MSc Individual Project

Mathematical Modelling of Spread of Fire Underpinning Algorithmic Developments for Autonomous Swarms of Fire-Fighting Drones

M07MAE_1718MAYSEP

Coventry University

Paolo Grasso
8-21-2017

Supervisor:
Dr. Mauro S. Innocente
Declaration of Originality

This project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet, etc) has been acknowledged within the main report to an item in the References or Bibliography lists.

I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Copyright Acknowledgement

I acknowledge that the copyright of this project and report belongs to Coventry University.

Signed: Paolo Grasso  Date: 21/08/17


**ABSTRACT**

This project is focused on studying and modelling the fire spread in a two-dimensional space. The output models are mechanistic, parametric and based on actual well studied thermodynamic physics. This work is part of a wider research, which aims to develop algorithms for autonomous swarms of drones that would automatically extinguish any wildfire. One of the main objectives of this work is to perform a meticulous study of fire physics in order to prepare a strong knowledge base for eventual future studies related to this topic. The crucial aim of formulating a model of fire spread is fulfilled by three presented models. Model A tries to model the full physics but it fails due to a problem related to the dimensionality reduction from a three-dimensional space to a two-dimensional space. Model B overcomes the previous problem but is affected by numerical instability of the solution because it uses an inappropriate sequential explicit, Euler, approach to solve a system of coupled non-linear partial differential equations. Model C is the best candidate for eventual future developments of two-dimensional fire modelling. The main assumption in Model C is that it considers a constant pressure combustion. Moreover, here the numerical method adopted is Runge Kutta of 4th order that, even if still explicit and conditionally stable, gives a better solution, both qualitatively and quantitatively, and overcomes the memory-limit problem of the previous models. The obtained results are discussed in detail in order to formulate some useful recommendation for eventual future developments. All considered, formulating a three-dimensional turbulent multiphase and multiscale fire model and solving it on a dynamically morphing mesh using implicit algorithms that allow to manage simultaneously coupled non-linear partial differential equations is the best concluding advice.
ACKNOWLEDGEMENTS

I am very grateful to professor Mauro Innocente for his marvellous help and inspiring thoughts that guided me throughout the progress of the whole project.

A big thanks to Dr. Arash Dizgah, Dr. Mauro Innocente and Dr. Martin Herreros to welcome me in the SVeCLab (Smart Vehicles Control Lab) and to let me use its facilities for my study and research, offering me a quiet and constructive place to obtain my goals.

Last but not least, thanks to my parents, who always believed in my skills and determination, encouraging me to study at Coventry University. This allowed me to experience new things that I probably would never have in my country, enabling me to succeed in my MSc course and eventually in this project.
## CONTENTS

List of Figures ................................................................................................................................. 9
List of Tables ................................................................................................................................. 11
Glossary ........................................................................................................................................ 13

1. **Introduction** ......................................................................................................................... 15
   1.1. Objectives .......................................................................................................................... 16

2. **Literature Review** .................................................................................................................. 17
   2.1. Fundamentals of Fire ......................................................................................................... 17
   2.2. Governing Equations ........................................................................................................ 21
   2.3. Modelling Fire Spread ....................................................................................................... 24
   2.4. Numerical Method ........................................................................................................... 30
      2.4.1. Numerical Stability ................................................................................................. 30
      2.4.2. Discretization .......................................................................................................... 32
      2.4.3. Initial and Boundary Conditions ............................................................................. 33
      2.4.4. Memory Constrain ................................................................................................. 34

3. **Methodology** .......................................................................................................................... 37
   3.1. General Assumptions ......................................................................................................... 37
   3.2. Dimensionality Reduction .............................................................................................. 40
      3.2.1. Radiation Modelling ................................................................................................ 40
   3.3. Model A ............................................................................................................................ 43
   3.4. Model B ............................................................................................................................ 46
   3.5. Model C ............................................................................................................................ 50
      3.5.1. Runge Kutta Fromulation ...................................................................................... 52

4. **Results** .................................................................................................................................. 55

5. **Discussion** ............................................................................................................................ 63
LIST OF FIGURES

Figure 2.1. Link between macroscopic, mesoscopic and microscopic scales [3]. ........ 18
Figure 2.2. Different scales and the associated phases and porosities [3]...................... 18
Figure 2.3. Brief diagram of pyrolysis mechanism [4]. .............................................. 19
Figure 2.4. Composition of products of wood pyrolysis. The mass ratios of components in wood: carbon \( c = 0.511 \), hydrogen \( h = 0.064 \), oxygen \( o = 0.425 \) [5]................................. 20
Figure 2.5. Composition of products of wood gasification. The mass ratios of components in wood: \( c = 0.511 \), \( h = 0.064 \), \( o = 0.425 \). Wood 47 kg/h, humidity 6.5%, argon 13.55 slm, water plasma 18 g/min, CO2 115 slm, O2 30 slm [5]......................... 20
Figure 2.6. Hemispherical total emissive power [7]. .................................................. 22
Figure 2.7. Comparison between PROLIF and experiments for a controlled fire [8]. .... 25
Figure 2.8. Time of ignition (min) for each SFIRE model grid cell, and terrain height (m) line contours for (C70) coupled and (N70) non-coupled simulation [9]. ....................... 27
Figure 2.9. Slope effect on fire propagation [10]........................................................ 29
Figure 2.10. Fire spread simulation based on Margerit and Guillaume model [10]. ...... 29
Figure 2.11. Finite Element Method triangular mesh following the steepest gradient of temperature field [11]. ................................................................. 33
Figure 2.12. Temperature initial condition: a) Gaussian distribution; b) top-view ignited circular area in a 2D domain. ................................................................. 34
Figure 3.1. Specific heat capacity at constant pressure vs. temperature for various gases. ........................................................................................................... 38
Figure 3.2. Radiation Flux in x direction. ................................................................. 41
Figure 3.3. 2D radiation across a control volume. .................................................... 42
Figure 3.4. Representation of the sequential solution adopted by Model A. ............ 45
Figure 3.5. Staggered grid adopted for both Model A and Model B ............................ 46
Figure 3.6. Flux diagram of the iterative sequential numerical solution of the Model B.

Figure 4.1. Flame domain (red) spread and zero-fuel domain formation (green) ....... 56
Figure 4.2. Fire spread: ring formation, Model C. ................................................. 56
Figure 4.3. Temperature field on a 25m$^2$ domain at 8 s, Model C. .......................... 57
Figure 4.4. Temperature profile at half domain section in $x$ direction (ignition point at $x=2m$), Model C................................................................. 58
Figure 4.5. Density field on a 25m$^2$ domain at 8 s, Model C..................................... 59
Figure 4.6. Evolution in time of some characteristic properties at the ignition point ($x=2m$, $y=2.5m$) with Model C: a) combustion power, b) temperature, c) fuel mole fraction and d) oxidizer mole fraction. Extinction time: $t_e=5.25s$ ........................................ 60
Figure 4.7. Temperature profile in $x$ direction at half domain passing through the ignition point ($x=2m$, $y=2.5m$) at different time from the ignition (data from T_store, Model C). ............................................................................................................. 61
Figure 5.1. Pressure field on a 25m$^2$ domain, ignition point located in ($x=2m$, $y=2m$), Model A................................................................. 64
Figure 5.2. Buoyancy effect and recirculation in an unconstrained plum flame......... 65
Figure 5.3. Temperature profile in $x$ direction at half domain passing through the ignition point ($x=2m$, $y=2.5m$) at different time from the ignition (data from T_store, Model B). ............................................................................................................. 66
LIST OF TABLES

Table 3.1. Main assumptions shared by Model A, B and C. .................................................. 39

Table 3.2. Boundary conditions for velocities, pressure (for Model B), temperature and density (for Model A). ............................................................................................................. 44

Table 4.1. Space and time discretization of models A, B and C............................................... 55
**GLOSSARY**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dx$</td>
<td>Spatial discretization in x direction</td>
<td>[m]</td>
</tr>
<tr>
<td>$dy$</td>
<td>Spatial discretization in y direction</td>
<td>[m]</td>
</tr>
<tr>
<td>$dz$</td>
<td>Spatial discretization in z direction</td>
<td>[m]</td>
</tr>
<tr>
<td>$dt$</td>
<td>Time discretization</td>
<td>[s]</td>
</tr>
<tr>
<td>$nx$</td>
<td>Number of cells in x direction</td>
<td>--</td>
</tr>
<tr>
<td>$ny$</td>
<td>Number of cells in y direction</td>
<td>--</td>
</tr>
<tr>
<td>$l_R$</td>
<td>Radiation absorption length</td>
<td>[m]</td>
</tr>
<tr>
<td>$\mathbf{\bar{v}}$</td>
<td>Wind velocity vector</td>
<td>[m/s]$^3$</td>
</tr>
<tr>
<td>$u$</td>
<td>Wind velocity component in x direction</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$v$</td>
<td>Wind velocity component in y direction</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$w$</td>
<td>Wind velocity component in z direction</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$p$</td>
<td>Pressure of the mixture</td>
<td>[Pa]</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Density of the mixture</td>
<td>[kg/m$^3$]</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Viscosity of the mixture</td>
<td>[kg/s/m]</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Kinematic viscosity of the mixture</td>
<td>[m$^2$/s]</td>
</tr>
<tr>
<td>$c_P$</td>
<td>Constant pressure heat capacity of the mixture</td>
<td>[J/kg/K]</td>
</tr>
<tr>
<td>$c_P\alpha$</td>
<td>Heat capacity of the $\alpha$-th species</td>
<td>[J/kg/K]</td>
</tr>
<tr>
<td>$c_s$</td>
<td>Constant pressure heat capacity of solid in Weber [1]</td>
<td>[J/kg/K]</td>
</tr>
<tr>
<td>$c_g$</td>
<td>Constant pressure heat capacity of gas mixture, Weber [1]</td>
<td>[J/kg/K]</td>
</tr>
<tr>
<td>$M$</td>
<td>Molar mass of the mixture</td>
<td>[kg/mol]</td>
</tr>
<tr>
<td>$M\alpha$</td>
<td>Molar mass of the $\alpha$-th species</td>
<td>[kg/mol]</td>
</tr>
<tr>
<td>$H\alpha$</td>
<td>Molar heat of formation of the $\alpha$-th species</td>
<td>[J/mol]</td>
</tr>
<tr>
<td>$e$</td>
<td>Non dimensional enthalpy in Ferragut formulation [2]</td>
<td>--</td>
</tr>
<tr>
<td>$h$</td>
<td>Specific enthalpy</td>
<td>[J/kg]</td>
</tr>
<tr>
<td>$h_c$</td>
<td>Specific heat of combustion of the mixture</td>
<td>[J/kg]</td>
</tr>
<tr>
<td>$k$</td>
<td>Thermal conductivity of the mixture</td>
<td>[J/m/K/s]</td>
</tr>
<tr>
<td>$k_s$</td>
<td>Thermal conductivity of solid in Weber [1]</td>
<td>[J/m/K/s]</td>
</tr>
<tr>
<td>$k_g$</td>
<td>Thermal conductivity of gas mixture in Weber [1]</td>
<td>[J/m/K/s]</td>
</tr>
<tr>
<td>$E_A$</td>
<td>Combustion activation energy per mole</td>
<td>[J/mol]</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>$A_r$</td>
<td>Pre-exponential multiplicator for reaction rate</td>
<td>$[1]$</td>
</tr>
<tr>
<td>$A_g$</td>
<td>Pre-exponential multiplicator for reaction rate Weber</td>
<td>$[2]$</td>
</tr>
<tr>
<td>$T$</td>
<td>Temperature field</td>
<td>$[K]$</td>
</tr>
<tr>
<td>$T_A$</td>
<td>Combustion activation temperature</td>
<td>$[K]$</td>
</tr>
<tr>
<td>$T_0$</td>
<td>Ambient temperature</td>
<td>$[K]$</td>
</tr>
<tr>
<td>$T_p$</td>
<td>Pyrolysis temperature</td>
<td>$[K]$</td>
</tr>
<tr>
<td>$u$</td>
<td>Non-dimensional temperature in Ferragut formulation [2]</td>
<td>--</td>
</tr>
<tr>
<td>$u_a$</td>
<td>Non-dimensional atmosphere temperature in Weber [1]</td>
<td>--</td>
</tr>
<tr>
<td>$u_s$</td>
<td>Non-dimensional porous solid temperature in Weber [1]</td>
<td>--</td>
</tr>
<tr>
<td>$u_g$</td>
<td>Non-dimensional gas temperature in Weber [1]</td>
<td>--</td>
</tr>
<tr>
<td>$i$</td>
<td>Space index in x direction</td>
<td>--</td>
</tr>
<tr>
<td>$j$</td>
<td>Space index in y direction</td>
<td>--</td>
</tr>
<tr>
<td>$k$</td>
<td>Space index in z direction</td>
<td>--</td>
</tr>
<tr>
<td>$n$</td>
<td>Time step index</td>
<td>--</td>
</tr>
<tr>
<td>$Y_\alpha$</td>
<td>Mass fraction with respect to total mixture mass</td>
<td>--</td>
</tr>
<tr>
<td>$X_\alpha$</td>
<td>Molar fraction</td>
<td>--</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Boltzman universal gas constant</td>
<td>$[J/m^2/K^4/s]$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>Emittance of the mixture (0 &lt; $\epsilon$ &lt; 1)</td>
<td>--</td>
</tr>
<tr>
<td>$R$</td>
<td>Universal gas constant</td>
<td>$[J/mol/K]$</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>General field property (such as $T$, $u$, $v$, etc...)</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$Trsp.$</td>
<td>Transport term</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$Diff.$</td>
<td>Diffusion term</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$S.$</td>
<td>Source or Sink</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$Rad$</td>
<td>Radiation term in 2D</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$Rad_z$</td>
<td>Radiation term sink in z direction</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$Conv_z$</td>
<td>Convection sink in z direction</td>
<td>$[3]$</td>
</tr>
<tr>
<td>$\tau_{RAD}$</td>
<td>Optical thickness of the medium</td>
<td>$[m]$</td>
</tr>
</tbody>
</table>

1. Depends on the specific empirical formula considered for the combustion reaction rate, which dimension must be $[s^{-1}]$
2. Depends on units used in Weber one-dimensional fire formulation [1]
3. Depends on the particular partial differential equation considered
1. INTRODUCTION

The fire is a quite fascinating phenomenon, its mysterious elusive nature always attracted the humankind, which finally managed to control it in an increasingly ingenious way. It is as enchanting as fearful. In fact, year by year wildfire endanger urban settlements becoming a treacherous enemy. It has to be controlled and finally extinguished, but to optimize the extinction process one has to know in deep all the involved phenomena and has to predict very accurately its spread in a quite inhomogeneous environment. This could be done by means of numerical modelling. Recently, the interest in wild fire modelling is particularly increased thanks to new advances of both computational power and remote sensing and geographic information systems (GIS) that make possible a more effective and detailed wildland fire behaviour modelling [3].

The presented work consists in defining a mechanistic parametric mathematical model, with a substantial physics base, that allows to simulate the spread of fire. This can be placed in a wider possible research act to developing algorithms for autonomous swarms of fire-fighting drones that would automate and improve the fire-extinction manoeuvres. A part my personal interest in mathematical modelling and UAVs (Unmanned Aerial Vehicles), the fact that this new future effective technology would save a lot of human lives and woodlands, decreasing the fire impact on health, safety and cost, makes me very willing to study in deep the problem of fire spreading modelling.

The presented work includes:

- In Chapter 2: a deep fundamental study of combustion physics, a detailed survey of the most relevant fire spreading models available in literature and a review of the possible applicable numerical methods that could better the quality of the simulation or save computational time.
- In Chapter 3: the methodology adopted to model the fire spread in a two-dimensional domain. Three models are proposed: Model A, Model B and Model C. Each one has its own particularity with regards to both modelled physics and numerical approach.
• In Chapter 4: the main results related to the simulation of Model C are proposed.
• In Chapter 5: an important discussion about different problematics related especially to Model A and Model B is dealt.

1.1. **OBJECTIVES**

The aim of this project, as previously explained, is to design a mechanistic parametric model of spread of fire in contrast with the available empirical models. The new model will allow a more systematic study of the fire-spread phenomenon. In particular, the main objectives are:

• To read at least twenty articles related to fire-spread modelling and review in detail the five most relevant in order to write a detailed and critical Literature Review by 12/06/2017.
• To build from scratches a parametric model of fire spread in 2D environment without atmospheric contribution using Matlab or other programming codes by 26/06/2017.
• To write a new, more complex, parametric model to predict more accurately the fire spread in 2D environment by 31/07/2017.
• To review the obtained results and suggest changes to the 2nd code in order to predict spread of fire in 2D or 3D environment with contributions of eventual other relevant aspects by 4/08/2017.

All these objectives are precisely fulfilled by the end of this project.
2. LITERATURE REVIEW

As it could be extracted from reliable and specific literature sources, the general direction is to achieve an accurate mechanistic wildfire model, based on actual fire physics and not neither stochastic nor empirical conjectures. All considered, the simulation is meant to be performed in real time. This seems to be a quite ambitious goal since there is no model in literature that satisfies all these objectives together, but new numerical techniques and new more powerful processor clusters could hopefully allow nowadays to reach that aim. In the following sections, the fundamentals of fire (Section 2.1) and its governing differential equations (Section 2.2) will be accurately described. Furthermore, the most relevant and available fire spread models (Section 2.3) and related numerical techniques (Section 2.4) will be introduced.

2.1. FUNDAMENTALS OF FIRE

The description of the fire phenomenon could not be strictly simplified to the only combustion of a certain available quantity of fuel but it has to consider a wider set of phenomena interrelated one to each other. The complexity of the phenomenology of fire is evident in both Figure 2.1 and Figure 2.2. The fire dynamics involve different length scales [4]:

- The Atmospheric Giga-scale describes the atmospheric wind that does not act only unidirectionally on the fire but, if the fire domain is quite large, the wind vector field can be affected by the fire presence (look Froude number in Section 0).
- The Macroscopic scale, the largest level of porosity. It is related to the fuel occupational density, the particular bushes structure and other macroscopic factors such as the terrain topography (slope).
- The Mesoscopic structure is related to the leaves distribution in the bushes and their individual structure. For instance, comparing with the Hardwood species, the Coniferous species allows a better mixing of fuel, provided by leaves, and oxidants, present in both air and leaves.
• The Microscopic scale, the smallest level of porosity. At this level, all the unnumbered chemical species are involved in reactions. However, one should take into account that, both solid and liquid phases can oxidize but only the species at the gaseous phase can take part in the exothermic combustion reaction.

![Diagram showing macroscopic, mesoscopic, and microscopic scales](image)

**Figure 2.1.** Link between macroscopic, mesoscopic and microscopic scales [4].

![Table showing different scales and porosities](image)

**Figure 2.2.** Different scales and the associated phases and porosities [4].

As previously stated, it is important to note that combustion takes place at the gas state, so, if the fuel is solid or liquid, it has to dry and separate creating some
gas. So, at the microscopic level, the pyrolysis phenomenon takes place at a certain pyrolysis temperature providing gaseous fuel to be burnt [5]. The pyrolysis phenomenon, depicted in Figure 2.3, is divided in three processes: the drying separates the moisture, water, from the dry fuel, the primary pyrolysis consists of a first extraction of liquid (tar) and gas from the remaining solid fuel (char) and finally the secondary pyrolysis is a reformulation of the chemical equilibrium of the fuel components, which gaseous part is ready to take part in the combustion reaction. Moreover, one should consider the mass formation rate of each chemical species in a stochastic way since the chemical equilibrium is quite unstable due to the unsteady change of the temperature field, especially in proximity of the flame front where the temperature gradient is the steepest.

As far as the wild fire is considered, the pyrolysis of wood is far more complex. In a quite detailed study on the progress in biomass and bioenergy production [6] by S. S. Shaukat, the pyrolysis of wood covers a large part of the research because this process is critical for fire ignition. In fact, the circumstances at which ignition takes place are very crucial for determining if a flame will spread or not. It is important to note that the pyrolysis is a continuous process producing products at different concentrations with regard to the source temperature. This happens because the chemical equilibrium shifts from high energy compounds to low energy compounds as far as the temperature is increased, Figure 2.4. This is considered as primary pyrolysis. During the secondary pyrolysis, with the
gasification process in Figure 2.5, the final gaseous species are produced. If the temperature is high enough to ignite the mixture of some combustible gasses with oxygen, the first flame will form. From this moment, if there is enough fuel concentration in the neighbourhoods, the heat generated by the first sight of combustion will accelerate the wood pyrolysis, triggering a chain effect.

Figure 2.4. Composition of products of wood pyrolysis. The mass ratios of components in wood: carbon $c = 0.511$, hydrogen $h = 0.064$, oxygen $o = 0.425$ [6].

Figure 2.5. Composition of products of wood gasification. The mass ratios of components in wood: $c = 0.511$, $h = 0.064$, $o = 0.425$. Wood 47 kg/h, humidity 6.5%, argon 13.55 slm, water plasma 18 g/min, CO: 115 slm, O: 30 slm [6].
2.2. GOVERNING EQUATIONS

As far as all the phenomena contributing to the formation of fire are well understood, one could define a set of general equilibrium equations [7] that describes the combustion of a reacting flow. In a three-dimensional space environment, the combustion dynamics are described by set of unsteady non-linear partial differential equations from (2.1) to (2.4) and by the pressure closure equation (2.5). In these equations the density, \( \rho \), is the total density of the mixture at a certain pressure and temperature condition that, in presence of a flame, changes quite consistently in space-time.

\[
\begin{align*}
\text{Mass conservation:} & \quad \frac{D\rho}{Dt} = 0 & (2.1) \\
\text{Momentum balance:} & \quad \frac{D(\rho \vec{V})}{Dt} = -\nabla \cdot (p) - \mu \nabla^2 \vec{V} & (2.2) \\
\text{Enthalpy equilibrium:} & \quad \frac{D (\rho h)}{Dt} = \frac{\partial p}{\partial t} - \frac{\partial}{\partial x_i} \left( \frac{k}{c_p} \frac{\partial h}{\partial x_i} \right) + \text{Rad}(T^4) & (2.3) \\
\text{Species conservation:} & \quad \frac{D (\rho Y_a)}{Dt} = \frac{\partial}{\partial x_i} \left( \rho D \frac{\partial Y_a}{\partial x_i} \right) + S_a & (2.4) \\
\text{Pressure:} & \quad p = \rho TR_a \sum_a \frac{Y_a}{M_a} & (2.5)
\end{align*}
\]

where \( \frac{\partial \phi}{\partial x_i} = \frac{\partial \phi}{\partial x} + \frac{\partial \phi}{\partial y} + \frac{\partial \phi}{\partial z} \) \hspace{1cm} (2.6)

Note that the total derivative of a general quantity \( \phi \) could be also written as (2.7):

\[
\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \vec{V})
\]

(2.7)

The additional partial derivatives of the mix products \( \phi_i \), (by the velocity vector field) form the Transport term in each one of the balance equations. Its meaning is actually how much the specific quantity (that could be mass, momentum,
enthalpy or species mass fraction) is transported throughout the domain by the velocity vector \( \vec{v} \), which component in \( x \), \( y \) and \( z \) are respectively \( u \), \( v \) and \( w \).

The \( \nabla^2 \vec{v} \) term is the Laplacian of the wind velocity vector. For example, in \( x \) direction it can be formulated as (2.8):

\[
\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}
\]

(2.8)

Furthermore, since the temperature field is of high interest with respect to the energy of the system, one could express the specific enthalpy (2.9) in function of the temperature of the mixture, \( T \), and of the specific enthalpy of formation of each gaseous chemical species \( h_{f,\alpha} \).

\[
h = \sum_{\alpha} (c_{p,\alpha}T + h_{f,\alpha}Y_\alpha) = \sum_{\alpha} (c_{p,\alpha}T) + h_c
\]

(2.9)

where \( c_{p,\alpha} \) is the specific heat capacity of the \( \alpha \)-th chemical species, and \( h_c \) is the specific combustion enthalpy of the particular mixture, which could also be non-stoichiometric.

For thermodynamics dominated by diffusion or convection, the radiation term is usually neglected but, in the fire phenomenon, it has a very important contribution. Therefore, very crucial for the energy balance in thermodynamics of fire is the radiation phenomenon. It is emission of a wide spectrum of electromagnetic waves that are absorbed and reflected by the medium in the neighbourhood of the emissive source,

Figure 2.6. The radiative power is proportional to the 4-th power of the temperature [8]. This thermal radiation heat transfer is precisely derived in Section 3.2.

Figure 2.6. Hemispherical total emissive power [8].
All considered, the system (2.1) - (2.5) could be reformulated in (2.10) - (2.16). It has to be highlighted that (2.2) consists actually of three equations in the three spatial directions, (2.11) - (2.13) and that (2.4) and (2.15) are actually represented by α equations as many are the chemical species.

\[
\begin{align*}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 & (2.10) \\
\frac{\partial (\rho \mathbf{u})}{\partial t} &= \frac{\partial (\rho \mathbf{u})}{\partial x} + \frac{\partial (\rho \mathbf{u})}{\partial y} + \frac{\partial (\rho \mathbf{u})}{\partial z} - \frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) & (2.11) \\
\frac{\partial (\rho \mathbf{v})}{\partial t} &= \frac{\partial (\rho \mathbf{v})}{\partial x} + \frac{\partial (\rho \mathbf{v})}{\partial y} + \frac{\partial (\rho \mathbf{v})}{\partial z} - \frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) & (2.12) \\
\frac{\partial (\rho \mathbf{w})}{\partial t} &= \frac{\partial (\rho \mathbf{w})}{\partial x} + \frac{\partial (\rho \mathbf{w})}{\partial y} + \frac{\partial (\rho \mathbf{w})}{\partial z} - \frac{\partial p}{\partial z} + \mu \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) & (2.13) \\
\frac{\partial (\rho Y_a)}{\partial t} &= \frac{\partial (\rho Y_a)}{\partial x} + \frac{\partial (\rho Y_a)}{\partial y} + \frac{\partial (\rho Y_a)}{\partial z} & (2.15)
\end{align*}
\]
\[ p = \rho TR_0 \sum_{\alpha} \frac{Y_\alpha}{M_\alpha} \]  

(2.16)

In the (2.10) - (2.15) one could recognise the various Transport and Diffusion terms and some Sources of energy (Combustion and Radiation) or species (S_\alpha). The system (2.10) - (2.16) is defined by 6+\alpha equations in 6+\alpha unknowns which are:

- Density, \( \rho \);
- Velocity vector, \( \mathbf{v} = \{u, v, w\} \);
- Temperature, \( T \);
- Mass fractions, \( Y_\alpha \);
- Pressure, \( p \).

### 2.3. MODELLING FIRE SPREAD

Many fire spread models have been developed throughout the last half century. Every modeler makes his own assumption based either on particular constraints of the simulated domain or on the simplified physics to be simulated.

Between the latest fire model formulations, the first validated model worthy to be considered is PROLIF [9]. This predicts the evolution of the fire front geometry with small regard to the energy equation. The fire contour is modelled by a complex Fourier series function and it is based on Cekirge formulation [10]. This is a very good model for unidirectional single fire front propagation, as shown in Figure 2.7. However, PROLIF becomes obsolete for more intricate situations like multiple ignition point and multidirectional fire propagation.
Other researchers, like Mell [11], use predefined commercial software specific for wildfire spread simulations. One of these is WFDS, Wildland-urban-interface Fire Dynamics Simulator. This is based on a Smagorinsky dynamic turbulent model [12], performing a Large Eddy Simulation (LES) with low Mach approximation. A peculiarity of WFDS is that it allows to simulate also the smoke production. This lacks in other reviewed models. Moreover, the fire fluid dynamics are better simulated because only the smallest turbulent scales are still modelled, while the largest scales are fully resolved.

Some other fire spread models consider the two gaseous and solid phases of the fuel, such as PIF97 [13]. The main idea of Dupuy and Larini’s model, PIF97, is to consider that the flaming combustion zone induces gas flows both inside and above the porous fuel medium. This phenomenon is neglected by models older than this one. However, PIF97 lacks of atmospheric wind interaction modeling, since the fire dynamics are assumed to be dominated by the fire kinetics themselves. PIF97 therefore becomes obsolete in case of modelling windy wildlands fire spread.

Other researchers, like Simpson [14], use commercial simulators like WRF, Weather Research Forecasting [15] v.3.6 with the Fire physics module developed...
by Coen [16]. This model, as the WFDS one, uses explicit LES. This is based on Rothermel’s fire spread formulation [17]. It consists of an algebraical addition of the wind, $\phi_{\text{wind}}$, and slope, $\phi_{\text{slope}}$, corrections to the original fire spread rate, $R_0$ (2.17), allowing to determine step by step the fire front spread.

$$ R = R_0 (1 + \phi_{\text{wind}} + \phi_{\text{slope}}) \quad (2.17) $$

Similarly to WRF’s model, Duane defines a weighted function for the fire spread ratio, $SR$ (2.18), that is used to calculate the speed derivative in time, $S_t$, defined in a stochastic way (2.19) [18]. Since the focus is on predicting only the fire front evolution with small regards to other domain properties, the simulation of Duane’s model runs faster than WRF.

$$ SR = w_W \cdot \text{Wind} + w_S \cdot \text{Slope} + w_A \cdot \text{Aspect} + w_F \cdot \text{Fuel} + w_{\text{Spp}} \cdot \text{SppFlam} \quad (2.18) $$

where $w_W, w_S, w_A, w_F, w_{\text{Spp}}$ are the weights obtained empirically, and $\text{SppFlam}$ refers to the species flammability.

$$ S_t = - \exp(-SR \cdot \text{Acceleration}) \cdot \text{Wind(StochasticSpread,1)} \quad (2.19) $$

Ferragut et al. [2] derive a model for fire spread in a two-dimensional domain that preserves the physical meaning of the fire phenomenon as described by the system (2.1) - (2.5). The energy equilibrium is well defined considering all the possible sources and sinks by the non-dimensional enthalpy model (2.20). However, the Navier Stokes equations (momentum balance) are not solved. The mass conservation and momentum balance equations are substituted by a convection model (2.21). This is based on the gradient of a scalar potential function of both the atmospheric wind database and the surface height function in the 3rd dimension. The moisture effect, related to the latent water heat, is also modeled by the use of a multivalued maximal monotone operator for the non-dimensional enthalpy [2].
\[
\frac{\partial e}{\partial t} + \mathbf{v} \cdot \nabla e - \nabla (k(u) \nabla u) + \alpha \cdot u = -\frac{\partial v}{\partial t} + R(u, x) \tag{2.20}
\]

\[
\mathbf{v} = \frac{\varepsilon}{c} (\nabla z \psi + e \nabla \bar{u} + \mathbf{v}_m) \tag{2.21}
\]

where \(-\nabla \cdot \psi = \left( \frac{\partial \psi}{\partial x_2}, \frac{\partial \psi}{\partial x_1} \right)\) \tag{2.22}

Furthermore, it is very important to emphasise that Simpson et al. [19] proved that a turbulent fire model that relies on convection gives the best prediction of fire spread in space-time. As shown in Figure 2.8, coupling convection to the fire dynamics allows the flame to spread naturally also transversally to the wind direction following the height isolines and considering the buoyancy effect.

![Figure 2.8](image)

**Figure 2.8.** Time of ignition (min) for each SFIRE model grid cell, and terrain height (m) line contours for (C70) coupled and (N70) non-coupled simulation [19].

It is very relevant also to model the heat absorbed by the water evaporation and the heat absorbed by the ‘passive’ pyrolyzing solid phase that contribute as sinks in the energy balance. Detaching the solid or liquid phase from the gas phase allows the model to predict an eventual re-ignition in those regions of the
domain where the flame was blown away by the wind but the fuel is not completely consumed. An exemplar formulation of the energy equation split between solid and gaseous phases is made by Weber [1]. The system’s unknowns are the non-dimensional solid and gas temperatures. The two states communicate through mutual convection modeled by the term \( h(u_g - u_s) \) in (2.23) and (2.24). The combustion, as stated in Section 2.1, takes place only at the gas phase, and here is expressed with the exponential reaction rate, also called Arrhenius Law [7].

\[
\begin{align*}
\text{solid phase:} & \quad \rho_s c_s \frac{\partial u_s}{\partial t} = k_s \frac{\partial^2 u_s}{\partial x^2} + h(u_g - u_s) \quad \text{Diff.} \\
\text{gas phase:} & \quad \rho_g c_g \frac{\partial u_g}{\partial t} = k_g \frac{\partial^2 u_g}{\partial x^2} - h(u_g - u_s) + A_k \exp \left( -\frac{\Gamma_g}{u_g} \right) \quad \text{Combustion}
\end{align*}
\]

Considering the Weber formulation (2.23) and (2.24), the fire will propagate only if (2.25) is valid, which means that the energy lost due to convection with the environment is smaller than the energy produced by combustion.

\[
\text{Fire spread condition [1]:} \quad h < \frac{A_k \Gamma_g}{u_g^2} \exp \left( -\frac{\Gamma_g}{u_g} \right)
\]

The most complete multiphase and multiscale fire model, that solves all the porosity scales shown in Figure 2.2 from the giga-scale to the micro-scale, is the one proposed by Guillaume and Margerit, [4] and [20]. This model considers also how the slope affects the flame spread due to radiation-related preheating as depicted in Figure 2.9. This model is quite complex and require further studies in order to be numerically implemented. Moreover, the actual simulations of this model are computationally quite time consuming, although the quality of the results is outstanding, as shown in Figure 2.10.
Some further general considerations about fire spread modelling are related to ignition time and opacity of the medium. As far as the *ignition time* is considered, even if the gaseous mixture exceeds the pyrolysis temperature, there will be a certain delay before the actual ignition takes place. This delay depends on the
partial pressure of the reactants in a continuous range between lean mixture and rich mixture [21]. Moreover, the opacity or optical thickness [8] of the medium has to be considered. It is the mean radiation penetration distance, a characteristic length scale that constraints the minimum grid size for numerical simulation. For example, the eventual presence of smoke smaller the optical thickness, that means that the flame radiation will not have effect largely far from the flame domain.

2.4. **NUMERICAL METHOD**

While trying to solve numerically any of the models shown in Section 2.3, the main obstacles one could encounter are related to the numerical stability of the solution, the domain discretization, the definition of initial and boundary conditions, the huge size of the required storage and the length of the required computational time. Furthermore, the fact that the highly complex fire phenomenon is described by a set of non-linear partial differential equations complicates the situation. A first attempt to simplify the problem could be to neglect some phenomena that, in the particular case, are not very manifest. However, in the most comprehensive general case, the main issues are explained below.

