Universitat Politectinca de Catalunya

Numerical Analysis for Convection-Diffusion Phenomenology Modelling

Computational Engineering

Student name: Altadill Llasat, Miquel

Professor name: Perez Segarra, Carlos David

Department: Centre Tecnològic de Transferència de Calor (CTTC)

Document: Final Report

Delivery Date: 25/04/2020

Contents

List of Figures

List of Tables

Acronyms

1D	One-Dimensional. 7, 10, 11
2D	Two-Dimensional. 3, 7, 11
3D	Three-Dimensional. 11
CDS	Central Difference Scheme. i, 8, 9
CV	Control Volume. 4, 6–9
EDS	Exponential Difference Scheme. i, 8, 10, 11
FVM	Finite Volume Method. 6
HDS	Hybrid Difference Scheme. i, 8, 9, 11
MOO	Matlab Object Oriented. 15, 18, 22
N-S	Navier-Stokes. i, iv, 3, 6, 24
PLDS	Powerlaw Difference Scheme. i, 8, 11
UDS	Upwind Difference Scheme. i, 8, 9

Chapter 1

Convection and Diffusion Phenomenon Mathematical Formulation

Contents

Many problems that involves the resolution of differential equations can be solved *analytically* specially those ones that involve simple geometries with simple boundary conditions. But when the problem involve complicated geometries with complex *boundary conditions* and variable properties its needed another method for solving the equations involved in the physical phenomenon. For this cases we can still obtain sufficiently accurate approximate solutions using *numerical methods*, those are based on replacing the differential equation by a set of n algebraic equations for the unknown medium property at n selected points of the medium, and the simultaneous solution of these equation results in the medium property values at those *discrete points*. We are going to call this arbitrary medium property or dependent variable ϕ to refer to it in the following sections. [3]

The numerical solution of heat transfer, fluid flow, and other related processes can begin when the laws governing these processes have been expressed in mathematical form, generally in terms of differential equations. In this section we are going to develop the mathematical formulation and complete derivation of the convection-diffusion equation as an initial step for developing the code for modelling these phenomenon [1]. The purpose in this section is to develop the familiarity with the form and meaning of these equations, geometric formulation of the control volume and the main ingredients for developing the *numerical simulation* tools for the case of study.

Accurate modelling of the interaction between convective and diffusive processes is a challenging task in numerical approximation of partial differential equations. Many different ideas and approaches have been proposed in different contexts in order to resolve the difficulties such as exponential fitting, compact differences, upwind, etc. being some examples from the fields of finite difference and finite element methods.

It is important to know that mathematical models that involve a combination of convective and diffusive processes are among the most widespread in all of science, engineering and other fields where mathematical models are involved. The convection term has an inseparable connection with the diffusion term so they need to be handled as one unit but its formulation is not as simple as the diffusion phenomena. This chapter gives us a better understanding of the Navier-Stokes equations before treating the final equation that merges the convection and diffusion phenomena. [4][1]

1.1 Navier-Stokes Equations

Before starting the formulation of the Convection-Diffusion equation its important to have clear the meaning of each one of the N-S equations. The N-S equations consists of the continuity equation, which represents the mass conservation principle (Eq.1.1); the momentum conservation equations, one for each problem dimension (Eq.1.2); and the energy conservation equation (Eq.1.3). [5]

$$\frac{d\rho}{dt} + \nabla \cdot (\rho \overrightarrow{v}) = 0 \tag{1.1}$$

$$\frac{d}{dt}(\rho \overrightarrow{v}) + \nabla \cdot (\rho \overrightarrow{v}\overrightarrow{v}) = -\nabla p + \nabla \cdot (\overrightarrow{\tau}) + \rho \overrightarrow{g} \tag{1.2}$$

$$\frac{d}{dt}(\rho(u + e_c)) + \nabla \cdot ((u + e_c)\rho \overrightarrow{v}) = -\nabla \cdot (\rho \overrightarrow{v}) + \nabla\mathring{u}(\overrightarrow{v} \cdot \overrightarrow{\tau}) - \nabla \cdot \overrightarrow{q} + \rho \overrightarrow{g} \cdot \overrightarrow{v} + G \tag{1.3}$$

Some previous assumptions have been done in the N-S equations, this hypothesis are:

- Continuity of matter
- Continuum medium assumption
- Relativity effects negligible
- Inertial reference system
- Magnetic and electromagnetic forces negligible

Table 1.1 gives an accurate physical description in order to have a better understanding of the Navier-Stokes equations before starting to work with them.

1.2 Equation Formulation

The Convection-Diffusion equation is a combination of the conservation equations of mass, linear momentum and energy also called Navier-Stokes equations. The convection is created by fluid flow, our task is to obtain a solution for the general variable ϕ in the presence of a given flow field for a given diffusion coefficient Γ . Having somehow acquired the flow field the temperature, concentration, enthalpy, or any such quantity that is represented by the general variable ϕ could be easily calculated.

1.2.1 Simplified N-S Equations

The N-S explained in the previous section can be simplified for the conditions and assumptions related to our Convection-Diffusion equation formulation.[6] Then it can be found that the simplified N-S equation that governs the flow of a Newtonian fluid in Cartesian coordinates assuming:

- Two-Dimensional model
- Laminar flow
- Incompressible flow
- Newtonian fluid
- Boussinesq hypothesis[1]

[1]Constant physical properties everywhere except in the body forces term

Equation Terms	Description
Continuity	This equation defines that the variation of mass in the control volume has to be equal to the mass flow through its faces.
$\frac{d\rho}{dt}$	Variation of mass inside the control volume in a differential time.
$\nabla \cdot (\rho \overrightarrow{v})$	Mass flow through the faces of the control volume.
Momentum	This equation shows us that the variation of linear momentum in the control volume plus the momentum flux through the CV faces has to be equal to the sum of the forces that act on the CV.
$\frac{d}{dt}(\rho \overrightarrow{v})$	Represents the variation of linear momentum in the control volume.
$\nabla \cdot (\rho \overrightarrow{v} \overrightarrow{v})$	Represents the momentum flux through the faces of its control volume.
∇p	Pressure gradient acting like an axial force on the faces of the CV.
$\nabla \cdot (\overrightarrow{\tau})$	Total stress tensor. This force acts axially and tangentially on the faces of the control volume. Its value depends on the type of fluid (Newtonian, non-Newtonian...).
$\rho \overrightarrow{g}$	Volumetric force. This force may be a gravitational, electrical, magnetic or electromagnetic.
Energy	Defines that the variation of internal energy and kinetic energy in a control volume plus the flow of their variables must be equal to the work done on the control volume plus the incoming heat flow through the faces of the CV plus the energy of the sources in the control volume.
$\frac{d}{dt}(\rho(u+e_c))$	Represents the variation of the internal and kinetic energy in the CV.
$\nabla \cdot ((u+e_c)\rho \overrightarrow{v})$	Represents the energy flow of these variables through the faces of its volume.
$-\nabla \cdot (\rho \overrightarrow{v})$	Work done by superficial forces like pressure and stress.
$\nabla \cdot \overrightarrow{q}$	Work done by superficial forces like pressure and stress.
$\nabla \mathring{u}(\overrightarrow{v} \cdot \overrightarrow{\tau})$	Incoming heat flow through the faces of the control volume.
$\nabla \cdot \overrightarrow{q}$	Represents the work done by the volumetric forces, in this case there is only the gravitational force work.
$\rho \overrightarrow{g} \cdot \overrightarrow{v}$	Work done by the internal forces.

Table 1.1: Navier-Stokes Equation description

- Negligible viscous dissipation
- Negligible compression or expansion work
- Non-participating medium in radiation
- Mono-component and mono-phase fluid

The use of constant properties of thermal conductivity, density... implies that it will not be possible to solve problems with a huge range in temperatures because all of these properties depend on it.

Simplifying equations from Eq.1.1 to 1.3:

$$\frac{du}{dx} + \frac{dv}{dy} = 0 \tag{1.4}$$

$$\rho\frac{du}{dx} + \rho u\frac{du}{dx} + \rho v\frac{du}{dy} = -\frac{dp}{dx} + \mu\Big(\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2}\Big) \tag{1.5}$$

$$\rho\frac{du}{dx} + \rho u\frac{dv}{dx} + \rho v\frac{dv}{dy} = -\frac{dp}{dy} + \mu\Big(\frac{d^2v}{dx^2} + \frac{d^2v}{dy^2}\Big) + \rho g\beta(T - T_\infty) \tag{1.6}$$

$$\rho\frac{dT}{dt} + \rho u\frac{dT}{dx} + \rho v\frac{dT}{dy} = \frac{k}{c_p}\Big(\frac{d^2T}{dx^2} + \frac{d^2T}{dy^2}\Big) + \frac{G}{c_p} \tag{1.7}$$

Noting that in this equation there are four unknown values: pressure, temperature and the two components of velocity u and v. Furthermore, a boundary condition and an initial condition are required to solve the problem.

Analysing the system of equations closely it can be observed a strong coupling between them:

- Pressure - Velocity: for the previous established conditions, there is no specific pressure equation, but the pressure distribution allows the velocity field to satisfy the mass conservation equation.
- Temperature - Velocity: there is only a coupling characterisation for natural convection, mixed connection or when the physical properties depend on the temperature. In forced convection and constant physical properties, the velocity field does not depend on temperature field.

1.2.2 Convection-Diffusion Equation

Acknowledging the coupling from the partial differential equations and applying the corresponding assumptions all equations from Eq.(1.4 - 1.7) can be summarised into the convection-diffusion equation:

$$\frac{d(\rho\phi)}{dt} + \nabla(\rho\overrightarrow{v}\phi) = \nabla(\Gamma\nabla\phi) + G \tag{1.8}$$

in Cartesian coordinates, incompressible flow and constant physical properties the equation can also be written as

$$\rho\frac{d\phi}{dt} + \rho u\frac{d\phi}{dx} + \rho v\frac{d\phi}{dy} = \frac{k}{c_p}\Big(\frac{d^2\phi}{dx^2} + \frac{d^2\phi}{dy^2}\Big) + G \tag{1.9}$$

In the previous equation, the first term is the accumulation of ϕ which tells how ϕ change along time. The second and the third term are the net convective flow in the control volume, which gives information about the spatial transport of ϕ. The sum of these has to be equal to the net diffusive flow, which represents the transport of ϕ due to the concentration of gradients, plus the generation of ϕ per unit volume (G).

Looking at Eq.1.8 the *diffusion flux* due to the gradient of the general variable ϕ is $-\Gamma(d\phi/dx)$ where ϕ could represent chemical-species diffusion, heat flux, viscous stress, etc. According to Eq.1.8 a table with the parameters of ϕ, τ and G could be written in order to reproduce the governing equations (Eq. 1.4 - Eq. 1.7). Te values for the pressure, temperature and density field are obtained by introducing parameters of Table 1.2 into the N-S equations once the field ϕ is solved.

Equation	ϕ	τ	G
Continuity	1	0	0
Momentum in X direction	u	μ	$-dp/dx$
Momentum in Y direction	v	μ	$-dp/dy + \rho g \beta (T - T_\infty)$
Energy (constant c_p)	T	k/c_p	ϕ/c_p

Table 1.2: Parameters to obtain N-S equations convection-diffusion equation

1.3 Equation Discretization

In this section the implicit finite-volume discretization (FVM) of the convection-diffusion equation is shown. First of all Eq.1.9 has to be integrated into a rectangular CV, Fig.1.1 shows the geometric parameters for the integration:

$$\frac{(\rho\phi)^1_P - (\rho\phi)^0_P}{\Delta t}\Delta x \Delta y + \Big[(\rho u \phi)^1_e - (\rho u \phi)^1_w\Big]\Delta y + \Big[(\rho v \phi)^1_n - (\rho v \phi)^1_s\Big]\Delta x =$$

$$= \Bigg[\left(\Gamma\frac{d\phi}{dx}\right)^1_e - \left(\Gamma\frac{d\phi}{dx}\right)^1_w\Bigg]\Delta y + \Bigg[\left(\Gamma\frac{d\phi}{dy}\right)^1_n - \left(\Gamma\frac{d\phi}{dy}\right)^1_s\Bigg]\Delta x + G^1_P \Delta x \Delta y \quad (1.10)$$

Note that superindex "1" is used for the value of property ϕ at time $t = t + \Delta t$ and "0" at the previous time step value, for an easier formulation it can be stated that $\phi^1 = \phi$. Assuming $\Delta x \Delta y$ as the CV volume V_P and separately as their surfaces S_e , S_w, S_n and S_s

$$\frac{(\rho\phi)_P - (\rho\phi)^0_P}{\Delta t}V_P + \Big[(\rho u \phi)_e S_e - (\rho u \phi)_w S_w\Big] + \Big[(\rho v \phi)^1_n S_n - (\rho v \phi)_s S_s\Big] =$$

$$= \Bigg[\left(\Gamma\frac{d\phi}{dx}\right)_e S_e - \left(\Gamma\frac{d\phi}{dx}\right)_w S_w\Bigg] + \Bigg[\left(\Gamma\frac{d\phi}{dy}\right)_n S_n - \left(\Gamma\frac{d\phi}{dy}\right)_s S_s\Bigg] + G_P V_P \quad (1.11)$$

This formulation can be simplified using the total flux term, defined by:

$$J_x = \rho v \phi - \Gamma \frac{d\phi}{dx} \quad (1.12a)$$

$$J_y = \rho v \phi - \Gamma \frac{d\phi}{dy} \quad (1.12b)$$

Equation 1.8 can be expressed with the flux term J as:

$$\frac{d(\rho\phi)}{dt} + \frac{dJ_x}{dx} + \frac{dJ_y}{dy} = G \quad (1.13)$$

Integrating the previous equation into a rectangular CV and assuming an implicit scheme for the temporal integration, Eq. 1.13 yields to:

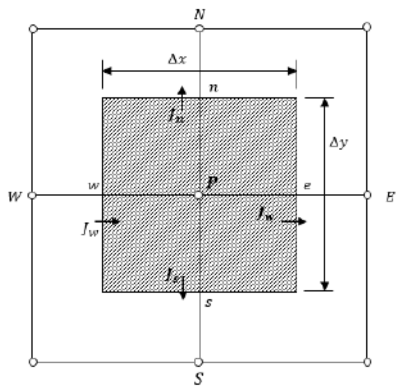


Figure 1.1: Two-Dimensional CV with flux vectors (J)

$$\frac{(\rho\phi)_P - (\rho\phi)_P^0}{\Delta t} V_P + J_e - J_w + J_n - J_s = (G_c + G_P \phi_P) V_P \tag{1.14}$$

The quantities J_e , J_w , J_s and J_n are the integrated total fluxes over the control-volume faces; that is, J_e stands for $\int J_x dy$ over the interface e and so on. The source term has been linearized as it could be seen in the last term of Eq. 1.14.

