Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Máster en Ingeniería Industrial

NUMERICAL SIMULATION OF EVAPORATION MODEL IN A WENO-Z BASED EULERIAN-LAGRANGIAN CODE

Autor:

Arenas Garrido, José Alejandro

María Teresa Parra Santos

San Diego State University

Valladolid, Julio 2018.

TFM REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO: NUMERICAL SIMULATION OF EVAPORATION MODEL IN A WENO-Z BASED EULERIAN-LAGRANGIAN CODE

ALUMNO: José Alejandro Arenas Garrido

FECHA: 23/07/2016

CENTRO: San Diego State University

TUTOR: Gustaaf Jacobs

Resumen

En este TFM se discute la implementación de un modelo de evaporación en un código WENO-Z de alto orden basado en el modelo Euleriano-Lagrangiano (EL) para un flujo de partículas a alta velocidad en cámaras de combustión supersónicas. El método EL aproxima la equacioens de Euler que gobiernan el movimiento del gas con el esquema mejorado "high order weighted essentially non-oscillatory" (WENO-Z), mientras que las partículas individuales son trazadas en el marco Lagrangiano utilizando esquemas de integración de alto orden. Tanto el gas portador como las partículas derivadas en función el tiempo mediante el método Runge-Kutta TVD de tercer orden. Una interpolación de alto orden (ENO) determina las propiedades del gas portador en la posición de las partículas. La evaporación de una partícula es verificada con la ley D^2 de evaporación y se estudia el problema unidimensional de la interacción de una onda de choque con una nube de partículas.

Palabras clave: Partículas, Evaporación, Onda de choque, Euleriano-Lagrangiano, WENO, WENO-Z.

Abstract

This thesis discusses the implementation of a evaporation model into a high-order WENO-Z based Eulerian-Lagrangian (EL) code for simulation of high-speed droplet-laden flow in supersonic combustors. The EL method approximates the Euler equations governing gas dynamics with the improved high order weighted essentially non-oscillatory (WENO-Z) scheme, while individual particles are traced in the Lagrangian frame using high-order time integration schemes. Both the carrier gas and the particles are updated in time without splitting with a third-order Runge-Kutta TVD method. A high-order ENO interpolation determines the carrier phase properties at the particle position. A high-order central weighting deposits the particle influence on the carrier phase. The droplet evaporation is verified against the D^2-law and study the one-dimensional problem of a shock wave running into a cloud of particles.

Palabras clave: Droplet, Evaporation, Sock-wave, Eulerian-Lagrangian, WENO, WENO-Z.

NUMERICAL SIMULATION OF EVAPORATION MODEL IN A WENO-Z BASED EULERIAN-LAGRANGIAN CODE

Author: José Alejandro Arenas Garrido

Advisor: Dr. Gustaaf B. Jacobs

Máster en Ingeniería Industrial

Universidad de Valladolid

San Diego, July of 2018

DEDICATION

To my parents.

ABSTRACT

NUMERICAL SIMULATION OF EVAPORATION MODEL IN A WENO-Z BASED EULERIAN-LAGRANGIAN CODE

José Alejandro Arenas Garrido
Universidad de Valladolid
July of 2018

This thesis discusses the implementation of a evaporation model into a high-order WENO-Z based Eulerian-Lagrangian (EL) code for simulation of high-speed droplet-laden flow in supersonic combustors. The EL method approximates the Euler equations governing gas dynamics with the improved high order weighted essentially non-oscillatory (WENO-Z) scheme, while individual particles are traced in the Lagrangian frame using high-order time integration schemes. Both the carrier gas and the particles are updated in time without splitting with a third-order Runge-Kutta TVD method. A high-order ENO interpolation determines the carrier phase properties at the particle position. A high-order central weighting deposits the particle influence on the carrier phase. The droplet module is verified against the D^2 -law and study the one dimensional problem of a shock wave running into a cloud of particles.

Keywords: *Droplet: Evaporation; Shock-wave; Eulerian-Lagrangian; WENO; WENO-Z.*

ACKNOWLEDGEMENTS

I would like to express my gratitude towards Dr. Gustaaf Jacobs from San Diego State University, who has been a great academic advisor and without whose constant help and advice I would not have been able to complete this work.

I would like to thank to my Spanish advisor, Teresa Parra Santos, from the University of Valladolid, without whom it would not have been possible to do this internship in California.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

A large diversity of multi-phase gas-liquid involve evaporation of near spherical liquid droplets in high-speed gas flows. Examples may be found in technologies such as supersonic aircraft, hypersonic space vehicles, gas turbines and explosions. Shocks are encountered in these type of flows and in many high-speed flows applications, shocks interact with solid or liquid particles. For example, fuel droplets interact with a chemically reacting gas containing shock waves in high-speed combustors. Droplet evaporation plays a crucial role in the fuel -air mixing process and consequently the performance of the combustor. The precise prediction of the evaporation time and the movement of droplets is important for optimum design of scramjet combustion chambers. All of these situations involve a dispersed liquid phase species in the form of a large number of discrete droplets convecting and vaporizing in a continuous gas phase species, and their mathematical description involves complex nonlinear coupling of momentum, energy and mass exchange.

The classical WENO schemes owe their success to the use of a dynamic set of stencils, where a nonlinear convex combination of lower order polynomials adapts either to a higher order approximation at smooth parts of the solution , or to an upwind spatial discretization that avoids interpolation across discontinuities and provides the necessary dissipation for shock capturing. The classical WENO schemes are designed based on the successful class of high order schemes called the essentially non-oscillatory or ENO schemes of Harten *et al.* [1,2]. The first classical WENO scheme was introduced by Liu *et al.* in their pionnering paper citeliu1994, in which a third order accurate fine volume WENO scheme was designed. In 1996, Jiang and Shu [3] provided a general framework to construct arbitrary order accurate finite difference WENO schemes, which are more efficient for multi-dimensional calculations. Higher order finite difference classical WENO schemes are designed in [4]. Jacobs and Don developed a high order WENO-Z interpolation from the carrier flow to the particles and a high order weighing of the momentum and energy from the particle t the carrier flow , yielding a high order solution of particle-laden gas flow with shocks in the Eulerian-Lagrangian (EL) frame [5].

