HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
School of Information and Communication Technology

BÁO CÁO TÍNH TOÁN KHOA HỌC

TÍNH TOÁN KHOA HỌC
(IT4110)

2D Laplace equations
Successive Over Relaxation Method

Nhóm 6

Thành viên	*Email*
Lê Hải Yến 20225780	yen.lh225780@sis.hust.edu.vn
Nguyễn Thị Huyền Trang 20225674	trang.nth225674@sis.hust.edu.vn
Nguyễn Thị Kiều Oanh 20225899	oanh.ntk225899@sis.hust.edu.vn
Nguyễn Ngân Hà 20225713	ha.nn225713@sis.hust.edu.vn
Bùi Thu Trang 20225938	trang.bt225938@sis.hust.edu.vn
Lê Thành Đạt 20225804	dat.lt225804@sis.hust.edu.vn

Lớp
147769 - IT4110
Giảng viên
Vu Van Thieu
thieu.vuvan@hust.edu.vn

Ngày 31 tháng 8 năm 2024

Thời gian: 23^{nd}June - 7^{th}July

Báo cáo

2D Laplace equations Successive Over Relaxation Method

Lớp: 147769

Giảng viên: Vũ Văn Thiệu

Mục lục

I Giới thiệu

I.1 Giải 2D Laplace bằng phương pháp Successive Over Relaxation (SOR)

Phương trình Laplace 2D: Là phương trình vi phân từng phần (PDE) mô tả nhiều hiện tượng vật lý như dòng nhiệt, điện thế trong một vùng không gian. Phương trình Laplace 2D có dạng:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Phương pháp SOR: Là một phương pháp để giải các hệ phương trình tuyến tính, đặc biệt là các bài toán liên quan đến phương trình Laplace trong hai chiều. Phương pháp SOR kết hợp các lợi thế của phương pháp Jacobi và Gauss-Seidel bằng cách sử dụng hệ số lặp để tăng tốc quá trình hội tụ.

I.2 Ứng dụng thực tế

Bài toán 2D Laplace – SOR có ứng dụng thực tế trong nhiều lĩnh vực khác nhau như vật lý, hóa học, sinh học, khoa học máy tính và trí tuệ nhân tạo. Nó thường được sử dụng để mô phỏng sự phân bố điện thế, nhiệt độ, trường điện từ trong các môi trường tương ứng. Ngoài ra nó còn được áp dụng để tối ưu hóa hình học và phân tích hình học. Việc nghiên cứu bài toán này góp phần nâng cao hiệu quả và chất lượng trong thiết kế, mô phỏng và tối ưu các hệ thống phức tạp.

I.3 Đặt vấn đề

Báo cáo này tập trung vào việc mô phỏng sự phân bố nhiệt độ khi đã ổn định trên miền hình vuông khi biết một số điều kiện biên cho trước (Dirichlet Boundary Condition and Mixed (Dirichlet & Neumann) Boundary Condition). Đồng thời tìm giá trị ω sao cho số bước lặp là ít nhất. Thông tin này có thể có ý nghĩa trong các lĩnh vực khác nhau, bao gồm vật lý, cơ khí và khoa học vật liệu. Sử dụng phương pháp SOR để giải các phương trình nhiệt 2D ổn định.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

II Cơ sở lý thuyết

II.1 2D Laplace

Ta sẽ giải bài toán trên theo phương pháp FDM (Finite Difference Approximations) Giả sử cần tính nhiệt độ tại điểm $(x = i, y = j)$, ta có $T(x, y) = T_{i,j}$.

Xấp xỉ đạo hàm bậc 2 của T theo x tại điểm (i, j) trên lưới $\Delta x = \Delta y = h$ theo phương pháp sai phân trung tâm bậc 2:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{(i,j)} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2}$$

$$\left.\frac{\partial^2 T}{\partial y^2}\right|_{(i,j)} \approx \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{h^2}$$

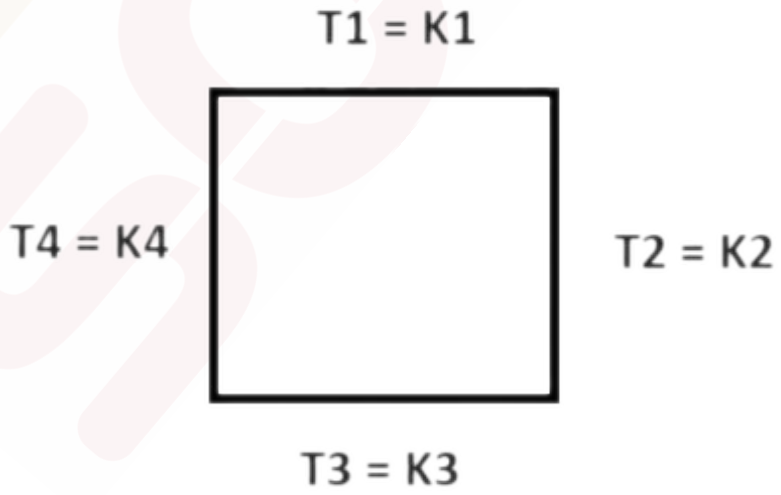
Do đó phương trình nhiệt ổn định trở thành:

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{h^2} = 0$$

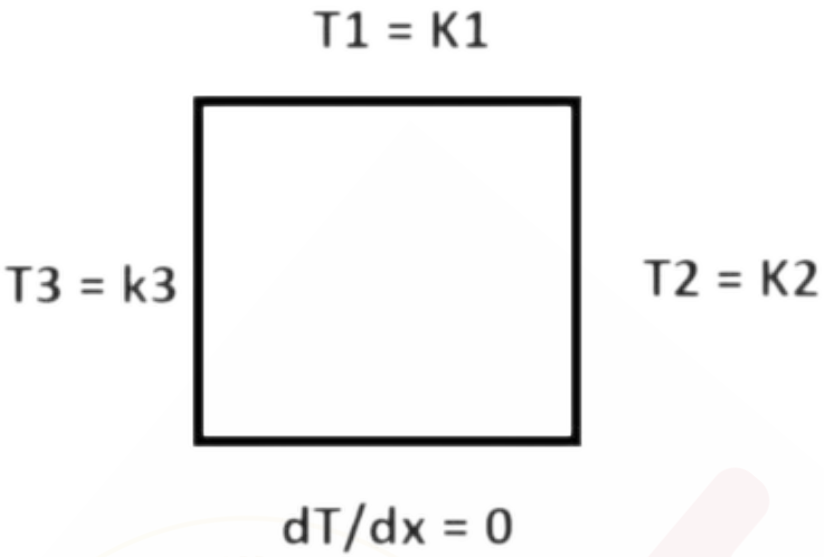
Ta sử dụng phương pháp SOR sẽ được nêu ở phần tiếp theo để giải ra $T(i, j)$ để giải.