Noting that the flow field has to satisfy the continuity equation (Eq. 1.4) in order to assume convergence:

$$\frac{d}{dx_j}(\rho u_j) = 0 \tag{1.15}$$

Integrating over a rectangular finite volume:

$$\frac{\rho_P - \rho_P^0}{\Delta t} V_P + F_e - F_w + F_n - F_s = 0 \tag{1.16}$$

where F_e, F_n, F_s and F_w are the mass flow rates through the faces of the Control Volume.

$$F_e = (\rho u)_e S_e \tag{1.17a}$$
$$F_w = (\rho u)_w S_w \tag{1.17b}$$
$$F_n = (\rho v)_n S_n \tag{1.17c}$$
$$F_s = (\rho v)_s S_s \tag{1.17d}$$

Multiplying Eq.1.16 by ϕ_P and subtracting it from Eq.1.14:

$$(\phi_P - \phi_P^0)\frac{\rho_P^0}{\Delta t} \cdot V_P + (J_e - F_e\phi_P) - (J_w - F_w\phi_P) + \\ + (J_n - F_n\phi_P) - (J_s - F_s\phi_P) = (S_c + S_P\phi_P) \tag{1.18}$$

The assumption of uniformity over a control-volume face enables to employ One-Dimensional practices from Ref.[1] for the Two-Dimensional situation.

1.4 Numerical Schemes

Numerical schemes in convection-diffusion problems evaluate the convective and diffusive terms at the CV faces while the dependent variable ϕ is evaluated at the centre. Convective flux on any face is given by the arithmetic mean between central node and its neighbours:

$$\left(\frac{d\phi}{dx}\right)_w = \frac{\phi_W - \phi_P}{\delta x_w} \tag{1.19a}$$

$$\left(\frac{d\phi}{dx}\right)_e = \frac{\phi_E - \phi_P}{\delta x_w} \tag{1.19b}$$

$$\left(\frac{d\phi}{dy}\right)_n = \frac{\phi_N - \phi_P}{\delta y_n} \tag{1.19c}$$

$$\left(\frac{d\phi}{dy}\right)_s = \frac{\phi_S - \phi_P}{\delta y_s} \tag{1.19d}$$

Convective and diffusive terms need to be calculated using numerical schemes that evaluate values of ϕ at the nodal points. There are two types of schemes: low order numerical schemes; and high order schemes. The *order* of a numerical scheme is the number of neighbour nodes that are involved to evaluate the dependent variable at the cell face.

For the scope of this project it is only going to be explained low order numerical schemes such as: CDS, UDS, HDS, EDS, PLDS. These numerical schemes evaluate the variable using nearest nodes (east (E), west(W), nort(N) and south (S)) with a scheme order of one or two. Figure 1.2 shows the values of the variable ϕ given by the different schemes for various values of the Peclet Number (Pe).

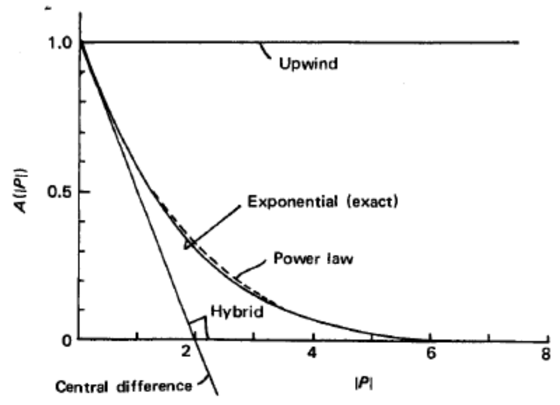


Figure 1.2: The function $A(|Pe|)$ for various low order schemes [1]

Scheme	Formula for $A(\lvert Pe\rvert)$
Central difference	$1 - 0.5\lvert Pe\rvert$
Upwind	1
Hybrid	$[\![0, 1 - 0.5\lvert Pe\rvert\ ]\!]$
Power Law	$[\![0, (1 - 0.1\lvert Pe\rvert)^5\ ]\!]$
Exponential	$\lvert Pe\rvert/(e^{\lvert Pe\rvert} - 1)$

Table 1.3: The function $A(|Pe|)$ for different schemes [1]

1.4.1 Central Difference Scheme (CDS)

It is a second order scheme where the variable at the cell face is calculated as the arithmetic mean of the variable at the neighbour nodes of the face. For the east face of the CV:

$$\phi_e = \frac{1}{2}(\phi_P + \phi_E) \tag{1.20}$$

Using a general notation to reefer to the control volume face or center:

$$\phi_{if} = \frac{1}{2}(\phi_P + \phi_{ib}) \tag{1.21}$$

Looking at Fig.1.2 it is seen that all schemes except the CDS give physically realistic solutions because it can produce values that lie outside the $[0-1]$ range established by the Scarborough criterion, see Appendix A. The formula of $A(|Pe|)$ for this scheme is found in Table 1.3:

$$A(|Pe_{if}|) = 1 - 0.5|Pe_{if}| \tag{1.22}$$

1.4.2 Upwind Difference Scheme (UDS)

It is a first order scheme where the value of ϕ at the cell face is equal to the value of ϕ at the grid point on the upwind side of the face.

$$\phi_e = \phi_P \quad if \quad F_e > 0 \tag{1.23a}$$

$$\phi_e = \phi_E \quad if \quad F_e < 0 \tag{1.23b}$$

What that means is that if u is positive, the value of ϕ at the face will be the value of ϕ at the left grid point. However, if u is negative, the value of ϕ at the face cell will be the value of ϕ at the right grid point. It will be the same reasoning for v. It is defined a new operator for this criterion as $[\![A, B]\!]$. Thus,

$$F_e\phi_e = \phi_P[\![F_e, 0]\!] - \phi_E[\![-F_e, 0]\!] \tag{1.24}$$

This scheme solves the problem that the CDS has because all the coefficients in the equations are always positive or null. That means that the Scarborough criterion is always satisfied. The formula of $A(|Pe|)$ for this scheme is found in Table 1.3:

$$A(|Pe_{if}|) = 1 \tag{1.25}$$

1.4.3 Hybrid Difference Scheme (HDS)

It is a combination of a central difference scheme and an upwind difference scheme as it exploits the favourable properties of both of these schemes. This scheme uses CDS for low velocities and UDS for high velocities, it consists on approximating the value of the dimensionless form of a_E (Eq. 1.26) to three linear zones. Fig. 1.3 shows this approximation.

$$\frac{a_E}{D_e} = \frac{Pe_e}{exp(Pe_e) - 1} \tag{1.26}$$

For positives values of Pe_e the grid point E is the *downstream* neighbor and its influence is seen to decrease as Pe_e increases. When Pe_e is negative the point E is the *usptream* neighbour and has a large influence. The tree straight lines represent the three limiting cases, they can be seen to form an envelope of, and represent a reasonable approximation to, the exact curve. Then,

For $Pe_e < -2$,

$$\frac{a_E}{D_e} = -Pe_e \tag{1.27}$$

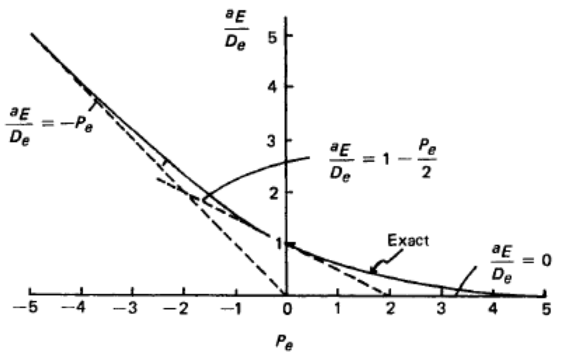


Figure 1.3: Variation of coefficient a_E with Peclet number [1]

For $-2 \leq Pe_e \leq -2$,

$$\frac{a_E}{D_e} = 1 - \frac{Pe_e}{2} \tag{1.28}$$

For $Pe_e > 2$,

$$\frac{a_E}{D_e} = 0 \tag{1.29}$$

This expressions can be compacted into the following form[2]:

$$a_E = D_e \left[\!\!\left[-Pe_e, 1 - \frac{Pe_e}{2}, 0 \right]\!\!\right] \tag{1.30a}$$

$$a_E = \left[\!\!\left[-F_e, D_e - \frac{F_e}{2}, 0 \right]\!\!\right] \tag{1.30b}$$

Noting that it is identical with the central-difference scheme for the Peclet number range $-2 \leq Pe_e \leq 2$, and outside this range it reduces to the upwind scheme in which the diffusion has been set equal to zero. For that reason and from Table 1.3 the function of $A(|Pe|)$ is

$$A(|Pe_i f|) = [\![0, 1 - 0.5|Pe_i f|]\!] \tag{1.31}$$

1.4.4 Exponential Difference Scheme (EDS)

It is a second order scheme where the evaluation of the variables at the cell faces come from the exact solution of the Eq. 1.9 for the steady One-Dimensional problem without source term [6]. From [1] it is known that exact solution is:

$$\frac{\phi - \phi_0}{\phi_L - \phi_0} = \frac{exp(Pe \cdot x/L) - 1}{exp(Pe) - 1} \tag{1.32}$$

Where ϕ_0 is the value of ϕ at the left boundary ($x = 0$); ϕ_L the value of ϕ at the right boundary ($x = L$); x is the position of the left interface node; Pe is the Peclet number; and L is the distance of the domain ($0 \leq x \leq L$). Remember that the Peclet number is defined by:

$$Pe \equiv \frac{\rho u L}{\Gamma} \tag{1.33}$$

From Eq.1.32 it can be seen that P is the ratio of strengths of convection and diffusion, which gives us a better understanding about the meaning of this number inside the case of study. The nature of Eq.1.32 has to be understood from Fig. 1.4 where the variation of $\phi \sim x$ for different values of the Peclet number is shown.

[2]This special symbol $[\![\quad]\!]$stands for the largest of the quantities contained within it.

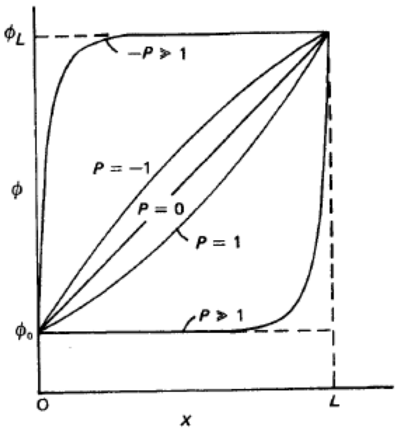


Figure 1.4: Exact solution for the one-dimensional convection-diffusion problem [1]

This scheme gives an exact solution for 1D for any Peclet number, although it is not exact for the 2D and 3D situations. Another disadvantage would be the extra time it takes to compute the solution with exponential functions. The formula of $A(|Pe|)$ for this scheme is found in Table1.3:

$$A(|Pe_{if}|) = |Pe_i f|/(e^{|Pe_i f|} - 1) \tag{1.34}$$

1.4.5 Power-law Difference Scheme (PLDS)

Taking the HDS it seems a little premature to set the diffusion effects equal to zero as soon as the Peclet number exceeds 2, a better approximation to the exact curve is given by the power-law scheme. It is a second order scheme where the variable at the cell face is calculated with an approximation of the EDS by a polynomial of fifth degree.

