UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Computational Engineering
Master's degree in Space and Aeronautical Engineering
Universitat Politècnica de Catalunya

Assignment 2 (CFD)

Generic convection-diffusion transport equation

González Martínez, Víctor

Contents

Assignment 2 (CFD):
Generic convection-diffusion transport equation
Autumn 2025.
Víctor González Martínez

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

1 Theoretical Introduction

The convection–diffusion equation stands as one of the fundamental principles in computational engineering applied to transport phenomena. It describes how a scalar quantity—such as temperature, concentration, or any conservative property—is transported within a fluid as a result of both the motion of the medium and the gradients of the scalar itself. In essence, the total transport of a scalar variable combines two complementary mechanisms: **convection**, associated with the bulk motion of the fluid, and **diffusion**, which acts to smooth local variations through molecular or turbulent processes.

Mathematically, the steady-state two-dimensional form of the equation can be written as:

$$\nabla \cdot (\rho \vec{v} \phi) = \nabla \cdot (\Gamma \nabla \phi) + S \tag{1.0.1}$$

where ϕ is the transported variable, ρ is the fluid density, $\vec{v} = (u, v)$ represents the velocity field, Γ is the diffusion coefficient, and S is a source or sink term. The equation expresses that the net flux of the property through a control volume must balance the effects of diffusion and internal generation.

A key parameter governing the behavior of the system is the **Péclet number (Pe)**, which quantifies the relative importance of convection versus diffusion. Small Pe values indicate diffusion-dominated regimes, while large values correspond to convection-dominated flows. In the latter case, steep gradients tend to appear, and numerical instabilities may arise unless an appropriate discretization scheme is employed.

1.1 Numerical approximation using the Finite Volume Method

Analytical solutions to this equation are only available for a few simplified cases. In most practical applications, numerical methods are required to approximate the distribution of ϕ throughout the domain. In this work, the **Finite Volume Method (FVM)** is employed due to its conservative nature and robustness in computational fluid dynamics (CFD). The core idea of the method is to integrate the transport equation over a set of control volumes that divide the domain, ensuring that the balance between convective, diffusive, and source fluxes is satisfied in each cell.

By applying the **Gauss–Ostrogradsky theorem**, the divergence terms are converted into surface integrals, which represent the fluxes across the faces of each control volume.

The variable ϕ is stored at cell centers, while convective and diffusive fluxes are evaluated at cell faces using suitable interpolation schemes. This process leads to a system of algebraic equations that can be solved iteratively until a stationary solution is reached.

1.2 Stability and accuracy considerations

Every numerical approach must satisfy three essential properties: **consistency**, **stability**, and **convergence**. A scheme is consistent when it reproduces the original differential equation as the mesh size tends to zero; it is stable when numerical errors do not grow uncontrollably; and it is convergent when the discrete solution approaches the exact one as the grid is refined. Within this framework, the choice of face-interpolation scheme—such as Upwind, Central, Hybrid, Exponential, or Power Law—is crucial to ensure both the stability of the calculation and the accuracy of the resulting field.

1.3 Assumptions and scope of the study

The present work focuses on the analysis of the convection–diffusion equation under **two-dimensional, steady, laminar** conditions, assuming a **Newtonian, incompressible fluid with constant properties**. Internal sources and density variations are neglected. Under these simplifying assumptions, two benchmark configurations are considered: a **Diagonal Flow Problem** and the **Smith–Hutton Problem**, both widely used as validation cases in computational fluid dynamics.

The aim of this study is to compare the performance of several discretization schemes within the Finite Volume framework, evaluating their capability to correctly predict flows under different convection–diffusion dominance conditions.

2 Problem Description

2.1 Diagonal Flow Problem

The first case corresponds to a two-dimensional steady flow in which the velocity field is uniform and oriented diagonally at 45°. This flow pattern causes the fluid to move across the domain from one corner to the opposite one, in a constant direction and without any recirculations.

The velocity field is defined as:

$$u = V_{\text{in}} \cos\left(\frac{\pi}{4}\right), \quad v = V_{\text{in}} \sin\left(\frac{\pi}{4}\right) \tag{2.1.1}$$

where both components have the same magnitude, resulting in a constant flow direction along the main diagonal of the domain.

The scalar field ϕ is imposed with fixed boundary values:

- **Inlet:** high values of ϕ, representing the transported quantity entering the domain.
- **Outlet:** low values of ϕ, corresponding to the region where the flow exits.
- **Walls (top and bottom):** constant values of ϕ that confine the scalar field within the domain.

In this configuration, the transport of ϕ follows the flow trajectory, creating a continuous gradient from the inlet to the outlet. This case is particularly useful for verifying the correct implementation of convective and diffusive fluxes in numerical schemes, as the expected physical behavior is smooth and predictable.

2.2 Smith–Hutton Problem

The second case, the **Smith–Hutton problem**, features a two-dimensional non-uniform velocity field that induces a rotational motion within the domain. It is a well-known benchmark in computational fluid dynamics, as it combines dominant convective transport with curved flow patterns and sharp scalar gradients.

The velocity field is defined by the following expressions:

$$u(x, y) = 2y(1 - x^2), \quad v(x, y) = -2x(1 - y^2) \tag{2.2.1}$$

These velocity components generate a clockwise flow: the fluid rises along the left side, turns right along the top, and descends near the right-hand boundary.

The behavior of the scalar field ϕ is determined by the following boundary conditions:

- **Inlet:** located at the lower-left region ($y = 0, x \in [-1, 0]$), where ϕ is prescribed with a smooth hyperbolic tangent profile that gradually introduces the scalar into the domain:

$$\phi = 1 + \tanh[\alpha(2x + 1)] \tag{2.2.2}$$

- **Outlet:** at the lower-right boundary ($y = 0, x \in [0, 1]$), where a zero normal-gradient condition is applied:

$$\frac{\partial \phi}{\partial y} = 0 \tag{2.2.3}$$

  allowing the scalar to exit freely from the domain.

- **Walls:** all remaining boundaries are fixed at a constant value of ϕ, defined as:

$$\phi = 1 - \tanh(\alpha) \tag{2.2.4}$$

The result is a rotational field in which the distribution of ϕ is distorted following the streamlines, displaying a characteristic transport and diffusion pattern of convection-dominated flows. This case is particularly useful for comparing the stability and accuracy of different interpolation schemes in the presence of recirculations and strong gradients.