2.4.1. **NUMERICAL STABILITY**

To assess the stability of the numerical model, a first rule of thumb is to verify that certain convergence coefficients are defined in the stability range, although while using completely implicit methods these numbers lose validity since the model is no-more conditionally stable. These values constrain both the time and the space discretization.

For instance, the well-known Peclèt number, (2.26), relates the convective rate to the advection rate for fluid dynamics (refer to (2.2)), and the convergence condition for explicit numerical methods is to be smaller than 2 [22], preferably
around 1. \( U_\infty \) is the asymptotic wind velocity and \( \nu \) the kinematic viscosity of the fluid.

\[
P_{\infty} c = \left[ \frac{U_\infty}{\nu} \right] dx \leq 2 \quad (2.26)
\]

Therefore, the space mesh size could be constrained by (2.27).

\[
dx = \beta \frac{2\nu}{U_\infty} \quad with \quad 0 < \beta \leq 1 \quad (2.27)
\]

Similarly, the C.F.L. (Currant Friedric Lewy) number (2.28) is strictly related to the transport term. This value must be lower than 1 [22].

\[
CFL = \left[ \frac{U_\infty}{dx} \right] dt \leq 1 \quad (2.28)
\]

In the same way, the CFL number can constrain the maximum allowable time step of the simulation, as expressed in (2.29).

\[
dt = \alpha \frac{dx}{U_\infty} \quad with \quad 0 < \alpha \leq 1 \quad (2.29)
\]

Finally, another characteristic value that gives an idea of the possible fire dynamics is the Froude number (2.30) that relates the wind atmospheric velocity, \( U \), to the flame characteristics, such as maximum expected temperature, \( T_{\text{max}} \), and width of the intensely heated region, \( W_f \) [23]. If the Froude number is greater than 1 the atmosphere flow is not very affected by the flame presence, while, if it is far smaller than one the fire starts to trigger its own circulation at a fire-scale level.

\[
F_c = \left[ \frac{U_\infty^2}{g \frac{\langle \Delta T \rangle}{T_{\text{max}}} W_f} \right] \quad (2.30)
\]
Moreover, similarly to the Peclèt number, one could obtain the Prandtl number (2.31), that is defined between enthalpy transport rate and enthalpy diffusion rate, refer to (2.3).

\[
Pr = \frac{\rho \cdot c_p \cdot \|U_c\| \cdot dx}{k} \leq 1
\]  

(2.31)

In general, the numerical solution of each partial differential equation, if solved explicitly with a specific numerical method, is conditionally stable. This stability condition could be obtained, but its derivation is quite complex for the particular set of equations describing the fire dynamics.

The best way to avoid convergence problems, as suggested by Cox [7], is to use SIMPLE (semi-implicit) model for simulation of fire and all its variations: SIMPLEST, SIMPLER, SIMPLEC or PISO. An intermediate compromise between Euler or Heun’s explicit methods and the implicit ones is the Runge Kutta method. The most used is Runge Kutta of 4-th order that, if explicit, is still conditionally stable. This method is explained and applied to Model C in Section 3.5.

2.4.2. DISCRETIZATION

As previously stated in Section 2.4.1, the spatial and time discretization is strictly related to the numerical method used to simulate the fire model. The solution stability condition must be satisfied for explicit or semi-implicit methods.

Moreover, as far as the only spatial discretization is considered, some other considerations would ameliorate significantly the quality of the simulation. For instance, a fully adaptive mesh, such as the one shown in Figure 2.11 that follows the temperature gradient, would give the most accurate results almost free of numerical instability related to rounding errors [24]. So, for the accuracy this would be the best choice, but it would compromise the computational speed. In fact, the morphing mesh has to be calculated at each time step. Nevertheless, while using processors clusters, parallelization seems to be a solution to quicken
the simulation, which would be divided into two main processes: the thermo-fluid dynamics computation and adaptive mesh generation.

![Figure 2.11. Finite Element Method triangular mesh following the steepest gradient of temperature field [24].](image)

Moreover, while using explicit methods, many researchers, such as Seibold [25] and Gautam [26], suggest to use staggered grid. This consist in defining different mesh for the scalar field and for the vector fields. For example, in the combustion case, the scalars (pressure, density, temperature, etc.) are calculated on a centred grid, while the velocity components \( u, v \) and \( w \) are defined on three shifted (by half of the space discretization) grids in \( x, y \) and \( z \) direction respectively. The explicit method based on a staggered grid seems to be more stable compared with the fully-centred grid, but it adds some complications in finding the best way to project the fields from one mesh to the other. Averaging the values is the most convenient choice, but cubic splines is probably the best compromise between complexity and quality. The staggered grid is applied to solve Models A and Model B described in Sections 3.3 and 3.4.

### 2.4.3. Initial and Boundary Conditions

Assigning the boundary conditions and initial conditions to the complex fire b.i.v.p. (boundary initial value problem) is a very delicate matter. In particular,
the temperature initial condition has to be defined in order not to trigger any numerical artifacts. Mercer and Weber [27] suggest to use a smooth gaussian distribution (2.32) around the initial ignition point \((x_{ig}, y_{ig})\) as illustrated in Figure 2.12a. The pick value is chosen arbitrarily higher than the pyrolysis temperature, \(T_p\), that is supposed to be the temperature at which the ignition occurs. Therefore, the initial flame domain is an ellipsoidal shown in Figure 2.12b. In order to define an ignition area of a different geometry, one could sum up different gaussian functions to generate the wanted initial condition over the environment temperature, \(T_0\).

\[
T(x, y, 0) = T_o + (T_o + T_p + \Delta T) \cdot \exp \left( -\frac{r(x, y)^2}{2\sigma_r^2} \right) \tag{2.32}
\]

where

\[
r(x, y)^2 = (x - x_{ig})^2 + (y - y_{ig})^2 \tag{2.33}
\]

Figure 2.12. Temperature initial condition: a) Gaussian distribution; b) top-view ignited circular area in a 2D domain.

### 2.4.4. Memory Constrain

Discretizing backward in time a linear partial differential equation results in solving a linear system. The algorithms to be predilected is the Cholesky triangular factorization, as suggested by Karypis [28]. It is a sparser version of the Kronecker matrix of the system. This factorization results in an optimization of the numerical model that requires less storage and a shorter time to solve the
linear system. Of course, the situation becomes quite complicated and requires formulating other algorithms when dealing with non-linear differential equations.

Nevertheless, improving the modality of storage of the resulting sparse matrices is crucial for formulating a quite performing numerical method. New ways of operating with these artificial structures, free of zeros, must be defined in order to allow a faster data manipulation [29].


3. METHODOLOGY

In this section three attempts to model and simulate the fire spread are meticulously described:

- Model A (Section 3.3): It is described by a system of nine equations in nine unknowns which are density, pressure, velocity component in x direction, velocity component in y direction, temperature, fuel mass fraction, mixture molar mass, mixture specific heat capacity and specific combustion enthalpy.

- Model B (Section 3.4): It is similar to Model A with the exception that here the density is direct function of the temperature while the pressure is obtained by imposing the mass conservation on the momentum balance equation using the Chorin pressure projection method.

- Model C (Section 3.5): It is the best model compared with the previous one in terms of solution stability and qualitative flame spreading. However, here the strong assumption is that the combustion occurs at constant pressure. So, the system consists of eight equations in eight unknowns.

The Model A and Model B equations are solved sequentially, even if they are all coupled, while the equations in Model C are solved simultaneously using a 4th order Runge Kutta algorithm. In all three cases, the numerical method is explicit in time. A small exception could be found in Model A and B, where the diffusion term in the momentum balance equations is solved backward in time.

The main problem, shared between all the presented models, is the dimensionality reduction (see Section 3.2). In fact, the simulation domain is two-dimensional in space, while the actual fire dynamics belong to a three-dimensional space.

3.1. GENERAL ASSUMPTIONS

The combustion reaction considered in the Matlab scripts in Appendix A: Script 1, Appendix B: Script 2 and Appendix C: Script 3 respectively for the Model A, B and C is the methane combustion in air (3.1), however the eventual user could consider a
different exothermic reaction by only changing the molecules properties and the stoichiometric coefficients.

\[ \text{CH}_4 + 2\text{O}_2 \rightarrow \text{CO}_2 + 2\text{H}_2\text{O} \]  \hspace{1cm} (3.1)

Important to note that the stoichiometric coefficients in all the scripts are stored negative if they appear in the left-hand-side of the equation. The coefficient vector, \( \theta \), describing the (3.1) reaction is then (3.2).

\[ \theta = (-1 \hspace{0.5cm} -2 \hspace{0.5cm} 1 \hspace{0.5cm} 2) \]  \hspace{1cm} (3.2)

Each chemical species has also its specific heat capacity, \( c_{pa} \), a reference molar enthalpy of formation, \( H_{a,\text{ref}} \), a molar mass, \( M_a \), and an initial mole fraction, \( X_a \). The specific heat capacity at constant pressure is assumed to be constant even if, for certain gaseous species, it varies quite consistently with temperature, Figure 3.1. The considered \( c_{pa} \) values could be found in [30]. Some researchers, such as Dr. Jarunghammachot et al. [31] and Dr. Shaukat [6], suggest to use a polynomial approximation of the \( c_{pa} \) valid within the specific expected temperature range. This could be considered for future development of Model C.

![Figure 3.1. Specific heat capacity at constant pressure vs. temperature for various gases.](image)

Furthermore, in all the following models, the chemical species transport and diffusion is neglected in order to reduce the complexity. This consists in asserting that the mixture is defined locally and it will not affect the neighbourhood mixtures.
Therefore, the only dynamic related to the chemical species is their consumption or formation via combustion. This can be expressed by the Arrhenius Law \([21]\) that gives the fuel consumption rate, \(r(3.3)\). When the mixture exceeds the activation energy, \(E_a\), or the activation temperature, \(T_a\), it reaches the autoignition condition. If the activation temperature is higher the reaction rate is slower, and vice versa. The particular empirical formulation and values for methane combustion could be found in Ragland et al. \([32]\).

\[
\frac{\partial Y}{\partial t} = \delta_{(T,Y)}^+ \cdot A_r \cdot T \cdot Y \exp\left( -\frac{E_a}{RT} \right) = -\delta_{(T,Y)}^+ \cdot A_r \cdot T \cdot Y \exp\left( -\frac{T_a}{T} \right) \tag{3.3}
\]

The Kronecker delta, \(\delta_{(T,Y)}^+ (3.4)\), assures that the mixture ignites when the temperature exceeds the pyrolysis temperature, \(T_p\), while the fuel mass fraction is larger than the extinction one, \(Y_e\) (which consists of an extremely lean mixture where the probability of combining fuel molecules with oxidant is very low).

\[
\delta_{(T,Y)}^+ = \begin{cases} 
1 & \text{if } T \geq T_p, \ Y \geq Y_e \\
0 & \text{else}
\end{cases}
\tag{3.4}
\]

<table>
<thead>
<tr>
<th>ID</th>
<th>Hypothesis</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Methane combustion in air.</td>
<td>(CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O)</td>
</tr>
<tr>
<td>2</td>
<td>Constant specific heat capacity.</td>
<td>(c_{pa} = \text{Const.})</td>
</tr>
<tr>
<td>3</td>
<td>Nor transport neither diffusion of the chemical species, only Arrhenius reaction rate.</td>
<td>(r = -\delta_{(T,Y)}^+ \cdot A_r \cdot T \cdot Y \exp\left( -\frac{T_a}{T} \right))</td>
</tr>
<tr>
<td>4</td>
<td>Fictional ‘switch’ that regulates the mixture ignition.</td>
<td>(\delta_{(T,Y)}^+(0,1))</td>
</tr>
<tr>
<td>5</td>
<td>Pseudo-3D convection in the energy balance. (Section 3.2)</td>
<td>(\text{Conv.3D} = -C_a (T - T_o) \leq 0)</td>
</tr>
<tr>
<td>6</td>
<td>Space discretisation larger than the optical thickness of the medium.</td>
<td>(dx \text{ and } dy &gt; t_{RAD})</td>
</tr>
<tr>
<td>7</td>
<td>Pseudo-3D radiation in the energy balance. (Section 3.2.1)</td>
<td>(\frac{\partial(\rho\phi)}{\partial t} = -\frac{\sigma_T\phi}{l_c} (T^4 - T_0^4) \leq 0)</td>
</tr>
</tbody>
</table>

Table 3.1. Main assumptions shared by Model A, B and C.
For clarity, in Table 3.1 are briefly listed the main hypothesis shared by the three models. In the relative sections, eventual additional assumptions are thoroughly explained.

### 3.2. DIMENSIONALITY REDUCTION

Reducing the dimensionality of the problem to two spatial dimensions generates some problems. For instance, some cross interaction will be missed, such as shear stresses in fluid dynamics. Some purely three-dimensional phenomenon must be reduced to 2D, otherwise, if neglected, could cause some issues that must be assessed scrupulously.

A simple example of dimensionality reduction expression is modelling the energy convection in the 3rd dimension. Many modelers, such as Ferragut [2] and Kim [29], model the convection linearly (3.5) by means of a convection coefficient, $C_a$. This quantity will always be equal or smaller than zero.

\[
\text{Conv.3D} = -C_a(T - T_o) \leq 0 \quad (3.5)
\]

#### 3.2.1. RADIATION MODELLING

The radiation is a volumetric phenomenon proportional to the 4th power of the temperature of the source [8]. The hemispherical radiation power emitted by a finite radiative volume is expressed by (3.6) where one could recognise the Boltzmann constant, $\sigma$, the emittance, $\varepsilon$, and the radiation aspect, $\phi$.

\[
\frac{\partial q_{\text{RAD}}}{\partial t} = q_{\text{RAD}} = \sigma T^4 \phi 
\]

(3.6)

In order to formulate the radiation in 2D one should divide it into two contributions: the auto-radiation in the 2D domain and the radiation dispersing energy outward from the domain in the z direction, or pseudo-3D radiation. As far as the first kind is considered, one should formulate at first the radiation flux passing through a certain control surface, as represented in Figure 3.2.
The 4\textsuperscript{th} power of the temperature in the B point could be simplified neglecting some small differentials, as shown in (3.9).

\[
\left(T + \frac{\partial T}{\partial x} \, dx\right)^4 = T^4 + 4T^3 \frac{\partial T}{\partial x} \, dx + 6T^2 \left(\frac{\partial T}{\partial x}\right)^2 \, dx + 4T \left(\frac{\partial T}{\partial x}\right)^3 \, dx + \left(\frac{\partial T}{\partial x}\right)^4 \, dx \tag{3.9}
\]

All considered one could obtain a formulation of the net heat flux in the x direction (3.10) just subtracting one flux to the other bearing in mind the right direction of the unit vector \( \hat{x} = (1 \quad 0 \quad 0) \).

\[
\dot{q}_{Rx} = \dot{q}_{RAD} \big|_x = \dot{q}_A - \dot{q}_B = -\sigma \varepsilon \phi T^4 \frac{\partial T}{\partial x} \, dx \tag{3.10}
\]

It is good practice to check that the dimension of the newly modelled quantity matches the expected one. In case of heat flux, it must be power, \( W \), divided by area, \( m^2 \), as shown in (3.11).

\[
\left[ \dot{q}_{RAD} \right]_x = \left[ -\sigma \varepsilon \phi T^4 \frac{\partial T}{\partial x} \right] \, dx = \left( \frac{W}{m^2 K^2} \right) \left( 1 \right) \left( K \right) \left( \frac{K}{m} \right) m = \left[ \frac{W}{m^2} \right] \tag{3.11}
\]

In order to consider the energy variation induced by radiation on a finite control volume of dimension \( dx \times dy \) one could consider the Figure 3.3. The energy change due to radiation heat flux through the \( W \)-face with respect to the reference point in the centre of the control volume is (3.12) and similarly, through the \( E \)-face is (3.13).
Figure 3.3. 2D radiation across a control volume.

\[ \left( \dot{q}_{Rs} - \frac{\partial \dot{q}_{Rs}}{\partial x} \frac{dx}{2} \right) dydz \] \hspace{1cm} (3.12)

\[ \left( \dot{q}_{Rs} + \frac{\partial \dot{q}_{Rs}}{\partial x} \frac{dx}{2} \right) dydz \] \hspace{1cm} (3.13)

So, the change in energy due to radiation in only x direction is the sum of the contributions of both surfaces (3.14).

\[ \Delta \dot{q}_{Rs} = -\frac{\partial \dot{q}_{Rs}}{\partial x} dx dydz = 4\sigma \varepsilon dx \frac{\partial}{\partial x} \left( T^3 \frac{\partial T}{\partial x} \right) dx dydz \] \hspace{1cm} (3.14)

The resulting expression of how the volumetric enthalpy changes in time due to two-dimensional radiation is (3.15) and in literature can be found similar derivation, such as Dr. Kim [29]. This formulation is valid only if the absorption length, also called optical thickness (t_{RAD}), is smaller or equal to the smallest cell size of the chosen spatial discretization (dx or dy). If the optical thickness is longer, it means that some radiation flux far from the origin will been neglected.

\[ \frac{\partial (\rho h)}{\partial t}_{RAD, 2D} = -\frac{\partial \dot{q}_{Rs}}{\partial x} - \frac{\partial \dot{q}_{Rs}}{\partial y} = 4\sigma \varepsilon \phi \left[ dx \frac{\partial}{\partial x} \left( T^3 \frac{\partial T}{\partial x} \right) + dy \frac{\partial}{\partial y} \left( T^3 \frac{\partial T}{\partial y} \right) \right] \] \hspace{1cm} (3.15)

The radiation in the 3rd direction is treated as a sink because it is assumed that in the embracing environment there is no flame. It is modelled similarly to the convection model in the 3rd dimension (3.5) with the exception that now the radiation is function of the 4th power of the temperature. The resulting adopted
model of 3D radiation is expressed in (3.16), where $l_z$ is the absorption length in $z$ direction and $T_0$ is the ambient temperature.

$$\frac{\partial (\rho h)}{\partial t}_{RAD,3D} = - \frac{\sigma \varepsilon \phi}{l_z} (T^4 - T_0^4) \leq 0$$  \hspace{1cm} (3.16)$$

### 3.3. Model A

This first two-dimensional model attempted to simulate directly the whole set of equations describing also the density dynamics. This consists of a mass balance equation (3.17); two momentum balance equations in $x$ direction (3.18) and $y$ direction (3.19); Arrhenius combustion rate equation (3.20); energy balance equation (3.21) and other closure equations regarding pressure (3.22), molar fraction of each chemical species (3.23), mixture molar mass (3.24), mixture specific heat capacity (3.25) and specific combustion enthalpy (3.26).

$$\frac{\partial (\rho)}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0$$  \hspace{1cm} (3.17)$$

$$\frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2)}{\partial x} + \frac{\partial (\rho uv)}{\partial y} = - \frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$  \hspace{1cm} (3.18)$$

$$\frac{\partial (\rho v)}{\partial t} + \frac{\partial (\rho uv)}{\partial x} + \frac{\partial (\rho v^2)}{\partial y} = - \frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$$  \hspace{1cm} (3.19)$$

$$r = \frac{\partial Y}{\partial t} = - \delta^{(i,j)}_{(x,y)} A_{ij} Y \exp \left( - \frac{T_r}{T} \right)$$  \hspace{1cm} (3.20)$$

$$\frac{\partial (\rho c_p T)}{\partial t} + \rho h_r + \frac{\partial (\rho u c_p T)}{\partial x} + \frac{\partial (\rho v c_p T)}{\partial y} + \frac{\partial (\rho u h_r)}{\partial x} + \frac{\partial (\rho v h_r)}{\partial y} = \frac{\partial p}{\partial t} + \cdots$$

$$+ k \frac{\partial}{\partial x} \left( \frac{1}{c_p} \frac{\partial c_p}{\partial x} \right) + k \frac{\partial}{\partial y} \left( \frac{1}{c_p} \frac{\partial c_p}{\partial y} \right) + k \frac{\partial}{\partial x} \left( \frac{1}{c_p} \frac{\partial h_r}{\partial x} \right) + k \frac{\partial}{\partial y} \left( \frac{1}{c_p} \frac{\partial h_r}{\partial y} \right) + \cdots$$  \hspace{1cm} (3.21)$$

$$4 \sigma \varepsilon \phi \left[ dx \frac{\partial}{\partial x} \left( T^4 \frac{\partial T}{\partial x} \right) + dy \frac{\partial}{\partial y} \left( T^4 \frac{\partial T}{\partial y} \right) \right] - C_a (T - T_a)$$
\[ p = \frac{\rho RT}{M} \]  \hspace{1cm} (3.22)

\[ \frac{\partial X_a}{\partial t} = -\frac{\theta_{a}}{\theta_{\text{fuel}}} \cdot \frac{M}{M_{\text{fuel}}} \cdot r \]  \hspace{1cm} (3.23)

\[ M = \sum_a X_a M_a \]  \hspace{1cm} (3.24)

\[ c_p = \sum_a X_a \frac{M_a}{M} c_{p_a} \]  \hspace{1cm} (3.25)

\[ h_c = \frac{mH(T)}{M} = -\frac{\sum a \theta_a \cdot H_a(T)}{M} = -\frac{\sum a \theta_a \cdot (H_{a,\text{ref}} + M_a \cdot c_{p_a} \cdot (T - T_{\text{ref}}))}{M} \]  \hspace{1cm} (3.26)

The system’s primary unknowns are the velocity components, \( u \) and \( v \), the pressure, \( p \), the temperature, \( T \), and the density, \( \rho \); while the other variables (mass fractions, molar mass, heat capacity and combustion enthalpy) are secondary unknowns derived algebraically by the respective closure equations. Model A is solved sequentially as described by the diagram in Figure 3.4 using Finite Difference Method (only centre differences are used) and respecting the set of boundary conditions listed in Table 3.2.

<table>
<thead>
<tr>
<th>( U ) (( nx \times ny ))</th>
<th>( V ) (( nx \times ny-1 ))</th>
<th>( P ) (( nx \times ny ))</th>
<th>( T ) (( nx \times ny ))</th>
<th>( \rho ) (( nx \times ny ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>Dirichlet</td>
<td>Dirichlet</td>
<td>Neumann</td>
<td>Dirichlet</td>
</tr>
<tr>
<td></td>
<td>( u = U_0 )</td>
<td>( v = V_0 )</td>
<td>( \frac{\partial p}{\partial x} \big</td>
<td>_{\text{W}} = 0 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \rho = \rho_{\text{amb}} )</td>
</tr>
<tr>
<td>South</td>
<td>Dirichlet</td>
<td>Dirichlet</td>
<td>Neumann</td>
<td>Dirichlet</td>
</tr>
<tr>
<td></td>
<td>( u = U_0 )</td>
<td>( v = V_0 )</td>
<td>( \frac{\partial p}{\partial y} \big</td>
<td>_{\text{S}} = 0 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \rho = \rho_{\text{amb}} )</td>
</tr>
<tr>
<td>North</td>
<td>Neumann</td>
<td>Neumann</td>
<td>Dirichlet</td>
<td>Dirichlet</td>
</tr>
<tr>
<td></td>
<td>( \frac{\partial u}{\partial y} \big</td>
<td>_{\text{N}} = 0 )</td>
<td>( \frac{\partial v}{\partial y} \big</td>
<td>_{\text{N}} = 0 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \rho = \rho_{\text{amb}} )</td>
</tr>
<tr>
<td>East</td>
<td>Neumann</td>
<td>Neumann</td>
<td>Dirichlet</td>
<td>Dirichlet</td>
</tr>
<tr>
<td></td>
<td>( \frac{\partial u}{\partial x} \big</td>
<td>_{\text{N}} = 0 )</td>
<td>( \frac{\partial v}{\partial x} \big</td>
<td>_{\text{N}} = 0 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \rho = \rho_{\text{amb}} )</td>
</tr>
</tbody>
</table>

Table 3.2. Boundary conditions for velocities, pressure (for Model B), temperature and density (for Model A).
Note however that the boundary condition for the pressure is not applicable in Model A, since the pressure is calculated by a closure equation (3.22), while it is applicable for Model B, where the Chorin projection method is used (Section 3.4). Vice versa, the boundary condition for the density is valid for Model A, while it is not applicable in Model B because it is calculated by means of the closure equation (3.27). The Matlab code for Model A can be found in Appendix A: Script 1.

Figure 3.4. Representation of the sequential solution adopted by Model A.

An example of finite difference formulation can be found in Appendix D. This is based on the staggered grid depicted in Figure 3.5. Note that, considering $nx$ the number of cells in $x$ direction and $ny$ the number of cells in $y$ direction, the unknown matrices have the following sizes:

- All the scalars (temperature, pressure, density, etc.): $nx \times ny$
- Velocity component in $x$ direction, $u$: $(nx-1) \times ny$
- Velocity component in $y$ direction, $v$: $nx \times (ny-1)$
Compared with Model A, this new version tries to overcome the density problem occurred in the previous one. Instead of considering the density dynamics in the mass balance, the density follows majorly the temperature dynamics by means of the closure equation (3.27). The pressure is obtained by Chorin projection method. This method is applicable only if the density change rate is negligible, that means that the mass conservation equation is reduced to the only divergence of the wind velocity (3.28). Therefore, the new Model B is described by the (3.27) - (3.32) system and by the closure equations from (3.23) to (3.26) for the chemical species.
mass fractions, mixture molar mass, specific heat capacity and specific combustion enthalpy.

\[ \rho = \frac{pM}{RT} \quad (3.27) \]

\[ \frac{\partial(u)}{\partial x} + \frac{\partial(v)}{\partial y} = 0 \quad (3.28) \]

\[ \frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} = -\frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (3.29) \]

\[ \frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho v^2)}{\partial y} = -\frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (3.30) \]

\[ r = \frac{\partial Y}{\partial t} = -\delta_{(x,y)} \Lambda T Y \exp \left( -\frac{T_r}{T} \right) \quad (3.31) \]

Also, this system is solved sequentially following the diagram in Figure 3.6, using Finite Difference Method (only centre differences are used) and respecting the boundary conditions in Table 3.2. The relative Matlab script could be found in Appendix B: Script 2. Model B uses the same staggered grid as Model A, shown in Figure 3.5.
Some explanations must be provided about the application of Chorin projection method. So, how the Navier Stokes equations are numerically solved on the...
staggered grid is briefly explained from (3.33) to (3.41). The process is divided in three steps:

1. the transport momentum is added explicitly to the old velocity field in (3.33) and (3.34);
2. the diffusion contribution is calculated backward in time (3.35) - (3.36), hence the use of the Kronecker matrix to solve the resulting system;
3. finally, in order to assure the mass conservation, represented by the divergence of the future timestep velocity (3.40), using the Chorin projection method the velocity field is updated to the new one.

The *Cholesky triangular decomposition* is used to solve both the diffusion problem at point (2) and the Poisson equation for the pressure (3.41).

Therefore, as it was stated, the transport system updates from \(u^{(n)}\), old time, to \(u^*\) explicitly.

\[
\frac{u^* - u^n}{dt} = -\frac{\partial (u^n u^n)}{\partial x} - \frac{\partial (u^n v^n)}{\partial y} \tag{3.33}
\]

\[
\frac{v^* - v^n}{dt} = -\frac{\partial (u^n v^n)}{\partial x} - \frac{\partial (v^n v^n)}{\partial y} \tag{3.34}
\]

The diffusion system is solved implicitly from \(u^*\) to \(u^{**}\). Note that \(\nu\) is the kinematic viscosity \(\mu/\rho\).

\[
\frac{u^{**} - u^*}{dt} = \nu \left( \frac{\partial^2 u^{**}}{\partial x^2} + \frac{\partial^2 u^{**}}{\partial y^2} \right) \tag{3.35}
\]

\[
\frac{v^{**} - v^*}{dt} = \nu \left( \frac{\partial^2 v^{**}}{\partial x^2} + \frac{\partial^2 v^{**}}{\partial y^2} \right) \tag{3.36}
\]

Finally, from \(u^{**}\) one has to update to the actual velocity field, \(u^{(n+1)}\), applying the pressure correction. Note that \(\tilde{p}\) is the specific pressure \(p/\rho\). 

49
\[ u^{(n+1)} - u^{**} = - \frac{\partial \widetilde{p}^{(n+1)}}{\partial x} \quad \text{(3.37)} \]
\[ v^{(n+1)} - v^{**} = - \frac{\partial \widetilde{p}^{(n+1)}}{\partial y} \quad \text{(3.38)} \]
\[ \frac{v^{n+1}}{dt} - \frac{v^{**}}{dt} = - \nabla(\widetilde{p}^{(n+1)}) \quad \text{(3.39)} \]
with \( \nabla(\widetilde{v}^{(n+1)}) = 0 \) \( \text{(3.40)} \)

The calculated velocity field must be corrected, in order to assure the mass conservation, using the pressure gradient. So, applying the divergence operator to the (3.39) and getting rid of the (3.40) term that comes up, the resulting equation to be solved in \( \widetilde{p} \) is the Poisson equation (3.41).

\[ \frac{\nabla(\widetilde{v}^{**})}{dt} = \nabla^2(\widetilde{p}^{(n+1)}) \quad \text{(3.41)} \]

### 3.5. Model C

The aim of Model C is to overcome the stability problem occurred while solving Model B. The sequential approach was not giving a numerically stable solution within the available memory limit of the used calculator. Therefore, the Runge Kutta of 4th order method has been chosen for solving the stability problem (Section 3.5.1). Since still explicit, this numerical method is still conditionally stable, but, since there’s no more need of Kronecker matrices, the discretisation can also be further refined. This approach is no more sequential, but the coupled equations can be simultaneously solved. The Runge Kutta method, however, requires some changes of variable, since the time derivative in the momentum equation and in the energy balance are mixed with the density and with the specific heat capacity of the mixture, which are not constant in time. The resulting changes of variable are the modified velocity components \( \widetilde{u} \) (3.42) and \( \widetilde{v} \) (3.43) and the modified temperature \( \widetilde{T} \) (3.44).
\[ \tilde{u} = \rho \cdot u \quad (3.42) \]
\[ \tilde{v} = \rho \cdot v \quad (3.43) \]
\[ \tilde{T} = \rho \cdot c_p \cdot T \quad (3.44) \]

Moreover, the flame is assumed to be isobaric, that means the pressure is supposed to remain constant while the density will change inversely proportionally to the temperature (3.49). Therefore, the pressure gradient terms in the momentum balance equations (3.45) and (3.46) and the pressure change rate in the energy equation (3.47) can be neglected.

\[
\frac{\partial \tilde{u}}{\partial t} = -\frac{\partial (u^2 / \rho)}{\partial x} - \frac{\partial (\tilde{u} \tilde{v} / \rho)}{\partial y} + \mu \left( \frac{\partial^2 (u / \rho)}{\partial x^2} + \frac{\partial^2 (\tilde{u} / \rho)}{\partial y^2} \right) \quad (3.45)
\]
\[
\frac{\partial \tilde{v}}{\partial t} = -\frac{\partial (\tilde{u} \tilde{v} / \rho)}{\partial x} - \frac{\partial (v^2 / \rho)}{\partial y} + \mu \left( \frac{\partial^2 (\tilde{v} / \rho)}{\partial x^2} + \frac{\partial^2 (v / \rho)}{\partial y^2} \right) \quad (3.46)
\]
\[
\frac{\partial \tilde{T}}{\partial t} = -\rho h_r - \frac{\partial (u \tilde{T} / \rho)}{\partial x} - \frac{\partial (\tilde{v} \tilde{T} / \rho)}{\partial y} - \frac{\partial (\tilde{u} h_r)}{\partial x} - \frac{\partial (\tilde{v} h_r)}{\partial y} + ... + k \frac{\partial}{\partial x} \left( \frac{1}{c_p} \frac{\partial (\tilde{T} / \rho)}{\partial x} \right) + k \frac{\partial}{\partial y} \left( \frac{1}{c_p} \frac{\partial (\tilde{T} / \rho)}{\partial y} \right) + k \frac{\partial}{\partial x} \left( \frac{1}{c_p} \frac{\partial h_r}{\partial x} \right) + k \frac{\partial}{\partial y} \left( \frac{1}{c_p} \frac{\partial h_r}{\partial y} \right) + ... \quad (3.47)
\]
\[-C_a \left( \frac{\tilde{T}}{(\rho c_p)} - T_a \right) \]
\[
\frac{\partial \mathbf{Y}}{\partial t} = -\delta_{(T,Y)} A_r \left( \hat{T} \right) Y \exp \left( -\frac{\rho c_p T_a}{\hat{T}} \right) \quad (3.48)
\]
\[
\rho = \frac{pM}{RT} \quad (3.49)
\]
3.5.1. RUNGE KUTTA FORMULATION

As far a general i.v.p (3.50) is considered, where the function \( f \) could be also non-linear, the aim of Runge Kutta method is to find an approximated solution, \( w_n \) (3.52).