Để giải phương trình trên chúng ta cần các điều kiện biên:

- Điều kiện biên Dirichlet: $T(x, y) = g(x, y)$ trên biên của miền.



- Điều kiện biên Dirichlet kết hợp với Neumann: điều kiện Dirichlet kết hợp các cạnh khác điều kiện sẽ là hàm vi phân của nhiệt độ theo dx hoặc dy.



- Đối với cạnh có điều kiện Neumann:
  Sử dụng công thức xấp xỉ đạo hàm:

$$f'(x) \approx \frac{3f(x) - 4f(x - \Delta x) + f(x - 2\Delta x)}{2\Delta x}$$

Khi biết 3 điểm liên tiếp:

$$\frac{3T(\text{end}) - 4T(\text{end} - 1) + T(\text{end} - 2)}{2\Delta x} = 0$$

$$\Rightarrow T(\text{end}) = \frac{4T(\text{end} - 1) - T(\text{end} - 2)}{3}$$

II.2 Successive Over Relaxation method (SOR method)

Phương pháp Successive Over Relaxation (SOR) sử dụng công thức sau

$$C_{i,j}^{k+1} = \frac{w}{4}\left(C_{i+1,j}^{(k)} + C_{i-1,j}^{(k+1)} + C_{i,j+1}^{(k)} + C_{i,j-1}^{(k+1)}\right) + (1 - w)C_{i,j}^{k}$$

Trong đó ω là tham số thư giãn. Tham số ω cho biết nồng độ tại một vị trí trong lần lặp trước ảnh hưởng đến lần lặp tiếp theo. Trong phương pháp SOR, chúng ta sử dụng giá trị ω trong phạm vi $(1, 2)$. Việc chọn ω có thể tăng tốc độ hội tụ.

Cụ thể báo cáo này sẽ nghiên cứu phương pháp SOR với giá trị ω được tối ưu.

II.3 Ứng dụng lý thuyết vào bài toán

Sau khi đã suy ra được công thức

$$T_{i,j} = \frac{1}{4}\left(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}\right)$$

Ta áp dụng phương pháp SOR như sau để giải phương trình này. Đồng thời khởi tạo vector $w = (0.5; 0.1; 1.9)$ và vector lưu để lưu số vòng lặp khi chạy với mỗi w để tìm ra w có số vòng lặp nhỏ nhất.

- Với điều kiện biên Dirichlet:
  1. Khởi tạo ma trận $\mathbf{T}_{\text{init}}$ là ma trận 0 để lưu kết quả, đồng thời khởi tạo nhiệt độ cho các điểm tại biên, sai số tolerance = 10^{-5}.
  2. Lặp lại cho đến khi hội tụ:
     - Với mỗi điểm (i,j):

$$T_{i,j}^{\text{new}} = (1-w)T_{i,j}^{\text{old}} + \frac{w}{4}\left(T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{new}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{new}}\right)$$

     - Cập nhật giá trị mới cho tất cả các điểm lưới.
     - Kiểm tra điều kiện hội tụ đến khi nào giá trị tolerance đúng.
- Với điều kiện biên Neumann:
  - Sau khi đã suy ra được:

$$T(\text{end}) = \frac{4T(\text{end}-1) - T(\text{end}-2)}{3}$$

  - Tại mỗi vòng lặp cập nhật lại điều kiện trên cạnh có điều kiện Neumann theo công thức trên cụ thể ở ví dụ nêu trên là cạnh cuối cùng.
- Vector thông lượng nhiệt
  - Vector thông lượng nhiệt (heat flux vector) là biểu diễn tốc độ, hướng, lượng của dòng nhiệt qua một đơn vị diện tích trong một khoảng thời gian đơn vị

$$\overrightarrow{\text{q}} = -k\nabla T$$

  Trong đó:
  - $\overrightarrow{\text{q}}$ là vector thông lượng nhiệt.
  - k là hệ số dẫn nhiệt của vật liệu.
  - ∇T là gradient của nhiệt độ.

III Thực hiện bài toán 2D Laplace SOR

III.1 Code

```
% Dimensions of the simulation grid in x (xdim) and y (ydim) directions
xdim=100;
ydim=100;

% CConvergence tolerance
tolerance = 1e-5;

% Create vector x, y
x=1:1:xdim+1;
y=1:1:ydim+1;

%--------DIRICHLET BOUNDARY CONDITION-----------------%

% x coordinates for boundary
i=1:1:xdim+1;

% Values of omega tested to find the optimal one
omega=0.5:0.1:1.9;

% Initializing initial temperature matrix used for all values of omega
t_init=zeros(xdim+1,ydim+1);

% A temperature of 100C is applied on one boundary,
% the remaining boundaries are going to remain at zero
t_init(i,ydim+1)=100; % right boundary
t_init(1,i)=200; % top boundary
t_init(i,1)=300; % left boundary
t_init(xdim+1,i)=400; % bottom boundary

% Matrix of iterations for various values of omega
iter=zeros(1,length(omega));

% Running for loop to compute number of iterations for different omega
for range=1:1:length(omega)

   % Initializing previous (t_prev) and present (t_now)
   % iterations' temperature matrix
   t_now=t_init;
   t_prev=t_init;

   % Giving initial difference between t_now and t_prev to start the iterations
   t_prev(2, ydim)=1;

   % Iteration loop
   while(max(max(abs(t_now-t_prev)))>tolerance) % Run this until convergence

```

```
        % Updating previous iteration matrix as the present
        % iteration matrix to continue iterations
        t_prev=t_now;

        % Iteration counter increment for each omega
        iter(range)=iter(range)+1;

        % Updating
        for i=2:1:xdim
            for j=2:1:ydim
                t_now(i,j)=t_now(i,j)+omega(range)*(((t_now(i-1,j)
                +t_now(i+1,j)+t_now(i,j-1)+t_now(i,j+1))/4)-t_now(i,j));
            end
        end

        % Movie type colour scaled image plot to see how solution
        % progresses for omega=1.9
        if range==length(omega)
            figure(1);
            imagesc(t_now);
            colorbar;
            colormap(jet);
         title(['\fontsize{7}Temperature distribution at iteration no ',
         int2str(iter(range)),' for omega=1.9 (Dirichlet BC)'],'Color','k');
            xlabel('x-axis','FontSize',10);
            ylabel('y-axis','FontSize',10);
            set(gca,'FontSize',10);
            getframe;
        end
    end
end

% Compute gradient to get the heat flux
[Tx, Ty] = gradient(t_now);

% Plot heat flux vectors
figure(2);
quiver(x, y, -Tx, -Ty);
title('Heat flux vectors (Dirichlet BC)');
xlabel('x');
ylabel('y');
axis tight equal;

% Plot image
figure(3);
surf(x,y,t_now);
title('Temperature distribution (Dirichlet BC)');
xlabel('x');
ylabel('y');
zlabel('Temperature');

