הטכניון – מכון טכנולוגי לישראל

NUMERICAL METHODS IN AEROSPACE ENGINEERING

HOMEWORK ASSIGNMENT x

סמסטר אביב תשפ"ה

SPRING SEMESTER 2025

GRADE	OUT OF	CHAPTER
	2	ABSTRACT
	2	CONTENTS, STYLE &C.
	4	PHYSICAL PROBLEM
	4	MATHEMATICAL MODEL
	26	NUMERICAL METHODS
	20	INFLUENCE OF NUMERICAL METHODS
	20	RESULTS
	2	SUMMARY & CONCLUSIONS
	20	COMPUTER PROGRAM
	100	**TOTAL**

Almog Dobrescu ID 214254252

June 28, 2025

Abstract

hii

Contents

List of Figures

Listings

Nomenclature

Δx	step size in the x coordinate
Δy	step size in the y coordinate
μ	viscosity of the fluid
ε	convergence criteria
c	pressure gradient on the flow in a section
i	index in the x coordinate
j	index in the y coordinate
n	iterative index
ni	number of indexes in the x direction
nj	number of indexes in the y direction

1 The Physical Problem

An incompressible viscose Newtonian fluid flows in a channel.

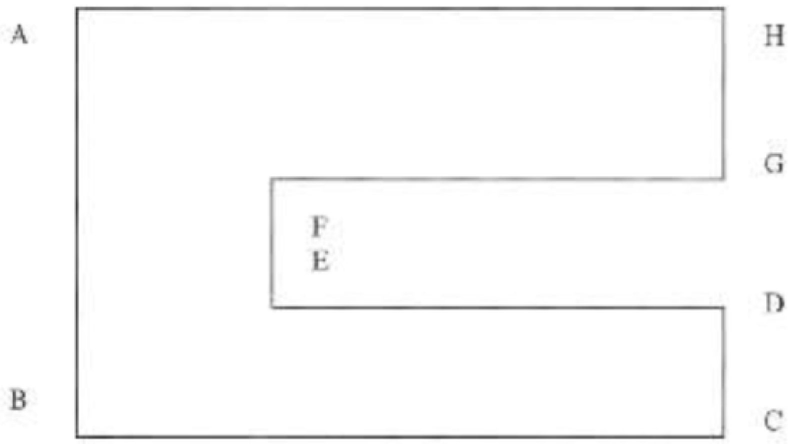


Figure 1: The Channel

Where:

- $AB = 12\,[in]$
- $BC = 12\,[in]$
- $CD = 2\,[in]$
- $DE = 6\,[in]$
- $EF = 6\,[in]$
- $FG = 6\,[in]$
- $GH = 4\,[in]$
- $HA = 12\,[in]$

The x-axis will be from left to right and the y-axis will be from bottom to top such that the origin is at point B.

2 The Mathematical Model

The steady-state velocity of the fluid is given by:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = -\frac{c}{\mu} \tag{1}$$

In our case:

- $c = 0.0002\left[\frac{lb}{in^3}\right]$
- $\mu = 0.25 \cdot 10^{-5}\left[\frac{lb \cdot sec}{in^2}\right]$

2.1 Boundary Conditions

Since the flow is viscose, the boundary conditions are no penetration and no slip. The flow is at a steady state so the only direction of the flow is normal the sectional area (outside the paper). Hence the velocity at the boundaries is:

$$\phi|_{AB} = \phi|_{BC} = \phi|_{CD} = \phi|_{DE} = \phi|_{EF} = \phi|_{FG} = \phi|_{GH} = \phi|_{HA} = 0 \tag{2}$$

