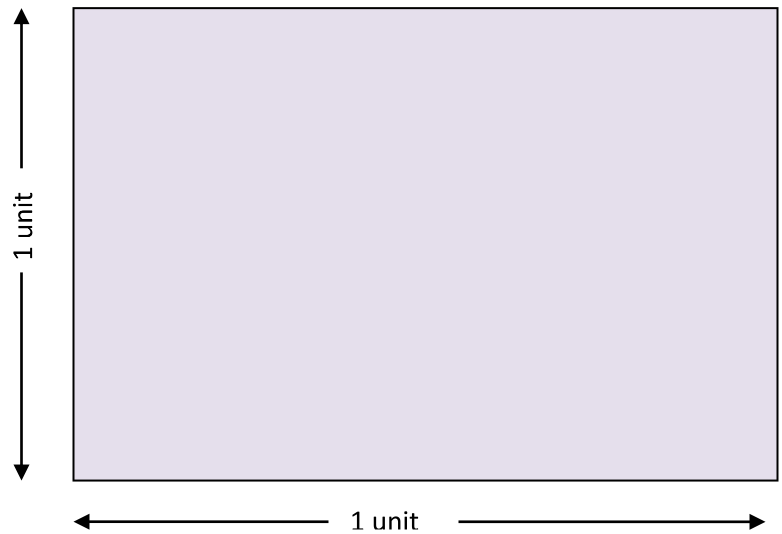
Advanced Computational Fluid Dynamics (AM6513)

Report on Assignment 3

Kumar Saurabh (MA14M004)

Problem definition:



$$\boldsymbol{u = \ \gamma * \cos\left(\pi * \left(x - \frac{1}{2}\right)\right) * \sin\left(\pi * \left(y - \frac{1}{2}\right)\right)}$$

$$\boldsymbol{v = \ -\gamma * \sin\left(\pi * \left(x - \frac{1}{2}\right)\right) * \cos\left(\pi * \left(x - \frac{1}{2}\right)\right)}$$

$\gamma = 1.0$

Solve the Convection diffusion equation using:

1. Convection term with Central Difference and Crank-Nicholson Scheme.
2. Convection term with first order upwind and Crank-Nicholson Scheme.

using :

1. Gauss Seidel Method without relaxation
2. Gauss Seidel Method with relaxation.
3. Conjugate Gradient Method
4. Bi-conjugate Gradient Stabilized Method

Governing Equation:

$$\boldsymbol{\frac{\partial \emptyset}{\partial t} + \frac{\partial (u\emptyset)}{\partial x} + \frac{\partial (v\emptyset)}{\partial y} = \ \alpha \left(\frac{\partial^2 \emptyset}{\partial x^2} + \frac{\partial^2 \emptyset}{\partial y^2}\right)}$$

$\alpha = 1$

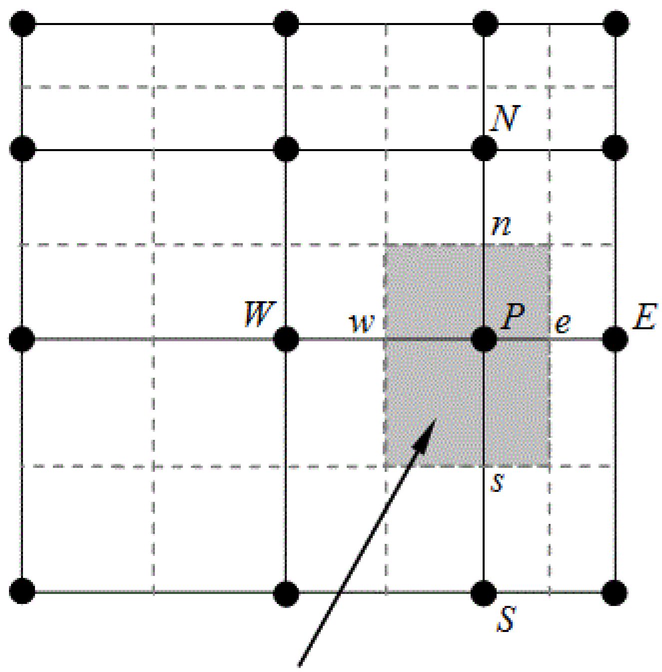
Initial condition:

$$\emptyset = \cos\left(\pi * \left(x - \frac{1}{2}\right)\right) * \cos\left(\pi * \left(y - \frac{1}{2}\right)\right)$$

Boundary Condition:

$$\emptyset = 0 \; for \; all \; boundaries$$

Numerical Formulation:



Control volume

- Discretizing the above equation with Finite Volume method over the control volume:

$$\int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial \emptyset}{\partial t} * dt * dy * dx + \int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial (u\emptyset)}{\partial x} * dt * dy * dx + \int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial (v\emptyset)}{\partial y} * dt * dy * dx = \alpha\left(\int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial^2 \emptyset}{\partial x^2} * dt * dy * dx + \int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial^2 \emptyset}{\partial y^2} * dt * dy * dx\right)$$

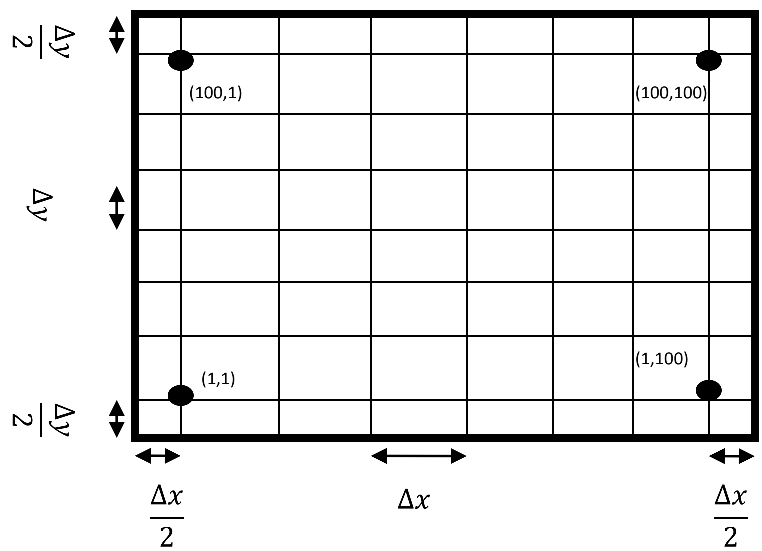
- On solving we get the equation of the form:

$$a_P\emptyset_P = \theta(a_E\emptyset_E + a_W\emptyset_W + a_N\emptyset_N + a_S\emptyset_S) + (1-\theta) * (a_E\emptyset_E^o + a_W\emptyset_W^o + a_N\emptyset_N^o + a_S\emptyset_S^o + a_P^{'}\emptyset_P^o)$$

where:

- $a_P = a_P^o + \theta(a_E + a_W + a_N + a_S + (F_e - F_w) + (F_n - F_s) - S_P)$

- $a_P^o = \frac{\Delta x * \Delta y}{\Delta t}$
- $a_P^{'} = a_P^o - (1 - \theta) * (a_E + a_W + a_N + a_S + (F_e - F_w) + (F_n - F_s) - S_P)$
- $F_e = u_e * \Delta y$
- $F_w = u_w * \Delta y$
- $F_n = v_n * \Delta x$
- $F_s = v_s * \Delta x$
- $D_e = D_w = \alpha * \frac{\Delta y}{\Delta x}$
- $D_n = D_s = \alpha * \frac{\Delta x}{\Delta y}$
- $\theta = \frac{1}{2}$ $for\ Crank\ Nicholson\ scheme$



