AEM – ADV19

Computational Fluid Dynamics

Coursework 1

Name: Zijun Fang

CID: 01811420

Contents

Question A:

(a) Discretise equation (1) in one dimension using a centred approximation.

For the Laplace equation:

$$\nabla^2 T = 0$$

The Discretise equation for the function above:

* Only **one-dimension**:

$$T_{xx} = f(x)$$

* Consider a region $-1 < x < 1$ split into N intervals so $\Delta x = 2/N$.

$$\frac{T_{i+1} - 2T_i + T_{i+1}}{\Delta x^2} = f_i$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1} - 2T_i + T_{i+1}}{\Delta x^2}$$

for every point i

where

$$T_i = T(x_i) \qquad\qquad f_i = f(x_i)$$

Question B

(b) The domain is a square in the region 0 ≤ x ≤ 1, 0 ≤ y ≤ 1 what boundary conditions are needed/possible on the edges? Justify your answer.

For the 2D Temperature flow:

$$\nabla^2 T = T_{xx} + T_{yy} = 0$$

* The Dirichlet Type of Boundary Conditions: at the time step $t = 0$

$$T(x=0, y=0) = 1 + x = 1 \qquad T(x=0, y=1) = 1$$

$$T(x=1, y=1) = \cos\left(6 * \frac{3}{2} * \pi * y\right) + 1 = 1 \quad T(x=1, y=0) = \cos\left(6 * \frac{3}{2} * \pi * y\right) + 1 = 2$$

* The Neumann Type of Boundary Conditions: at the time step $t = 0$

$@x = 0, y = 0$:

$$\frac{\partial T(0,0)}{\partial \hat{n}} = \frac{\partial(1)}{\partial \hat{n}} = \frac{\partial(1)}{\partial y} = 0; \frac{\partial T(0,0)}{\partial \hat{n}} = \frac{\partial(1+x)}{\partial x} = 1$$

$@x = 1, y = 0$:

$$\frac{\partial T(1,0)}{\partial \hat{n}} = \frac{\partial(1+x)}{\partial x} = 1; \frac{\partial T(1,0)}{\partial \hat{n}} = \frac{\partial(\cos\left(6 * \frac{3}{2} * \pi * y\right) + 1)}{\partial y} = 0$$

$@x = 0, y = 1$:

$$\frac{\partial T(0,1)}{\partial \hat{n}} = \frac{\partial(1)}{\partial y} = 0; \frac{\partial T(0,1)}{\partial \hat{n}} = \frac{\partial(1)}{\partial x} = 0$$

$@x = 1, y = 1$:

$$\frac{\partial T(1,1)}{\partial \hat{n}} = \frac{\partial(1)}{\partial x} = 0; \frac{\partial T(1,1)}{\partial \hat{n}} = \frac{\partial(\cos\left(6 * \frac{3}{2} * \pi * y\right) + 1)}{\partial y} = 0$$

* The Robin Type of Boundary Conditions: at the time step $t = 0$

For the general Robin BCs equation: $h(\partial\Omega) = T(\partial\Omega) + \frac{\partial T}{\partial \hat{n}}(\partial\Omega)$

$@x = 0, y = 0$:

$$h(x=0, y=0) = 1 + 0 \neq 1 + 1$$

$@x = 1, y = 0$:

$$h(x=1, y=0) = 2 + 1 \neq 2 + 0$$

$@x = 0, y = 1$:

$$h(x=0, y=1) = 1 + 0 = 0 + 1$$

$@x = 1, y = 1$:

$$h(x=0, y=0) = 1 + 0 = 1 + 0$$

Summary the upon calculations, according to final results, which **Dirichlet Boundary Conditions** are possible on the edges, the situations at both axes magnitude is equally fitted to the initial setting; however, for the last two methods, only a part of points are aligning to satisfied both sides results.

Question C

(c) Use TWO numerical schemes to discretize your equation and solve the PDE assuming the following boundary conditions $T = 1 \,@\, x = 0, T = \cos\left(6 * \frac{3}{2}\pi y\right) @\, x = 1, T = 1 \,@\, y = 1, T = (1 + x) \,@\, y = 0$. In the domain there is a point at $T = 1.5$ located at $x = 0.5, y = 0.5$ and one $T = 0.5$ located at $x = 0.2, y = 0.2$. Discuss the advantage of one scheme against the other and plot the iso-contours of the two-dimensional thermal field in both cases.

