FINITE DIFFERENCE ANALYSIS OF 2-D STEADY STATE HEAT CONDUCTION IN A RECTANGULAR DOMAIN

MID-TERM PROJECT

in

MEEN 689 – COMPUTATIONAL FLUID DYNAMICS

By

B SATYA AKHIL REDDY

826006779

Date: 11/7/2018

INTRODUCTION

This report presents a finite difference analysis of 2-D steady state heat conduction in a rectangular domain. The heat conduction in the domain is governed by the partial differential equation

$$\frac{\delta^2 T}{\delta x^2} + \frac{\delta^2 T}{\delta y^2} = 0 \quad (1)$$

The above elliptic partial differential equation is also known as Laplace equation. The information is transferred instantaneously without any sense of direction. To discretize this PDE, we use the 'five-point' formula. It is obtained by using second order numerical approximation along both x and y directions.

We use iterative methods to find the solution of a discretized elliptic partial differential equation. In our case, we use Successive Over Relaxation (SOR) and Alternating Direction Implicit methods (ADI). SOR is an explicit-like method i.e. the value at a node in the current iteration is dependent on the values of nodes from the previous iteration and already calculated values from the current iteration. In addition, it involves a relaxation coefficient (ω) for fine-tuning the algorithm. The scheme is stable only when $0 < \omega < 2$, the particular value being determined on the basis of numerical analysis.

ADI is an implicit-like method where a set of equations for the current iteration are solved along the X-direction followed by Y-direction. It forms a tri-diagonal system along each direction thus making it easier to solve computationally. Since the scheme is implicit, it is unconditionally unstable.

In our problem, we consider a rectangular plate with dimension $L = 0.4m$ and $W = 0.3m$. The plate is made of copper with thermal diffusivity of $\alpha = 11.123e - 05 \frac{m^2}{s}$ and thermal conductivity $k = \frac{280W}{m°C}$. The boundary conditions applied are $T1 = 40°C$, $T2 = 0°C$, $T3 = 10°C$ and $T4 = 0°C$.

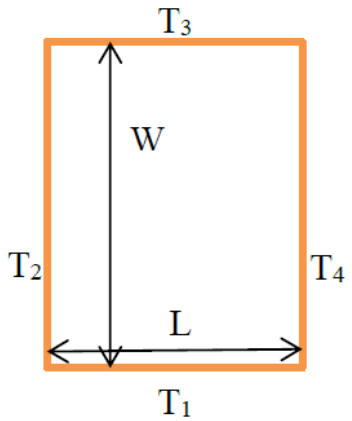

Figure 1 Rectangular plate of dimensions LxW with applied boundary conditions

In the upcoming sections, we show the method of solution followed to arrive at the solution and discuss the results obtained by performing analysis based on certain metrics.

METHOD OF SOLUTION

The partial differential equation for heat conduction in a 2-D domain is given in Eq. (1). We develop non-dimensional form of this PDE to parameterize the problem. This is done by selecting suitable scales for length and temperature and substituting the non-dimensional variables into the equation.

a) Non-Dimensionalization

The following temperature and length scales were selected to solve the problem

$$\bar{T} = \frac{T - T3}{T1 - T3} \tag{2}$$

$$\bar{x}, \bar{y} = \frac{x}{L}, \frac{y}{L} \tag{3}$$

Here, bar indicates a non-dimensional variable.

Substituting Eq. (2) and (3) in Eq. (1), we modify the PDE as

$$\frac{\delta^2 \bar{T}}{\delta \bar{x}^2} + \frac{\delta^2 \bar{T}}{\delta \bar{y}^2} = 0 \tag{4}$$

b) Discretization

To discretize the non-dimensional PDE, we use central difference approximation, which is of second order to develop the five-point formulation (stencil shown in Fig. 2).

$$\frac{\bar{T}_{i-1,j} - 2\bar{T}_{i,j} + \bar{T}_{i+1,j}}{\overline{\Delta x}^2} + \frac{\bar{T}_{i,j-1} - 2\bar{T}_{i,j} + \bar{T}_{i,j+1}}{\overline{\Delta y}^2} \tag{5}$$

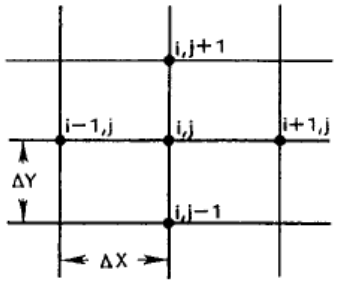

Figure 2 Stencil for a 5-point formula

The solution for the current node $\bar{T}_{i,j}$ from the Eq. (5) can be expressed as

$$\bar{T}_{i,j} = \frac{1}{2(1+\beta^2)}[\bar{T}_{i-1,j} + \bar{T}_{i+1,j} + \beta^2(\bar{T}_{i,j-1} + \bar{T}_{i,j+1})] \tag{6}$$

Where, $\beta = \frac{\Delta x}{\Delta y}$

c) Solution Methods

Equation (6) stands as the root equation on which several schemes can be applied. In our study, we are interested in conducting an analysis on ADI and SOR schemes.

c.1) Successive Over Relaxation (SOR)

This method is a variant of Gauss-Seidel method for solving iterative equations. It helps in achieving faster convergence, which is dependent on the relaxation coefficient ω. Thus, it is intuitive that obtaining an optimal value of ω is crucial in SOR.