Convection term with Central difference scheme	**Convection term with first order upwind**
For the node point in the interior domain: • $a_E = D_e - \frac{F_e}{2}$ • $a_W = D_w + \frac{F_w}{2}$ • $a_N = D_n - \frac{F_n}{2}$ • $a_S = D_s + \frac{F_s}{2}$ • $S_p = 0$	**For the node point in the interior domain:** • $a_E = D_e + \max(-F_e, 0)$ • $a_W = D_w + \max(F_w, 0)$ • $a_N = D_n + \max(-F_n, 0)$ • $a_S = D_s + \max(F_s, 0)$ • $S_p = 0$
For the node point of the bottom face except the corner points:	**For the node point of the bottom face except the corner points:** • $a_E = D_e + \max(-F_e, 0)$

• $a_E = D_e - \frac{F_e}{2}$ • $a_W = D_w + \frac{F_w}{2}$ • $a_N = D_n - \frac{F_n}{2}$ • $a_S = 0$ • $S_p = -2 * D_s$	• $a_W = D_w + \max(F_w, 0)$ • $a_N = D_n + \max(-F_n, 0)$ • $a_S = 0$ • $S_p = -2 * D_s$
For the node point of the top face except the corner points: • $a_E = D_e - \frac{F_e}{2}$ • $a_W = D_w + \frac{F_w}{2}$ • $a_N = 0$ • $a_S = D_s + \frac{F_s}{2}$ • $S_p = -2 * D_n$	**For the node point of the bottom face except the corner points:** • $a_E = D_e + \max(-F_e, 0)$ • $a_W = D_w + \max(F_w, 0)$ • $a_N = 0$ • $a_S = D_s + \max(F_s, 0)$ • $S_p = -2 * D_n$
For the node point of the right face except the corner points: • $a_E = 0$ • $a_W = D_w + \frac{F_w}{2}$ • $a_N = D_n - \frac{F_n}{2}$ • $a_S = D_s + \frac{F_s}{2}$ • $S_p = -2 * D_e$	**For the node point of the right face except the corner points:** • $a_E = 0$ • $a_W = D_w + \max(F_w, 0)$ • $a_N = D_n + \max(-F_n, 0)$ • $a_S = D_s + \max(F_s, 0)$ • $S_p = -2 * D_e$
For the node point of the left face except the corner points: • $a_E = D_e - \frac{F_e}{2}$ • $a_W = 0$ • $a_N = D_n - \frac{F_n}{2}$ • $a_S = D_s + \frac{F_s}{2}$ • $S_p = -2 * D_w$	**For the node point of the left face except the corner points:** • $a_E = D_e + \max(-F_e, 0)$ • $a_W = 0$ • $a_N = D_n + \max(-F_n, 0)$ • $a_S = D_s + \max(F_s, 0)$ • $S_p = -2 * D_w$
For the corner point (1,1): • $a_E = D_e - \frac{F_e}{2}$ • $a_W = 0$ • $a_N = D_n - \frac{F_n}{2}$ • $a_S = 0$ • $S_p = -2 * (D_w + D_s)$	**For the corner point (1,1):** • $a_E = D_e + \max(-F_e, 0)$ • $a_W = 0$ • $a_N = D_n + \max(-F_n, 0)$ • $a_S = 0$ • $S_p = -2 * (D_w + D_s)$
For the corner point (1,100): • $a_E = 0$ • $a_W = D_w + \frac{F_w}{2}$ • $a_N = D_n - \frac{F_n}{2}$	**For the corner point (1,100):** • $a_E = 0$ • $a_W = D_w + \max(F_w, 0)$ • $a_N = D_n + \max(-F_n, 0)$

• $a_S = 0$ • $S_p = -2 * (D_e + D_s)$	• $a_S = 0$ • $S_p = -2 * (D_e + D_s)$
For the corner point (100,1): • $a_E = D_e - \frac{F_e}{2}$ • $a_W = 0$ • $a_N = 0$ • $a_S = D_s + \frac{F_s}{2}$ • $S_p = -2 * (D_w + D_n)$	**For the corner point (100,1):** • $a_E = D_e + \max(-F_e, 0)$ • $a_W = 0$ • $a_N = 0$ • $a_S = D_s + \max(F_s, 0)$ • $S_p = -2 * (D_w + D_n)$
For the corner point (100,100): • $a_E = 0$ • $a_W = D_w + \frac{F_w}{2}$ • $a_N = 0$ • $a_S = D_s + \frac{F_s}{2}$ • $S_p = -2 * (D_e + D_n)$	**For the corner point (100,100):** • $a_E = 0$ • $a_W = D_w + \max(F_w, 0)$ • $a_N = 0$ • $a_S = D_s + \max(-F_s, 0)$ • $S_p = -2 * (D_e + D_n)$

Algorithm:

The equation can be written in the form of:

$$\boldsymbol{a_P \emptyset_P - \theta(a_E \emptyset_E + a_W \emptyset_W + a_N \emptyset_N + a_S \emptyset_S)}$$
$$\boldsymbol{= (1 - \theta) * (a_E \emptyset_E^o + a_W \emptyset_W^o + a_N \emptyset_N^o + a_S \emptyset_S^o + a_P' \emptyset_P^o)}$$

Where the right side is known and left side is unknown.

Thus solving above equation is equivalent of solving AX = B.

Steps:

1. *Calculate the number of grid points.*
2. *Calculate Δx, Δy, Δt.*
3. *Form the coefficient matrix.*
4. *Solve the system of linear equation through various iterative methods whose algorithm is given later.*
5. *Plot the results.*

Gauss Seidel Iteration without relaxation:

- Input: Matrix A and Matrix B.
- Output: Matrix X.

Steps:

1. *Choose an initial guess X to the solution.*
2. *Repeat until convergence (error of the order of 1e-6)*
3. *For i from 1 to n do*
4. *Sum = 0*
5. *For j = 1 to n do*
6. *If $j \neq i$ do*
7. *$Sum = Sum + A_{ij}*X_j$*
8. *End if*
9. *End (j - loop)*
10. *$X_i = (B_i - Sum)/A_i$*
11. *End (i – loop)*
12. *Check if convergence is reached*
13. *End or repeat.*

Gauss Seidel Iteration with relaxation:

- Input: Matrix A and Matrix B and relaxation factor (α)
- Output: Matrix X.

Steps:

1. *Choose an initial guess X to the solution.*
2. *Repeat until convergence (error of the order of 1e-6)*
3. *For i from 1 to n do*
4. *Sum = 0*
5. *For j = 1 to n do*
6. *If $j \neq i$ do*
7. *$Sum = Sum + A_{ij}*X_j$*
8. *End if*
9. *End (j - loop)*
10. *$X_i = X_i + \alpha*(((B_i - Sum)/A_i) - X_i)$*
11. *End (i – loop)*
12. *Check if convergence is reached*
13. *End or repeat.*

- $0 < \alpha < 1$: Under relaxation factor.
- $1 < \alpha < 2$: Over relaxation factor.