```

```
%Plotting alpha v/s iterations
figure(4);
plot(omega,iter);
title(['\fontsize{7}Plot of omega v/s no. of iterations on a grid of ',
int2str(xdim),' x ',int2str(ydim), ' Dirichlet BC'],'color','k');
xlabel('omega','FontSize',10);
ylabel('No of iterations','FontSize',10);

%--------MIXED BOUNDARY CONDITION (DIRICHLET AND NEUMANN)-----------------%

% x coordinates for boundary
i=1:1:xdim+1;

% Values of omega tested to find the optimal one
omega=0.5:0.1:1.9;

% Initializing initial temperature matrix used for all values of omega
t_init=zeros(xdim+1,ydim+1);

% Boundary conditions
t_init(i,ydim+1)=76; % right boundary (Dirichlet BC)
t_init(1,i)=78; % top boundary (Dirichlet BC)
t_init(i,1)=75; % left boundary (Dirichlet BC)

% bottom boundary is Neumann BC

% Matrix of iterations for various values of omega
iter=zeros(1,length(omega));

% Running for loop to compute number of iterations for different omega
for range=1:1:length(omega)

    % Initializing previous (t_prev) and present (t_now)
    % iterations' temperature matrix
   t_now=t_init;
   t_prev=t_init;

   % Giving initial difference between t_now and t_prev to start the iterations
   t_prev(2, ydim)=1;

   % Iteration loop
   while(max(max(abs(t_now-t_prev)))>tolerance) % Run this until convergence

       % Updating previous iteration matrix as the present
       % iteration matrix to continue iterations
       t_prev=t_now;

       % Iteration counter increment for each omega
       iter(range)=iter(range)+1;

       % Updating
```

```
        for i=2:1:xdim
            for j=2:1:ydim
                t_now(i,j)=t_now(i,j)+omega(range)*(((t_now(i-1,j)
                +t_now(i+1,j)+t_now(i,j-1)+t_now(i,j+1))/4)-t_now(i,j));
            end
        end

        % Update Neumann BC
        t_now(xdim + 1,:) = (4.*t_now(xdim,:)-t_now(xdim-1,:))/3;

        % Movie type colour scaled image plot to see how solution
        % progresses for omega=1.9
        if range==length(omega)
            figure(5);
            imagesc(t_now);
            colorbar;
            colormap(jet);
    title(['\fontsize{7}Temperature distribution on a at iteration no ',
    int2str(iter(range)),' for an alpha=1.9 (Mixed BC)'],'Color','k');
            xlabel('x-axis','FontSize',10);
            ylabel('y-axis','FontSize',10);
            set(gca,'FontSize',10);
            getframe;
        end
    end
end

% Compute gradient to get the heat flux
[Tx, Ty] = gradient(t_now);

% Plot heat flux vectors
figure(6);
quiver(x, y, -Tx, -Ty);
title('Heat flux vectors (Mixed BC)');
xlabel('x');
ylabel('y');
axis tight equal;

% Plot image
figure(7);
surf(x,y,t_now);
title('Temperature distribution (Mixed BC)');
xlabel('x');
ylabel('y');
zlabel('Temperature');

%Plotting alpha v/s iterations
figure(8);
plot(omega,iter);
title(['\fontsize{7}Plot of omega v/s no. of iterations on a grid of ',
int2str(xdim),' x ',int2str(ydim), ' Mixed BC'],'color','k');
```

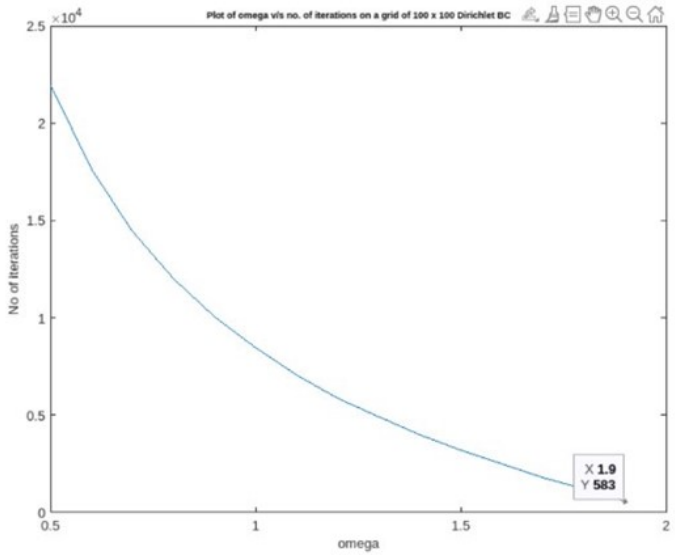
```
200 xlabel('omega','FontSize',10);
201 ylabel('No of iterations','FontSize',10);
```

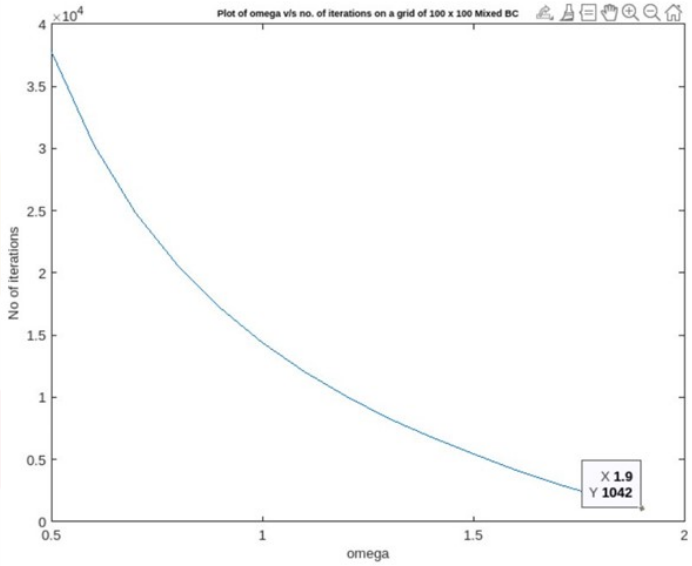
III.2 Các khái niệm và bước chính

1. **Khởi tạo lưới:** Lưới được định nghĩa với kích thước 100×100. Ma trận nhiệt độ t_{init} được khởi tạo bằng không.
2. **Điều kiện biên Dirichlet:**
   - Biên phải: 100°C.
   - Biên trên, trái và dưới: 0°C.
3. **Điều kiện biên Hỗn hợp:**
   - Biên phải: 50°C (Dirichlet).
   - Biên trên: 100°C (Dirichlet).
   - Biên trái: 75°C (Dirichlet).
   - Biên dưới: Neumann (gradient bằng 0).
4. **Hệ số thư giãn (ω):** ω thay đổi từ 0.5 đến 1.9.
5. **Vòng lặp lặp lại:** Đối với mỗi ω, phân bố nhiệt độ được cập nhật lặp đi lặp lại cho đến khi sự thay đổi tối đa giữa các lần lặp nhỏ hơn mức dung sai được chỉ định (1×10^{-5}). Số lần lặp cho mỗi ω được ghi lại.
6. **Hội tụ và Hình ảnh hóa:** Trong các lần lặp cho $\omega = 1.9$, phân bố nhiệt độ được hiển thị. Sau khi hội tụ, phân bố nhiệt độ và các vector thông lượng nhiệt được vẽ. Cuối cùng, một biểu đồ số lần lặp theo ω được tạo để tìm ω tối ưu.