* The unit mesh length $\Delta\boldsymbol{x} = \Delta\boldsymbol{y} = \mathbf{0.01}$

The mesh number: $\mathbf{100} * \mathbf{100} = \mathbf{10^4}$

* Functional expressions of Jacobi and Gauss – Seidel:

To discretise the equation of Laplace of general:

$$\nabla^2 T(x,y) = T_{xx} + T_{yy} = 0$$

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

$$\Rightarrow \mathbf{4}\boldsymbol{T}_{\boldsymbol{i,j}}^{\boldsymbol{n}} = \boldsymbol{a}_{\mathbf{1}} T_{i-1,j}^{n} + \boldsymbol{a}_{\mathbf{2}} T_{i+1,j}^{n} + \boldsymbol{a}_{\mathbf{3}} T_{i,j-1}^{n} + \boldsymbol{a}_{\mathbf{4}} T_{i,j+1}^{n}$$

$$\Rightarrow \boldsymbol{a}_{\mathbf{1}} = \boldsymbol{a}_{\mathbf{2}} = \frac{\Delta y^2}{2(\Delta x^2 + \Delta y^2)};\ \boldsymbol{a}_{\mathbf{3}} = \boldsymbol{a}_{\mathbf{4}} = \frac{\Delta x^2}{2(\Delta x^2 + \Delta y^2)}$$

For $\Delta x = \Delta y$:

$$T_{i-1,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,j+1}^{n} - \mathbf{4}\boldsymbol{T}_{\boldsymbol{i,j}}^{\boldsymbol{n}} = 0$$

The Jacobi Scheme:

$$\boldsymbol{B} = \boldsymbol{D}; \boldsymbol{C} = \boldsymbol{L} + \boldsymbol{U}$$

$$\boldsymbol{B}x^{n+1} = \boldsymbol{A}x - \boldsymbol{C}x^{n}$$

$$\boldsymbol{D}x^{n+1} = \boldsymbol{A}x - (\boldsymbol{L} + \boldsymbol{U})x^{n}$$

$$\boldsymbol{T}_{\boldsymbol{i,j}}^{\boldsymbol{n+1}} = \frac{1}{4}(T_{i-1,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,j+1}^{n})$$

The Gauss – Seidel Scheme:

$$\boldsymbol{B} = \boldsymbol{L} + \boldsymbol{D}; \boldsymbol{C} = \boldsymbol{U}$$

$$\boldsymbol{B}x^{n+1} = \boldsymbol{A}x - \boldsymbol{C}x^{n}$$

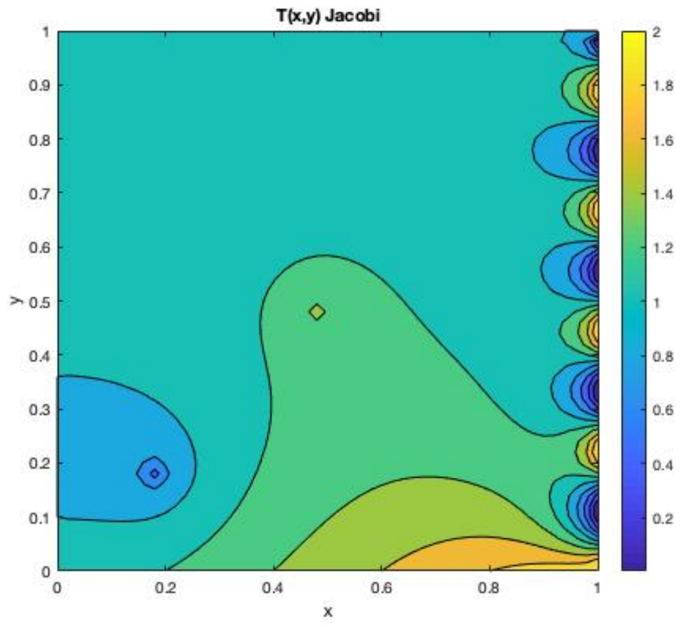
$$(\boldsymbol{L} + \boldsymbol{D})x^{n+1} = \boldsymbol{A}x - \boldsymbol{U}x^{n}$$

$$\boldsymbol{T}_{\boldsymbol{i,j}}^{\boldsymbol{n+1}} = \frac{1}{4}(T_{i-1,j}^{n+1} + T_{i+1,j}^{n} + T_{i,j-1}^{n+1} + T_{i,j+1}^{n})$$

* n: the time step, $(n + 1)$ is the forward time step;

* $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{U} + \boldsymbol{D}$**,** the matrices locate at lower, upper and diagonal position in original.