Conjugate Gradient Method:

- Input: Matrix A and matrix B
- Output: Matrix X

Steps:

1. $r_0 = B - A*X_0$
2. $p_0 = r_0$
3. $k = 0$
4. *repeat:*
5. $\alpha_K = \frac{r_K^T * r_K}{p_K^T * A * p_K}$
6. $X_{(K+1)} = X_K + \alpha_K P_K$
7. $r_{(K+1)} = r_k - \alpha_K A p_K$
8. *If* r_{K+1} *is sufficiently small, then exit the loop*
9. $\beta_K = r_{K+1}^T * \frac{r_{K+1}}{r_K^T * r_K}$
10. $p_{(K+1)} = r_{(K+1)} + \beta_K * p_K$
11. *end repeat.*

Bi -conjugate Stabilized Gradient Method:

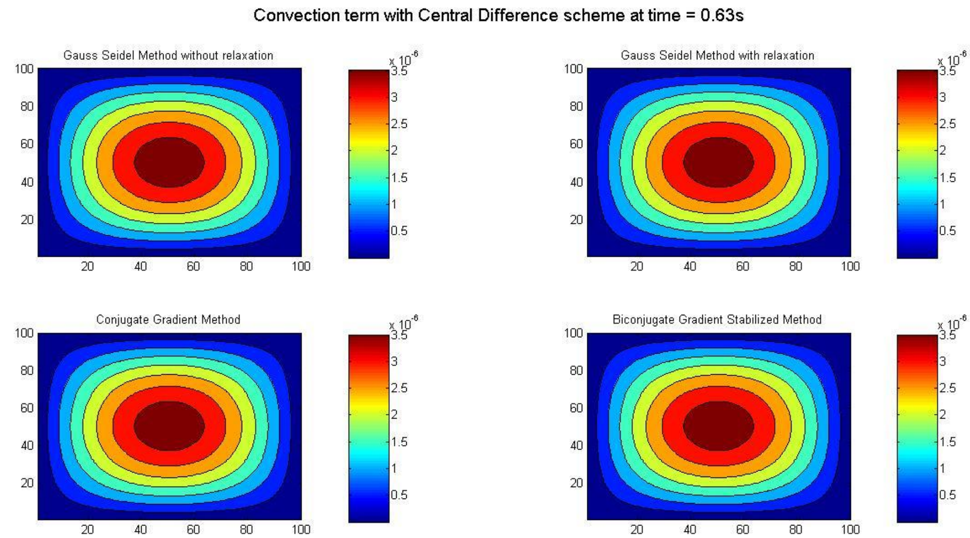
- Input: Matrix A and matrix B
- Output: Matrix X

Steps:

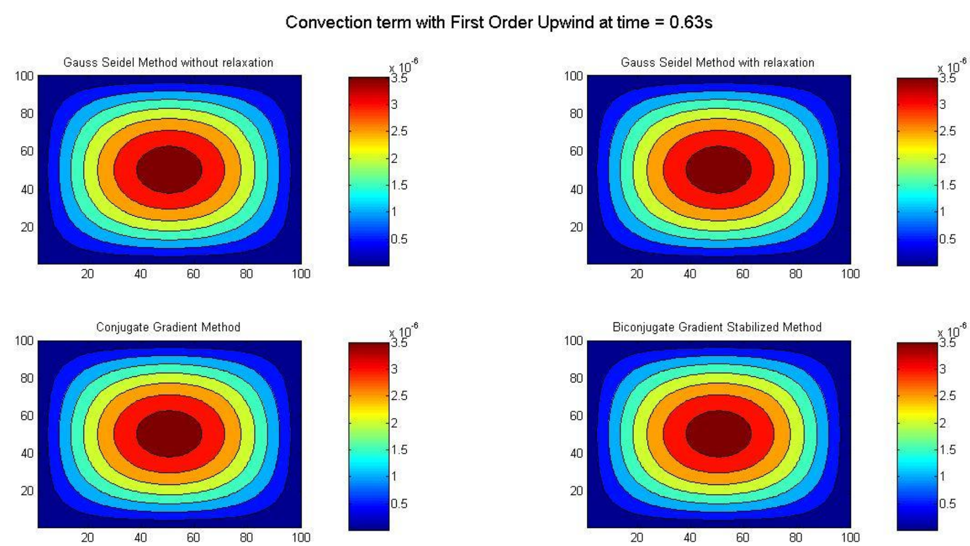
1. $r_0 = B - A*X_0$
2. $\hat{r}_0 = r_0$
3. $\rho_0 = \alpha = \omega_0 = 1$
4. $v_0 = p_0 = 0$
5. *For* $i = 1, 2, 3, \ldots$
6. $\rho_i = (\hat{r}_0, r_{i-1})$
7. $\beta = (\rho_i/\rho_{i-1})(\alpha/\omega_{i-1})$
8. $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1} v_{i-1})$
9. $v_i = A p_i$
10. $\alpha = \rho_i/(\hat{r}_0, v_i)$
11. $s = r_{i-1} - \alpha v_i$
12. $t = As$
13. $\omega_i = (t, s)/(t, t)$
14. $x_i = x_{i-1} + \alpha p_i + \omega_i s$
15. *If* x_i *is accurate enough then quit*
16. $r_i = s - \omega_i t$

Results:

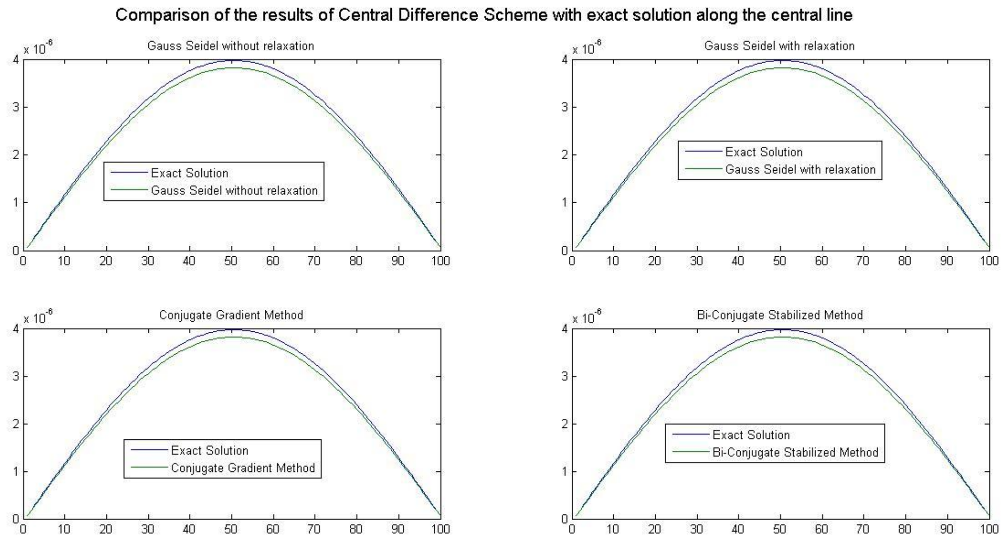
- **Convection term with Central Difference Scheme:**