In this paper, we discuss the implementation of a evaporation models presented by Miller [6] into a high-order WENO-Z based Eulerian-Lagrangian code [5] for simulation of high-speed droplet-laden flow in supersonic combustor.

A significant body of literature exists for modeling droplet evaporation. Mashayek *et al.* [7] simulate doplet dispersion in isotropic turbulence for wich the evaporation is governed by the D^2-law and the mass loading is considered small enough to neglect turbulence modulation by the dispersed phase. The first well established evaporation model is the spherically symmetric D^2-law developed by Spalding [8] for an isolated single-component droplet in a stagnant environment. In this basic lumped model, quasi-steadiness is assumed and the

droplet temperature is uniform and held constant at a reference temperature. Under these assumptions the D^2 -law predicts a linear relationship for the change in droplet surface area with time. Mashayek *et al.* [7] removed this restriction and considered droplet dispersion in compressible homogeneous turbulence with two-way coupling and droplet evaporation governed by a heat-mass model.

This paper is organized as follows: In chapter2, the physical model of the carrier flow and the particles are presented. In chapter 3, the coupling of the carrier flow in the Eulerian frame and the particle in the Lagrangian frame is discussed in detail. We describe the high order ENO interpolation scheme that is used to interpolate the gas properties from the Eulerian frame to the particle in the Lagrangian frame. The high order weighing for exchange of momentum and energy from the Eulerian frame to the carrier are also given. We validate the numerical methods against analytical solution in chapter 3 and we present simulations for an evaporating cloud of liquid droplets in chapter 4. Finally, we consider the one dimensional problem of a shock wave running into a cloud of particles as conclusionin the chapters 5 and direction of future research are reserved in chapter 6 .

CHAPTER 2

Governing Equations

2.1 Droplet Equations

2.1.1 Dimensional Form

Particles are tracked individually in the Lagrangian frame. The kinematic equation describing the particle's position x'_p, is given as

$$\frac{dx'_i}{dt'} = v'_i, \tag{2.1}$$

where v'_p is the dimensional particle velocity.

The particles acceleration is governed by Newton's second law forced by the drag on the particle. With particles assumed spherical, we take the drag as as combination of the Stokes drag corrected for Reynolds and March number and the pressure drag leading to the following equations governing the particle velocity,

$$\frac{dv'_i}{dt'} = \frac{18\mu f_1}{\rho'_p d'^2_p} * (u'_i - v'_i), \tag{2.2}$$

where v'_f is the dimensional velocity of the gas at the particle position, and ρ'_p the dimensional particle density. The first term describes the particle acceleration resulting from the velocity difference between the particle and the gas. The second term in the right hand represents the particle acceleration introduced by the pressure gradient in the carrier flow at particle position.

From the first law of thermodynamics and Fourier's law for heat transfer , the equation for temperature is derived as,

$$m'_p C_l \frac{dT'_p}{dt'} = Nu\pi K_f D'_p (T' - T'_p) + L_v \frac{dm'_p}{dt'}, \tag{2.3}$$

where Nu is the Nusselt number correcter for high Reynolds number.

Finally, the droplet continuity equation The mass rate of change of the droplet due to mass flux through the control surface is

$$\frac{dm'_p}{dt'} = -Sh\rho'_f D\pi d'_p (Y_s - Y_f), \tag{2.4}$$

where Y_s and Y_f are mass fraction of vapor at the droplet surface and mass fraction of vapor in carrier phase respectively, and Sh is the Sherwood number. The Sherwood number is 2 for a droplet which is evaporation with radial symmetry (no forced or free convection effects) [9].

2.1.2 Normalization

All of the variables are normalized by using reference values ρ_o, u_o, l_o and T_o.

$$\begin{array}{ll} x_i = x_i'/l_o, & t_i = t'/(l_o/u_o) = u_o t'/l_o \\ u_i = u_i'/u_o, & p_i = p'/(\rho_o u_o^2) \\ T = T'/T_o & \end{array}$$

Non-dimensional numbers:

$$\begin{array}{lll} Re_o = \frac{\rho_o u_o l_o}{\mu}, & Sc = \frac{\nu}{D}, & D_a = \frac{k\rho_o l_o}{u_o}, \\ M_o = \frac{u_o^2}{\gamma R T_o}, & \lambda = \frac{L_v}{C_p T_o}. & \end{array}$$

2.1.3 Non-Dimensional Form

The droplet trajectories, mass and temperature in non-dimensional form are obtained by using the normalization equivalences described in below. The non-dimensional Lagrangian equations governing the position x_p, velocity u_p and temperature T_p of each droplet that consist of diameter d_p and mass m_p,

$$\frac{dx_i}{dt} = v_i, \tag{2.5}$$

$$\frac{dv_i}{dt} = \frac{f_1}{\tau_p}(u_{fp,i} - u_{p,i}), \tag{2.6}$$

$$\frac{dT_p}{dt} = \frac{f_2 A_2 C_1}{\tau_p}(T_{fp} - T_p) - \frac{f_3}{\tau_p}(Y_s - Y_{inf}), \tag{2.7}$$

$$\frac{dm_p}{dt} = -f_4 \tau_p^{0.5}(Y_s - Y_{inf}), \tag{2.8}$$

where Sherwood (Sh) numbers are the empirically modified convective correlations to heat and mass transfer by Ranz-Marshal

$$Nu = \frac{2 + 0.6 Re_p^{0.5} Pr^{0.33}}{1 + B}, \tag{2.9}$$

$$Pr = \frac{C_p \mu}{K_f}, \tag{2.10}$$

$$Sh = 2 + 0.6 Re_p^{0.5} Sc^{0.33} = 2 \tag{2.11}$$

The evaporation rate [10] can be determined by the local slip vapor mass fraction. It is equal to the non-dimensional vaporization pressure of the droplet and obeys the Clausius-Clapeyron equation, in normalized form given by:

$$Y_s = \frac{P_B}{P_{fluid}} exp\left[\frac{\gamma \alpha_1}{(\alpha - 1) T_B}\left(1 - \frac{T_B}{T_P}\right)\right], \tag{2.12}$$

where T_B is the boiling point of the liquid and is assumed to be independent of pressure, which is generally a small varying parameter for evaporation conditions.