3 The Numerical Methods

3.1 Finite Differencing

In order to solve the partial differential equation we will use finite differences. For the second derivative in space, we will use central differencing:

$$\frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{\Delta x^2} + \mathcal{O}\left(\Delta x^2\right) + \frac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{\Delta y^2} + \mathcal{O}\left(\Delta y^2\right) = -\frac{c}{\mu}$$

$$\left(\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}\right)\Delta y^2 + \left(\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}\right)\Delta x^2 = -\Delta x^2 \Delta y^2 \frac{c}{\mu} + \mathcal{O}\left(\Delta x^2, \Delta y^2\right)$$

$$-2\phi_{i,j}\left(\Delta y^2 + \Delta x^2\right) + \left(\phi_{i-1,j} + \phi_{i+1,j}\right)\Delta y^2 + \left(\phi_{i,j-1} + \phi_{i,j+1}\right)\Delta x^2 = -\Delta x^2 \Delta y^2 \frac{c}{\mu}$$

$$2\phi_{i,j}\left(\Delta y^2 + \Delta x^2\right) = \left(\phi_{i-1,j} + \phi_{i+1,j}\right)\Delta y^2 + \left(\phi_{i,j-1} + \phi_{i,j+1}\right)\Delta x^2 + \Delta x^2 \Delta y^2 \frac{c}{\mu}$$

$$2\phi_{i,j} = \left(\phi_{i-1,j} + \phi_{i+1,j}\right)\frac{\Delta y^2}{\Delta y^2 + \Delta x^2} + \left(\phi_{i,j-1} + \phi_{i,j+1}\right)\frac{\Delta x^2}{\Delta y^2 + \Delta x^2} + \frac{\Delta x^2 \Delta y^2}{\Delta y^2 + \Delta x^2}\frac{c}{\mu}$$

$$\phi_{i,j} = \frac{1}{2}\left(\left(\phi_{i-1,j} + \phi_{i+1,j}\right)\frac{\Delta y^2}{\Delta y^2 + \Delta x^2} + \left(\phi_{i,j-1} + \phi_{i,j+1}\right)\frac{\Delta x^2}{\Delta y^2 + \Delta x^2} + \frac{\Delta x^2 \Delta y^2}{\Delta y^2 + \Delta x^2}\frac{c}{\mu}\right) \tag{3}$$

Where:

- $\Delta x = \dfrac{x_{max} - x_{min}}{ni}$
- $\Delta y = \dfrac{y_{max} - y_{min}}{nj}$
- ni and nj will be chosen such that the walls will be at the middle of an element.

To solve the system of equations we will use the Gauss-Seidel method:

$$\phi_{i,j}^{n+1} = \frac{1}{2}\left(\left(\phi_{i-1,j}^{n+1} + \phi_{i+1,j}^{n}\right)\frac{\Delta y^2}{\Delta y^2 + \Delta x^2} + \left(\phi_{i,j-1}^{n+1} + \phi_{i,j+1}^{n}\right)\frac{\Delta x^2}{\Delta y^2 + \Delta x^2} + \frac{\Delta x^2 \Delta y^2}{\Delta y^2 + \Delta x^2}\frac{c}{\mu}\right) + \mathcal{O}\left(\Delta x^2, \Delta y^2\right) \tag{4}$$

3.2 Boundary Conditions

The computational mesh is a grid so we will set all the cell outside the channel and on the walls to be zero. Therefore:

$$\begin{gathered} \phi|_{i_{AB}} = \phi|_{i_{CD}} = \phi|_{i_{GH}} = \phi|_{i_{EF}, j_{EF}} = 0 \\ \phi|_{j_{BC}} = \phi|_{i_{DE}, j_{DE}} = \phi|_{i_{FG}, j_{FG}} = \phi|_{j_{AH}} = 0 \end{gathered} \tag{5}$$

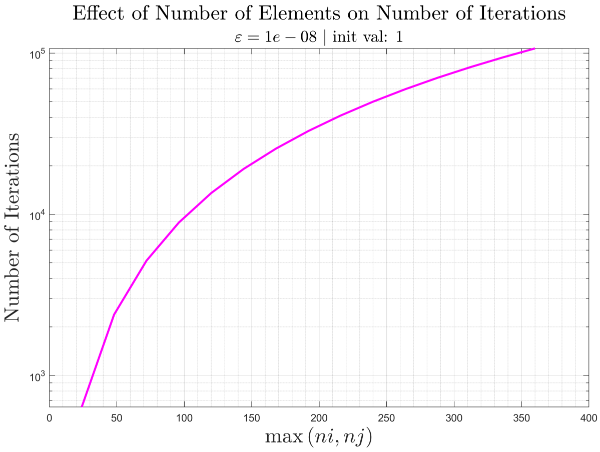
3.3 Convergence Criteria

In order determined if the iterative method for solving the system of equation has converged, we will check if the temperature vector at a specific time has changed from step n to step *n+1* in the following way:

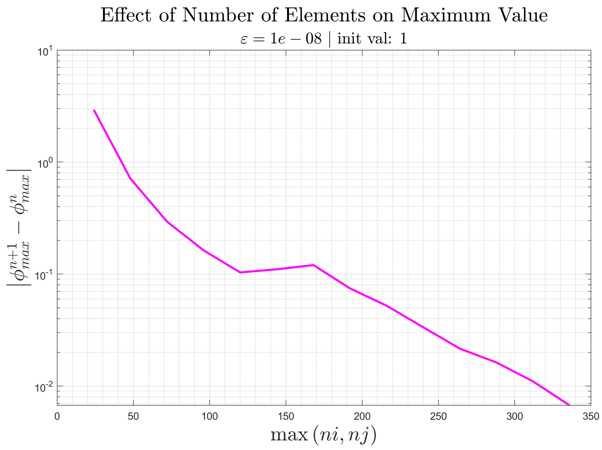
$$\left|T_{i,j}^{n+1} - T_{i,j}^{n}\right| < \varepsilon \tag{6}$$

4 Influence of The Numerical Methods

4.1 Influence of Number of Elements ni, nj



(a) Number of iterations as a function of maximum of ni and nj

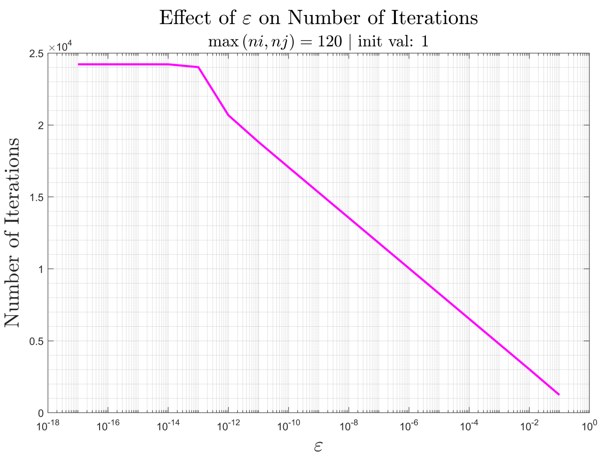


(b) Change in max value as a function of maximum of ni and nj

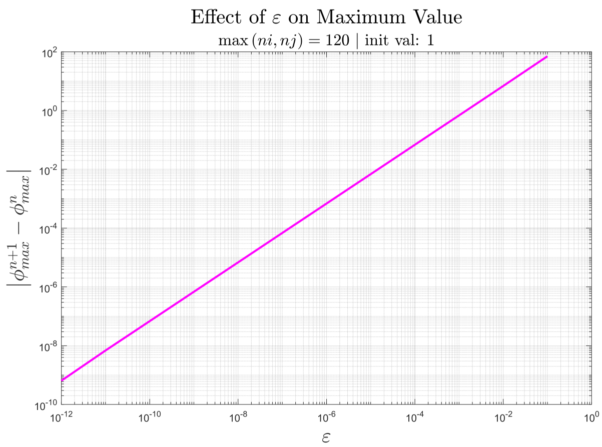
Figure 2: Influence of the number of elements ni nj

We can see that as the number of grid points increases, the number of iterations increases and the error decreases. Since there is no clear optimal number of grid points we will choose a number that will have a sufficiently small error but not too much of iterations. Hence the chosen number of grid points will be 120.

