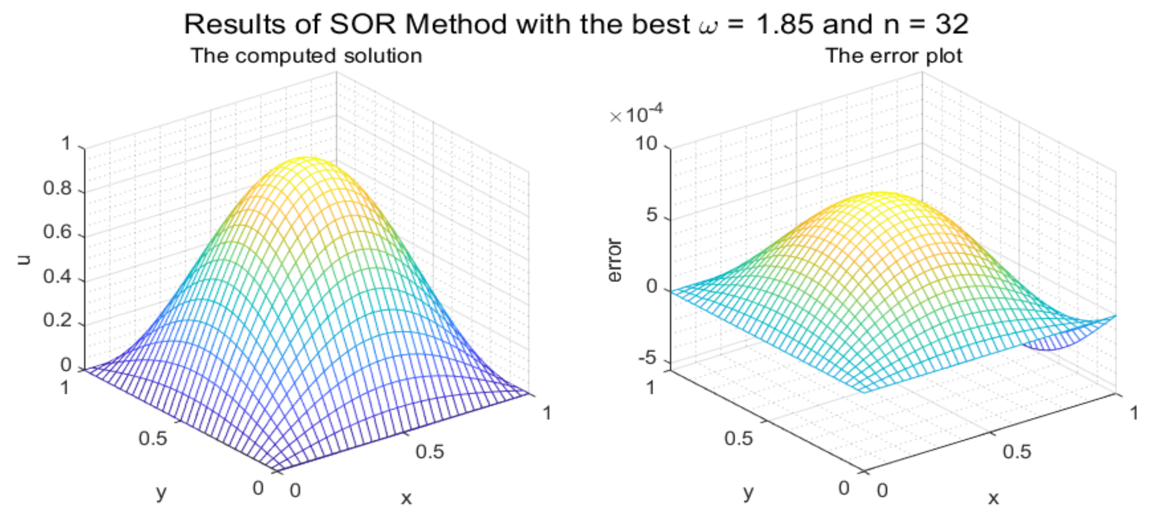


将网格取为$n=32$,迭代法的停机准则设置为$tol=10^{-5}$，在区间$(1,2)$中对ω变步长搜索，得到最佳松弛因子应为$\omega=1.85$，此时的迭代步数为 68.

3.4 数值结果

取$u_x(b,y)=\varphi(y)=-\pi\sin(\pi y)$，$p(x,y)=1+x^2+y^2$，$r(x,y)=-xy$，并取椭圆问题的真实解为$u(x,y)=\sin(\pi x)\sin(\pi y)$，在网格$n=32$下使用 Guass-Seidel 迭代法和 SOR 迭代法进行计算，迭代法的停机准则为$tol=10^{-5}$，得到如下的结果