The compact form for the coefficient a_E obtained from [1]:

$$a_E = D_e \left[\!\left[0, \left(1 - \frac{0.1|F_e|}{D_e}\right)^5 \right]\!\right] + [\![0, -F_e]\!] \tag{1.35}$$

The formula of $A(|Pe|)$ for this scheme is found in Table1.3:

$$A(|Pe_{if}|) = [\![0, (1 - 0.1|Pe_i f|)^5]\!] \tag{1.36}$$

1.5 Final Discretization Equation

From Eq. 1.18 and according to [1, 6] the Two-Dimensional final discretization equation can now be rewritten as

$$a_P\phi_P = a_E\phi_E + a_S\phi_S + a_W\phi_W + a_N\phi_N + b \tag{1.37}$$

Where the coefficients a_i can be evaluated as:

$$a_E = D_e \cdot A(|Pe_e|) + [\![-F_e, 0]\!] \tag{1.38a}$$

$$a_W = D_w \cdot A(|Pe_w|) + [\![F_w, 0]\!] \tag{1.38b}$$

$$a_N = D_n \cdot A(|Pe_n|) + [\![-F_n, 0]\!] \tag{1.38c}$$

$$a_S = D_s \cdot A(|Pe_s|) + [\![F_s, 0]\!] \tag{1.38d}$$

$$a_P^0 = \frac{\phi_P^0 V_P}{\Delta t} \tag{1.38e}$$

$$b = G_C V_P + a_P^0 \phi_P^0 \tag{1.38f}$$

$$a_P = a_E + a_S + a_W + a_N + a_P^0 - G_P V_P \tag{1.38g}$$

Where the flow rates through the faces (F_{if}) are:

$$F_e = (\rho u)_e S_e \tag{1.39a}$$

$$F_w = (\rho u)_w S_w \tag{1.39b}$$

$$F_n = (\rho v)_n S_n \tag{1.39c}$$

$$F_s = (\rho v)_s S_s \tag{1.39d}$$

The corresponding conductances are defined by

$$D_e = \frac{\Gamma_e S_e}{(\delta x)_e} \tag{1.40a}$$

$$D_w = \frac{\Gamma_w S_w}{(\delta x)_w} \tag{1.40b}$$

$$D_n = \frac{\Gamma_n S_n}{(\delta x)_n} \tag{1.40c}$$

$$D_s = \frac{\Gamma_s S_s}{(\delta x)_s} \tag{1.40d}$$

and the Peclet numbers by

$$P_e = \frac{F_e}{D_e} \quad P_n = \frac{F_n}{D_n} \quad P_s = \frac{F_s}{D_s} \quad P_w = \frac{F_w}{D_w} \tag{1.41}$$

Noting that all the formulation has been done in order that the value of $A(|Pe_i f|)$ depends on the numerical scheme used. This value can be found in Table 1.3.

Chapter 2

Numerical Analysis Application Cases

Contents

In this chapter it is intended to apply the convection-diffusion discretized equation for solving simple case of studies in order to get in touch with the phenomena modelling. This step is essential for further studies because it gives the student a better comprehension on the physics involved and how the flow field behaves when the convection and diffusion effects are present. The numerical analysis has been done using a self developed non-commercial code. For the coding it has been used Matlab for its simplicity and performance, making this programming environment the suitable one for this project.

The chapter is divided into three simple application cases needed to validate the results of the code. The two first cases would be essential for the code validation since it exists an analytical solution for each one. The comparison in between this results and the ones obtained using the numerical simulation tool would give the model validation, needed for further studies or real case applications. The last presented case is the Smith-Hutton problem, one of the most common practices for the convection-diffusion phenomenology study.

2.1 One-Dimensional Flow (X)

In this section it is applied the concepts explained in Chapter 1 applied to a practical case where the convection-diffusion phenomena is involved. This problem consists in the study of a one-dimensional flow with a one-dimensional variation of the variable solved in the same direction of the flow. In order to verify the symmetry of the solution the results over both spatial dimensions will be presented. [7]

2.1.1 Problem Definition

For the situation presented in Fig.2.1 it is known that an analytical solution could be easily obtained. The solution is an exponential function for an arbitrary value of the velocity field.

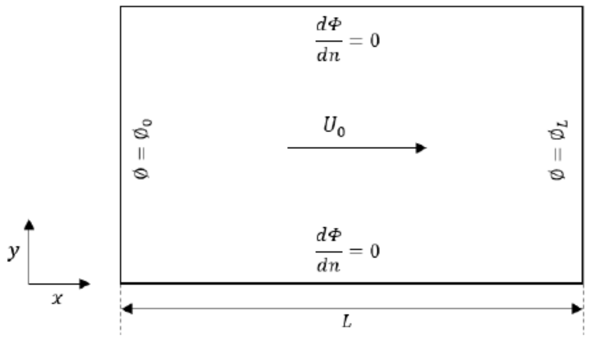


Figure 2.1: One-Dimensional flow with a uni-dimensional variation of the variable solved in the same direction of the flow.

From the previous figure it is shown that the flow field follows the "x" direction along a "L" longitude: $0 \leq x \leq L$. For the case of study the velocity field is set as:

$$u(x, y) = U_0 \tag{2.1a}$$

$$v(x, y) = 0 \tag{2.1b}$$

2.1.2 Boundary Conditions

For this case of study it is seen for the right and left boundary the use of a Dirichlet boundary condition which gives a constant value of ϕ for each side. For the upper and bottom boundary the use of Neumann Boundary Condition is seen which describes the flux over both surfaces , in this case means that there is no normal convective flow over this surface.

- Left Boundary: $\phi_0 = 273.15$
- Right Boundary: $\phi_L = 373.15$
- Bottom Boundary: $d\phi/dy = 0$
- Upper Boundary: $d\phi/dy = 0$
- Initial value field ϕ: $\phi = 333.15$

After some simulations the velocity field has been set to

$$U_0 = 0.005m/s \tag{2.2}$$

for developing the most critical computing case, where the diffusion phenomena predominates. A greater velocity would suppose a greater influence of the convective term to the results meaning that the field would adopt its final form in less time.

2.1.3 Discretization

From Section 1.2 the convection-diffusion equation (Eq.1.9) could be rewritten considering the source term value as zero:

$$\rho\frac{d\phi}{dt}+\rho u\frac{d\phi}{dx}+\rho v\frac{d\phi}{dy}=\frac{k}{c_p}\Big(\frac{d^2\phi}{dx^2}+\frac{d^2\phi}{dy^2}\Big) \tag{2.3}$$

The implicit coefficient form of the previous equation is known from Section 1.5

$$a_P\phi_P=a_E\phi_E+a_S\phi_S+a_W\phi_W+a_N\phi_N+b \tag{2.4}$$

Even though the problem asks for the steady state condition, the terms that contain time dependency were taken into account because more conclusions could be obtained about time evolution of the phenomena and its computational cost. The spatial discretization and control volume geometry chosen for the case of study are shown in Table 2.1.

Geometrical Property	L	H	N_x	N_y	Δx	Δy
Value	0.1	0.1	200	2	0.0005	0.05

Table 2.1: Domain spatial discretization for one-dimensional flow (X)

2.1.4 Program

This problem has been solved using a MOO program born from the case of study presented in Section 2.3. In order to provide a generic solution that provides numerical solutions to the one-dimensional case without having to do almost any modification[1] The core of the code is the Main function, which contains all the necessary methods and input data. Inside the main it is found four principal functions:

- Uniform Mesh: in charge of the domain discretization and compute of the velocity field needed for each problem.
- Coefficient Compute: in charge of computing the needed coefficients for each case, the ones that are dependent on the field ϕ and the non-dependent.
- Solver: implicit formulation and application of the Gauss-Seidel method.
- Solver Shell: that function returns us to the final results for the ϕ field. It contains the Solver.

The necessary data for starting the computations is modified from the ”inputData” file. It contains the points that define our domain (P_1, P_2), the requested solutions points, the sizes of our mesh (N_x, N_y), the initial field ϕ value and its boundary conditions, the physical properties needed for solving the coefficients and finally the solver parameters.

Figure 2.9 shows the algorithm flowchart in order to represent that problem reached the steady state. The first program iterated until the steady state was reached without taking into account what happened with each time step.

2.1.5 Results

As it is expected Fig.2.2 shows the results are one-dimensional along the X axis. Once it is seen the physical coherence of the results it is needed to compare the numerical solution with the analytical one. The analytic result for this case of study is:

[1]This code can be downloaded by clicking ”here”.

$$\frac{\phi - \phi_0}{\phi_L - \phi_0} = \frac{e^{\frac{Pe \cdot x}{L}} - 1}{e^{Pe} - 1} \tag{2.5}$$

Once having the analytic solution it is needed to define the fluid of study and its properties in order to compute the Peclet number which is:

$$Pe = \frac{\rho v L}{\Gamma} = \frac{\rho v L}{k/c_p} \tag{2.6}$$

For this and the following cases of study the fluid study has been set as air. Table 2.2 shows the physical parameters of this fluid for a defined pressure and temperature for computing Peclet number.

Physical Property	$T(K)$	$P(Pa)$	$\rho(kg/m^3)$	$k(W/mK)$	$c_p(J/kgK)$
Value	333.15	101325	1.0595	0.2916	1007.04

Table 2.2: Domain spatial discretization for one-dimensional flow (Y)

As is has been said, the following figure shows the evolution of the property ϕ along the x direction. This solution shows the steady state for this case of study.

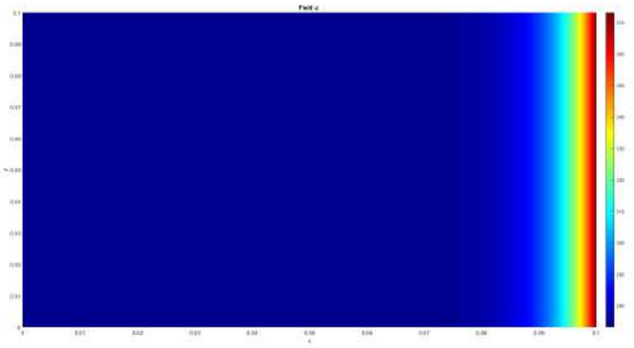


Figure 2.2: Field ϕ for one-dimensional flow along X-axis

Once known the distribution of ϕ along the X-Axis it is time to compare these numerical results with the analytic ones. The following plot shows the evolution of both results in order to study the nature of the results and its validity.

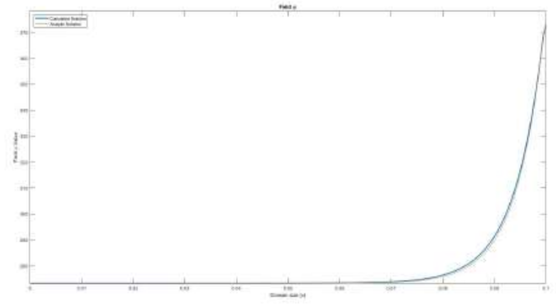


Figure 2.3: Numerical Solution vs Analytic for one-dimensional flow along X-axis

In this figure it is seen that there is a little deviation of the numerical result against the analytic one. For more representations of the results see Appendix B.

2.1.6 Conclusions

First of all it is necessary to comment on the nature of the results, since the field was set to a mid value between it was expected a more or less symmetrical solution but due to the fact that the convective value (F) is bigger than the diffusive (D) [2] the upstream value covers almost all the domain.

The distance between the ϕ value remains constant depending on the domain length, the initial velocity and the air properties. Changing these values it has reached a smoother curve which would indicate a decrease of the influence of the convective term. It is also possible to find cases where the computing time increases exponentially, which are the ones where the diffusion phenomena predominate.

Finally, it is necessary to comment that the error between the numerical and analytical result could be due to a little asymmetry inside the code or maybe due to an error source that magnifies this error each iteration.

2.2 One-Dimensional Flow (Y)

In this section the concepts explained in Chapter 1 applied to a practical case where the convection-diffusion phenomena is involved. This problem consists in the study of a one-dimensional flow with a uni dimensional variation of the variable solved in the perpendicular direction of the flow. In order to verify the symmetry of the solution the results over both spatial dimensions will be presented. [7]

2.2.1 Problem Definition

For the situation presented in Fig.2.4 it is known that an analytical solution could be easily obtained. The solution is an exponential function for an arbitrary value of the velocity field.

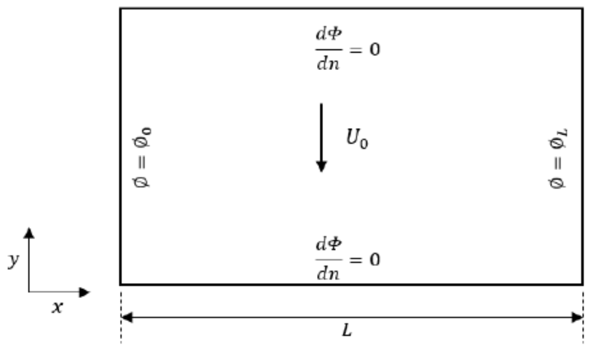


Figure 2.4: One-Dimensional flow with a uni dimensional variation of the variable solved in the perpendicular direction of the flow.

From the previous figure it is seen that the flow field follows the "y" direction. For the case of study the velocity field is set as:

$$u(x,y) = 0 \tag{2.7a}$$

$$v(x,y) = U_0 \tag{2.7b}$$

2.2.2 Boundary Conditions

For this case of study it is seen for the right and left boundary the use of a Dirichlet boundary condition which gives a constant value of ϕ for each side. For the upper and bottom boundary it is seen the use of Neumann boundary condition which describes the flux over both surfaces is seen, in this case it means

[2] "Upstream" is the direction towards the fluid source, or where the fluid is coming from.[8]

that there is no normal convective flow over this surface. See Section **??** for more information about this boundary conditions.