According to Gauss-Seidel, Eq. (6) can be written in an iterative approach as follows

$$\bar{T}_{i,j}^{*k+1} = \frac{1}{2(1+\beta^2)}[\bar{T}_{i-1,j}^{*k+1} + \bar{T}_{i+1,j}^{k} + \beta^2(\bar{T}_{i,j-1}^{*k+1} + \bar{T}_{i,j+1}^{k})] \tag{7}$$

Here, k+1 refers to the current iteration and k refers to the previous iteration. To obtain the final solution for an iteration in SOR, we introduce the relaxation coefficient in the following way.

$$\bar{T}_{i,j}^{k+1} = \bar{T}_{i,j}^{k} + \omega(\bar{T}_{i,j}^{*k+1} - \bar{T}_{i,j}^{k}) \tag{8}$$

ω helps in accelerating the solution. But for the solution to converge, the value of ω must be in the range of $0 < \omega < 2$. If $\omega < 1$, it helps in achieving stability whereas $\omega > 1$ increases the rate of convergence.

Optimal value of ω is obtained by looking at the error and number of iterations for every value of ω, which is dealt with in the results section. The code for SOR method has been developed in MATLAB and can be found in the appendix for reference. The stencil for SOR is shown in Fig. 3.

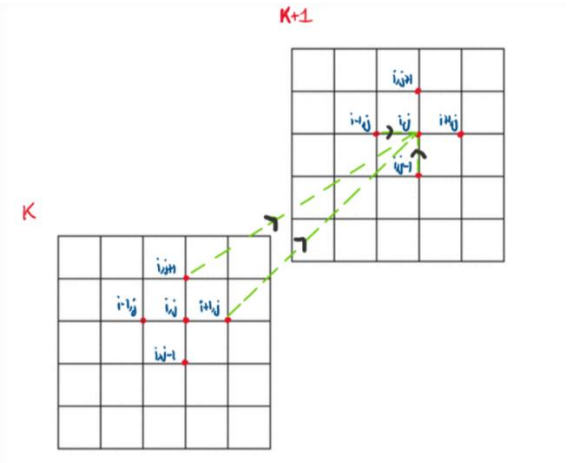

Figure 3 Stencil for $\bar{T}_{i,j}^{*k+1}$

c.2) Alternating Direction Implicit (ADI)

The similarity between time dependent and iterative problems enables us to use ADI in solving discretized elliptical partial differential equations. In ADI, for a 2-D domain, an iteration is considered as complete when a tri-diagonal system is solved for all the rows followed by all the columns. Thus, Eq. (6) is modified in the following way

X-Sweep*:* In this step of the iteration, all the rows are swept in the x-direction to obtain the values of $\bar{T}_{i,j}^{k+1/2}$.

$$\bar{T}_{i-1,j}^{k+1/2} - 2(1+\beta^2)\bar{T}_{i,j}^{k+1/2} + \bar{T}_{i+1,j}^{k+1/2} = -\beta^2(\bar{T}_{i,j+1}^{k} + \bar{T}_{i,j-1}^{k+1/2}) \tag{9}$$

The terms are arranged in such a way that the values from the previous row of current iteration are considered for the tri diagonal system of current row. This tri-diagonal system is then solved using Thomas algorithm to obtain the solution. This is carried out for (n_rows-2) times for Dirichlet boundary conditions.

Y-Sweep: In this step, we perform a column sweep similar to row sweep in the y-direction to obtain the values of $\bar{T}_{i,j}^{k+1}$.

$$\beta^2\bar{T}_{i,j-1}^{k+1} - 2(1+\beta^2)\bar{T}_{i,j}^{k+1} + \bar{T}_{i,j+1}^{k+1} = -(\bar{T}_{i+1,j}^{k+1/2} + \bar{T}_{i-1,j}^{k+1}) \tag{10}$$

Like X-sweep, the tri-diagonal system obtained is solved using Thomas algorithm. This is carried out for (n_columns-2) times for Dirichlet boundary conditions. The stencil for ADI is shown in Fig. 4 below

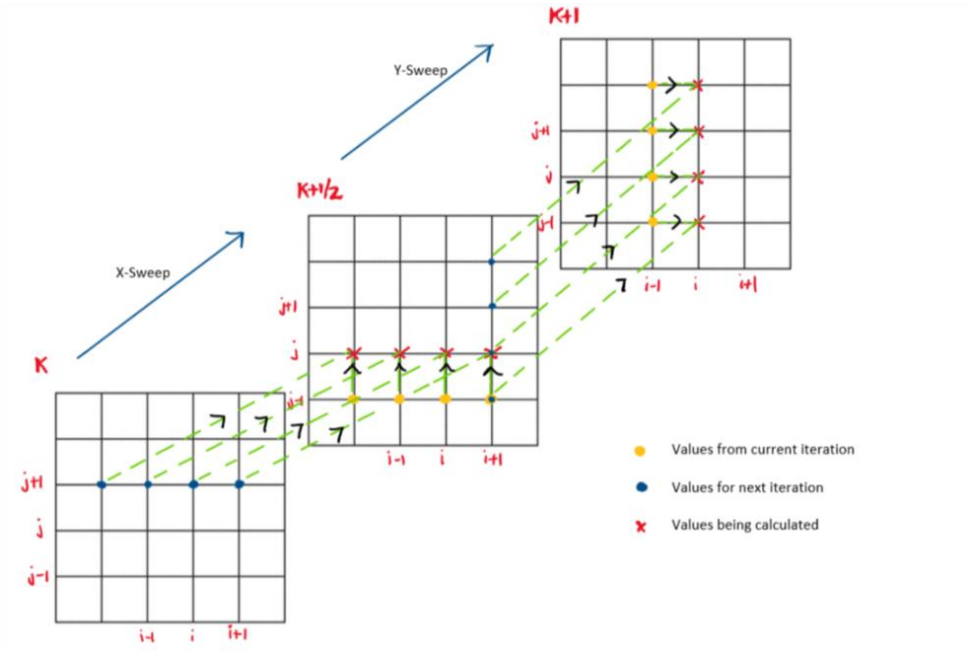

Figure 4 Stencil for ADI

The iterations are carried out until we reach convergence or till the maximum number of iterations is reached.