* Jacobi Method:



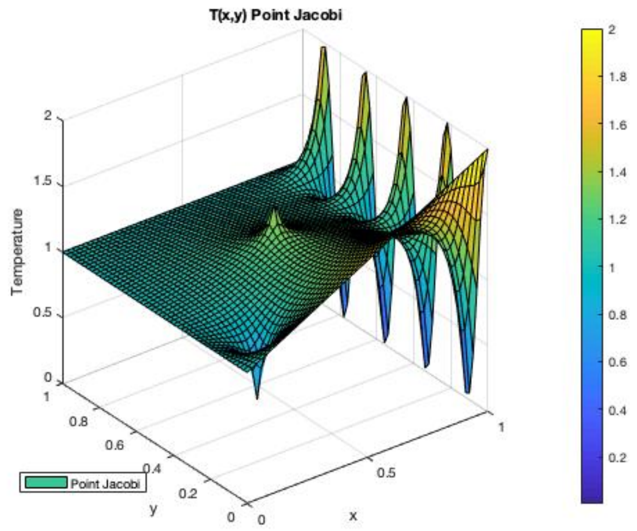
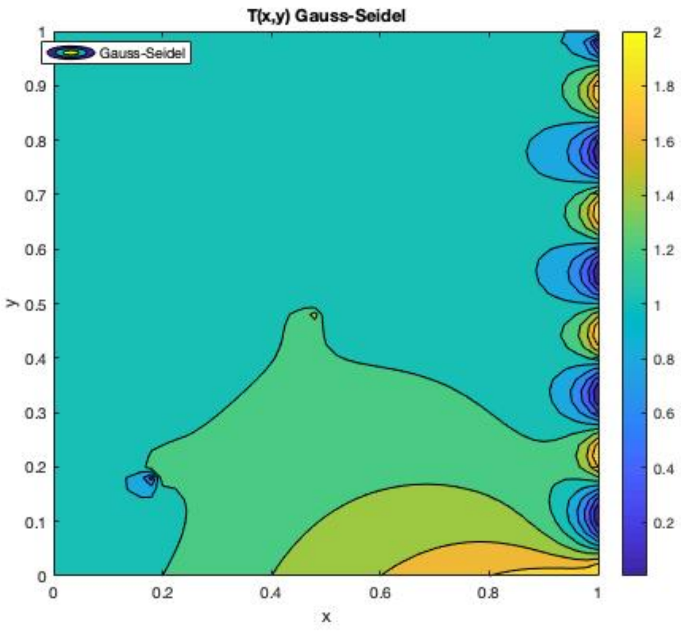


Figure 1A, B: Jacobi Method at 10000 meshing

* Gauss – Seidel Method:



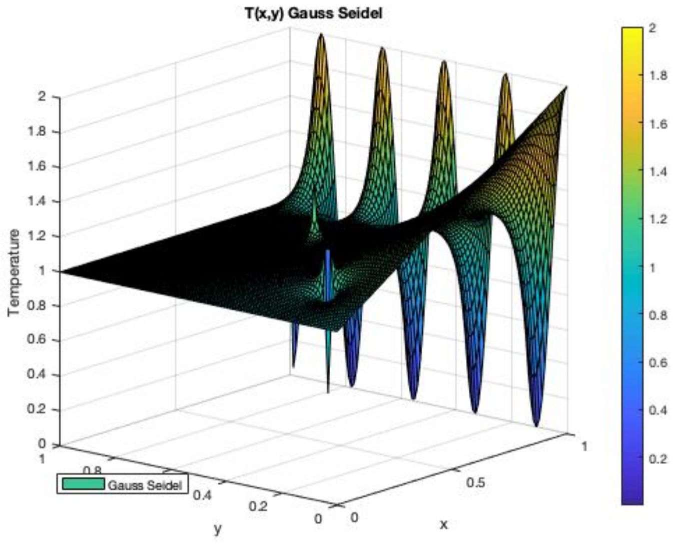


Figure 2A, B: Gauss – Seidel Method at 10000 meshing

Question D

(d) Using the same boundary conditions of question c, plot ϵ_{max} versus Δx on a set of log-log axes and comment on the form of the plot. Mesh with $\Delta x = \Delta y$, at least 3 different mesh sizes. Explain how you have decided the minimum $\Delta x = \Delta y$ of your computational domain.

* General error calculating results: with **Gauss – Seidel Method**

Set the analysis as 4 groups, comparing with the highest accurate meshing model ($\Delta x = \Delta y = 0.001$).