4.2 Influence of Convergence Criteria ε



(a) Number of iterations as a function of ε

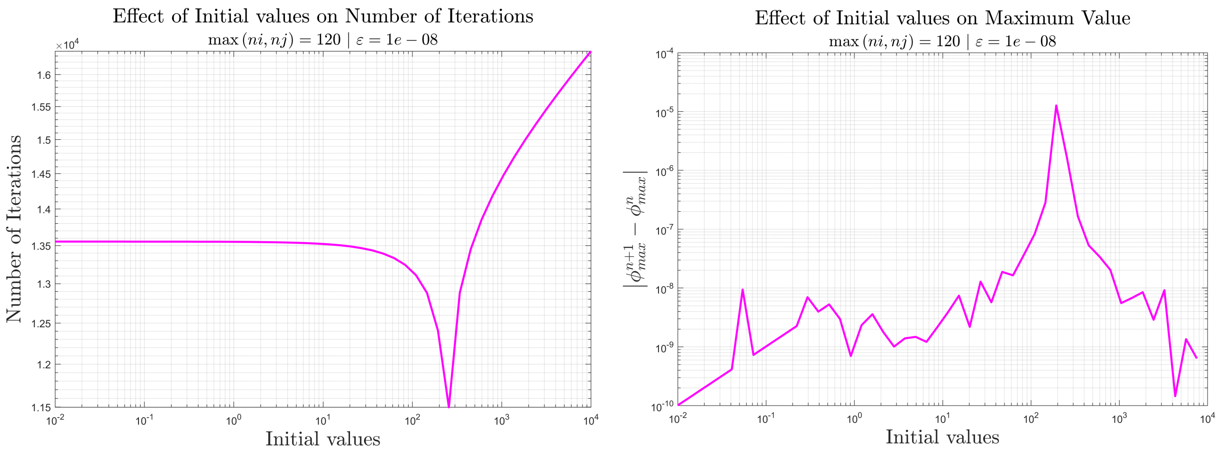


(b) Change in max value as a function of ε

Figure 3: Influence of the convergence criteria ε

Figure 3 shows that as the convergence criteria decreases, the number of iterations increases up to a certain point. This point is probably the accuracy limitation of floating double point. Moreover, the error decreases exponentially with the decrease of the convergence criteria. We can declare that $\varepsilon = 1 \cdot 10^{-8}$ is a good value for the convergence criteria.

4.3 Influence of Initial Conditions



(a) Number of iterations as a function of initial value

(b) Change in max value as a function of initial value

Figure 4: Influence of the initial value

We can see from Fig.4 that when the initial value is close the the maximal value of the flow, the number of iterations decreases but the error increases. Furthermore, for all initial values below 10, the number of iterations stays constant and the error has close to no change. So all initial values smaller than 10 are good and we will choose the initial value to be 1.

5 Results and Discussion

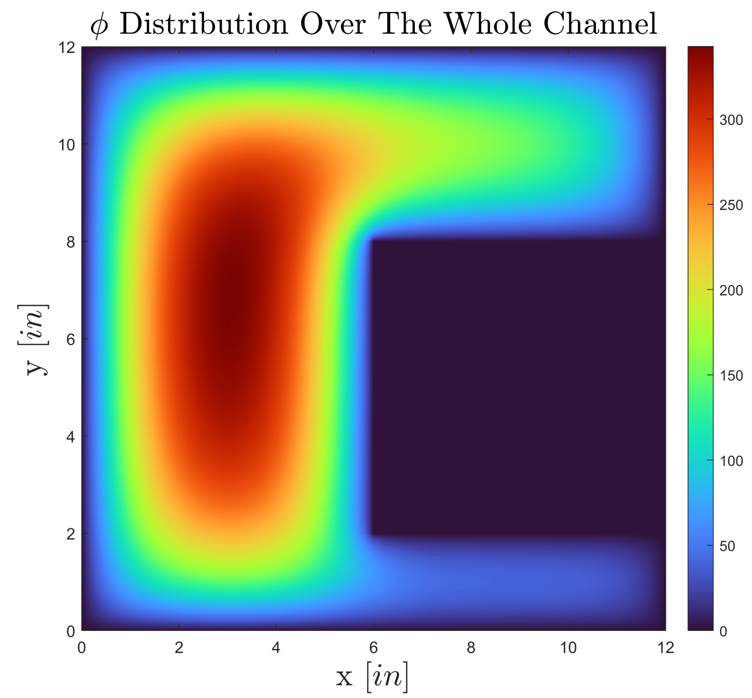


Figure 5: ϕ distribution over the whole field

We can see that as expected, the highest value of ϕ in the flow field is at the widest section of the channel. Moreover, this figure helps us to make sure that the no-slip and no-penetration conditions are enforced.