System to be solved:

\[
\begin{align*}
\frac{\partial g}{\partial t} &= f(t, g) \\
g(t_0) &= \alpha
\end{align*}
\]  \hspace{1cm} (3.50)

Time discretization:

\[ t_n = t_0 + n \cdot dt \]  \hspace{1cm} (3.51)

Approximate solution:

\[ w_n \approx g(t_n) \]  \hspace{1cm} (3.52)

Considering the time discretization (3.51), the values \( k_1, k_2, k_3 \) and \( k_4 \) represent some rough estimations of the possible solution.

\[ w_0 = \alpha \]  \hspace{1cm} (3.53)

\[ k_1 = dt \cdot f(t_i, w_i) \]  \hspace{1cm} (3.54)

\[ k_2 = dt \cdot f\left(t_i + \frac{dt}{2}, w_i + \frac{k_1}{2}\right) \]  \hspace{1cm} (3.55)

\[ k_3 = dt \cdot f\left(t_i + \frac{dt}{2}, w_i + \frac{k_2}{2}\right) \]  \hspace{1cm} (3.56)

\[ k_4 = dt \cdot f(t_i + dt, w_i + k_3) \]  \hspace{1cm} (3.57)

Eventually, the approximated solution (3.58) is the previous time step solution value corrected by a linear combination of the intermediate time step values \( k_2 \) and \( k_3 \) and the entire time step predictions \( k_1 \) and \( k_4 \).

\[ w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \]  \hspace{1cm} (3.58)

In the Model C case, the main equations involved are (3.59) - (3.63), and the relative state variable, \( S \), is calculated at once at each time step (3.64). In (3.64) one could notice the actual Runge Kutta procedure, from (3.53) to (3.58), adapted for the solution of Model C. The only difference is that now the equations in the system are coupled, so the coefficients \( k_1, k_2, k_3 \) and \( k_4 \), newly called \( a, b, c \) and \( d \) are vectors which components are reciprocally dependant.
For instance, \( b_4 \) (temperature) depends on all the component of the vector coefficient \( a \), as suggested by equation (3.61)

\[
\frac{\partial \theta}{\partial t} = F_\theta \left( \rho, u, v, p \right)
\]

(3.59)

\[
\frac{\partial \theta}{\partial t} = F_\theta \left( \rho, u, v, p \right)
\]

(3.60)

\[
\frac{\partial \theta}{\partial t} = F_T \left( \rho, u, v, \frac{\partial p}{\partial t}, T, X_f \right)
\]

(3.61)

\[
\frac{\partial \theta}{\partial t} = F_X \left( \rho, X_f, T \right)
\]

(3.62)

\[
\rho = \frac{pM}{RT}
\]

(3.63)

\[
S^{(n)} = \begin{bmatrix}
\rho \\
u \\
v \\
T \\
X
\end{bmatrix} \Rightarrow a = \begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5
\end{bmatrix} \Rightarrow b(a) \Rightarrow c(b) \Rightarrow d(c) \Rightarrow S^{(n+1)} = f \left( S^{(n)}, a, b, c, d \right)
\]

(3.64)

‘Knitting’ this algorithm is quite complex, as it could be appreciated in Appendix C: Script 3. Note that the staggered grid is no more used; for Model C, all the unknowns are estimated in the centre of each cell.
4. RESULTS

The following results are the output of the simulation of only Model C, because it is the only one which numerical solution is stable thanks to the use of Runge Kutta of 4th order. The main output are however defined in the same way for all the models:

- Flame domain spread and zero-fuel domain formation which evolution history has been saved in both a video.avi file and in an animated figure.gif (Figure 4.1).
- Two-dimensional temperature field which history has been saved in both .avi and .gif files. In Figure 4.3 the frame at 8s from the ignition is illustrated.
- The temperature profile in x direction at half domain. The section plane passes exactly by the ignition point. The data are saved in the matrix $T_{\text{store}}$. Moreover, also in this case two animations .avi and .gif are produced. Some examples are shown in Figure 4.4 and in Figure 4.7.
- Two-dimensional density field which evolution history has been saved in both a .avi and .gif files. In Figure 4.5 the frame at 8s from the ignition is illustrated.
- Evolution in time of some characteristic properties at the ignition point located in $(x=2m, y=2.5m)$ is stored in a Matlab figure.fig. In particular the combustion power, the temperature, the fuel mole fraction and the oxidizer mole fraction are of pivotal interest and are shown in Figure 4.6.

The discretization of Model A and B is coarser than the one of Model C, as shown in Table 4.1, because the 32Gb available RAM of the used computer were not enough for storing the large sparse Kronecker matrices used to solve backward in time the momentum diffusion.

<table>
<thead>
<tr>
<th>Model</th>
<th>$dt$ [×10^{-2}s]</th>
<th>$dx$ [×10^{-2}m]</th>
<th>$dy$ [×10^{-2}m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2.5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2.5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>2.24</td>
<td>1.6</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 4.1. Space and time discretization of models A, B and C.
In Figure 4.2 the flame spread is shown at different times from the ignition. As predicted, the flame evolves in a ring shape, since it starts extinguishing from the centre.
The highest temperature, approximately 800 K, is found inside the developed ring flame next to the outer flame boundary, while the temperature gradient is smoother inside the ring, Figure 4.3.

Figure 4.3. Temperature field on a 25m² domain at 8 s, Model C.
Figure 4.4 is quite self-explanatory: whenever the mixture temperature exceeds the pyrolysis temperature, $T_p$, it ignites starting to consume fuel and oxidizer. Figure 4.4, in particular at 8s from the ignition, shows the developed flame in a ring shape. In fact, one could imagine a flame domain toroid around the original extinguished ignition point.

![Temperature profile (half domain section)](image)

Figure 4.4. Temperature profile at half domain section in x direction (ignition point at $x=2m$), Model C.
An exemplar density field representation could be found in Figure 4.5, which correspond to the state at 8s. The darker the colour is, the smaller the density of the mixture is. In particular, inside the flame domain, since the temperatures are the highest, the density is the lowest. In the remaining regions of the simulated domain, the density remains approximately at the constant value 1.16 kg/m$^3$. As one could notice it is lower than the air density, which at sea level and standard condition is 1.225 kg/m$^3$. This because the fluid considered is not air, but it is a mixture of air with methane instead. In fact, the methane’s density is 0.656 kg/m$^3$.

![Density field on a 25m$^2$ domain at 8 s, Model C.](image)

Figure 4.5. Density field on a 25m$^2$ domain at 8 s, Model C.
In Figure 4.6 some crucial quantities are illustrated in function of time in the centre of the flame. In particular the focus is on the combustion power generated per unit area, $\dot{Q}$, and the temperature. It is interesting to observe that the maximum pick temperature, $T_{\text{MAX}} = 800 K$, has a delay with respect to the pick of produced combustion energy, $\dot{Q}_{\text{MAX}} = 3.2 \times 10^5 \text{ W/m}^2$. Moreover, the consumption in time of both fuel and oxygen mole fractions is depicted. The extinction fuel molar fraction is found to be approximately 0.02.

![Figure 4.6](image)

Figure 4.6. Evolution in time of some characteristic properties at the ignition point (x=2m, y=2.5m) with Model C: a) combustion power, b) temperature, c) fuel mole fraction and d) oxidizer mole fraction. Extinction time: $t_e=5.25$s.
Some data from $T_{\text{store}}$ are shown in Figure 4.7, that represents the temperature profiles at half domain related to the flame spread in time.

![Temperature profile in x direction at half domain passing through the ignition point (x=2m, y=2.5m) at different time from the ignition (data from $T_{\text{store}}$, Model C).](image)

Figure 4.7. Temperature profile in x direction at half domain passing through the ignition point (x=2m, y=2.5m) at different time from the ignition (data from $T_{\text{store}}$, Model C).
5. Discussion

As it could be appreciated in the results, Section 4, Model C succeeded in modelling a two-dimensional fire spread at constant pressure. All the unknowns’ values, and in some cases more roughly orders of magnitude, are quantitatively reasonable. Moreover, the ring-shaped fire spread, Figure 4.1 and Figure 4.2, is the actual geometry that was supposed to be.

The Model C could model also the contribution of the atmospheric wind, but the simulated energy transport related to a windy condition is extremely high leading to numerical instability that could be strictly related to a mass unbalance. The presented model could be improved in this extent probably solving the “density paradox” (Section 5.1) or modelling the fire spread in a three-dimensional environment overcoming many issues related to the reduction to a 2D environment.

In Section 5.1 and Section 5.2 it is explained why the Model A and Model B respectively failed to emulate the fire spread dynamics.

5.1. Density Paradox in 2D

This section is dedicated to explaining why the simulation of Model A did not work. Model A was the first attempt to model the fire spread, and the first thought was to add as many phenomena as possible to describe the dynamics at the best level of accuracy. However, this resulted in a very complicated numerical implementation that blinded from understanding the first concrete problem related to the dimensionality reduction: the density paradox.

In fact, what happened is that assigning the calculation of the density to the mass balance equation (3.17) and the pressure to the pressure closure equation (3.22) the unpredicted catastrophic events are the following (refer to Figure 5.1):

1. The temperature rises in the centre of the flame;
2. The pressure increases linearly with the temperature;
3. The wind is strongly affected by this pressure gradient that accelerates the fluid outward from the flame;

4. The new velocity vectors transport the mass outside from the flame domain decreasing drastically the density and requiring far more flux to balance the mass;

5. The density eventually decreases dramatically until reaching unlikely negative values.

However, focusing the attention on the actual three-dimensional fire’s dynamics, it seems quite linear to understand why the Model A failed. Consider Figure 5.2. In absence of atmospheric wind, the main fluxes are produced by the buoyancy effect: the lightest hot gas in the centre of the flame tends to float upward, while the cold air slides downward and eventually converge to the base of the flame where, mixed with the fuel and heated up, it will eventually ignite feeding the fire plum. Therefore, what is
modelled by Model A is true at the “top” height, with the exception that the source of mass coming from the bottom is not modelled leading to the density issue. And at the base of the flame the actual wind flux is inward, but this could not be predicted by the ‘blind’ Model A.

Figure 5.2. Buoyancy effect and recirculation in an unconstrained plum flame.

5.2. ABOUT STABILITY

As previously mentioned, the main issue with Model B is the instability of the numerical solution related to the sequential explicit algorithm used. The approximation errors build up especially in those location in the domain where the fields’ magnitude change drastically in time. In fact, as shown in Figure 5.3, in proximity of the flame front, where the temperature gradient is the steepest, one could observe the greatest temperature overshoot at time 0.45 s. This huge error unfortunately is destined to grow exponentially in time, crashing the simulation.
Figure 5.3. Temperature profile in x direction at half domain passing through the ignition point (x=2m, y=2.5m) at different time from the ignition (data from $T_{store}$, Model B).
CONCLUSIONS

This project has successfully accomplished entirely the main objectives:

- The literature review is based on far more than twenty articles about fire-spread fundamentals and modelling, providing a strong theoretical and practical base for this work and for eventual future continuation of the research in this topic.
- The first two parametric Models A and B of fire spread in 2D environment without atmospheric contribution using Matlab was built by 26/06/2017. These were not successfully simulating the fire, but they gave a valid base for the third model.
- A third model, Model C, of fire spread in 2D environment was formulated and simulated by 31/07/2017 providing appreciable results. The main success of this last model is to overcome the problems, of Model A and B, related to the stability of the solution by the use of 4th order Runge Kutta method. Moreover, in this case the fire is supposed to spread in a constant pressure environment.
- The main output of Model A, B and C are presented and discussed in deep, suggesting eventual changes to be carried out in eventual future developments.

In general, the main issue, shared by all the presented models, is related to the dimensionality reduction. In fact, describing the fire, that has three-dimensional dynamics, in a 2D formulation simplifies and neglects a lot of phenomena, such as the very important buoyancy effect and the terrain slope effect. Of course, the solution to this problem would be modelling in 3D, but this will increase the complexity of the model and lengthen the computational time dramatically.
RECOMMENDATIONS OF FUTURE WORKS

The following suggestions for future development of the presented models are mostly numerical in nature, and should be scrupulously evaluated considering the particular objectives of eventual future researches:

- To use the permutated advanced version of the Cholesky factorization algorithm for a processor cluster seems to be the most accurate while being also very time saving and memory saving for solving linear systems, as proven by Karypis [28].
- To use Runge Kutta or other algorithms that allows to solve simultaneously coupled non-linear partial differential equations.
- To choose possibly and implicit or semi-implicit numerical method in order to make the numerical solution the most unconditionally stable possible.
- To model in 3D in order to capture completely the fire dynamics.
- To consider different porosities and different state (solid, liquid and gas) aiming to build a multiscale and multiphase fire model.
- Aim to model turbulent flames using LES (Large Eddy Simulation) or similar methods that model the fields turbulent fluctuations. In this extent, a spectral method could be considered. This should allow also backscatter of energy from the smallest scales to the largest ones. If real time simulation is a requirement, probably there is still need to model the smallest scales to avoid extremely heavy simulations.
- To use a morphing mesh that adapts dynamically to the temperature gradient. Intuitively, this seems to be an aid to the convergence of the solution and to get high quality results.
APPENDIX A: SCRIPT 1

The following Matlab script is original and it simulates the Model A explained in Section 3.3.

```matlab
function Flame2D_MOD_A
% _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
% |
% |
% |
% |
% | Flame2D_MOD_A
% |
% |
% | Solves the fire spread system (Temperature, Velocity, Pressure and Fuel mass fraction) in a rectangular domain with prescribed velocities along the boundary. The solution method for incompressible Navier-Stokes equations is finite difference on a staggered grid with implicit diffusion and a Chorin projection method for the pressure.
% |
% |
% | 05/2017
% |
% | by Paolo Grasso
% |
% | paologk90@gmail.com
% |
% | grassop@uni.coventry.ac.uk
% |
% |
% | CONTENTS:
% |
% | NOMENCLATURE
% |
% | - Fluid dynamics; - Thermal Energy and Chemistry;
% |
% | - Discretisation; - Setting Parameters;
% |
% | - Visualization Settings.
% |
% | 1. CONSTANTS
% |
% | 2. Setting PARAMETERS
% |
% | 3. INITIALIZATION
% |
% | 4. SIMULATION
% |
% | 4.1 Density (Rho)
```
% | 4.2 Momentum balance in x (U)
% | 4.3 Momentum balance in y (V)
% | 4.4 Combustion rate (r)
% | 4.5 Molar fractions (Xi)
% | 4.6 Mixture molar mass (M)
% | 4.7 Mixture specific heat capacity (cp)
% | 4.8 Specific combustion enthalpy (hc)
% | 4.9 Energy Balance (T)
% | 4.10 Pressure update (P)
% | 5. VISUALISATION

clc
close all
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% NOMENCLATURE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Fluid dynamics
% Re : Reynolds' number
% U  : velocity component in x direction (inner domain) [m/s]
% V  : velocity component in y direction (inner domain) [m/s]
% mu : viscosity [kg/m/s]
% ni : kinematic viscosity [m^2/s]
% rho : gas density [kg/m^3]
% Po : reference pressure [Pa]
% P  : specific pressure [m^2/s^2]
% Pold : previous time step pressure field [m^2/s^2]
% uS : South Dirichlet b.c. for u wind component [m/s]
% vS : South Dirichlet b.c. for v wind component [m/s]
% uW : West Dirichlet b.c. for u wind component [m/s]
% vW : West Dirichlet b.c. for v wind component [m/s]
% Ua : averaged along x velocity field (x component) [m/s]
% Va : averaged along y velocity field (y component) [m/s]
% Ue : extended velocity field (x component) with D.b.c. [m/s]
% Ve : extended velocity field (y component) with D.b.c. [m/s]
% UVx : 1st derivative in x of the (UV) matrix [m/s^2]
% UVy : 1st derivative in y of the (UV) matrix [m/s^2]
% U2x : 1st derivative in x of (U^2) matrix [m/s^2]
% V2y : 1st derivative in y of (V^2) matrix [m/s^2]

% Thermal Energy and Chemistry
% R  : gas constant [J/mol/K]
% M  : molar mass mixture [kg/mol]
% To : ambient temperature [K]
% Tp : pyrolysis temperature [K]
% BHf : specific heat of formation [Btu/lb]
% Hf : specific heat of formation [J/kg]
% r : rate of combustion [1/s]
% mHr : molar heat rate of combustion [J/mol/s]
% mH : molar heat of combustion [J/mol]
% Hr : heat rate of combustion [J/s]
% cp : specific heat capacity at constant pressure [J/kgK]
% Ca : diffusion coefficient in atm (turbulent?) [?]
% sigma : Boltzman constant [W/m^2/K^4]
% emit : emittance (ideal black body) [W/m/K]
% kth : thermal conductivity constant [J/K]
% Ahc : pre-exponential factor in Arrhenius (depends?)
% Wf : flame width [m]
% dist2 : squared radial distance from the ignition point [m]
% coef : vector of coeff. balance chemical equilibrium each species
% Mi : vector of molar mass each species [g/mol]
% Hrefi : vector of partial molar formation enthalpies ref. [J/mol]
% cpi : vector of specific heat capacity each species [J/kgK]
% Xio : vector of mole fraction each species initial
% xi : 3D matrix mole fraction

% pedixes:
% _ch : CxHy (hydrocarbon)
% _o : O2 (oxygen)
% _co : CO2 (carbon dioxide)
% _ho : H2O (water)
% _a : Air
% ref : reference

% Discretisation
% lx : length of domain [m]
% ly : width of domain [m]
% tf : final time of simulation [s]
% hx : length of unit cell [m]
% hy : width of unit cell [m]
% Vol : unit cell volume fraction refered to the domain
% Lp : laplacian kroneker matrix for pressure
% perp : premutation for sparser matrix
% Rp : Cholesky triangular permutated matrix
% Lu : laplacian kroneker matrix for pressure
% peru : premutation for sparser matrix
% Ru : Cholesky triangular permutated matrix
% Lv : laplacian kroneker matrix for pressure
% perv : premutation for sparser matrix
% Rv : Cholesky triangular permutated matrix
% CFL : Courant Friedrichs Lewy value
% PecW : Peclet number for N.S.
% PecT : Peclet number for energy
% Froid : Froid number

% Setting Parameters
% Xig : x Ignition Point [m]
% Yig : y Ignition Point [m]
% sigmaT : deviation for Gaussian distribution
% Uinf : asymptotic velocity component in x direction [m/s]
% Vinf : asymptotic velocity component in y direction [m/s]
% alpha : CFL condition value (<1)
% dtmax : maximum time step allowable [s]
% nx : number of cells in x direction
% ny : number of cells in y direction
% nsteps: number of output visualised time steps
% Ye : extinction mass fraction

% Visualization Settings
% Pixcreen : number of screen's pixels
% Nqvx : N vectors for qiver in x direction
% nqvx : rounded n vectors for qiver in x dir.
% xqv : x qiver space
% Nqvy : N vectors for qiver in y direction
% nqvy : rounded n vectors for qiver in y dir.
% yqv : y qiver space
% pmin : caxis minimum specific pressure [m^2/s^2]
% pmax : caxis maximum specific pressure [m^2/s^2]
% levels: number of contours levels
% Tmin : caxis minimum Temperature [K]
% Tmax : caxis maximum Temperature [K]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% 1. CONSTANTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

global Tp Ye Xe Xo2

Tolerance
global R Uinf Vinf mu Uold Vold Rhold
global Tref coef Hrefs Ms cpS YD Xs FlD
global To dt kth sigm emit Ahc
global Ca Ea Ta PolD
global nx ny nt hx hy

mu = 18.27e-6; % air at 291.15 K (it should be higher
ni = 4e-5; % kinematic viscosity [m^2/s] from 15.03e-6 to 62.53e-6
rho = 1.2; % air density [kg/m^3] from 1.204 to 0.524(400°C)
R = 8.314; % gas constant [J/mol/K]
M = 28.9644e-3; % molar mass dry air [kg/mol]

tf = 15e-0; % final time
lx = 5;
ly = 5;
lz = 0.02; % height of domain

% for enthalpy
To = 293.14; % atm temperature [K]
Tp = 600; % autoignition Temperature 573; % minimum pyrolysis
temperature (200-300°C for wood)

BHf = 8660; % Btu/lb = kJ/kg× 2.326
hc = 1000*BHf/2.326; % heat of formation [J/kg]
moHf = 802.301e3; % combustion of CH4 [J/mol]
Tref = 298.15; % [K]
Pref = 101325; % [Pa]
cp = 1013; % heat capacity at constant pressure [J/kg/K]
% cp(250K) = 1003 ; cp(400K)=1013
sigm = 56.7e-9; % [W/m^2/K^4] Boltzman constant
emit = 0.45; % <1 (ideal black body) emittance

% CH4 O2 CO2 H20 air
coef = [ -1 -2 1 2 ];
Ms = [ 16.043 32 44.011 18.016 28.9644 ]; % g/mol
Ms = Ms*0.001; % kg/mol
Hrefs = [ -74.873 0 -393.522 -241.826 ]; % J/mol
Hrefs = Hrefs*1000; % J/kg
cps = [ 2220 919 844 1996 1010 ]; % J/kg/K
Xs = [ 0.1 0.232 0.008 0.001 0.759 ];
Xs(2:end) = Xs(2:end)*(1-Xs(1));
rho = Pref*sum(Ms.*Xs)/(R*To);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% 2. SETTING PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Xig = lx/2;
%%%%% Yig = ly/2;
%%%%% sigmaT = 0.9;

Nqvx = 40;
Nqvy = 40;

Uinf = 0;
Vinf = 0;

dtmax = 1e-3;
nx = 100; % max 140
ny = 100; % max 140

pmin = -1000;
pmax = 1000;
levels = 30;
Tmin = 300;
Tmax = 350; %1200;
Ye = 0.2;
Xe = 0.01;
Xo2e = 0.01;
toll = 0.001; % tolerance on Xi

alpha = 0.9; % < 1 (for cfl)

kth = 0.0515; % thermal conductivity constant (between 0.0257 W/(m*K)mand 0.0515(400°C)
Ca = 0.6; % diffusion coefficient in atm (turbulent?)

Ahc = (1e-8)*2.07e4; %(new CH4) %2.07e4;(previous) %2.07e4/aaa; % pre-
ene factor for fuel consumption HC
Ea = 35000; %(new CH4) %1.917e7;(previous) %1.917e7; % activation
energy for HC: 1.917e7 cal/mol (1 cal = 4.184 J)
Ta = 330; %800; % autoignition temperature
To = 293.14; % atm temperature [K]
Tp = 320; %600; % autoignition Temperature 573; % minimum pyrolysis
temperature (200-300°C for wood)
overT = 5; %80;
enthalpy_reduction = 1; % scale energy
dPr = 1; % reduction delta P
% tolerance = 3;
% 3. INITIALIZATION

fprintf(' \n')
fprintf('  INITIALIZATION  \n')
% .GIF and .AVI initialisation
filename1 = 'Flame_Spread.gif';
filename2 = 'Temperature_Field.gif';
vid1 = VideoWriter('Flame_Spread.avi'); % vid1.FrameRate = 1; %
vid2.FrameRate = 1;
vid2 = VideoWriter('Temperature_Field.avi');
vid5 = VideoWriter('Temperature_cut.avi');
vid1.FrameRate = 5; % Default 30
vid1.Quality = 100; % Default 75
vid2.FrameRate = 5; % Default 30
vid2.Quality = 100; % Default 75
vid5.FrameRate = 5; % Default 30
vid5.Quality = 100; % Default 75
open(vid1);
open(vid2);
open(vid5);
fprintf('##'); % ASCII alt+254;

% qiver initialisation
x = linspace(0,lx,nx+1); hx = lx/nx;
nqvx = round(lx/Nqvx/hx);
xqv = avg((0:hx*nqvx:lx)');
y = linspace(0,ly,ny+1); hy = ly/ny;
nqvy = round(ly/Nqvy/hy);
yqv = avg((0:hy*nqvy:ly)');
nxig = round(Xig/hx)+1;
nyig = round(Yig/hy)+1;
[X,Y] = meshgrid(y,x);

% time discretisation
dt = alpha*min(hx/Uinf,hy/Vinf);
if dt>dtmax dt=dtmax; end;
nt = ceil(tf/dt); dt = tf/nt;
CFL=Uinf*dt/hx;
PecT=hx*Uinf/kth;

% Velocity initial conditions
U = Uinf*ones(nx-1,ny); V = Vinf*ones(nx,ny-1);
Re = ceil(max(Uinf,Vinf)*lx/ni);
fprintf('##');

% Fuel mass fraction
Xnew = zeros(nx,ny,5);
for tt=1:1:5
  Xnew(:,:,tt) = Xs(tt);
end;

% mixture molar mass estimation
M = zeros(nx,ny);
for s=1:1:5
  M = M + Xnew(:,:,s)*Ms(s);
end;

% Velocity initial conditions
U = Uinf*ones(nx-1,ny); V = Vinf*ones(nx,ny-1);
Re = ceil(max(Uinf,Vinf)*lx/ni);
fprintf('##');
Yf = ones(nx, ny);

% Temperature starter ('Mercer and Weber 1997' Gaussian distribution
% around ignition point
T = To*ones(nx, ny);
for cx1=1:1:nx
  for cx2=1:1:ny
    dist2=(((cx1-nxig)*hx)^2+((cx2-nyig)*hy)^2);
    T(cx1,cx2)=T(cx1,cx2)+(Tp+overT-To)*exp(-dist2/sigmaT^2);
  end
end

% Density
rho0 = Pref*M(1,1)/To/R;
Rho = rho0*ones(nx, ny);
Rhold = Rho;

% heat capacity
for s=1:1:5
  cp0 = Xnew(:,s)*Ms(s)*cps(s);
end;

for s=1:1:5
  cp0 = (M.^(-1)).*cp0;
  cp=cp0;
  r = zeros(nx, ny);
  fprintf('##');
end;

% Pressure estimation
P = R*rho0*T.*(M.^(-1));
Pold = P;

% Combustion enthalpy estimation
hc = zeros(nx, ny);
for s=1:1:4
  hc = hc - coef(s)*(Hrefs(s)+(Ms(s)*cps(s))*(T-Tref)).*(M.^(-1));
end;

hc=hc*enthalpy_reduction;
r = zeros(nx, ny);

% Figures' sizes
Pixcreen = get(0,'screensize');

% figure(1) % Flame Domain Plot
FigHandle1 = figure(1);

% set(FigHandle1, 'Position', [+20, Pix_SS(4)/2+20, Pix_SS(3)/2-40, Pix_SS(4)/2-120]);

% figure(2) % Temperature Field Plot
FigHandle2 = figure(2);

% set(FigHandle2, 'Position', [Pix_SS(3)/2+20, Pix_SS(4)/2+20, Pix_SS(3)/2-40, Pix_SS(4)/2-120]);
set(FigHandle2, 'Position', [Pixscreen(3)/2+20, 50, Pixscreen(3)/2-40, 2*(Pixscreen(4)/2-80)]);

% figure(5) % Temperature cut half domain
FigHandle5 = figure(5);

% set(FigHandle1, 'Position', [+20, Pix_SS(4)/2+20, Pixscreen(4)-200]);
FigHandle5 = figure(5);

set(FigHandle5, 'Position', [20, 50, Pixscreen(3)-80, Pixscreen(4)-200]);
clear Pixscreen
fprintf('##
');
fprintf('##
');
fprintf('##
');
fprintf('dx = %1.2s m    dy = %1.2s m
',hx, hy);
fprintf('dt = %1.2s s
',dt);
fprintf('CFL= %1.2s    Peclet= %1.2s  Froid= ???
',CFL,PecT);
fprintf('##
');

k=0;
figure(5)
g_y=250:50:Tmax; % user defined grid Y [start:spaces:end]
g_x=hx:hx:lx-hx; % user defined grid X [start:spaces:end]
for i=1:length(g_x)
    plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:'); % y grid lines
    hold on
end
for i=1:length(g_y)
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:'); % x grid lines
    hold on
end
title(sprintf('Temperature profile (half domain section) Re = %0.1g  t = %0.2g',Re,0))
xlabel('x [m]');
ylabel('T [K]');
hold on
xspace = avg((linspace(0,lx,nx+1))')';
plot(xspace,T(:,round(ny/2)),-b',xspace,Tp*ones(length(xspace)),-.r',xspace,To*ones(length(xspace)),-.k');
axis([0 lx Tmin-50 Tmax]);
drawnow
frame = getframe(5);
writeVideo(vid5,frame);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%% 4. SIMULATION
%%%%%

%%% for k = 1:nt
%%%% 4.1 Density (Rho) (nx)*(ny)
% DRhoUx = diff(avg([rho0*ones(1,ny); Rho; rho0*ones(1,ny)])'*[Uinf*ones(1,ny); U; U(end,:)])/hx;
% DRhoVy = diff(avg([rho0*ones(nx,1) Rho rho0*ones(nx,1)])'*[Vinf*ones(nx,1) V V(:,end)])'/hy;
% dRhodt = -(DRhoUx+DRhoVy);
% RhoNew = Rho +dt*dRhodt;
% fprintf('**** Density ( %1.3f , %1.3f )
[kg/m^3]\n',min(min(RhoNew)),max(max(RhoNew)));
% fprintf('**** dRho/dt = %1.3f [kg/m^3/s]\n',max(max(dRhodt)));
%%% 4.2 Navier Stokes (U) (nx-1)*(ny)
DRhoU2x = -diff(Rho.*((avg([Uinf*ones(1,ny); U; U(end,:)]).^2)))/hx;
% RhoAvg4 =avg((avg([rho0*ones(nx,1) Rho rho0*ones(nx,1)]))');
% avgUey = avg((avg([rho0*ones(nx,1) Rho rho0*ones(nx,1)]))'.*avg([Uinf*ones(nx,1) U U(:,end)]'))'
DRhoUVy = -diff((avg((avg([rho0*ones(nx,1) Rho rho0*ones(nx,1)]))')'.*avg([Uinf*ones(nx,1) U U(:,end)]'))'/hy;
% DPx = -diff(P)/hx;
% DRhoUttold = avg(Rho).*U/dt;
RHS=reshape((DRhoU2x+DRhoUVy-dPr*diff(P)/hx+avg(Rho).*U/dt),[[],1]);
clear DRhoU2x DRhoUVy
muhx2=mu/(hx^2);
muh2=mu/(hy^2);
RHS(1:nx-1)=RHS(1:nx-1)+muh2*Uinf; % U south correction
for jj=1:ny
RHS((nx-1)*(jj-1)+1)=RHS((nx-1)*(jj-1)+1)+muhx2*Uinf; % U west correction
RHS((nx-1)*jj)=RHS((nx-1)*jj)+muhx2*U(end,jj); % U east correction
end;
for ii=1:nx-1
RHS((nx-1)*ny-ii)=RHS((nx-1)*ny-ii)+muh2*U(ii,end); % U north correction
end;
% Kronecker build
% RhoAvgDt=reshape(avg(RhoNew)/dt,[],1);
Kronek = sparse(diag(2*(muhx2+muh2)*ones((nx-1)*ny,1)+reshape(avg(RhoNew)/dt,[],1)));
Kronek(1:end-1,2:end)=sparse(Kronek(1:end-1,2:end)-diag(muhx2*ones((nx-1)*ny-1,1)));
Kronek(2:end,1:end-1)=sparse(Kronek(2:end,1:end-1)-diag(muhx2*ones((nx-1)*ny-1,1)));
for jj=1:ny-1
jjj=(nx-1)*jj;
Kronek(jjj,jjj)=Kronek(jjj,jjj)-muhx2; % U east correction
Kronek(jjj+1,jjj)=0;
Kronek(jjj,jjj+1)=0;
end;
jjj=(nx-1)*ny;
Kronek(jjj,jjj)=Kronek(jjj,jjj)-muhx2; % U east correction
aa=(nx-1)*(ny-1)+1;
bb=(nx-1)*ny;
Kronek(aa:bb,aa:bb)=sparse(Kronek(aa:bb,aa:bb)-diag(muh2*ones(ny-1,1)));
Kronek(nx:end,1:aa-1)=sparse(Kronek(nx:end,1:aa-1)-diag(muh2*ones(ny-1,1)));
Kronek(1:aa-1,nx:end)=sparse(Kronek(1:aa-1,nx:end)-diag(muh2*ones(ny-1,1)));
UNew = reshape((Kronek\RHS),nx,ny);
clear RHS Kronek
fprintf('**** velocity component U : ( %1.3f , %1.3f )
[m/s] \n',min(min(U)),max(max(U)));

4.3 Navier Stokes
DRhoV2y = -diff((Rho.*((avg([Vinf*ones(nx,1) V V(:,end)]).^2))')'/hy;
DRhoUVx = -diff(avg((avg([rho0*ones(1,ny); rho0*ones(1,ny); Rho rho0*ones(1,ny); U U(end,:)]).^2))'.*avg([Uinf*ones(1,ny); U V(:,end)]'))'/hx;
RHS = reshape((DRhoUVx + DRhoV2y -
   dPr*diff(P')'/hy+avg(Rho')'.*V/dt),[],1);
clear DRhoUVx DRhoV2y
muhx2=mu/(hx^2);
muh2=mu/(hy^2);
RHS(1:nx)=RHS(1:nx)+muh2*Vinf; % V south correction
for jj=1:1:ny-1
   RHS(nx*(jj-1)+1)=RHS(nx*(jj-1)+1)+muhx2*Vinf; % V west correction
end;
RHS(nx*jj)=RHS(nx*jj)+muh2*V(end,jj); % V east correction
for ii=1:1:nx
   RHS(nx*(ny-2)+ii)=RHS(nx*(ny-2)+ii)+muh2*V(ii,end); % V north correction
end;

% Kronecker build
Kronek = sparse(diag(2*(muhx2+muh2)*ones(nx*(ny-1),1)+
   (reshape(avg(RhoNew')'/dt,[],1))));
Kronek(1:end-1,2:end)=sparse(Kronek(1:end-1,2:end)-
   diag(muh2*ones(nx*(ny-1)-1,1)));
Kronek(2:end,1:end-1)=sparse(Kronek(2:end,1:end-1)-
   diag(muh2*ones(nx*(ny-1)-1,1)));
for jj=1:1:ny-2
   jji=nx'*jj;
   Kronek(jji,jji)=Kronek(jji,jji)-muhx2; % V east correction
   Kronek(jji+1,jji)=0;
   Kronek(jji,jji+1)=0;
end;
jj=nx*(ny-1);
Kronek(jjj,jjj)=Kronek(jjj,jjj)-muh2; % U east correction
aa=nx*(ny-2)+1;
bb=nx*(ny-1);
Kronek(aa:bb,aa:bb)=sparse(Kronek(aa:bb,aa:bb)-
   diag(muh2*ones(nx,1)));
Kronek(nx+1:end,1:aa-1)=sparse(Kronek(nx+1:end,1:aa-1)-
   diag(muh2*ones(nx*(ny-2),1)));
Kronek(1:aa-1,nx+1:end)=sparse(Kronek(1:aa-1,nx+1:end)-
   diag(muh2*ones(nx*(ny-2),1)));
VNew = reshape((Kronek\RHS),nx,ny-1);
Kronek = sparse(diag(2*(muhx2+muh2)*ones(nx*(ny-1),1)+
   (reshape(avg(RhoNew')'/dt,[],1))));
Kronek(1:end-1,2:end)=sparse(Kronek(1:end-1,2:end)-
   diag(muh2*ones(nx*(ny-1)-1,1)));
Kronek(2:end,1:end-1)=sparse(Kronek(2:end,1:end-1)-
   diag(muh2*ones(nx*(ny-1)-1,1)));
for jj=1:1:ny-2
   jji=nx'*jj;
   Kronek(jji,jji)=Kronek(jji,jji)-muhx2; % V east correction
   Kronek(jji+1,jji)=0;
   Kronek(jji,jji+1)=0;
end;
jj=nx*(ny-1);
Kronek(jjj,jjj)=Kronek(jjj,jjj)-muh2; % U east correction
aa=nx*(ny-2)+1;
bb=nx*(ny-1);
Kronek(aa:bb,aa:bb)=sparse(Kronek(aa:bb,aa:bb)-
   diag(muh2*ones(nx,1)));
Kronek(nx+1:end,1:aa-1)=sparse(Kronek(nx+1:end,1:aa-1)-
   diag(muh2*ones(nx*(ny-2),1)));
Kronek(1:aa-1,nx+1:end)=sparse(Kronek(1:aa-1,nx+1:end)-
   diag(muh2*ones(nx*(ny-2),1)));
VNew = reshape((Kronek\RHS),nx,ny-1);
VNew = reshape((Kronek\RHS),nx,ny-1);
clear RHS Kronek
fprintf('**** velocity component V : ( %1.3f , %1.3f )
[m/s]\n',min(min(V)),max(max(V)));