d) Analytical Solution

The analytical solution for the current problem is defined as

$$T(x,y) = T_A(x,y) + T_B(x,y) \tag{11}$$

$$T_A(x,y) = 2T_1 \sum_{m=1}^{\infty} \frac{1-\cos(m\pi)}{m\pi} \frac{\sinh\left(\frac{m\pi(W-y)}{L}\right)}{\sinh\left(\frac{m\pi W}{L}\right)} \sin\left(\frac{m\pi x}{L}\right)$$

$$T_B(x,y) = 2T_3 \sum_{m=1}^{\infty} \frac{1-\cos(m\pi)}{m\pi} \frac{\sinh\left(\frac{m\pi y}{L}\right)}{\sinh\left(\frac{m\pi W}{L}\right)} \sin\left(\frac{m\pi x}{L}\right)$$

After completing the above steps, we develop plots to visualize the temperature contour over the 2-D domain. Then, the discretized solution is compared to the analytical solution and steps required to minimize the error would be discussed. The order of the particular scheme is then determined using the logarithmic plot between error and step size. The effect of relaxation coefficient would be discussed by evaluating it against the number of iterations and error associated with every value. Later, we would compare the computational cost and accuracy of the schemes relative to one and the other.

DISCUSSION OF RESULTS

The above explained discretization was formulated and solved in MATLAB and the observations have been discussed below.

First, let us review our method of solving the 2-D heat conduction equation. The PDE is elliptic in nature. Since elliptic equations have no real characteristic curves, there is no meaningful sense of information propagation. Thus, elliptic equations are well suited for static problems. Elliptic partial differential equations only require boundary conditions for solving the problem. In our problem, the boundary conditions were clearly defined. Thus, all the necessary steps were taken to solve the heat conduction problem.

In SOR, as already discussed before, the value of ω is crucial in finding an accurate solution to our problem. Figure 5 shows the number of iterations required for the solution to converge by varying the value of ω for a 10x10 grid. The rate of convergence is the fastest at a value of $\omega = 1.4$. But looking at the rate of convergence is not the only step in determining the optimal value of ω. We must also look at the variation of error with ω, which is shown in Fig. 6.

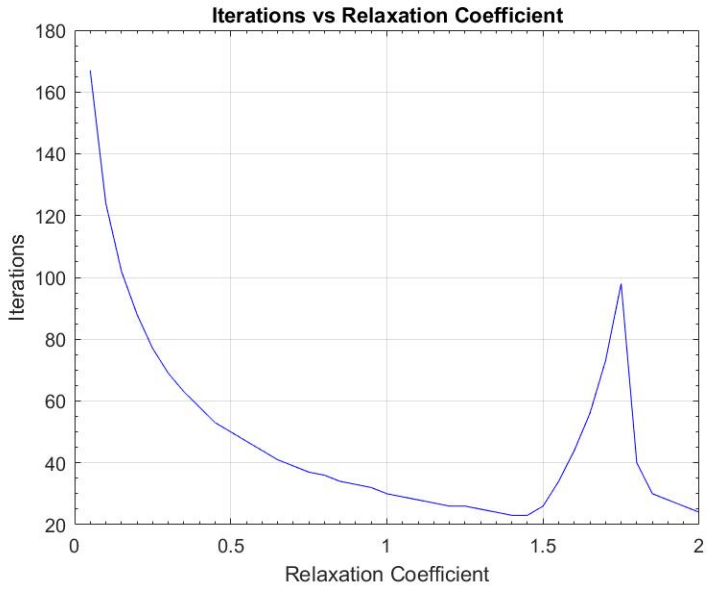

Figure 5 Plot showing variation of iterations with ω

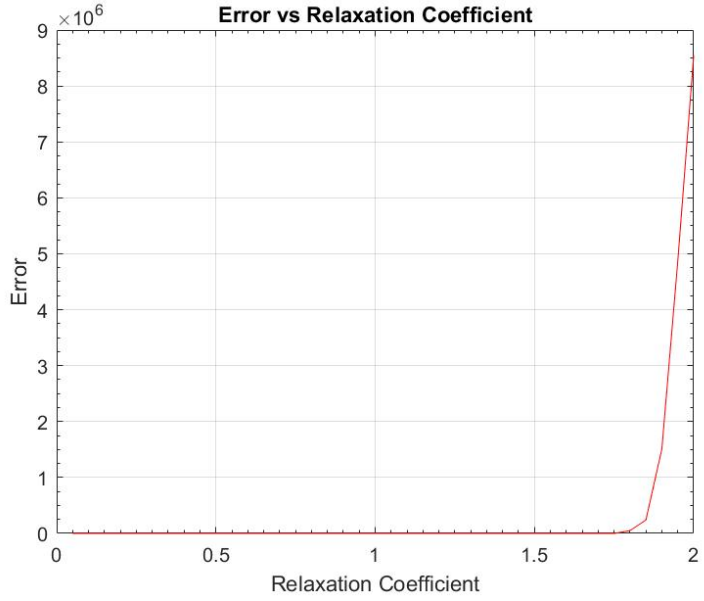

Figure 6 Plot showing variation of error with ω

We find that the value for the least number of iterations has a negligible error. Thus, the solution process was carried forward at a value of $\omega = 1.4$ for a 10x10 grid. Also, we see that as the value of ω approaches 2, the instability starts to increase, which is in congruence with the theory.

