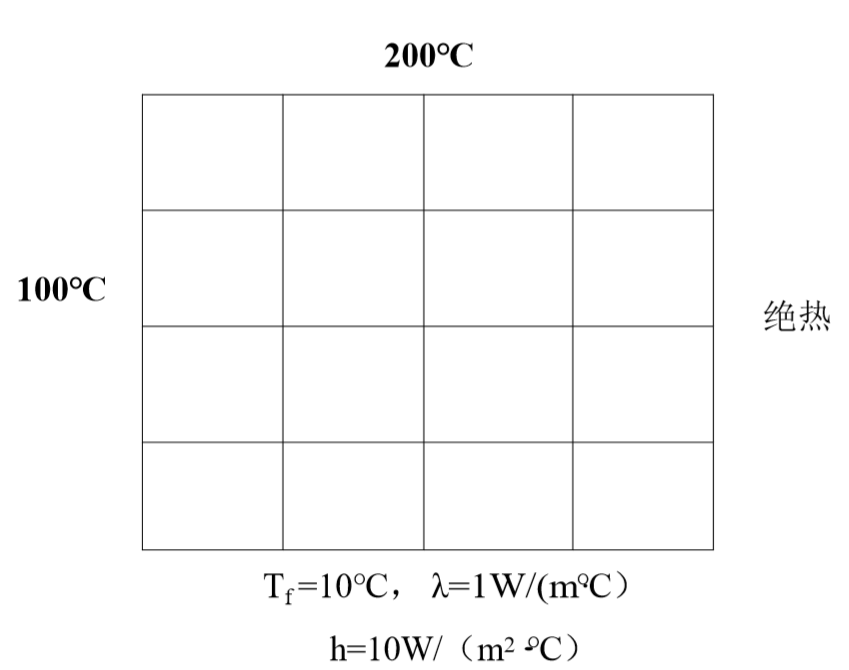
# 二维稳态计算问题

## 题目要求



• 1. 写出各未知温度节点的代数方程

• 2. 分别给出G-S迭代和Jacobi迭代程序

• 3. 程序中给出两种自动判定收敛的方法

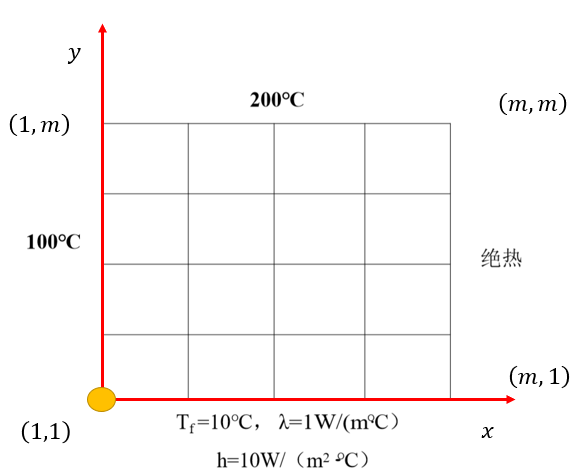
• 4. 考察三种不同初值时的收敛快慢

• 5. 上下边界的热流量 [λ=1W/(m2·℃)]