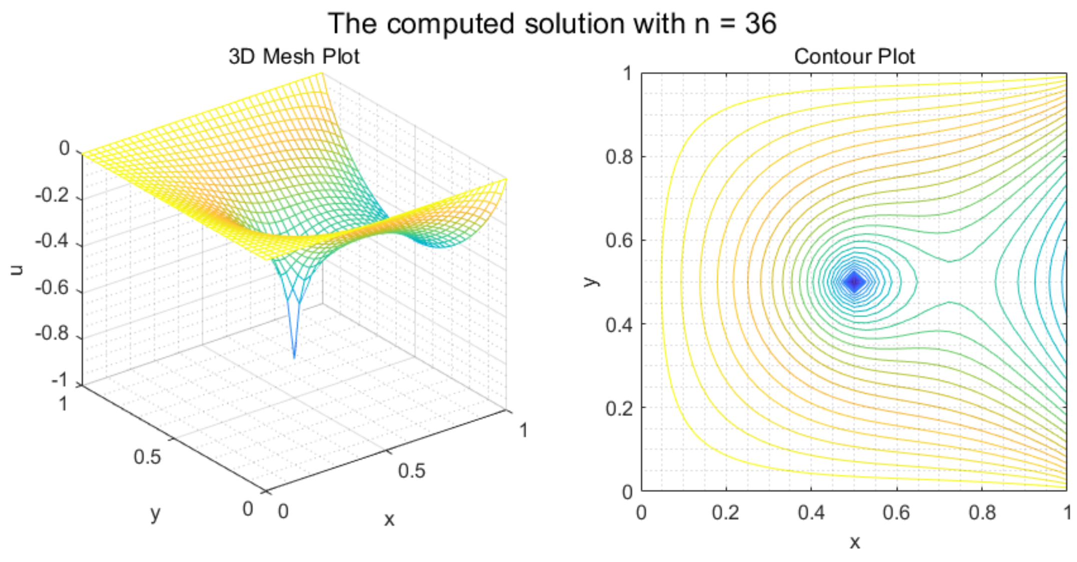


选择不同的网格尺度，得到 Guass-Seidel 迭代法和 SOR 迭代法的迭代次数如下表，可见在最佳松弛因子下，SOR 迭代法比 Guass-Seidel 迭代法收敛得更快。

n	Iterations	
	Guass-Seidel	SOR
16	266	67
32	876	68
64	2740	310

3.5 附加问题数值结果

将原微分方程简化为 $\Delta u = f$，边界条件分别为 $u(0,y)=u(x,0)=u(x,1)=0,\ \ u_x(1,y)=-1$，源项取 $f(x,y)=\delta(x-0.5)\delta(y-0.5)$，并用 $f(i,j)=\begin{cases}1/h^2 & i=j=n/2\\ 0 & else\end{cases}$ 代替，得到如下的数值结果



可见上面的数值结果模拟了一个三面有隔热墙的房间的稳态温度分布，其中一个恒定的热量从 $x=1$ 流入，且房间内 $x=0.5$, $y=0.5$ 处还有一个点源。

4 代码

4.1 问题一代码

```
% Main function for Solving the self_adjoint BVP
% (beta(x)u')'- gamma(x)u(x) = f(x) in (a,b)
% u(lb) = ua, au(ub) + bu'(ub) = c
clear;close all;
%% Input
lb = 0;
ub = 1;
ua = ufunc(lb);
a = 2;
b = -3;
c = a*ufunc(ub)+b*dufunc(ub);
%% solve
n = 80;% grid
[x,U] = problem1_solver(lb,ub,ua,a,b,c,@func,@beta,@gamma,n);
for i = 1:n
u(i) = ufunc(x(i));
end
%% plot
figure()
subplot(1,2,1)
plot(x,u,'o');hold on
plot(x,U,'-');
grid minor;title('Solution plots');xlabel('x');ylabel('u')
legend('exact solution', 'numerical solution')
subplot(1,2,2)
plot(x,U-u');
```

```matlab
grid minor;title('Error plot');xlabel('x');ylabel('error')
suptitle('Numerical Results with n = 80')
%% grid redinement analysis
n = 4;
for i = 1:17
H(i) = 1/n;
[x,U] = problem1_solver(lb,ub,ua,a,b,c,@func,@beta,@gamma,n);
u = [];
for j = 1:n
u(j) = ufunc(x(j));
end
error(i) = max(abs(U-u'));
n = 2*n;
end
p = log(error(1:16)./error(2:17))/log(2);
figure();
subplot(1,2,1);loglog(H,error);grid minor;
axis('equal'); axis('square'); xlabel('h');ylabel('error')
gtext('Slope of the method is 2')
subplot(1,2,2);plot(log10(H(2:17)),p);grid minor;