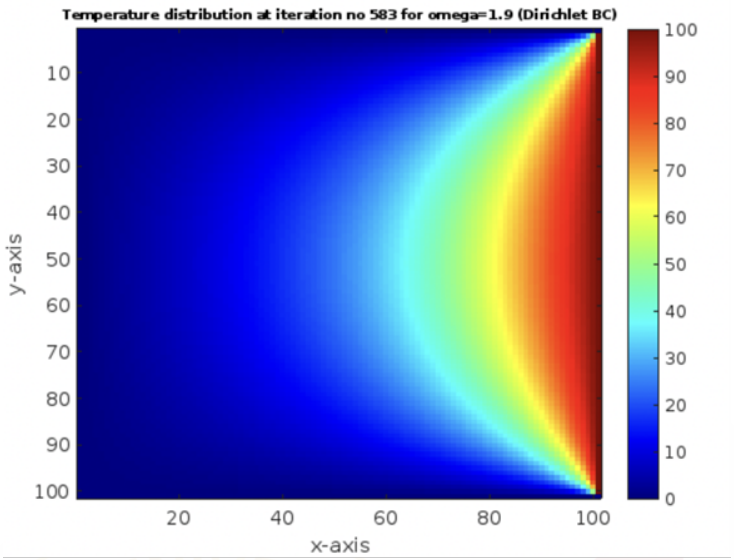
IV Kết quả và giải thích

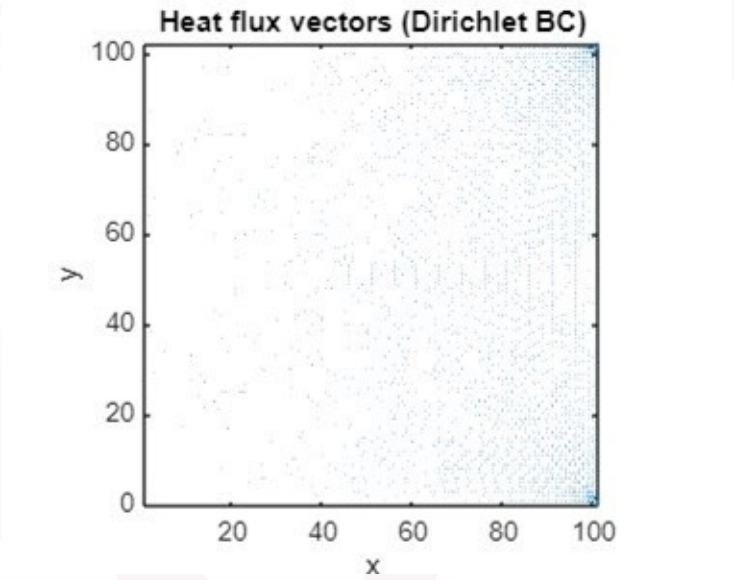
1. Kết quả và giải thích khi thực hiện bài toán 2D Laplace



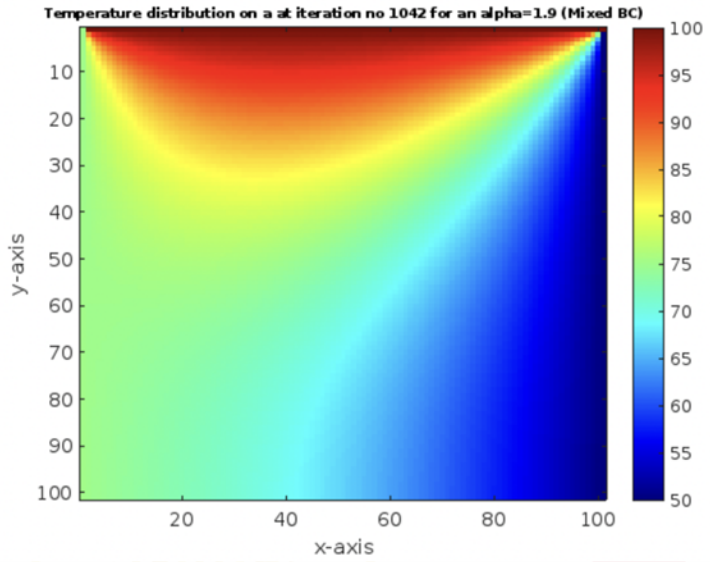


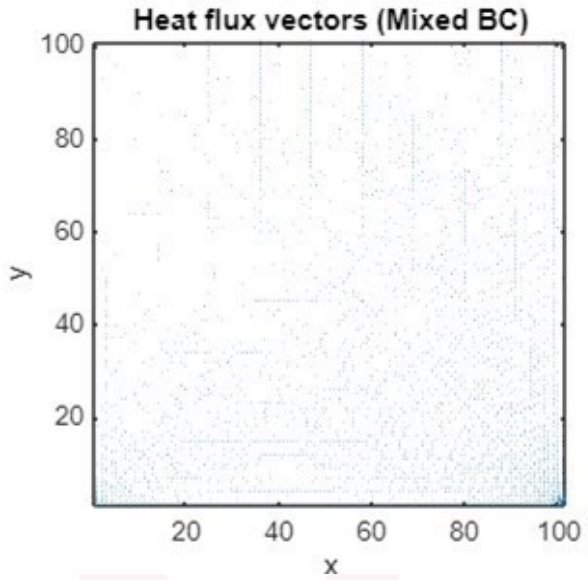
- Hệ số ω ảnh hưởng đến số vòng lặp của cả hai trường hợp biên. Từ biểu đồ ω - số vòng lặp, khi $\omega = 1.9$ thì đều cho ra số vòng lặp là nhỏ nhất (số vòng lặp lần lượt là 583 với điều kiện biên Dirichlet, 1042 với điều kiện biên Dirichlet kết hợp Neuman). Vì vậy, với $\omega = 1.9$, thuật toán cho ra kết quả tối ưu nhất.