3 Numerical Schemes Used

The solution of the convection–diffusion equation using the Finite Volume Method requires an accurate approximation of the fluxes across the control volume faces. To achieve this, it is necessary to define an interpolation scheme that determines the value of the scalar variable ϕ at the faces from the known values at the cell centers. The choice of scheme is crucial, as it determines the stability, accuracy, and the possible appearance of numerical oscillations, especially in convection-dominated regimes.

In this work, five different interpolation schemes are implemented and compared in order to analyze their behavior under different Péclet number conditions.

3.1 Upwind Differencing Scheme (UDS)

The **Upwind Differencing Scheme (UDS)** uses the value of the variable ϕ from the upstream node, i.e., from the direction the flow comes from. In this way, the convective transport is evaluated using only the incoming information. Its formulation for the interpolation coefficient is:

$$A(|P|) = 1 \tag{3.1.1}$$

This method ensures stability even for strongly convective flows but introduces **significant numerical diffusion**, which artificially smooths the gradients of the scalar field. For this reason, the UDS is considered a **robust but low-accuracy** scheme.

3.2 Central Differencing Scheme (CDS)

The **Central Differencing Scheme (CDS)** calculates the value of ϕ at the face as the weighted average of the two adjacent cell-center values. Its expression for the interpolation coefficient depends linearly on the local Péclet number and is defined as:

$$A(|P|) = 1 - 0.5|P| \tag{3.2.1}$$

This scheme provides **higher spatial accuracy**, especially when diffusion is significant. However, since it does not take into account the flow direction, it may produce **non-physical oscillations** in convection-dominated situations, limiting its use to low or moderate Péclet number cases.

3.3 Hybrid Differencing Scheme (HDS)

The **Hybrid Differencing Scheme (HDS)** combines the advantages of the two previous methods. It applies the central scheme when convection is weak and automatically switches to the upwind scheme when convection becomes dominant. Its formulation is expressed as:

$$A(|P|) = \max(0, 1 - 0.5|P|) \tag{3.3.1}$$

This approach maintains stability under all flow regimes and improves accuracy in diffusion-dominated regions. Although it achieves a good balance between robustness and precision, it still introduces some numerical diffusion under strong convection.

3.4 Exponential Differencing Scheme (EDS)

The **Exponential Differencing Scheme (EDS)** is derived from the one-dimensional analytical solution of the convection–diffusion equation, assuming exponential variations of the scalar field between nodes. The interpolation coefficient is defined as a function of the local Péclet number:

$$A(|P|) = \frac{|P|}{e^{|P|} - 1} \tag{3.4.1}$$

This formulation directly accounts for the effect of the Péclet number and provides a **more accurate interpolation** in both diffusion- and convection-dominated regimes. However, its implementation is more complex and computationally more expensive than the previous schemes.

3.5 Power Law Scheme (PLDS)

Finally, the **Power Law Scheme (PLDS)** introduces an empirical correction based on a power-law expression that progressively reduces the diffusive contribution as the Péclet number increases. Its general formulation is given by:

$$A(|P|) = \max\left(0, (1 - 0.5|P|)^5\right) \tag{3.5.1}$$

where $A(|P|)$ represents the interpolation coefficient and P is the local Péclet number at the face. The Power Law scheme offers a **good compromise between accuracy and stability**, and is considered one of the most suitable options for convection-dominated problems such as the Smith–Hutton case.

4 Code Structure

The implemented code for solving the convection–diffusion problems is organized into several well-defined blocks. Its structure aims to maintain clarity and modularity, allowing different mesh configurations, schemes, and diffusion regimes to be tested.

First, the **initial parameters** and the problem/scheme selectors are defined. The variable `problem_type` allows choosing between the two analyzed cases: the **Diagonal Problem (1)** and the **Smith–Hutton Problem (2)**. Likewise, the parameter `scheme` determines the interpolation method used for the control-volume faces, which can be one of the following:

1. UDS (Upwind Difference Scheme)
2. CDS (Central Difference Scheme)
3. HDS (Hybrid Difference Scheme)
4. EDS (Exponential Difference Scheme)
5. PLDS (Power-Law Difference Scheme)

Next, the **geometric and physical conditions** of each case are defined (length, height, inlet velocity, reference density, and diffusion coefficient). The vectors and matrices that compose the **computational grid** are generated for both cell-center and cell-edge coordinates. In the Smith–Hutton problem, the domain extends between $-1 \leq x \leq 1$ and $0 \leq y \leq 1$, while the diagonal problem covers a rectangular channel with flow at a 45° angle.

The following block constructs the **velocity field**:

- **Diagonal Problem:** constant velocity components are imposed,

$$u = V_{in}\cos(45^\circ), \qquad v = V_{in}\sin(45^\circ),$$

  generating a uniform diagonal flow.

- **Smith–Hutton Problem:** the field is bidimensional and steady, defined by

$$u(x,y) = 2y(1-x^2), \qquad v(x,y) = -2x(1-y^2),$$

  describing a closed recirculation within the domain, with curved and symmetric streamlines.

Subsequently, **storage matrices** are initialized for convection and diffusion coefficients (F and D), Péclet numbers, and interpolation weights. The coefficients of the linear system (a_N, a_S, a_E, a_W, a_P) and the source term b_P are also preallocated, avoiding dynamic resizing within loops and improving computational efficiency.

Each numerical scheme is implemented through its **weighting function** $A(|P|)$, depending on the face Péclet number:

$$A(|P|) = \begin{cases} 1 & \text{(UDS)} \\ 1 - 0.5|P| & \text{(CDS)} \\ \max(0, 1 - 0.5|P|) & \text{(HDS)} \\ \dfrac{|P|}{e^{|P|} - 1} & \text{(EDS)} \\ \max(0, (1 - 0.5|P|)^5) & \text{(PLDS)} \end{cases}$$

These coefficients determine the balance between convection and diffusion, adapting to the flow regime.