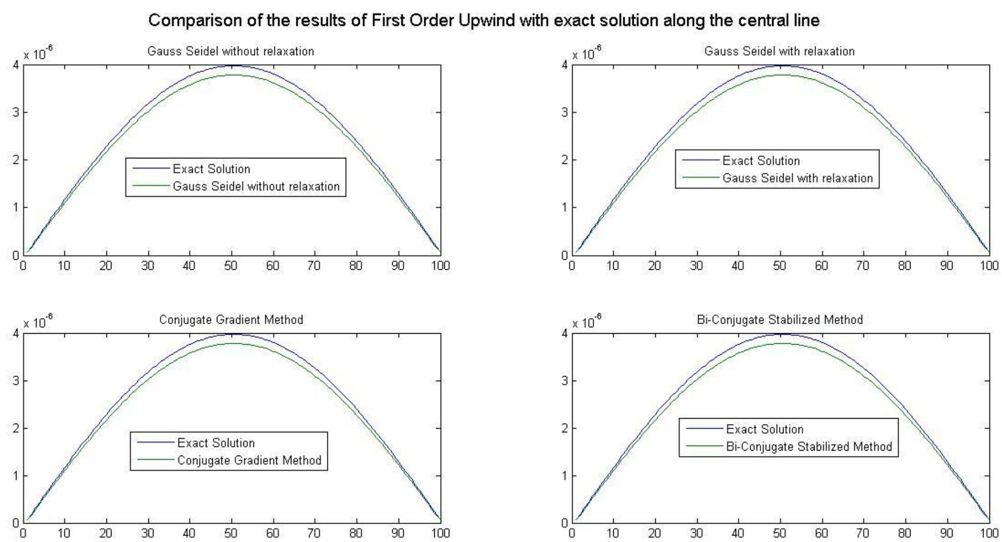
- **Convection term with first order upwind:**



- **Comparison of the result of central difference with exact solution**



- **Comparison of the result of First order upwind with exact solution**



Output File:

Convection term with Central Difference Scheme				
	Number of iterations	Time to reach steady state	Total time elapsed	Total CPU time
Gauss Siedel Iteration without relaxation	97902	0.63	217.301043	214.033372
Gauss Siedel Iteration with relaxation	12852	0.63	32.5757506	32.4326079
Conjugate Gradient Method	1592	0.63	8.98921856	9.3912602
Biconjugate Gradient Stabilized Method	1379	0.63	11.61479962	12.1524779

Convection term with First Order Upwind				
	Number of iterations	Time to reach steady state	Total time elapsed	Total CPU time
Gauss Siedel Iteration without relaxation	98162	0.63	215.7189809	214.7509766
Gauss Siedel Iteration with relaxation	12891	0.63	33.0764636	33.0254117
Conjugate Gradient Method	4492	0.63	16.02801853	16.8169078
Biconjugate Gradient Stabilized Method	3053	0.63	19.30701932	20.592132

Appendix:

Matlab Code:

File 1: MA14M004_Main File.m

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% Main File: Run this file
close all;


%% Code for Convection term with Central Difference Scheme
clear all;
clc;

% Gauss Seidel Iteration without relaxation
[X1, time, iteration, t1, t2] = MA14M004_CDS(1);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B4:B4');
```

```
xlswrite('outputMA14M004_asign3.xlsx',time,'C4:C4');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D4:D4');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E4:E4');

% Gauss Seidel Iteration with Relaxation
[X2, time, iteration, t1, t2] = MA14M004_CDS(2);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B6:B6');
xlswrite('outputMA14M004_asign3.xlsx',time,'C6:C6');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D6:D6');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E6:E6');

% Conjugate Gradient Method
[X3, time, iteration, t1, t2] = MA14M004_CDS(3);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B8:B8');
xlswrite('outputMA14M004_asign3.xlsx',time,'C8:C8');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D8:D8');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E8:E8');

% Bi-Conjugate Gradient Stabilized Method
[X4, time, iteration, t1, t2] = MA14M004_CDS(4);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B10:B10');
xlswrite('outputMA14M004_asign3.xlsx',time,'C10:C10');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D10:D10');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E10:E10');

% Code for exact Solution:
D = MA14M004_exact(time);

% plotting the results:
figure(1);
subplot(2,2,1);
contourf(X1);
colorbar;
title('Gauss Seidel Method without relaxation');
subplot(2,2,2);
contourf(X2);
colorbar;
title('Gauss Seidel Method with relaxation');
subplot(2,2,3);
contourf(X3);
colorbar;
title('Conjugate Gradient Method');
subplot(2,2,4);
contourf(X4);
colorbar;
title('Biconjugate Gradient Stabilized Method');
suptitle(['Convection term with Central Difference scheme at time = 
',num2str(time), 's']);

% Comparison of Results:
X = 1:100;
figure(2);
subplot(2,2,1);
plot(X,D(:,50),X,X1(:,50));
legend('Exact Solution','Gauss Seidel without relaxation');
title('Gauss Seidel without relaxation');
```

```
subplot(2,2,2);
plot(X,D(:,50),X,X2(:,50));
legend('Exact Solution','Gauss Seidel with relaxation');
title('Gauss Seidel with relaxation');
subplot(2,2,3);
plot(X,D(:,50),X,X3(:,50));
legend('Exact Solution','Conjugate Gradient Method');
title('Conjugate Gradient Method');
subplot(2,2,4);
plot(X,D(:,50),X,X4(:,50));
legend('Exact Solution','Bi-Conjugate Stabilized Method');
title('Bi-Conjugate Stabilized Method');
suptitle('Comparison of the results of Central Difference Scheme with exact
solution along the central line');

% %% Code for Convection term with First Order Upwind.
clear all;
clc;

% Gauss Seidel Iteration without relaxation
[X1, time, iteration, t1, t2] = MA14M004_upwind(1);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B16:B16');
xlswrite('outputMA14M004_asign3.xlsx',time,'C16:C16');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D16:D16');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E16:E16');

% Gauss Seidel Iteration with relaxation
[X2, time, iteration, t1, t2] = MA14M004_upwind(2);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B18:B18');
xlswrite('outputMA14M004_asign3.xlsx',time,'C18:C18');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D18:D18');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E18:E18');

% Conjugate Gradient Method
[X3, time, iteration, t1, t2] = MA14M004_upwind(3);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B20:B20');
xlswrite('outputMA14M004_asign3.xlsx',time,'C20:C20');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D20:D20');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E20:E20');

% Bi-Conjugate Gradient Stabilized Method
[X4, time, iteration, t1, t2] = MA14M004_upwind(4);
xlswrite('outputMA14M004_asign3.xlsx',iteration,'B22:B22');
xlswrite('outputMA14M004_asign3.xlsx',time,'C22:C22');
xlswrite('outputMA14M004_asign3.xlsx',t1,'D22:D22');
xlswrite('outputMA14M004_asign3.xlsx',t2,'E22:E22');

% Code for exact Solution:
D = MA14M004_exact(time);

% plotting the results
figure(3);
subplot(2,2,1);
contourf(X1);
colorbar;
```

```
title('Gauss Seidel Method without relaxation');
subplot(2,2,2);
contourf(X2);
colorbar;
title('Gauss Seidel Method with relaxation');
subplot(2,2,3);
contourf(X3);
colorbar;
title('Conjugate Gradient Method');
subplot(2,2,4);
contourf(X4);
colorbar;
title('Biconjugate Gradient Stabilized Method');
suptitle(['Convection term with First Order Upwind at time =
',num2str(time),'s']);