- Left Boundary: $\phi_0 = 273.15$
- Right Boundary: $\phi_L = 373.15$
- Bottom Boundary: $d\phi/dy = 0$
- Upper Boundary: $d\phi/dy = 0$
- Initial value field ϕ: $\phi = 333.15$

After some simulations the velocity field has been set to

$$U_0 = 0.005m/s \tag{2.8}$$

for developing the most critical computing case, where the diffusion phenomena predominates. A greater velocity would suppose a greater influence of the convective term to the results meaning that the field would adopt its final form in less time.

2.2.3 Discretization

From Section 1.2 the convection-diffusion equation (Eq.1.9) could be rewritten considering the source term value as zero:

$$\rho\frac{d\phi}{dt} + \rho u\frac{d\phi}{dx} + \rho v\frac{d\phi}{dy} = \frac{k}{c_p}\Big(\frac{d^2\phi}{dx^2} + \frac{d^2\phi}{dy^2}\Big) \tag{2.9}$$

The implicit coefficient form of the previous equation is known from Section 1.5

$$a_P\phi_P = a_E\phi_E + a_S\phi_S + a_W\phi_W + a_N\phi_N + b \tag{2.10}$$

Even though our problem asks for the steady state condition, the terms that contain time dependency were taken into account because more conclusions about time evolution of the phenomena and its computational cost could be obtained. The spatial discretization and control volume geometry chosen for our case of study are shown in Table 2.3.

Geometrical Property	L	H	N_x	N_y	Δx	Δy
Value	0.1	0.1	200	2	0.0005	0.05

Table 2.3: Domain spatial discretization for one-dimensional flow (Y)

2.2.4 Program

This problem has been solved using a MOO program born from the case of study presented in Section 2.3. In order to provide a generic solution that provides numerical solutions to the one-dimensional case without having to do almost any modification[3] The core of the code is the Main function, which contains all the needed methods and input data. Inside the main it is found four principal functions:

- Uniform Mesh: in charge of the domain discretization and compute of the velocity field needed for each problem. The velocity field is the only variation from the previous case.

[3]This code can be downloaded with a document that describes the case of study by clicking "here".

- Coefficient Compute: in charge of computing the needed coefficients for each case, the ones that are dependent on the field ϕ and the non-dependent.
- Solver: implicit formulation and application to the Gauss-Seidel method.
- Solver Shell: that function returns us the final results for the ϕ field. It contains the Solver (G-S).

The needed data for starting the computations is modified from the ”inputData” file. It contains the points that define the domain (P_1, P_2), the requested solutions points, the sizes of the mesh (N_x, N_y), the initial field ϕ value and its boundary conditions, the physical properties needed for solving the coefficients and finally the solver parameters.

Figure 2.9 shows the algorithm flowchart in order to represent that the problem reached the steady state. The first program iterated until the steady state was reached without taking into account what happened at each time step.

2.2.5 Results

As it is expected Fig.2.5 shows the results are one-dimensional along the X axis. Once the physical coherence of the results are seen it is necessary to compare the numerical solution with the analytical one. The analytic result for this case of study is:

$$\frac{\phi - \phi_0}{\phi_L - \phi_0} = \frac{e^{\frac{Pe \cdot x}{L}} - 1}{e^{Pe} - 1} \tag{2.11}$$

Once having the analytic solution it is necessary to define the fluid of study and its properties in order to compute the Peclet number which is:

$$Pe = \frac{\rho v L}{\Gamma} = \frac{\rho v L}{k/c_p} \tag{2.12}$$

Table 2.2 shows the physical parameters of this fluid for a defined pressure and temperature for computing Peclet number.As it has been said, the following figure shows the evolution of the property ϕ along the x direction. This solution shows the steady state for this case of study.

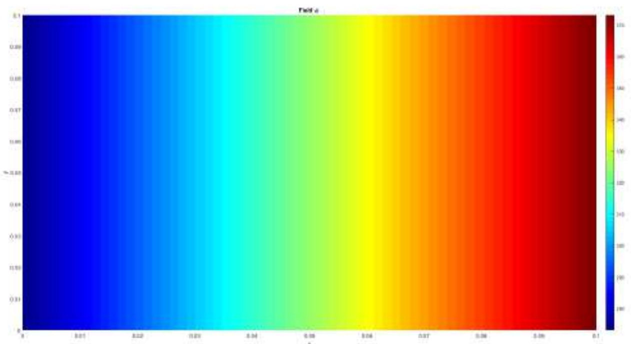


Figure 2.5: Field ϕ for one-dimensional flow along Y-axis

Once known the distribution of ϕ along the X-Axis it is time to compare these numerical results with the analytic ones. The following plot shows the evolution of both results in order to study the nature of the results and its validity.

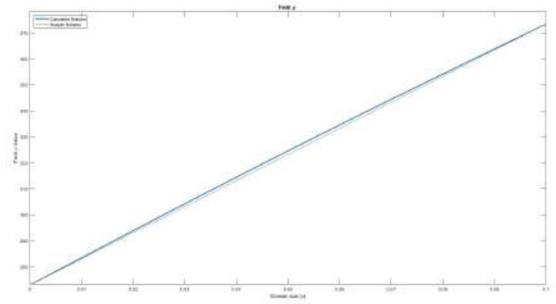


Figure 2.6: Numerical Solution vs Analytic for one-dimensional flow along Y-axis

In this figure it is seen that there is a little deviation of the numerical result against the analytic one. For more representations of the results see Appendix C.

2.2.6 Conclusions

It is seen that the result obtained matches with the expected solution. Since the convective value along the x direction is equal to zero, due to the velocity field, so that the only term that affects to the coefficients and the problem result is the diffusive one. This physically means that this solution is equivalent to the solution obtained in a conduction problem.

2.3 Convection-Diffusion Solenoidal Flow Problem

In this section the concepts explained in Chapter 1 applied to a practical case where the convection-diffusion phenomena is involved. This problem can also be called Smith-Hutton Problem. This is a recirculating flow problem which involves streamline curvature studied by Smith and Hutton. In their study they concluded that in a high-convection regime modelling ”remains the art of compromise between diffusive and oscillatory errors”.[7]

2.3.1 Problem Definition

This problem is based on a two-dimensional test problem devised by Smith and Hutton which concerns steady-state convection and diffusion of a scalar field ϕ in a prescribed velocity field $\overrightarrow{v}$ with a known constant diffusivity D. The objective is to find the field ϕ for a given value of the relation ϕ/Γ. Figure 2.7 shows a visual scheme of the problem.

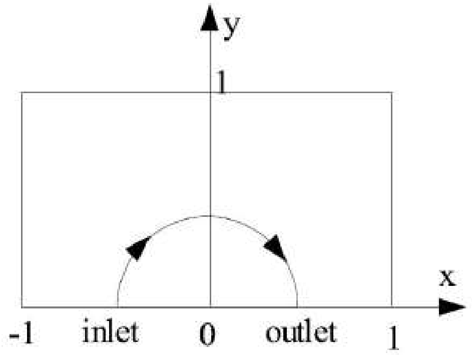


Figure 2.7: Smith-Hutton problem

As it is seeen in Fig.2.7 the flow domain considered is a rectangle: $-1 \leq x \leq 1$, $0 \leq y \leq 1$. And the velocity field $\overrightarrow{v}$ is given by

$$u(x,y) = 2y(1-x^2) \tag{2.13a}$$

$$v(x,y) = -2x(1-y^2) \tag{2.13b}$$

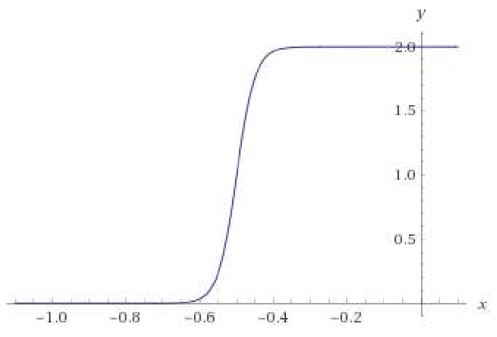
2.3.2 Boundary Conditions

Table 2.4 gives us the boundary conditions for the parameter ϕ in our case of study.

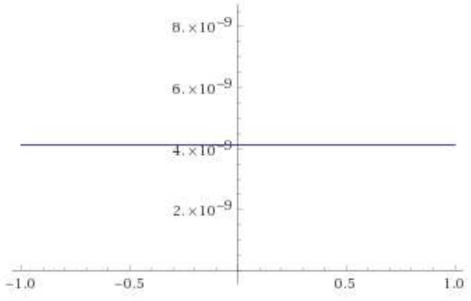
Field ϕ	$x[m]$	$y[m]$
$\phi = 1 + tanh[(2x+1)\alpha]$	$-1 < x < 0$	$y = 0$
	$x = -1$	$0 < y < 1$
$\phi = 1 + tanh(\alpha)$	$-1 < x < 1$	$y = 1$
	$x = 1$	$0 < y < 1$
$d\phi/dy = 0$	$0 < x < 1$	$y = 0$

Table 2.4: Boundary conditions for Smith-Hutton Problem ($\alpha = 10$)

As it is seen in the plots shown in Fig. 2.8 the hyperbolic tangent function[4] in the inlet boundary condition ($-1 < x < 0$, $y = 0$) gives a values of ϕ almost 0 for $-1 < x < -0.5$ and rapidly grows to a 2 value for $-0.5 < x < 0$. For all the other boundaries the value of $\phi$is approximated to 0.



(a)



(b)

Figure 2.8: Plots for the hyperbolic tangent functions (a)Inlet (b)No flow boundaries

2.3.3 Discretization

From Section 1.2 the convection-diffusion equation (Eq.1.9) could be rewritten considering the source term value as zero:

$$\rho\frac{d\phi}{dt} + \rho u\frac{d\phi}{dx} + \rho v\frac{d\phi}{dy} = \frac{k}{c_p}\Big(\frac{d^2\phi}{dx^2} + \frac{d^2\phi}{dy^2}\Big) \tag{2.14}$$

The implicit coefficient form of the previous equation is known from Section 1.5

$$a_P\phi_P = a_E\phi_E + a_S\phi_S + a_W\phi_W + a_N\phi_N + b \tag{2.15}$$

[4]Smith and Hutton proposed $\alpha = 10$ as representative of a relatively sharp transmission[7]

Even though the problem asks for the steady state condition, the terms that contain time dependency were taken into account because more conclusions about time evolution of the phenomena and its computational cost could be obtained. The scheme chosen to solve the convection-diffusion equation is a implicit scheme because it gives physically satisfactory results . The spatial discretization and control volume geometry chosen for the case of study are shown in Table 2.5.

Geometrical Property	L	H	N_x	N_y	Δx	Δy
Value	$[-1, 1]$	1	200	100	0.01	0.01

Table 2.5: Domain spatial discretization for Smith-Huton Problem

The mesh used for this problem is the Grid B shown in the previous Assignment, which consists in a non uniform distribution of the grid points at the centre of the CV and the domain boundaries. In this disposal of the grid points it is important to note that there are grid nodes around all the boundary conveniently located at wall faces of the control volume, this mesh structure is saved in a matrix of dimensions $[N_x + 2][N_y + 2]$. The boundary condition information is saved inside these extra dimensions. Using this Grid allows the direct determination of boundary conditions and an easier analysis of the coefficients at these points. It is important to take into account that the distance between between these nodes and their neighbours is half of the central grid nodes.

Finally, it is seen that in Table 2.5 the control volume dimensions are the same ($\Delta x = \Delta y$) because the number of nodes for the y-dimension are half of the domain length. Selecting the same number of nodes in both dimensions would suggest adding more importance to the y-dimension which is not useful for the computational performance of the code.

2.3.4 Algorithm

With the purpose of understanding the algorithm developed for solving this problem Fig.2.9 gives us an idea about the main processes inside the code. For this problem a Matlab Object Oriented code[5] has been developed as a first contact with this programming method. The core of the code is the Main function, which contains all the necessary methods and input data. Inside the main it is found four principal functions:

- Uniform Mesh: in charge of the domain discretization and compute of the velocity field needed for each problem.
- Coefficient Compute: in charge of computing the needed coefficients for each case, the ones that are dependent on the field ϕ and the non-dependent.
- Solver: it is in charge of finding the value of ϕ for the coefficients previously calculated.
- Solver Shell: that function returns to the final results for the ϕ field. It contains the Solver.

The necessary data for starting the computations is modified from the ”inputData” file. It contains the points that define our domain (P_1, P_2), the requested solutions points, the sizes of our mesh (N_x, N_y), the initial field ϕ value and its boundary conditions, the physical properties needed for solving the coefficients and finally the solver parameters.

Figure 2.9 shows a diagram in order to represent the transient state of the Smith-Hutton problem. The first program iterated until the steady state was reached without taking into account what happened at each time step. With this new algorithm it is possible to represent the evolution of the problem along the time steps. The code used for developing this program was the same as the first but with some modifications inside the solver shell. The developed Matlab OOP code can be found inside Code Attachments document.

[5]This code can be downloaded with a document that describes the case of study by clicking ”here”.