In Equation 2.12 the density of the droplet is considered to be constant and much larger than the density of the carrier phase such a that only the inertia and the drag forces are significant to the droplet dynamics.

In the code we have implemented a simplification presented by Mashayek [7] for slow evaporation,

$$Y_s = exp\left[\frac{\gamma \alpha_1}{(\alpha - 1) T_B}\left(1 - \frac{T_B}{T_P}\right)\right]. \tag{2.13}$$

Correction factor f_1 is for Stokes drag, f_2 represents a correlation for the convective heat transfer coefficient, and the Nusselt (Nu). Finally, f_3 and f_4 are both an function of the empirically correcter Sherwood number.

$$f_1 = \frac{C_d Re_p}{24}, \tag{2.14}$$

$$f_2 = \frac{Nu}{3 Pr \alpha_2}, \tag{2.15}$$

$$f_3 = \frac{A_1 A_2}{3Sc} \rho_f Sh, \tag{2.16}$$

$$f_4 = \frac{\pi (18/\rho_p)^{1/2}}{Re_o^{1.5} Sc} \rho_f Sh, \tag{2.17}$$

where $A_1 =$ is the normalized latent heat of evaporation: $A_1 = L_v/(C_{p,g} T_f)$ and A_2 is the ratio of gas capacities between the carrier gas and the liquid droplet phase is $A_2 = C_{p,p}/C_p$. In the

code we use a simplification presented by Mashayek, $A_2 = 0$. The particle response time, τ_p, is defined in terms of d_p, $\tau_p = \rho_p d_p^2 Re_o/18$.

In the present study study we use this correlation for the standard drag curve,

$$C_D = \frac{24(1 + 0.15Re_p^{0.687}}{Re_p(1 + B}. \tag{2.18}$$

2.2 Continuum Equations

The governing Euler equations for the continuum phase are the conservation laws for mass, momentum and energy. As such, the carrier phase is modeled as a compressible Newtonian and inviscid fluid in this code. We shall denote the subscript p or $part$ for the droplet variables and f or $fluid$ for the gas variables at particle position. The equations are presented in a non-dimensional form,

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = S_{mass}, \tag{2.19}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j + p\delta_{ij})}{\partial x_j} = S_{momentum}, \tag{2.20}$$

$$\frac{\partial E}{\partial t} + \frac{\partial\left[(E + pi)u_i\right]}{\partial x_i} = S_{energy}, \tag{2.21}$$

where ρ, u_i, p and E are the density, velocitu, thermodynamic pressure, and total energy per volume unit. The heat conduction, is modeled within the source terms and it is assumed that the carrier gas obeys the perfect gas law. The conservation for the vapor mass fraction in non-dimensional form is described as:

$$\frac{\partial \rho Y}{\partial t} + \frac{\partial(\rho Y_i u_i)}{\partial x_i} = S_{mass}, \tag{2.22}$$

with Y, the mass fraction of the evaporated liquid vapor species.

For simplicity, the vapor is assumed to have the same molecular weigh, mass diffusivity and specific heat as those of hte gas. In this manner, the gas-vapor mixture is trated as one entity and the equations for the continuum describe the combined gas-vapor flow. The energy equation is written sich that it governs the total energy,

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) \tag{2.23}$$

The set of equations is closed by the following equations of state,

$$p = \frac{\rho T}{\gamma M_f^2} \tag{2.24}$$

where the refence Mach number is $M_f = U_f/\sqrt{\gamma R T_f}$. Here R denotes the universal gas constant, γ denotes the ratio of C_p/C_v. All the variables used in the equations are normalized using refences values (see 2.1.2).

2.3 Source terms S for the Euler equation

Each particle generates a momentum and energy that affects the carrier flow. The volume averaged summation of all these contribution of all these contributions give a continuum source contribution on the momentum and energy equation,

$$S_m = \sum_{i=1}^{N_p} K(\vec{x}_p, \vec{x}) \vec{W}_m, \tag{2.25}$$

$$S_e = \sum_{i=1}^{N_p} K(\vec{x}_p, \vec{x}) (\vec{W}_m \vec{v}_p + W_e), \tag{2.26}$$

where $K(x, y) = K(|x - y|)/V$ is normalized weighing function that distributes the influence of each particle onto the carrier flow. And N˙p is the total number of particles in a finite volime V.

Weigh functions describing the momentum and energy contribution of one particle to the carrier flow.

$$\vec{W_m} = \frac{f_1}{\tau_p}(v_f - u_p), \tag{2.27}$$

$$W_e = \frac{f_2}{\tau_p}(T_f - T_p) - \frac{m_p f_3}{\tau_p}(Y_s - Y_f). \tag{2.28}$$

CHAPTER 3

Numerical Model

The numerical mehrod used in this work is the high-order Eulerian-Langrangian (EL) method bases on the Weighted Essentially Non-Oscilatory (WENO) conservative finite difference scheme (WENO-Z) on a uniform mesh [5].

The particle and the carrier phase exhcange momentum and energy. The carrier phase velocity, temperature and mss at the particle position determines the influence of hte carrier phase on the particle. This velocity and temperature is determined by interpolation of carrier flow values at the cell centers, *i*, surrounding the particle position. The particle indluence is distributed on hte carrier phase by the beightink funcion **K**. The interpolation method and weighing function determine the accuracy and characteristics of the Eulerian-Lagrangian frame coupling.