% 4.4 Combustion rate
indexFl=0;
for jj=1:1:ny
   for ii=1:1:nx
      kdFl = S(T(ii,jj),Xnew(ii,jj,1),Xnew(ii,jj,2)); % Kronecker
delta Flame ignition
      if (kdFl==1)
         indexFl=indexFl+1;
         % record flame domain
         FlD(indexFl,:)=[ii jj];
      end;
   end
end;
r(ii,jj) = -
kdFl*Ahc*T(ii,jj)*(RhoNew(ii,jj)^0.3)*(Xnew(ii,jj,1)^0.5)*Xnew(ii,jj,2)*exp(Ta/T(ii,jj));
end;
end;
fprintf('**** r ( %1.2s , %1.2s )
[1/s]\n',min(min(r)),max(max(r)));

%% 4.5 Molar fractions  (Xi) (nx)*(ny)
    if (exist('Y0D')==1); indexY0=length(Y0D); end;
    for jj=1:1:ny
    for ii=1:1:nx
    for s=1:1:4
    Xsval = Xnew(ii,jj,s) - (dt*coef(s)/Ms(1))*r(ii,jj)*M(ii,jj);
        if ((s==1)||(s==2)) && (Xsval<Xe); Xsval=Xe;
        if ((s==1)||(s==2)) & (Xsval<Xe);
            % combustion rate correction in case of extinction
            r(ii,jj)=(-Xnew(ii,jj,s)+Xsval)*Ms(1)/(dt*coef(s)*M(ii,jj));
        end;
    end
    end
    end
end;
fprintf('**** Molar fractions estimated: Xi ( %1.2s %1.2s , %1.2s %1.2s)
[1/s]\n',min(min(Xnew(:,:,1:2))),max(max(Xnew(:,:,1:2))));

%% 4.6 Mixture molar mass  (M) (nx)*(ny)
    M = zeros(nx,ny);
    for s=1:1:5
        M = M + Xnew(:,:,s)*Ms(s);
    end;
    fprintf('**** mixture molar mass estimated: M ( %1.2s , %1.2s ) [g/mol]
[1/s]\n',min(min(M))*1000, max(max(M))*1000);

%% 4.7 Mixture specific heat capacity (cp) (nx)*(ny)
    cpNew = zeros(nx,ny);
    for s=1:1:5
        cpNew = cpNew + (cps(s)*Ms(s))*Xnew(:,:,s).*((M.^(-1)));
    end
    fprintf('**** Mixture specific heat capacity: cp ( %1.2s , %1.2s ) [J/kg/K]
[1/s]\n',min(min(cpNew)), max(max(cpNew)));

%% 4.8 Specific combustion enthalpy  (hc) (nx)*(ny)
    hcNew = zeros(nx,ny);
    for s=1:1:4
        hcNew = hcNew - coef(s)*((Hrefs(s)+(Ms(s)*cps(s)))*(T-Tref)).*(M.^(-1));
    end
    hcNew = hcNew*enthalpy_reduction;
    fprintf('**** Specific combustion enthalpy: hc ( %1.2s , %1.2s ) [J/kg]
[1/s]\n',min(min(hcNew)), max(max(hcNew)));

%% 4.9 Energy Balance  (T) (nx)*(ny)
    invCp  = cp.^(-1);
    invRCp = (RhoNew.^(-1)).*invCp;
    Rhoex = [rho0*ones(1,ny) ; avg(Rho) ; rho0*ones(1,ny) ];
    Rhoey = [rho0*ones(nx,1) ; avg(Rho)' ; rho0*ones(nx,1) ];
    Tex   = [To*ones(1,ny) ; avg(T) ; To*ones(1,ny) ];
    Tey   = [To*ones(nx,1) ; avg(T)' ; To*ones(nx,1) ];

81
hcex = [hc(1,:); avg(hc); hc(end,:)];
hcex = [hc(:,1); avg(hc)'; hc(:,end)];
cpex = [cp0(1,1)*ones(1,ny); avg(cp); cp0(1,1)*ones(1,ny)];
cpex = [cp0(1,1)*ones(nx,1); avg(cp)'; cp0(1,1)*ones(nx,1)];
Uex = [Uinf*ones(1,ny); U; U(end,:)];
Uex = [Vinf*ones(ny,1); V; V((1:end),:)];
dRCTUx = diff(Rhoex.*cpex.*Tex.*Uex)/hx;
dRCTUx = diff((Rhoex.*cpex.*Tex.*Uex)'/hx);
dRCTVy = diff((Rhoex.*hcex.*Uex))'/hy;
dRCTVy = diff((Rhoex.*hcex.*Uex))'/hy;

dRhcUx = diff(Rhoex.*hcex.*Uex)/hx;
dRhcUx = diff((Rhoex.*hcex.*Uex))'/hx;
dRhcVy = diff((Rhoex.*hcey.*Vey))'/hy;
dRhcVy = diff((Rhoex.*hcey.*Vey))'/hy;

dTRad = -sigm*emit*invRCp.*((diff(Tex.^4)/hx+diff((Tey.^4))')'/hy);
dTRad = -sigm*emit*invRCp.*((diff(Tex.^4)/hx+diff((Tey.^4))')'/hy);
cpex = [cp0(1,1)*ones(1,ny); cp; cp0(1,1)*ones(1,ny)];
cpex = [cp0(1,1)*ones(nx,1); cp; cp0(1,1)*ones(nx,1)];
Tex = [To*ones(1,ny); T; To*ones(1,ny)];
Tex = [To*ones(nx,1); T; To*ones(nx,1)];
hcey = [hc(1,:); hc; hc(end,:)];
hcey = [hc(:,1); hc; hc(:,end)];
kdx2 = kth*diff(invCp(1,:); avg(invCp); invCp(end,:)).*diff(cpex.*Tex)/hx^2;
kdx2 = kth*diff(invCp(:,1); avg(invCp)'.'); diff((cpey.*Tey))'/hy^2;
kdx2 = kth*diff(invCp(:,1); avg(invCp); invCp(end,:)).*diff(hcey)/hx^2;
kdx2 = kth*diff(invCp(:,1); avg(invCp)'.'); diff((hcey)')'/hy^2;
Td = invRCp.*(Rho.*cp.*T);
dT = invRCp.*(Rho.*cp.*T); dTpressure = invRCp.*(P-Pold)/dt;
dT3Dconv = -invRCp.*(Ca*(T-To));
dT3Dconv = -invRCp.*(Ca*(T-To));

dTcombustion = -invRCp.*(RhoNew.*hcNew.*r);
TNew = Tbase + dt*(dTpressure + dT3Dconv + dTcombustion);

dTransport = -invRCp.*(dRCTUx+dRCTVy+dRhcUx+dRhcVy);
TNew = TNew + dt*dTransport; % enthalpy transport
dTransport = -invRCp.*(dRCTUx+dRCTVy+dRhcUx+dRhcVy);
TNew = TNew + dt*dTransport; % enthalpy transport
dTdiffusion = invRCp.*(kdTx2+kdTy2+kdhcx2+kdhcy2);
TNew = TNew + dt*dTdiffusion; % diffusion
dTdiffusion = invRCp.*(kdTx2+kdTy2+kdhcx2+kdhcy2);
TNew = TNew + dt*dTdiffusion; % diffusion
dTdiffusion = invRCp.*(kdTx2+kdTy2+kdhcx2+kdhcy2);

fprintf('%*s dT(3D) : ( %1.2s , %1.2s ) [K/s]
',min(min(dT3Dconv)), max(max(dT3Dconv)));

fprintf('%*s dT(U,V) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(kth) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTdiffusion)), max(max(dTdiffusion)));

fprintf('%*s dT(Rad) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(F) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)))

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));

fprintf('%*s dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));

fprintf('%*s dT(hc) : ( 0 , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));

fprintf('%*s dT(T) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));

fprintf('%*s T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)))
%% 4.10 Pressure update (P) (nx)*(ny)
PNew = R*RhoNew.*T.*(M.^(-1));
fprintf('**** Pressure P : (%1.2s, %1.2s) Pa

min(min(P)), max(max(P));
fprintf('**** Pressure Pnew : (%1.2s, %1.2s) Pa

min(min(PNew)), max(max(PNew));
% CHECKED! OK

%% Time update
Pold=P;
P=PNew;
Rho = RhoNew;
U = UNew;
V = VNew;
hc = hcNew;
% CHECKED! OK

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%% 5. VISUALISATION
figure(1)
clear
% Pressure plot
[hP hP]=contourf(avg(x),avg(y),(P-Po)',levels);
[hP hP]=contourf(x(1:end-1),y(1:end-1),(P)',levels);
set(hP,'LineStyle','none');
colormap(pink)
% https://uk.mathworks.com/help/matlab/ref/colormap.html
caxis([95000, 115000]);
colorbar
axis equal, axis([0 lx 0 ly])
% Wind plot
Usg = [Uinf*ones(1,ny); avg(U); Uinf*ones(1,ny)]; % U on scalar grid
Vsg = [Vinf*ones(nx,1) avg(V') Vinf*ones(nx,1)]; % V on scalar grid
Wind = (Usg.^2+Vsg.^2).^0.5; % total velocity magnitude combined on scalar grid
WindMax = max(max(Wind));
for gg=1:1:length(xqv)
    for hh=1:1:length(yqv)
        ggg=(gg-1)*nqvx+1;
        hhh=(hh-1)*nqvy+1;
        QVX(gg,hh)=Usg(ggg,hhh)/WindMax; %*lx
        QVY(gg,hh)=Vsg(ggg,hhh)/WindMax; %*ly
    end;
end;
hold on
[Xqv,Yqv]=meshgrid(xqv,yqv);
quiver(Xqv,Yqv,QVX',QVY',hx*nqvx*lx,'k-');

% % IF WANT TO PLOT FLAME AND ZERO FUEL DOMAIN UNLOCK THE FOLLOWING
% if exist('FlD') == 1
% % Flame Domain Plot
% for indx=1:1:length(FlD)
%     hold on
%     FlDx=hx*(FlD(indxx,1)-1.5);
%     FlDy=hy*(FlD(indxx,2)-1.5);
% end;
rectangle('Position',[FlDx,FlDy,hx,hy], 'FaceColor','r', 'EdgeColor','r', 'LineWidth',1)
end;
% 0 fuel domain
% title(sprintf('Pressure field (colormap)\n Induced velocity field (arrows)\n Re = %0.1g  t = %0.2g',Re,k*dt))
% if exist('Y0D')==1
% for indxx=1:1:length(Y0D)
% hold on
% Y0Dx=hx*(Y0D(indxx,1)-1.5);
% Y0Dy=hy*(Y0D(indxx,2)-1.5);
% rectangle('Position',[Y0Dx,Y0Dy,hx,hy], 'FaceColor','g', 'EdgeColor','g', 'LineWidth',1)
% end
% grid on
drawnow
frame = getframe(1);
im = frame2im(frame);
[imind,cm] = rgb2ind(im,256);
if k == 1;
imwrite(imind,cm,filename1, 'gif', 'Loopcount', inf);
else
imwrite(imind,cm,filename1, 'gif', 'WriteMode', 'append');
end;
writeVideo(vid1,frame);

figure(2) % Temperature Field Plot
clf(2)
v = linspace(To,Tp+300,20);
% contourf(avg(avg(X)')-0.5*hx,avg(avg(Y')')-0.5*hy,T',v,'w-
% )
% [hT hT]=contourf(avg(x),avg(y),T',levels);
% [hT hT]=contourf(x(1:end-1),y(1:end-1),T',levels);
% set(hT,'LineStyle','none');
colormap(jet)
https://uk.mathworks.com/help/matlab/ref/colormap.html
caxis([Tmin,Tmax])
colorbar ('AxisLocation','in')
axis equal, axis([0 lx 0 ly])
title(sprintf('Temperature field (colormap)\n Ambient and pyrolysis temperatures (white contours) Re = %0.1g  t = %0.2g',Re,k*dt))
hold on
% contour(avg(x),avg(y),round(T')', [To,Tp],'w-') % 'ShowText','on'
contour(x(1:end-1),y(1:end-1),round(T')', [To,Tp],'w-') % 'ShowText','on'
drawnow
frame = getframe(2);
writeVideo(vid2,frame);

figure(5)
xspace = avg((linspace(0,lx,nx+1))')';
clf(5)
g_y=[250:50:Tmax]; % user defined grid Y [start:spaces:end]
g_x=[hx:hx:1x-hx]; % user defined grid X [start:spaces:end]
for i=1:length(g_x)
plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:') %y grid lines
hold on
end
for i=1:length(g_y)
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:') %x grid lines
hold on
end
hold on
title(sprintf('Temperature profile (half domain section)\nRe = %0.1g  t = %0.2g',Re,k*dt))
xlabel('x [m]');
ylabel('T [K]');
plot(xspace,T(:,round(ny/2)),'-b',xspace,Tp*ones(length(xspace)),'-r',xspace,To*ones(length(xspace)),'-.k');
axis([0 lx Tmin -50 Tmax]);
drawnow
frame = getframe(5);
writeVideo(vid5,frame);
clear FlD
fprintf('n
');

% 6. SUB-ROUTINES

% Average matrix
function B = avg(A,k)
    if nargin<2, k = 1; end
    if size(A,1)==1, A = A'; end
    if k<2
        B = (A(2:end,:) + A(1:end-1,:))/2;
    else
        B = avg(A,k-1); end
    if size(A,2)==1, B = B'; end

% Kroneker matrix
function A = K1(n,h,a11)
    % all: Neumann=1, Dirichlet=2, Dirichlet mid=3;
    A = spdiags([-1 a11 0;ones(n-2,1)*[-1 2 -1];0 all -1],[-1:1,n,n]')/h^2;

% Ignition Condition
function spread = S(T,X,Xo)
    global Tp Xe Xo2e
    if (T>Tp) & (X>Xe) & (Xo>Xo2e) spread = 1;
    else spread = 0; end
APPENDIX B: SCRIPT 2

The following Matlab script is original and it simulates the Model B explained in Section 3.4.

```
function Flame2D_MOD_B

% Flame2D_MOD_B: "Back to Chorin"
% Solves the fire spread system (Temperature, Velocity, Pressure and Fuel mass fraction) in a rectangular domain with prescribed velocities along the boundary. The solution method for incompressible Navier-Stokes equations is finite difference on a staggered grid with implicit diffusion and a Chorin projection method for the pressure.
% 05/2017
% by Paolo Grasso
% paologk90@gmail.com
% grassop@uni.coventry.ac.uk

% CONTENTS:
% NOMENCLATURE
% - Fluid dynamics; - Thermal Energy and Chemistry;
% - Discretisation; - Setting Parameters;
% - Visualtization Settings.
% 1. CONSTANTS
% 2. Setting PARAMETERS
% 3. INITIALIZATION
% 4. SIMULATION
% 4.1 Density (Rho)
```

86
% | 4.2 Momentum balance in x (U) *no pressure
% | 4.3 Momentum balance in y (V) *no pressure
% | 4.4 Pressure projection (Chorin) (P) (U*) (V*)
% | 4.5 Combustion rate (r)
% | 4.6 Molar fractions (Xi)
% | 4.7 Mixture molar mass (M)
% | 4.8 Mixture specific heat capacity (cp)
% | 4.9 Specific combustion enthalpy (hc)
% | 4.10 Energy Balance (T)
% | 5. VISUALISATION

clc
close all
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% NOMENCLATURE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Fluid dynamics
% Re    : Reynolds' number
% U     : velocity component in x direction (inner domain) [m/s]
% V     : velocity component in y direction (inner domain) [m/s]
% mu    : viscosity [kg/m/s]
% ni    : kinematic viscosity [m^2/s]
% rho   : gas density [kg/m^3]
% Po    : reference pressure [Pa]
% P     : specific pressure [m^2/s^2]
% Pold  : previous time step pressure field [m^2/s^2]
% uS    : South Dirichlet b.c. for u wind component [m/s]
% vS    : South Dirichlet b.c. for v wind component [m/s]
% uW    : West Dirichlet b.c. for u wind component [m/s]
% vW    : West Dirichlet b.c. for v wind component [m/s]
% Ua    : averaged along x velocity field (x component) [m/s]
% Va    : averaged along y velocity field (y component) [m/s]
% Ue    : extended velocity field (x component) with D.b.c. [m/s]
% Ve    : extended velocity field (y component) with D.b.c. [m/s]
% UVx   : 1st derivative in x of the (UV) matrix [m/s^2]
% UVy   : 1st derivative in y of the (UV) matrix [m/s^2]
% U2x   : 1st derivative in x of (U^2) matrix [m/s^2]
% V2y   : 1st derivative in y of (V^2) matrix [m/s^2]

%% Thermal Energy and Chemistry
% R      : gas constant [J/mol/K]
% M      : molar mass mixture [kg/mol]
% To     : ambient temperature [K]
% Tp     : pyrolysis temperature [K]
% BHf : specific heat of formation [Btu/lb]
% Hf : specific heat of formation [J/kg]
% r : rate of combustion [1/s]
% mHR : molar heat rate of combustion [J/mol/s]
% mH : molar heat of combustion [J/mol]
% HR : heat rate of combustion [J/s]
% cp : specific heat capacity at constant pressure [J/kg/K]
% Ca : diffusion coefficient in atm (turbulent?) [?]
% sigma : Boltzman constant [W/m^2/K^4]
% emit : emittance (ideal black body) [W/m^2]
% kth : thermal conductivity constant [K/m]
% Ea : activation energy for HC [J]
% AHC : pre-esponential factor in Arrhenius [depends?]
% Wf : flame width [m]
% dist2 : squared radial distance from the ignition point [m]
% coef : vector of coeff. balance chemical equilibrium each species
% Mi : vector of molar mass each species [g/mol]
% Hrefi : vector of partial molar formation enthalpies ref. [J/mol]
% cpi : vector of specific heat capacity each species [J/kg/K]
% Xio : vector of mole fraction each species initial
% Xi : 3D matrix mole fraction

% pedixes:
% _ch : CxHy (hydrocarbon)
% _o : O2 (oxygen)
% _co : CO2 (carbon dioxide)
% _ho : H2O (water)
% _a : Air
% ref : reference

% Discretisation
% lx : length of domain [m]
% ly : width of domain [m]
% tf : final time of simulation [s]
% dx : length of unit cell [m]
% dy : width of unit cell [m]
% Vol : unit cell volume fraction refered to the domain
% Lp : laplacian kroneker matrix for pressure
% perp : premutation for sparser matrix
% Rp : Cholesky triangular permutated matrix
% Lu : laplacian kroneker matrix for pressure
% peru : premutation for sparser matrix
% Ru : Cholesky triangular permutated matrix
% Lv : laplacian kroneker matrix for pressure
% perv : premutation for sparser matrix
% Rv : Cholesky triangular permutated matrix
% CFL : Courant Friedrichs Lewy value
% PecW : Peclet number for N.S.
% PecT : Peclet number for energy
% Froid : Froid number

% Setting Parameters
% Xig : x Ignition Point [m]
% Yig : y Ignition Point [m]
% sigmaT : deviation for Gaussian distribution
% Uinf : asymptotic velocity component in x direction [m/s]
% Vinf : asymptotic velocity component in y direction [m/s]
% alpha : CFL condition value (<1)
% dtmax : maximum time step allowed [s]
% nx : number of cells in x direction
% ny : number of cells in y direction
% nsteps: number of output visualised time steps
% Ye : extinction mass fraction

% Visualization Settings
% Pixcreen : number of screen's pixels
% Nqvx : N vectors for qiver in x direction
% nqvx : rounded N vectors for qiver in x dir.
% xqv : x qiver space
% Nqvy : N vectors for qiver in y direction
% nqvy : rounded N vectors for qiver in y dir.
% yqv : y qiver space
% pmin : caxis minimum specific pressure [m^2/s^2]
% pmax : caxis maximum specific pressure [m^2/s^2]
% levels: number of contours levels
% Tmin : caxis minimum Temperature [K]
% Tmax : caxis maximum Temperature [K]

% 1. CONSTANTS
%
% global Tp Ye Xe Xo2e
% global R Uinf Vinf mu
% global Tref coef Hrefs Ms cps Y0D Xs FlD
% global To dt kth sigm emit Ahc
% global Ca Ea Ta
% global nx ny nt dx dy

mu = 18.27e-6; % air at 291.15 K (it should be higher
ni = 4e-5; % kinematic viscosity [m^2/s] from 15.03e-6 to 62.53e-6
rho = 1.2; %0.9; % air density [kg/m^3] from 1.204 to 0.524(400°C)
R = 8.314; % gas constant [J/mol/K]
M = 28.9644e-3; % molar mass dry air [kg/mol]
tf = 0.45; % final time
lx = 5;
ly = 5;
% lz = 0.02; % height of domain

% for enthalpy
To = 293.14; % atm temperature [K]
Tp = 600; % autoignition Temperature 573; % minimum pyrolysis
temperature (200-300°C for wood)

BHf = 8660; % Btu/lb = kJ/kgx 2.326
hc = 1000*BHf/2.326; % heat of formation [J/kg]
molHf = 802.301e3; % combustion of CH4 [J/mol]
Tref = 298.15; % [K]
Pref = 101325; % [Pa]
cp = 1013; % heat capacity at constant pressure [J/kg/K]
cp(250K) = 1003 ; cp(400K) = 1013

sigm = 5.6704e-8; % [W/m^2/K^4] Boltzmann constant
emit = 0.85; % <1 (ideal black body) emittance

coef = [ -1 -2 1 2 ];
Ms = [ 16.043 32 44.011 18.016 28.964 ]; % g/mol
Ms = Ms*0.001; % kg/mol
Hrefs = [ -74.873 0 -393.522 -241.826 ]; % kJ/mol
Hrefs = Hrefs*1000; % J/mol
cps = [ 2220 919 844 1996 101 ]; % J/kg/K
Xs = [ 0.1 0.232 0.008 0.001 0.759 ];
Xs(2:end) = Xs(2:end)*(1-Xs(1));
rho = Pref*sum(Ms.*Xs)/(R*To);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% 2. SETTING PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Xig = 2*lx/5;
Yig = ly/2;
sigmaT = 0.07;
sigmaT = sigmaT*lx;
Nqvx = 30;
Nqvy = 30;

Uinf = 0;
Vinf = 0;

nx = 125; % max 130
ny = 125; % max 130

pmin = -1000;
pmax = 1000;
levels = 30;
Tmin = 300;

Ye = 0.2;
Xe = 0.0268;
Xo2e = 0.0268;

alpha = 0.9; % < 1 (for cfl)
kth = 0.096; % thermal conductivity constant (between 0.0257 W/(m*K) and 0.0515 (400°C)
Ca = 100.6; % diffusion coefficient in atm (turbulent?)
dtmax = 2.5e-2;
hz = 0.1; % [m]
Ahc = (1e-7)*2.07e4; % (new CH4) %2.07e4;(previous) %2.07e4/aaa; % pre-e factor for fuel consumption HC
Ea = 35000; % (new CH4) %1.917e7;(previous) %1.917e7; % activation energy for HC: 1.917e7 cal/mol (1 cal = 4.184 J)
Ta = 900; %800; % autoignition temperature
To = 293.14; % atm temperature [K]
Tp = 500; %600; % autoignition Temperature 573; % minimum pyrolysis
temperature (200-300°C for wood)
overT = 50; %80;
enthalpy_reduction = 0.75; % scale energy
Tmax = 750; %ceil(Ta/50)*50+50; %1200;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%% 3. INITIALIZATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
fprintf( 'INITIALIZATION  
' )

% .GIF and .AVI initialisation
nameT= ('T_store.mat');
filename1 = 'Density.gif';
filename2 = 'Flame_Spread.gif';
filename3 = 'Temperature_Field.gif';
filename4 = 'Temperature_Profile.gif';
vid1 = VideoWriter('Density.avi');
vid2 = VideoWriter('Flame_Spread.avi');
vid3 = VideoWriter('Temperature_Field.avi');
vid4 = VideoWriter('Temperature_Profile.avi');
vid1.FrameRate = 5; % Default 30
vid1.Quality = 100; % Default 75
vid2.FrameRate = 5; % Default 30
vid2.Quality = 100; % Default 75
vid3.FrameRate = 5; % Default 30
vid3.Quality = 100; % Default 75
vid4.FrameRate = 5; % Default 30
vid4.Quality = 100; % Default 75
open(vid1);
open(vid2);
open(vid3);
open(vid4);
fprintf( '   ##' ); % ASCII alt+254;

% qiver initialisation
x = linspace(0,lx,nx+1); dx = lx/nx;
nqvx= round(lx/Nqvx/dx);
xqv =avg((0:dx*nqvx:lx)')';
y = linspace(0,ly,ny+1); dy = ly/ny;
nqvy= round(ly/Nqvy/dy);
yqv =avg((0:dy*nqvy:ly)')';
nxig  = round(Xig/dx)+1;
nyig  = round(Yig/dy)+1;
[X,Y] = meshgrid(y,x);

% time discretisation
dt = alpha*min(dx/Uinf,dy/Vinf);
if dt>dtmax dt=dtmax; end;
n= ceil(tf/dt); dt = tf/n;
CFL=Uinf*dt/dx;
PecT=dx*Uinf/kth;

% Velocity initial conditions
U = Uinf*ones(nx-1,ny); V = Vinf*ones(nx,ny-1);
Re = ceil(max(Uinf,Vinf)*lx/ni);
fprintf('##');

% Fuel mass fraction
Xnew = zeros(nx,ny,5);
for tt=1:1:5
    Xnew(:,:,tt) = Xs(tt);
end;

% mixture molar mass estimation
M = zeros(nx,ny);
for s=1:1:5
    M = M + Xnew(:,:,s)*Ms(s);
end;

Yf = ones(nx,ny);

% Temperature starter ("Mercer and Weber 1997" Gaussian distribution
% around ignition point
T = To*ones(nx,ny);
for cx1=1:1:nx
    for cx2=1:1:ny
        dist2=(((cx1-nxig)*dx)^2+((cx2-nyig)*dy)^2);
        T(cx1,cx2)=T(cx1,cx2)+(Tp+overT-To)*exp(-dist2/sigmaT^2);
    end
end;

% Density
Rho = (Pref/R)*M.*(T.^(-1));
RhoNew = Rho;
rho0=Rho(1,1);

% heat capacity
for s=1:1:5
    cp0 = Xnew(:,:,s)*Ms(s)*cps(s);
end;
cp0 = (M.^(-1)).*cp0;
cp=cp0;
fprintf('##');

% Pressure estimation
P = Pref*ones(nx,ny);
PNew = P;

% Kroneker P with Dirichlet
KronekP = kron(speye(ny),K1(nx,dx,2))+kron(K1(ny,dy,2),speye(nx));
perp = symamd(KronekP); Rp = chol(KronekP(perp,perp)); Rpt = Rp';

% Combustion enthalpy estimation
hc = zeros(nx,ny);
for s=1:1:4
    hc = hc - coef(s)*(Hrefs(s)+(Ms(s)*cps(s))*(T-Tref)).*(M.^(-1));
end;
hc=hc*enthalpy_reduction;
r = zeros(nx,ny);

% Figures' sizes
Pixcreen = get(0,'screensize');
% figure(1) % Density Plot
% FigHandle1 = figure(1);
% set(FigHandle1, 'Position', [+20, Pix_SS(4)/2+20, Pix_SS(3)/2-40, Pix_SS(4)/2-120]);
% set(FigHandle1, 'Position', [20, 40, Pixscreen(3)-40, Pixscreen(4)-140]);
% figure(2) % Flame Spread Plot
% FigHandle2 = figure(2);
% set(FigHandle2, 'Position', [20, 40, Pixscreen(3)-40, Pixscreen(4)-140]);
% figure(3) % Temperature Field Plot
% FigHandle3 = figure(3);
% set(FigHandle3, 'Position', [20, 40, Pixscreen(3)-40, Pixscreen(4)-140]);
% figure(4) % Temperature Profile half domain Plot
% FigHandle4 = figure(4);
% set(FigHandle4, 'Position', [20, 40, Pixscreen(3)-40, Pixscreen(4)-140]);
% figure(5) % Combustion enthalpy, temperature, mass fractions in the
% centre of the flame
% FigHandle5 = figure(5);
% set(FigHandle5, 'Position', [20, 40, Pixscreen(3)-40, Pixscreen(4)-140]);
clear Pixscreen
fprintf('%#\n');
fprintf('
');

CFL=max(abs(Uinf),0.001)*dt/dx;
PecT=dx*max(abs(Uinf),0.001)/kth;
Froid = 0.01/(9.81*1*(850-To)/850);
fprintf(' dx = %1.2s m    dy = %1.2s m\n',dx, dy);
fprintf(' dt = %1.2s s\n',dt);
fprintf(' CFL= %1.2s    Peclet= %1.2s  Froid= %1.2s\n',CFL,PecT,Froid);
fprintf('
');

% figure(5) t=0
k=0;
figure(4)
xspace = avg((linspace(0,lx,nx+1))')';
clf(5)
g_y=[250:50:Tmax+100]; % user defined grid Y [start:spaces:end]
g_x=[dx:dx:lx-dx]; % user defined grid X [start:spaces:end]
for i=1:length(g_x)
    plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:') %y grid lines
    hold on
end
for i=1:length(g_y)
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:') %x grid lines
    hold on
end
title(sprintf('Temperature profile (half domain section)\n Re = %0.1g  t = %0.2g s',Re,k*dt))
xlabel('x [m]');
ylabel('T [K]');
plot(xspace,Tp*ones(length(xspace)),'-b',xspace,To*ones(length(xspace)),'-k','LineWidth',1.5);
hold on
plot(xspace,(T(:,round(ny/2))),'-r','LineWidth',2);
\[
T_{\text{store}}(:,1) = T(:,\text{round}(\text{ny}/2));
\]

```matlab
% axis([0 1x Tmin-50 Tmax+100]);
drawnow
frame = getframe(4);
im = frame2im(frame);
[imind, cm] = rgb2ind(im, 256);
if k == 0;
    imwrite(imind, cm, filename4, 'gif', 'Loopcount', inf);
else
    imwrite(imind, cm, filename4, 'gif', 'WriteMode', 'append');
end;
writeVideo(vid4, frame);
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% 4. SIMULATION

for k = 1:nt

% 4.1 Density update (P) (nx)*(ny)
RhoNew = P*M.*T.^(-1))/R;
fprintf('**** density Rho : ( %1.2s , %1.2s ) [kg/m^3]\n', min(min(Rho)), max(max(Rho)));
fprintf('**** density RhoNew : ( %1.2s , %1.2s ) [kg/m^3]\n', min(min(RhoNew)), max(max(RhoNew)));

% 4.2 Navier Stokes (U) (nx-1)*(ny)
DRhoU2x = -diff(Rho.*((avg([Uinf*ones(1,ny); U; U(end,:)]).^2))/dx);
DRhoUVy = -diff((avg((avg([rho0*ones(nx,1) Rho; rho0*ones(nx,1)]'))'.*avg([Uinf*ones(nx-1,1) U U(:,end)]').*avg([Vinf*ones(nx,1) V V(:,end)]'))'/dy);
RHSp=reshape((DRhoU2x+DRhoUVy+avg(Rho).*U/dt),[],1);
clear DRhoU2x DRhoUVy
muhx2=mu/(dx^2);
muhy2=mu/(dy^2);
RHSp(1:nx-1)=RHSp(1:nx-1)+muhy2*Uinf; % U south correction
for jj=1:ny
    RHPsp((nx-1)*(jj-1)+1)=RHSp((nx-1)*(jj-1)+1)+muhx2*Uinf; % U west correction
end;
RHSp((nx-1)*(jj-1)+1)=RHSp((nx-1)*(jj-1)+1)+muhx2*Uinf; % U east correction
end;
end
```

\[
Kronek = \text{sparse}(-\text{diag}(2*(\text{muhx2}+\text{muhy2})\text{ones((nx-1)*ny,1))}+\text{reshape}(\text{avg}(\text{RhoNew})/\text{dt},[1,1])));\]

```matlab
Kronek(1:end-1,2:end)=\text{sparse}(\text{Kronek}(1:end-1,2:end)-
-\text{diag}(\text{muhx2}^2\text{ones((nx-1)*ny,1))});
Kronek(2:end,1:end-1)=\text{sparse}(\text{Kronek}(2:end,1:end-1)-
-\text{diag}(\text{muhy2}^2\text{ones((nx-1)*ny-1,1)))));
for jj=1:ny-1
    jjj=(nx-1)*jj;
    Kronek(jjj,jjj)=\text{Kronek}(jjj,jjj)-\text{muhx2}; % U east correction
    Kronek(jjj+1,jjj)=0;
    Kronek(jjj,jjj+1)=0;
end;
```
\[
\text{jjj}=(n-1)\cdot ny;
\text{Kronek}([\text{jjj}, \text{jjj}])=\text{Kronek}([\text{jjj}, \text{jjj}]) - \mu_{hx}2; \quad \% \text{U east correction}
\]
\[
\text{aa}=(n-1)\cdot (ny-1)+1;
\text{bb}=(n-1)\cdot ny;
\text{Kronek}([\text{aa}:\text{bb}, \text{aa}:\text{bb}])=\text{sparse(}\text{Kronek}([\text{aa}:\text{bb}, \text{aa}:\text{bb}])-\text{diag(}\mu_{hy}2\cdot \text{ones}(nx-1,1)));\]
\[
\text{Kronek}(nx:end,1:aa-1)=\text{sparse(}\text{Kronek}(nx:end,1:aa-1)-\text{diag(}\mu_{hx}2\cdot \text{ones}(nx-1,1)));\]
\[
\text{Kronek}(1:aa-1,nx:end)=\text{sparse(}\text{Kronek}(1:aa-1,nx:end)-\text{diag(}\mu_{hx}2\cdot \text{ones}(nx-1,1)));\]
\[
\text{UNew}=\text{reshape(}(\text{Kronek}\text{\textbackslash RHSp}),nx-1,ny);\]
\[
\text{clear RHS Kronek};\]

\[
\% 4.3 \text{Navier Stokes} \quad (V) (nx)\cdot (ny-1)
\]
\[
\text{DRhoV2y}=-\text{diff}((\text{Rho}\cdot((\text{avg}([\text{Vinf}\cdot \text{ones}(nx,1) V V(:,end)])'.^2))'/dy);
\text{DRhoUVx}=-\text{diff}((\text{avg}([\text{rho0}\cdot \text{ones}(1,ny) ; \text{Rho}; \text{rho0}\cdot \text{ones}(1,ny)])').*\text{avg}([\text{Uinf}\cdot \text{ones}(1,ny) ; \text{U}; \text{U(end,:)}]).'/dx);
\]
\[
\text{RHSp}=\text{reshape}((\text{DRhoUVx}+\text{DRhoV2y}+\text{avg}(\text{Rho}')\cdot V/dt),[],1);
\]
\[
\text{clear DRhoUVx DRhoV2y};
\mu_{hx}2=\text{mu}/(dx^2);
\mu_{hy}2=\text{mu}/(dy^2);
\text{RHSp}(1:nx)=\text{RHSp}(1:nx)+\mu_{hx}2\cdot \text{Vinf}; \quad \% \text{V south correction}
\]
\[
\text{for jj=1:1:ny-1}
\text{RHSp(nx*(jj-1)+1)=RHSp(nx*(jj-1)+1)+\mu_{hx}2\cdot \text{Vinf}; \quad \% \text{V west correction}
\text{end};
\text{RHS}(nx*jj)=\text{RHS}(nx*jj)+\mu_{hx}2\cdot \text{Vinf}; \quad \% \text{V east correction}
\text{end};
\text{RHS}(nx*(ny-2)+ii)=\text{RHS}(nx*(ny-2)+ii)+\mu_{hy}2\cdot \text{V(ii,end)}; \quad \% \text{V north correction}
\text{end};
\%
\text{Kronecker build}
\%
\text{RhoAvgDt}=\text{reshape(}(\text{avg(}\text{RhoNew})/dt,[],1);
\text{Kronek} = \text{sparse(}\text{diag}(2*\mu_{hx}2+\mu_{hy}2)*\text{ones}(nx*(ny-1),1)+\text{reshape(}(\text{avg(}\text{RhoNew})'/dt,[],1))));
\text{Kronek}(1:end-1,2:end)=\text{sparse(}\text{Kronek}(1:end-1,2:end)-\text{diag(}\mu_{hx}2\cdot \text{ones}(nx*(ny-1)-1,1)));\]
\[
\text{Kronek}(2:end,1:end-1)=\text{sparse(}\text{Kronek}(2:end,1:end-1)-\text{diag(}\mu_{hx}2\cdot \text{ones}(nx*(ny-1)-1,1)));\]
\[
\text{for jj=1:1:ny-2}
\text{jjj}=nx*jj;
\text{Kronek}(jjj,jjj)=\text{Kronek}(jjj,jjj)-\mu_{hx}2; \quad \% \text{V east correction}
\text{Kronek}(jjj+1,jjj)=0;
\text{Kronek}(jjj,jjj+1)=0;
\text{end};
\]
\[
\text{jjj}=nx*(ny-1);
\text{Kronek}(jjj,jjj)=\text{Kronek}(jjj,jjj)-\mu_{hx}2; \quad \% \text{U east correction}
\text{aa}=nx*(ny-2)+1;
\text{bb}=nx*(ny-1);
\text{Kronek}(aa:bb,aa:bb)=\text{sparse(}\text{Kronek}(aa:bb,aa:bb)-\text{diag(}\mu_{hy}2\cdot \text{ones}(nx,1)));\]
\[
\text{Kronek}(nx+1:end,1:aa-1)=\text{sparse(}\text{Kronek}(nx+1:end,1:aa-1)-\text{diag(}\mu_{hx}2\cdot \text{ones}(nx*(ny-2),1)));\]
\[
\text{Kronek}(1:aa-1,nx+1:end)=\text{sparse(}\text{Kronek}(1:aa-1,nx+1:end)-\text{diag(}\mu_{hx}2\cdot \text{ones}(nx*(ny-2),1)));\]
\[
\text{VNew}=\text{reshape(}(\text{Kronek}\text{\textbackslash RHSp}),nx,ny-1);\]
\[
\text{clear RHS Kronek};\]

\[
\%
\text{4.4 Chorin Pressure Projection Method (UNew, VMew, P)}\]
DU = diff([Uinf*ones(1,ny);UNew;UNew(end,:)]/(dx)+diff([Uinf*ones(nx,1)
VNew VNew(:,end)]')/(dy);