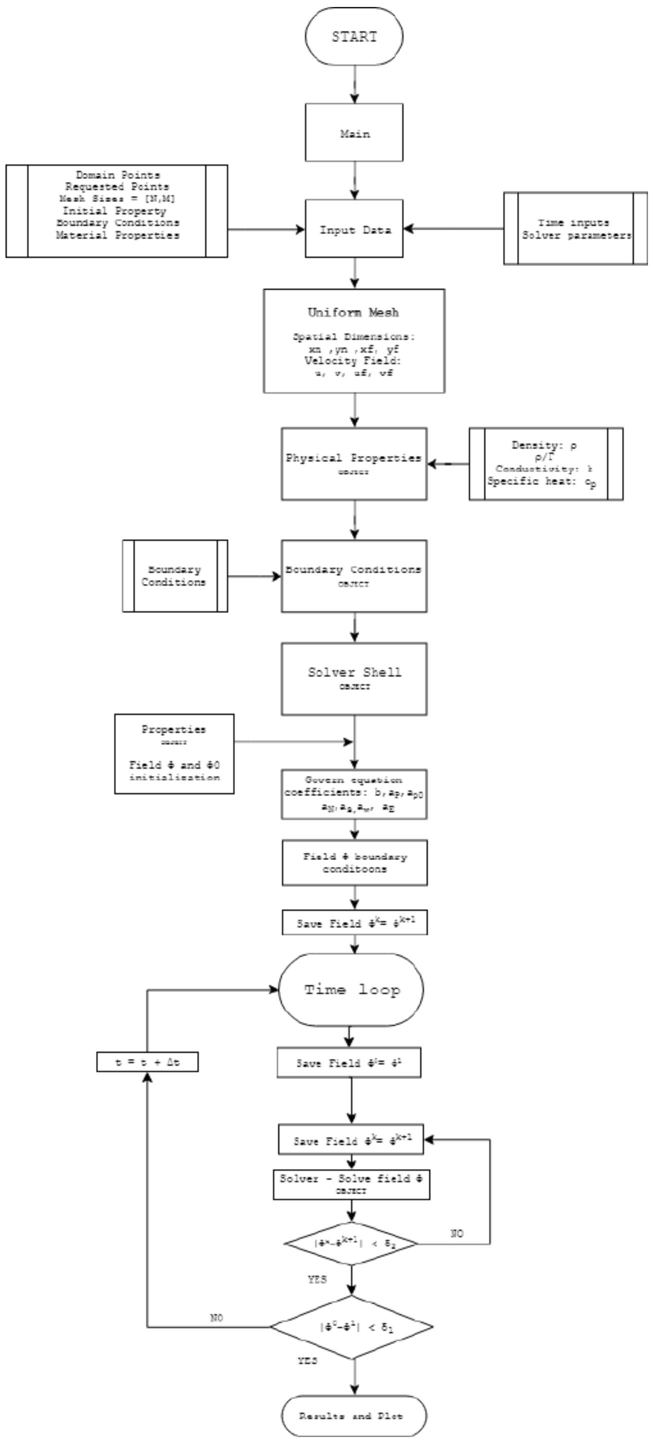


Figure 2.9: Smith-Hutton Transient Problem algorithm flowchart

2.3.5 Results

Once the code is running and working correctly it is necessary to compare the results obtained at the outlet of the contour with numerical results provided by [2] in order to check their validity. The results are displayed in Table 2.6 and the complete field for each situation is shown below (Fig. 2.10-B.2).

These results were obtained using the Upwind Numerical Scheme in the computation of the coefficients. As explained in Section 1.4 there are different possible and more optimal schemes to apply. In this study it is only necessary to check if it gives the correct results for each case and for that it is not necessary to apply different N-S or Solvers. In this case a Gauss-Seidel solver was implemented for its programming simplicity but the algorithms developed in the code are easy to attach with any iterative solver type.

	$\rho/\Gamma = 10$		$\rho/\Gamma = 1000$		$\rho/\Gamma = 10^6$	
Position x	Expected	Calculated	Expected	Calculated	Expected	Calculated
0.0	1.989	1.908	2.0000	2.0000	2.000	2.000
0.1	1.402	1.392	1.9990	1.9998	2.000	2.000
0.2	1.146	1.149	1.9997	1.9989	2.000	2.000
0.3	0.946	0.964	1.9850	1.9599	1.999	1.993
0.4	0.775	0.808	1.8410	1.6347	1.000	1.767
0.5	0.621	0.665	0.9510	0.8466	0.036	0.839
0.6	0.480	0.529	0.1546	0.2060	0.001	0.125
0.7	0.349	0.395	0.0010	0.01968	0.000	0.005
0.8	0.227	0.263	0.0000	0.0006	0.000	0.001
0.9	0.111	0.131	0.0000	0.0002	0.000	0.000
1.0	0.000	0.000	0.0000	0.0000	0.000	0.000

Table 2.6: Numerical results at the outlet for different ρ/Γ [2]

For more plots you can check Attachment E.

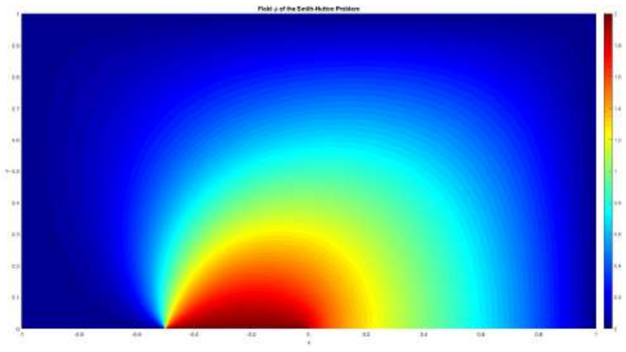


Figure 2.10: Field ϕ of the Smith-Hutton Problem for $\rho/\Gamma = 10$

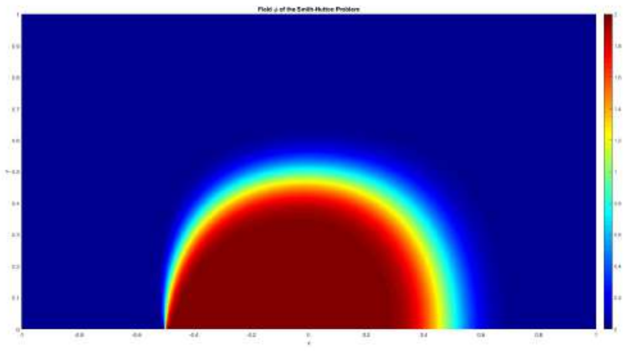


Figure 2.11: Field ϕ of the Smith-Hutton Problem for $\rho/\Gamma = 1000$

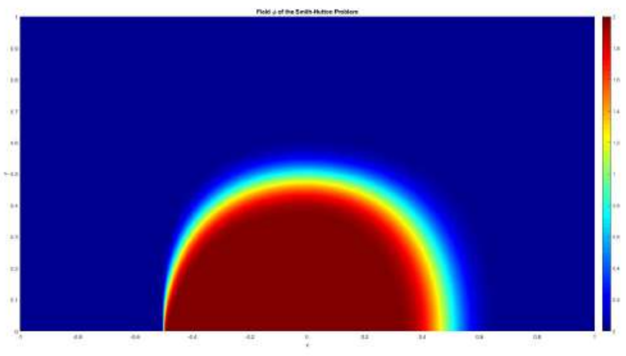
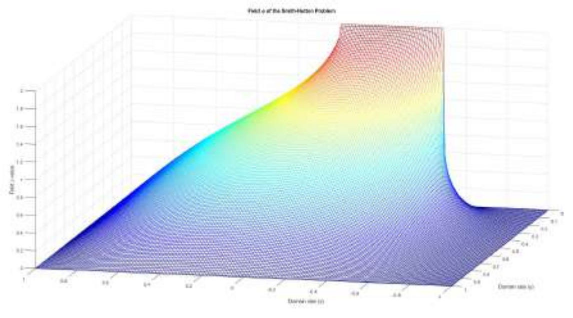


Figure 2.12: Field ϕ of the Smith-Hutton Problem for $\rho/\Gamma = 10^6$

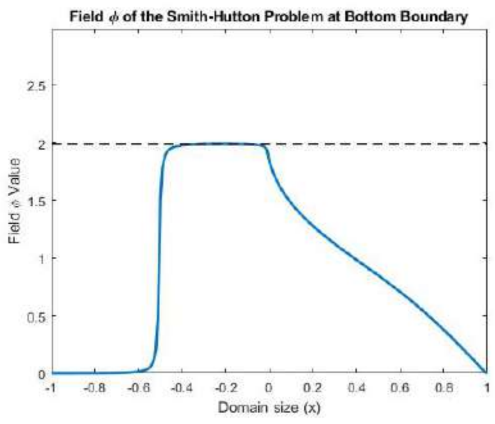
2.3.6 Conclusions

As the ρ/Γ ratio increases, the convective term grows taking a predominant role against the diffusive term, which decreases. This behaviour can be observed in the field ϕ plots shown in the figures below. In the first case where $\rho/\Gamma = 10$ it is known from the Peclet Eq. 1.4 that for these values the Peclet number is low. Which means that for low Peclet numbers the problem tends to have greater diffusive effects. But as the Peclet number increases the convective term gains influence, for that reason a solenoidal field is observed for the cases of greater ϕ/Γ.

The following plots shows us the nature of the results at the "outlet". Each one shows the evolution of the ϕ field values at the bottom boundary nodes. It is seen that the maximum values for this field are defined by the inlet boundary condition at the right half of the inlet for each case. These plots are shown in order to give a deeper insight into the nature of the results.



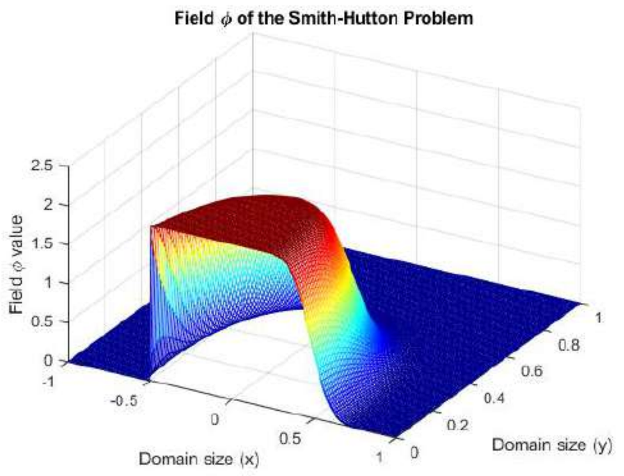
(a)



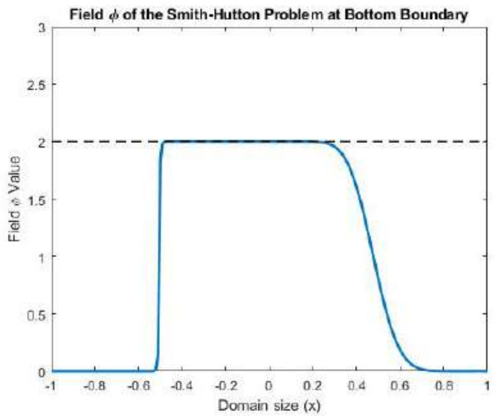
(b)

Figure 2.13: Evolution of the "outlet" Field ϕ for $\rho/\Gamma = 10$
(a)3-D Field (b)2-D Plot

In the first case, for low values of ρ/Γ, the diffusive term has the main role. For this reason the field value at the Point $(0, 0)$ rapidly decreases untill it reaches value zero at the end of the boundary. These results could be explained from the particle concentration view, thus it is an almost diffusive problem, the particle density is higher that means that our flow of study will not easily reach the form of the velocity field. Figure D.3a shows the Field ϕ values in a 3D plot as an extra way of analyzing the plots previously shown. .

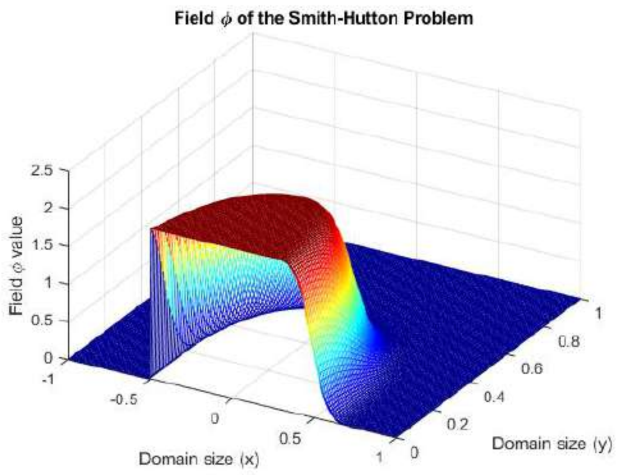


(a)

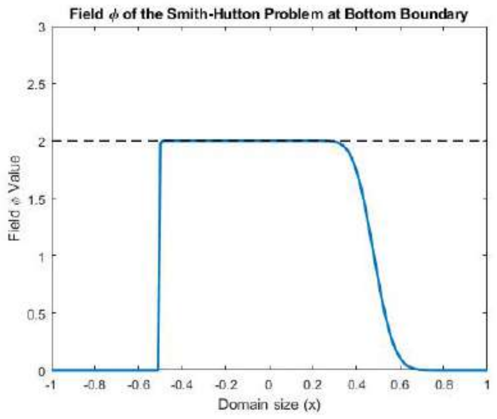


(b)

Figure 2.14: Evolution of the "outlet" Field ϕ for $\rho/\Gamma = 1000$
(a)3-D Field (b)2-D Plot



(a)



(b)

Figure 2.15: Evolution of the "outlet" Field ϕ for $\rho/\Gamma = 10^6$
(a)3-D Field (b)2-D Plot

Other cases (Fig 2.14 - 2.15) works with higher values of ρ/Γ. Thus the nature of the results would be convective, the comparison of these cases confirms that because there is almost no difference between results in both situations. In these cases, the fluid particles are easily able to follow the path marked by the velocity field. In the Field ϕ plot of the bottom boundary there is a symmetry of the results to the Y axis. This symmetry is imposed by the velocity to our domain and the bigger the value of ρ/Γ used, the faster this symmetry is achieved. In a physical sense this would mean that the fluid particles can easily follow the path described by the velocity field. This symmetry could be even better if it had not been directly supposed that

$$\phi = 2 \quad for \quad -0.5 < x < 0 \tag{2.16}$$

for this reason it is seen a step in the plots presented.