Low-order interpolation are well known to lead to aliasing errors and instability in particle-mesh methods [11]. High-order interpolation and weighing reduces theser inaccuracies [11, 12]. We therefore seach for the high-order interpolation and weighing that is consistend with the high-order WENO-Z mehtod.

In what follows, we preset both the lower-order and higher-order interpolation, and weighing.

3.1 Low-order Interpolation

In smooth flow areas without shocks, the WENO-Z method uses a central difference scheme. Under these conditions, centered interpolation to the particle position is more accurate and therefore preferred. Lagrange interpolating polynomials of degree k,

$$P_k(x_p) = \sum_{i=i_p-k/2}^{i_p+k/2} Q(x_i) l_i(x_p) \tag{3.1}$$

where i_p represents the nearest cell center to the left of the particle position. The number of interpolating points k equals the number of points used in the k-th order WENO scheme. In the case of the fifth order WENO scheme, k = 5.In shocked regions the centered interpolation will produce undesirable Gibbs oscillations. With an ENO interpolation [18], these oscillations are essentially removed. In ENO interpolation, the interpolating points are determined based on smoothness of the function measured by the divided differences. The k-th degree divided differences are determined first.

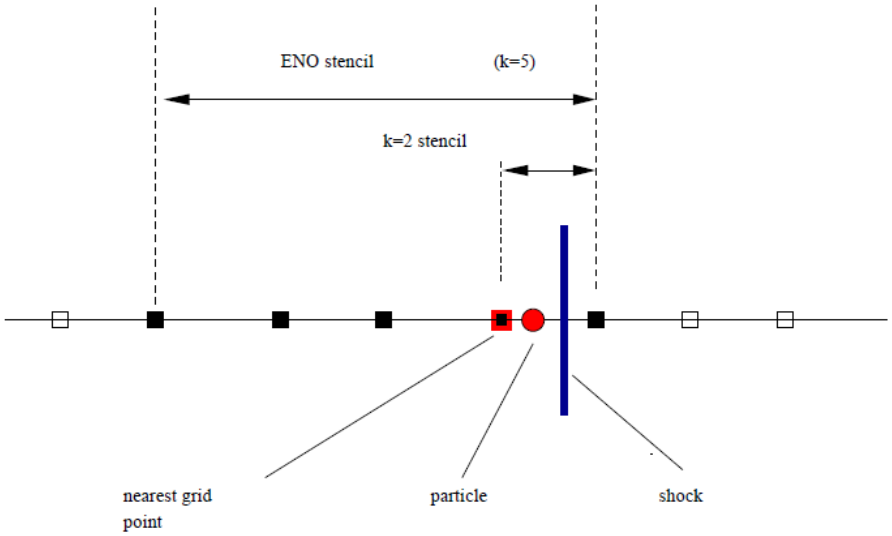


Figure 3.1: One dimensional ENO stencil for interpolation to a particle located near a shock. The interpolation stencil is determined based on the divided differences at the particles nearest grid point to the left of the particle.

The $0-$th order divided differences of q are defined by:

$$Q[x_i] = Q(x_i). \tag{3.2}$$

The $j-$th degree divided difference for $j \geqslant 1$ are defined by

$$Q[x_i, ..., x_{i+j}] = \frac{Q[x_{i+1}, ..., x_{i+j}] - Q[x_i, ..., x_{i+j-1}]}{x_{i+j} - x_i}. \tag{3.3}$$

Starting from a two point stencil, x_{ip}, x_{ip+1}, the interpolation stencil is expanded to k points based on a comparison of the divided differences of the increasing order at i_p (See Fig. 3.1). The smallest second order divided differences at i_p of the two potential three point stencils $minQ[x_{i_p-1}, x_{i_p}, x_{i_p+1}], Q[x_{i_p}, x_{i_p+1}, x_{i_p+2}]$ indicates the smoothest interpolation stencil and is therefore chosen. This procedure is repeated until a k point interpolation is found.

In the Figure 3.1 , we give an example of a typical ENO stencil close to a shock. We find the nearest grid point to the left of the particle. The magnitude of the first order divided differences at this grid point are larger than the divided differences to the left because of the shock jump. The stencil is therefore extended to the left. The same holds for the divided difference of a higher order than one. So, the ENO stencil will be preferential one-sided to the left of the particle if the particle is located in the cell including a shock. In two dimensions, the same procedure can be used along the separate dimension on the tensor grid. The divided differences are determined along horizontal and vertical lines in the grid. With the

one dimensional approach outlined above, we find the left most and bottom most grid point of the interpolation stencil with k k points for each grid point in the domain.

3.2 Weighing

Low-order particle-methods usually weigh with a zeroth or first order function for K(x, y) in (11) as shown in Figure 3.2.

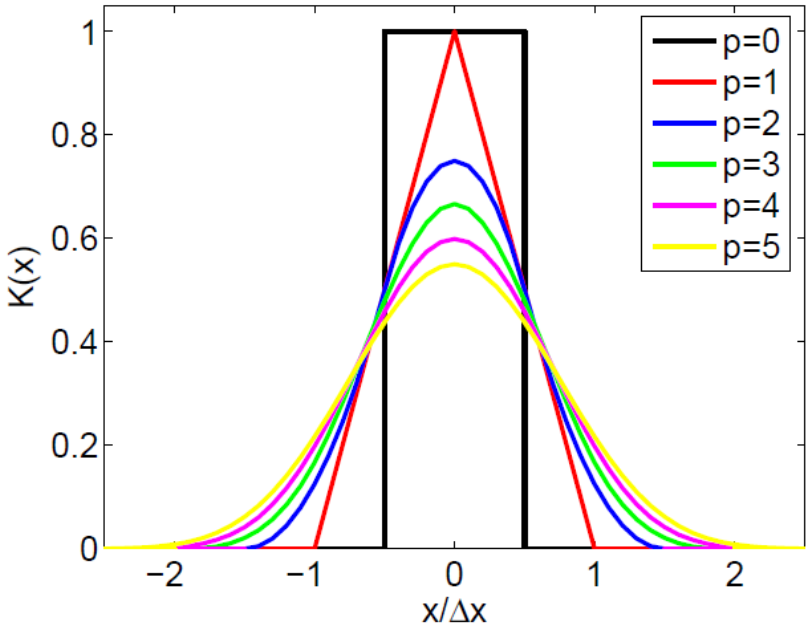


Figure 3.2: The $p-$th order weighing function.