6 Summary and Conclusion

A Listing of The Computer Program

A.1 Parameters

```
x_min = 0;
x_max = 12;
y_min = 0;
y_max = 12;
c  = 0.0002;
mu = 0.25e-5;
factor = 4;
init_val = 1;
epsilon = 1e-5;
```

Listing 1: Parameters file

```
AB = 12; BC = 12; CD = 2; DE = 6 ;
EF = 6 ; FG = 6 ; GH = 4; HA = 12;

ni = lcm(BC, BC-DE) * factor;
nj = lcm(lcm(CD, AB), lcm(CD+EF, AB)) * factor;
delta_x = (x_max - x_min) / (ni);
delta_y = (y_max - y_min) / (nj);
x_vec = x_min + (0:ni)*delta_x;
y_vec = y_min + (0:nj)*delta_y;

i_AB = 0;
j_AB = 0:(AB / delta_y);

i_BC = 0:(BC / delta_x);
j_BC = 0;

i_CD = BC / delta_x;
j_CD = 0:(CD / delta_y);

i_DE = ((BC - DE) / delta_x):(BC / delta_x);
j_DE = CD / delta_y;

i_EF = (BC - DE) / delta_x;
j_EF = (CD / delta_y):((CD + EF) / delta_y);

i_FG = ((BC - DE) / delta_x):(BC / delta_x);
j_FG = (CD + EF) / delta_y;

i_GH = BC / delta_x;
j_GH = ((CD + EF) / delta_y):((CD + EF + GH) / delta_y);

i_HA = i_BC;
j_HA = (AB / delta_y);

for j = j_AB+1
    x_mat(j, :) = x_vec;
end
for i = i_BC+1
    y_mat(:, i) = y_vec;
end
```