• 6. 绘出最终结果的等值线

## 各节点的离散化方程

建立如图所示直角坐标系，以左下一点为原点坐标，将长为的边界分成的离散化单元格，并写出各节点的离散化方程



### 2.1 四个外部角点

设左部边界的温度，上部边界的温度

**特别说明：**如果直接将边界的温度定为、，那么在角点则会出现温度突变，所以在此做出说明，不会画出边界上的点，默认在整个图形的左边和上边有、的边界点，但在此温度分布图中不会做出。

坐标为的外部角点离散化方程建立：

整理得：

坐标为的外部角点离散化方程建立：

整理得：

坐标为 的外部角点离散化方程建立：

整理得：

坐标为 的外部角点离散化方程建立：

整理得：

### 2.2 平直边界上的点

横坐标为的所有点（除去四个外部角点），设其点坐标为：

整理得：

横坐标为的所有点（除去四个外部角点），设其点坐标为：

整理得：

纵坐标为的所有点（除去四个外部角点），设其点坐标为：

整理得：

纵坐标为的所有点（除去四个外部角点），设其点坐标为：

整理得：

### 2.3 内部点

设内部点的坐标为，建立其离散化方程：

整理得：

### 2.4 模型综述

**外部角点:**

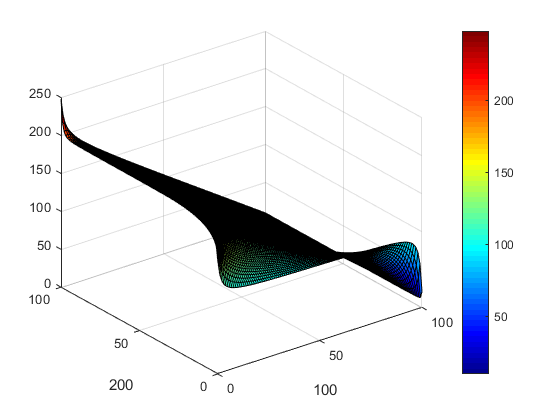
**边界点：对于、**

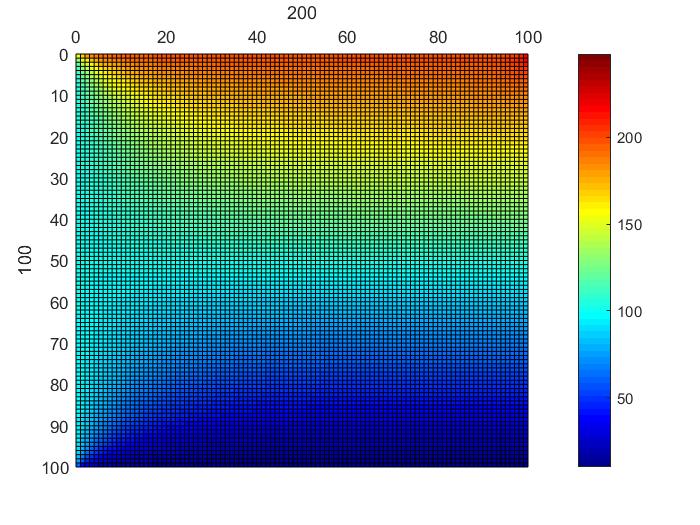
**内部点：**

## 源程序

### G-S迭代解法

clear all;  
L = 100;  
h = 10;  
Tf = 10;  
lamda = 1;  
T\_zuo = 100;  
T\_shang = 200;  
x = 0:1:100;  
y = 0:1:100;  
[X,Y] = meshgrid(x,y);  
z = zeros(size(X,1),size(Y,1));  
figure(1)  
surf(x,y,z);  
dx = L/(size(X,1)-1);  
dy = L/(size(Y,1)-1);  
n\_x = size(X,1);  
n\_y = size(Y,1);  
t = 0;  
while 1  
 %t\_n = z(floor(n\_x/2),floor(n\_y/2));  
 z\_n = z;  
 z(1,1) = (T\_zuo+T\_shang+z(1,2)+z(2,1))/4;  
 z(1,n\_y) = (lamda/2\*(100+z(1,n\_y-1)+z(2,n\_y))+h/2\*dx\*Tf)/(3/2\*lamda+h/2\*dx);  
 z(n\_x,1) = (z(n\_x-1,1)+300+z(n\_x,2))/3;  
 z(n\_x,n\_y) = (lamda/2\*(z(n\_x-1,n\_y)+z(n\_x,n\_y-1))+dx/2\*h\*Tf)/(dx/2\*h+lamda);  
  
 for i = 2:1:n\_x-1  
 z(i,1) = (T\_shang+z(i-1,1)+z(i+1,1)+z(i,2))/4;  
 z(i,n\_y) = (lamda/2\*(z(i-1,n\_y)+z(i+1,n\_y)+2\*z(i,n\_y-1))+dx\*h\*Tf)/(2\*lamda+h\*dx);  
 end  
 for i = 2:1:n\_y-1  
 z(1,i) = (T\_zuo+z(1,i+1)+z(1,i-1)+z(2,i))/4;  
 z(n\_x,i) = (z(n\_x,i+1)+z(n\_x,i-1)+2\*z(n\_x-1,i))/4;  
 end  
 for i = 2:1:n\_x-1  
 for j = 2:1:n\_y-1  
 z(i,j) = (z(i-1,j)+z(i+1,j)+z(i,j-1)+z(i,j+1))/4;  
 end  
 end  
 %t\_m = z(floor(n\_x/2),floor(n\_y/2));  
 t\_n = abs(z-z\_n);  
 t = t+1;  
 if max(t\_n) < 1e-30  
 break;  
 end  
  