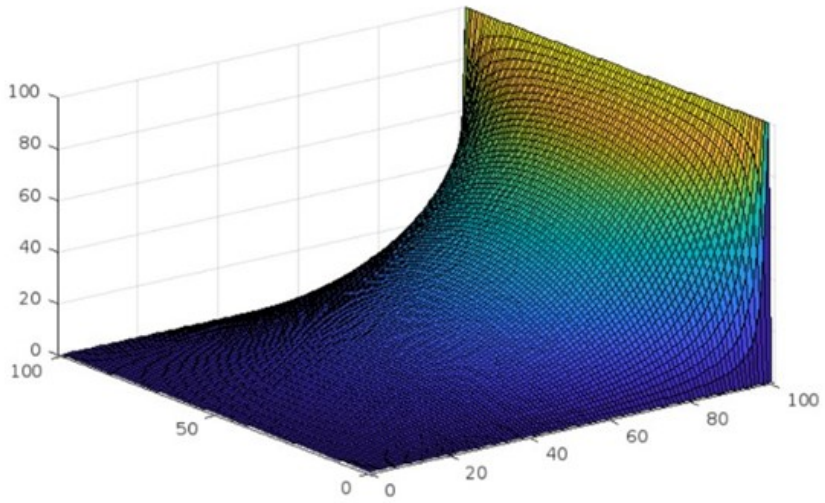


- Hình màu biểu diễn sự phân bố nhiệt độ trong thanh sắt. Màu đỏ thể hiện nhiệt độ nóng nhất. Dải màu giảm cho đến màu xanh dương thể hiện sự cân bằng nhiệt độ với môi trường.
- Hình Heat flux vector thể hiện sự phân bố của vector thông lượng. Vector thông lượng nhiệt sẽ phân bố dày đặc (có giá trị lớn) tại các vị trí mà sự truyền nhiệt diễn ra mạnh mẽ hơn. Đó là các vị trí có sự chênh lệch nhiệt độ lớn trên một khoảng cách nhỏ hay những vị trí có nguồn nhiệt mạnh.

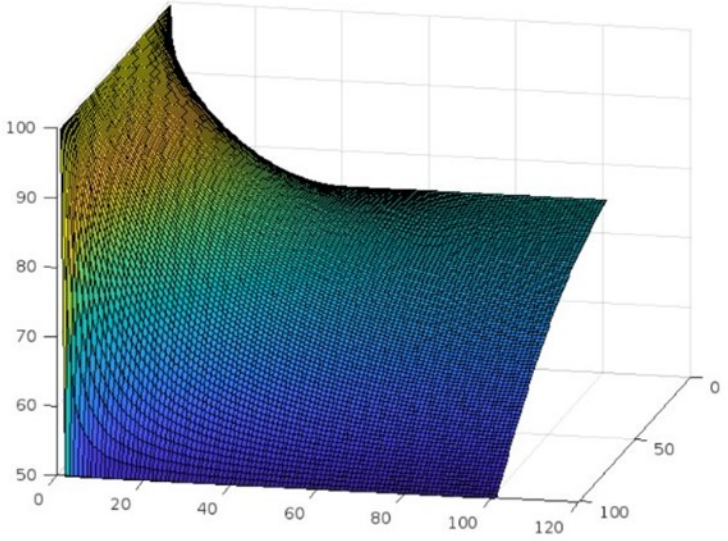




- Do trục tung của hình biểu diễn phân bố nhiệt độ và hình biểu diễn phân bố vector thông lượng nhiệt là ngược nhau nên chúng ta sẽ thấy kết quả như hai hình trên.



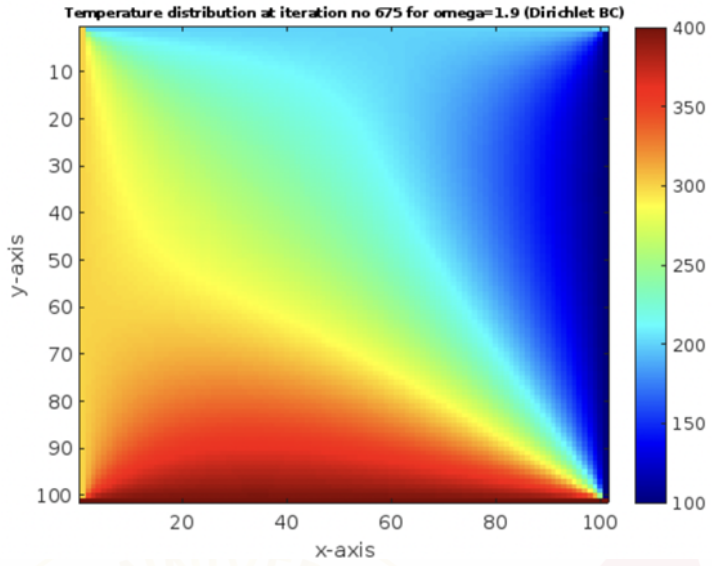
Hình 1: Phân bố nhiệt độ trên 3 trục vector của điều kiện biên Dirichlet