RHSp = -reshape((1/dt)*Rho.*DU,[],1); % LaplacianT added
Prhx2=Pref/dx^2;
Prhy2=Pref/dy^2;
RHSp(end-nx+1:end)=RHSp(end-nx+1:end)+Prhy2;
if all dirichlet
RHSp(1:lx)=RHSp(1:lx)+Prhx2;
for iii=1:lx:ny
RHSp(iii*lx)=RHSp(iii*lx)+Prhx2;
end
RHSp((iii-1)*lx+1)=RHSp((iii-1)*lx+1)+Prhx2;

% RHSp
p(perp) = (Rp\(Rpt\RHSp(perp)));
PNew = reshape(p,nx,ny);
UNew = ((avg(Rho)).*((avg(RhoNew)).^(-1))).*UNew*dt*diff(PNew)/dx;
% U(t:n+1)
VNew = ((avg(Rho')).*((avg(RhoNew')).^(-1))).*VNew-dt*diff(PNew')/dy;
% V(t:n+1)
fprintf('**** Pressure : ( %1.2s , %1.2s ) [Pa]/n',max(max(PNew)),max(max(PNew)));
fprintf('**** velocity component U : ( %1.2s , %1.2s ) [m/s]/n',min(min(UNew)),max(max(UNew)));
fprintf('**** velocity component V : ( %1.3f , %1.2s ) [m/s]/n',min(min(VNew)),max(max(VNew)));
clear RHSp

% 4.5 Combustion rate (r) (nx)*(ny)
indexFl=0;
for jj=1:ny
for ii=1:nx
kdFl = S(T(ii,jj),Xnew(ii,jj,1),Xnew(ii,jj,2)); % Kronecker delta Flame ignition
if (kdFl==1)
indexFl=indexFl+1;
% record flame domain
FxD(indexFl,:)=[ii jj];
end;
r(ii,jj) = -(kdFl*Ahc*T(ii,jj)*(RhoNew(ii,jj)^0.3)*(Xnew(ii,jj,1).^0.5)*Xnew(ii,jj,2)*exp(-Ta/T(ii,jj));
end;
end
fprintf('**** r ( %1.2s , %1.2s ) [1/s]/n',min(min(r)),max(max(r)));

% 4.6 Molar fractions (Xi) (nx)*(ny)
if (exist('Y0D')==1); indexY0=length(Y0D); end;
for jj=1:ny
for ii=1:nx
for s=1:4
Xsval = Xnew(ii,jj,s) - (dt*coef(s)/Ms(1))*r(ii,jj)*M(ii,jj);
if ((s==1)||(s==2))&(Xsval<Xe); Xsval=Xe;
% combustion rate correction in case of extinction
r(ii,jj)=(-Xnew(ii,jj,s)+Xsval)*Ms(1)/(dt*coef(s)*M(ii,jj));
end;
% record Zero Fuel condition
    if (s==1)&(Xsval==Xe); indexY0=indexY0+1;
    YOD(indexY0,:)=[ii,jj]; end;
    Xnew(ii,jj,s) = Xsval;
    end;
end;
fprintf('**** Molar fractions estimated: Xi ( %1.2s %1.2s , %1.2s %1.2s)
\n',min(min(Xnew(:,:,1:2))), max(max(Xnew(:,:,1:2))));

%%% 4.7 Mixture molar mass (M) (nx)*(ny)
    M = zeros(nx,ny);
    for s=1:5
        M = M + Xnew(:,:,s)*Ms(s);
    end;
    fprintf('**** mixture molar mass estimated: M ( %1.2s , %1.2s )
[g/mol]\n',min(min(M))*1000, max(max(M))*1000);

%%% 4.8 Mixture specific heat capacity (cp) (nx)*(ny)
    cpNew = zeros(nx,ny);
    for s=1:5
        cpNew = cpNew + (cps(s)*Ms(s))*Xnew(:,:,s).*M.^(-1));
    end;
    fprintf('**** Mixture specific heat capacity: cp ( %1.2s , %1.2s )
[J/kg/K]\n',min(min(cpNew)), max(max(cpNew)));

%%% 4.9 Specific combustion enthalpy (hc) (nx)*(ny)
    hcNew = zeros(nx,ny);
    for s=1:4
        hcNew = hcNew - coef(s)*(Hrefs(s)+(Ms(s)*cps(s))*(T-Tref)).*M.^(-1));
    end;
    hcNew = hcNew*enthalpy_reduction;
    fprintf('**** Specific combustion enthalpy: hc ( %1.2s , %1.2s )
[J/kg]\n',min(min(hcNew)), max(max(hcNew)));

%%% 4.10 Energy Balance (T) (nx)*(ny)
    invCp  = cp.^(-1);
    invRCp = (RhoNew.^(-1)).*invCp;
    Rhoex =[rho0*ones(1,ny) ; avg(Rho) ; rho0*ones(1,ny) ];
    Rhoey =[rho0*ones(nx,1)  avg(Rho')'  rho0*ones(nx,1) ];
    Tex  =[To*ones(1,ny) ; avg(T) ; To*ones(1,ny) ];
    Tey  =[To*ones(nx,1)  avg(T')'  To*ones(nx,1) ];
    hcex  =[hc(1,:) ; avg(hc) ; hc(end,:) ];
    hcey  =[hc(:,1)  avg(hc')'  hc(:,end) ];
    cpx =[cp0(1,1)*ones(1,ny) ; avg(cp) ; cp0(1,1)*ones(1,ny)];
    cpex=[cp0(1,1)*ones(nx,1) avg(cp')' cp0(1,1)*ones(nx,1)];
    Uex  =[Uinf*ones(1,ny) ; U ; U(end,:) ];
    Vex  =[Vinf*ones(nx,1) V ; V(end,:) ];
    dRCTUx = diff(Rhoex.*cpxex.*Tex.*Uex)/dx;
    dRCTVx = diff((Rhoex.*cpxex.*Tey.*Vex))'/dy;
    dRhcxUx = diff(Rhoex.*hcex.*Uex)/dx;
    dRhcxVy = diff((Rhoex.*hcex.*Vex))'/dy;
    dTx = diff(Tex)/dx;
    dTy = diff(Tey)'/dy;
    dTRad  = sigm*emit*invRCp.*((1/hz)*((Tref^4)*ones(nx,ny)-T.^4) +
    diff(4*(Tex.^3).*[dTx(1,:) ; avg(dTx) ; dTx(end,:)])+diff(4*(Tey.^3).*[dT
y(:,1) ; avg(dTy) ; dTy(:,end)])')))';

97
\[
cpex = [cp0(1,1)*\text{ones}(1,ny) \ ; \ cp \ ; \ cp0(1,1)*\text{ones}(1,ny)];
\]
\[
cpey = [cp0(1,1)*\text{ones}(nx,1) \ ; \ cp \ ; \ cp0(1,1)*\text{ones}(nx,1)];
\]
\[
Tex = [To*\text{ones}(1,ny) \ ; \ T \ ; \ To*\text{ones}(1,ny)];
\]
\[
Tey = [To*\text{ones}(nx,1) \ ; \ T \ ; \ To*\text{ones}(nx,1)];
\]
\[
hcex = [hc(1,:) \ ; \ hc \ ; \ hc(end,:)];
\]
\[
hcey = [hc(:,1) \ ; \ hc \ ; \ hc(:,end)];
\]

\[
kth = kth*\text{diff}([[\text{invCp}(1,:);\text{avg}(\text{invCp});\text{invCp}(end,:)]*\text{diff}(cpex.*Tex))/\text{dx}^2;
\]
\[
kty2 = kth*\text{diff}([[\text{invCp}(:,1)\text{avg}(\text{invCp})';\text{invCp}(:,end)]*\text{diff}(cpey.*Tey)')')'/\text{dy}^2;
\]
\[
kdhcx2 = kth*\text{diff}([[\text{invCp}(1,:);\text{avg}(\text{invCp});\text{invCp}(end,:)]*\text{diff}(hcex))/\text{dx}^2;
\]
\[
kdhcy2 = kth*\text{diff}([[\text{invCp}(:,1)\text{avg}(\text{invCp})';\text{invCp}(:,end)]*\text{diff}(hcey')')')'/\text{dy}^2;
\]
\[
Tbase = \text{invRCp.*}(\text{Rho.*cp.*T});
\]
\[
dTpressure = \text{invRCp.*}(\text{PNew-P})/\text{dt};
\]
\[
dT3Dconv = -\text{invRCp.*}(\text{Ca*(T-To)});
\]
\[
dTcombustion = -\text{invRCp.*}(\text{RhoNew.*hcNew.*r});
\]
\[
TNew = Tbase + \text{dt*}(\text{dTpressure} + \text{dT3Dconv} + \text{dTcombustion});
\]
\[
dTransport = -\text{invRCp.*}(\text{dRCTUx+dRCTVy+dRhcUx+dRhcVy});
\]
\[
TNew = TNew + \text{dt*Transport}; \ % enthalpy transport
\]
\[
dTdissipation = \text{invRCp.*}(\text{kdTx2+kdy2+kdhcx2+kdhcy2});
\]
\[
TNew = TNew + \text{dt*dTdissipation}; \ % diffusion
\]
\[
TNew = TNew + \text{dt*TRad}; \ % Radiation term
\]
\[
T = TNew;
\]

fprintf('** dT(P) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTpressure)), max(max(dTpressure)));
\]
\[
fprintf('** dT(hc) : ( 0 , %1.2s ) [K/s]
',max(max(dT3Dconv)));
\]
\[
fprintf('** dT(3D) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTransport)), max(max(dTransport)));
\]
\[
fprintf('** dT(U,V) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTdiffusion)), max(max(dTdiffusion)));
\]
\[
fprintf('** dT(Rad) : ( %1.2s , %1.2s ) [K/s]
',min(min(dTRad)), max(max(dTRad)));
\]
\[
fprintf('** T : ( %1.2s , %1.2s ) [K/s]
',min(min(T)), max(max(T)));
\]

%% Time update
\]
\[
dRhodt=(\text{RhoNew-Rho})/\text{dt};
\]
\[
fprintf('** dRho/dt : ( %1.2s , %1.2s ) [kg/m^3/s]
',min(min(dRhodt)), max(max(dRhodt)));
\]
\[
clear dRhodt
\]
\[
P=PNew;
\]
\[
Rho = RhoNew;
\]
\[
U = UNew;
\]
\[
V = VNew;
\]
\[
hc = hcNew;
\]
% 5. VISUALISATION

% Density plot
figure(1)
clf(1)
[hR hR]=contourf(avg(x),avg(y),(Rho)',levels);
set(hR,'LineStyle','none');
colormap(pink)
https://uk.mathworks.com/help/matlab/ref/colormap.html
caxis([0.3*rho0 rho0]);
colorbar
title(sprintf('Density field [kg/m^3] (colormap) \n Re = %0.1g \ t = %0.2g s',Re,k*dt))
axis equal, axis([0 lx 0 ly])
grid on, hold on

% Wind plot
Usg = [Uinf*ones(1,ny); avg(U); Uinf*ones(1,ny)];
Vsg = [Vinf*ones(nx,1) avg(V'); Vinf*ones(nx,1)];
Wind = (Usg.^2+Vsg.^2).^0.5;
Win = max(max(Wind));
for gg=1:length(xqv)
    for hh=1:length(yqv)
        QVX(gg,hh)=Usg(ggg,hhh)/WinMax;
        QVY(gg,hh)=Vsg(ggg,hhh)/WinMax;
    end
end;
hold on
[Xqv,Yqv]=meshgrid(xqv,yqv);
quiver(Xqv,Yqv,QVX',QVY',nqvx*0.15,'k-');
xlabel('x [m]')
ylabel('y [m]')
drawnow
frame = getframe(1);
im = frame2im(frame);
[imind,cm] = rgb2ind(im,256);
if k == 1;
imwrite(imind,cm,filename1,'gif','Loopcount',inf);
else
    imwrite(imind,cm,filename1,'gif','WriteMode','append');
end;
writeVideo(vid1,frame);

% Flame Domain Plot
figure(2)
clf(2)
grid on
axis equal, axis([0 lx 0 ly])
title(sprintf('Flame Domain (red) \n Zero Fuel Domain (green) \n Re = %0.1g \ t = %0.2g s',Re,k*dt))
if exist('FlD')==1
    if (length(FlD)>1)
        for indxx=1:length(FlD)
            FlDx=dx*(FlD(indxx,1)-1.5);
            FlDy=dy*(FlD(indxx,2)-1.5);
        end
    end
else
end
rectangle('Position',[FlDx,FlDy,dx,dy], 'FaceColor', 'r', 'EdgeColor', 'r', 'LineWidth', 1)
end  
end  
end;
% Zero Fuel domain 
if exist('Y0D')==1
  for indxx=1:1:length(Y0D)
    hold on
    Y0Dx=dx*(Y0D(indxx,1)-1.5);
    Y0Dy=dy*(Y0D(indxx,2)-1.5);
  rectangle('Position',[Y0Dx,Y0Dy,dx,dy], 'FaceColor', 'g', 'EdgeColor', 'g', 'LineWidth', 1)
  end  
end 
grid on 
xlabel('x [m]')
ylabel('y [m]')
drawnow 
frame = getframe(2); 
im = frame2im(frame); 
[imind,cm] = rgb2ind(im,256); 
if k == 1; 
  imwrite(imind,cm,filename2,'gif', 'Loopcount',inf); 
else 
  imwrite(imind,cm,filename2,'gif','WriteMode','append'); 
end; 
writeVideo(vid2,frame);

% Temperature Field Plot 
figure(3) 
clf(3) 
[hT hT]=contourf(x(1:end-1),y(1:end-1),T',levels); 
set(hT,'LineStyle','none'); 
colormap(jet) 
https://uk.mathworks.com/help/matlab/ref/colormap.html 
caxis([Tmin,Tmax+50]) 
colorbar %('AxisLocation','in') 
axis equal, axis([0 lx 0 ly]) 
title(sprintf('Temperature field [K] (colormap)\nAmbient temperature (green contour)\nPyrolysis temperature (red contour)\nRe = %0.1g t = %0.2g s',Re,k*dt)) 
hold on 
  contour(x(1:end-1),y(1:end-1),round(T)',[To To],'g-', 'LineWidth',1.5) 
hold on 
  contour(x(1:end-1),y(1:end-1),round(T)',[Tp Tp],'r-', 'LineWidth',1.5) 
xlabel('x [m]')
ylabel('y [m]')
drawnow 
frame = getframe(3); 
im = frame2im(frame); 
[imind,cm] = rgb2ind(im,256); 
if k == 1; 
  imwrite(imind,cm,filename3,'gif', 'Loopcount',inf); 
else 
  imwrite(imind,cm,filename3,'gif','WriteMode','append'); 
end;
writeVideo(vid3,frame);

% Temperature profile centre domain in x direction
figure(4)
xspace = avg((linspace(0,lx,nx+1))')';
clf(4)
g_y=[250:50:Tmax+100]; % user defined grid Y [start:spaces:end]
g_x=[dx:dx:lx-dx]; % user defined grid X [start:spaces:end]
for i=1:length(g_x)
    plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:') % y grid lines
    hold on
end
for i=1:length(g_y)
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:') % x grid lines
    hold on
end
hold on
plot(xspace,Tp*ones(length(xspace)),-'b',xspace,To*ones(length(xspace)'),'-.k','LineWidth',1.5);
hold on
plot(xspace,(T(:,round(ny/2))),'-r','LineWidth',2);
T_store(:,k+1)=T(:,round(ny/2));
save(nameT,'T_store');
% axis([0 lx Tm-50 Tmax+100]);
drawnow frame = getframe(4);
im = frame2im(frame);
[imind,cm] = rgb2ind(im,256);
imwrite(imind,cm,filename4,'gif','WriteMode','append');
writeVideo(vid4,frame);

% combustion energy, temperature, mass fractions in flame ignition
% point (centre of the flame)
if k>1
    figure(5)
    subplot(4,1,1)
    hold on
    plot(dt*[k-1 k],[Old(1) dTcombustion(nxig,nyig)*cpNew(nxig,nyig)*RhoNew(nxig,nyig)],'r','LineWidth',2)
    xlabel('$\text{Time}$\quad [\ s]$','interpreter','latex');
ylabel('$\dot{\text{Q}}$\quad [\ W/m^3]$','interpreter','latex');
    subplot(4,1,2)
    hold on
    plot(dt*[k-1 k],[Old(2) T(nxig,nyig)],'m','LineWidth',2);
    xlabel('$\text{Time}$\quad [\ s]$','interpreter','latex');
ylabel('$\text{Temperature}$\quad [\ K]$','interpreter','latex');
    subplot(4,1,3)
    hold on
    plot(dt*[k-1 k],[Old(3) Xnew(nxig,nyig,1)],'b','LineWidth',2);
    xlabel('$\text{Time}$\quad [\ s]$','interpreter','latex');
ylabel('$\text{X}_{\text{fuel}}$\quad [\ 1]$','interpreter','latex');
    subplot(4,1,4)
    hold on
    plot(dt*[k-1 k],[Old(4) Xnew(nxig,nyig,2)],'k','LineWidth',2);
xlabel('Time $s$', 'interpreter', 'latex');
ylabel('X oxidizer $1$', 'interpreter', 'latex');
end;
Old=[dTcombustion(nxig,nyig) T(nxig,nyig) Xnew(nxig,nyig,1)
Xnew(nxig,nyig,2)];
diary('LOG.txt')
clear F1D
fprintf('
');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
end;
%       figure(13)
%       savefig('heat_combustion.fig');
close(vid1);
close(vid2);
close(vid3);
close(vid4);
%       figure(17)
%       title('END')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%% 6. SUB-ROUTINES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% Average matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function B = avg(A,k)
    if nargin<2, k = 1; end
    if size(A,1)==1, A = A'; end
    if k<2
        B = (A(2:end,:)+A(1:end-1,:))/2;
    else
        B = avg(A,k-1);
    end
    if size(A,2)==1, B = B'; end

% Kroneker matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = K1(n,h,a11)
    % all: Neumann=1, Dirichlet=2, Dirichlet mid=3;
    A = spdiags([-1 a11 0;ones(n-2,1)*[-1 2 -1];0 all -1],-
        1:1,n,n)'/h^2;

% Ignition Condition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function spread = S(T,X,Xo)
    global Tp Xe Xo2e
    if (T>Tp) & (X>Xe) & (Xo>Xo2e) spread = 1;
    else spread = 0; end;
APPENDIX C: SCRIPT 3

The following Matlab script is original and it simulates the Model C explained in Section 3.5.

```matlab
function Flame2D_MOD_C

% Flame2D_MOD_C: Flame Spread using "Runge Kutta 4th order" solver

% 07/2017
% by Paolo Grasso
% paologk90@gmail.com
% grassop@uni.coventry.ac.uk

% CONTENTS:
% NOMENCLATURE
% - Fluid dynamics; - Thermal Energy and Chemistry;
% - Discretisation; - Setting Parameters;
% - Visualization Settings.

% 1. CONSTANTS
% 2. Setting PARAMETERS
% 3. INITIALIZATION
% 4. SIMULATION
% 4.1 "a" calculation (U,V,T,Rho)
% 4.2 "b" calculation (U,V,T,Rho)
% 4.3 "c" calculation (U,V,T,Rho)
% 4.4 "d" calculation (U,V,T,Rho)
% 4.5 RK 4th order approximated solution
% 4.6 Update other properties (Xs,M,cp,hc)

% 5. VISUALISATION
```

103
clc
close all
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                  NOMENCLATURE                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  % Fluid dynamics                                           %
% Re    : Reynolds' number [m/s]                                   %
% U     : velocity component in x direction (inner domain) [m/s]   %
% V     : velocity component in y direction (inner domain) [m/s]   %
% mu    : viscosity [kg/m/s]                                       %
% ni    : kinematic viscosity [m^2/s]                              %
% rho   : gas density [kg/m^3]                                     %
% Po    : reference pressure [Pa]                                  %
% P     : specific pressure [m^2/s^2]                              %
% Pold  : previous time step pressure field [m^2/s^2]              %
% uS    : South Dirichlet b.c. for u wind component [m/s]          %
% vS    : South Dirichlet b.c. for v wind component [m/s]          %
% uW    : West Dirichlet b.c. for u wind component [m/s]           %
% vW    : West Dirichlet b.c. for v wind component [m/s]           %
% Ua    : averaged along x velocity field (x component) [m/s]      %
% Va    : averaged along y velocity field (y component) [m/s]      %
% Ue    : extended velocity field (x component) with D.b.c. [m/s]  %
% Ve    : extended velocity field (y component) with D.b.c. [m/s]  %
% UVx   : 1st derivative in x of the (UV) matrix [m/s^2]           %
% UVy   : 1st derivative in y of the (UV) matrix [m/s^2]           %
% U2x   : 1st derivative in x of (U^2) matrix [m/s^2]              %
% V2y   : 1st derivative in y of (V^2) matrix [m/s^2]              %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  % Thermal Energy and Chemistry                              %
% R     : gas constant [J/mol/K]                                   %
% M     : molar mass mixture [kg/mol]                              %
% To    : ambient temperature [K]                                 %
% Tp    : pyrolysis temperature [K]                               %
% BHF   : specific heat of formation [Btu/lb]                      %
% Hf    : specific heat of formation [J/kg]                       %
% r     : rate of combustion [1/s]                                %
% mHR   : molar heat rate of combustion [J/mol/s]                  %
% mH    : molar heat of combustion [J/mol]                        %
% Hr    : heat rate of combustion [J/s]                           %
% cp    : specific heat capacity at constant pressure [J/kg/K]     %
% Ca    : diffusion coefficient in atm (turbulent?) [?]           %
% sigma : Boltzman constant [W/m^2/K^4]                           %
% emit  : emittance (ideal black body) [W/m/K]                     %
% kth   : thermal conductivity constant [W/m/K]                   %
% Ea    : activation energy for HC [J]                            %
% Ahc   : pre-exponensial factor in Arrhenius [depends?]          %
% Wf    : flame width [m]                                         %
% dist2 : squared radial distance from the ignition point [m]     %
% coef  : vector of coeff. balance chemical equilibrium each species %
% Mi    : vector of molar mass each species [g/mol]                %
%Href   : vector of partial molar formation enthalpies ref. [J/mol] %
% cpi   : vector of specific heat capacity each species [J/kg/K]   %
% Xio   : vector of mole fraction each species initial            %

104
% Xi    : 3D matrix mole fraction

% pedixes:
% _ch   : CxHy (hidrocarbon)
% _o    : O2   (oxigen)
% _co   : CO2  (carbon dioxide)
% _ho   : H2O  (water)
% _a    : Air
% ref   : reference

% Discretisation
% lx    : length of domain                                  [m]
% ly    : width  of domain                                  [m]
% tf    : final time of simulation                          [s]
% dx    : length of unit cell                               [m]
% dy    : width  of unit cell                               [m]
% Vol   : unit cell volume fraction refered to the domain   
% Lp    : laplacian kroneker matrix for pressure           
% perp  : premutation for sparser matrix                    
% Rp    : Cholesky triangular permutated matrix             
% Lu    : laplacian kroneker matrix for pressure           
% peru  : premutation for sparser matrix                    
% Ru    : Cholesky triangular permutated matrix             
% perv  : premutation for sparser matrix                    
% Rv    : Cholesky triangular permutated matrix             
% CFL   : Courant Friedrichs Lewy value                    
% PecW  : Peclet number for N.S.                           
% PecT  : Peclet number for energy                          
% Froid : Froid number                                     

% Setting Parameters
% Xig   : x Ignition Point                                  [m]
% Yig   : y Ignition Point                                  [m]
% sigmaT: deviation for Gaussian distribution               
% Uinf  : asymptotic velocity component in x direction     [m/s]
% Vinf  : asymptotic velocity component in y direction     [m/s]
% alpha : CFL condition value                               (<1)
% dtmax : maximum time step allowable                       [s]
% nx    : number of cells in x direction                    
% ny    : numer of cells in y direction                     
% nsteps: number of output visualised time steps            
% Ye    : extinction mass fraction                          

% Visualization Settings
% Pixscreen : number of screen's pixels                     
% Nqvx   : N vectors for qiver in x direction
% nqvx   : rounded n vectors for qiver in x dir.
% xqv    : x qiver space                                    
% Nqvy   : N vectors for qiver in y direction
% nqvy   : rounded n vectors for qiver in y dir.
% yqv    : y qiver space                                    
% pmin   : caxis minimum specific pressure                  [m^2/s^2]
% pmax   : caxis maximum specific pressure                  [m^2/s^2]
% levels: number of contours levels                         
% Tmin   : caxis minimum Temperature                        [K]
% Tmax   : caxis maximum Temperature                        [K]
% 1. CONSTANS

%$$$$$$

global Tp Ye Xe Xo2e
global R  Uinf Vinf mu
global Tref coef Hrefs Pref Po Ms Y0D Xso F1D
global To dt kth sigm emit Ahc hc0 rho0 cp0
global Ca Ea Ta phi
global nx ny nt dx dy dz l2D
global dTcomb dTUx dTVy dhcUx dhcVy dTPt Tdiffx Tdiffy Hcdiffx Hcdiffy
dTRadx dTRady dTRad3 dTCa

mu = 18.27e-6; % air at 291.15 K (it should be higher
ni = 4e-5; % kinematic viscosity [m^2/s] from 15.03e-6 to 62.53e-6
rho = 1.2; % 0.9; % air density [kg/m^3] from 1.204 to 0.524(400°C)
R = 8.314; % gas constant [J/mol/K]
M = 28.9644e-3; % molar mass dry air [kg/mol]
tf = 8e-0; % final time
lx = 5;
ly = 5;

% for enthalpy
To = 293.14; % atm temperature [K]
Tp = 600; % autoignition Temperature 573; % minimum pyrolysis
temperature (200-300°C for wood)

BHf = 8660; % Btu/lb = kJ/kg× 2.326
hc = 1000*BHf/2.326; % heat of formation [J/kg]
molHf = 802.301e3; % combustion of CH4 [J/mol]
Tref = 298.15; % [K]
Pref = 101325; % [Pa]

phi = 1;

% CH4     O2     CO2     H2O     air
coef = [ -1   -2     1     2     ];
Ms = [ 16.043 32 44.011 18.016 28.9644]; % g/mol
Ms = Ms*0.001; % kg/mol
Hrefs = [-74.87 0 0 0]; % kJ/mol
Hrefs = Hrefs*1000; % J/mol
cps = [ 2220 906 844 1996 1010]; % J/kg/K

Xso = [ 0.1 0.032 0.008 0.001 0.759];
Xso(2:end) = Xso(2:end).*(1-Xso(1));
% rho = Pref*sum(Ms.*Xso)/(R*To);

% 2. SETTING PARAMETERS

Xig = 2*lx/5;
Yig = ly/2;
sigmaT = 0.07;
\[ \sigma_T = \sigma_T \times l_x; \]

\[ N_{qvx} = 30; \]
\[ N_{qvy} = 30; \]

\[ U_{inf} = 0; \]
\[ V_{inf} = 0; \]

\[ n_x = 300; \quad \% \text{max 130} \]
\[ n_y = 300; \quad \% \text{max 130} \]

\[ p_{min} = -1000; \]
\[ p_{max} = 1000; \]
\[ \text{levels} = 30; \]
\[ T_{min} = 300; \]

\[ Y_e = 0.2; \]
\[ X_e = 0.0268; \]
\[ X_{o2e} = 0.0268; \]

\[ \alpha = 0.9; \quad \% < 1 \text{ (for cfl)} \]
\[ k_{th} = 0.096; \quad \% \text{thermal conductivity constant (between 0.0257 W/(m*K)m} \]
\[ \text{and 0.0515 (400°C) 0.0765 (650°C)} \]
\[ C_a = 100.6; \quad \% \text{diffusion coefficient in atm (turbulent?)} \]
\[ d_{tmax} = 2.5 \times 10^{-2}; \]
\[ d_z = 0.1; \]

\[ A_{hc} = (1 \times 10^{-7}) \times 2.07 \times 10^4; \quad \% \text{(new CH4) 2.07e4;(previous) 2.07e4/aaa; \% pre-e factor for fuel consumption HC} \]
\[ E_a = 35000; \quad \% \text{(new CH4) 1.917e7;(previous) 1.917e7; \% activation energy for HC: 1.917e7 cal/mol (1 cal = 4.184 J)} \]
\[ T_a = 900; \quad \% \text{800; \% autoignition temperature} \]
\[ T_o = 293.15; \quad \% \text{atm temperature [K]} \]
\[ T_p = 500; \quad \% \text{600; \% autoignition Temperature 573; \% minimum pyrolysis temperature (200-300°C for wood)} \]
\[ \overline{T} = 50; \quad \% 80; \]
\[ \text{enthalpy reduction} = 1; \quad \% \text{scale energy} \]
\[ T_{max} = 750; \quad \% \text{ceil}(T_a/50) \times 50+50; \%1200; \quad \% \text{autoignition Temperature 573; \% minimum pyrolysis temperature (200-300°C for wood)} \]
\[ \overline{T} = 50; \quad \% 80; \]

%% 3. INITIALIZATION

fprintf(' 
')
fprintf(' INITIALIZATION 
')

% .GIF and .AVI initialisation
filename1 = 'Density.gif';
filename2 = 'Flame_Spread.gif';
filename3 = 'Temperature_Field.gif';
filename4 = 'Temperature_Profile.gif';
vid1 = VideoWriter('Density.avi');
vid2 = VideoWriter('Flame_Spread.avi');
vid3 = VideoWriter('Temperature_Field.avi');
vid4 = VideoWriter('Temperature_Profile.avi');
vid1.FrameRate = 5; % Default 30
vid1.Quality = 100; % Default 75
vid2.FrameRate = 5; % Default 30
vid2.Quality = 100; % Default 75
vid3.FrameRate = 5; % Default 30
vid4.FrameRate = 5; % Default 30
vid3.Quality = 100;  % Default 75
vid4.FrameRate = 5;  % Default 30
vid4.Quality = 100;  % Default 75
open(vid1);
open(vid2);
open(vid3);
open(vid4);
fprintf('%###');  % ASCII alt+254;