Mesh Size ($\Delta x = \Delta y$)	Number of Meshing Elements	Method	Timing / s	Error
0.1	10*10=100	Gauss – Seidel	0.404191	0.9750
0.01	100*100=10^4	Gauss – Seidel	0.901786	0.0374
0.005	200*200=4*10^4	Gauss – Seidel	3.034470	0.0193
0.002	500*500=2.5*10^5	Gauss – Seidel	73.105140	0.0101
0.001	1000*1000=10^6	Gauss – Seidel	891.457656	-

The error magnitude plotting diagram:

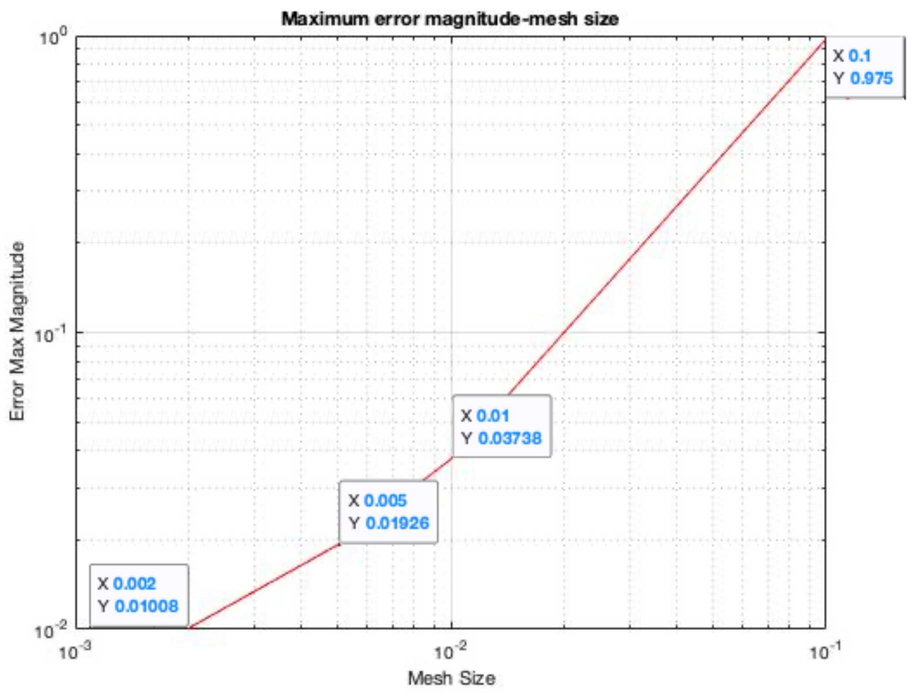


Figure 3: Max error plotting against the mesh size

For around the testing, we have selected different length of mesh, which including 0.1, 0.01, 0.005,0.002 and 0.001. As the number of mesh is increasing, the figures of plotting become more accurate. First, coming to discuss the mesh elements in size 0.1, as the Boundary Conditions (BCs) in the initial stage, which $T = 1.5$ located at $x = 0.5, y = 0.5$, if the mesh size is just equal to the location point distance, which also means that the physical solution could not be covered by the numerical meshing speed, the error of exact point value will not the same as provided in initial. In the next part of meshing, which combining the size 0.01 and 0.005 together, due to the similar issues are appearing in the matrix, when the mesh in this level, generally in right side BC, where the trigonometric temperature function will not stay as the accurate level as 0.002 can have, the tendency of the curve growth cannot be written in same words, some of the peak points (size: 0.01 and 0.005) are in low quality, not a visually smooth. When the final check to the plotting at 0.001 mesh size, generating the largest matrix in this test has been verified that the longest time duration of programming running, also known as the flip-flop computational time generation in different machines, while for the most amount of personal devices are in a Gigaflop level, the average time for the 10^6 number of elements generation will cost almost 1000 sec, in the accurate way, this type of meshing is more looking forward to using in the CFD analysis of this case; otherwise, in the time efficiency and computing expensive way, which will leave the project in a low efficiency level of the results outcome, also the cost of project is growing strongly, which the huge time difference than the mesh in $2.5*10^5$ level. So, based on the upon contents of analysing, the most suitable of size of meshing is 0.002, which the number of elements is $2.5*10^5$.