Listing 2: Extra parameters file

A.2 Main Code

```
clc; clear; close all;

% flow_field_init = init_flow_field(ni, nj, i_AB, j_AB, i_BC, j_BC, i_CD, j_CD, i_DE, j_EF, i_GH, j_GH, i_HA, j_HA);
% figure
% contourf(x_mat, y_mat, flow_field_init, 100, "LineStyle","none")
% colormap('turbo')
% colorbar()
% axis equal

parameters
factor = 5;
% init_val = 0.1;

result = solve_Gauss_Seidel(x_min, x_max, y_min, y_max, c, mu, factor, init_val, epsilon);

fig1 = figure('Name','1','Position',[0, 250, 900, 600]);
contourf(result.x_mat, result.y_mat, result.flow_field, 200, "LineStyle","none")
colormap('turbo')
colorbar()
axis equal
title('$\phi$ Distribution Over The Hole Channel','FontSize',20,'Interpreter','latex')
xlabel('x $[in]$','FontSize',20,'Interpreter','latex')
ylabel('y $[in]$','FontSize',20,'Interpreter','latex')
% exportgraphics(fig1, 'images/phi ditribution.png','Resolution',400);


%% effect of factor =======================================================

parameters
factors = 1:1:15;
results = {};
lg = {};
for factor_index = 1:length(factors)
    factor = factors(factor_index);
    results{end+1} = solve_Gauss_Seidel(x_min, x_max, y_min, y_max, c, mu, factor, init_val, epsilon);
end

fig2 = figure('Name','2','Position',[0, 250, 900, 600]);
size = 20;
colors = cool(length(factors));
max_ni_nj_vec = [];
n_vec = [];
for factor_index = 1:length(factors)
    max_ni_nj_vec(factor_index) = max(results{factor_index}.ni, results{factor_index}.nj);
    n_vec(factor_index) = results{factor_index}.n;
end

semilogy(max_ni_nj_vec, n_vec, 'LineStyle','--','LineWidth',2,'Color',colors(end,:))

title('Effect of Number of Elements on Number of Iterations', 'FontSize', size,'Interpreter','latex')
```

```matlab
subtitle(sprintf('$\\varepsilon=%g$ $|$ init val: %g', results{end}.epsilon, results{end}.init_val), 'FontSize',size-4,'Interpreter','latex')
ylabel('Number of Iterations', 'FontSize', size,'Interpreter','latex')
xlabel('$\max{\left(ni,nj\right)}$', 'FontSize', size,'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig2, 'images/ni nj - n.png','Resolution',400);

fig3 = figure('Name','3','Position',[200, 250, 900, 600]);
size = 20;
colors = cool(length(factors));
max_vec = [];
for factor_index = 1:length(factors)-1
    max_vec(factor_index) = calc_diff(results{factor_index}, results{factor_index+1});
end

semilogy(max_ni_nj_vec(1:end-1), max_vec, 'LineStyle','-','LineWidth',2,'Color',colors(end,:))

title('Effect of Number of Elements on Maximum Value', 'FontSize', size,'Interpreter','latex')
subtitle(sprintf('$\\varepsilon=%g$ $|$ init val: %g', results{end}.epsilon, results{end}.init_val), 'FontSize',size-4,'Interpreter','latex')
ylabel('$\left|\phi_{max}^{n+1}-\phi_{max}^{n}\right|$', 'FontSize', size,'Interpreter','latex')
xlabel('$\max{\left(ni,nj\right)}$', 'FontSize', size,'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig3, 'images/ni nj - max diff.png','Resolution',400);
% exportgraphics(fig2, 'images/ni nj - n.png','Resolution',400); exportgraphics(fig3, 'images/ni nj - max diff.png','Resolution',400);


%% effect of epsilon ======================================================

parameters
epsilons = logspace(-1, -17, 17);
% epsilons = logspace(0, -5, 6);
results = {};
lg = {};
for epsilons_index = 1:length(epsilons)
    epsilon = epsilons(epsilons_index);
    results{end+1} = solve_Gauss_Seidel(x_min, x_max, y_min, y_max, c, mu, factor, init_val, epsilon);
end

fig4 = figure('Name','4','Position',[400, 250, 900, 600]);
size = 20;
colors = cool(length(epsilons));
max_ni_nj_vec = [];
n_vec = [];
for epsilons_index = 1:length(epsilons)
    n_vec(epsilons_index) = results{epsilons_index}.n;
end

semilogx(epsilons, n_vec, 'LineStyle','-','LineWidth',2,'Color',colors(end,:))
```

```
title('Effect of $\phi$ on Number of Iterations', 'FontSize', size,'Interpreter','latex')