% qiver initialisation
x = linspace(0,lx,nx+1); dx = lx/nx;
% nqvx = round(lx/Nqvx/dx);
% xqv = avg((0:dx*nqvx:lx)')';
y = linspace(0,ly,ny+1); dy = ly/ny;
% nqvy = round(ly/Nqvy/dy);
% yqv = avg((0:dy*nqvy:ly)')';
(nxig = round(Xig/dx)+1;
(nyig = round(Yig/dy)+1;
[X,Y] = meshgrid(y,x);

% time discretisation
dt = alpha*min([dx/Uinf,dy/Vinf,dtmax]);
nt = ceil(tf/dt); dt = tf/nt;

% Velocity initial conditions
U = Uinf*ones(nx,ny);
V = Vinf*ones(nx,ny);
Re = ceil(max(Uinf,Vinf)*lx/ni);
fprintf('%###');

% Fuel mass fraction
Xs = zeros(nx,ny,5);
for tt=1:1:5
    Xs(:,:,tt) = Xso(tt);
end;

% mixture molar mass estimation
M = zeros(nx,ny);
for s=1:1:5
    M = M + Xs(:,:,s)*Ms(s);
end;
Mf = Ms(s)*Xs(:,:,1)./M;

% Temperature starter ("Mercer and Weber 1997") Gaussian distribution
% around ignition point
T = To*ones(nx,ny);
for cx1=1:1:nx
    for cx2=1:1:ny
        dist2 = (((cx1-nxig)*dx)^2+((cx2-nyig)*dy)^2);
        T(cx1,cx2)=T(cx1,cx2)+(Tp+overT-To)*exp(-dist2/sigmaT^2);
    end
end;

% Density
\begin{verbatim}
Rho = (Pref/R) * M. *(T. ^(-1));
RhoNew = Rho;
rho0=Rho(1,1);

% heat capacity
    cp = zeros(nx,ny);
    for s=1:1:5
        cp = cp + Xs(:,s) * Ms(s) * cps(s);
    end;
    cp = cp./M;
cp0=cp(1,1);
fprintf('###

% Combustion enthalpy estimation
    hc = zeros(nx,ny);
    for s=1:1:4
        hc = hc - coef(s) * (Hrefs(s)+(Ms(s) * cps(s)) * (T-Tref)).* (M. ^(-
1));
    end;
hc=hc*enthalpy_reduction;
hc0=hc;
r = zeros(nx,ny);

% Figures' sizes
Pixcreen = get(0,'screensize');
% figure(1) % Density Plot
FigHandle1 = figure(1);
% set(FigHandle1, 'Position', [+20, Pix_SS(4)/2+20, 
Pix_SS(3)/2-40, Pix_SS(4)/2-120]);
set(FigHandle1, 'Position', [20, 40, Pixcreen(3)-40, 
Pixscreen(4)-140]);
% figure(2) % Flame Spread Plot
FigHandle2 = figure(2);
% set(FigHandle2, 'Position', [20, 40, Pixscreen(3)-40, 
Pixscreen(4)-140]);
% figure(3) % Temperature Field Plot
FigHandle3 = figure(3);
set(FigHandle3, 'Position', [20, 40, Pixscreen(3)-40, 
Pixscreen(4)-140]);
% figure(4) % Temperature Profile half domain Plot
FigHandle4 = figure(4);
set(FigHandle4, 'Position', [20, 40, Pixscreen(3)-40, 
Pixscreen(4)-140]);
% figure(5) % Combustion enthalpy, temperature, mass fractions in the
% centre of the flame
FigHandle5 = figure(5);
set(FigHandle5, 'Position', [20, 40, Pixscreen(3)-40, 
Pixscreen(4)-140]);
clear Pixscreen
fprintf('###

CFL=max(abs(Uinf),0.001)*dt/dx;
PecT=dx*max(abs(Uinf),0.001)/kth;
Froid = 0.01/(9.81*1*(850-To)/850);
fprintf(' dx = %1.2s m    dy = %1.2s m 
',dx, dy);
fprintf(' dt = %1.2s s\n',dt);
fprintf(' CFL= %1.2s    Peclet= %1.2s Froid= 
%1.2s\n',CFL,PecT,Froid);

CFL=max(abs(Uinf),0.001)*dt/dx;
PecT=dx*max(abs(Uinf),0.001)/kth;
Froid = 0.01/(9.81*1*(850-To)/850);
fprintf(' dx = %1.2s m    dy = %1.2s m 
',dx, dy);
fprintf(' dt = %1.2s s\n',dt);
fprintf(' CFL= %1.2s    Peclet= %1.2s Froid= 
%1.2s\n',CFL,PecT,Froid);

\end{verbatim}
% figure(5) t=0
k=0;
figure(4)
xspace = avg((linspace(0, lx, nx+1))')';  
clf(5)
g_y=[250:50:Tmax+100]; % user defined grid Y [start:spaces:end]
g_x=[dx:dx:lx-dx]; % user defined grid X [start:spaces:end]
for i=1:length(g_x)
    plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:') %y grid lines
    hold on
end
for i=1:length(g_y)
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:') %x grid lines
    hold on
end
hold on
title(sprintf('Temperature profile (half domain section)\nRe = %0.1g  t = %0.2g s',Re,k*dt))
xlabel('x [m]');
ylabel('T [K]');
plot(xspace,Tp*ones(length(xspace)),'-b',xspace,To*ones(length(xspace)),'-k','LineWidth',1.5);
hold on
plot(xspace,(T(:,round(ny/2))),'-r','LineWidth',2);
T_store(:,1)=T(:,round(ny/2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% 4. SIMULATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Po = Pref;
P = Po*ones(nx,ny);
rho0 = Rho(1,1);
Ur = U.*Rho;
Vr = V.*Rho;
Tr = T.*Rho.*cp;
Pt = zeros(nx,ny);
l2D = dx;
r = F_Yf(Rho,Tr,cp,Xs(:,:,1),Xs(:,:,2));
fprintf('initial Temperature : T    ( %1.1f ,  %1.1f )
[K],min(min(Tr./Rho./cp)), max(max(Tr./Rho./cp)));
fprintf('initial Density     : Rho    ( %1.2f ,  %1.2f )
[kg/m^3],min(min(Rho)), max(max(Rho)));
fprintf('initial Pressure    : P    ( %1.0f ,  %1.0f )
[Pa],min(min(P)), max(max(P)));
fprintf('initial M   ( %1.2s , %1.2s )
[g/mol],min(min(M*1000)),max(max(M*1000)))

% figure(5) t=0
k=0;
figure(4)
xspace = avg((linspace(0, lx, nx+1))')';
clf(5)
g_y=[250:50:Tmax+100]; % user defined grid Y [start:spaces:end]
g_x=[dx:dx:lx-dx]; % user defined grid X [start:spaces:end]
for i=1:length(g_x)
    plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:') %y grid lines
    hold on
end
for i=1:length(g_y)
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:') %x grid lines
    hold on
end
hold on
title(sprintf('Temperature profile (half domain section)\nRe = %0.1g  t = %0.2g s',Re,k*dt))
xlabel('x [m]');
ylabel('T [K]');
plot(xspace,Tp*ones(length(xspace)),'-b',xspace,To*ones(length(xspace)),'-k','LineWidth',1.5);
hold on
plot(xspace,(T(:,round(ny/2))),'-r','LineWidth',2);
T_store(:,1)=T(:,round(ny/2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% 4. SIMULATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Po = Pref;
P = Po*ones(nx,ny);
rho0 = Rho(1,1);
Ur = U.*Rho;
Vr = V.*Rho;
Tr = T.*Rho.*cp;
Pt = zeros(nx,ny);
l2D = dx;
r = F_Yf(Rho,Tr,cp,Xs(:,:,1),Xs(:,:,2));
fprintf('initial Temperature : T    ( %1.1f ,  %1.1f )
[K],min(min(Tr./Rho./cp)), max(max(Tr./Rho./cp)));
fprintf('initial Density     : Rho    ( %1.2f ,  %1.2f )
[kg/m^3],min(min(Rho)), max(max(Rho)));
fprintf('initial Pressure    : P    ( %1.0f ,  %1.0f )
[Pa],min(min(P)), max(max(P)));
fprintf('initial M   ( %1.2s , %1.2s )
[g/mol],min(min(M*1000)),max(max(M*1000)))

110
% 4.1 "a" calculation

a(:,:,2) = Rho.*F_Ur(Rho, Ur, Vr, P);
a(:,:,3) = Rho.*F_Vr(Rho, Ur, Vr, P);
a(:,:,4) = dT.*F_Tr(Rho, Ur, Vr, Tr, Pt, hc, r, cp);

% combustion rate correction in case of extinction

ra = F_Yf(Rho, Tr, cp, Xs(:,:,1), Xs(:,:,2));
Xsa = Xs;
for jj=1:1:ny
    for ii=1:1:nx
        for s=1:1:4
            Xval = Xs(ii,jj,s) -
            0.5*(dT*coef(s)/Ms(1))*ra(ii,jj,s) -
            % combustion rate correction in case of extinction
\[ ra(ii,jj) = \frac{Xsa(ii,jj,s) - Xval}{Ms(1) / (dt*coef(s)*M(ii,jj))}; \]

end;

% record Zero Fuel condition
Xsa(ii,jj,s) = Xval;
end;
end;
end;

a(:,:,5) = dt*ra;
Ma = zeros(nx,ny);
for s=1:1:5
    Ma = Ma + Xsa(:,:,s)*Ms(s);
end;

cpa = zeros(nx,ny);
for s=1:1:5
    cpa = cpa + (cps(s)*Ms(s))*Xsa(:,:,s)./Ma;
end;

hca = zeros(nx,ny);
for s=1:1:4
    hca = hca - coef(s)*((Hrefs(s)+Ms(s)*cps(s))*((Tr+0.5*a(:,:,4))./(Rho+0.5*a(:,:,1))./cpa - Tref)).*(Ma.^(\(-1\)));
end;

hca = hca*enthalpy_reduction;
fprintf('$$$ a rho ( %1.2f , %1.2f )\[kg/m^3\]n',min(min(Rho+0.5*a(:,:,1))),max(max(Rho+0.5*a(:,:,1))));
fprintf('$$$ a ra ( %1.2s , %1.2s )\[1\]n',min(min(ra)),max(max(ra)));
fprintf('$$$ a Xf   ( %1.2s , %1.2s )\[1\]n',min(min(Xsa(:,:,1))),max(max(Xsa(:,:,1))));
fprintf('$$$ a Xo2  ( %1.2s , %1.2s )\[1\]n',min(min(Xsa(:,:,2))),max(max(Xsa(:,:,2))));
fprintf('$$$ a Xco2 ( %1.2s , %1.2s )\[1\]n',min(min(Xsa(:,:,3))),max(max(Xsa(:,:,3))));
fprintf('$$$ a Xh2o ( %1.2s , %1.2s )\[1\]n',min(min(Xsa(:,:,4))),max(max(Xsa(:,:,4))));
fprintf('$$$ a Xair ( %1.2s , %1.2s )\[1\]n',min(min(Xsa(:,:,5))),max(max(Xsa(:,:,5))));
fprintf('$$$ a M  ( %1.2s , %1.2s )\[g/mol\]n',min(min(Ma*1000)),max(max(Ma*1000)));
fprintf('$$$ a cp ( %1.2s , %1.2s )\[J/kg/K\]n',min(min(cpa)),max(max(cpa)));
fprintf('$$$ a hc ( %1.2s , %1.2s )\[J/kg\]n',min(min(hca)),max(max(hca)));
fprintf('$$** a T ( %1.1f , %1.1f )\n',min(min((Tr+0.5*a(:,:,4))./(Rho+0.5*a(:,:,1))./cpa)),max(max((Tr+0.5*a(:,:,4))./(Rho+0.5*a(:,:,1))./cpa)));

fprintf('---------------------\n');

\( \text{Rhoa} = \text{F_density}(Tr+0.5*a(:,:,4),Ma,cpa,Rho,P); \)

% 4.2 "b" calculation
b(:,:,2) = dt*F_Ur(Rhoa,Ur+0.5*a(:,:,2),Vr+0.5*a(:,:,3),P);
b(:,:,3) = dt*F_Vr(Rhoa,Ur+0.5*a(:,:,2),Vr+0.5*a(:,:,3),P);
b(:,:,4) = dt*F_Tr(Rho+0.5*a(:,:,1),Ur+0.5*a(:,:,2),Vr+0.5*a(:,:,3),Tr+0.5*a(:,:,4),Pt,ra,cpa);
fprintf('dTcomb dTUx dTVy dhcUx dhcVy dTPt Tdiffx Tdiffy Hcdiffx Hcdiffy dTRadx dTRady dTRad3 dTCa\n');
fprintf('1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f 1.2f\n',max(max(dTcomb./Rho./cp)),max(max(dTUx./Rho./cp)),max(max(dTVy./Rho./cp)),max(max(dTPt./Rho./cp))\n');
\[
\begin{align*}
\text{fprintf}\left(\%1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ \%1.2f \ %1.2f \ n', \min\left(\min\left(dTcomb./Rho./cp\right), \min\left(dTUx./Rho./cp\right), \min\left(dhcxUx./Rho./cp\right), \min\left(dhcvVy./Rho./cp\right), \min\left(dTPt./Rho./cp\right), \min\left(Tdiffx./Rho./cp\right), \min\left(Hcdiffx./Rho./cp\right), \min\left(Hcdiffy./Rho./cp\right), \min\left(dTRadx./Rho./cp\right), \min\left(dTRady./Rho./cp\right), \min\left(dTRad3./Rho./cp\right), \min\left(dTCa./Rho./cp\right)\right)\right); \\
rb = F_Yf(Rho+0.5*a(:,:,1),Tr+0.5*a(:,:,4),cp,Xs(:,:,1),Xs(:,:,2)); \\
\text{clear} \ Xsa \\
Xsb = Xs; \\
\text{for} \ jj=1:1:ny \\
\hspace{1cm} \text{for} \ ii=1:1:nx \\
\hspace{2cm} \text{for} \ s=1:1:4 \ % only for fuel and oxigen (partial for RK steps) \\
\hspace{3cm} Xval = Xs(ii,jj,s) - 0.5*(dt*coef(s)/Ms(1))*rb(ii,jj)*M(ii,jj); \\
\hspace{3cm} \text{if} \ (s=1) \& (Xval<Xe); Xval=Xe; \\
\hspace{4cm} \% combustion rate correction in case of extinction \\
\hspace{3cm} rb(ii,jj)=(Xsb(ii,jj,s)-Xval)*Ms(1)/(dt*coef(s)*M(ii,jj)); \\
\hspace{2cm} \text{end}; \\
\hspace{2cm} \% record Zero Fuel condition \\
\hspace{3cm} Xsb(ii,jj,s) = Xval; \\
\hspace{1cm} \text{end}; \\
\hspace{1cm} \text{end}; \\
\hspace{1cm} b(:,:,5) = dt*rb; \\
\hspace{1cm} Mb = zeros(nx,ny); \\
\hspace{2cm} \text{for} \ s=1:1:5 \\
\hspace{3cm} Mb = Mb + Xsb(:,:,s)*Ms(s); \\
\hspace{1cm} \text{end}; \\
\hspace{1cm} cpb = zeros(nx,ny); \\
\hspace{2cm} \text{for} \ s=1:1:5 \\
\hspace{3cm} cpb = cpb + (cps(s)*Ms(s))*Xsb(:,:,s)./Mb; \\
\hspace{1cm} \text{end}; \\
\hspace{1cm} hcb = zeros(nx,ny); \\
\hspace{2cm} \text{for} \ s=1:1:4 \\
\hspace{3cm} hcb = hcb - coef(s)*(Hrefs(s)+Ms(s)*cps(s)*((Tr+0.5*b(:,:,4))./(Rho+0.5*b(:,:,1))./cpb-Tref)).*(Mb.^(^-1)); \\
\hspace{2cm} \text{end}; \\
\hspace{1cm} hcb = hcb*enthalpy_reduction; \\
\text{fprintf}\left(\%1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ %1.2f \ \%1.2f \ %1.2f \ n', \min\left(\min\left(Rho+0.5*b(:,:,1)\right), \max\left(Rho+0.5*b(:,:,1)\right)\right)\right); \\
\text{fprintf}\left(\%1.2f \ %1.2f \ %1.2f \ n', \min\left(\min\left(rb\right), \max\left(rb\right)\right)\right); \\
\text{fprintf}\left(\%1.2f \ %1.2f \ \%1.2s \ %1.2s \ %1.2s \ %1.2s \ %1.2s \ n', \min\left(\min\left(Xf\right), \max\left(Xf\right)\right)\right); \\
\text{fprintf}\left(\%1.2f \ %1.2f \ \%1.2s \ %1.2s \ %1.2s \ %1.2s \ %1.2s \ n', \min\left(\min\left(Xo2\right), \max\left(Xo2\right)\right)\right); \\
\text{fprintf}\left(\%1.2f \ %1.2f \ %1.2s \ %1.2s \ %1.2s \ %1.2s \ %1.2s \ n', \min\left(\min\left(Xco2\right), \max\left(Xco2\right)\right)\right); \\
\end{align*}
\]
fprintf('$$$ b Xh2o ( %1.2s , %1.2s )
',min(min(Xsb(:,:,4))),max(max(Xsb(:,:,4))));
fprintf('$$$ b Xair ( %1.2s , %1.2s )
',min(min(Xsb(:,:,5))),max(max(Xsb(:,:,5))));
fprintf('$$$ b M ( %1.2s , %1.2s )
[g/mol]
',min(min(Mb*1000)),max(max(Mb*1000)));
fprintf('$$$ b cp ( %1.2s , %1.2s )
[J/kg/K]
',min(min(cpb)),max(max(cpb)));
fprintf('$$$ b hc ( %1.2s , %1.2s )
[J/kg]
',min(min(hcb)),max(max(hcb)));
fprintf('$$$ b T ( %1.1f , %1.1f )
[K]
',min(min((Tr+0.5*b(:,:,4))./(Rho+0.5*b(:,:,1))./cpb)),max(max((Tr+0.5*b(:,:,4))./(Rho+0.5*b(:,:,1))./cpb)));
fprintf('------------------
');
% 4.3 "c" calculation
% c(:,:,2) = dt*F_Ur(Rhob,Ur+0.5*b(:,:,2),Vr+0.5*b(:,:,3),P);
% c(:,:,3) = dt*F_Vr(Rhob,Ur+0.5*b(:,:,2),Vr+0.5*b(:,:,3),P);
% c(:,:,4) =
% dt*F_Tr(Rho+0.5*b(:,:,1),Ur+0.5*b(:,:,2),Vr+0.5*b(:,:,3),Tr+0.5*b(:,:,4),P);
% fprintf('dTcomb    dTUx     dTVy   dhcUx   dhcVy   dTPt  Tdiffx  Tdiffy  Hcdiffx Hcdiffy  dTRadx  dTRady  dTRad3   dTCa
% %1.2f   %1.2f   %1.2f   %1.2f   %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f
% %1.2f
% %1.2f   %1.2f   %1.2f   %1.2f   %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f    %1.2f
% %1.2f\n',max(max(dTcomb./Rho./cp)),max(max(dTUx./Rho./cp)),max(max(dTVy./Rho./cp)),max(max(dhcUx./Rho./cp)),max(max(dhcVy./Rho./cp)),max(max(dTdiffx./Rho./cp)),max(max(dTdiffy./Rho./cp)),max(max(Hcdiffx./Rho./cp)),max(max(Hcdiffy./Rho./cp)),max(max(dTRadx./Rho./cp)),max(max(dTRady./Rho./cp)),max(max(dTRad3./Rho./cp)),max(max(dTCa./Rho./cp)));
% fprintf('------------------
');
% rc =
% F_Yf(Rho+0.5*b(:,:,1),Tr+0.5*b(:,:,4),P);
% clear Xsb hcb rb cpb
% rc =
% F_Yf(Rho+0.5*b(:,:,1),Tr+0.5*b(:,:,4),P,cbp);
% clear Xsb hcb rb cpb
Xsc = Xs;
for jj=1:1:ny
  for ii=1:1:nx
    for s=1:1:4 % only for fuel and oxigen (partial for RK steps)
      Xval = Xs(ii,jj,s) -
          (dt*coef(s)/Ms(1))*rc(ii,jj)*M(ii,jj);
      if (s==1)&(Xval<Xe); Xval=Xe; end;
      % combustion rate correction in case of extinction
      rc(ii,jj)=(Xsc(ii,jj,s)-Xval)*Ms(1)/(dt*coef(s)*M(ii,jj));
      Xsc(ii,jj,s) = Xval;
    end
  end
end
114
\[ c(:,:,5) = dt*rc; \]
\[ Mc = zeros(nx,ny); \]
\[ \text{for } s=1:1:5 \]
\[ Mc = Mc + Xsc(:,:,s)*Ms(s); \]
\[ \text{end}; \]
\[ cpc = zeros(nx,ny); \]
\[ \text{for } s=1:1:5 \]
\[ cpc = cpc + (cps(s)*Ms(s))*Xsc(:,:,s)./Mc; \]
\[ \text{end}; \]
\[ hcc = zeros(nx,ny); \]
\[ \text{for } s=1:1:4 \]
\[ hcc = hcc - coef(s)*(Hrefs(s)+Ms(s)*((Tr+0.5*c(:,:,4))./(Rho+0.5*c(:,:,1))./cpc - Tref)).*(Mc.^(-1)); \]
\[ \text{end}; \]
\[ hcc = hcc*enthalpy_reduction; \]
\[ \text{fprintf('$$ c rho ( %1.2f , %1.2f ) [kg/m^3] n',min(min(Rho+c(:,:,1))),max(max(Rho+c(:,:,1))));} \]
\[ \text{fprintf('$$ c rc ( %1.2s , %1.2s ) [1] n',min(min(rc)),max(max(rc))));} \]
\[ \text{fprintf('$$ c Xf   ( %1.2s , %1.2s ) [1] n',min(min(Xsc(:,:,1))),max(max(Xsc(:,:,1))));} \]
\[ \text{fprintf('$$ c Xo2  ( %1.2s , %1.2s ) [1] n',min(min(Xsc(:,:,2))),max(max(Xsc(:,:,2))));} \]
\[ \text{fprintf('$$ c Xco2 ( %1.2s , %1.2s ) [1] n',min(min(Xsc(:,:,3))),max(max(Xsc(:,:,3))));} \]
\[ \text{fprintf('$$ c Xh2o ( %1.2s , %1.2s ) [1] n',min(min(Xsc(:,:,4))),max(max(Xsc(:,:,4))));} \]
\[ \text{fprintf('$$ c Xair ( %1.2s , %1.2s ) [1] n',min(min(Xsc(:,:,5))),max(max(Xsc(:,:,5))));} \]
\[ \text{fprintf('$$ c M  ( %1.2s , %1.2s ) [g/mol] n',min(min(Mc*1000)),max(max(Mc*1000))));} \]
\[ \text{fprintf('$$ c cp ( %1.2s , %1.2s ) [J/kg/K] n',min(min(cpc)),max(max(cpc))));} \]
\[ \text{fprintf('$$ c hc ( %1.2s , %1.2s ) [J/kg] n',min(min(hcc)),max(max(hcc))));} \]
\[ \text{fprintf('$$ c T  ( %1.1f , %1.1f ) [K] n',min(min((Tr+c(:,:,4))./(Rho+0.5*c(:,:,1))./cpc)),max(max((Tr+c(:,:,4))./(Rho+0.5*c(:,:,1))./cpc))));} \]
\[ \text{fprintf('------------- n');} \]
\[ Rhoc = F_density(Tr+0.5*c(:,:,4),Mc,cpc,Rhob,P); \]

\% 4.4 "d" calculation
\[ d(:,:,2) = dt*F_Ur(Rhoc,Ur+c(:,:,2),Vr+c(:,:,3),P); \]
\[ d(:,:,3) = dt*F_Vr(Rhoc,Ur+c(:,:,2),Vr+c(:,:,3),P); \]
\[ d(:,:,4) = \]
\[ dt*F_Tr(Rhoc+Ur+c(:,:,1),Vr+c(:,:,2),Tr+c(:,:,3),Pt,hcc,rc,cpc); \]
\[ \text{fprintf('dTcomb dTUx dTVy dhcUx dhcVy dTPt Tdiffx Tdiffy Hcdiffx Hcdiffy dTRadx dTRady dTRad3 dTCa\n');} \]
\[ \text{fprintf('%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f %1.2f
%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f
%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f
%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f
\% 4.4 "d" calculation
\[ d(:,:,2) = dt*F_Ur(Rhoc,Ur+c(:,:,2),Vr+c(:,:,3),P); \]
\[ d(:,:,3) = dt*F_Vr(Rhoc,Ur+c(:,:,2),Vr+c(:,:,3),P); \]
\[ d(:,:,4) = \]
\[ dt*F_Tr(Rhoc+Ur+c(:,:,1),Vr+c(:,:,2),Tr+c(:,:,3),Pt,hcc,rc,cpc); \]
\[ \text{fprintf('dTcomb dTUx dTVy dhcUx dhcVy dTPt Tdiffx Tdiffy Hcdiffx Hcdiffy dTRadx dTRady dTRad3 dTCa\n');} \]
\[ \text{fprintf('%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f %1.2f
%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f
%1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f %1.2f
%1.2f
\]
\( y./Rho./cp)\),\( \min(min(dhcUx./Rho./cp)),\( \min(min(dhcVy./Rho./cp)),\( \min(min(dTpt./Rho./cp)),\( \min(min(Tdiffx./Rho./cp)),\( \min(min(Tdiffy./Rho./cp)),\( \min(min(Hcdiffx./Rho./cp)),\( \min(min(Hcdiffy./Rho./cp)),\( \min(min(dTRadx./Rho./cp)),\( \min(min(dTRady./Rho./cp)),\( \min(min(dTRad3./Rho./cp)),\( \min(min(dTCa./Rho./cp)))\);
\[\]
\( \text{rd} = F_{Yf}(Rho+c(:,:,1),Tr+c(:,:,4),cp,Xsc(:,:,1),Xsc(:,:,2));\)
\[\]
\( Xsd = Xs;\)
\[\]
\( \text{for } jj=1:1:ny\)
\( \text{for } ii=1:1:nx\)
\( \text{for } s=1:1:2 \text{ % only for fuel and oxigen (partial for RK steps)}\)
\( Xval = Xs(ii,jj,s) -\)
\( (dt*coef(s)/Ms(1))*rd(ii,jj)*M(ii,jj);\)
\( \text{if } (s=1)\&(Xval<Xe); Xval=Xe;\)
\( \% \text{ combustion rate correction in case of extinction}\)
\( \text{rd(ii,jj)=(Xsd(ii,jj,s)-}\)
\( Xval)*Ms(1)/(dt*coef(s)*M(ii,jj));\)
\( \text{end};\)
\( \% \text{ record Zero Fuel condition}\)
\( Xsd(ii,jj,s) = Xval;\)
\( \text{end};\)
\( \text{end};\)
\( d(:,:,5) = dt*rd;\)
\( \text{fprintf('' $$$ d rd ( %1.2s , %1.2s ) [1]
\n',min(min(rd)),max(max(rd))));\)
\( \text{fprintf('' --------------------- \n');\)
\( \text{clear Xsc hcc rc cp};\)
\[\]
\% 4.5 Runge Kutta 4th order approximated solution
\( \text{Rhon} = (Rho + 2*Rhoa + 2*Rhob + Rhoc)/6;\)
\( \text{Ur} = (a(:,:,2)+2*b(:,:,2)+2*c(:,:,2)+d(:,:,2))/6;\)
\( \text{Vr} = (a(:,:,3)+2*b(:,:,3)+2*c(:,:,3)+d(:,:,3))/6;\)
\( \text{Tr} = (a(:,:,4)+2*b(:,:,4)+2*c(:,:,4)+d(:,:,4))/6;\)
\( \text{DYfn} = (a(:,:,5)+2*b(:,:,5)+2*c(:,:,5)+d(:,:,5))/6;\)
\( r = DYfn/dt;\)
\( \text{fprintf(''**** r ( %1.2s , %1.2s ) [1/s\n\n',min(min(r)),max(max(r))));\)
\( \text{fprintf('' --------------------- \n');\)
\( \text{clear ifl jfl};\)
\( Yfn = Yf + DYfn;\)
\( \text{fprintf(''**** Yf ( %1.2s , %1.2s ) [1]
\n',min(min(Yf)),max(max(Yf))));\)
\( \text{Pn} = F_{P}(Trn,M,cp);\)
\[\]
\% 4.6 Final update other properties
\( \text{for } jj=1:1:ny\)
\( \text{for } ii=1:1:nx\)
\( \text{for } s=1:1:4\)
\( Xval = Xs(ii,jj,s) -\)
\( DYfn(ii,jj)*coef(s)/Ms(1)*M(ii,jj);\)
\( \text{if } ((s=1)||(s=2))\&(Xval<Xe); Xval=Xe;\)
\( \% \text{ combustion rate correction in case of extinction}\)
\( \text{r(ii,jj)=(Xs(ii,jj,s)-}\)
\( Xval)*Ms(1)/(dt*coef(s)*M(ii,jj));\)
\( \text{end};\)
\( Xs(ii,jj,s) = Xval;\)
\( \text{end};\)
\( \text{end};\)
[i0f, j0f] = find(Xs(:, :, 1) == Xe);
Y0D = [i0f j0f];
clear i0f j0f
fprintf('**** Molar fractions estimated: Xi ( %1.2s %1.2s , %1.2s %1.2s )
\n', min(min(Xs(:, :, 1:2))), max(max(Xs(:, :, 1:2))));
M = zeros(nx, ny);
for s = 1:1:5
    M = M + Xs(:,:,s)*Ms(s);
end;
fprintf('**** mixture molar mass estimated: M ( %1.2s , %1.2s )
[g/mol]\n', min(min(M))*1000, max(max(M))*1000);
    cp = zeros(nx, ny);
    for s = 1:1:5
        cp = cp + (cps(s)*Ms(s))*Xs(:,:,s)/M;
    end;
    fprintf('**** Mixture specific heat capacity: cp ( %1.2s , %1.2s ) [J/kg/K]
\n', min(min(cp)), max(max(cp)));
hc = zeros(nx, ny);
for s = 1:1:4
    hc = hc - coef(s)*(Hrefs(s)+(Ms(s)*cps(s))**(Trn./Rhon./cp-Tref)).*(M.^(-1));
end;
hc = hc*enthalpy_reduction;
fprintf('**** Specific combustion enthalpy: hc ( %1.2s , %1.2s )
[J/kg]\n', min(min(hc)), max(max(hc)));
    T = Trn./Rhon./cp;
    fprintf('**** Temperature: T ( %1.2s , %1.2s )
[K]\n', min(min(T)), max(max(T)));

% Update
Rho = Rhon; clear Rhon
Ur = Urn; clear Urn
Vr = Vrn; clear Vrn
Tr = Trn; clear Trn
Yf = Yfn; clear Yfn
Pt = (Pn-P)/dt;
P = Pn; clear Pn
fprintf('**** Density : Rho ( %1.2f , %1.2f )
[kg/m^3]\n', min(min(Rho)), max(max(Rho)));
fprintf('**** Pressure: P ( %1.2s , %1.2s )
[Pa]\n', min(min(P)), max(max(P)))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% 5. VISUALISATION
% Density plot
figure(1)
clf(1)
[hR hR] = contourf(avg(x), avg(y), (Rho)', 'levels');
set(hR, 'LineStyle', 'none');
colormap(pink)
caxis([0.3*rho0 rho0]);
colorbar
title(sprintf('Density field [kg/m^3] (colormap)\n
 Re = %0.1g  t = %0.2g s', Re, k*dt))
axis equal, axis([0 1x 0 1y])
grid on, hold on
xlabel('x [m]')
ylabel('y [m]')
drawnow
frame = getframe(1);
im = frame2im(frame);
[imind, cm] = rgb2ind(im,256);
if k == 1;
    imwrite(imind, cm, filename1, 'gif', 'Loopcount', inf);
else
    imwrite(imind, cm, filename1, 'gif', 'WriteMode', 'append');
end;
writeVideo(vid1, frame);

% Flame Domain Plot
figure(2)
clf(2)
grid on
axis equal, axis([0 lx 0 ly])
title(sprintf('Flame Domain (red)\nZero Fuel Domain (green)\nRe = %0.1g \ t = %0.2g s', Re, k*dt))
if exist('FLD')==1
    if (length(FLD)>1)
        for indxx=1:1:length(FLD)
            hold on
            FL Dx = dx*(FLD(indxx,1)-1.5);
            FL Dy = dy*(FLD(indxx,2)-1.5);
            rectangle('Position',[FL Dx, FL Dy, dx, dy], 'FaceColor', 'r', 'EdgeColor', 'r', 'LineWidth', 1)
        end
    end
end;

% Zero Fuel Domain
if exist('YOD')==1
    for indxx=1:1:length(YOD)
        hold on
        YODx = dx*(YOD(indxx,1)-1.5);
        YODy = dy*(YOD(indxx,2)-1.5);
        rectangle('Position',[YODx, YODy, dx, dy], 'FaceColor', 'g', 'EdgeColor', 'g', 'LineWidth', 1)
    end
end;
grid on
xlabel('x [m]')
ylabel('y [m]')
drawnow
frame = getframe(2);
im = frame2im(frame);
[imind, cm] = rgb2ind(im,256);
if k == 1;
    imwrite(imind, cm, filename2, 'gif', 'Loopcount', inf);
else
    imwrite(imind, cm, filename2, 'gif', 'WriteMode', 'append');
end;
writeVideo(vid2, frame);

% Temperature Field Plot
figure(3)
clf(3)
[hT hT] = contourf(x(1:end-1), y(1:end-1), T', levels);
set(hT, 'LineStyle', 'none');
colormap(jet)  
https://uk.mathworks.com/help/matlab/ref/colormap.html  
caxis([Tmin,Tmax+50])
colorbar ('AxisLocation','in')  
axis equal, axis([0 lx 0 ly])  
title(sprintf('Temperature field [K] (colormap)\nAmbient temperature (green contour)\nPyrolysis temperature (red contour)\nRe = %0.1g  t = %0.2g s',Re,k*dt))  
hold on  
contour(x(1:end-1),y(1:end-1),round(T)',[To To],'g-',
'LineWidth',1.5)  
hold on  
contour(x(1:end-1),y(1:end-1),round(T)',[Tp Tp],'r-',
'LineWidth',1.5)  
xlabel('x [m]')
ylabel('y [m]')  
drawnow  
frame = getframe(3);  
im = frame2im(frame);  
[imind,cm] = rgb2ind(im,256);  
if  
if k == 1;  
    imwrite(imind,cm,filename3,'gif', 'Loopcount',inf);  
else  
    imwrite(imind,cm,filename3,'gif','WriteMode','append');  
end  
end  
writeVideo(vid3,frame);  