The **boundary conditions** vary according to the problem:

- **Diagonal Problem:** the scalar field $\phi = 1$ is imposed on the left and bottom walls, and $\phi = 0$ on the top and right walls.
- **Smith–Hutton Problem:** a non-uniform inlet profile is applied:

$$\phi_{inlet} = 1 + \tanh[\alpha(2x + 1)], \quad x \in [-1, 0],\ y = 0,$$

and constant values on the walls and outlet:

$$\phi_{walls} = \phi_{outlet} = 1 - \tanh(\alpha),$$

where α controls the slope of the inlet profile.

The discretized system of equations is solved using an **iterative Gauss–Seidel method**, updating ϕ values until a fixed number of iterations is reached. This process is repeated for three different ρ/Γ ratios (10, 10^3, 10^6) to analyze diffusion-dominated, mixed, and convection-dominated regimes.

Finally, the **post-processing stage** generates contour maps of ϕ for each scheme and ρ/Γ value. These plots allow assessing the influence of the numerical scheme and the flow regime on the sharpness and stability of the transported scalar front.

5 Results

5.1 Diagonal Problem

Scheme 1 (UDS) (Fig. 1, Fig. 2, Fig.3). In the most diffusive regime, the diagonal front of ϕ appears wide and smooth, with a uniform transition. As the ρ/Γ ratio increases, the transition band progressively narrows, maintaining stability and no oscillations.

Scheme 2 (CDS) (Fig. 4, Fig. 5, Fig.6). The field presents greater sharpness than UDS. The front is clean, well-defined, and thinner as s increases. The transitions remain smooth, although the gradient intensifies in convective regimes.

Scheme 3 (HDS) (Fig. 7, Fig. 8, Fig.9). A good balance between definition and stability is observed. The front becomes progressively thinner with increasing s, maintaining regular behavior without instabilities.

Scheme 4 (EDS) (Fig. 10, Fig. 11, Fig.12). The transition narrows uniformly as ρ/Γ increases. The front clearly adheres to the diagonal, and the field remains smooth and continuous across all regimes.

Scheme 5 (PLDS) (Fig. 13, Fig. 14, Fig.15). The front becomes very sharp, especially in the most convective regime. The transition is clean, oscillation-free, and numerically stable in all cases.

5.2 Smith–Hutton Problem

Scheme 1 (UDS) (Fig. 16, Fig. 17, Fig.18). The scalar field ϕ is smooth and symmetric, showing a semicircular distribution in the lower region. As s increases, the pattern slightly deforms toward the right due to convection but remains stable and continuous.

Scheme 2 (CDS) (Fig. 19, Fig. 20, Fig.21). In the diffusive regime, the contour lines are regular and smooth. As ρ/Γ increases, a slight asymmetry appears, and the front becomes steeper, reducing smoothness but without generating oscillations.

Scheme 3 (HDS) (Fig. 22, Fig. 23, Fig.24). The results are similar to CDS for $s = 1$, though with slightly higher stability at larger s values. The front becomes more concentrated and defined, maintaining a clean pattern.

Scheme 4 (EDS) (Fig. 25, Fig. 26, Fig.27). The field is stable and well-defined in all regimes. At intermediate s, some front deformation is observed, but the overall result remains coherent and symmetric.

Scheme 5 (PLDS) (Fig. 28, Fig. 29, Fig.30). The response becomes progressively sharper as s increases. The front concentrates in the lower region, and the field preserves smoothness and the absence of oscillations.

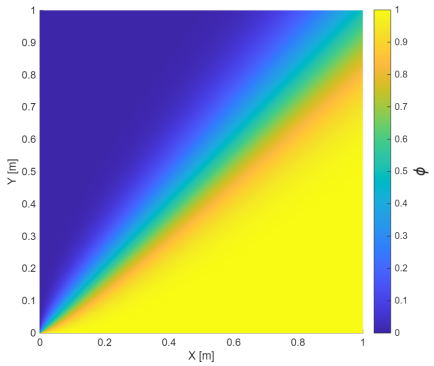

Figure 1: ϕ distribution ($\rho/\gamma = 10$)

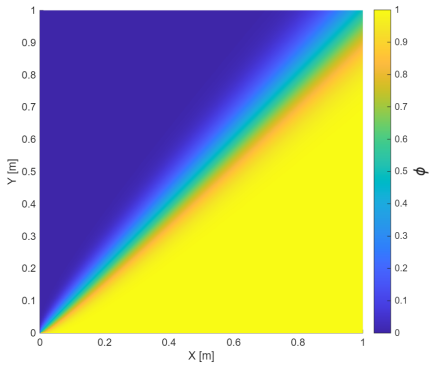

Figure 2: ϕ distribution ($\rho/\gamma = 10^3$)

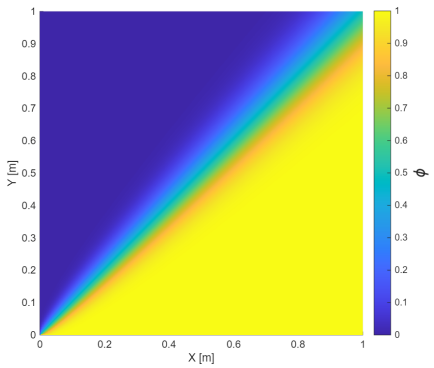

Figure 3: ϕ distribution ($\rho/\gamma = 10^6$)

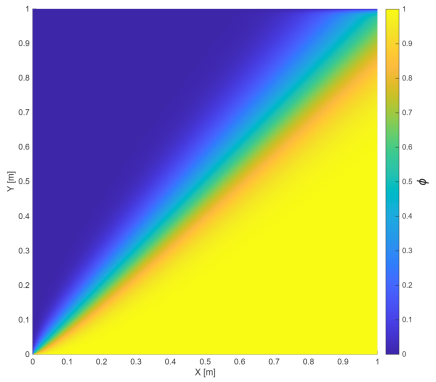

Figure 4: ϕ distribution ($\rho/\gamma = 10$)

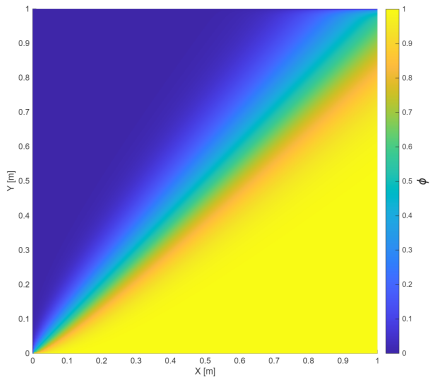

Figure 5: ϕ distribution ($\rho/\gamma = 10^3$)