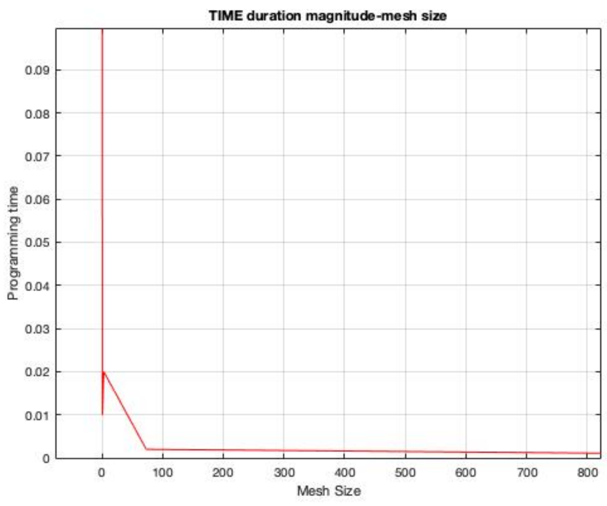
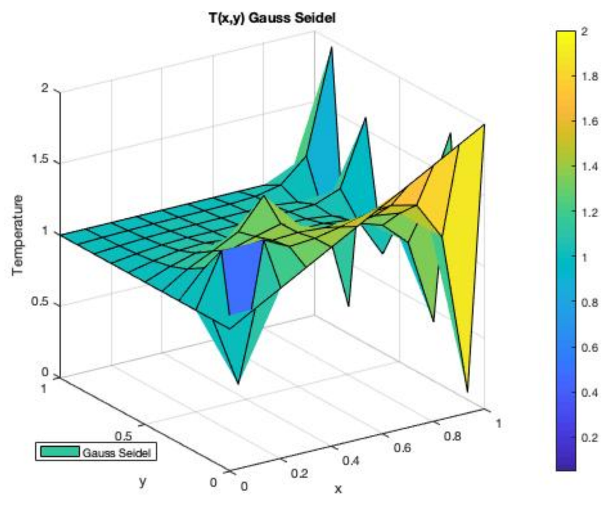
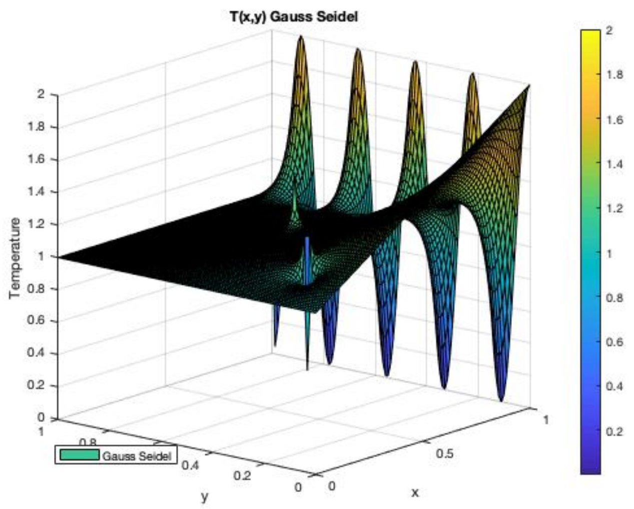


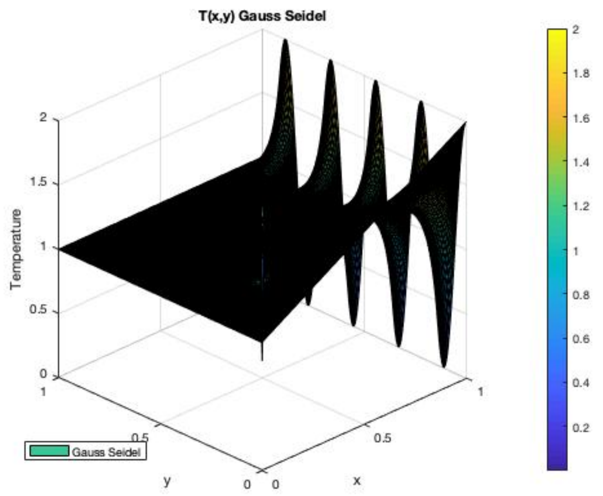
Figure 4: Time duration plotting against the mesh size (number)

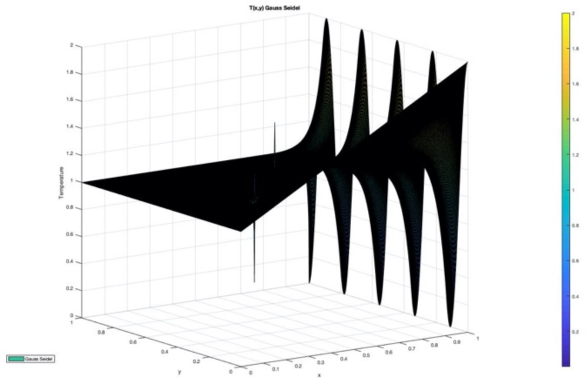
Appendix

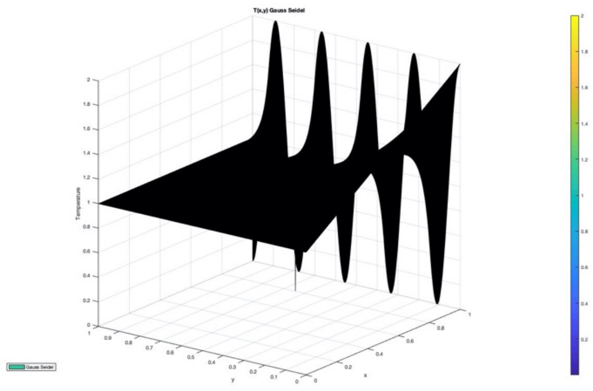
1. Meshing Elements in different mesh sizes