The $0-$th order weighing function is a tophat function,

$$K(x,0) = \frac{1}{\Delta x}, 0x\frac{1}{2}\Delta x, \tag{3.4}$$

that weighs the particle influence to the nearest grid point. The first order weighing function,

$$K(x,0) = \frac{1}{\Delta x}\frac{\Delta x - x}{\Delta x}, \tag{3.5}$$

typically used, can be interpreted as a local area weighing. The particle splits a cell into two subcells. The area of the sub-cell to the left determines the particle influence to the grid point on the right of the particle and vice versa. In two dimensions the quadrilateral cell is

divided into four new subcells by the particle. The subareas of these are weighted to opposite grid points.

The lower order weighing functions are inconsistent with the higher-order method. The lack of smoothness of the particle shape results in Gibbs type phenomena that affect accuracy and introduce noise in the source term, S. The non-smooth shape is also enhancing instability.

3.3 Weighted Essentially Non-Oscillatory Schemes

In the numerical simulation of compresible flows modeled by means of hyperbolic conservation laws in the form:

$$\frac{\partial u}{\partial t} + \Delta F(u) = 0. \tag{3.6}$$

Without loss of generality, we will restrict our discussion to the one dimensional scalar case. Extensions to the system of equations and higher spatial dimensions present no extra complexity.

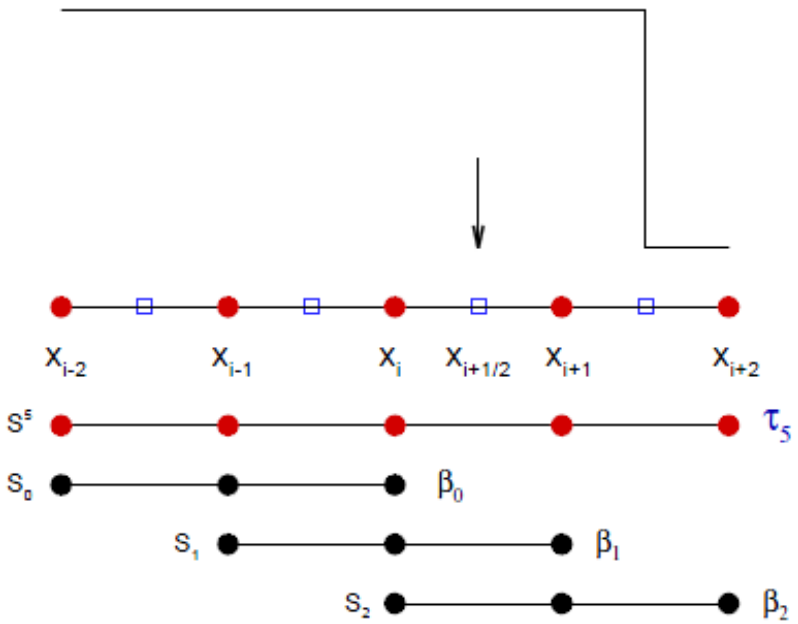


Figure 3.3: The computational uniform grid x_i and the five points stencil S^5, composed of three 3-points stencils S_0, S_1, S_2, used for the fifth-order WENO reconstruction step.

In this section we describe the fifth-order weighted essentially non-oscillatory conservative finite difference scheme WENO-Z [13] when applied to hyperbolic conservation

laws. The extension to a system of equations and to higher spatial dimensions is straightforward in the Cartesian coordinates.

Considering an uniform grid defined by the points $x_i = i\delta x, i = 0, ..., N$, which are also called centers, with cell boundaries given by $x_{i+\frac{1}{2}} = x_i + \frac{\Delta x}{2}$, where Δx is the uniform grid spacing. The semi-discretized form of Equation 3.6, by the method of lines, yields a system of ordinary differential equations:

$$\frac{du_i(t)}{dt} = \frac{\partial f}{\partial x}\bigg|_{x=x_i}, i = 0, ..., N, \tag{3.7}$$

where $u_i(t)$ is a numerical approximation to the point value u(x˙i, t).

A conservative finite-difference formulation for hyperbolic conservation laws requires high-order consistent numerical fluxes at the cell boundaries in order to form the fix differences across the uniformly-spaced cells. The conservative property of the spatial discretization is obtained by implicitly defining the numerical flux function $h(x)$ as

$$f(x) = \frac{1}{\Delta x}\int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} h(\xi)d\xi, \tag{3.8}$$

such that the spatial derivative in Eqn. 3.7 is exactly expressed by a conservative finite difference formula at the cell boundaries,

$$\frac{du_i(t)}{dt} = \frac{1}{\Delta x}\left(h_{i+\frac{1}{2}} - h_{i-\frac{1}{2}}\right) \tag{3.9}$$

where $h_{i\pm\frac{1}{2}} = h(x_{i\pm\frac{1}{2}})$.