end  
figure(2)  
surf(x,y,z);  
colormap(jet);  
colorbar;  
xlabel('100');  
ylabel('200');  
hold on;  
phi\_up = 0;  
phi\_down = 0;  
for i = 1:1:size(z,1)  
 phi\_up = lamda\*(z(i,1)-T\_shang)+phi\_up;  
 phi\_down = h\*(z(i,n\_y)-Tf)\*dx+phi\_down;  
end  
figure(3);  
[X,Y]=meshgrid(min(x):max(x),min(y):max(y));  
Z=griddata(x,y,z,X,Y);  
[c,h]=contour(X,Y,Z);  
colormap(jet);  
colorbar;

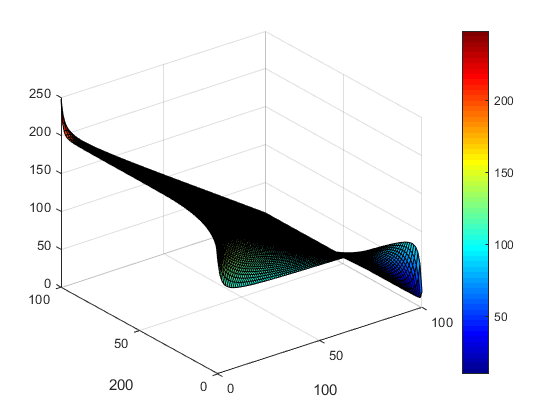


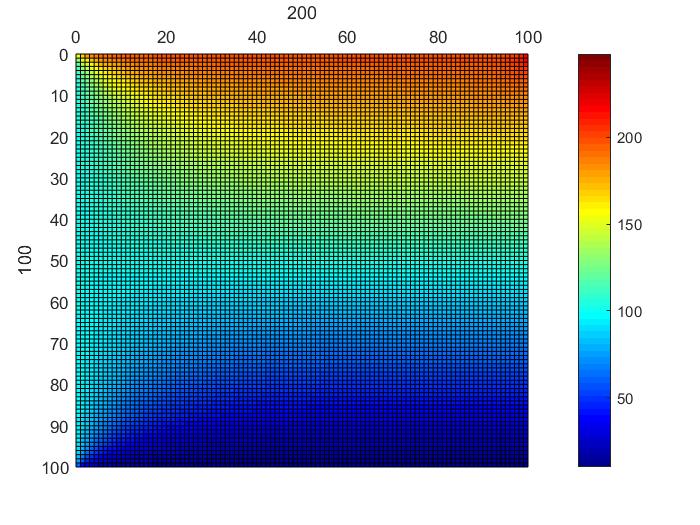


[*Published with MATLAB® R2017b*](http://www.mathworks.com/products/matlab)

### Jacobi迭代法

clear all;  
L = 100;  
h = 10;  
Tf = 10;  
lamda = 1;  
T\_zuo = 100;  
T\_shang = 200;  
x = 0:1:100;  
y = 0:1:100;  
[X,Y] = meshgrid(x,y);  
z = zeros(size(X,1),size(Y,1));  
figure(1)  
surf(x,y,z);  
dx = L/(size(X,1)-1);  
dy = L/(size(Y,1)-1);  
n\_x = size(X,1);  
n\_y = size(Y,1);  
z\_s = z;  
t = 0;  
while 1  
 %t\_n = z(floor(n\_x/2),floor(n\_y/2));  
 z\_s(1,1) = (T\_zuo+T\_shang+z(1,2)+z(2,1))/4;  
 z\_s(1,n\_y) = (lamda/2\*(100+z(1,n\_y-1)+z(2,n\_y))+h/2\*dx\*Tf)/(3/2\*lamda+h/2\*dx);  
 z\_s(n\_x,1) = (z(n\_x-1,1)+300+z(n\_x,2))/3;  
 z\_s(n\_x,n\_y) = (lamda/2\*(z(n\_x-1,n\_y)+z(n\_x,n\_y-1))+dx/2\*h\*Tf)/(dx/2\*h+lamda);  
  