T(x,y) Gauss Seidel
Temperature
x
y
Gauss Seidel


T(x,y) Gauss Seidel
Temperature
x
y
Gauss Seidel

2. Programming

```
% Gauss Seidel to solve Laplace Eqn
%     T_xx + T_yy = 0
% with Dirichlet BCs.

%% clear the workspace
clc
clear

    % Parameters
    nx = 51;
    ny = nx;                           % number of space steps
    x=linspace(0,1,nx);                % x range
    y=linspace(0,1,ny);                % y range
    dx = 0.02;
    dy = dx;
    B=(dx/dy)^2;
    T_gs=zeros(nx,ny);                 % initialize T for Point Jacobi
    tol=1e-5;                          % error tolerance
    err=1;                             % error
    k=1;                               % iteration index
    omega=1.95;                        % (optimal) relaxation parameter
    %%
    % Boundary Conditions
    T_gs(1,:) = 1;%left
    T_gs(nx,:) = cos(6*1.5*pi*y)+1;%right
    T_gs(:,1) = 1+x;%bottom
    T_gs(:,end) = 1;%top
    Texact=T_gs;     %analytical

    %%
    % Time loop
    i=1;j=1;
    rms=0;
    while err>tol
        T_gsold=T_gs;
        for i=2:nx-1
            for j=2:ny-1
                %Gauss-Seidel
                T_gs(i,j)= (1-omega)*T_gsold(i,j) +(omega/(2*(1+B)))*(T_gs(i-1,j)+T_gsold(i+1,j)+B*(T_gs(i,j-1)+T_gsold(i,j+1)));
            end
        end
        %boundary conditions
        T_gs(1,:)= 1;%left
        T_gs(nx,:) = cos(6*1.5*pi*y)+1;%right
        T_gs(:,1) = 1+x;%bottom
        T_gs(:,end) = 1;%top
        T_gs(25,25)=1.5;
        T_gs(10,10)=0.5;

        err= max(max(abs(T_gs-T_gsold)));
        k=k+1;
```

```matlab
end
%%
% analytical solution
for iii=1:nx
    for jjj=1:ny
        A=0;
        for n=1:101
            if mod(n,2)==1
                A = A +((n*pi)^-2 * csch(2*n*pi) * sinh(n*pi*x(iii)) * cos(n*pi*y(jjj)));
            end
        end
        Texact(iii,jjj)=(x(iii)/4)-(4*A);
    end
end


% plot
figure(1)
figure,surf(x,y,T_gs');
xlabel('x');ylabel('y');zlabel('Temperature');
axis square;
title('T(x,y) Gauss-Seidel');
legend('Gauss Seidel','Analytical','location','sw');
colorbar
pts={[0.5 0.25] [0.5 0.75] [1.5 0.25] [1.5 0.75]};

figure(2)
contourf(x,y,T_gs');
xlabel('x');ylabel('y');
axis square;
title('T(x,y) Gauss-Seidel');
legend('Gauss-Seidel','Analytical','location','best');
colorbar
```

```matlab
% Point Jacobi Method to solve Laplace Eqn
%     T_xx + T_yy = 0
% with Dirichlet BCs.
%% clear the workspace
clc
clear
%%
clearvars
% Parameters
nx = 51;
ny = nx;                       % number of space steps
x=linspace(0,1,nx);            % x range
y=linspace(0,1,ny);            % y range

dx = 0.02;
dy = dx;                       % each unit length
B=(dx/dy)^2;
```

```
T_pj=zeros(nx,ny);              % initialize T
tol=1e-5;                       % error tolerance for convergence
err=1;                          % to check convergence
k=1;                            % iteration index

%%
% Boundary Conditions
T_pj(1,:) = 1;%left
T_pj(nx,:) = cos(6*1.5*pi*y)+1;%right
T_pj(:,1) = 1+x;%bottom
T_pj(:,end) = 1;%top
Texact=T_pj;     % initialize analytical solution



%%
while err>tol
    T_pjold=T_pj;

    for i=2:nx-1
        for j=2:ny-1
            %Point Jacobi
            T_pj(i,j)=(1/(2*((1+B))))*(T_pjold(i-
1,j)+T_pjold(i+1,j)+B*(T_pjold(i,j-1)+T_pjold(i,j+1)));
        end
    end
    %boundary conditions
    T_pj(1,:) = 1;%left
    T_pj(nx,:) = cos(6*1.5*pi*y)+1;%right
    T_pj(:,1) = 1+x;%bottom
    T_pj(:,end) = 1;%top