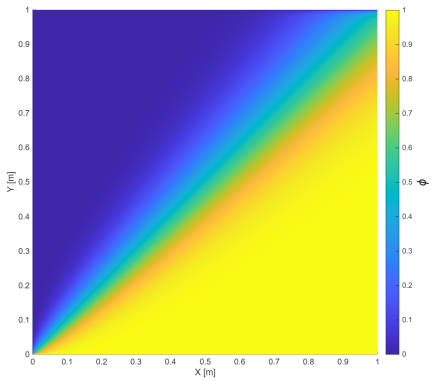

Figure 6: ϕ distribution ($\rho/\gamma = 10^6$)

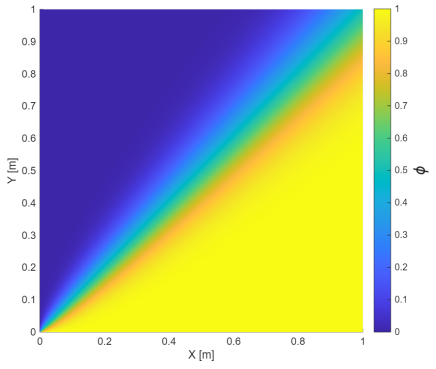

Figure 7: ϕ distribution ($\rho/\gamma = 10$)

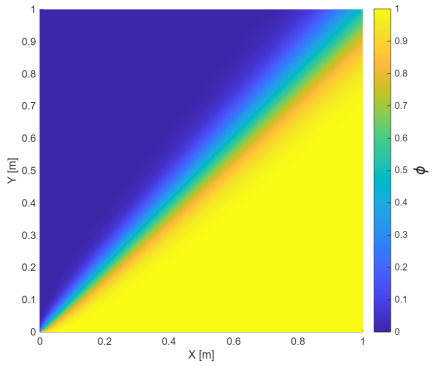

Figure 8: ϕ distribution ($\rho/\gamma = 10^3$)

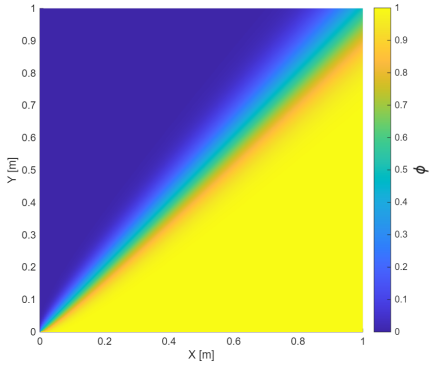

Figure 9: ϕ distribution ($\rho/\gamma = 10^6$)

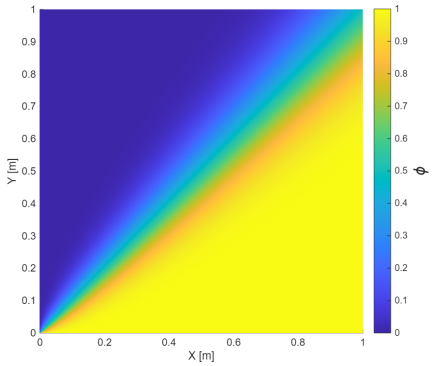

Figure 10: ϕ distribution ($\rho/\gamma = 10$)

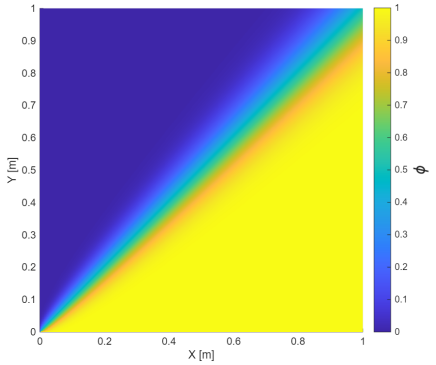

Figure 11: ϕ distribution ($\rho/\gamma = 10^3$)

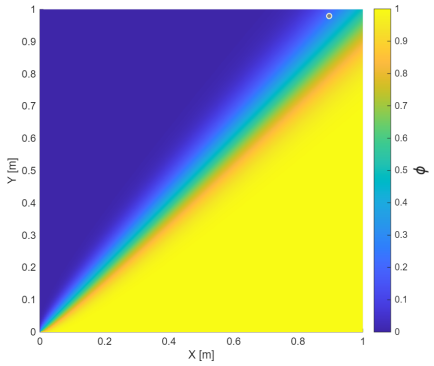

Figure 12: ϕ distribution ($\rho/\gamma = 10^6$)

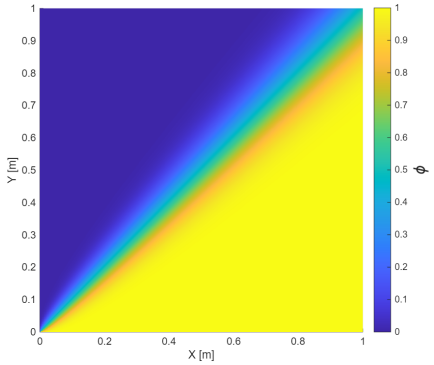

Figure 13: ϕ distribution ($\rho/\gamma = 10$)

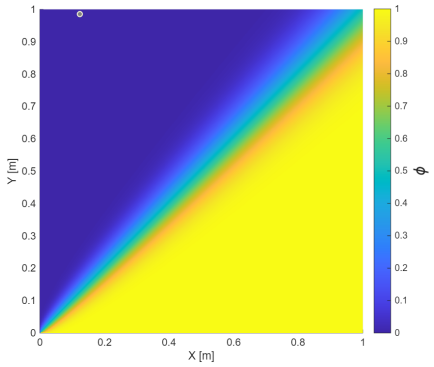

Figure 14: ϕ distribution ($\rho/\gamma = 10^3$)