axis('equal'); axis('square'); xlabel('log_{10}h');ylabel('Order p')
suptitle('Grid refinement analysis')

function [x,U] = problem1_solver(lb,ub,ua,a,b,c,f,beta,gamma,n)
% solving the self-adjoint boundary value problem:
% (beta(x)u')'- gamma(x)u(x) = f(x) in (a,b)
% u(lb) = ua  au(ub) + bu'(ub) = c
% xi=lb+ih
h = (ub-lb)/n;
A = sparse(n,n);
F = zeros(n,1);
h2 = h^2;
for i = 2:n-1
    beta1 = feval(beta,lb+(i-1/2)*h);
    beta2 = feval(beta,lb+(i+1/2)*h);
    A(i,i-1) = beta1/h2;
    A(i,i) = -(beta1+beta2)/h2 - feval(gamma,lb+i*h);
    A(i,i+1) = beta2/h2;
end
beta1 = feval(beta,lb+(1/2)*h);
beta2 = feval(beta,lb+(3/2)*h);
A(1,1) = -(beta1+beta2)/h2 - feval(gamma,lb+h);
A(1,2) = beta2/h2;
beta1 = feval(beta,lb+(n-1/2)*h);
beta2 = feval(beta,lb+(n+1/2)*h);
A(n,n-1) = (beta1+beta2)/h2;
A(n,n) = - beta1/h2 - (1+2*h*a/b)*beta2/h2 - feval(gamma,lb+n*h);

for i=1:n
```

```
    x(i)=lb+i*h;
    F(i)=feval(f,x(i));
end
beta1 = feval(beta,lb+(1/2)*h);
F(1) = F(1) - beta1*ua/h2;
beta2 = feval(beta,lb+(n+1/2)*h);
F(n) = F(n) - beta2*2*h*c/(h2*b);
if b==0 % deal with b = 0
    A(n,n)=1;
    F(n)=c;
end
U=A\F;
return

%% functions
function b = beta(x)
b = 1 + x^2;
end
function g = gamma(x)
g = x;
end
function u = ufunc(x)
u = exp(-x)*((x-1)^2);
end
function du = dufunc(x)
du = exp(-x)*(2*x - 2) - exp(-x)*(x - 1)^2;
end
function d2u = d2ufunc(x)
d2u = 2*exp(-x) - 2*exp(-x)*(2*x - 2) + exp(-x)*(x - 1)^2;
end
function f=func(x)
% (beta(x)u')'- gamma(x)u(x) = f(x)
b = 1 + x^2;
db = 2*x;
g = x;
u = exp(-x)*((x-1)^2);
du =  exp(-x)*(2*x - 2) - exp(-x)*(x - 1)^2;
d2u =  2*exp(-x) - 2*exp(-x)*(2*x - 2) + exp(-x)*(x - 1)^2;
f = (db*du)+(b*d2u)-g*u;
end
```