    T_pj(25,25)=1.5;
    T_pj(10,10)=0.5;

    err= norm(T_pj(:)-T_pjold(:),Inf);
    k=k+1;
end
rmspj=rms(T_pj(:)-T_pjold(:));
%%
% analytical solution
for iii=1:nx
    for jjj=1:ny
        A=0;
        for n=1:101
            if mod(n,2)==1
                A = A +((n*pi)^-2 * csch(2*n*pi) * sinh(n*pi*x(iii)) *
cos(n*pi*y(jjj)));
            end
        end
        Texact(iii,jjj)=(x(iii)/4)-(4*A);
    end
end

% plot
figure(1)
figure,surf(x,y,T_pj');
xlabel('x');ylabel('y');zlabel('Temperature');
axis square;
title('T(x,y) Point Jacobi');
legend('Point Jacobi','Analytical','location','sw');
colorbar
pts={[0.5 0.25] [0.5 0.75] [1.5 0.25] [1.5 0.75]};
```

```
figure(2)
contourf(x,y,T_pj');
hold on
xlabel('x');ylabel('y');
axis square;
title('T(x,y) Jacobi');
legend('Point-Jacobi','Analytical','location','best');
colorbar

%truncation error plotting
%Error through the step number
%step number from: 0 to 10^3

%% clear the workspace
clc
clear





%% 10

% Parameters
nx = 11;
ny = nx;                        % number of space steps
x=linspace(0,1,nx);             % x range
y=linspace(0,1,ny);             % y range
dx = 0.1;
dy = dx;
B=(dx/dy)^2;
T_gs_10=zeros(nx,ny);              % initialize T for Point Jacobi
tol=1e-5;                       % error tolerance
err=1;                          % error
k=1;                            % iteration index
omega=1.95;                     % (optimal) relaxation parameter
%%
% Boundary Conditions
T_gs_10(1,:) = 1;%left
T_gs_10(nx,:) = cos(6*1.5*pi*y)+1;%right
T_gs_10(:,1) = 1+x;%bottom
T_gs_10(:,end) = 1;%top
Texact_10=T_gs_10;     %analytical

%%
% Time loop
i=1;j=1;
rms=0;
while err>tol
    T_gsold_10=T_gs_10;
    for i=2:nx-1
        for j=2:ny-1
            %Gauss-Seidel
            T_gs_10(i,j)= (1-omega)*T_gsold_10(i,j)
+(omega/(2*(1+B)))*(T_gs_10(i-1,j)+T_gsold_10(i+1,j)+B*(T_gs_10(i,j-
1)+T_gsold_10(i,j+1)));
        end
    end
    %boundary conditions
    T_gs_10(1,:)= 1;%left
    T_gs_10(nx,:) = cos(6*1.5*pi*y)+1;%right
    T_gs_10(:,1) = 1+x;%bottom
    T_gs_10(:,end) = 1;%top
    T_gs_10(5,5)=1.5;
    T_gs_10(2,2)=0.5;

    err_10= max(max(abs(T_gs_10-T_gsold_10)));
    k=k+1;
end
%%
% analytical solution
for iii=1:nx
    for jjj=1:ny
        A=0;
        for n=1:101
            if mod(n,2)==1
                A = A +((n*pi)^-2 * csch(2*n*pi) * sinh(n*pi*x(iii)) *
cos(n*pi*y(jjj)));
            end
        end
        Texact(iii,jjj)=(x(iii)/4)-(4*A);
    end
end


% plot
figure,surf(x,y,T_gs_10');
xlabel('x');ylabel('y');zlabel('Temperature');
axis square;
title('T(x,y) Gauss Seidel');
legend('Gauss Seidel','Analytical','location','sw');
colorbar
pts={[0.5 0.25] [0.5 0.75] [1.5 0.25] [1.5 0.75]};










%% 100
% Parameters
nx = 101;
ny = nx;                       % number of space steps
x=linspace(0,1,nx);            % x range
y=linspace(0,1,ny);            % y range
dx = 0.01;
dy = dx;
B=(dx/dy)^2;
T_gs_100=zeros(nx,ny);             % initialize T for Point Jacobi
tol=1e-5;                      % error tolerance
err=1;                         % error
k=1;                           % iteration index
omega=1.95;                    % (optimal) relaxation parameter
%%
% Boundary Conditions
T_gs_100(1,:) = 1;%left
T_gs_100(nx,:) = cos(6*1.5*pi*y)+1;%right
```

```
T_gs_100(:,1) = 1+x;%bottom
T_gs_100(:,end) = 1;%top
Texact_100=T_gs_100;     %analytical

%%
% Time loop
i=1;j=1;
rms=0;
while err>tol
    T_gsold_100=T_gs_100;