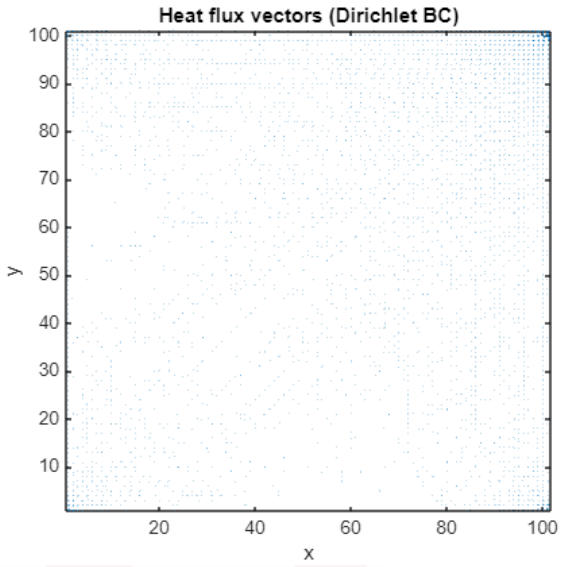


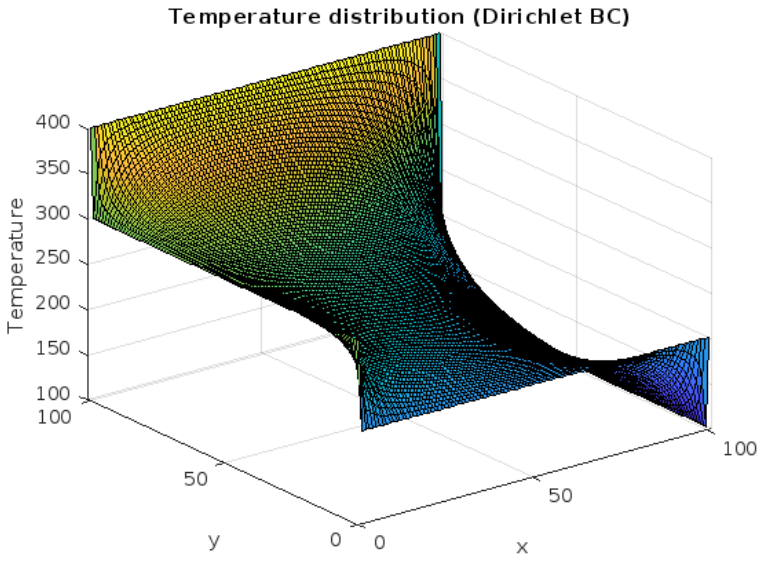
Hình 2: Phân bố nhiệt độ trên 3 trục vector của điều kiện biên Dirichlet kết hợp Neumann

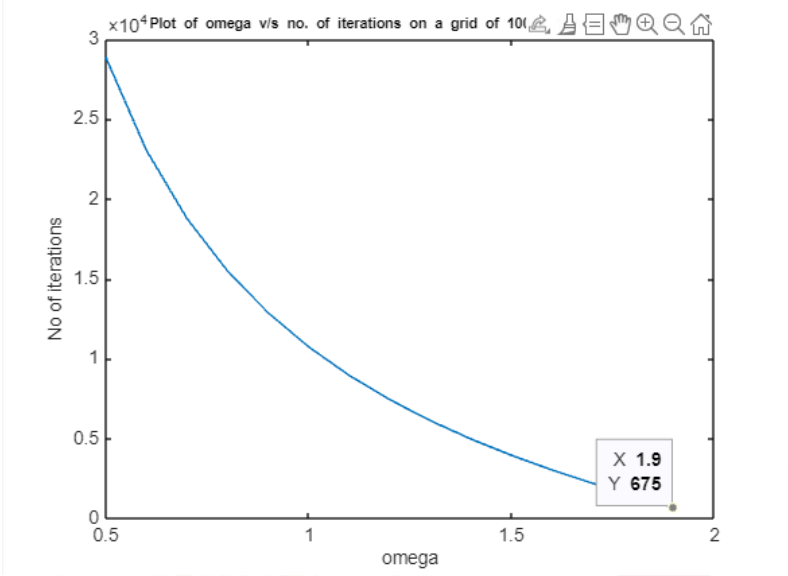
2. Kết quả khi thực hiện một số trường hợp khác

- Trường hợp 1
  - Với điều kiện biên Dirichlet
    % right boundary = 100
    % top boundary = 200
    % left boundary = 300
    % bottom boundary = 400


Temperature distribution at iteration no 675 for omega=1.9 (Dirichlet BC)
y-axis
x-axis
10
20
30
40
50
60
70
80
90
100
20
40
60
80
100
400
350
300
250
200
150
100

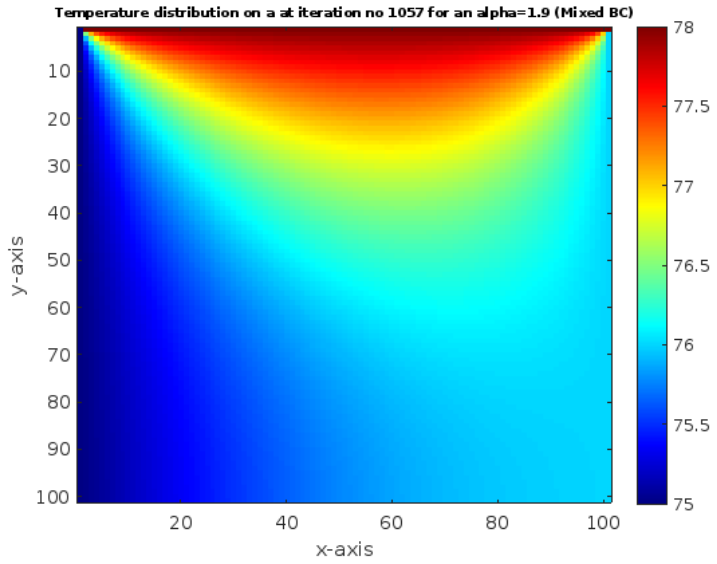

Heat flux vectors (Dirichlet BC)
y
x
100
90
80
70
60
50
40
30
20
10
20
40
60
80
100

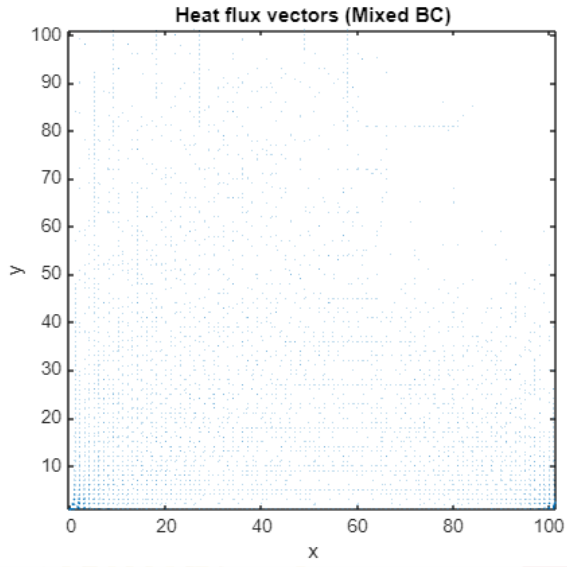

Temperature distribution (Dirichlet BC)
Temperature
y
x
400
350
300
250
200
150
100
100
50
0
0
50
100