 for i = 2:1:n\_x-1  
 z\_s(i,1) = (T\_shang+z(i-1,1)+z(i+1,1)+z(i,2))/4;  
 z\_s(i,n\_y) = (lamda/2\*(z(i-1,n\_y)+z(i+1,n\_y)+2\*z(i,n\_y-1))+dx\*h\*Tf)/(2\*lamda+h\*dx);  
 end  
 for i = 2:1:n\_y-1  
 z\_s(1,i) = (T\_zuo+z(1,i+1)+z(1,i-1)+z(2,i))/4;  
 z\_s(n\_x,i) = (z(n\_x,i+1)+z(n\_x,i-1)+2\*z(n\_x-1,i))/4;  
 end  
 for i = 2:1:n\_x-1  
 for j = 2:1:n\_y-1  
 z\_s(i,j) = (z(i-1,j)+z(i+1,j)+z(i,j-1)+z(i,j+1))/4;  
 end  
 end  
 %t\_m = z\_s(floor(n\_x/2),floor(n\_y/2));  
 t = t+1;  
 t\_n = z\_s - z;  
 if max(abs(t\_n))/(max(z)+1e-50) < 1e-30  
 break;  
 end  
 z = z\_s;  
end  
figure(2)  
surf(x,y,z);  
colormap(jet);  
colorbar;  
xlabel('100');  
ylabel('200');  
hold on;  
phi\_up = 0;  
phi\_down = 0;  
for i = 1:1:size(z,1)  
 phi\_up = lamda\*(z(i,1)-T\_shang)+phi\_up;  
 phi\_down = h\*(z(i,n\_y)-Tf)\*dx+phi\_down;  
end  
figure(3);  
[X,Y]=meshgrid(min(x):max(x),min(y):max(y));  
Z=griddata(x,y,z,X,Y);  
[c,h]=contour(X,Y,Z);  
colormap(jet);  
colorbar;





[*Published with MATLAB® R2017b*](http://www.mathworks.com/products/matlab)

## 不同初值的收敛快慢

### G-S迭代解法

将左边和上边初始温度分别改变，记录迭代次数，结果如下表所示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 左 上 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

可以从上表中看出，在边界温度改变的情况下，迭代次数不会发生太大的改变，都在5万次左右，改变h的值或者的值，迭代次数如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 迭代次数 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 迭代次数 |  |  |  |

可见改变初始条件不会对迭代次数的多少产生太大的影响，影响迭代次数的主要因素只有网格数和迭代方法。

### Jacobi迭代解法

将左边和上边初始温度，h的值或者的值分别改变，记录迭代次数，结果如下表所示

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 迭代次数 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 迭代次数 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 迭代次数 |  |  |  |

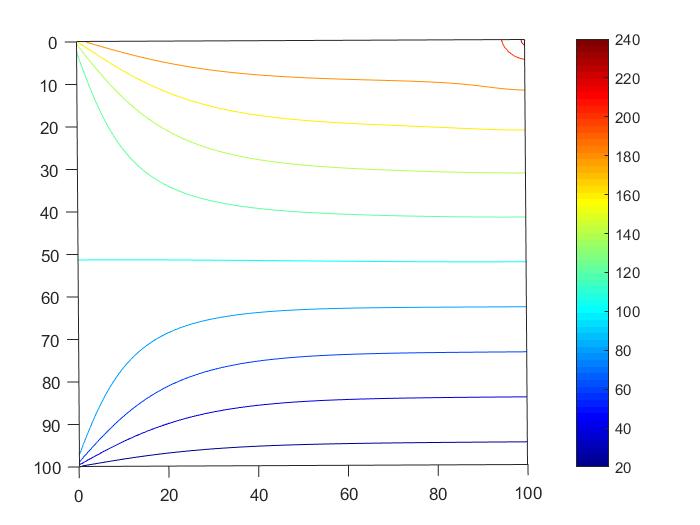
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 迭代次数 |  |  |  |

与G-S迭代法得到的结论相同

## 五、 上下边界的热流量

计算每一个节点的热流量，然后热流量加和就是上下边界总的热流量，

## 六、 计算结果的等温线图



图中坐标有误，左边坐标应该是。

## 七、 图片