After the extraction of these conclusions from the development of this case, it has been noticed that the density for each case of study have been kept to $\rho = 10kg/m^3$. Realising that, some studies were realised modifying the density value but there were not variations in the results except in the isobaric plot for the $\phi/\Gamma = 10^6$ situation. For this case ($\rho = 10^6$) the field ϕ maintains the value of the second half of the inlet in its nearby region as seen in Fig.D.8.

Finally, it needs to be commented that the results shown in Table 2.6 do not exactly match the ones provided by the CTTC. For almost all the cases this error is similar but for bigger values of ϕ/Γ this error increases to the point where the curve slope is more pronounced. From the results it is extracted that the code gives a physically satisfactory result because they have the expected form but there will be some errors in the code that generates an error comparing to valid results.

References

[1] S. V.Patankar, *Numerical Heat Transfer and Fluid Flow*, 1st ed., 1980.

[2] D. CTTC, *Validation of the Convection-Diffusion Equation, Tech. rep. , ESEIAAT*.

[3] Y. A. Cengel, *Heat transfer: a practical approach*, 2nd ed., 2004.

[4] K. W. Morton, *Numerical solution of convection-diffusion problems*, 1st ed., 1996.

[5] F. M. White, *Fluid Mechanics*, 5th ed.

[6] D. CTTC, *Numerical solution of convection. Tech. rep. , ESEIAAT*, 2010.

[7] B. Leonard and S. Mokhtari, *ULTRA-SHARP Solution of the Smith-Hutton Problem.* Department of Mechanical Engineering, The University of Akron, 1992.

[8] G. Guiraldo, "What are "upstream" and "downstream" in fluid dynamics?" [Online]. Available: https://www.quora.com/What-are-upstream-and-downstream-in-fluid-dynamics

[9] H. K. Varsteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, 2007.

[10] J. B. Scarborough, *Numerical Mathematical Analysis*, 1st ed., 1955.

Appendix A

Scarborough Criterion

This Appendix explains the definition of the Scarborough criterion and its mathematical formulation. This criterion is used for satisfying the convergence of the solution of linear equation systems in iterative problems. It can be expressed in terms of the values of the coefficients of the discretised equations:

$$\frac{\Sigma|a_{ib}|}{|a_P|}\left\{\begin{array}{cc} \leq 1, & \text{at all nodes} \\ < 1, & \text{at one node at least} \end{array}\right\} \tag{A.1}$$

Where a_p corresponds to a random grid point P and the numerator sum states for the sum of all neighbour coefficients to node P. This is a sufficient condition but not a necessary one, what means that convergence could be achieved even if the criterion at some time is violated and the satisfaction of this criterion ensures that the equations will be converged by at least one iterative method. For more information see [9, 10, 1].

Appendix B

One-Dimensional Flow (X) Problem Results

This Appendix shows isobaric and mesh plots of the study of a one-dimensional flow with a unidimensional variation of the variable solved in the same direction of the flow. These results have been obtained using a self developed code[1] with the numerical scheme and solver indicated in the case of study.

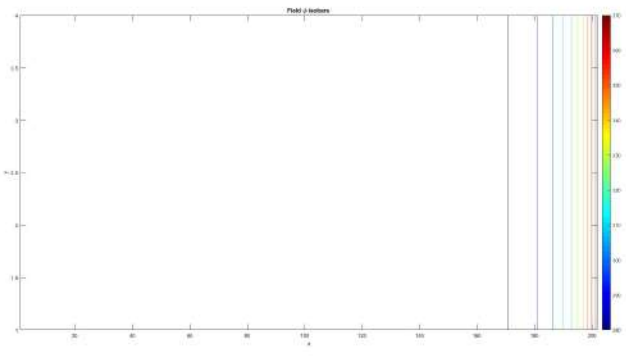


Figure B.1: Field ϕ isobars for one-dimensional flow along X-axis

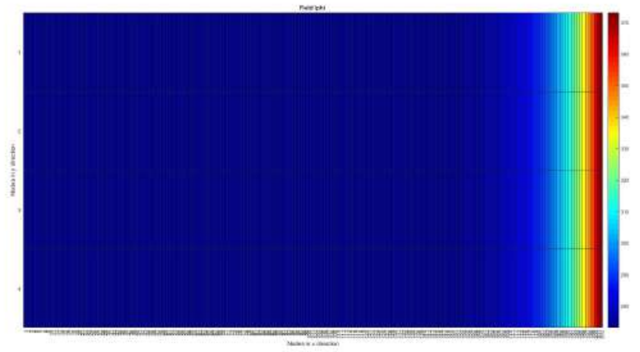


Figure B.2: Field ϕ mesh for one-dimensional flow along X-axis

[1]This code can be downloaded by clicking ”here”.

Appendix C

One-Dimensional Flow (Y) Problem Results

This Appendix shows isobaric and mesh plots of the study of a one-dimensional flow with a unidimensional variation of the variable solved in the perpendicular direction of the flow. These results have been obtained using a self developed code[1] with the numerical scheme and solver indicated in the case of study.

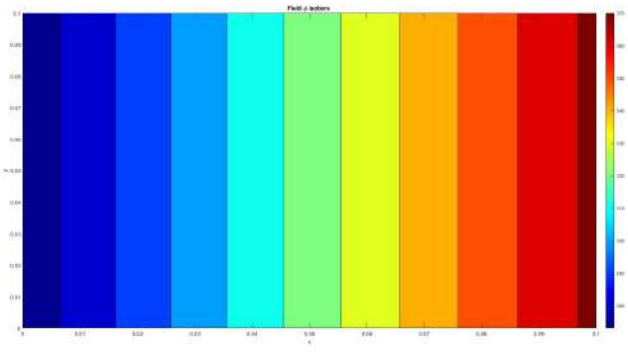


Figure C.1: Field ϕ isobars for one-dimensional flow along Y-axis

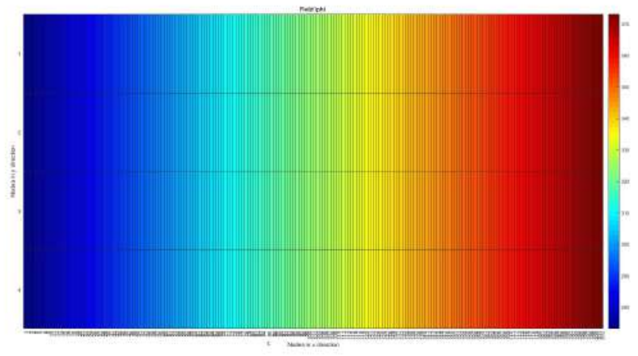


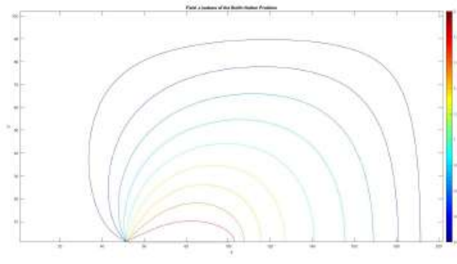
Figure C.2: Field ϕ mesh for one-dimensional flow along Y-axis

[1]This code can be downloaded by clicking "here".

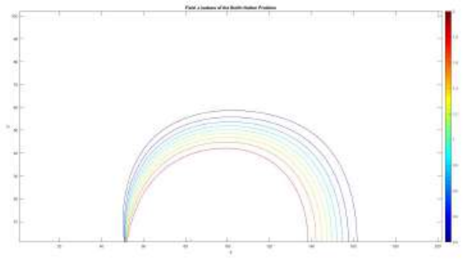
Appendix D

Smith-Hutton Problem Results

In this Appendix we can find isobaric and mesh plots of the Smith-Hutton Problem. These results have been obtained using a self developed code[1] with the numerical scheme and solver indicated in the case of study.



(a)



(b)

Figure D.1: Field ϕ isobars of the Smith-Hutton Problem for (a)$\rho/\Gamma = 10$ (b) $\rho/\Gamma = 10$

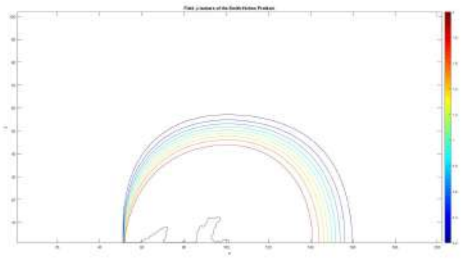
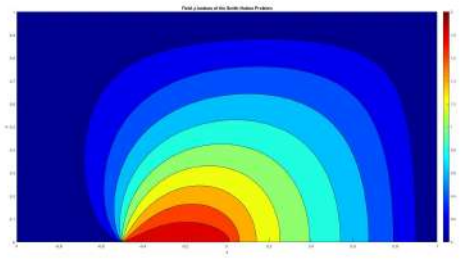


Figure D.2: Field ϕ isobars of the Smith-Hutton Problem for $\rho/\Gamma = 10^6$

[1]This code can be downloaded with a document that describes the case of study by clicking "here".


(a)

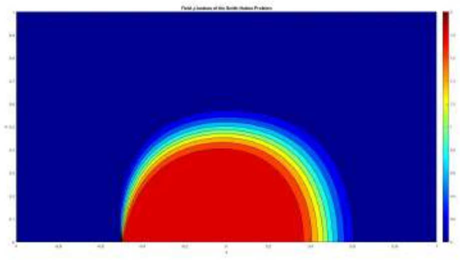

(b)

Figure D.3: Field ϕ color isobars of the Smith-Hutton Problem for (a)$\rho/\Gamma = 10$ (b) $\rho/\Gamma = 10$

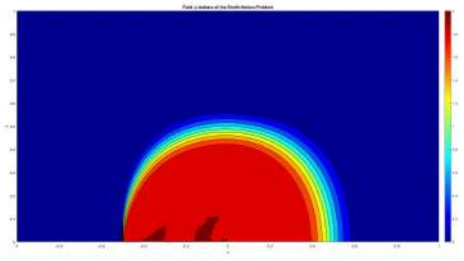


Figure D.4: Field ϕ color isobars of the Smith-Hutton Problem for $\rho/\Gamma = 10^6$