All the cell faces, $h_{i\pm\frac{1}{2}}$ are interpolated from the known flux function $F(x)$ at the cell centers, $f_i = F(x_i)$. The fifth-order WENO-Z scheme uses a 5-points global stencil S^5, which is subdivided into three sub-stencil S_0, S_1, S_2, as shown in Figure 3.3. The fifth-order polynomial approximation $\hat{f}_{i\pm\frac{1}{2}} = h_{i\pm\frac{1}{2}} + O(\Delta x^5)$ is built through the convex combination of the interpolated values $\hat{f}^k(x_{i\pm\frac{1}{2}})$, in which $f^k(x)$ is the third degree polynomial below, defined in each one of the stencils S_k:

$$\hat{f}_{i\pm\frac{1}{2}} = \sum_{k=0}^{2} \omega_k^z \hat{f}^k(x_{i\pm\frac{1}{2}}), \tag{3.10}$$

where

$$\hat{f}^k(x_{i\pm\frac{1}{2}}) = \hat{f}^k_{i\pm\frac{1}{2}} = \sum_{j=0}^{2} c_{kj} f_{i-k-j}, i = 0, ..., N. \tag{3.11}$$

The $c_{kj}$are Lagrangian interpolation coefficients, qith depend on the left-shift parameter $k = 0, 1, 2$, but not on the values f_i.

The WENO-Z weights ω_k^z are defined as

$$\omega_k^z = \frac{\alpha_k^z}{\sum_{l=0}^{2} \alpha_l^z}, \alpha_k^z = \frac{d_k}{B_k^z}, k = 0, 1, 2. \tag{3.12}$$

The coefficients $d_0 = \frac{3}{10}, d_1 = \frac{3}{5}, d_2 = \frac{1}{10}$ are called the ideal weights since they generate the central upstream fifth-order scheme for the 5-points stencil S^5. The weights ω_k^z are a function of the smoothness indicators β_k^z, namely,

$$\beta_k^z = \left(1 + \left(\frac{\tau_5}{\beta_k + \epsilon}\right)\right), \tag{3.13}$$

where ϵ is a small number (typically $\epsilon = 10^{-12}$) used to avoid the division by zero in the denominator and the classical smoothness indicators β_k measure the regularity of the kth polynomial approximation $hatf^k(x_i)$ at the stencil S_k and are given by

$$\beta_k = \sum_{l=1}^{2} \Delta x^{2l-1} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(\frac{d^l}{dx^l} \hat{f}^k(x)\right)^2 dx. \tag{3.14}$$

The expresion of the β_k in terms of the cell averaged values of $f(x)$, f_i are given by

$$\beta_0 = \frac{13}{12}(f_{i-2} - 2f_{i-1} + f_i)^2 + \frac{1}{4}(f_{i-2} - 4f_{i-1} + 3f_i)^2, \tag{3.15}$$

$$\beta_1 = \frac{13}{12}(f_{i-1} - 2f_i + f_{i+1})^2 + \frac{1}{4}(f_{i-1} - f_i)^2, \tag{3.16}$$

$$\beta_2 = \frac{13}{12}(f_i - 2f_{i+1} + f_{i+2})^2 + \frac{1}{4}(3f_i - 4f_{i+1} + 3f_{i+2})^2, \tag{3.17}$$

and

$$\tau_5 = \left|\beta_0 - \beta_2\right|, \tag{3.18}$$

being all β_k^z smaller than unity and they are all close to 1 at smooth parts of the solution.

The idea of the weighths definiton is that on smooth parts of the solution the smoothness indicators β_k are all small and about the same size, generating weights ω_k that are good aproximations to the ideal weights d_k.

CHAPTER 4

Evaporation Model Verification

4.1 D^2 Law Verification

4.1.1 Analitycal Method

For the analytical verification, we can reduce the number of unknowns taken these assumptions [14].

1. The evaporation process is quasi-steady.
2. The droplet temperature is uniform and, furthermore, the temperature is assumed to be some fixed value below the boiling point of the liquid.
3. The mass fraction of vapor at the droplet surface is determined by liquid-vapor equilibrium at the droplet evaporation problem.
4. We also assume that all thermophysical properties -specifically, the ρD product- are constant.

We can find the mass evaporation rate, $\dot{m}$, and the droplet radius history, $r_s(t)$, by writing a droplet vapor species conservation equation.

$$\dot{m} = 4\pi r_s \rho \mathcal{D}_{AB}(Y_s - Y_f) \,, \tag{4.1}$$

Starting from the mass balance we obtain the droplet diameter evolution,

$$\frac{dm_d}{dt} = -m, \tag{4.2}$$

where the droplet mass, m_d, is given by

$$m_d = \rho_l V = \rho_l \pi D^3/6 \tag{4.3}$$

and V and D are the droplet volume and diameter, respectively.

Substituting Eqn. 4.1 and Eqn. 4.3 into 4.2 leads,

$$m = 4\pi r_s \rho_l D_{AB}(Y_s - Y_f), \tag{4.4}$$

$$\frac{dD}{dt} = -\frac{4\rho D_{AB}}{\rho_l D}(Y_s - Y_f). \tag{4.5}$$

In the combustion literature, however, Eqn.4.5 is commonly expressed in terms of D^2 rather than D. This forms is:

$$\frac{dD^2}{dt} = -\frac{8\rho D_{AB}}{\rho_l D}(Y_s - Y_f), \tag{4.6}$$

Eqn.4.6 tells us that the time derivative of the square of the droplet diameter is constant; hence, D^2 varies linearly with t with the slope $-(8\rho D_{AB}/(\rho_l D)(Y_s - Y_f)$. This slope is defined to be the **evaporation constant K:**

$$K = -\frac{8\rho D_{AB}}{\rho_l D}(Y_s - Y_f). \tag{4.7}$$

We can use Eqn. 4.6 to find the time it takes a droplet of given initial size to evaporate completely; i.e., the droplet lifetime, t_d. Thus,

$$\int_{D_o^2}^{0} dD^2 = -\int_0^{t_d} K dt, \tag{4.8}$$

wich yields

$$t_d = D_o^2/K. \tag{4.9}$$

We change upper limits of the integral to provide a general relationship expressing the variation of D whti time (t):

$$D^2(t) = D_o^2 - K_t, \tag{4.10}$$

Equation (4.10) is referred to as the D^2 Law for droplet evaporation.