%comparison of results
X = 1:100;
figure(4);
subplot(2,2,1);
plot(X,D(:,50),X,X1(:,50));
legend('Exact Solution','Gauss Seidel without relaxation');
title('Gauss Seidel without relaxation');
subplot(2,2,2);
plot(X,D(:,50),X,X2(:,50));
legend('Exact Solution','Gauss Seidel with relaxation');
title('Gauss Seidel with relaxation');
subplot(2,2,3);
plot(X,D(:,50),X,X3(:,50));
legend('Exact Solution','Conjugate Gradient Method');
title('Conjugate Gradient Method');
subplot(2,2,4);
plot(X,D(:,50),X,X4(:,50));
legend('Exact Solution','Bi-Conjugate Stabilized Method');
title('Bi-Conjugate Stabilized Method');
suptitle('Comparison of the results of First Order Upwind with exact solution
along the central line');
```

File 2: MA14M004_CDS.m

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% This file computes the results when convection term is discretized by
% Central Difference Schme.

%% Function
function [D, t, it, t1, t2] = MA14M004_CDS(option)

tic;
t2 = cputime;

thetha = 0.5; % Crank Nicholson Scheme
N = 100;
gama = 1;
```

```
len = 1;
delta_x = len/N;
delta_y = len/N;
delta_t = 0.01;
ap_not = delta_x*delta_y/delta_t;
B = zeros(N*N,5);
C = zeros(N*N,1);
X = zeros(N*N,1);
coeff = zeros(N*N,5);
n = 1;
De = 1*delta_y/delta_x;
Dw = 1*delta_y/delta_x;
Dn = 1*delta_x/delta_y;
Ds = 1*delta_x/delta_y;
u = @(x,y) gama*cos(pi*(x - 0.5))*sin(pi*(y - 0.5));
v = @(x,y) -gama*sin(pi*(x - 0.5))*cos(pi*(y - 0.5));
phi = @(x,y) cos(pi*(x - 0.5))*cos(pi*(y - 0.5));
it = 0;

%% Computation of coeffcient matrix
for j = 1:N
    for i = 1:N
        east = (i)*delta_x;
        East = delta_x/2 + (i)*delta_x;
        west = (i - 1)*delta_x;
        West = delta_x/2  + (i - 2)*delta_x;
        north = j*delta_y;
        North = delta_y/2 + (j)*delta_y;
        south = (j - 1)*delta_y;
        South = delta_y/2 + (j - 2)*delta_y;
        Point_x = delta_x/2 + (i - 1)*delta_x;
        Point_y = delta_y/2 + (j - 1)*delta_y;
        %% left face
        Fe = (u(east,Point_y))*delta_y;
        Fw = (u(west,Point_y))*delta_y;
        Fn = (v(Point_x,north))*delta_x;
        Fs = (v(Point_x,south))*delta_x;

        if (i == 1)
            if (j == 1) %left bottom corner
                aw = 0;
                an = Dn - Fn/2;
                as = 0;
                ae = De - Fe/2;
                Sp = -(2*Dw + 2*Ds);
                Fw = 0;
                Fs = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
                B(n,4) = -thetha *ae;
                B(n,3) = ap;
                B(n,5) = -thetha * an;
            elseif (j == N) % top left corner
                ae = De - Fe/2;
                aw = 0;
                an = 0;
```

```
                as = Ds + Fs/2;
                Sp = -(2*Dw + 2*Dn);
                Fn = 0;
                Fw = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs)- Sp);
                B(n,3) = ap;
                B(n,1) = -thetha * as;
                B(n,4) = -thetha*ae;
            else % left face except at the corners
                ae = De - Fe/2;
                aw = 0;
                an = Dn - Fn/2;
                as = Ds + Fs/2;
                Sp = -(2*Dw);
                Fw = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
                B(n,3) = ap;
                B(n,1) = -thetha*as;
                B(n,5) = -thetha * an;
                B(n,4) = -thetha*ae;
            end

        elseif(i == N)
            % right face
            if(j == 1) % bottom right corner
                ae = 0;
                aw = Dw + Fw/2;
                an = Dn - Fn/2;
                as = 0;
                Sp = -(2*De + 2*Ds);
                Fe = 0;
                Fs = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
                B(n,3) = ap;
                B(n,2) = -thetha*aw;
                B(n,5) = -thetha * an;
            elseif(j == N) % top right corner
                ae = 0;
                aw = Dw + Fw/2;
                an = 0;
                as = Ds + Fs/2;
                Sp = -(2*De + 2*Dn);
                Fe = 0;
                Fn = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
                B(n,3) = ap;
                B(n,2) = -thetha*aw;
                B(n,1) = -thetha*as;
            else % right face except at the corners
                ae = 0;
                aw = Dw + Fw/2;
                an = Dn - Fn/2;
                as = Ds + Fs/2;
```

```
                Sp = -(2*De);
                Fe = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
                B(n,3) = ap;
                B(n,2) = -thetha*aw;
                B(n,1) = -thetha*as;
                B(n,5) = -thetha*an;
            end

        elseif(j == 1) % bottom face except at the corners
            ae = De - Fe/2;
            aw = Dw + Fw/2;
            an = Dn - Fn/2;
            as = 0;
            Sp = -(2*Ds);
            Fs = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn - Fs) -
Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,4) = -thetha*ae;
            B(n,5) = -thetha*an;
        elseif(j == N) % top face except at the corners
            ae = De - Fe/2;
            aw = Dw + Fw/2;
            an = 0;
            as = Ds + Fs/2;
            Sp = -(2*Dn);
            Fn = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn - Fs) -
Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,4) = -thetha*ae;
            B(n,1) = -thetha*as;
        else % interior points
            ae = De - Fe/2;
            aw = Dw + Fw/2;
            an = Dn - Fn/2;
            as = Ds + Fs/2;
            Sp = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn - Fs) -
Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,4) = -thetha*ae;
            B(n,5) = -thetha * an;
            B(n,1) = -thetha*as;
        end
        X(((j - 1)*N+i)) = phi(Point_x,Point_y);
        n = n + 1;
        coeff(n,1) = as;
        coeff(n,2) = aw;
        coeff(n,3) = ap;
        coeff(n,4) = ae;
        coeff(n,5) = an;
```

```
    end
end

Y2 = X;

%% Iteration begins

diff = 20;
t = 0;
while (diff > 10^(-6))
    C1 = zeros(N*N,1);
    n = 1;
    %% Computation of the right hand side.
    for j = 1:N
        for i = 1:N

            %% left face
            east = (i)*delta_x;
            East = delta_x/2 + (i)*delta_x;
            west = (i - 1)*delta_x;
            West = delta_x/2  + (i - 2)*delta_x;
            north = j*delta_y;
            North = delta_y/2 + (j)*delta_y;
            south = (j - 1)*delta_y;
            South = delta_y/2 + (j - 2)*delta_y;
            Point_x = delta_x/2 + (i - 1)*delta_x;
            Point_y = delta_y/2 + (j - 1)*delta_y;
            %% left face
            Fe = (u(east,Point_y))*delta_y;
            Fw = (u(west,Point_y))*delta_y;
            Fn = (v(Point_x,north))*delta_x;
            Fs = (v(Point_x,south))*delta_x;
            left = (j - 1)*N + i - 1;
            right = (j - 1)*N + i + 1;
            top = j*N + i;
            point = (j - 1)*N + i;
            bottom = (j - 2)*N + i;
            if (i == 1)
                if (j == 1) %left bottom corner
                    aw = 0;
                    an = Dn - Fn/2;
                    as = 0;
                    ae = De - Fe/2;
                    Sp = -(2*Dw + 2*Ds);
                    Fw = 0;
                    Fs = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*ae*X(right) +(1 -
thetha)*an*X(top) ...
                        + X(point)*ap_prime;
                elseif (j == N) % top left corner
                    ae = De - Fe/2;
                    aw = 0;
                    an = 0;
                    as = Ds + Fs/2;
```

```
                    Sp = -(2*Dw + 2*Dn);
                    Fn = 0;
                    Fw = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*ae*X(right) + (1 -
thetha)*as*X(bottom)+ X(point)*ap_prime;
                else % left face except at the corners
                    ae = De - Fe/2;
                    aw = 0;
                    an = Dn - Fn/2;
                    as = Ds + Fs/2;
                    Sp = -(2*Dw);
                    Fw = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*ae*X(right) +(1 -
thetha)*an*X(top) + (1 - thetha)*as*X(bottom)+ X(point)*ap_prime;
                end

            elseif(i == N)
                %% right face
                if(j == 1) % bottom right corner
                    ae = 0;
                    aw = Dw + Fw/2;
                    an = Dn - Fn/2;
                    as = 0;
                    Sp = -(2*De + 2*Ds);
                    Fe = 0;
                    Fs = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);

                    C1(n,1) = (1 - thetha)*aw*X(left) +(1 -
thetha)*an*X(top)+ X(point)*ap_prime;
                elseif(j == N) % top right corner
                    ae = 0;
                    aw = Dw + Fw/2;
                    an = 0;
                    as = Ds + Fs/2;
                    Sp = -(2*De + 2*Dn);
                    Fe = 0;
                    Fn = 0;


                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*aw*X(left)+ (1 -
thetha)*as*X(bottom)+ X(point)*ap_prime;
                else % right face except at the corners
                    ae = 0;
                    aw = Dw + Fw/2;
                    an = Dn - Fn/2;
                    as = Ds + Fs/2;
                    Sp = -(2*De);
                    Fe = 0;
```

```
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1)  = (1 - thetha)*aw*X(left)+(1 - thetha)*an*X(top)
+ (1 - thetha)*as*X(bottom)+ X(point)*ap_prime;
                end

            elseif(j == 1) % bottom face except at the corners
                ae = De - Fe/2;
                aw = Dw + Fw/2;
                an = Dn - Fn/2;
                as = 0;
                Sp = -(2*Ds);
                Fs = 0;
                ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe -
Fw) + (Fn - Fs) - Sp);
                C1(n,1)  = (1 - thetha)*ae*X(right) +(1 -
thetha)*aw*X(left)+(1 - thetha)*an*X(top) + X(point)*ap_prime;
            elseif(j == N) % top face except at the corners
                ae = De - Fe/2;
                aw = Dw + Fw/2;
                an = 0;
                as = Ds + Fs/2;
                Sp = -(2*Dn);
                Fn = 0;
                ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe -
Fw) + (Fn - Fs) - Sp);
                C1(n,1)  = (1 - thetha)*ae*X(right) +(1 - thetha)*aw*X(left)+
(1 - thetha)*as*X(bottom) + X(point)*ap_prime;
            else % interior points
                ae = De - Fe/2;
                aw = Dw + Fw/2;
                an = Dn - Fn/2;
                as = Ds + Fs/2;
                Sp = 0;
                ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe -
Fw) + (Fn - Fs) - Sp);
                C1(n,1)  = (1 - thetha)*ae*X(right) +(1 -
thetha)*aw*X(left)+(1 - thetha)*an*X(top) + (1 - thetha)*as*X(bottom)+
X(point)*ap_prime;
            end
            n = n + 1;
        end
    end
    %% Gauss Seidel Iteration without relaxation
    if (option == 1)
        [Y2,it] = MA14M004_Gauss(B,C1,Y2,it,1);
    end
    %% Gauss Seidel Iteration with relaxation
    if (option == 2)
        [Y2,it] = MA14M004_Gauss(B,C1,Y2,it,1.8);
    end
    %% Conjugate Gradient Method
    if (option == 3)
        [Y2,it] = MA14M004_CG(B,C1,Y2,it);
    end
    %% Bi-Conjugate Gradient Stabilized Method:
    if (option == 4)
```

```
        [Y2,it] = MA14M004_BCG(B,C1,Y2,it);
    end

    diff = (abs(X(49*100 + 50) - Y2(49*100 + 50)));
    t = t + delta_t;
    X = Y2;
end

%% Final Reults
l = 1;
D = zeros(N,N);
for j = 1:N
    for i = 1:N
        D(i,j) = Y2(l);

        l = l  + 1;
    end
end

D = flipdim(D,1);
t1 = toc;
t2 = cputime - t2;
end
```