4.2 问题二代码

```
% Main function for Solving non-linear ODE
% theta'' + K*sin(theta) = 0
% theta(0) = theta1, theta(2*pi) = theta2
clear;clc;
%% input
K = 20;
```

```matlab
theta = [0,-0.05];
%% solve
n = 40;% grid
h = 2*pi/n;
t = h:h:(2*pi-h);t=t';
x_initial = -0.1*cos(sqrt(K).*t+pi/2);% initial guess
% call the Newton solver
x_nonlin_numerical=...
    Newton(x_initial,@f_nonlin,@df_nonlin,K,theta,n,1.e-6);
x_lin_numerical=...
    Newton(x_initial,@f_lin,@df_lin,K,theta,n,1.e-6);
%% the exact solution of linear problem: theta''+K*theta=0
lin_exact = dsolve('D2u+K*u=0','u(0)=a','u(2*pi)=b');
lin_exact = matlabFunction(lin_exact);
x_lin_exact = lin_exact(K,theta(1),theta(2),t);
%% plot
figure();
plot([0;t;2*pi],[theta(1);x_nonlin_numerical;theta(2)],'-
*','lineWidth',1.2);hold on
plot([0;t;2*pi],[theta(1);x_lin_numerical;theta(2)],'-
+','lineWidth',1.2);hold on
plot([0;t;2*pi],[theta(1);x_lin_exact;theta(2)],'-','lineWidth',1.2);
legend('Numerical for $\frac{{{d^2}\theta }}{{d{t^2}}} + K\sin \theta  =
0$',...
    'Numerical for $\frac{{{d^2}\theta }}{{d{t^2}}} + K\theta  = 0$',...
    'Exact for $\frac{{{d^2}\theta }}{{d{t^2}}} + K\theta  = 0$',...
    'Interpreter','latex','FontSize',12)
grid minor;xlabel('t');ylabel('\theta')
title(['Solution plots with n = ',num2str(n)],'Interpreter','latex')

function x = Newton(x,func,dfunc,K,theta,n,tol)
% Newton Iteration Method
k = 0; b = ones(n-1,1);
while norm(b) > tol
    b = feval(func,x,K,theta,n);
    A = feval(dfunc,x,K,n);
    deta = linsolve(A,-b);
    x = x+deta;
    k = k + 1;
    if norm(deta)<= tol^2 % MinStep
        disp('Newton: Change in X too small.');return;
    elseif(k > 1000) % MaxIter
        disp('Newton£ºToo many function iterations.');return;
    end
end
disp('Newton converged to a root!');
end

%% functions for Discretizations and their Jacobian Matrix
```

```
function f = f_nonlin(x,K,theta,n)
f = zeros(n-1,1);
h = 2*pi/n; h2 = h^2;
f(2:n-2) = (x(1:n-3)-2*x(2:n-2)+x(3:n-1))/h2 + K*sin(x(2:n-2));
f(1) = (theta(1)-2*x(1)+x(2))/h2 + K*sin(x(1));
f(n-1) = (x(n-2)-2*x(n-1)+theta(2))/h2 + K*sin(x(n-1));
end
function df = df_nonlin(x,K,n)
df = zeros(n-1,n-1);
h = 2*pi/n; h2 = h^2;
df = df + diag(-2/h2 + K*cos(x));
df(1:n-2,2:n-1) = df(1:n-2,2:n-1) + eye(n-2)/h2;
df(2:n-1,1:n-2) = df(2:n-1,1:n-2) + eye(n-2)/h2;
end
function f = f_lin(x,K,theta,n)
f = zeros(n-1,1);
h = 2*pi/n; h2 = h^2;
f(2:n-2) = (x(1:n-3)-2*x(2:n-2)+x(3:n-1))/h2 + K*x(2:n-2);
f(1) = (theta(1)-2*x(1)+x(2))/h2 + K*x(1);
f(n-1) = (x(n-2)-2*x(n-1)+theta(2))/h2 + K*x(n-1);
end
function df = df_lin(x,K,n)
df = zeros(n-1,n-1);
h = 2*pi/n; h2 = h^2;
df = df + (-2/h2 + K)*eye(n-1);
df(1:n-2,2:n-1) = df(1:n-2,2:n-1) + eye(n-2)/h2;
df(2:n-1,1:n-2) = df(2:n-1,1:n-2) + eye(n-2)/h2;
end
```