4.1.2 Numerical Method

To use Eqn.2.8 in the code, m_p is written in terms of τ_p. In order to verify the D^2 evaporation law, m_p has to be write in terms of d_p.

The expresion of τ_p is,

$$\tau_p = \frac{Re_o \rho_p d_p^2}{18}. \tag{4.11}$$

Substituting Eqn.4.11 into Eqn.2.8 and rearranging,

$$\frac{dm_p}{dt} = -\frac{\pi \rho_f d_p Sh}{Re_o Sc}(Y_s - Y_f). \tag{4.12}$$

Finally, substituting $m_p = \frac{4}{3}\pi \left(\frac{d_p}{3}\right)^2 \rho_p$ and taking the simplification of low Reynolds number, Sherwood number (2.11) can be considered equal to 2. The analog expression of Eqn.4.6 is obtained,

$$\frac{dd_p^2}{dt} = -\frac{8\frac{\rho_f}{\rho_p}}{Re_o Sc}(Y_s - Y_f). \tag{4.13}$$

Eqn.4.13 and Eqn.4.6 are equivalent and the results of both methods can be compared directly.

4.1.3 Problem Setup

In order to verify the numerical method implemented in the code and the analytical form, we consider single $n-$dodecane droplet evaporating in dry nitrogen at 1 atm if the droplet temperature is 10 K below dodecane boiling point in a quiescent environment.
For simplicity, we assume that the mean gas density is that of nitrogen at a mean temperature of 800 K. The density of liquid dodecane is 749 kg/m^3.

Name	Simbol	Value
Density	ρ_{part}	$730 kg/m^3$
Temperature	T_{part}	$437.7K$
Evaporation Temperature	T_{boil}	$447.7K$
Particle Velocity	U_{part}	$0m/s$
Molar weight	MWP	$142g/mol$

Table 4.1: N-dodecane Droplet values

4.1.4 Normalization

Name	Simbol	Value
Density	ρ_{N_2}	$0.4267kg/m^3$
Temperature	T_{rigth}	$800K$
Velocity	U_{Right}	$0m/s$
Molar weight	MWG	$28.97g/mol$

Table 4.2: Nitrogen (Gas-phase) - Initial conditions

For numerical solutions the dimensional form is not convenient get the equivalent solution between both analytical and numerical methods according to the 2.1.2. All variables are non-dimensionalized by references values [15] .

Name	Simbol	Value
Mach number	M_{ref}	$1ud/ud$
Velocity	U_{ref}	$29.30m/s$
Tunnel width	L_{ref}	$5.30x10^{-02}m$
Density	ρ_{ref}	$1.29kg/m^3$
Specific heat ratio	γ	1.4
Temperature	T_{ref}	$10K$
Viscosity at $T = 300K$	μ_{ref}	0.187
Reynolds number	Re_o	$1.3800x10^5$
Particle response time	τ_o	$2.1413x10^5$

Table 4.3: Normalized values.

4.1.5 Properties Calculations

This models takes the assumption that gas and vapor properties are constant in space, and therefore independent of the temperature [16]. Thus, for a physical consistency, these models must be bases on characteristic average constant property values that accurately account for the real spatial and thermal property variations.

These properties are generally used at each numerical time step and can add significant computational expense when many droplets are involved. For this study we chose to calculate each step, computational time does not increase in a significant time.

At the beginning, estimated wel bulb temperature is calculated [17],

$$T_{WB} = 137 \left(\frac{T_B T_{ref}}{373.15} \right)^{0.68} log_{10}(T_{fluid} T_{ref}) - 45, \tag{4.14}$$

it is essentially the steady state surface temperature achieved during evaporation. This approach assumes that droplet surface temperature is quickly raised from initial condition to the wet bulb value and that this surface temperature is the appropriate condition for evaluating both the vapor and carrier gas properties.

The temperature dependent properties for the carrier gas and vapor species are required for the reference condition methods described by Harpole [18],

$$\begin{cases} T_{WB} > 600\ , Pr = 0.647 + 5.510^{-5} T_{WB} \\ T_{WB} \leq 600\ , Pr = 0.815 - 4.95810^{-4} T_{WB} + 4.51410^{-7} T_{WB}^2, \end{cases} \tag{4.15}$$

where Pr is Prandtl numer and $Sc.$ is Schmidt number. In this specie neither the diffusivity nor the Schmidt number are provided we take the assumption $Pr = Sc.$

Heat capacities of particle are calculated using these aproximations [19],

$$L_v = 1.04(3.95810^4(619 - T - WB2) -^0 .38)J/kg \tag{4.16}$$

$$C_{p,l} = 2.5210^7 J/(kgK), \tag{4.17}$$

$$\begin{cases} T_{WB2} > 0.8\ , C_{p,v} = 0.982 + 1.304 T_{WB2} - 0.593 T_{WB2}^2 + 0.101 T_{WB2}^3 J/(KgK) \\ T_{WB2} \leq 0.8\ , C_{p,v} = 0.2547 + 1.377 T_{WB2} - 0.4 T_{WB2}^2 + 0.113 T_{WB2}^3 J/(KgK), \end{cases} \tag{4.18}$$

where $T_{WB2} = T_{WB}/1000.$

4.1.6 Results

Considering the problem setup, droplet and gas properties are defined in Table.4.1 and 4.2 respectively. Solving the problem for both methods the $D^2 Law$ is validated.