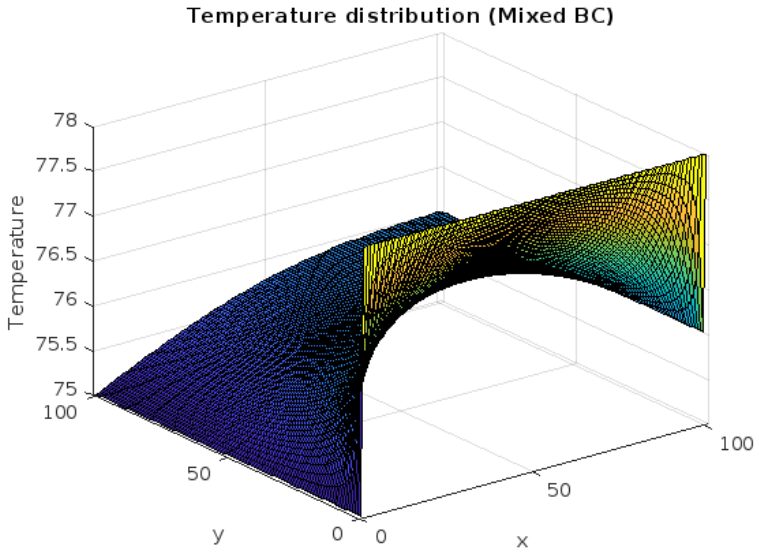


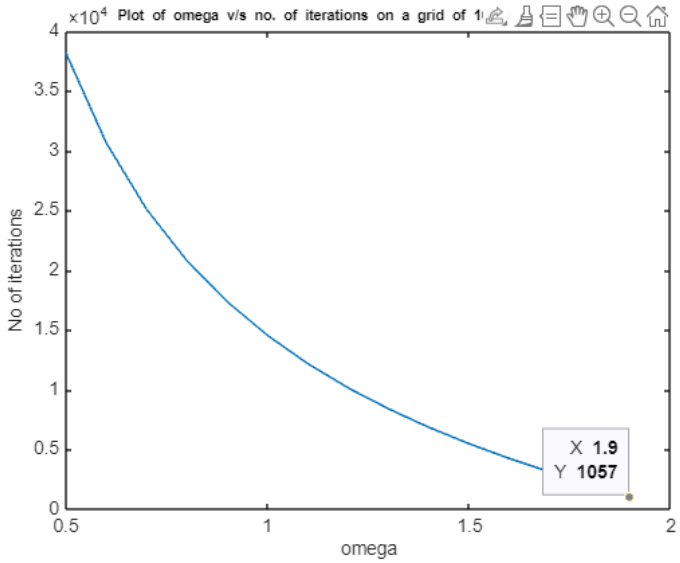
– Với điều kiện biên Dirichlet kết hợp Neumann

```
% right boundary (Dirichlet BC) = 76
% top boundary (Dirichlet BC) = 78
% left boundary (Dirichlet BC) = 75
```



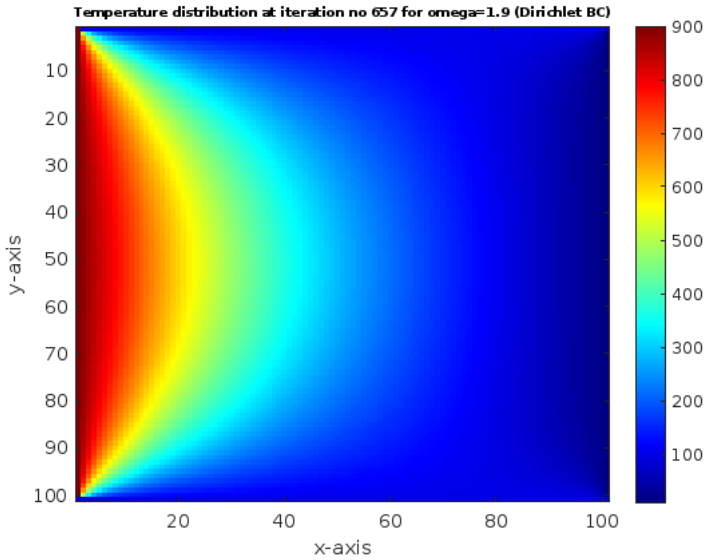

Heat flux vectors (Mixed BC)
y
x
100
90
80
70
60
50
40
30
20
10
0
20
40
60
80
100

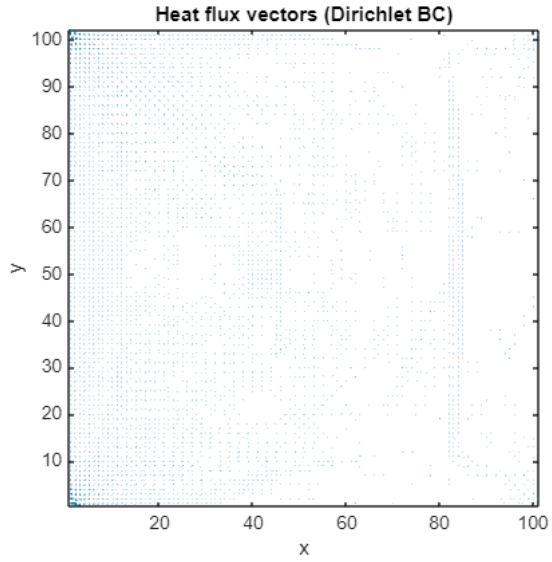

Temperature distribution (Mixed BC)
Temperature
78
77.5
77
76.5
76
75.5
75
100
50
0
y
0
50
100
x

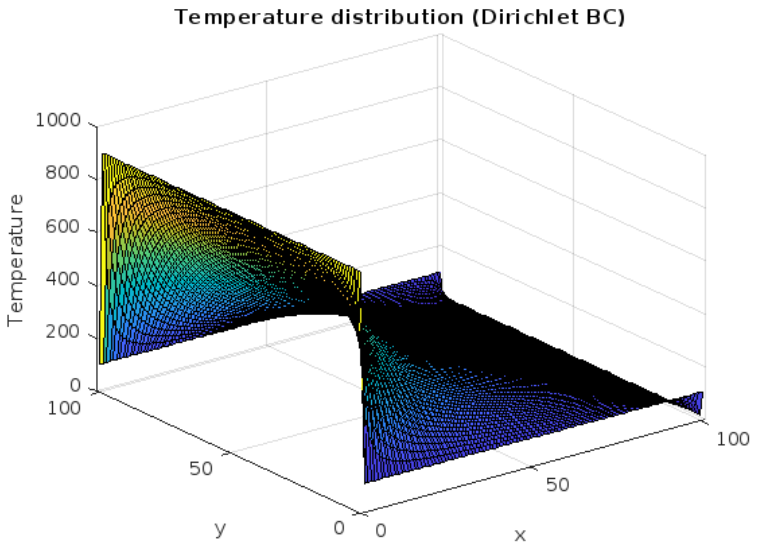

×10^4 Plot of omega v/s no. of iterations on a grid of 1
No of iterations
omega
4
3.5
3
2.5
2
1.5
1
0.5
0
0.5
1
1.5
2
X 1.9
Y 1057