File 3: MA14M004_upwind.m

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% This file computes the results when convection term is discretized by
% First Order Upwind.

%% Function:
function [D, t, it, t1, t2] = MA14M004_upwind(option)
tic;
t2 = cputime;

thetha = 0.5; % Crank Nicholson Scheme
N = 100;
gama = 1;
len = 1;
delta_x = len/N;
delta_y = len/N;
delta_t = 0.01;
ap_not = delta_x*delta_y/delta_t;
B = zeros(N*N,5);
X = zeros(N*N,1);
coeff = zeros(N*N,5);
n = 1;
De = 1*delta_y/delta_x;
Dw = 1*delta_y/delta_x;
Dn = 1*delta_x/delta_y;
Ds = 1*delta_x/delta_y;
u = @(x,y) gama*cos(pi*(x - 0.5))*sin(pi*(y - 0.5));
v = @(x,y) -gama*sin(pi*(x - 0.5))*cos(pi*(y - 0.5));
phi = @(x,y) cos(pi*(x - 0.5))*cos(pi*(y - 0.5));
```

```
%% Computation of Coefficient Matrix
for j = 1:N
    for i = 1:N
        east = (i)*delta_x;
        East = delta_x/2 + (i)*delta_x;
        west = (i - 1)*delta_x;
        West = delta_x/2  + (i - 2)*delta_x;
        north = j*delta_y;
        North = delta_y/2 + (j)*delta_y;
        south = (j - 1)*delta_y;
        South = delta_y/2 + (j - 2)*delta_y;
        Point_x = delta_x/2 + (i - 1)*delta_x;
        Point_y = delta_y/2 + (j - 1)*delta_y;
        %% left face
        Fe = (u(east,Point_y))*delta_y;
        Fw = (u(west,Point_y))*delta_y;
        Fn = (v(Point_x,north))*delta_x;
        Fs = (v(Point_x,south))*delta_x;

        if (i == 1)
            if (j == 1) %left bottom corner
                aw = 0;
                an = Dn + max(-Fn,0);
                as = 0;
                ae = De + max(-Fe,0);
                Sp = -(2*Dw + 2*Ds);
                Fw = 0;
                Fs = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
                B(n,4) = -thetha *ae;
                B(n,3) = ap;
                B(n,5) = -thetha * an;
            elseif (j == N) % top left corner
                ae = De + max(-Fe,0);
                aw = 0;
                an = 0;
                as = Ds + max(Fs,0);
                Sp = -(2*Dw + 2*Dn);
                Fn = 0;
                Fw = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs)- Sp);
                B(n,3) = ap;
                B(n,1) = -thetha * as;
                B(n,4) = -thetha*ae;
            else % left face except at the corners
                ae = De + max(-Fe,0);
                aw = 0;
                an = Dn + max(-Fn,0);
                as = Ds + max(Fs,0);
                Sp = -(2*Dw);
                Fw = 0;
                ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
```

```
            B(n,3) = ap;
            B(n,1) = -thetha*as;
            B(n,5) = -thetha * an;
            B(n,4) = -thetha*ae;
        end

    elseif(i == N)
        % right face
        if(j == 1) % bottom right corner
            ae = 0;
            aw = Dw + max(Fw,0);
            an = Dn + max(-Fn,0);
            as = 0;
            Sp = -(2*De + 2*Ds);
            Fe = 0;
            Fs = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,5) = -thetha * an;
        elseif(j == N) % top right corner
            ae = 0;
            aw = Dw + max(Fw,0);
            an = 0;
            as = Ds + max(Fs,0);
            Sp = -(2*De + 2*Dn);
            Fe = 0;
            Fn = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,1) = -thetha*as;
        else % right face except at the corners
            ae = 0;
            aw = Dw + max(Fw,0);
            an = Dn + max(-Fn,0);
            as = Ds + max(Fs,0);
            Sp = -(2*De);
            Fe = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn -
Fs) - Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,1) = -thetha*as;
            B(n,5) = -thetha*an;
        end

    elseif(j == 1) % bottom face except at the corners
        ae = De + max(-Fe,0);
        aw = Dw + max(Fw,0);
        an = Dn + max(-Fn,0);
        as = 0;
        Sp = -(2*Ds);
        Fs = 0;
```