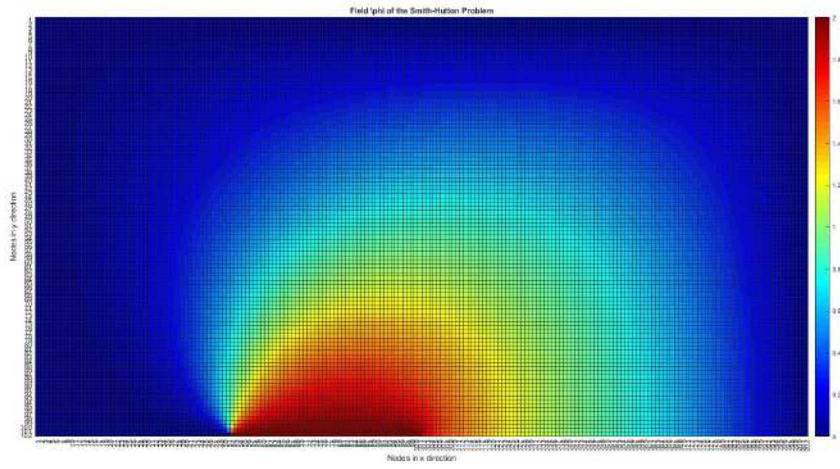


Figure D.5: Field ϕ grid of the Smith-Hutton Problem for $\rho/\Gamma = 10$

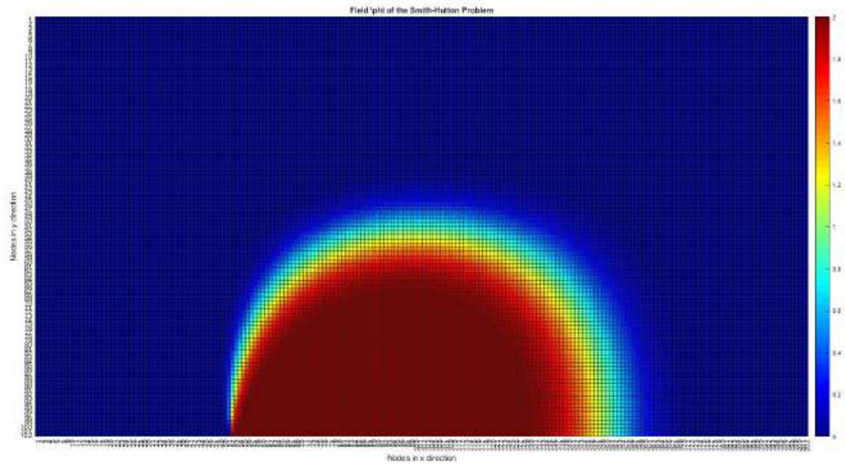


Figure D.6: Field ϕ grid of the Smith-Hutton Problem for $\rho/\Gamma = 1000$

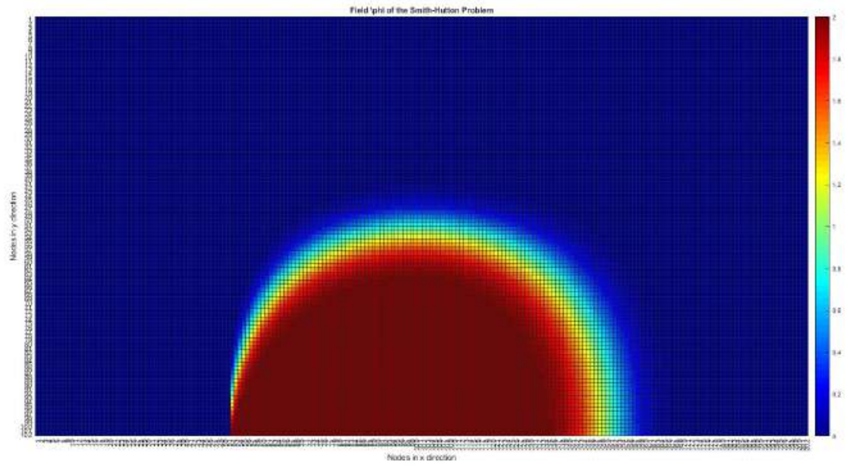


Figure D.7: Field ϕ grid of the Smith-Hutton Problem for $\rho/\Gamma = 10^6$

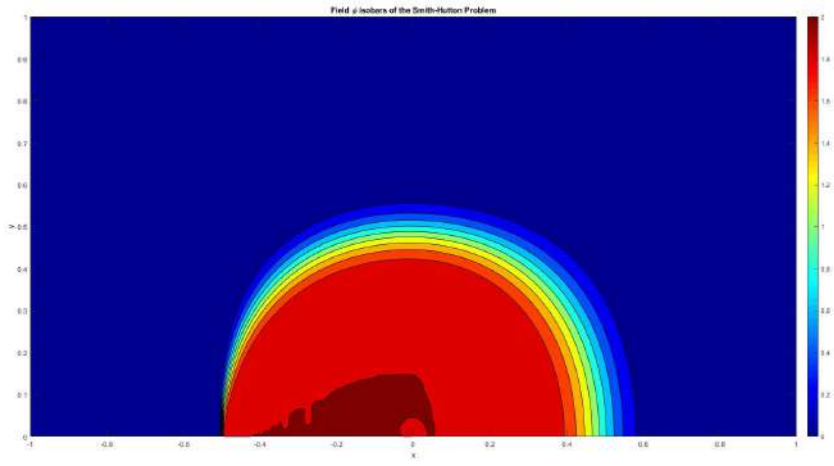


Figure D.8: Field ϕ of the Smith-Hutton Problem for $\rho/\Gamma = 10^6$ and $\rho = 10^6$

Appendix E

Smith-Hutton Problem Code

In this Appendix we can see the code developed with Matlab OOP for solving the Smith-Hutton Problem. This results has been obtained using a self developed code[1] with the numerical scheme and solver indicated in the case of study.

E.1 Main Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%  SMITH—HUTTON  %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%  Miquel Altadill Llasat  %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
more off
InputData

tic;
mesh=UniformMesh(domainPoints,meshSizes);
fprintf('MeshTime %f\n',toc); tic;

physProp=PhysProp(mesh,rhogamma,cp,k,rho);
fprintf('PhysPropTime %f\n',toc); tic;

boundCond=BoundCond(inletProp, outletProp, leftProp, rightProp, upperProp);
fprintf('BoundCondTime %f\n',toc); tic;

tcd2D=TransientConvectionDiffusion2D(mesh, physProp, boundCond, timeStep, initProp, refTime);
fprintf('CreateTHC2DTime %f\n',toc); tic;

tcd2D.solveTime( maxIter, maxDiff,PostProcess,maxtDiff, reqPoints);
fprintf('Solver Time %f\n',toc);
```

E.2 Input Data

```
%————————————————INPUT DATA————————————————
%—————————————————————————————————————————

%Domain lengths
%————————————————————————
```

[1]This code can be downloaded with a document that describes the case of study by clicking ”here”.

```
domainPoints=[-1 1; 0 1];       %First  row for X dim
                                %Second row for Y dim


%Requested points (x: OUTLET)
%-----------------------------------
reqPoints=[0.1 0; 0.2 0; 0.3 0 ; 0.4 0;  0.5 0; 0.6 0; 0.7 0; 0.8 0; 0.9 0; 1 0]; %[x1 y1; ...
    ... ...] points

%Mesh sizes
%---------------------
meshSizes=[100 50];

%Initial properties
%----------------------
initProp=1;

%Boundary conditions
%-----------------------
inletProp = [0 2];
outletProp = 0;
leftProp = 0;
rightProp = 0;
upperProp = 0;

%Time inputs
%----------------------
timeStep=1;
refTime=5.0e3;
maxtDiff=1e-5;

%Material properties
%----------------------
rhogamma=1000;
rho=1000;
cp=4;
k=170;

%Iterative solver parameters
%------------------------------
maxIter=1e4;
maxDiff=1e-5;


%Postprocessor Options
PostProcess = 11;     % 0 for no plots
                     % 11 for evolutive plot
```

E.3 Uniform Mesh Generation

```
%UNIFOR MESH & VELOCITY FIELD GENERATION
%------------------------------------------

classdef UniformMesh < handle
  properties (SetAccess=private)
    nodeX, nodeY, faceX, faceY, domain, U, V, Uf, Vf
  end
  methods

    function obj = UniformMesh(domainPoints,meshSizes)

      [domainLengths] = DomainLength(domainPoints);

      dim=1;
```

```
        [obj.nodeX,obj.faceX]=facesZVB(domainLengths(dim),...
            meshSizes(dim),domainPoints([1],[1]));

        dim=2;
        [obj.nodeY,obj.faceY]=facesZVB(domainLengths(dim),...
            meshSizes(dim),domainPoints([2],[1]));

          U   = zeros(numel(obj.nodeX),numel(obj.nodeY));
          V   = zeros(numel(obj.nodeX),numel(obj.nodeY));
          Uf  = zeros(numel(obj.faceX),numel(obj.faceY));
          Vf  = zeros(numel(obj.faceX),numel(obj.faceY));

          for indPX=1:numel(obj.nodeX)
              for indPY=1:numel(obj.nodeY)

                  x = obj.nodeX(indPX);
                  y = obj.nodeY(indPY);

                  obj.U(indPX,indPY)  =  2*y*(1-x^2);
                  obj.V(indPX,indPY)  =  2*x*(1-y^2);


              end
          end

          for indPX=2:(numel(obj.faceX))
              for indPY=1:(numel(obj.faceY))

                  xf = obj.faceX(indPX);
                  yf = obj.faceY(indPY);

                  obj.Uf(indPX,indPY)  =  2*yf*(1-xf^2);
                  obj.Vf(indPX,indPY)  =  -2*xf*(1-yf^2);

              end
          end

          %No slip boundary Condition

          obj.Uf(2:end,end)=0;    %TopBoundary
          obj.Vf(2:end,end)=0;
          obj.Uf(1,2:end)=0;      %LeftBoundary
          obj.Vf(1,2:end)=0;
          obj.Uf(end,2:end)=0;    %RightBoundary
          obj.Vf(end,2:end)=0;

      end



      function [s]=surfX(obj)
        s=obj.faceX(2:end)-obj.faceX(1:end-1);
      end
      function [s]=surfY(obj)
        s=obj.faceY(2:end)-obj.faceY(1:end-1);
      end
    end
end

%Domain Length
function [domainLengths] = DomainLength (domainPoints)

xLength = domainPoints([1],[2])-domainPoints([1],[1]);
yLength = domainPoints([2],[2])-domainPoints([2],[1]);
domainLengths=[xLength, yLength];

end
```

```
%facesZeroVolumeBoundaries
function [nx,fx]=facesZVB(length,numCV,initPoint)

fx=linspace(initPoint,initPoint+length,numCV+1);
nx(1,1)=initPoint;
nx(1,2:numCV+1)=(fx(2:end)+fx(1:end-1))*0.5;
nx(1,numCV+2)=initPoint+length;

end
```

E.4 Physical Properties

```
% PHYSICAL PROPERTIES DOMAIN FILLING
%----------------------------------------

classdef PhysProp < handle
  properties (SetAccess=private)
    rhogamma, cp, k, rho
  end
  methods
    function obj=PhysProp(mesh,rhogamma,cp,k,rho)

      sizeX=numel(mesh.nodeX);
      sizeY=numel(mesh.nodeY);

      obj.rhogamma=zeros(sizeX,sizeY);
      obj.cp=zeros(sizeX,sizeY);
      obj.k=zeros(sizeX,sizeY);
      obj.rho = zeros(sizeX,sizeY);

      %for one material

      obj.rhogamma(:,:)=rhogamma;
      obj.rho(:,:)=rho;
      obj.cp(:,:)=cp;
      obj.k(:,:)=k;

    end
  end
end
```

E.5 Boundary Conditions

```
%BOUNDARY CONDITIONS for defined PROPERTY
%----------------------------------------
classdef BoundCond < handle
    properties (SetAccess = private)
        inletProp, outletProp, leftProp, rightProp, upperProp
    end

    methods
        function obj = BoundCond(inletProp, outletProp, leftProp, rightProp, upperProp)
            obj.inletProp = inletProp;
            obj.outletProp= outletProp;
            obj.leftProp  = leftProp;
            obj.rightProp = rightProp;
            obj.upperProp = upperProp;
        end
    end
```

```
end
```

E.6 Equation Coefficients Compute

```
classdef Coefficients < handle
  properties (SetAccess=private)
    ap, ae, aw, an, as, ap0, b, Fe
  end
  methods
    function obj = Coefficients(mesh)
      obj.ap=zeros(numel(mesh.nodeX),numel(mesh.nodeY));
      obj.ap0=zeros(size(obj.ap));
      obj.ae=zeros(size(obj.ap));
      obj.aw=zeros(size(obj.ap));
      obj.an=zeros(size(obj.ap));
      obj.as=zeros(size(obj.ap));
      obj.b=zeros(size(obj.ap));
      obj.Fe=zeros(size(obj.ap));

    end


    %INNER MATRIX COEFFICIENTS
    %—————————————————————————
    function innerAfor(obj,physProp,mesh,timeStep,Prop)

      sizeX=size(obj.ap,1);
      sizeY=size(obj.ap,2);

      for indPX=2:sizeX−1
        for indPY=2:sizeY−1

          Se= (mesh.faceY(indPY)−mesh.faceY(indPY−1));
          Sw= Se;
          Sn= (mesh.faceX(indPX)−mesh.faceX(indPX−1));
          Ss= Sn;

          obj.Fe(indPX,indPY)  = Se*physProp.rho(indPX+1,indPY)*mesh.Uf(indPX,indPY);
          Fw  = Sw*physProp.rho(indPX−1,indPY)*mesh.Uf(indPX − 1,indPY);
          Fn  = Sn*physProp.rho(indPX,indPY+1)*mesh.Vf(indPX,indPY);
          Fs  = Ss*physProp.rho(indPX,indPY−1)*mesh.Vf(indPX,indPY − 1);


          De  = ((physProp.rho(indPX+1,indPY)/physProp.rhogamma(indPX,indPY))*Se)/...
               (mesh.nodeX(indPX+1) − mesh.nodeX(indPX));
          Dw  = ((physProp.rho(indPX−1,indPY)/physProp.rhogamma(indPX,indPY))*Sw)/...
               (mesh.nodeX(indPX) − mesh.nodeX(indPX−1));
          Dn=   ((physProp.rho(indPX,indPY+1)/physProp.rhogamma(indPX,indPY))*Sn)/...
               (mesh.nodeY(indPY+1) − mesh.nodeY(indPY));
          Ds=   ((physProp.rho(indPX,indPY−1)/physProp.rhogamma(indPX,indPY))*Ss)/...
               (mesh.nodeY(indPY) − mesh.nodeY(indPY−1));

          %Peclet number
          Pe  = obj.Fe/De;
          Pw  = Fw/Dw;
          Pn  = Fn/Dn;
          Ps  = Fs/Ds;