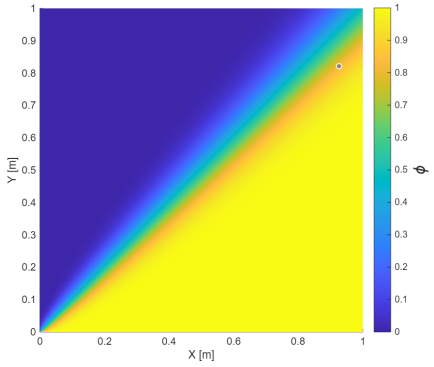

Figure 15: ϕ distribution ($\rho/\gamma = 10^6$)

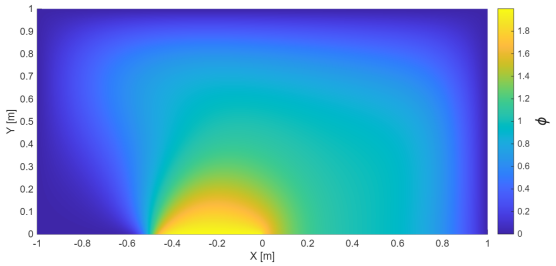

Figure 16: ϕ distribution ($\rho/\gamma = 10$)

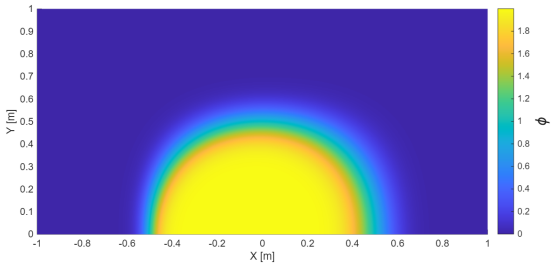

Figure 17: ϕ distribution ($\rho/\gamma = 10^3$)

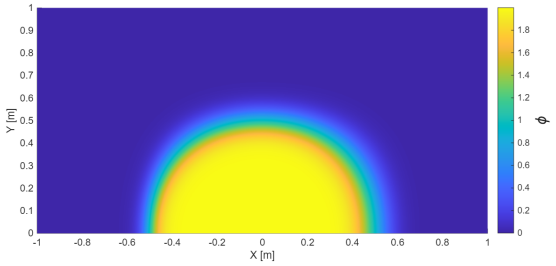

Figure 18: ϕ distribution ($\rho/\gamma = 10^6$)

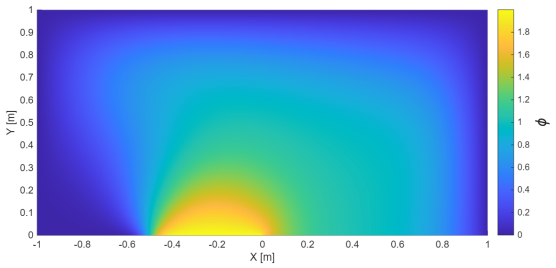

Figure 19: ϕ distribution ($\rho/\gamma = 10$)

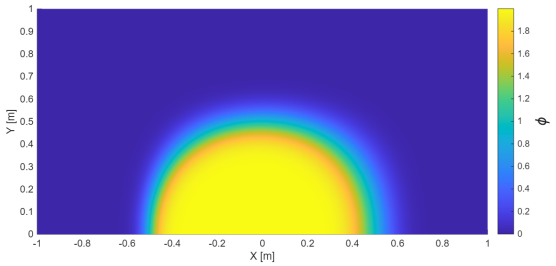

Figure 20: ϕ distribution ($\rho/\gamma = 10^3$)

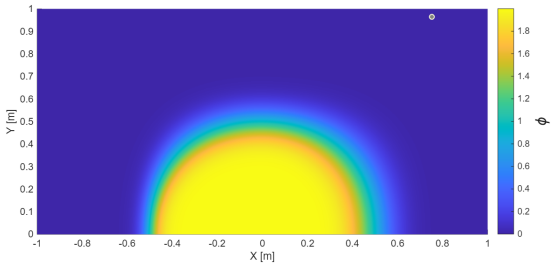

Figure 21: ϕ distribution ($\rho/\gamma = 10^6$)

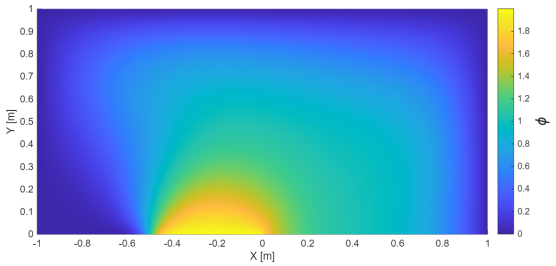

Figure 22: ϕ distribution ($\rho/\gamma = 10$)

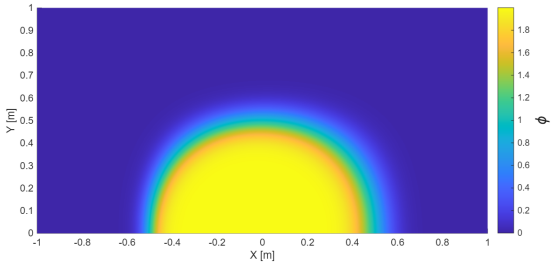

Figure 23: ϕ distribution ($\rho/\gamma = 10^3$)

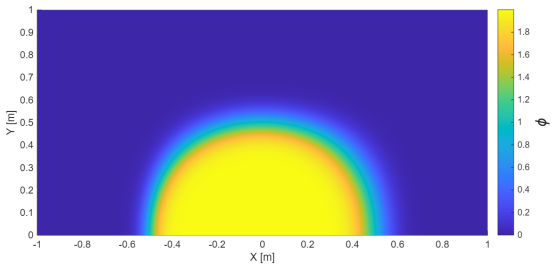

Figure 24: ϕ distribution ($\rho/\gamma = 10^6$)

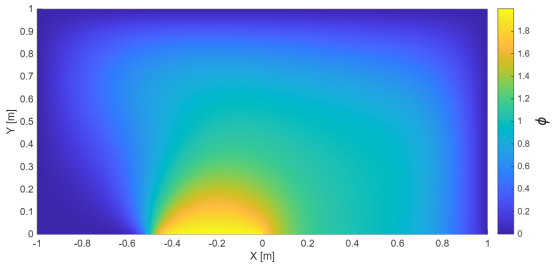

Figure 25: ϕ distribution ($\rho/\gamma = 10$)