subtitle(sprintf('$\\max{\\left(ni,nj\\right)}=%d$ $|$ init val: %g', max(results{end}.ni, results{end}.nj), results{end}.init_val), 'FontSize',size-4,'Interpreter','latex')
ylabel('Number of Iterations', 'FontSize', size,'Interpreter','latex')
xlabel('$\varepsilon$', 'FontSize', size,'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig4, 'images/epsilon - n.png','Resolution',400);

fig5 = figure('Name','5','Position',[600, 250, 900, 600]);
size = 20;
colors = cool(length(epsilons));
max_vec = [];
for epsilons_index = 1:length(epsilons)-1
    max_vec(epsilons_index) = calc_diff(results{epsilons_index}, results{epsilons_index+1});
end

loglog(epsilons(1:end-1), max_vec, 'LineStyle','-','LineWidth',2,'Color',colors(end,:))

title('Effect of $\phi$ on Maximum Value', 'FontSize', size,'Interpreter','latex')
subtitle(sprintf('$\\max{\\left(ni,nj\\right)}=%d$ $|$ init val: %g', max(results{end}.ni, results{end}.nj), results{end}.init_val), 'FontSize',size-4,'Interpreter','latex')
ylabel('$\left|\phi_{max}^{n+1}-\phi_{max}^{n}\right|$', 'FontSize', size,'Interpreter','latex')
xlabel('$\varepsilon$', 'FontSize', size,'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig5, 'images/epsilon - max diff.png','Resolution',400);
% exportgraphics(fig5, 'images/epsilon - max diff.png','Resolution',400); exportgraphics(fig4, 'images/epsilon - n.png','Resolution',400);


%% effect of init val ======================================================

parameters
init_vals = logspace(-2,4,50);
results = {};
lg = {};
for init_vals_index = 1:length(init_vals)
    init_val = init_vals(init_vals_index);
    results{end+1} = solve_Gauss_Seidel(x_min, x_max, y_min, y_max, c, mu, factor, init_val, epsilon);
end

fig6 = figure('Name','6','Position',[0, 150, 900, 600]);
size = 20;
colors = cool(length(init_vals));
max_ni_nj_vec = [];
n_vec = [];
for init_vals_index = 1:length(init_vals)
    n_vec(init_vals_index) = results{init_vals_index}.n;
end

loglog(init_vals, n_vec, 'LineStyle','-','LineWidth',2,'Color',colors(end,:))
```

```

title('Effect of Initial values on Number of Iterations', 'FontSize', size,'Interpreter', 'latex')
subtitle(sprintf('$\\max{\\left(ni,nj\\right)}=%d$ $|$ $\\varepsilon=%g$', max(results{end}.ni, results{end}.nj), results{end}.epsilon), 'FontSize',size-4,'Interpreter','latex')
ylabel('Number of Iterations', 'FontSize', size,'Interpreter','latex')
xlabel('Initial values', 'FontSize', size,'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig6, 'images/init val - n.png','Resolution',400);

fig7 = figure('Name','7','Position',[200, 150, 900, 600]);
size = 20;
colors = cool(length(init_vals));
max_vec = [];
for init_vals_index = 1:length(init_vals)-1
    max_vec(init_vals_index) = calc_diff(results{init_vals_index}, results{init_vals_index+1});
end

loglog(init_vals(1:end-1), max_vec, 'LineStyle','--','LineWidth',2,'Color',colors(end,:))

title('Effect of Initial values on Maximum Value', 'FontSize', size,'Interpreter','latex')
subtitle(sprintf('$\\max{\\left(ni,nj\\right)}=%d$ $|$ $\\varepsilon=%g$', max(results{end}.ni, results{end}.nj), results{end}.epsilon), 'FontSize',size-4,'Interpreter','latex')
ylabel('$\left|\phi_{max}^{n+1}-\phi_{max}^{n}\right|$', 'FontSize', size,'Interpreter','latex')
xlabel('Initial values', 'FontSize', size,'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig7, 'images/init val - max diff.png','Resolution',400);
% exportgraphics(fig7, 'images/init val - max diff.png','Resolution',400); exportgraphics(fig6, 'images/init val - n.png','Resolution',400);





%% Functions ==============================================================

function flow_field = set_BC(flow_field, i_AB, j_AB, i_BC, j_BC, i_CD, j_CD, i_DE, j_EF, i_GH, j_GH, i_HA, j_HA)
    for i = i_AB+1
        for j = j_AB+1
            flow_field(j, i) = 0;
        end
    end
    for i = i_BC+1
        for j = j_BC+1
            flow_field(j, i) = 0;
        end
    end
    for i = i_CD+1
```

```
        for j = j_CD+1
            flow_field(j, i) = 0;