Now, if we choose a value of ω<1.4 for a 10x10 grid, there wouldn't be any increased error in the solution. But the time required to converge would increase. This is evident from Fig. 5, where we see that as we move towards the left of ω=1.4, the number of iterations starts to increase.

Similarly, if we increase the value of ω beyond 1.4, we begin to observe some oscillations in the convergence rate plot. Figure 7 shows the convergence rate plot for three different values of ω for a 10x10 grid to exemplify this discussion. The method with ω=0.7 takes 39 iterations to converge, whereas with ω=1.4, it takes only 23 iterations to converge. Also, we observe oscillations with increase in iterations when ω=1.9.

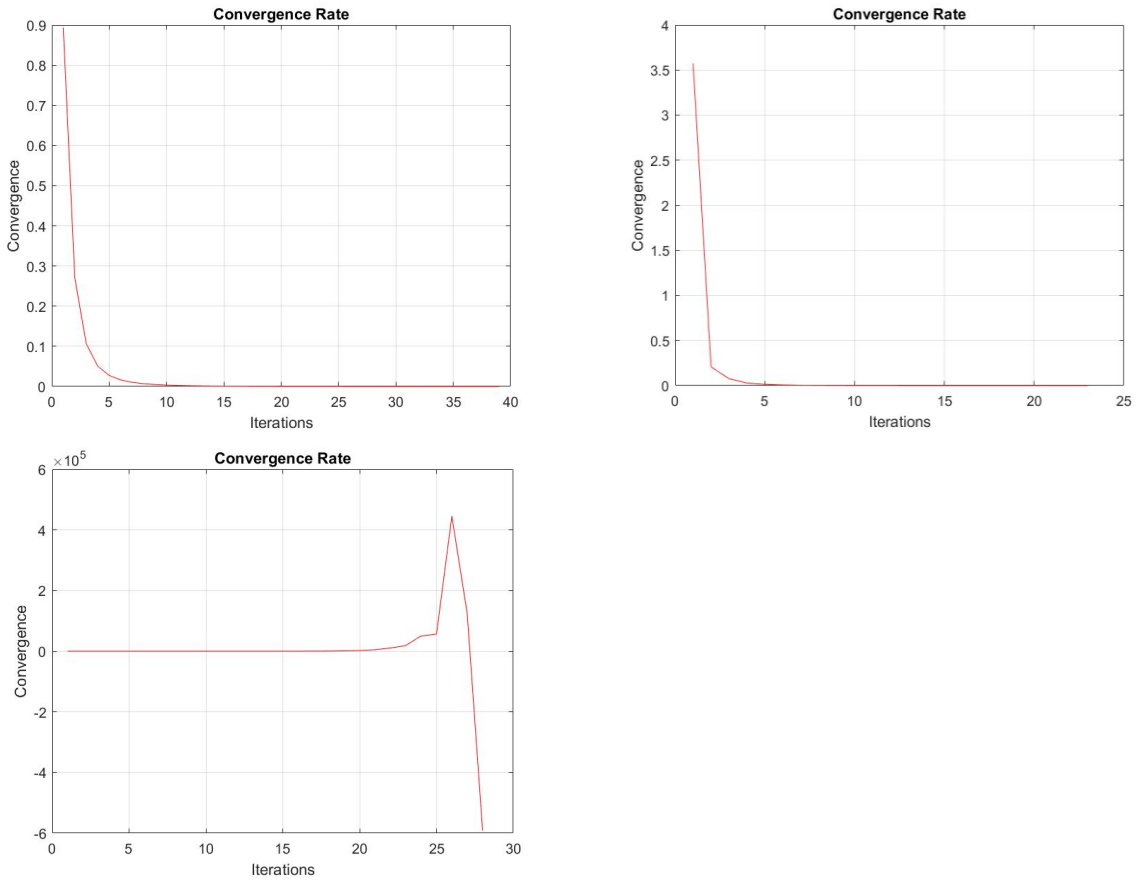

*Figure 7 Convergence Rate plots for **ω=0.7** (top-left), **ω=1.4** (top-right) and **ω=1.9** (bottom-left)*

No fine tuning has been done to ADI since it is an implicit method, which implies that it is unconditionally stable for any size of the grid. But, to obtain near accurate solution, we must decide on the grid size according to the threshold error to carry forward our analysis. Thus, we have chosen a grid of size 10x10 since its error is below the assumed threshold error of 0.02 and maintains uniformity in our analysis.

Figure 8 shows the contour plot of the analytical solution obtained using Eq. (11). This can be used as a benchmark to visually discern the differences in ADI and SOR schemes. To understand the contour plot, we must provide a brief discussion about the physics involved in this problem. The bottom surface of the plate is at a higher temperature which causes heat to flow upwards. Due to this, the temperatures decrease as we move from the bottom to the top of the plate. The two sides of the plate are held at the same temperature, which is lesser than the temperature at the top and bottom parts of the plate. Due to this, the temperature in the interior part of the

plate is always greater than the temperature on its sides causing the heat to flow sideways also. Therefore, the temperature starts to decrease as we move from the center to either side of the plate.