% Temperature profile centre domain in x direction  
figure(4)  
xspace = avg((linspace(0,lx,nx+1))');  
clf(4)  
g_y=[250:50:Tmax+100]; % user defined grid Y [start:spaces:end]  
g_x=[dx:dx:lx-dx]; % user defined grid X [start:spaces:end]  
for i=1:length(g_x)  
    plot([g_x(i) g_x(i)],[g_y(1) g_y(end)],'k:') %y grid lines  
    hold on  
end  
for i=1:length(g_y)  
    plot([g_x(1) g_x(end)],[g_y(i) g_y(i)],'k:') %x grid lines  
    hold on  
end  
hold on  
title(sprintf('Temperature profile (half domain section)\nRe = %0.1g  t = %0.2g s',Re,k*dt))  
xlabel('x [m]');  
ylabel('T [K]');  
Tplot=Tr./Rho./cp;  
plot(xspace,Tp*ones(length(xspace)),'-b',xspace,To*ones(length(xspace)),'-.k','LineWidth',1.5);  
hold on  
plot(xspace,(Tplot(:,round(ny/2))),'-r','LineWidth',2);  
T_store(:,k+1)=Tplot(:,round(ny/2));  
%axis([0 lx Tmin-50 Tmax+100]);  
drawnow  
frame = getframe(4);  
im = frame2im(frame);  
[imind,cm] = rgb2ind(im,256);  
imwrite(imind,cm,filename4,'gif','WriteMode','append');  
writeVideo(vid4,frame);
% combustion energy, temperature, mass fractions in flame ignition
% point (centre of the flame)
if k>1
figure(5)
subplot(4,1,1)
hold on
plot(dt*[k-1 k],[Old(1) dTcomb(nxig,nyig)],'r','LineWidth',2);
xlabel('\$Time\quad [\ s ]\$','interpreter','latex');
ylabel('\$\dot{Q}\quad [\ W/m^3\ ]\$','interpreter','latex');
subplot(4,1,2)
hold on
plot(dt*[k-1 k],[Old(2) Tplot(nxig,nyig)],'m','LineWidth',2);
xlabel('\$Time\quad [\ s ]\$','interpreter','latex');
ylabel('\$Temperature\quad [\ K ]\$','interpreter','latex');
subplot(4,1,3)
hold on
plot(dt*[k-1 k],[Old(3) Xs(nxig,nyig,1)],'b','LineWidth',2);
xlabel('\$Time\quad [\ s ]\$','interpreter','latex');
ylabel('\$X\quad fuel\quad [\ 1 ]\$','interpreter','latex');
subplot(4,1,4)
hold on
plot(dt*[k-1 k],[Old(4) Xs(nxig,nyig,2)],'k','LineWidth',2);
xlabel('\$Time\quad [\ s ]\$','interpreter','latex');
ylabel('\$X\quad oxidizer\quad [\ 1 ]\$','interpreter','latex');
end

Old=[dTcomb(nxig,nyig) Tplot(nxig,nyig) Xs(nxig,nyig,1)
Xs(nxig,nyig,2)];
clear Tplot
diary('LOG.txt')
clear FlD
fprintf('
');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
end;

nameT=('T_store.mat');
save(nameT,'T_store'); % Darude Sandstorm
close(vid1);
close(vid2);
close(vid3);
close(vid4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% 6. SUB-ROUTINES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function value = F_Rho(Ur,Vr) %
global Uinf Vinf nx ny dx dy rho0
% dUr/dx (Centre difference) mass transport in x-dir
value(1,:) = -(Ur(2,:)+3*Ur(1,:)-4*Uinf*rho0*ones(1,ny))/(3*dx);
% if Dirichlet East
value(nx,:) = -(4*Uinf*rho0*ones(1,ny)-Ur(nx-1,:)-3*Ur(nx,:))/(3*dx);
% if Neumann East
% value(nx,:)=-((Ur(nx,:)-Ur(nx-1,:))/(2*dx);
for i=2:1:nx-1
    value(i,:)=-((Ur(i+1,:)-Ur(i-1,:))/(2*dx);
end;
% dVr/dy (Centre difference) mass transport in y-dir
value(:,1) = value(:,1) - (Vr(:,2)+3*Vr(:,1) - 4*Vinf*rho0*ones(nx,1))/(3*dy);
% if Dirichlet North
value(:,ny) = value(:,ny) - (4*Vinf*ones(nx,1)-Vr(:,ny-1) - 3*Vr(:,ny))/(3*dy);
% if Neumann North
for j=2:1:ny-1
    value(:,j) = value(:,j) - (Vr(:,j+1)-Vr(:,j-1))/(2*dy);
end;

function value = F_Ur(Rho,Ur,Vr,P)
% global Uinf Vinf nx ny dx dy mu rho0 Po
% d(Ur^2/Rho)/dx (Centre difference) U-momentum transport in x-dir
value(1,:) = -(Ur(2,:).^2)./Rho(2,:)+3*(Ur(:,1).*Vr(:,1))./Rho(:,1) - (4*rho0*Uinf*Vinf^2)*ones(1,ny))/(3*dx);
% if Dirichlet East
value(nx,:) = -(4*rho0*Uinf^2)*ones(1,ny)-(Ur(nx-1,:).^2)./Rho(nx-1,:)-3*(Ur(nx,:).*Vr(nx-1,:))./Rho(nx-1,:))/(3*dx);
% if Neumann East
%     value(nx,:) = -(Ur(nx,:).*Vr(nx-1,:))./rho0-Ur(nx,:)./(2*dx);
U2Rho = (Ur.^2)./Rho;
for i=2:1:nx-1
    value(i,:) = -(U2Rho(i+1,:)-U2Rho(i-1,:))/(2*dx);
end;

clear U2Rho

% d(UrVr/Rho)/dy (Centre difference) U-momentum transport in y-dir
value(:,1) = value(:,1) - (Ur(:,2).*Vr(:,2)./Rho(:,2)+3*Ur(:,1).*Vr(:,1)./Rho(:,1) - (4*rho0*Uinf*Vinf)*ones(nx,1))/(3*dy);
% if Dirichlet North
value(:,ny) = value(:,ny) - (4*rho0*Uinf*Vinf)*ones(nx,1)-Ur(:,ny-1).*Vr(:,ny-1)./Rho(:,ny-1))/(3*dy);
% if Neumann North
%     value(:,ny) = value(:,ny) - (Ur(:,ny).*Vr(:,ny))./rho0-Ur(:,ny-1).*Vr(:,ny-1)./Rho(:,ny-1))/(2*dy);
UVRho = Ur.*Vr./Rho;
for j=2:1:ny-1
    value(:,j) = value(:,j) - (UVRho(:,j+1)-UVRho(:,j-1))/(2*dy);
end;

clear UVRho

% Diffusion in x-dir
value(1,:) = value(1,:) + mu*(4*Ur(2,:)/Rho(2,:)-12*Ur(1,:)/Rho(1,:)+8*Uinf*ones(1,ny))/(3*dx^2);
% Neumann East
%     value(nx,:) = value(nx,:) + 2*mu*(Ur(nx-1,:)/Rho(nx-1,:)-Ur(nx,:)/Rho(nx,:))/(dx^2);
% Dirichlet East
value(nx,:) = value(nx,:) + mu*(8*Uinf*ones(1,ny)-12*Ur(nx,:)/Rho(nx,:)+4*Ur(nx-1,:)/Rho(nx-1,:))/(3*dx^2);
URho = Ur./Rho;
for i=2:1:nx-1
    value(i,:) = value(i,:) + mu*(URho(i-1,:)-2*URho(i,:)+URho(i+1,:))/(dx^2);
end;

clear URho

% Diffusion in y-dir
value(:,1) = value(:,1) + mu*(4*Ur(:,2)/Rho(:,2)-12*Ur(:,1)/Rho(:,1)+8*Uinf*ones(nx,1))/(3*dy^2);
% Neumann North
value(:,ny) = value(:,ny) + 2*mu*(Ur(:,ny-1)./Rho(:,ny-1)-Ur(:,ny)./Rho(:,ny))/(dy^2);
% Dirichlet North
value(:,ny) = value(:,ny) + mu*(8*Uinf*ones(nx,1)-12*Ur(:,ny)./Rho(:,ny)+4*Ur(:,ny-1)./Rho(:,ny-1))/(3*dy^2);
URho = Ur./Rho;
for j=2:1:ny-1
 value(:,j) = value(:,j) + mu*(URho(:,j-1)-2*URho(:,j)+URho(:,j+1))/(dy^2);
end;
clear URho
% Pressure gradient dP/dx
value(nx,:)= value(nx,:)-(4*Po*ones(1,ny)-P(nx,:)-3*P(nx-1,:))/(3*dx);
% if Neumann West
value(1,:)= value(1,:)-(P(2,:)-P(1,:))/(2*dx);
% if Dirichlet West
value(1,:)= value(1,:)-(P(2,:)+3*P(1,:)-4*Po*ones(1,ny))/(3*dx);
% for i=2:1:nx
value(i,:)= value(i,:)-(P(i+1,:)-P(i-1,:))/(2*dx);
end;
% case : grad(p)|bc=0
value(nx,:)= value(nx,:)-((P(nx,:)-P(nx-1,:))/(3*dx));
% if Neumann West
value(1,:)= value(1,:)-(P(2,:)-P(1,:))/(2*dx);
% if Dirichlet West
value(1,:)= value(1,:)-(P(2,:)-P(1,:))/(2*dx);
for i=2:1:nx
 value(i,:)= value(i,:)-(P(i+1,:)-P(i-1,:))/(2*dx);
end;

function value = F_Vr(Rho,Ur,Vr,P)
% d(Vr^2/Rho)/dy (Centre difference) V-momentum transport in y-dir
value(:,1) = -((Vr(:,2).^2)./Rho(:,2)+3*(Vr(:,1).^2)./Rho(:,1)-(4*rho0*Vinf^2)*ones(nx,1))/(3*dy);
% if Dirichlet East
value(:,ny) = -((4*rho0*Vinf^2)*ones(nx,1)-(Vr(:,ny-1).^2)./Rho(:,ny-1)-3*(Vr(:,ny).^2)./Rho(:,ny))/(3*dy);
% if Neumann East
value(:,ny) = -(Vr(:,ny).^2)/rho0-(Vr(:,ny-1).^2)./(Rho(:,ny-1))/(3*dy);
V2Rho = (Vr.^2)./Rho;
for j=2:1:ny-1
 value(:,j) = -(V2Rho(:,j+1)-V2Rho(:,j-1))/(2*dy);
end;
clear V2Rho
% d(Ur*Vr/Rho)/dx (Centre difference) V-momentum transport in x-dir
value(1,:) = value(1,:) - (Ur(2,:).*Vr(2,:))./Rho(2,:)+3*Ur(1,:).*Vr(1,:)./(Rho(1,:)-(4*rho0*Uinf*Vinf)*ones(1,ny))/(3*dx);
% if Dirichlet North
value(nx,:) = value(nx,:) -((4*rho0*Uinf*Vinf)*ones(1,ny)-Ur(nx-1,:).*Vr(nx-1,:)-3*Ur(nx-1,:)*Vr(nx,:)./(Rho(nx-1,:)))/(3*dx);
% if Neumann North
value(nx,:) = value(nx,:) -(Ur(nx,:).*Vr(nx,:)/rho0-Ur(nx-1,:).*Vr(nx-1,:)./(Rho(nx-1,:)))/(2*dx);
UVRho = Ur.*Vr./Rho;
for i=2:1:nx
    value(i,:) = value(i,:)-(UVRho(i+1,:)-UVRho(i-1,:))/(2*dx);
end;
clear UVRho

% Diffusion in x-dir
value(1,:) = value(1,:) + mu*(4*Vr(1,:)/Rho(1,:)-12*Vr(1,:)/Rho(1,:)+8*Vinf*ones(1,ny))/(3*dx^2);
% Neumann East
value(nx,:) = value(nx,:) + 2*mu*(Vr(nx-1,:)/Rho(nx-1,:)-Vr(nx,:)/Rho(nx,:))/(dx^2);
% Dirichlet East
value(nx,:) = value(nx,:) + mu*(8*Vinf*ones(1,ny)-12*Vr(nx,:)/Rho(nx,:)+4*Vr(nx-1,:)/Rho(nx-1,:))/(3*dx^2);
VRho = Vr./Rho;
for i=2:1:nx-1
    value(i,:) = value(i,:) + mu*(VRho(i-1,:)-2*VRho(i,:)+VRho(i+1,:))/(dx^2);
end;
clear VRho

% Pressure gradient dP/dy
value(:,ny) = value(:,ny)-(4*Po*ones(nx,1)-P(:,ny-1)-3*P(:,ny))/(3*dy);
% Neumann South
value(:,1) = value(:,1)-(P(:,2)-P(:,1))/(2*dy);
% Dirichlet South
value(:,1) = value(:,1)-(P(:,2)+3*P(:,1)-4*Po*ones(nx,1))/(3*dy);
for j=2:1:ny-1
    value(:,j) = value(:,j)-(P(:,j+1)-P(:,j-1))/(2*dy);
end;
clear VRho

function value = F_Tr(Rho, Ur, Vr, Tr, Pt, hc, r, cp)
% Pt = dP/dt
global Uinf Vinf nx ny dx dy dz rho0 To cp0 hc0 kth sigm emit phi Ca l2D
global dTcomb dTUX dTVy dhcUX dhcVy dTFtTdify Hcdiffx Hcdiffy dTRadx dTRady dTRad3 dTCa
\[ Tcpr0 = To*cp0*rho0; \]
\[ hcr0 = hc0*rho0; \]

% Combustion enthalpy \(hc\) * \(r\) 

OK !
\[ dTcomb = -Rho.*hc.*r; \]
\[ value = dTcomb; \]

% T Transport

OK !

% \(dUT/dx\)
% West
\[ dTUx(1,:) = -( Ur(2,:).*Tr(2,:)/Rho(2,:)+3*Ur(1,:).*Tr(1,:)/Rho(1,:) - (4*Uinf*Tcpr0)*ones(1,ny))/(3*dx); \]

% East (Dirichlet)
\[ dTUx(nx,:) = -((4*Uinf*Tcpr0)*ones(1,ny)-Ur(nx-1,:).*Tr(nx-1,:)/Rho(nx-1,:)-3*Ur(nx,:).*Tr(nx,:)/Rho(nx,:))/(3*dx); \]

% Internal
\[ UrTrR = Ur.*Tr./Rho; \]
% West
\[ for i=2:1:nx-1 \]
\[ dTUx(i,:) = -\left(UrTrR(i+1,:)-UrTrR(i-1,:)/2\right); \]
\[ end; \]
\[ value = value + dTUx; \]
% clear UrTrR

% \(dVT/dy\)
% South
\[ dTVy(:,1) = -( Vr(:,2).*Tr(:,2)/Rho(:,2)+3*Vr(:,1).*Tr(:,1)/Rho(:,1) - (4*Vinf*Tcpr0)*ones(nx,1))/(3*dy); \]

% North (Dirichlet)
\[ dTVy(:,1) = -((4*Vinf*Tcpr0)*ones(nx,1)-Vr(:,ny-1,:).*Tr(:,ny-1,:)/Rho(:,ny-1,:)-3*Vr(:,ny,:).*Tr(:,ny,:)/Rho(:,ny,:))/(3*dy); \]

% Internal
\[ VrTrR = Vr.*Tr./Rho; \]
% South
\[ for j=2:1:ny-1 \]
\[ dTVy(:,j) = -\left(VrTrR(:,j+1)-VrTrR(:,j-1)/2\right); \]
\[ end; \]
\[ value = value + dTVy; \]
% clear VrTrR

% hc Transport

OK !

% \(dUhc/dx\)
% West
\[ dUhcx(1,:) = -( Ur(2,:).*hc(2,:)+3*Ur(1,:).*hc(1,:)- (4*Uinf*hcr0(1,:)).*ones(1,ny))/(3*dx); \]

% East (Dirichlet)
\[ dUhcx(nx,:) = -((4*Uinf*hcr0(1,:)).*ones(1,ny)-Ur(nx-1,:).*hc(nx-1,:)-3*Ur(nx,:).*hc(nx,:))/(3*dx); \]

% Internal
\[ Urhc = Ur.*hc; \]
% West
\[ for i=2:1:nx-1 \]
\[ dUhcx(i,:) = -(Urhc(i+1,:)-Urhc(i-1,:)/2; \]
\[ end; \]
\[ value = value + dUhcx; \]
% clear Urhc

% \(dVhc/dy\)
% South
\[ dVhc(1,:) = -( Vr(:,2).*hc(:,2)+3*Vr(:,1).*hc(:,1)- (4*Vinf*hcr0).*ones(nx,1))/(3*dy); \]

% North (Neumann)
% value(:,ny) = value(:,ny) - (Vr(:,ny).*hc(:,ny)-Vr(:,ny-1))/(2*dy);
% North (Dirichlet)
dhcVy(:,ny) = -(4*Vinf*hcr0)*ones(nx,1)-Vr(:,ny-1).*hc(:,ny-1)-
3*Vr(:,ny).*hc(:,ny))/(3*dy);
% Internal
Vrhc = Vr.*hc;
for j=2:1:ny-1
dhcVy(:,j) = -(Vrhc(:,j+1)-Vrhc(:,j-1))/(2*dx);
end;
value = value + dhcVy;
clear Vrhc

% Pressure (dP/dt)                                                                                                       ! CHECK
OK!
   dTPr = 0*Pt;
   value = value + dTPr;

% T Diffusion                                                                                                             ! CHECK
OK!
% d2T/dx2
% West (Dirichlet)
al = 2*2./(cp(1,:)+cp(2,:))1./cp(1,:));
a2 = 2*2./(cp(1,:)+cp(2,:))-3./cp(1,:)+16./(cp(1,:)+cp0));
a3 = 2*(-4./cp(1,:)+16./(cp(1,:)+cp0));
Tdiffx(1,:) = kth*(al.*Tr(2,:)/Rho(2,:)-a2.*Tr(1,:)/Rho(1,:)+a3*cp0*To)/(3*dx^2);
% East (Dirichlet)
al = 2*2./(cp(nx,:)+cp(nx-1,:))1./cp(nx,:));
a2 = 2*2./(cp(nx,:)+cp(nx-1,:))-3./cp(nx,:)+16./(cp(nx,:)+cp0));
a3 = 2*(-4./cp(nx,:)+16./(cp(nx,:)+cp0));
Tdiffx(nx,:) = kth*(al.*Tr(nx-1,:)/Rho(nx-1,:)-a2.*Tr(nx,:)/Rho(nx,:)+a3*cp0*To)/(3*dx^2);
% Internal
TrR = Tr./Rho;
cpm = 1./avg(cp);
for i=2:1:nx-1
   Tdiffx(i,:) = kth*(TrR(i+1,:).*cpm(i,:)-TrR(i,:).*cpm(i-1,:)+cpm(i,:)*TrR(i-1,:).*cpm(i-1,:))//(dx^2);
end;
value = value + Tdiffx;
clear cpm

% d2T/dy2
% South (Dirichlet)
al = 2*2./(cp(:,1)+cp(:,2))1./cp(:,1));
a2 = 2*2./(cp(:,1)+cp(:,2))-3./cp(:,1)+16./(cp(:,1)+cp0));
a3 = 2*(-4./cp(:,1)+16./(cp(:,1)+cp0));
Tdiffy(:,1) = kth*(al.*Tr(:,2)/Rho(:,2)-a2.*Tr(:,1)/Rho(:,1)+a3*cp0*To)/(3*dx^2);
% North (Dirichlet)
al = 2*2./(cp(:,ny)+cp(:,ny-1))1./cp(:,ny));
a2 = 2*2./(cp(:,ny)+cp(:,ny-1))-3./cp(:,ny)+16./(cp(:,ny)+cp0));
a3 = 2*(-4./cp(:,ny)+16./(cp(:,ny)+cp0));
Tdiffy(:,ny) = kth*(al.*Tr(:,ny-1)/Rho(:,ny-1)-a2.*Tr(:,ny)/Rho(:,ny)+a3*cp0*To)/(3*dx^2);
% Internal
TrR = Tr./Rho;
cpm = 1./avg((cp)');
for j=2:1:ny-1
   Tdiffy(:,j) = kth*(TrR(:,j+1).*cpm(:,j)-TrR(:,j).*cpm(:,j-1)+cpm(:,j)*TrR(:,j-1).*cpm(:,j-1))/(dy^2);
end;
value = value + Tdiffy;
clear TrR cpm

% hc Diffusion
% CHECK OK!
% d2hc/dx2
% West (Dirichlet)
a1 = 2*(2./(cp(1,:)+cp(2,:))+1./cp(1,:));
a2 = 2*(2./(cp(1,:)+cp(2,:)) - 3./cp(1,:) + 16./(cp(1,:) + cp0));
a3 = 2*(-4./(cp(1,:)) + 16./(cp(1,:) + cp0));
Hcdiffx(1,:) = kth*(a1.*hc(2,:) -
  a2.*hc(1,:) + a3.*hc0(1,:))/(3*dx^2);
% East (Dirichlet)
a1 = 2*(2./(cp(nx,:)+cp(nx-1,:))+1./cp(nx,:));
a2 = 2*(2./(cp(nx,:)+cp(nx-1,:)) - 3./cp(nx,:) + 16./(cp(nx,:) + cp0));
a3 = 2*(-4./(cp(nx,:)) + 16./(cp(nx,:) + cp0));
Hcdiffx(nx,:) = kth*(a1.*hc(nx-1,:) -
  a2.*hc(nx,:) + a3.*hc0(nx,:))/(3*dx^2);
% Internal
cpm = 1./avg(cp);
for i=2:1:nx-1
  Hcdiffx(i,:) = kth*(hc(i+1,:).*cpm(i,:)-hc(i,:).*cpm(i-1,:)+
    hc(i-1,:).*cpm(i,:))/(dx^2);
end;
value = value + Hcdiffx;
clear cpm
% d2hc/dx2
% South (Dirichlet)
a1 = 2*(2./(cp(:,1)+cp(:,2))+1./cp(:,1));
a2 = 2*(2./(cp(:,1)+cp(:,2)) - 3./cp(:,1) + 16./(cp(:,1) + cp0));
a3 = 2*(-4./(cp(:,1)) + 16./(cp(:,1) + cp0));
Hcdiffy(:,1) = kth*(a1.*hc(:,2) -
  a2.*hc(:,1) + a3.*hc0(:,1))/(3*dy^2);
% North (Dirichlet)
a1 = 2*(2./(cp(:,ny)+cp(:,ny-1))+1./cp(:,ny));
a2 = 2*(2./(cp(:,ny)+cp(:,ny-1)) - 3./cp(:,ny) + 16./(cp(:,ny) + cp0));
a3 = 2*(-4./(cp(:,ny)) + 16./(cp(:,ny) + cp0));
Hcdiffy(:,ny) = kth*(a1.*hc(:,ny-1) -
  a2.*hc(:,ny) + a3.*hc0(:,ny))/(3*dy^2);
% Internal
cpm = 1./avg((cp)');
for j=2:1:ny-1
  Hcdiffy(:,j) = kth*(hc(:,j+1).*cpm(:,j)-hc(:,j).*cpm(:,j-1)+
    hc(:,j-1).*cpm(:,j))/(dy^2);
end;
value = value + Hcdiffy;
clear cpm

% Radiation
% CHECK OK!
T = Tr./Rho./cp;
% dRad/dx
% West (Dirichlet)
dTRadx(1,:) =
  (12*D*sign*emit*phi/(3*dx^2))*(8*(To^4)+16*(To^3)*T(1,:)-
  4*To*(T(1,:).^3)+15*(T(1,:).^4)+6*(T(1,:).^3).*T(2,:)+2*T(1,:).*
    (T(2,:).^3)+T(2,:).^4);
% East (Dirichlet)
dTRadx(nx,:) =
  (12*D*sign*emit*phi/(3*dx^2))*(8*(To^4)+16*(To^3)*T(nx,:)-
48*To^3*(T(nx,:),^3)+15*(T(nx,:),^4)+6*(T(nx-,1,:),^3)*T(nx-,1,:)+2*T(nx,:),^3)*T(nx-,1,:)+T(nx-,1,:).^4);

% Internal
Tavg = avg(T);
for i=2:1:nx-1
    dTRadx(i,:) = (l2D*sigm*emit*phi*4/dx^2)*((T(i+1,:)-T(i,:)).*Tavg(i,:).^3+(T(i-1,:)-T(i,:)).*Tavg(i-1,:).^3);
end;
value = value + dTRadx;
clear Tavg
% dRad/dy
% West (Dirichlet)
dTRady(:,1) = (12D*sigm*emit*phi/(3*dy^2))*((8*(To^4)+16*(To^3)*T(:,1)-48*To*(T(:,1).^3)+15*(T(:,1).^4)+6*(T(:,1).^3).*T(:,2)+2*T(:,1).*(T(:,2).^3)+T(:,2).^4);
% East (Dirichlet)
dTRady(:,ny) = (12D*sigm*emit*phi/(3*dy^2))*((8*(To^4)+16*(To^3)*T(:,ny)-48*To*(T(:,ny).^3)+15*(T(:,ny).^4)+6*(T(:,ny).^3).*T(:,ny-1)+2*T(:,ny).*T(:,ny-1).^3)+(T(:,ny-1).^3)+T(:,ny-1).^4);
% Internal
Tavg = avg(T');
for j=2:1:ny-1
    dTRady(:,j) = (12D*sigm*emit*phi*4/dy^2)*((T(:,j+1)-T(:,j)).*Tavg(:,j).^3+(T(:,j-1)-T(:,j)).*Tavg(:,j-1).^3);
end;
value = value + dTRady;
clear Tavg

% 3D pseudo-Radiation (z-dir.)                                 ! CHECK
OK !
dTRad3 = -sigm*emit*phi*(T.^4 - To^4)*(1/dz);
value = value + dTRad3;
% 3D pseudo-Convection (z-dir.)                                 ! CHECK
OK !
dTCa = -Ca*(T - To);
value = value + dTCa;
clear T

function value = F_Yf(Rho,Tr,cp,Xf,Xo2)
% global nx ny Ta Ahc
Temp = Tr./Rho./cp;
value = zeros(nx,ny);
for j=1:1:ny
    for i=1:1:nx
        kdFl = Spr(Temp(i,j),Xf(i,j),Xo2(i,j)); % switch Flame ignition
        value(i,j) = -kdFl*Ahc*Temp(i,j)*Xf(i,j)*Xo2(i,j)*exp(-Ta/Temp(i,j));
        % end;
    end;
end;

function value = F_P(Tr,M,cp)
% global R
value = R*Tr./M./cp;

function value = F_density(Tr,M,cp,Rho,P)
global R value = M.*P.*Rho.*cp./Tr/R;

% Average matrix
function B = avg(A,k)
if nargin<2, k = 1; end
if size(A,1)==1, A = A'; end
if k<2
    B = (A(2:end,:)+A(1:end-1,:))/2;
else
    B = avg(A,k-1);
end
if size(A,2)==1, B = B'; end

% Ignition Condition
function spread = Spr(T,X,Xo)
    global Tp Xe Xo2e
    if (T>Tp) & (X>Xe) & (Xo>Xo2e)
        spread = 1;
    else
        spread = 0;
    end;
APPENDIX D: FD FORMULATION EXAMPLE

Follows an example of Finite Difference formulation of the fire-spread system of partial differential equations.

**Note 1:** When the variable has no \(^{(n)}\) exponent means that is the value at the old time-step, while the one with \(^{(n+1)}\) apex has to be calculated.

**Note 2:** In order to pass form a grid to the staggered ones, the average value between the neighbour nodes.

**EQ.1:** Mass balance on scalar grid.
\[
\frac{\rho_{i,j}^{(n+1)} - \rho_{i,j}}{\Delta t} + \frac{\left( \rho_{i,j} + \rho_{i+1,j} \right) u_{i,j} - \left( \rho_{i-1,j} + \rho_{i,j} \right) u_{i-1,j}}{2 \cdot \Delta x} + \frac{\left( \rho_{i,j} + \rho_{i,j+1} \right) v_{i,j} - \left( \rho_{i,j} + \rho_{i,j-1} \right) v_{i,j-1}}{2 \cdot \Delta y} = 0
\]

**EQ.2:** Momentum conservation in x direction on ‘u’ grid.
\[
\frac{\left( \rho_{i+1,j}^{(n+1)} + \rho_{i,j}^{(n+1)} \right) u_{i,j}^{(n+1)} - \left( \rho_{i-1,j} + \rho_{i,j} \right) u_{i,j}}{2 \cdot \Delta t} + \frac{\left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i,j+1}^{(n+1)} \right) \left( u_{i,j} + u_{i,j+1} \right) \left( v_{i,j} + v_{i,j+1} \right)}{4 \cdot \Delta x} + \frac{\left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i,j+1}^{(n+1)} \right) \left( u_{i,j} + u_{i,j+1} \right) \left( v_{i,j} + v_{i,j+1} \right)}{16 \cdot \Delta y} - \frac{p_{i+1,j} - p_{i,j}}{\Delta x} + \mu \left( \frac{u_{i,j}^{(n+1)} - 2 u_{i,j}^{(n+1)} + u_{i,j}^{(n+1)}}{\Delta x^2} + \frac{v_{i,j}^{(n+1)} - 2 v_{i,j}^{(n+1)} + v_{i,j}^{(n+1)}}{\Delta y^2} \right) = 0
\]

**EQ.3:** Momentum conservation in y direction on ‘v’ grid.
\[
\frac{\left( \rho_{i,j+1}^{(n+1)} + \rho_{i,j}^{(n+1)} \right) v_{i,j}^{(n+1)} - \left( \rho_{i,j-1} + \rho_{i,j} \right) v_{i,j}}{2 \cdot \Delta t} + \frac{\left( \rho_{i,j} + \rho_{i,j+1} + \rho_{i+1,j} + \rho_{i,j+1} \right) \left( v_{i,j} + v_{i,j+1} \right) \left( u_{i,j} + u_{i,j+1} \right)}{4 \cdot \Delta y} + \frac{\left( \rho_{i,j} + \rho_{i,j-1} + \rho_{i,j+1} + \rho_{i,j+1} \right) \left( v_{i,j} + v_{i,j+1} \right) \left( u_{i,j} + u_{i,j+1} \right)}{16 \cdot \Delta x} - \frac{p_{i,j+1} - p_{i,j}}{\Delta y} + \mu \left( \frac{v_{i,j}^{(n+1)} - 2 v_{i,j}^{(n+1)} + v_{i,j}^{(n+1)}}{\Delta x^2} + \frac{u_{i,j}^{(n+1)} - 2 u_{i,j}^{(n+1)} + u_{i,j}^{(n+1)}}{\Delta y^2} \right) = 0
\]
EQ.4: Pressure closure on scalar grid.

\[ p_{i,j}^{(n+1)} = -R \frac{\rho_{i,j}^{(n+1)} T_{i,j}^{(n+1)}}{M_{i,j}^{(n+1)}} \]

EQ.5: Energy equation on scalar grid

\[ \frac{\rho_{i,j}^{(n+1)} c_{p,i,j} T_{i,j}^{(n+1)}}{\Delta t} - \rho_{i,j}^{(n+1)} c_{p,i,j} T_{i,j}^{(n+1)} \] \[ \frac{1}{8 \cdot \Delta x} \left[ \left( \rho_{i,j} + \rho_{i+1,j} \right) c_{p,i,j} + c_{p,i+1,j} \right] \left( T_{i,j} + T_{i+1,j} \right) - \left( \rho_{i,j} + \rho_{i-1,j} \right) c_{p,i,j} \left( T_{i,j} + T_{i-1,j} \right) + ... \]

\[ \frac{1}{8 \cdot \Delta y} \left[ \left( \rho_{i,j} + \rho_{i,j+1} \right) c_{p,i,j} + c_{p,i,j+1} \right] \left( T_{i,j} + T_{i,j+1} \right) - \left( \rho_{i,j} + \rho_{i,j-1} \right) c_{p,i,j} \left( T_{i,j} + T_{i,j-1} \right) + ... \]

\[ \frac{1}{4 \cdot \Delta x} \left[ \left( \rho_{i,j} + \rho_{i+1,j} \right) c_{p,i,j} + c_{p,i+1,j} \right] u_{i,j} - \left( \rho_{i,j} + \rho_{i-1,j} \right) c_{p,i,j} u_{i,j} + ... \]

\[ \frac{1}{4 \cdot \Delta y} \left[ \left( \rho_{i,j} + \rho_{i,j+1} \right) c_{p,i,j} + c_{p,i,j+1} \right] v_{i,j} - \left( \rho_{i,j} + \rho_{i,j-1} \right) c_{p,i,j} v_{i,j} + ... \]

\[ \frac{p_{i,j}^{(n+1)} - p_{i,j} - C_{u} \left( T_{i,j} - T_{amb} \right)}{\Delta t} + ... \]

\[ 2k \frac{c_{p,i,j} T_{i,j}^{(n+1)} - c_{p,i,j} T_{i,j} - c_{p,i-1,j} T_{i-1,j}}{c_{p,i,j} + c_{p,i-1,j}} + ... \]

\[ 2k \frac{c_{p,i,j} T_{i,j}^{(n+1)} - c_{p,i,j} T_{i,j} - c_{p,i,j-1} T_{i,j-1}}{c_{p,i,j} + c_{p,i,j-1}} + ... \]

\[ 2k \frac{c_{p,i,j} T_{i,j}^{(n+1)} - c_{p,i,j} T_{i,j} - c_{p,i,j+1} T_{i,j+1}}{c_{p,i,j} + c_{p,i,j+1}} + ... \]

\[ 2k \frac{c_{p,i,j} T_{i,j}^{(n+1)} - c_{p,i,j} T_{i,j} - c_{p,i+1,j} T_{i+1,j}}{c_{p,i,j} + c_{p,i+1,j}} + ... \]

\[ \frac{\sigma c \phi}{16 \cdot \Delta x} \left[ \left( T_{i,j} + T_{i+1,j} \right)^{4} - \left( T_{i,j} + T_{i,j} \right)^{4} \right] - \frac{\sigma c \phi}{16 \cdot \Delta y} \left[ \left( T_{i,j} + T_{i,j+1} \right)^{4} - \left( T_{i,j} + T_{i,j} \right)^{4} \right] \]

EQ.6: Fuel consumption rate (Arrhenius) on scalar grid.

\[ r_{i,j}^{(n+1)} = A \cdot T_{i,j} \left( \rho_{i,j}^{(n+1)} \right)^{0.3} X_{j}^{0.5} X_{O_2} \exp \left( \frac{T_{a}}{T_{i,j}} \right) \]
And below other closure equation for combustion enthalpy, molar fraction of each species in time and mixture molar mass respectively.

\[ C1: \quad h_{c,j}^{(n+1)} = \frac{mH(T_{i,j})}{M_{i,j}} = -\sum_{s=1}^{4} \theta_s \cdot H_s(T_{i,j}) = -\sum_{s=1}^{4} \theta_s \cdot \left( H_{s,ref} + M_s \cdot cp_s \cdot (T_{i,j} - T_{ref}) \right) \]

\[ C2: \quad X^{(n+1)}_{i,j,s} = X^{(n)}_{i,j,s} - \Delta t \cdot \theta_s \cdot \frac{M_{i,j}}{M_{fuel}} \cdot \dot{l}^{(n+1)}_{i,j} \quad \text{(Molar fraction s-th species)} \]

\[ C3: \quad M_{i,j}^{(n+1)} = \sum_{s=1}^{5} X^{(n+1)}_{i,j,s} \cdot M_s \quad \text{(Molar mass of the mixture)} \]

\[ C4: \quad cp_{i,j}^{(n+1)} = \sum_{s=1}^{5} \frac{X^{(n+1)}_{i,j,s} \cdot M_s}{M_{i,j}^{(n+1)}} \cdot cp_s \quad \text{(mixture heat capacity)} \]

\textbf{Note:} \( \theta_s \) is the coefficient for each ‘s’ species in the reaction (3.9), defined negative if in the left-hand-side and positive if in the right-hand-side.