          %NUMERICAL SCHEME POWERLAW
          Ae =1;% max(0,  (1−0.1*abs(Pe))^5);
          Aw =1;% max(0,  (1−0.1*abs(Pw))^5);
          An =1;% max(0,  (1−0.1*abs(Pn))^5);
          As =1;% max(0,  (1−0.1*abs(Ps))^5);

```

```
            obj.ae(indPX,indPY)= De*Ae + max(-obj.Fe(indPX,indPY),0);
            obj.aw(indPX,indPY)= Dw*Aw + max(Fw,0);
            obj.an(indPX,indPY)= Dn*An + max(-Fn,0);
            obj.as(indPX,indPY)= Ds*As + max(Fs,0);

            obj.ap0(indPX,indPY)=Se*Sn*Prop.T(indPX,indPY)/timeStep;

            obj.ap(indPX,indPY) = obj.ap0(indPX,indPY)+obj.ae(indPX,indPY)+...
            obj.as(indPX,indPY)+obj.an(indPX,indPY)+obj.aw(indPX,indPY);

            %Same as function newInnerB
            obj.b(indPX,indPY) = obj.ap0(indPX,indPY)*Prop.T(indPX,indPY);

          end
        end
      end

      %INNER MATRIX TIME DEPENDENT COEFFICIENTS
      %--------------------------------------------
      %-Density = ct
      %-Velocity field = ct
      function innerAforTime(obj,mesh,timeStep,Prop)

          sizeX=size(obj.ap,1);
          sizeY=size(obj.ap,2);

          %Neuman Boundary Condition

          for indPX=1:sizeX
              for indPY=1:sizeY
                  if mesh.nodeX(indPX) > 0
                      Prop.T(indPX , 1) =  Prop.T(indPX,2);
                  end
                  if indPX==sizeX-1
                      Prop.T(indPX+1 , 1) =  Prop.T(indPX+1,2);
                  end
              end
          end


           for indPX=2:sizeX-1
              for indPY=2:sizeY-1

                  Se= (mesh.faceY(indPY)-mesh.faceY(indPY-1));
                  Sw= Se;
                  Sn= (mesh.faceX(indPX)-mesh.faceX(indPX-1));
                  Ss= Sn;

                  obj.ap0(indPX,indPY)=(Se*Sn*Prop.T(indPX,indPY))/timeStep;

                  obj.ap(indPX,indPY) = obj.ap0(indPX,indPY)+obj.ae(indPX,indPY)+...
                      obj.as(indPX,indPY)+obj.an(indPX,indPY)+obj.aw(indPX,indPY);

                  obj.b(indPX,indPY) = obj.ap0(indPX,indPY)*Prop.T(indPX,indPY);

              end
           end
      end

      function newInnerB(obj,Prop)
        obj.b(2:end-1,2:end-1)=obj.ap0(2:end-1,2:end-1).*Prop.T(2:end-1,2:end-1);
      end


      %BOUNDARY COEFFICIENTS
      %-------------------------------------------------
      function topBoundary(obj,upperProp,Prop)

        Prop.T(2:end-1,end) = upperProp;

      end

      function bottomBoundary(obj,outletProp,inletProp,Prop,mesh)

          sizeX=size(obj.ap,1);

          for indPX=1:sizeX

              if mesh.nodeX(indPX) < -0.5 && mesh.nodeX(indPX) ≥ -1
                  Prop.T(indPX , 1) =  inletProp(1);
              end
              if mesh.nodeX(indPX) > -0.5 && mesh.nodeX(indPX) ≤ 0
                  Prop.T(indPX , 1) = inletProp(2);
              end

              %Neuman Boundary Condition
              if(outletProp==0)
                  if mesh.nodeX(indPX) > 0
                     Prop.T(indPX , 1) =  Prop.T(indPX,2);
                  end
              end
          end

      end

      function leftBoundary(obj,leftProp,Prop)

        Prop.T(1,2:end) = leftProp;

      end

      function rightBoundary(obj, rightProp, Prop)
        Prop.T(end,2:end) = rightProp;
      end

    end
  end
```

E.7 Solver Function

```
% ITERATIVE SOLVER METHODS
%-----------------------------------------

classdef Solver < handle

    properties (SetAccess = private)

    end

    methods

        function obj=Solver(coef, Prop)


            %POINT-BY-POINT SOLVER
            %-------------------------------------------------
            % - Option 1
            sizeX=size(coef.ap,1);
            sizeY=size(coef.ap,2);

            for indPX=2:sizeX-1
                for indPY=2:sizeY-1
```

```
                    Prop.T(indPX,indPY) = (coef.ae(indPX,indPY)*Prop.T0(indPX+1,indPY)+ ...
                                          coef.aw(indPX,indPY)*Prop.T0(indPX−1,indPY)+ ...
                                          coef.an(indPX,indPY)*Prop.T0(indPX,indPY+1)+ ...
                                          coef.as(indPX,indPY)*Prop.T0(indPX,indPY−1)+ ...
                                          coef.b(indPX,indPY))/(coef.ap(indPX,indPY))   ;
                end
            end

            % − Option 2:
%             Prop.T(2:sizeX−1,2:sizeY−1) = ...
    (coef.ae(2:sizeX−1,2:sizeY−1).*Prop.T0((2:sizeX−1)+1,2:sizeY−1)+ ...
%                                         ...
    coef.aw(2:sizeX−1,2:sizeY−1).*Prop.T0((2:sizeX−1)−1,(2:sizeY−1))+ ...
%                                         ...
    coef.an(2:sizeX−1,2:sizeY−1).*Prop.T0((2:sizeX−1),(2:sizeY−1)+1)+ ...
%                                         ...
    coef.as(2:sizeX−1,2:sizeY−1).*Prop.T0((2:sizeX−1),(2:sizeY−1)−1)+ ...
%                                         ...
    coef.b(2:sizeX−1,2:sizeY−1))./(coef.ap(2:sizeX−1,2:sizeY−1))   ;

             %LINE−BY−LINE SOLVER
             %−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−



        end



    end

end
```

E.8 Field ϕ Properties

```
% PROPERTIES TO BE SAVED FOR THE POST PROCESSING AND SOLVING
%−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−


classdef Properties < handle
    properties  (SetAccess = public)

        T, T0, Tt

    end

    methods
        function obj = Properties(mesh, initProp)

            obj.T = zeros(numel(mesh.nodeX),numel(mesh.nodeY))+initProp;
            obj.T0 = zeros(numel(mesh.nodeX),numel(mesh.nodeY))+initProp;
            obj.Tt = zeros(numel(mesh.nodeX),numel(mesh.nodeY))+initProp;
        end
    end
end
```

E.9 Core of the code

```
%TRANSIENT 2−D CONVECTION−DIFFUSSION EQUATION
```

```matlab
%—————————————————————————————————————————————————

classdef TransientConvectionDiffusion2D < handle

    properties (SetAccess=public)
        mesh, physProp,boundCond ,timeStep, refTime , coef, Prop,Pref, err, tempReqPoints
    end

    %Prop = property to compute

    methods

        function  obj = TransientConvectionDiffusion2D(mesh, physProp, boundCond, ...
            timeStep, initProp, refTime)
            obj.mesh=mesh;
            obj.physProp=physProp;
            obj.boundCond=boundCond;
            obj.timeStep=timeStep;
            obj.refTime = refTime;

            obj.Prop = Properties(mesh, initProp);
            obj.coef = Coefficients(obj.mesh);

            obj.tempReqPoints = zeros(1000,10);


        end


        %Main function
        function solveTime(obj, maxIter, maxDiff,PostProcess,maxtDiff, reqPoints)


            %CONSTANT COEFFICIENTS
            %———————————————————————————————————————————————————————————————————————
            %Inner matrix
            obj.coef.innerAfor(obj.physProp,obj.mesh,obj.timeStep, obj.Prop);


            %Boundaries
            obj.coef.topBoundary(obj.boundCond.upperProp,obj.Prop);
            obj.coef.leftBoundary(obj.boundCond.leftProp, obj.Prop);
            obj.coef.bottomBoundary(obj.boundCond.outletProp,...
                                    obj.boundCond.inletProp,obj.Prop,obj.mesh);
            obj.coef.rightBoundary(obj.boundCond.rightProp,obj.Prop);



            %MAIN ALGORITHM
            %————————————————————————————————————————————————————————————————————————
            obj.Prop.T0 = obj.Prop.T;
            obj.Prop.Tt = obj.Prop.T;
            Time = zeros;                %Iteration time
            time = zeros;                    %Real time
            Error = zeros;

            obj.err = zeros (size(obj.coef.ap,1)-1,size(obj.coef.ap,2)-1)+1;
            tit = 0;                     %Time iterations

            %CORE OF THE CODE
            %————————————————————————————————————————————————————————————————
            diff2=inf;
            tic;

            while diff2 > maxtDiff

                tit = tit +1;         %Time iteration count
                time(tit) = tit*obj.timeStep;

                %INNER COEFFICIENTS
                %-------------------
                obj.coef.innerAforTime(obj.mesh,obj.timeStep,obj.Prop);

                %DOMAIN CONVERGENCE
                %------------------
                it = 0;
                diff1=inf;
                stop=0;

                while (diff1 > maxDiff)  || stop == 1

                    it = it +1;

                    %SOLVER
                    %------------------
                    Solver(obj.coef, obj.Prop);

                    % CONVERGENCE CHECK
                    %------------------
                    obj.err = abs(obj.Prop.T0-obj.Prop.T);
                    a = max(obj.err);
                    diff1 = max(a);

                    obj.Prop.T0 = obj.Prop.T;

                    if it > maxIter
                        error("Can not reach convergence of the results, check Input Data")
                        stop=1;
                    end
                end

                d2 = abs(obj.Prop.Tt-obj.Prop.T);
                d2i = max(d2);
                diff2 = max(d2i);%/obj.timeStep;
                obj.Prop.Tt=obj.Prop.T;
                Time(tit) = toc;
                fprintf('Time= %d; Ctime = %d;  Error = %d;  Iterations = %d; TimeStep = ...
                    %d\n',...
                            time(tit),Time(tit),diff2,it,tit);

                Error(tit) = diff2;

                obj.tempReqPoints(tit,:)  = interp2(obj.mesh.nodeY,obj.mesh.nodeX,...
                obj.Prop.T,reqPoints(:,1),reqPoints(:,2));



                %Postproces Evolutive Plot
                if PostProcess == 11
                    o = rot90(obj.Prop.T,-1);
                    O=fliplr(o);
                    figure(3);
                    pcolor(obj.mesh.nodeX,obj.mesh.nodeY,O);
                    shading interp;
                    colormap(jet);
                    colorbar;
                    title('Field \phi of the Smith-Hutton Problem');
                    xlabel('x'), ylabel('y');
                end


            end

            if PostProcess ==1

                postprocess(obj.Prop, obj.mesh);
```

```


                    %——————— 3—D FIELD PLOT ———————%
                    o = rot90(obj.Prop.T,—1);
                    O=fliplr(o);
                    figure(5)
                    mesh(obj.mesh.nodeX,rot90(obj.mesh.nodeY,—2),rot90(o,—2));
                    colormap(jet);
                    title('Field \phi of the Smith—Hutton Problem');
                    xlabel('Domain size (x)'), ylabel('Domain size (y)');
                    zlabel('Field \phi value');

                    % ———————— INLET AND OUTLET FIELD PLOT ——————————%
                    sizeX=size(obj.coef.ap,1);

                    for indPX=2:sizeX
                        vals(indPX—1)=obj.Prop.T(indPX,2);
                    end

                    x=linspace(—1,1,sizeX—1);

                    figure(6)
                    y= zeros(sizeX—1)+  max(vals);
                    p=plot(x,vals, x,y,'——k','LineWidth',1);
                    title('Field \phi of the Smith—Hutton Problem at Bottom Boundary');
                    xlabel('Domain size (x)'), ylabel('Field \phi Value');

                    ylim([min(vals) max(vals)+1]);
                    p(1).LineWidth = 2;

                end

            end

        end



    end

    %———————————————————————————————————————————————————————%
    %————————————————POSTPROCESSOR FUNCTION——————————————————%
    %———————————————————————————————————————————————————————%
    function postprocess(Prop, mesh)


        o = rot90(Prop.T,—1);
        O=fliplr(o);

        % ————————————ISOTHERM MAP ——————————————%
        figure(1);
        contour(O);
        colormap(jet);
        colorbar;
        title('Field \phi isobars of the Smith—Hutton Problem');
        xlabel('x'), ylabel('y');

        %—————————ISOTHERM COLOR MAP —————————%
        figure(2);
        contourf(mesh.nodeX,mesh.nodeY,O);      %mostra les isotermes
        colormap(jet);
        colorbar;
        title('Field \phi isobars of the Smith—Hutton Problem');
        xlabel('x'), ylabel('y');

        % —————————————COLOR MAP ———————————————%
        figure(3);
        pcolor(mesh.nodeX,mesh.nodeY,O);
        shading interp;
        colormap(jet);
        colorbar;
        title('Field \phi of the Smith—Hutton Problem');
        xlabel('x'), ylabel('y');



        %————————————  MESH PLOT ——————————————————%
        figure(4);
        h = heatmap(rot90(o,—2));  %heatmap
        colormap(jet);
        title('Field \phi of the Smith—Hutton Problem');
        xlabel('Nodes in x direction'), ylabel('Nodes in y direction');








     end
```