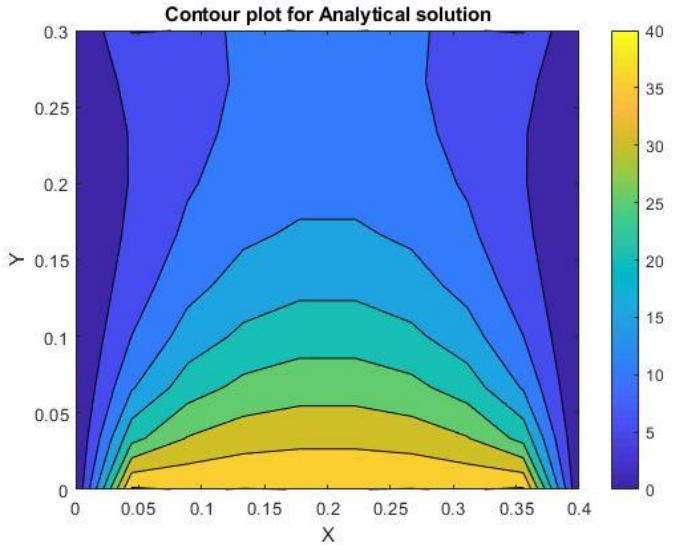

Figure 8 Contour plot showing Analytical solution

Figure 9 and Fig. 10 show the contour plots obtained after solving the problem using SOR and ADI methods respectively. In general, both the methods follow the physics of the problem very well. As we see, the region of analysis is very small when comparing both the methods to the analytical solution. In fact, on refining the mesh further, we move in the direction of analytical solution. To assess the performance of these methods, we must compare them against each other rather than comparing them to the analytical solution. By using other metrics such as error, computational cost, accuracy etc., we assess the usefulness of each method in a particular case.

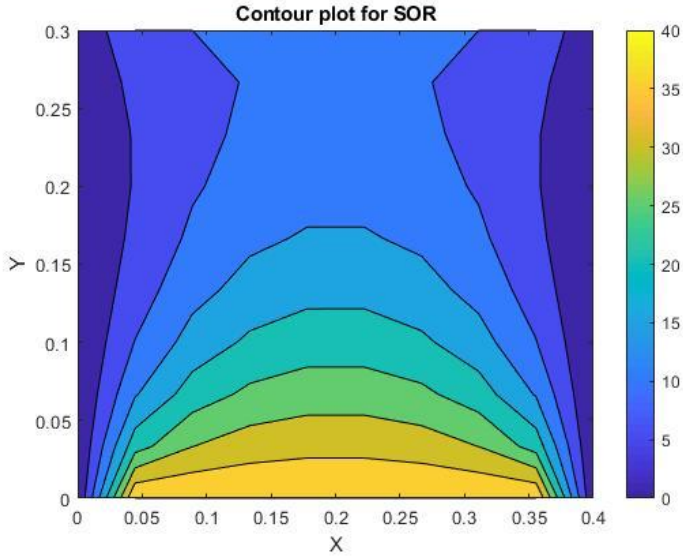

Figure 9 Contour plot showing SOR solution

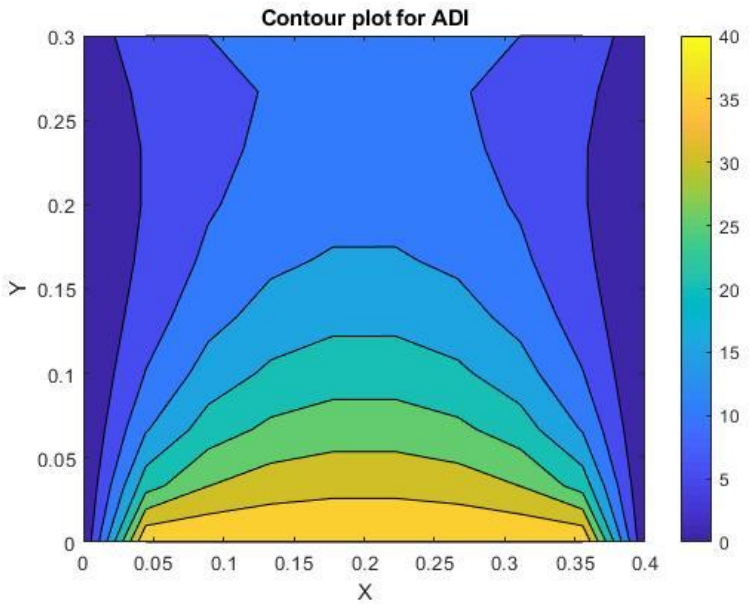

Figure 10 Contour plot showing ADI solution

Before that, let us confirm the order of discretization by using a logarithmic plot between error and step size. We calculate the slope of the logarithmic plot along each direction to confirm the order of discretization. Figures 11 and 12 show the error plots on logarithmic scale for both SOR and ADI respectively. The slopes were calculated from the data and are shown in Table 1.

	Δx (Δy *is constant*)	Δy (Δx *is constant*)
ADI	2.1632	1.8968
SOR	1.8775	2.0243

Table 1 Order of accuracy of SOR and ADI obtained from slopes of loglog plots of error vs step size