```
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn - Fs) -
Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,4) = -thetha*ae;
            B(n,5) = -thetha*an;
        elseif(j == N) % top face except at the corners
            ae = De + max(-Fe,0);
            aw = Dw + max(Fw,0);
            an = 0;
            as = Ds + max(Fs,0);
            Sp = -(2*Dn);
            Fn = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn - Fs) -
Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,4) = -thetha*ae;
            B(n,1) = -thetha*as;
        else % interior points
            ae = De + max(-Fe,0);
            aw = Dw + max(Fw,0);
            an = Dn + max(-Fn,0);
            as = Ds + max(Fs,0);
            Sp = 0;
            ap = ap_not + thetha*(aw + an + as + ae + (Fe - Fw) + (Fn - Fs) -
Sp);
            B(n,3) = ap;
            B(n,2) = -thetha*aw;
            B(n,4) = -thetha*ae;
            B(n,5) = -thetha * an;
            B(n,1) = -thetha*as;
        end
        X(((j - 1)*N+i)) = phi(Point_x,Point_y);
        coeff(n,1) = as;
        coeff(n,2) = aw;
        coeff(n,3) = ap;
        coeff(n,4) = ae;
        coeff(n,5) = an;
        n = n + 1;
    end

end


%% Iteration begins:

Y2 = X;
diff = 20;
t = 0;
it = 0;

while (diff > 10^(-6))
    C1 = zeros(N*N,1);
    n = 1;
```

```
    for j = 1:N
        for i = 1:N
            %% Compuatation of the right hand Side.
            east = (i)*delta_x;
            East = delta_x/2 + (i)*delta_x;
            west = (i - 1)*delta_x;
            West = delta_x/2  + (i - 2)*delta_x;
            north = j*delta_y;
            North = delta_y/2 + (j)*delta_y;
            south = (j - 1)*delta_y;
            South = delta_y/2 + (j - 2)*delta_y;
            Point_x = delta_x/2 + (i - 1)*delta_x;
            Point_y = delta_y/2 + (j - 1)*delta_y;
            %% left face
            Fe = (u(east,Point_y))*delta_y;
            Fw = (u(west,Point_y))*delta_y;
            Fn = (v(Point_x,north))*delta_x;
            Fs = (v(Point_x,south))*delta_x;
            left = (j - 1)*N + i - 1;
            right = (j - 1)*N + i + 1;
            top = j*N + i;
            point = (j - 1)*N + i;
            bottom = (j - 2)*N + i;
            if (i == 1)
                if (j == 1) %left bottom corner
                    aw = 0;
                    an = Dn + max(-Fn,0);
                    as = 0;
                    ae = De + max(-Fe,0);
                    Sp = -(2*Dw + 2*Ds);
                    Fw = 0;
                    Fs = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*ae*X(right) +(1 -
thetha)*an*X(top) + X(point)*ap_prime;
                elseif (j == N) % top left corner
                    ae = De + max(-Fe,0);
                    aw = 0;
                    an = 0;
                    as = Ds + max(Fs,0);
                    Sp = -(2*Dw + 2*Dn);
                    Fn = 0;
                    Fw = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*ae*X(right) + (1 -
thetha)*as*X(bottom)+ X(point)*ap_prime;
                else % left face except at the corners
                    ae = De + max(-Fe,0);
                    aw = 0;
                    an = Dn + max(-Fn,0);
                    as = Ds + max(Fs,0);
                    Sp = -(2*Dw);
                    Fw = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
```

```
                    C1(n,1) = (1 - thetha)*ae*X(right) +(1 -
thetha)*an*X(top) + (1 - thetha)*as*X(bottom)+ X(point)*ap_prime;
                end

            elseif(i == N)
                %% right face
                if(j == 1) % bottom right corner
                    ae = 0;
                    aw = Dw + max(Fw,0);
                    an = Dn + max(-Fn,0);
                    as = 0;
                    Sp = -(2*De + 2*Ds);
                    Fe = 0;
                    Fs = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*aw*X(left) +(1 -
thetha)*an*X(top)+ X(point)*ap_prime;
                elseif(j == N) % top right corner
                    ae = 0;
                    aw = Dw + max(Fw,0);
                    an = 0;
                    as = Ds + max(Fs,0);
                    Sp = -(2*De + 2*Dn);
                    Fe = 0;
                    Fn = 0;

                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*aw*X(left)+ (1 -
thetha)*as*X(bottom)+ X(point)*ap_prime;
                else % right face except at the corners
                    ae = 0;
                    aw = Dw + max(Fw,0);
                    an = Dn + max(-Fn,0);
                    as = Ds + max(Fs,0);
                    Sp = -(2*De);
                    Fe = 0;
                    ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe
- Fw) + (Fn - Fs) - Sp);
                    C1(n,1) = (1 - thetha)*aw*X(left)+(1 - thetha)*an*X(top)
+ (1 - thetha)*as*X(bottom) + X(point)*ap_prime;
                end

            elseif(j == 1) % bottom face except at the corners
                ae = De +max(-Fe,0);
                aw = Dw + max(Fw,0);
                an = Dn + max(-Fn,0);
                as = 0;
                Sp = -(2*Ds);
                Fs = 0;
                ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe -
Fw) + (Fn - Fs) - Sp);
                C1(n,1) = (1 - thetha)*ae*X(right) +(1 -
thetha)*aw*X(left)+(1 - thetha)*an*X(top) + X(point)*ap_prime;
            elseif(j == N) % top face except at the corners
```

```
                ae = De + max(-Fe,0);
                aw = Dw + max(Fw,0);
                an = 0;
                as = Ds + max(Fs,0);
                Sp = -(2*Dn);
                Fn = 0;
                ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe -
Fw) + (Fn - Fs) - Sp);
                C1(n,1) = (1 - thetha)*ae*X(right) +(1 - thetha)*aw*X(left)+
(1 - thetha)*as*X(bottom)+ X(point)*ap_prime;
            else % interior points
                ae = De + max(-Fe,0);
                aw = Dw + max(Fw,0);
                an = Dn + max(-Fn,0);
                as = Ds + max(Fs,0);
                Sp = 0;
                ap_prime = ap_not - (1 - thetha)*(aw + an + as + ae + (Fe -
Fw) + (Fn - Fs) - Sp);
                C1(n,1) = (1 - thetha)*ae*X(right) +(1 -
thetha)*aw*X(left)+(1 - thetha)*an*X(top) + (1 - thetha)*as*X(bottom) +
X(point)*ap_prime;
            end
            coeff(n,1) = as;
            coeff(n,2) = aw;
            coeff(n,3) = ap;
            coeff(n,4) = ae;
            coeff(n,5) = an;


            n = n + 1;
        end
    end
    %% Gauss Seidel Method without relaxation
    if (option == 1)
        [Y2,it] = MA14M004_Gauss(B,C1,Y2,it,1);
    end
    %% Gauss Seidel Method with relaxation
    if (option == 2)
        [Y2,it] = MA14M004_Gauss(B,C1,Y2,it,1.8);
    end
    %% Conjugate Gradient Method
    if (option == 3)
        [Y2,it] = MA14M004_CG(B,C1,Y2,it);
    end
    %% Biconjugate Gradient Stabilized Method
    if (option == 4)
        [Y2,it] = MA14M004_BCG(B,C1,Y2,it);
    end

    diff = (abs(X(49*100 + 50) - Y2(49*100 + 50)));

    t = t + delta_t;
    X = Y2;
end

l = 1;
```

```
%% Final result
D = zeros(N,N);
for j = 1:N
    for i = 1:N
        D(i,j) = Y2(l);

        l = l  + 1;
    end
end

D = flipdim(D,1);
t1 = toc;
t2 = cputime - t2;
end
```

File 4: MA14M004_Gauss.m

```
%% Individual Details
% Author: Kumar Saurabh
% Roll No. MA14M004
% Gauss siedel Method to solve system of linear equation