        end
    end
    for i = i_DE+1
        for j = j_EF+1
            flow_field(j, i) = 0;
        end
    end
    for i = i_GH+1
        for j = j_GH+1
            flow_field(j, i) = 0;
        end
    end
    for i = i_HA+1
        for j = j_HA+1
            flow_field(j, i) = 0;
        end
    end
end

function flow_field = init_flow_field(ni, nj, init_val, i_AB, j_AB, i_BC, j_BC, i_CD, j_CD, i_DE, j_EF, i_GH, j_GH, i_HA, j_HA)
    flow_field = ones(nj+1,ni+1) * init_val;
    flow_field = set_BC(flow_field, i_AB, j_AB, i_BC, j_BC, i_CD, j_CD, i_DE, j_EF, i_GH, j_GH, i_HA, j_HA);
end

function converged = check_convergence(flow_field_next, flow_field_current, epsilon,j_AB, i_BC)
    converged = true;
    for i = i_BC+1
        for j = j_AB+1
            if abs(flow_field_next(j, i) - flow_field_current(j, i)) > epsilon
                converged = false;
                return
            end
        end
    end
end

function result = solve_Gauss_Seidel(x_min, x_max, y_min, y_max, c, mu, factor, init_val, epsilon)
    calc_extra_param
    flow_field_current = init_flow_field(ni, nj, init_val, i_AB, j_AB, i_BC, j_BC, i_CD, j_CD, i_DE, j_EF, i_GH, j_GH, i_HA, j_HA);
    flow_field_next = flow_field_current;
    for n = 1:1e6
        if ~mod(n,100)
            fprintf('factor: %d | epsilon: %g | init val: %g | n: %d\n', factor, epsilon, init_val, n)
        end
        for i = i_BC(2:end-1)+1
            for j = j_AB(2:end-1)+1
                % flow_field_next(j, i) = 0.5 * ((flow_field_next(j, i-1) + flow_field_current(j, i+1)) * delta_y^2 / (delta_y^2+delta_x^2) + (flow_field_next(j-1, i) + flow_field_current(j+1, i)) * delta_x^2 / (delta_y^2+delta_x^2) + delta_x^2*delta_y^2 / (delta_y^2+delta_x^2) *
```

```
                    c / mu);
                flow_field_next(j, i) = 0.5 * ((flow_field_current(j, i-1) +
                    flow_field_current(j, i+1)) * delta_y^2 / (delta_y^2+delta_x^2) + (
                    flow_field_current(j-1, i) + flow_field_current(j+1, i)) * delta_x^2
                    / (delta_y^2+delta_x^2) + delta_x^2*delta_y^2 / (delta_y^2+delta_x^2)
                     * c / mu);
            end
        end
        flow_field_next = set_BC(flow_field_next, i_AB, j_AB, i_BC, j_BC, i_CD, j_CD,
            i_DE, j_EF, i_GH, j_GH, i_HA, j_HA);
        if check_convergence(flow_field_next, flow_field_current, epsilon,j_AB, i_BC)
            break
        end
        flow_field_current = flow_field_next;
    end
    result.n          = n;
    result.x_min      = x_min;
    result.x_max      = x_max;
    result.y_min      = y_min;
    result.y_max      = y_max;
    result.c          = c;
    result.mu         = mu;
    result.factor     = factor;
    result.epsilon    = epsilon;
    result.x_mat      = x_mat;
    result.y_mat      = y_mat;
    result.delta_x    = delta_x;
    result.delta_y    = delta_y;
    result.ni         = ni;
    result.nj         = nj;
    result.i_BC       = i_BC;
    result.j_AB       = j_AB;
    result.init_val   = init_val;
    result.flow_field = flow_field_next;
end

% function rms = RMS(result1, result2)
% % This functions calcultates the RMS value for 1 compared to 2
%     flow_field2_at_coord_of_1 = [];
%     for i = result1.i_BC+1
%         for j = result1.j_AB+1
%             flow_field2_at_coord_of_1(j, i) = interp2(result2.x_mat, result2.y_mat,
    result2.flow_field, result1.x_min + i*result1.delta_x, result1.y_min + j*result1.
    delta_y);
%         end
%     end
%     flow_field2_at_coord_of_1(isnan(flow_field2_at_coord_of_1)) = 0;
%     flow_field1 = result1.flow_field;
%     rms = sqrt(sum(sum((flow_field1-flow_field2_at_coord_of_1).^2)) / (length(
    flow_field1(:,1)) * length(flow_field1(1,:))));
% end

function diff = calc_diff(result1, result2)
    max1 = max(max(result1.flow_field));
    max2 = max(max(result2.flow_field));
    diff = abs(max1-max2);
end
```

Listing 3: The main file