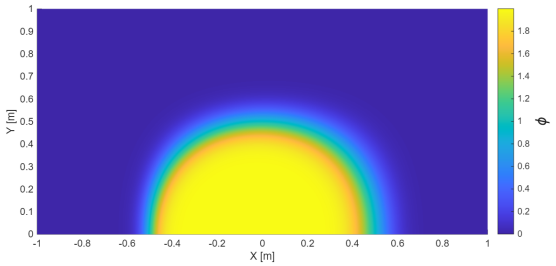

Figure 26: ϕ distribution ($\rho/\gamma = 10^3$)

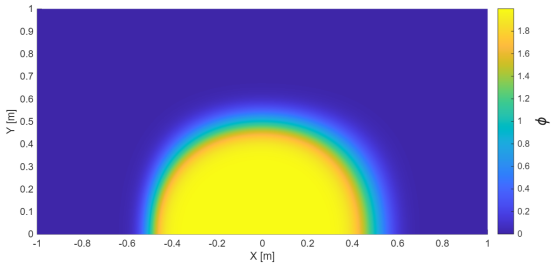

Figure 27: ϕ distribution ($\rho/\gamma = 10^6$)

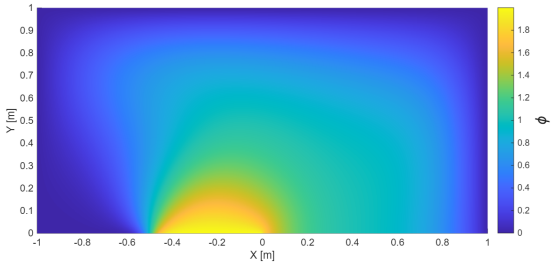

Figure 28: ϕ distribution ($\rho/\gamma = 10$)

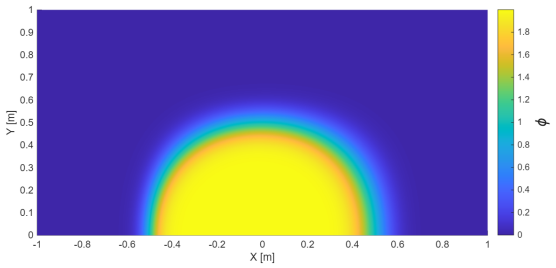

Figure 29: ϕ distribution ($\rho/\gamma = 10^3$)

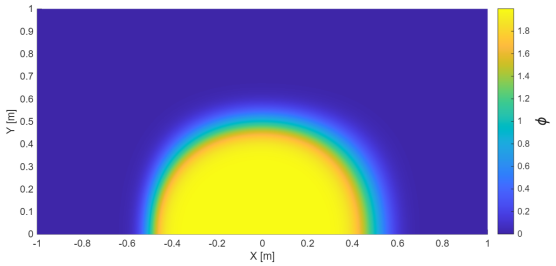

Figure 30: ϕ distribution ($\rho/\gamma = 10^6$)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