%% Function :
function [  X, n ] = MA14M004_Gauss( A, B, X, n, relax)

len = size(B,1);
error = 1999;
X1 = zeros(len,1);
%X = zeros(len,1);
N = sqrt(len);


%% Fast Gauss Siedel specific to this case %%
while(error > 10^(-6))
    for k = 1:len
        row = floor((k-1)/N) + 1;
        column = mod((k - 1),N) + 1;
        index = (row - 1)*N + column;
        left = index - 1;
        right = index + 1;
        top = index + N;
        bottom = index - N;

        if (row == 1)
            if (column == 1)
                sum = A(k,5)*X(top) + A(k,4)*X(right);
            elseif (column == N)
                sum = A(k,5)*X(top) + A(k,2)*X(left);
            else
                sum = A(k,5)*X(top) + A(k,2)*X(left) + A(k,4)*X(right);
            end
        elseif (row == N)
            if (column == 1)
                sum = A(k,1)*X(bottom) + A(k,4)*X(right);
            elseif (column == N)
```

```
                sum = A(k,1)*X(bottom) + A(k,2)*X(left);
            else
                sum = A(k,1)*X(bottom) + A(k,2)* X(left) + A(k,4)*X(right);
            end
        else
            if(column == 1)
                sum = A(k,1)*X(bottom) + A(k,4)*X(right) + A(k,5)*X(top);
            elseif (column == N)
                sum = A(k,1)*X(bottom) + A(k,2)*X(left) + A(k,5)*X(top);
            else
                sum = A(k,1)*X(bottom) + A(k,2)*X(left) + A(k,5)*X(top) +
A(k,4)*X(right);
            end
        end
        X(k) =  X1(k) + relax*((B(k) - sum)/A(k,3) - X1(k));
    end

    % Calculation of error
    error = 0;
    for i = 1:len
        error = error + abs((X(i) - X1(i))/X(i));
    end


    %disp(error);
    clear X1;
    X1 = X;
    n = n + 1;

end

%disp(n);
end
```

File 5: MA14M004_CG.m

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% Conjugate Gradient Method to solve AX = B

%% Function:
function [ X,n ] = MA14M004_CG( A, B, X, n )

len = size(B,1);
%X = ones(len,1);
X1 = ones(len,1);
Q = MA14M004_mul(A,X);
r = (B - Q);
d = r;
rnew = r'*r;

error = 199;

while (error > 10^(-6))
    q = MA14M004_mul(A,d);
```

```
    alpha = rnew/(d'*q);
    X = X + alpha*d;
    r = r - alpha*q;
    rold = rnew;
    rnew = r'*r;
    beeta = rnew/rold;
    d = r + beeta*d;
    error = 0;
    for i = 1:len
        error = error + abs((X1(i) - X(i))/X1(i));
    end
    X1 = X;
    n = n + 1;
end
end
```

File 6: MA14M004_BCG.m

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% Bi-Conjugate Gradient Stabilized Method to solve AX = B

%% Function:
function [ X,n ] = MA14M004_BCG( A, B, X,n )
len = size(B,1);
%X = ones(len,1);
X1 = ones(len,1);
Q = MA14M004_mul(A,X);
r = (B - Q);
r_star = r;
alpha = 1;
omega = 1;
rho = 1;
v = 0;
p1 = 0;
error = 199;

while (error > 10^(-6))
    rho_1 = r_star'*r;
    beeta = (rho_1/rho)*(alpha/omega);
    p1 = r + beeta*(p1 - omega*v);
    v = MA14M004_mul(A,p1);
    alpha = rho_1/(r_star'*v);
    s = r - alpha*v;
    t = MA14M004_mul(A,s);
    omega = (t'*s)/(t'*t);
    X = X + omega*s + alpha*p1;
    r = s - omega*t;
    rho = rho_1;
    n = n + 1;

    error = 0;
    for i = 1:len
        error = error + abs((X1(i) - X(i))/X1(i));
```

```
    end
    X1 = X;
    %disp(error);
end

end
```

File 7: MA14M004_mul.m

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% Matrix Multiplication:

%% Function:
function [ C ] = MA14M004_mul( A,B )
len = size(B,1);
C = zeros(len,1);
N = sqrt(len);
for k = 1:len
    row = floor((k-1)/N) + 1;
    column = mod((k - 1),N) + 1;
    index = (row - 1)*N + column;
    left = index - 1;
    right = index + 1;
    top = index + N;
    bottom = index - N;

    if (row == 1)
        if (column == 1)
            C(k,1) = A(k,5)*B(top) + A(k,4)*B(right) + A(k,3)*B(index);
        elseif (column == N)
            C(k,1) = A(k,5)*B(top) + A(k,2)*B(left) + A(k,3)*B(index);
        else
            C(k,1) = A(k,5)*B(top) + A(k,2)*B(left) + A(k,4)*B(right) +
A(k,3)*B(index);
        end
    elseif (row == N)
        if (column == 1)
            C(k,1) = A(k,1)*B(bottom) + A(k,4)*B(right) + A(k,3)*B(index);
        elseif (column == N)
            C(k,1) = A(k,1)*B(bottom) + A(k,2)*B(left)+ A(k,3)*B(index);
        else
            C(k,1) = A(k,1)*B(bottom) + A(k,2)* B(left) + A(k,4)*B(right) +
A(k,3)*B(index);
        end
    else
        if(column == 1)
            C(k,1) = A(k,1)*B(bottom) + A(k,4)*B(right) + A(k,5)*B(top) +
A(k,3)*B(index);
        elseif (column == N)
            C(k,1) = A(k,1)*B(bottom) + A(k,2)*B(left) + A(k,5)*B(top) +
A(k,3)*B(index);
        else
            C(k,1) = A(k,1)*B(bottom) + A(k,2)*B(left) + A(k,5)*B(top) +
A(k,4)*B(right) + A(k,3)*B(index);
```

```
        end

    end

end

end
```

File 8: MA14M004_exact.m:

```
%% Author: Kumar Saurabh
% Roll No. MA14M004
% Computes the exact solution of the given problem.

%% Function
function [D1] = MA14M004_exact(t1)

    N = 100;
    D1 = zeros(N,N);
    delta_x = 1/N;
    delta_y = 1/N;

    for i = 1:N
        for j = 1:N
            D1(i,j) = exp(-2*pi*pi*t1)*cos(pi*((delta_x/2) + (i - 1)*delta_x
- 0.5))*cos(pi*((delta_y/2) + (j - 1)*delta_y - 0.5));

        end
    end
    D1 = flipdim(D1,1);
end
```