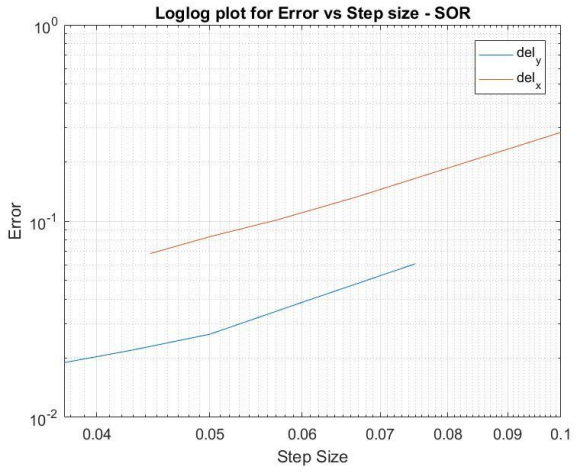

Figure 11 Logarithmic plot of error vs step size for SOR

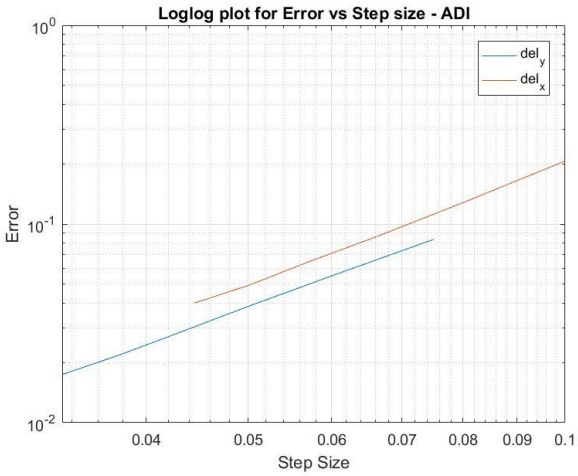

Figure 12 Logarithmic plot of error vs step size for ADI

Thus, we find that the schemes are second order accurate in space, showing congruence with the theory. The errors in Fig. 11 and 12 were found out using the expression

$$error = \frac{1}{JM * IM} \sum_{j=2}^{JM-1} \sum_{i=2}^{IM-1} (T_{(i,j)}^{computed} - T_{(i,j)}^{analytical})^2 \quad (12)$$

Table 2 shows the computational time required to find the converged solution for 10x10 and 20x20 grids using ADI and SOR schemes. We find that for both the grids, SOR outperforms ADI in computing the solution. For a 10x10 grid, ADI takes almost 2.85 times the time taken by SOR and for a 20x20 grid, it takes almost 1.22 times the taken by SOR. In SOR, we compute the value at a node during current iteration explicitly. We don't solve a set of equations to compute the values, which contrasts with the method of ADI. In ADI we solve different sets of equations multiple times during X-sweep and Y-sweep. Thus, we clearly see that a lot of computation is involved in ADI, thereby taking up higher computational resources.

	$10x10$ grid	**$20x20$ grid**
ADI	0.021261 s	0.017706 s
SOR	0.007475 s	0.014523 s

Table 2 Computational time of ADI and SOR for 10x10 and 20x20 grids

Table 3 shows the accuracy involved in the computation of discretized solution using SOR and ADI methods. For large step size (5x5 grid), the error associated with SOR and ADI are almost the same. As we decrease the step size, the error associated with ADI begins to decrease at a faster rate than the error associated with SOR. In SOR, the value at the current node is dependent on the values from current and previous iterations. But, in ADI, the current node gets the information from solving simultaneous equations at current iteration. This gives better stability to the problem, leading to smaller error. This effect gets pronounced as we increase the grid size.

	$5x5$ grid	**$7x7$ grid**	**$10x10$ grid**
ADI	0.06884	0.032348	0.016144
SOR	0.06826	0.03444	0.02236

Table 3 Error associated with ADI and SOR for different grids

SUMMARY

To solve an elliptic partial differential equation, we can approach it either by direct methods or iterative methods. In our analysis, we considered two iterative methods, SOR and ADI to solve for the temperature distribution due to steady state heat conduction in a rectangular plate. In SOR, the optimal value of ω was found using the metrics such as convergence rate and error. With the computed optimal value of ω, the SOR discretized solution was found. Then, the ADI discretized solution was found by solving a set of simultaneous equations along the X and Y

directions. Then, the contour plots were plotted for each of the discretized solutions obtained. Later, a case by case comparative study was done between ADI and SOR to determine the best possible method in a possible case, which has been discussed in the next section

CONCLUSION

There is no single scheme which would suit all the purposes required by a problem. Suitable analysis must be done on different schemes to find out the best suited one to our problem. In this case, we conducted an analysis on SOR and ADI, where ADI and SOR performed the same for large step size grids. As we decreased the step size, ADI outperformed SOR, but this came at a cost of additional computational expense. Thus, if the solution to a problem must be accurate, then a grid with small step size must be employed, thereby favoring ADI method for computation. If the problem is dependent on relative comparison amongst the nodes (Ex. Knowing the distribution in a domain), then SOR would be effective since our problem would not be accuracy-specific.