A Appendix

Matlab code used:

```
clc;
clear;
close all;

%% ASSIGNMENT 2: Víctor González Martínez
%% ESEIAAT MSc in Space and Aeronautical Engineering
% Problem type selector
problem_type = 2;    % 1 (Diagonal Problem); 2 (Smith-Hutton Problem)

% Scheme selector
scheme = 5;          % 1 (UDS (Upwind)); 2 (CDS (Central)); 3 (HDS (Hybrid)); 4 (EDS); 5 (PLDS)

% Data
if problem_type == 1
   L = 2.4;          % Channel length [m]
   H = 1.2;          % Channel height [m]
elseif problem_type == 2
   L = 2;            % Channel length [m]
   H = 1;            % Channel height [m]
end
V_in = 18;           % Inlet velocity [m/s]
rho_in = 1.2;        % Reference density [kg/m^3]

% Grid resolution and solver controls
N = 320;             % Number of control volumes in X
M = 160;             % Number of control volumes in Y
n_it = 1600;         % Number of iterations
dx = L/N;            % Cell size in X
dy = H/M;            % Cell size in Y

phi_0 = 1;                % Initial scalar field value (uniform initialization for phi)
alpha = 10;               % Sharpness parameter for inlet profile in Smith-Hutton (controls tanh() steepness)
gamma = [10; 10^3; 10^6];  % Diffusion coefficient values (used via rho/gamma ratio to sweep Peclet regimes)
```

```
34
35 % Grid generation
36 if (problem_type == 1)
37 i = linspace(0, N-1, N);
38 j = linspace(0, M-1, M);
39
40 x_C = (i + 0.5) * dx;       % Cell-center coordinates in X for problem 1
41 y_C = (j + 0.5) * dy;       % Cell-center coordinates in Y for problem 1
42 xEdge = [0, x_C, L];        % Cell-edge coordinates in X (including domain
   ↪  edges) for problem 1
43 yEdge = [0, y_C, H];        % Cell-edge coordinates in Y (including domain
   ↪  edges) for problem 1
44
45 elseif (problem_type == 2)
46 i = linspace(0, N-1, N);
47 j = linspace(0, M-1, M);
48
49 x_C = -H + (i + 0.5) * dx; % Cell-center coordinates in X for problem 2
50 y_C = (j + 0.5) * dy;       % Cell-center coordinates in Y for problem 2
51 xEdge = [-H, x_C, H];       % Cell-edge coordinates in X (including domain
   ↪  edges) for problem 2
52 yEdge = [0, y_C, H];        % Cell-edge coordinates in Y (including domain
   ↪  edges) for problem 2
53 end
54
55 % Velocity field for problem 1
56 if (problem_type == 1)
57 for i = 1:N+2
58     for j = 1:M+2
59         u(i, j) = V_in * cos(pi/4);
60         v(i, j) = V_in * sin(pi/4);
61     end
62 end
63 % Velocity field for problem 2
64 elseif (problem_type == 2)
65     for i = 1:N+2
66     for j = 1:M+2
67         u(i, j) = 2.*yEdge(j) .* (1 - (xEdge(i)).^2);
68         v(i, j) = -2.*xEdge(i) .* (1 - (yEdge(j)).^2);
69     end
70     end
71 end
72
73 % Utility arrays: preallocation prevents dynamic resizing inside loops
74 mat0_NM = zeros(N, M);          % Zero matrix for interior-coefficient storage
75 mat0_NM2 = zeros(N+2, M+2);     % Zero matrix with 1-cell ghost layer
76 mat1_NM2 = ones(N+2, M+2);      % Ones matrix with ghost layer
77 vect0_N2 = zeros(N+2, 1);       % Zero column vector (N+2)
78 vect0_M2 = zeros(1, M+2);       % Zero row vector    (M+2)
79
80 % Coefficient storage (on extended grid with ghosts)
81 phi = mat1_NM2 * phi_0;
```

```
phi_10 = mat0_NM2;
phi_1000 = mat0_NM2;
phi_1000000 = mat0_NM2;
rho = mat1_NM2 * rho_in;

[Fn, Fs, Fe, Fw] = deal(mat0_NM2);          % Convective flows
[Dn, Ds, De, Dw] = deal(mat0_NM2);          % Diffusion flows
[Pn, Ps, Pe, Pw] = deal(mat0_NM2);          % Péclet numbers
[An, As, Ae, Aw] = deal(mat0_NM2);          % Weighting coefficients
[aN, aS, aE, aW, bP] = deal(mat0_NM2);      % Coefficients
aP = mat1_NM2;                              % Central coefficient

% Phi Boundary conditions
phi(1, 1) = 0;            % Initialize corner ghost nodes to a consistent value (not used as interior unknowns)
phi(1, M+2) = 0;          % Top-left corner ghost
phi(N+2, M+2) = 0;        % Top-right corner ghost
phi(N+2, 1) = 0;          % Bottom-right corner ghost

% Coefficient calculation
for s = 1:length(gamma)                     % Loop over the three diffusion settings (rho/gamma = 10, 10^3, 10^6)
diff_coef = rho_in./gamma(s);               % Equivalent conductance factor used to build Df terms

for i = 2:N+1                               % Sweep interior CVs (indices include one-ghost frame in arrays)
    for j = 2:M+1
        Fn(i,j) = (rho(i,j) .* (v(i,j+1) + v(i,j)) ./ 2) .* dx;
        Fs(i,j) = (rho(i,j) .* (v(i,j-1) + v(i,j)) ./ 2) .* dx;
        Fe(i,j) = (rho(i,j) .* (u(i+1,j) + u(i,j)) ./ 2) .* dy;
        Fw(i,j) = (rho(i,j) .* (u(i-1,j) + u(i,j)) ./ 2) .* dy;

        Dn(i,j) = diff_coef * dx ./ abs(yEdge(j+1) - yEdge(j));
        Ds(i,j) = diff_coef * dx ./ abs(yEdge(j-1) - yEdge(j));
        De(i,j) = diff_coef * dy ./ abs(xEdge(i+1) - xEdge(i));
        Dw(i,j) = diff_coef * dy ./ abs(xEdge(i-1) - xEdge(i));

        Pn(i,j) = Fn(i,j) ./ Dn(i,j);
        Ps(i,j) = Fs(i,j) ./ Ds(i,j);
        Pe(i,j) = Fe(i,j) ./ De(i,j);
        Pw(i,j) = Fw(i,j) ./ Dw(i,j);

        if (scheme == 1)        % UDS (Upwind): A(|P|) = 1
            An(i,j) = 1;
            As(i,j) = 1;
            Ae(i,j) = 1;
            Aw(i,j) = 1;
        elseif (scheme == 2)    % CDS (Central): A(|P|) = 1 - 0.5*|P|
            An(i,j) = abs(1 - 0.5.*abs(Pn(i,j)));
            As(i,j) = abs(1 - 0.5.*abs(Ps(i,j)));
            Ae(i,j) = abs(1 - 0.5.*abs(Pe(i,j)));
```

```
130             Aw(i,j) = abs(1 - 0.5.*abs(Pw(i,j)));
131         elseif (scheme == 3)    % HDS (Hybrid): A(|P|) = max(0, 1 - 0.5*|P|)
132             An(i,j) = max(0, 1 - 0.5*abs(Pn(i,j)));
133             As(i,j) = max(0, 1 - 0.5*abs(Ps(i,j)));
134             Ae(i,j) = max(0, 1 - 0.5*abs(Pe(i,j)));
135             Aw(i,j) = max(0, 1 - 0.5*abs(Pw(i,j)));
136         elseif (scheme == 4)    % EDS: A(|P|) = |P|/(exp(|P|)-1)
137             An(i,j) = abs(Pn(i,j))./(exp(abs(Pn(i,j))) - 1);