    for i=2:nx-1
        for j=2:ny-1
            %Gauss-Seidel
            T_gs_100(i,j)= (1-omega)*T_gsold_100(i,j)
+(omega/(2*(1+B)))*(T_gs_100(i-1,j)+T_gsold_100(i+1,j)+B*(T_gs(i,j-
1)+T_gsold_100(i,j+1)));
        end
    end
    %boundary conditions
    T_gs_100(1,:)= 1;%left
    T_gs_100(nx,:) = cos(6*1.5*pi*y)+1;%right
    T_gs_100(:,1) = 1+x;%bottom
    T_gs_100(:,end) = 1;%top
    T_gs_100(50,50)=1.5;
    T_gs_100(20,20)=0.5;

    err_100= max(max(abs(T_gs_100-T_gsold_100)));
    k=k+1;
end
%%
% analytical solution
for iii=1:nx
    for jjj=1:ny
        A=0;
        for n=1:101
            if mod(n,2)==1
                A = A +((n*pi)^-2 * csch(2*n*pi) * sinh(n*pi*x(iii)) *
cos(n*pi*y(jjj)));
            end
        end
        Texact(iii,jjj)=(x(iii)/4)-(4*A);
    end
end


% plot
figure,surf(x,y,T_gs');
xlabel('x');ylabel('y');zlabel('Temperature');
axis square;
title('T(x,y) Gauss Seidel');
legend('Gauss Seidel','Analytical','location','sw');
colorbar
pts={[0.5 0.25] [0.5 0.75] [1.5 0.25] [1.5 0.75]};
```

```matlab
%% 500

% Parameters
nx = 501;
ny = nx;                        % number of space steps
x=linspace(0,1,nx);             % x range
y=linspace(0,1,ny);             % y range
dx = 0.002;
dy = dx;
B=(dx/dy)^2;
T_gs=zeros(nx,ny);              % initialize T for Point Jacobi
tol=1e-5;                       % error tolerance
err=1;                          % error
k=1;                            % iteration index
omega=1.95;                     % (optimal) relaxation parameter
%%
% Boundary Conditions
T_gs(1,:) = 1;%left
T_gs(nx,:) = cos(6*1.5*pi*y)+1;%right
T_gs(:,1) = 1+x;%bottom
T_gs(:,end) = 1;%top
Texact=T_gs;     %analytical

%%
% Time loop
i=1;j=1;
rms=0;
while err>tol
    T_gsold=T_gs;
    for i=2:nx-1
        for j=2:ny-1
            %Gauss-Seidel
            T_gs(i,j)= (1-omega)*T_gsold(i,j) +(omega/(2*(1+B)))*(T_gs(i-
1,j)+T_gsold(i+1,j)+B*(T_gs(i,j-1)+T_gsold(i,j+1)));
        end
    end
    %boundary conditions
    T_gs(1,:)= 1;%left
    T_gs(nx,:) = cos(6*1.5*pi*y)+1;%right
    T_gs(:,1) = 1+x;%bottom
    T_gs(:,end) = 1;%top
    T_gs(250,250)=1.5;
    T_gs(100,100)=0.5;

    err= max(max(abs(T_gs-T_gsold)));
    k=k+1;
end
%%
% analytical solution
for iii=1:nx
    for jjj=1:ny
        A=0;
        for n=1:101
            if mod(n,2)==1
                A = A +((n*pi)^-2 * csch(2*n*pi) * sinh(n*pi*x(iii)) *
cos(n*pi*y(jjj)));
            end
        end
        Texact(iii,jjj)=(x(iii)/4)-(4*A);
    end
end


% plot
figure,surf(x,y,T_gs');
xlabel('x');ylabel('y');zlabel('Temperature');
axis square;
title('T(x,y) Gauss Seidel');
legend('Gauss Seidel','Analytical','location','sw');
colorbar
pts={[0.5 0.25] [0.5 0.75] [1.5 0.25] [1.5 0.75]};


%%error calculator
%% clear workspace
clear;
clc;
%% input loading matrics
load('T_gs_10');
load('T_gs_50');
load('T_gs_100');
load('T_gs_200');
load('T_gs_500');
load('T_gs_1000');

A=T_gs_10;
B=T_gs_100;
C=T_gs_200;
D=T_gs_500;
E=T_gs_1000;



I=E(1:100:1001,1:100:1001);
H=D(1:50:501,1:50:501);
G=C(1:20:201,1:20:201);
J=B(1:10:101,1:10:101);



err1=abs(max(max(A-I)))/36;
err2=abs(max(max(A-H)))/18;
err3=abs(max(max(A-G)))/9;
err4=abs(max(max(A-J)))*3;

% plot error magnitude against the mesh size

ERR=[err4,err3,err2,err1];
SIZE=[0.1,0.01,0.005,0.002];
loglog(SIZE,ERR,'r','linewidth',1);
xlabel('Mesh Size');
ylabel('Error Max Magnitude');
title('Maximum error magnitude-mesh size');
grid on
```