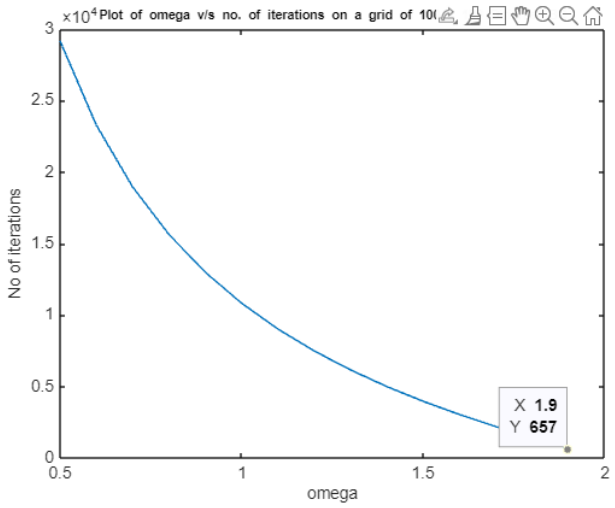
- Trường hợp 2
  - Với điều kiện biên Dirichlet

    % right boundary = 10
    % top boundary = 100
    % left boundary = 900
    % bottom boundary = 98.98



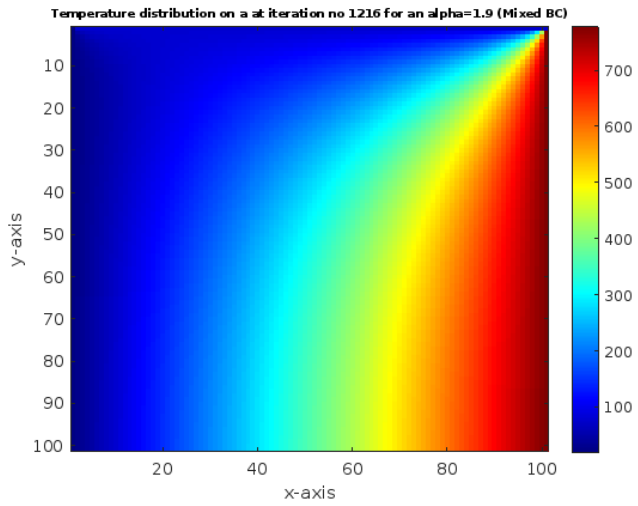


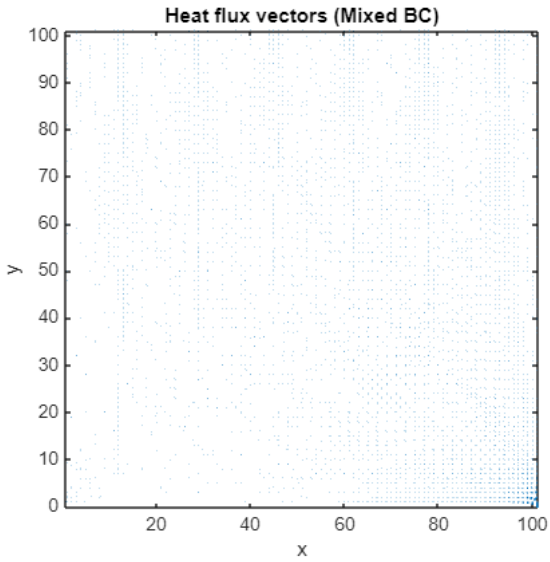


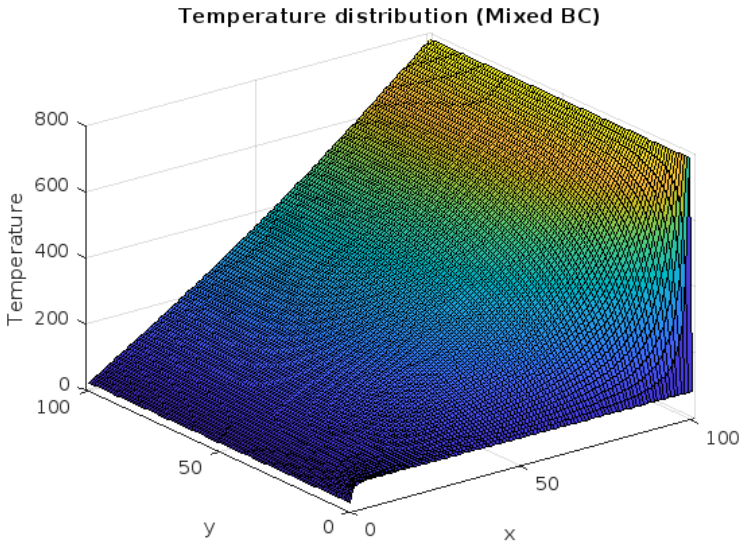


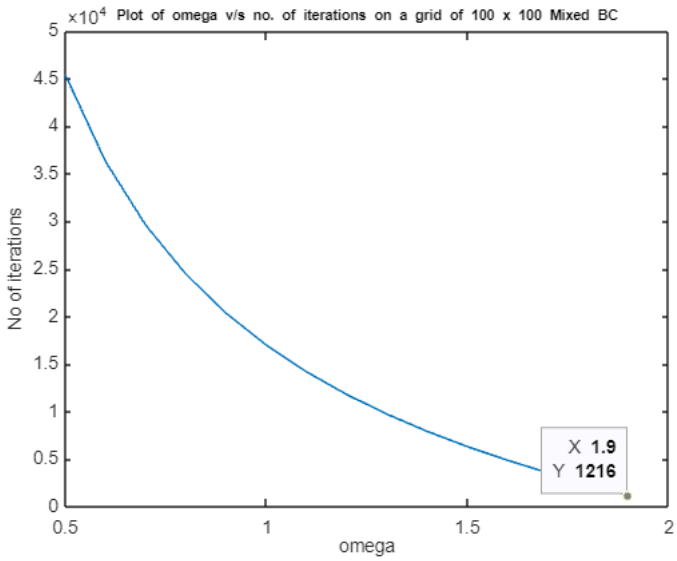
– Với điều kiện biên Dirichlet kết hợp Neumann

```
t_init(i, ydim+1) = 778.90; % right boundary (Dirichlet BC)
t_init(1, i) = 78; % top boundary (Dirichlet BC)
t_init(i, 1) = 20; % left boundary (Dirichlet BC)
```




Heat flux vectors (Mixed BC)
y
x


Temperature distribution (Mixed BC)
Temperature
y
x


Plot of omega v/s no. of iterations on a grid of 100 x 100 Mixed BC
No of iterations
omega
X 1.9
Y 1216

V Kết luận

Mô phỏng nhiệt độ bằng phương pháp SOR để giải phương trình Laplace 2D với 2 điều kiện biên là Dirichlet và Mixed (Dirichlet & Neumann) là phương pháp tốt nhất để giải bài toán đã đặt ra. Tối ưu hóa ω để giảm số lần lặp ít nhất có thể để thuật toán kết thúc nhanh hơn.

VI Tham khảo

Tài liệu

[1] Rahmat Sunarya. *Elliptic PDE-Laplace Equation Heat Transfer in Square Plate insulated. Neumann Boundary Condition.*

[2] William F. Trench Trinity University. *Laplace's Equation in Rectangular Coordinates.*