4.3 问题三代码

4.3.1 Gauss-Seidel 法求解椭圆问题代码

```
% Gauss-Seidel method for Solving the elliptic problem
% u_xx + p*u_yy + r*u = f a<x<b c<y<d
% u(a,y) = 0, u(x,c) = 0, u(x,d) = 0, u_x(b,y) = -pi*sin(pi*y)
clear;clc;
%% Input
a = 0; b = 1; c = 0; d = 1;
n = 32;% grid
tol = 1e-5;
maxIter = 1e4;
%% generate matrix
hx = (b-a)/n; hx2=hx*hx;
hy = (d-c)/n; hy2=hy*hy;
x = linspace(a,b,n+1);
y = linspace(c,d,n+1);

P = zeros(n+1,n+1);% generate P_ij
```

```matlab
R = zeros(n+1,n+1);% generate R_ij
F = zeros(n+1,n+1);% generate F_ij
B = zeros(1,n+1);% generate Neumann boundary
for j = 1:n+1
    for i = 1:n+1
        P(i,j) = p_func(x(i),y(j));
        R(i,j) = r_func(x(i),y(j));
        F(i,j) = f_func(x(i),y(j));
    end
    B(j) = b_func(y(j));
end
u_old = ones(n+1,n+1);
u_new = zeros(n+1,n+1);
for i=1:n+1
    u_new(1,i) = u_func(a,y(i));
    u_new(i,1) = u_func(x(i),c);
    u_new(i,n+1) = u_func(x(i),d);
end
%% Gauss_Seidel
k = 0;
tic;
while max(max(abs(u_new-u_old))) > tol && k < maxIter
    k = k + 1;
    u_old = u_new;
    for i = 2:n+1
        for j = 2:n
            if i==n+1
                u_new(i,j) = ( F(i,j) - 2*B(j)/hx - 2*u_new(i-1,j)/hx2 ...
                    - P(i,j).*(u_new(i,j+1)+u_new(i,j-1))/hy2 )...
                    ./(-2/hx2 + -2*P(i,j)/hy2 + R(i,j));
            else
                u_new(i,j) = ( F(i,j) - (u_new(i+1,j)+u_new(i-1,j))/hx2 ...
                    - P(i,j).*(u_new(i,j+1)+u_new(i,j-1))/hy2 ) ...
                    ./(-2/hx2 + -2*P(i,j)/hy2 + R(i,j));
            end
        end
    end
end
toc;
display(['Iterations:' num2str(k)]);
%% exact solution
u_exact = zeros(n+1,n+1);
for i = 1:n+1
    for j = 1:n+1
        u_exact(i,j) = u_func(x(i),y(j));
    end
end
%% plots
e = max(max(abs(u_new-u_exact)));
```

```
display(['Error:' num2str(e)]);
figure();
subplot(1,2,1); mesh(x,y,u_new'); title('The computed solution')
xlabel('x');ylabel('y');zlabel('u');grid minor
subplot(1,2,2); mesh(x,y,u_new'-u_exact'); title('The error plot')
xlabel('x');ylabel('y');zlabel('error');grid minor
suptitle('Results of Gauss-Seidel Method with n = 32')
%% test functions
function u = u_func(x,y)
u = sin(pi*x)*sin(pi*y);
end
function p = p_func(x,y)
p = 1 + x^2 + y^2;
end
function r = r_func(x,y)
r = -x*y;
end
function f = f_func(x,y)
p = 1 + x^2 + y^2;
r = -x*y;
u = sin(pi*x)*sin(pi*y);
u_xx = -(pi^2)*sin(pi*x)*sin(pi*y);
u_yy = -(pi^2)*sin(pi*x)*sin(pi*y);
f = u_xx + p*u_yy + r*u;
end
function b = b_func(y)
b = -pi*sin(pi*y);
end
```

4.3.2 SOR 法求解椭圆问题代码

```
% SOR method for Solving the elliptic problem
% u_xx + p*u_yy + r*u = f a<x<b c<y<d
% u(a,y) = 0, u(x,c) = 0, u(x,d) = 0, u_x(b,y) = -pi*sin(pi*y)
clear;clc;
%% Input
a = 0; b = 1; c = 0; d = 1;
n = 32;% grid
omega = 1.85;% the best omega found by testing
tol = 1e-5;
maxIter = 1e4;
%% generate matrix
hx = (b-a)/n; hx2=hx*hx;
hy = (d-c)/n; hy2=hy*hy;
x = linspace(a,b,n+1);
y = linspace(c,d,n+1);

P = zeros(n+1,n+1);% generate P_ij
R = zeros(n+1,n+1);% generate R_ij
```

```
F = zeros(n+1,n+1);% generate F_ij
B = zeros(1,n+1);% generate Neumann boundary
for j = 1:n+1
    for i = 1:n+1
        P(i,j) = p_func(x(i),y(j));
        R(i,j) = r_func(x(i),y(j));
        F(i,j) = f_func(x(i),y(j));
    end
    B(j) = b_func(y(j));
end
u_old = ones(n+1,n+1);
u_new = zeros(n+1,n+1);
for i=1:n+1
    u_new(1,i) = u_func(a,y(i));
    u_new(i,1) = u_func(x(i),c);
    u_new(i,n+1) = u_func(x(i),d);
end
%% SOR with the best omega
k = 0;
tic;
while max(max(abs(u_new-u_old))) > tol
    k = k + 1;
    u_old = u_new;
    for i = 2:n+1
        for j = 2:n
            if i==n+1
                u_new(i,j) = (1-omega)*u_old(i,j) + omega*(...
                    ( F(i,j) - 2*B(j)/hx - 2*u_new(i-1,j)/hx2 ...
                    - P(i,j).*(u_new(i,j+1)+u_new(i,j-1))/hy2 )...
                    ./(-2/hx2 + -2*P(i,j)/hy2 + R(i,j)) );
            else
                u_new(i,j) = (1-omega)*u_old(i,j) + omega*(...
                    ( F(i,j) - (u_new(i+1,j)+u_new(i-1,j))/hx2 ...
                    - P(i,j).*(u_new(i,j+1)+u_new(i,j-1))/hy2 ) ...
                    ./(-2/hx2 + -2*P(i,j)/hy2 + R(i,j)));
            end
        end
    end
end
toc;
display(['Iterations:' num2str(k)]);
%% exact solution
u_exact = zeros(n+1,n+1);
for i = 1:n+1
    for j = 1:n+1
        u_exact(i,j) = u_func(x(i),y(j));
    end
end
%% plot
```

```
e = max(max(abs(u_new-u_exact)));
display(['Error:' num2str(e)]);
figure();
subplot(1,2,1); mesh(x,y,u_new'); title('The computed solution')
xlabel('x');ylabel('y');zlabel('u');grid minor
subplot(1,2,2); mesh(x,y,u_new'-u_exact'); title('The error plot')
xlabel('x');ylabel('y');zlabel('error');grid minor
suptitle('Results of SOR Method with the best \omega = 1.85 and n = 32')
```