APPENDIX

Code for SOR:

```
%% Parameters

%alpha=11.234e-5;
%k=280;
L=0.4;
W=0.3;
T1=40;
T2=0;
T3=10;
T4=0;

Nx=7; % Number of elements in X-direction (Variation of columns in
matrix)
Ny=7; % Number of elements in Y-direction (Variation of rows in
matrix)
itr_max=500;
w_max=2;
w_d=0.05;
w_min=0.05;

% The PDE is elliptic, therefore the information is transferred
% instantaneously.

%% Non-dimensionalization

% x=L*x_nd
% y=L*x_nd
% T_nd=(T-T3)/(T1-T3)

T1_nd=(T1-T3)/(T1-T3);
T2_nd=(T2-T3)/(T1-T3);
T3_nd=(T3-T3)/(T1-T3);
T4_nd=(T4-T3)/(T1-T3);

del_x=L/(Nx-1);
del_xnd=del_x/L;

del_y=W/(Ny-1);
del_ynd=del_y/L;

%% Analytical Solution

syms m;
x=0:del_x:L;
y=W:-del_y:0;
[X,Y]=meshgrid(x,y);
```

```
Ta=((1-cos(m*pi))/(m*pi))*(sinh((m*pi*(W-
Y))/L)).*sin((m*pi*X)/L)/(sinh((m*pi*W)/L));
Tb=((1-
cos(m*pi))/(m*pi))*(sinh((m*pi*Y)/L)).*sin((m*pi*X)/L)/(sinh((m*pi*W)/
L));
T_an=vpa(2*T1*symsum(Ta,m,1,100)+2*T3*symsum(Tb,m,1,100),4);

%% SOR solution method

beta=del_xnd/del_ynd;
% iteration=zeros(1,(w_max-w_min)/w_d+1);
% error=zeros(1,(w_max-w_min)/w_d+1);
% z=0;
% for w=w_min:0.05:w_max
%     z=z+1;
%
%     % Initial Conditions Setup
%     T_nd=zeros(Ny,Nx);
%     T_nd(1,:)=T3_nd;
%     T_nd(Ny,:)=T1_nd;
%     T_nd(2:Ny-1,1)=T2_nd;
%     T_nd(2:Ny-1,Nx)=T4_nd;
%
%     T_bar=zeros(Ny,Nx);
%     T_bar(1,:)=T3_nd;
%     T_bar(Ny,:)=T1_nd;
%     T_bar(2:Ny-1,1)=T2_nd;
%     T_bar(2:Ny-1,Nx)=T4_nd;
%
%     for itr=1:itr_max
%         T_ndold=T_nd;
%         for i=Ny-1:-1:2
%             for j=2:Nx-1
%                 T_bar(i,j)=(1/(2*(1+beta^2)))*(T_bar(i,j-
1)+T_ndold(i,j+1)+beta^2*(T_ndold(i-1,j)+T_bar(i+1,j)));
%                 T_nd=T_ndold+w*(T_bar-T_ndold);
%             end
%         end
%
%         T_nd(1,2:Nx)=T3_nd;
%         T_nd(Ny,2:Nx)=T1_nd;
%         T_nd(:,1)=T2_nd;
%         T_nd(:,Nx)=T4_nd;
%
%         T=T3+T_nd*(T1-T3);
%         T_old=T3+T_ndold*(T1-T3);
%
%         if sum(sum((T(2:Ny-1,2:Nx-1)-T_old(2:Ny-1,2:Nx-
1)).^2))/sum(sum(T_old(2:Ny-1,2:Nx-1)))<1e-5
%             iteration(z)=itr;
%             error(z)=sum(sum((T(2:Ny-1,2:Nx-1)-T_an(2:Ny-1,2:Nx-
1)).^2))/(Nx*Ny);
```

```
%             break;
%         end
%
%         if itr==itr_max
%             iteration(z)=itr;
%             error(z)=sum(sum((T(2:Ny-1,2:Nx-1)-T_an(2:Ny-1,2:Nx-
1)).^2))/(Nx*Ny);
%         end
%     end
% end
%
% figure;
% plot(w_min:w_d:w_max,error,'r-')
% title('Error vs Relaxation Coefficient')
% xlabel('Relaxation Coefficient')
% ylabel('Error')
% set(gca,'XMinorTick','on','YMinorTick','on')
% grid on
%
% figure
% plot(w_min:w_d:w_max,iteration,'b-')
% title('Iterations vs Relaxation Coefficient')
% xlabel('Relaxation Coefficient')
% ylabel('Iterations')
% set(gca,'XMinorTick','on','YMinorTick','on')
% grid on

% Initial Conditions Setup
T_nd=zeros(Ny,Nx);
T_nd(1,:)=T3_nd;
T_nd(Ny,:)=T1_nd;
T_nd(2:Ny-1,1)=T2_nd;
T_nd(2:Ny-1,Nx)=T4_nd;

T_bar=zeros(Ny,Nx);
T_bar(1,:)=T3_nd;
T_bar(Ny,:)=T1_nd;
T_bar(2:Ny-1,1)=T2_nd;
T_bar(2:Ny-1,Nx)=T4_nd;

w=1.4;
tic
for itr=1:itr_max
    T_ndold=T_nd;
    for i=Ny-1:-1:2
        for j=2:Nx-1
            T_bar(i,j)=(1/(2*(1+beta^2)))*(T_bar(i,j-
1)+T_ndold(i,j+1)+beta^2*(T_ndold(i-1,j)+T_bar(i+1,j)));
            T_nd=T_ndold+w*(T_bar-T_ndold);
        end
    end
```

```
    T_nd(1,2:Nx)=T3_nd;
    T_nd(Ny,2:Nx)=T1_nd;
    T_nd(:,1)=T2_nd;
    T_nd(:,Nx)=T4_nd;

    T=T3+T_nd*(T1-T3);
    T_old=T3+T_ndold*(T1-T3);

    %conv(itr)=sum(sum((T(2:Ny-1,2:Nx-1)-T_old(2:Ny-1,2:Nx-
1)).^2))/sum(sum(T_old(2:Ny-1,2:Nx-1)));
    %iter(itr)=itr;

    if sum(sum((T(2:Ny-1,2:Nx-1)-T_old(2:Ny-1,2:Nx-
1)).^2))/sum(sum(T_old(2:Ny-1,2:Nx-1)))<1e-5
        err=sum(sum((T(2:Ny-1,2:Nx-1)-T_an(2:Ny-1,2:Nx-
1)).^2))/(Nx*Ny);
        break;
    end

    if itr==itr_max
        err=sum(sum((T(2:Ny-1,2:Nx-1)-T_an(2:Ny-1,2:Nx-
1)).^2))/(Nx*Ny);
    end
end
toc

% figure;
% plot(iter,conv,'r-')
% title('Convergence Rate')
% xlabel('Iterations')
% ylabel('Convergence')
% grid on

% figure;
% contourf(0:del_x:L,0:del_y:W,flipud(T))
% colorbar
% title('Contour plot for SOR')
% xlabel('X')
% ylabel('Y')
```