138             As(i,j) = abs(Ps(i,j))./(exp(abs(Ps(i,j))) - 1);
139             Ae(i,j) = abs(Pe(i,j))./(exp(abs(Pe(i,j))) - 1);
140             Aw(i,j) = abs(Pw(i,j))./(exp(abs(Pw(i,j))) - 1);
141         elseif (scheme == 5)    % PLDS: A(|P|) = max(0, (1 - 0.5*|P|)^5)
142             An(i,j) = max(0, (1 - 0.5 * abs(Pn(i,j))).^5);
143             As(i,j) = max(0, (1 - 0.5 * abs(Ps(i,j))).^5);
144             Ae(i,j) = max(0, (1 - 0.5 * abs(Pe(i,j))).^5);
145             Aw(i,j) = max(0, (1 - 0.5 * abs(Pw(i,j))).^5);
146         end
147 
148         aN(i,j) = Dn(i,j) .* An(i,j) + max(-Fn(i,j), 0);  % North neighbor
            ↪ coefficient
149         aS(i,j) = Ds(i,j) .* As(i,j) + max(Fs(i,j), 0);   % South neighbor
            ↪ coefficient
150         aE(i,j) = De(i,j) .* Ae(i,j) + max(-Fe(i,j), 0);  % East  neighbor
            ↪ coefficient
151         aW(i,j) = Dw(i,j) .* Aw(i,j) + max(Fw(i,j), 0);   % West  neighbor
            ↪ coefficient
152         aP(i,j) = aE(i,j)+aW(i,j)+aN(i,j)+aS(i,j);        % Central coefficient
            ↪ (steady: sum of neighbors)
153     end
154 end
155 
156 % bP Boundary conditions
157 if (problem_type == 1)
158     bP(1,:) = 0;                        % Bottom edge Dirichlet/source
        ↪ contribution (problem 1)
159     bP(:,M+2) = 0;                      % Top edge Dirichlet/source contribution
160     bP(N+2,:) = 1;                      % Right edge Dirichlet/source
        ↪ contribution
161     bP(:,1) = 1;                        % Left  edge Dirichlet/source
        ↪ contribution
162 elseif (problem_type == 2)
163     bP([1, N+2], :) = 1 - tanh(alpha);                              % Top and
        ↪ bottom walls set to constant phi = 1 - tanh(alpha)
164     bP(:, M+2) = 1 - tanh(alpha);                                   % Right wall
        ↪ set to constant phi
165     bP(1:(N/2)+1,1) = 1+tanh(alpha*(2*xEdge(1:(N/2)+1)+1));         % Inlet
        ↪ profile at bottom (y=0) for x in (-1,0)
166     aN((N/2)+2:N+1,1) = 1;                                          % Outlet
        ↪ segment (y=0, x in (0,1)): enforce zero-normal-gradient (Neumann)
167 end
168 
169 % Solving discretized equations with Gauss-Seidel
```

```
iteration = 1;                          % Iteration counter for the linear
    ↪  solver

while iteration <= n_it                 % Fixed number of Gauss-Seidel sweeps
    for i = 2:N+1
        for j = 2:M+1
            phi(i,j) = (aE(i,j) .* phi(i+1,j) + ...
            aS(i,j) .* phi(i,j-1) + ...
            aW(i,j) .* phi(i-1,j) + ...
            aN(i,j) .* phi(i,j+1) + ...
            bP(i,j)) ./ aP(i,j);
        end

        phi(i,1) = (aE(i,1) .* phi(i+1,1) + ...
            aW(i,1) .* phi(i-1,1) + ...
            aN(i,1) .* phi(i,2) + ...
            bP(i,1)) ./ aP(i,1);

        phi(i,M+2) = (aE(i,M+2) .* phi(i+1,M+2) + ...
            aS(i,M+2) .* phi(i,M+1) + ...
            aW(i,M+2) .* phi(i-1,M+2) + ...
            bP(i,M+2)) ./ aP(i,M+2);
    end

    j = 2:M+1;
    phi(1,j) = (aE(1,j) .* phi(2,j) + ...
            aS(1,j) .* phi(1,j-1) + ...
            aN(1,j) .* phi(1,j+1) + ...
            bP(1,j)) ./ aP(1,j);
    phi(N+2,j) = (aS(N+2,j) .* phi(N+2,j-1) + ...
            aW(N+2,j) .* phi(N+1,j) + ...
            aN(N+2,j) .* phi(N+2,j+1) + ...
            bP(N+2,j)) ./ aP(N+2,j);

    iteration = iteration + 1;

    % Store converged field for this rho/gamma setting (s = 1,2,3)
    if s == 1
    phi_10 = phi(:,:);                  % Snapshot for rho/gamma = 10
    elseif s == 2
    phi_1000 = phi(:,:);                % Snapshot for rho/gamma = 10^3
    elseif s == 3
    phi_1000000 = phi(:,:);             % Snapshot for rho/gamma = 10^6
    end
end
end

% Figures generation for problem 1
[x, y] = meshgrid(yEdge, xEdge);    % Build plotting grid

if (problem_type == 1)
% Phi distribution when rho/gamma = 10
```

```
figure(1);
surf(y,x,phi_10(:,:),'EdgeColor', 'none', 'FaceColor', 'interp');

c=colorbar;
c.Label.String = '\phi';
c.Label.FontSize = 16;

axis equal
view(0, 90);  % Look from above (2D map)

xlim([0, 1]);
ylim([0, 1]);

xlabel ('X [m]');
ylabel ('Y [m]');
colormap;

% Phi distribution when rho/gamma = 10^3
figure(2);
surf(y,x,phi_1000(:,:),'EdgeColor', 'none', 'FaceColor', 'interp');

c=colorbar;
c.Label.String = '\phi';
c.Label.FontSize = 16;

axis equal
view(0, 90);

xlim([0, 1]);
ylim([0, 1]);

xlabel ('X [m]');
ylabel ('Y [m]');
colormap;

% Phi distribution when rho/gamma = 10^6
figure(3);
surf(y,x,phi_1000000(:,:),'EdgeColor', 'none', 'FaceColor', 'interp');

c=colorbar;
c.Label.String = '\phi';
c.Label.FontSize = 16;

axis equal
view(0, 90);

xlim([0, 1]);
ylim([0, 1]);

xlabel ('X [m]');
ylabel ('Y [m]');
colormap;
```

```
273
274 % Figures generation for problem 2
275 elseif (problem_type == 2)
276 % Phi distribution when rho/gamma = 10
277 figure(1);
278 surf(y,x,phi_10(:,:),'EdgeColor', 'none', 'FaceColor', 'interp');
279
280 c=colorbar;
281 c.Label.String = '\phi';
282 c.Label.FontSize = 16;
283
284 axis equal
285 view(0, 90);
286
287 xlim([-1, 1]);
288 ylim([0, 1]);
289
290 xlabel ('X [m]');
291 ylabel ('Y [m]');
292 colormap;
293
294 % Phi distribution when rho/gamma = 10^3
295 figure(2);
296 surf(y,x,phi_1000(:,:),'EdgeColor', 'none', 'FaceColor', 'interp');
297
298 c=colorbar;
299 c.Label.String = '\phi';
300 c.Label.FontSize = 16;
301
302 axis equal
303 view(0, 90);
304
305 xlim([-1, 1]);
306 ylim([0, 1]);
307
308 xlabel ('X [m]');
309 ylabel ('Y [m]');
310 colormap;
311
312 % Phi distribution when rho/gamma = 10^6
313 figure(3);
314 surf(y,x,phi_1000000(:,:),'EdgeColor', 'none', 'FaceColor', 'interp');
315
316 c=colorbar;
317 c.Label.String = '\phi';
318 c.Label.FontSize = 16;
319
320 axis equal
321 view(0, 90);
322
323 xlim([-1, 1]);
324 ylim([0, 1]);
```

```
325
326 xlabel ('X [m]');
327 ylabel ('Y [m]');
328 colormap;
329 end
```