4.3.3 Gauss-Seidel 法求解稳态温度分布代码

```
% solve the steady state temperature distribution
clear;clc;
%% Input
a = 0; b = 1; c = 0; d = 1;
n = 36;% grid
tol = 1e-5;
maxIter = 1e4;
f= @(x,y,n) n^2*(x==0.5)*(y==0.5);% deta function
%% generate matrix
hx = (b-a)/n; hx2=hx*hx;
hy = (d-c)/n; hy2=hy*hy;
x = linspace(a,b,n+1);
y = linspace(c,d,n+1);

P = ones(n+1,n+1);% generate P_ij
R = zeros(n+1,n+1);% generate R_ij
F = zeros(n+1,n+1);% generate F_ij
B = -ones(1,n+1);% generate Neumann boundary
for j = 1:n+1
    for i = 1:n+1
        F(i,j) = f(x(i),y(j),n);
    end
end
u_old = ones(n+1,n+1);
u_new = zeros(n+1,n+1);
for i=1:n+1
    u_new(1,i) = 0;
    u_new(i,1) = 0;
    u_new(i,n+1) = 0;
end
%% Gauss_Seidel
k = 0;
tic;
while max(max(abs(u_new-u_old))) > tol
    k = k + 1;
    u_old = u_new;
    for i = 2:n+1
        for j = 2:n
```

```
            if i==n+1
                u_new(i,j) = ( F(i,j) - 2*B(j)/hx - 2*u_new(i-1,j)/hx2 ...
                    - P(i,j).*(u_new(i,j+1)+u_new(i,j-1))/hy2 )...
                    ./(-2/hx2 + -2*P(i,j)/hy2 + R(i,j));
            else
                u_new(i,j) = ( F(i,j) - (u_new(i+1,j)+u_new(i-1,j))/hx2 ...
                    - P(i,j).*(u_new(i,j+1)+u_new(i,j-1))/hy2 ) ...
                    ./(-2/hx2 + -2*P(i,j)/hy2 + R(i,j));
            end
        end
    end
end
toc;
display(['Iterations:' num2str(k)]);
%% plot
figure();
subplot(1,2,1)
mesh(x,y,u_new'); grid minor; title('3D Mesh Plot')
xlabel('x');ylabel('y');zlabel('u')
subplot(1,2,2)
contour(x,y,u_new',30); grid minor; title('Contour Plot')
xlabel('x');ylabel('y');
suptitle('The computed solution with n = 36')
```

教 师 评 语	指导教师：　　　　年　月　日