Code for ADI

```
%% Parameters

%alpha=11.234e-5;
%k=280;
L=0.4;
W=0.3;
T1=40;
T2=0;
T3=10;
```

```
T4=0;

Nx=10; % Number of elements in X-direction (Variation of columns in
matrix)
Ny=10; % Number of elements in Y-direction (Variation of rows in
matrix)
itr_max=500;

% The PDE is elliptic, therefore the information is transferred
% instantaneously.

%% Non-dimensionalization

% x=L*x_nd
% y=L*x_nd
% T_nd=(T-T3)/(T1-T3)

T1_nd=(T1-T3)/(T1-T3);
T2_nd=(T2-T3)/(T1-T3);
T3_nd=(T3-T3)/(T1-T3);
T4_nd=(T4-T3)/(T1-T3);

del_x=L/(Nx-1);
del_xnd=del_x/L;

del_y=W/(Ny-1);
del_ynd=del_y/L;

%% Initial Conditions Setup

T_nd=zeros(Ny,Nx);
T_nd(1,2:Nx)=T3_nd;
T_nd(Ny,2:Nx)=T1_nd;
T_nd(:,1)=T2_nd;
T_nd(:,Nx)=T4_nd;

%% ADI method for solution

beta=del_xnd/del_ynd;

ax=zeros(Nx,1);
bx=zeros(Nx,1);
cx=zeros(Nx,1);
dx=zeros(Nx,1);

ay=zeros(Ny,1);
by=zeros(Ny,1);
cy=zeros(Ny,1);
dy=zeros(Ny,1);

T_str=zeros(Ny,Nx);
T_str(1,2:Nx)=T3_nd;
```

```
T_str(Ny,2:Nx)=T1_nd;
T_str(:,1)=T2_nd;
T_str(:,Nx)=T4_nd;

tic
for itr=1:itr_max
    T_ndold=T_nd;

    % X-sweep
    for i=Ny-1:-1:2
        for jj=2:Nx-1
            ax(jj)=1;
            bx(jj)=-2*(1+beta^2);
            cx(jj)=1;
            dx(jj)=-beta^2*(T_str(i+1,jj)+T_ndold(i-1,jj));
        end

        dx(2)=dx(2)-T_nd(i,1);
        dx(Nx-1)=dx(Nx-1)-T_nd(i,Nx);

        dx=TDMA(2,Nx-1,ax,bx,cx,dx);

        for p=2:(Nx-1)
            T_str(i,p)=dx(p);
        end
    end

    T_str(1,2:Nx)=T3_nd;
    T_str(Ny,2:Nx)=T1_nd;
    T_str(:,1)=T2_nd;
    T_str(:,Nx)=T4_nd;


    % Y-sweep
    for j=2:Nx-1
        for ii=2:Ny-1
            ay(ii)=beta^2;
            by(ii)=-2*(1+beta^2);
            cy(ii)=beta^2;
            dy(ii)=-(T_str(ii,j+1)+T_nd(ii,j-1));
        end

        dy(2)=dy(2)-beta^2*T_nd(1,j);
        dy(Ny-1)=dy(Ny-1)-beta^2*T_nd(Ny,j);

        dy=TDMA(2,Ny-1,ay,by,cy,dy);

        for p=2:(Ny-1)
            T_nd(p,j)=dy(p);
        end
    end
```

```
    T_nd(1,2:Nx)=T3_nd;
    T_nd(Ny,2:Nx)=T1_nd;
    T_nd(:,1)=T2_nd;
    T_nd(:,Nx)=T4_nd;

    T=T3+T_nd*(T1-T3);
    T_old=T3+T_ndold*(T1-T3);

    if sum(sum((T(2:Ny-1,2:Nx-1)-T_old(2:Ny-1,2:Nx-
1)).^2))/sum(sum(T(2:Ny-1,2:Nx-1)))<1e-5
        break;
    end
end
toc
%% Analytical Solution

syms m;
x=0:del_x:L;
y=W:-del_y:0;
[X,Y]=meshgrid(x,y);
Ta=((1-cos(m*pi))/(m*pi))*(sinh((m*pi*(W-
Y))/L)).*sin((m*pi*X)/L)/(sinh((m*pi*W)/L));
Tb=((1-
cos(m*pi))/(m*pi))*(sinh((m*pi*Y)/L)).*sin((m*pi*X)/L)/(sinh((m*pi*W)/
L));
T_an=vpa(2*T1*symsum(Ta,m,1,100)+2*T3*symsum(Tb,m,1,100),4);

%% Error

err=sum(sum((T(2:Ny-1,2:Nx-1)-T_an(2:Ny-1,2:Nx-1)).^2))/(Nx*Ny);

% %% Visualization - T
%
% figure;
% contourf(0:del_x:L,0:del_y:W,flipud(T))
% colorbar
% title('Contour plot for ADI')
% xlabel('X')
% ylabel('Y')
%
% %% Visualization - T_an
%
% figure;
% contourf(0:del_x:L,0:del_y:W,flipud(T_an))
% colorbar
% title('Contour plot for Analytical solution')
% xlabel('X')
% ylabel('Y')
```