---

FD formulation considering boundary conditions:

\textbf{EQ.1: on scalar grid}

\[
\begin{align*}
\frac{\rho_{i,j}^{(n+1)} - \rho_{i,j}}{\Delta t} & + \ldots \\
& = \left[ (1 - \delta_{i,nx-1}) \rho_{i,j} + \rho_{i+1,j} \right] + 2 \cdot \delta_{i,nx-1} \cdot \rho_E \cdot \left[ (1 - \delta_{i,nx-1} \mu_{i,j} + \delta_{i,nx-1} \cdot u_{nx-1,j} \right] + \\
& - \left[ (1 - \delta_{i,1}) \rho_{i-1,j} + \rho_{i,j} \right] + 2 \cdot \delta_{i,1} \cdot \rho_W \cdot \left[ (1 - \delta_{i,1} \mu_{i,j} + \delta_{i,1} \cdot u_{1,j} \right] + \\
& - \left[ (1 - \delta_{j,ny}) \rho_{i,j} + \rho_{i,j+1} \right] + 2 \cdot \delta_{j,ny} \cdot \rho_N \cdot \left[ (1 - \delta_{j,ny} \nu_{i,j} + \delta_{j,ny} \cdot v_{ny} \right] + \\
& - \left[ (1 - \delta_{j,1}) \rho_{i,j} + \rho_{i,j-1} \right] + 2 \cdot \delta_{j,1} \cdot \rho_S \cdot \left[ (1 - \delta_{j,1} \nu_{i,j} + \delta_{j,1} \cdot v_{1} \right] = 0
\end{align*}
\]
EQ.2: on ‘u’ grid

\[
\left( \frac{\rho_{i+1,j}^{(n+1)} + \rho_{i,j}^{(n+1)}}{2 \Delta t} \right) u_{i+1,j}^{(n+1)} - \left( \frac{\rho_{i+1,j} + \rho_{i,j}}{2} \right) u_{i,j} + ... \\
\rho_{i,j} \left( u_{i,j} + (1 - \delta_{i,n+1}) u_{i+1,j} + \delta_{i,n+1} \cdot u_{i,j} \right)^2 - \rho_{i,j} \left( u_{i,j} + (1 - \delta_{i,l}) u_{i-1,j} + \delta_{i,l} \cdot u_{i,j} \right)^2 + ... \\
\left[ (1 - \delta_{j,m,y}) \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j-1} + \rho_{i+1,j-1} \right] + 4 \cdot \delta_{j,m,y} \cdot \rho_{N} \left[ (1 - \delta_{j,m,y}) \left( u_{i,j} + u_{i,j+1} \right) + 2 \cdot \delta_{j,m,y} \cdot u_{i,j} \right] \\

16 \cdot \Delta y
\]

EQ.3: on ‘v’ grid

\[
\left( \frac{\rho_{i,j+1}^{(n+1)} + \rho_{i,j}^{(n+1)}}{2 \Delta t} \right) v_{i,j+1}^{(n+1)} - \left( \frac{\rho_{i,j+1} + \rho_{i,j}}{2} \right) v_{i,j} + ... \\
\rho_{i,j} \left( v_{i,j} + (1 - \delta_{j,m,x}) v_{i,j+1} + \delta_{j,m,x} \cdot v_{i,j} \right)^2 - \rho_{i,j} \left( v_{i,j} + (1 - \delta_{j,l}) v_{i,j-1} + \delta_{j,l} \cdot v_{i,j} \right)^2 + ... \\
\left[ (1 - \delta_{j,m,x}) \rho_{i,j} + \rho_{i,j+1} + \rho_{i,j-1} + \rho_{i+1,j-1} \right] + 4 \cdot \delta_{j,m,x} \cdot \rho_{E} \left[ (1 - \delta_{j,m,x}) \left( u_{i,j} + u_{i,j+1} \right) + 2 \cdot \delta_{j,m,x} \cdot u_{i,x} \right] \\

16 \cdot \Delta x
\]

\[
\left( \frac{\rho_{i,j+1}^{(n+1)} + \rho_{i,j}^{(n+1)}}{2 \Delta t} \right) v_{i,j} + \frac{1}{2} \left( \rho_{i,j+1}^{(n+1)} + \rho_{i,j}^{(n+1)} \right) v_{i,j} + ... \\
\rho_{i,j} \left( v_{i,j} + (1 - \delta_{j,m,x}) v_{i,j+1} + \delta_{j,m,x} \cdot v_{i,j} \right)^2 - \rho_{i,j} \left( v_{i,j} + (1 - \delta_{j,l}) v_{i,j-1} + \delta_{j,l} \cdot v_{i,j} \right)^2 + ... \\
\left[ (1 - \delta_{j,m,x}) \rho_{i,j} + \rho_{i,j+1} + \rho_{i,j-1} + \rho_{i+1,j-1} \right] + 4 \cdot \delta_{j,m,x} \cdot \rho_{E} \left[ (1 - \delta_{j,m,x}) \left( u_{i,j} + u_{i,j+1} \right) + 2 \cdot \delta_{j,m,x} \cdot u_{i,x} \right] \\

16 \cdot \Delta x
\]
\[ p_{i,j} = -R \frac{\rho_{i,j} T_{i,j}}{M_{i,j}} \]

\[ \rho_{i,j}^{(n+1)} c_{p_{i,j}}^{(n+1)} T_{i,j}^{(n+1)} - \rho_{i,j}^{(n)} c_{p_{i,j}}^{(n)} T_{i,j}^{(n)} + \rho_{i,j}^{(n+1)} h c_{i,j}^{(n+1)} r_{i,j}^{(n+1)} + \]

\[ \frac{1}{\Delta t} \left[ \left( 1 - \delta_{i,n} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) u_{i,j} + 8 \delta_{i,n} \rho_{E} c_{p_{E}} T_{E} u_{n-1,j} + \ldots \right] + \ldots \]

\[ \frac{1}{8 \cdot \Delta x} \left[ \left( 1 - \delta_{i,1} \right) \rho_{i,j} + \rho_{i-1,j} \right) (c_{p_{i,j}} + c_{p_{i-1,j}}) (T_{i,j} + T_{i-1,j}) u_{i-1,j} - 8 \delta_{i,1} \rho_{W} c_{p_{W}} T_{W} u_{W} + \ldots \right] + \ldots \]

\[ \frac{1}{\Delta y} \left[ \left( 1 - \delta_{i,1} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) v_{i,j} + 8 \delta_{i,1} \rho_{N} c_{p_{N}} T_{N} v_{i,n-1} + \ldots \right] + \ldots \]

\[ \frac{1}{\Delta t} \left[ \left( 1 - \delta_{i,n} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) v_{i,j} + 8 \delta_{i,n} \rho_{E} c_{p_{E}} T_{E} u_{n-1,j} + \ldots \right] + \ldots \]

\[ \frac{1}{\Delta y} \left[ \left( 1 - \delta_{i,1} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) v_{i,j} - 2 \delta_{i,1} \rho_{W} h c_{i,j} u_{W} + \ldots \right] + \ldots \]

\[ \frac{1}{\Delta t} \left[ \left( 1 - \delta_{i,n} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) v_{i,j} - 2 \delta_{i,1} \rho_{W} h c_{i,j} u_{W} + \ldots \right] + \ldots \]

\[ \frac{1}{\Delta t} \left[ \left( 1 - \delta_{i,n} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) v_{i,j} - 2 \delta_{i,1} \rho_{W} h c_{i,j} u_{W} \right] = \ldots \]

\[ \frac{1}{\Delta t} \left[ \left( 1 - \delta_{i,n} \right) \rho_{i,j} + \rho_{i,j+1} \right) (c_{p_{i,j}} + c_{p_{i,j+1}}) (T_{i,j} + T_{i,j+1}) v_{i,j} - 2 \delta_{i,1} \rho_{W} h c_{i,j} u_{W} \right] = \ldots \]
Final formulation:

EQ.1: on scalar grid (hyp: \( \rho_E = \rho_S = \rho_N = \rho_W = \rho_0 \)) (unknown: \( \rho^{(n+1)} \))

\[
\rho_{i,j} = \rho_{i,j} + \Delta t \cdot \left\{ \begin{array}{l}
\rho_{i,j} \left[ -\frac{(1-\delta_{i,n-1})}{2\Delta x} \left[ (1-\delta_{j,n-1}) u_{i,j} + \delta_{j,n-1} \cdot u_{n-1,j} \right] + \frac{(1-\delta_{i,1})}{2\Delta x} \left[ (1-\delta_{j,1}) u_{i,j} + \delta_{j,1} \cdot u_w \right] + \ldots \right] + \\
\rho_{i,j} \left[ -\frac{(1-\delta_{j,n-1})}{2\Delta y} \left[ (1-\delta_{j,n-1}) v_{i,j} + \delta_{j,n-1} \cdot v_{i,n} \right] + \frac{(1-\delta_{j,1})}{2\Delta y} \left[ (1-\delta_{j,1}) v_{i,j} + \delta_{j,1} \cdot v_s \right] + \ldots \right] + \\
- \rho_{i+1,j} \left[ \frac{(1-\delta_{i,n-1})}{2\Delta x} \left[ (1-\delta_{i,n-1}) u_{i,j} + \delta_{i,n-1} \cdot u_{i-1,j} \right] + \frac{(1-\delta_{i,1})}{2\Delta x} \left[ (1-\delta_{i,1}) u_{i,j} + \delta_{i,1} \cdot u_w \right] + \ldots \right] + \\
- \rho_{i,j+1} \left[ \frac{(1-\delta_{j,n-1})}{2\Delta y} \left[ (1-\delta_{j,n-1}) v_{i,j} + \delta_{j,n-1} \cdot v_{i-1,j} \right] + \frac{(1-\delta_{j,1})}{2\Delta y} \left[ (1-\delta_{j,1}) v_{i,j} + \delta_{j,1} \cdot v_s \right] + \ldots \right] + \\
- \frac{\rho_E}{\Delta x} \left[ (1-\delta_{i,n-1}) u_{i,j} + \delta_{i,n-1} \cdot u_{i-1,j} \right] + \frac{\rho_w}{\Delta x} \left[ (1-\delta_{i,1}) u_{i,j} + \delta_{i,1} \cdot u_w \right] + \ldots \\
- \frac{\rho_N}{\Delta y} \left[ (1-\delta_{j,n-1}) v_{i,j} + \delta_{j,n-1} \cdot v_{i-1,j} \right] + \frac{\rho_s}{\Delta y} \left[ (1-\delta_{j,1}) v_{i,j} + \delta_{j,1} \cdot v_s \right] + \ldots 
\end{array} \right. \]

Simplified (to be considered for implementation):

\[
\rho_{i,j}^{(n+1)} = \rho_{i,j} + \Delta t \cdot \left\{ \begin{array}{l}
\rho_{i,j} \left[ -\frac{(1-\delta_{i,n-1})}{2\Delta x} \left[ (1-\delta_{j,n-1}) u_{i,j} + \delta_{j,n-1} \cdot u_{n-1,j} \right] + \frac{(1-\delta_{j,1})}{2\Delta y} \left[ (1-\delta_{j,1}) v_{i,j} + \delta_{j,1} \cdot v_{j+1} \right] + \ldots \right] + \\
- \rho_{i,j} \left[ (1-\delta_{i,n-1}) v_{i,j} + \delta_{i,n-1} \cdot v_{n-1,j} \right] + \frac{(1-\delta_{i,1})}{2\Delta x} \left[ (1-\delta_{i,1}) u_{i,j} + \delta_{i,1} \cdot u_w \right] + \ldots \\
- \rho_{i,j} \left[ (1-\delta_{j,n-1}) v_{i,j} + \delta_{j,n-1} \cdot v_{i-1,j} \right] + \frac{(1-\delta_{j,1})}{2\Delta y} \left[ (1-\delta_{j,1}) v_{i,j} + \delta_{j,1} \cdot v_s \right] + \ldots \\
- \frac{\rho_E}{\Delta x} \left[ (1-\delta_{i,n-1}) u_{i,j-1} + \delta_{i,n-1} \cdot u_{n-1,j} \right] + \frac{\rho_w}{\Delta x} \left[ (1-\delta_{i,1}) u_{i,j} + \delta_{i,1} \cdot u_w \right] + \ldots \\
- \frac{\rho_N}{\Delta y} \left[ (1-\delta_{j,n-1}) v_{i,j} + \delta_{j,n-1} \cdot v_{i-1,j} \right] + \frac{\rho_s}{\Delta y} \left[ (1-\delta_{j,1}) v_{i,j} + \delta_{j,1} \cdot v_s \right] + \ldots 
\end{array} \right. \]

134
EQ.2: on ‘u’ grid (unknown: $u_{i,j}^{n+1}$) The non-linear term in $u$ is between {} brackets.

\[
\begin{align*}
\frac{1}{2 \cdot \Delta t} \left[ u_{i,j}^{n+1} \left( \rho_{i+1,j}^{n+1} + \rho_{i,j}^{n+1} \right) + \mu \left( \frac{2 - \delta_{i,nx-1}}{\Delta x^2} + \frac{2 - \delta_{j,ny}}{\Delta y^2} \right) \right] - \frac{\mu (1 - \delta_{i,j})}{\Delta x^2} - \frac{\mu (1 - \delta_{i,j})}{\Delta y^2} - \frac{\mu (1 - \delta_{j,ny})}{\Delta x^2} - \frac{\mu (1 - \delta_{nx-1,j})}{\Delta y^2} + ... \\
\end{align*}
\]

\[
\begin{align*}
- \frac{\mu (1 - \delta_{i,j})}{\Delta x^2} - \frac{\mu (1 - \delta_{j,ny})}{\Delta y^2} = \frac{p_{i,j} - p_{i+1,j}}{\Delta x} + ... \\
\end{align*}
\]

\[
\begin{align*}
\frac{1}{4 \cdot \Delta x} \left[ \left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1} \right) + 4 \cdot \delta_{j,ny} \cdot \rho_N \right] \left( 1 - \delta_{j,ny} \right) + 2 \delta_{j,ny} \\
\end{align*}
\]

\[
\begin{align*}
\frac{1}{16 \cdot \Delta y} \left[ \left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1} \right) + 4 \cdot \delta_{j,ny} \cdot \rho_N \right] \left( 1 - \delta_{j,ny} \right) + 2 \delta_{j,ny} \\
\end{align*}
\]

\[
\begin{align*}
\frac{1}{16 \cdot \Delta y} \left[ \left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1} \right) + 4 \cdot \delta_{j,ny} \cdot \rho_N \right] \left( 1 - \delta_{j,ny} \right) + 2 \delta_{j,ny} \\
\end{align*}
\]

\[
\begin{align*}
\frac{1}{8 \cdot \Delta y} \left[ \left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1} \right) + 4 \cdot \delta_{j,ny} \cdot \rho_N \right] \left( 1 - \delta_{j,ny} \right) + 2 \delta_{j,ny} \\
\end{align*}
\]

\[
\begin{align*}
\frac{\mu (1 - \delta_{i,j})}{\Delta x^2} + \frac{\mu (1 - \delta_{j,ny})}{\Delta y^2} \\
\end{align*}
\]
Simplified (to be considered for implementation):

\[
\begin{align*}
&\mathbf{u}_{i,j}^{n+1} \left[ \frac{(\rho_{i+1,j}^{n+1} + \rho_{i,j}^{n+1})}{2 \cdot \Delta t} + \mu \left( \frac{2 - \delta_{i,\text{ax}-1}}{\Delta x^2} + \frac{2 - \delta_{i,\text{ay}}}{\Delta y^2} \right) \right] - \mathbf{u}_{i,j}^{n+1} \mu \left( \frac{1 - \delta_{i,\text{i}}}{\Delta x^2} - \mathbf{u}_{i,j}^{n+1} \mu \left( \frac{1 - \delta_{i,\text{ax}-1}}{\Delta x^2} \right) + \ldots \\
&- \mathbf{u}_{i,j}^{n+1} \frac{\mu(1 - \delta_{i,j})}{\Delta y^2} - \mathbf{u}_{i,j}^{n+1} \frac{\mu(1 - \delta_{i,j})}{\Delta y^2} = p_{i,j} - p_{i+1,j} + \ldots \\
&+ u_{i,j} \left[ \frac{(\rho_{i+1,j} + \rho_{i,j})}{2 \cdot \Delta t} - \frac{\delta_{j,\text{ay}} \cdot \rho_{i,j} \cdot v_{i,j+1} + \delta_{i,\text{ax}} \cdot \rho_{i,j} \cdot w_j}{2 \cdot \Delta y} + \frac{\delta_{j,\text{ay}} \cdot \rho_{i,j+1} + \rho_{i,j+1} + \rho_{i,j+1}}{2 \cdot \Delta y} \right] + \ldots \\
&+ \frac{(1 - \delta_{i,j}) \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i,j+1}}{16 \cdot \Delta y} + \ldots \\
&+ \frac{(1 - \delta_{i,j}) \rho_{i,j} + \rho_{i,j+1} + \rho_{i,j+1} + \rho_{i,j+1}}{16 \cdot \Delta y} + \ldots \\
&\rho_{i,j} \left( 1 - \delta_{i,j} \right) u_{i,j} \rho_{i,j} - \rho_{i,j} \left( 1 - \delta_{i,\text{ax}-1} \right) u_{i,j} \\
&- u_{i,j}^{n+1} \frac{(1 - \delta_{i,j}) \rho_{i,j} + \rho_{i,j+1} + \rho_{i,j+1} + \rho_{i,j+1}}{16 \cdot \Delta y} + \ldots \\
&+ \frac{(1 - \delta_{i,j}) \rho_{i,j} + \rho_{i,j+1} + \rho_{i,j+1} + \rho_{i,j+1}}{16 \cdot \Delta y} + \ldots \\
&+ \frac{(\rho_{i,j} - \rho_{i,j+1})}{4 \cdot \Delta x} - \frac{3 \cdot \delta_{i,\text{ax}-1} \cdot \rho_{i,j+1}}{4 \cdot \Delta x} + \frac{\mu \delta_{j,i} \cdot w_j}{\Delta x^2} + \frac{\mu \delta_{j,i} \cdot s_j}{\Delta x^2} + u_s \frac{\delta_{j,i} \cdot \rho_{i,j} v_{i,j} + v_{i,j}}{2 \cdot \Delta y} + \ldots \\
&+ u_{i,j}^2 \left[ \frac{\rho_{i,j} - \rho_{i,j+1}}{4 \cdot \Delta x} + \frac{(1 - \delta_{i,\text{ax}-1}) \rho_{i,j}}{4 \cdot \Delta x} + \frac{\rho_{i,j+1} - \rho_{i,j+1}}{4 \cdot \Delta x} + \frac{\rho_{i,j+1} - \rho_{i,j+1}}{4 \cdot \Delta x} \right] + \ldots \\
&+ \frac{(1 - \delta_{i,j}) \rho_{i,j} - \rho_{i,j} \left( 1 - \delta_{i,\text{ax}-1} \right) \rho_{i,j}}{4 \cdot \Delta x} + \ldots \\
&+ u_{i,j}^2 \frac{(1 - \delta_{i,j}) \rho_{i,j} - \rho_{i,j} \left( 1 - \delta_{i,\text{ax}-1} \right) \rho_{i,j}}{4 \cdot \Delta x} + \ldots \\
&+ u_{i,j}^2 \frac{(1 - \delta_{i,j}) \rho_{i,j} - \rho_{i,j} \left( 1 - \delta_{i,\text{ax}-1} \right) \rho_{i,j}}{4 \cdot \Delta x} + \ldots \\
\end{align*}
\]
EQ.3: on 'V' grid (unknown: $V^{n+1}$) The non-linear term in $v$ is between { } brackets.

$$V^{n+1}_{i,j} = V^n_{i,j} + \mu \left( \frac{2 - \delta_{,nx}}{\Delta x^2} + \frac{2 - \delta_{,ny}}{\Delta y^2} \right) - \frac{\mu(1 - \delta_{,i,1})}{\Delta x^2} - \frac{\mu(1 - \delta_{,j,1})}{\Delta y^2} + ...$$

$$- V^n_{i+1,j} - V^n_{i-1,j} - V^n_{i,j+1} - V^n_{i,j-1} + \frac{\mu(1 - \delta_{,j,1})}{\Delta y} + ...$$

$$\left\{ \rho_{i,j} v_{i,j} + \frac{(1 - \delta_{,j,ny}) v_{i,j+1} + \delta_{,j,ny} v_{i,j+1} - \rho_{i,j} v_{i,j} + \left(1 - \delta_{,i,1}\right) v_{i,j+1} + \delta_{,j,1} v_{s} \right\}^{2} + ...$$

$$\frac{\rho_{i,j+1} + \rho_{i,j}}{2 \cdot \Delta t} \left[ \frac{1 - \delta_{,nx}}{\Delta x} \left( u_{i,j} + u_{i,j+1} \right) + \frac{\delta_{,nx} \cdot u_{x,-i,j}}{\Delta x} \right] + \frac{1}{16 \cdot \Delta x} \left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1} + \rho_{i-1,j} \cdot \rho_{W} \right) + ...$$

$$\frac{\rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1}}{16 \cdot \Delta x} \left[ \frac{1 - \delta_{,nx}}{\Delta x} \left( u_{i,j} + u_{i,j+1} \right) + \frac{2 \cdot \delta_{,nx} \cdot u_{x,-i,j}}{\Delta x} \right] + ...$$

$$\left( \rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1} + \rho_{i-1,j} \cdot \rho_{W} \right) \left[ \frac{1 - \delta_{,i,1}}{\Delta x} \left( u_{i,j} + u_{i,j+1} \right) + \frac{2 \cdot \delta_{,i,1} \cdot u_{x,-i,j}}{\Delta x} \right] + ...$$

$$\frac{\mu \delta_{,i,1} v_{W} + \mu \delta_{,j,1} v_{s}}{\Delta x^2} + \frac{\mu \delta_{,j,1} v_{s}}{\Delta y^2}$$
Simplified (to be considered for implementation):

\[
\begin{align*}
V_{i,j}^{n+1} & = \left[ \frac{\rho_{i,j}^{n+1} + \rho_{i,j}^{n}}{2 \cdot \Delta t} + \mu \left( \frac{2 - \delta_{i,j,x} + \Delta x^2}{\Delta x^2} + \frac{2 - \delta_{i,j,y} + \Delta y^2}{\Delta y^2} \right) \right] V_{i,j}^{n+1} - \frac{\mu(1 - \delta_{i,j,x})}{\Delta x^2} - \frac{\mu(1 - \delta_{i,j,y})}{\Delta y^2} + \ldots \\
- V_{i,j}^{n+1} & = \mu \frac{(1 - \delta_{i,j,x})}{\Delta x^2} - \mu \frac{(1 - \delta_{j,i,y})}{\Delta y^2} = \rho_{i,j}^{n+1} - \rho_{i,j+1}^{n+1} + \ldots \\
+ v_{i,j} & = \left[ \frac{\rho_{i,j}^{n+1} + \rho_{i,j}^{n}}{2 \cdot \Delta t} - \frac{\delta_{i,j,x} \cdot \rho E \cdot u_{i,j+1} - \delta_{i,j,y} \cdot \rho_{i,j} \cdot v_s}{2 \cdot \Delta y} + \ldots \right] \\
& - \frac{(1 - \delta_{i,j,x}) \rho_{i,j}^{n+1} + \rho_{i,j+1}^{n+1} + \rho_{i,j+1}^{n+1} \cdot u_{i,j} + u_{i,j+1}}{2 \cdot \Delta x} + \ldots \\
& + \frac{(1 - \delta_{i,j,y}) \rho_{i,j+1}^{n+1} \cdot v_{i,j+1} - \delta_{i,j,y} \cdot \rho_{i,j} \cdot v_{i,j-1}}{2 \cdot \Delta y} - \frac{(1 - \delta_{i,j,x}) \rho_{i,j}^{n+1} + \rho_{i,j+1}^{n+1} + \rho_{i,j+1}^{n+1} \cdot u_{i,j} + u_{i,j+1}}{2 \cdot \Delta x} + \ldots \\
& + \frac{v_{w} \delta_{i,j} \cdot v_{w} \cdot u_{w}}{\Delta x} + \frac{\mu \delta_{i,j} \cdot v_{w} \cdot x_{s}}{\Delta x^2} + \frac{\mu \delta_{i,j} \cdot v_{w} \cdot y_{s}}{\Delta y^2} + v_{i,j} \left[ \frac{\rho_{i,j+1}^{n+1} - \rho_{i,j}^{n+1}}{4 \cdot \Delta y} + \frac{3 \cdot \delta_{i,j,y-1} \cdot \rho_{i,j}^{n+1}}{4 \cdot \Delta y} \right] + \ldots \\
& + \frac{v_{i,j+1} \left[ \frac{1 - \delta_{i,j,y-1} \cdot \rho_{i,j+1}^{n+1}}{4 \cdot \Delta y} - v_{i,j-1} \left[ \frac{1 - \delta_{i,j,x} \cdot \rho_{i,j}^{n+1}}{4 \cdot \Delta y} - v_{s} \delta_{i,j} \cdot \rho_{i,j}^{n+1} \right] \right]}{4 \cdot \Delta y} \\
\end{align*}
\]

EQ.5: on scalar grid. (unknown \( \mathbf{T}_{i,j}^{n+1} \)) The Radiation non-linear term in \( T \) is between \( \{ \) \)

brackets.

(to be considered for implementation)
\[
\sum_{\nu=1}^{\mu+1} = \frac{\rho_{i,j} \Delta T_{i,j}}{c_p_{i,j} \rho_{i,j}} + \frac{\rho_{i,j}^{(n+1)} - \rho_{i,j}}{c_p_{i,j} \rho_{i,j}^{(n+1)}} + \frac{\Delta T_{i,j}}{c_p_{i,j} \rho_{i,j}^{(n+1)}} \left[ C_{\nu} T_{i,j}^{\text{amb}} - \rho_{i,j}^{(n+1)} h_{c_{i,j}}^{(n+1)} r_{i,j}^{(n+1)} \right] + ...
\]

\[
T_{i,j} = \left[ \begin{array}{c}
- C_{\nu} - \frac{1}{8 \cdot \Delta x} (1-\delta_{i,x})(\rho_{i,j} + \rho_{i,j+1})(c_{p_{i,j}} + c_{p_{i,j+1}}) \mu_{i,j} + ...
+ \frac{1}{8 \cdot \Delta x} (1-\delta_{i,1})(\rho_{i,j} + \rho_{i,j-1})(c_{p_{i,j}} + c_{p_{i,j-1}}) \mu_{i,j-1} + ...
+ \frac{1}{8 \cdot \Delta y} (1-\delta_{j,y})(\rho_{i,j} + \rho_{i,j+1})(c_{p_{i,j}} + c_{p_{i,j+1}}) \nu_{i,j} + ...
+ \frac{1}{8 \cdot \Delta y} (1-\delta_{j,1})(\rho_{i,j} + \rho_{i,j-1})(c_{p_{i,j}} + c_{p_{i,j-1}}) \nu_{i,j-1} + ...
- \frac{2k}{\Delta x} \left( \frac{cp_{i,j}}{cp_{i,j} + (1-\delta_{i,x})np_{i,j+1} + \delta_{i,x}np_{w} + \delta_{i,1}np_{i,j+1} + \delta_{i,1}np_{w}} \right)
\end{array} \right]
\]

\[
\Delta T_{i,j} = \left[ \begin{array}{c}
\frac{u_{i,j}}{8 \cdot \Delta x} (1-\delta_{i,x})(\rho_{i,j} + \rho_{i,j+1})(c_{p_{i,j}} + c_{p_{i,j+1}}) + \frac{2k}{\Delta x} \left( \frac{1-\delta_{i,x}(np_{i,j+1}}{np_{i,j+1} + \delta_{i,x}np_{w} + \delta_{i,1}np_{i,j+1} + \delta_{i,1}np_{w}} \right)
+ \frac{v_{i,j}}{8 \cdot \Delta y} (1-\delta_{j,y})(\rho_{i,j} + \rho_{i,j+1})(c_{p_{i,j}} + c_{p_{i,j+1}}) + \frac{2k}{\Delta y} \left( \frac{1-\delta_{j,y}(np_{i,j+1}}{np_{i,j+1} + \delta_{j,y}np_{w} + \delta_{j,1}np_{i,j+1} + \delta_{j,1}np_{w}} \right)
+ \frac{1}{8 \cdot \Delta x} (1-\delta_{i,x})(\rho_{i,j} + \rho_{i,j+1})(c_{p_{i,j}} + c_{p_{i,j+1}}) + \frac{2k}{\Delta x} \left( \frac{1-\delta_{i,x}(np_{i,j+1}}{np_{i,j+1} + \delta_{i,x}np_{w} + \delta_{i,1}np_{i,j+1} + \delta_{i,1}np_{w}} \right)
+ \frac{1}{8 \cdot \Delta y} (1-\delta_{j,y})(\rho_{i,j} + \rho_{i,j+1})(c_{p_{i,j}} + c_{p_{i,j+1}}) + \frac{2k}{\Delta y} \left( \frac{1-\delta_{j,y}(np_{i,j+1}}{np_{i,j+1} + \delta_{j,y}np_{w} + \delta_{j,1}np_{i,j+1} + \delta_{j,1}np_{w}} \right)
\end{array} \right]
\]

\[
\frac{1}{4 \cdot \Delta x} \left[ - (1-\delta_{i,x})(\rho_{i,j} + \rho_{i,j+1})(hc_{i,j} + hc_{i,j+1}) u_{i,j} - 2\delta_{i,x}(u)hc_{i,j} u_{i,x,j} + ... \right]
+ \frac{1}{4 \cdot \Delta y} \left[ - (1-\delta_{j,y})(\rho_{i,j} + \rho_{i,j+1})(hc_{i,j} + hc_{i,j+1}) v_{i,j} - 2\delta_{j,y}(u)hc_{i,j} v_{i,y,j} + ... \right]
- \frac{\sigma \phi}{16 \cdot \Delta x} \left( \left( (1-\delta_{i,x})(T_{i,j} + T_{i,j+1}) + 2\delta_{i,x} T_{w} \right)^{1/2} - \left( (1-\delta_{i,x})(T_{i-1,j} + T_{i,j}) + 2\delta_{i,x} T_{w} \right)^{1/2} \right) + ...
+ ... (continue next page)
\[
\begin{align*}
&\frac{\Delta M}{\rho_{i,j}^{(n+1)} cp_{i,j}^{(n+1)}}
\left[\frac{\delta_{j,ny} cp_{i,j} T_E}{cp_{i,j} + (1 - \delta_{j,ny}) cp_{i,j+1} + \delta_{j,ny} cp_{E}} + \ldots\right]
\left[\frac{\delta_{j,ny} cp_{i,j} T_N}{cp_{i,j} + (1 - \delta_{j,ny}) cp_{i,j+1} + \delta_{j,ny} cp_{N}} + \ldots\right]
\left[\frac{\delta_{j,ny} cp_{i,j} T_S}{cp_{i,j} + (1 - \delta_{j,ny}) cp_{i,j+1} + \delta_{j,ny} cp_{S}} + \ldots\right]
\right]
\end{align*}
\]
\[ \tau_{i,j}^{n+1} = \frac{\rho_{i,j} c p_{i,j}}{\rho_{i,j}^{(n+1)} c p_{i,j}^{(n+1)}} T_{i,j}^{n+1} + \frac{\rho_{i,j}^{(n+1)} c p_{i,j}^{(n+1)}}{\rho_{i,j}^{(n+1)} c p_{i,j}^{(n+1)}} + \frac{\Delta t}{\rho_{i,j}^{(n+1)} c p_{i,j}^{(n+1)}} \left[ \Delta T_{i,j}^{n} - \rho_{i,j}^{(n+1)} \Delta c T_{i,j}^{n} - \rho_{i,j}^{(n+1)} h c_{i,j}^{(n+1)} \right] + \ldots \]
\[
\begin{align*}
\Delta T = & \frac{1}{\rho_{i,j} \cdot \Delta x} \left( \frac{\partial \rho E \cdot cp E \cdot T_E \cdot \delta_{i,n-1,1} + \partial \rho w \cdot cp w \cdot T_w \cdot \delta_{i,1} - \delta_{j,ny} \cdot \rho N \cdot cp N \cdot T_N \cdot \delta_{i,ny-1} + \delta_{j,1} \cdot \rho s \cdot cp S \cdot T_S \cdot \delta_{i,1}}{\Delta x} \right) \\
& + \frac{2k \left( \delta_{i,n,1} \cdot cp E + \delta_{i,1} \cdot cp w \cdot T_w \right)}{\Delta x \cdot (cp_{i,j} + cp E)} \\
& + \frac{2k \left( \delta_{j,n,1} \cdot cp N \cdot T_N + \delta_{i,1} \cdot cp S \cdot T_S \right)}{\Delta y \cdot (cp_{i,j} + cp N)}
\end{align*}
\]
Figure D.1. Scheme for equation 1: on density grid.

Figure D.2. Scheme for equation 2: on ‘U’ grid.
Figure D.3. Scheme for equation 3: on ‘V’ grid.
REFERENCES