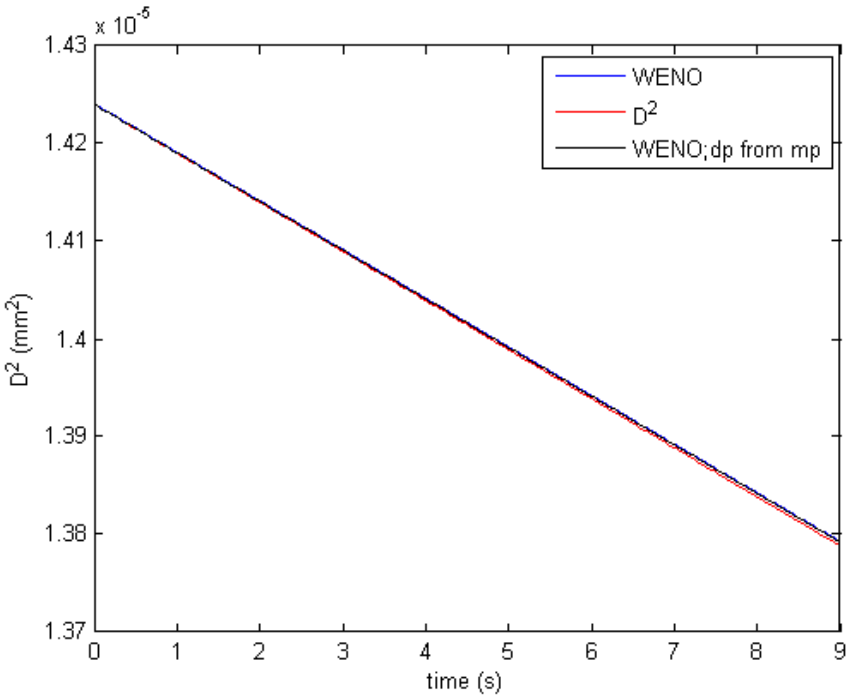


Figure 4.1: Analytical (D^2) and numerical (WENO) droplet diameter evolution.

In the Figure 4.1 the evolution of the particle diameter is plotted for a initial diameter of 200 μm. It can be easily seen that these solutions do not differ much. The numerical results slightly overestimate the droplet lifetime, the result is not fully accurate because the numerical calculation error.

4.2 Particle Physics Verification

The particle tracking physics have been validated by comparing the WENO solution and the analytical solution of the non-dimensional Lagrangian equations for the particle position (Eqn. 2.5) and velocity (Eqn. 2.6).

For the analytical solution particle position and particle velocity have been derived using the same time step that the numerical program.

The program is run in the same conditions that for the D^2 Law Verification, except for the particle velocity, v_p, being the initial value of this parameter equal to 1 m/s.

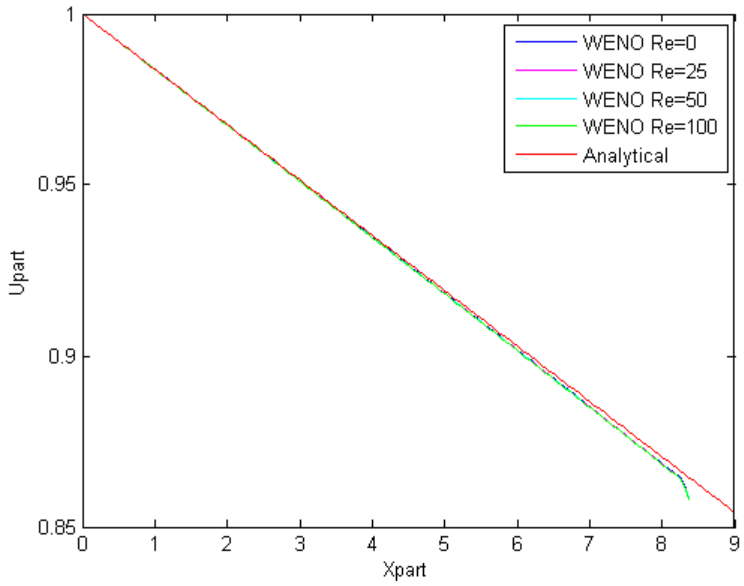


Figure 4.2: Graphic representation of particle velocity vs. position.

Analytical solution match with the WENO simulation as we can see in Figure 4.2. It is important to check if the particle velocity is affected by the Reynolds number. At low Reynolds numbers, where viscous forces are dominant, particle velocity decreases when Reynolds number increase. In order to verify this evolution, particle velocity is plotted for low Reynolds numbers in Figure 4.3.

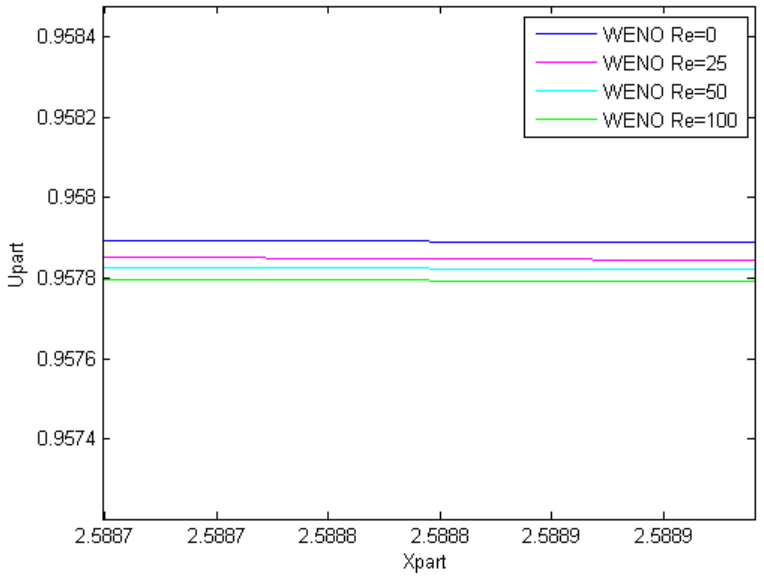


Figure 4.3: Influence of Reynolds Number in WENO solution.

CHAPTER 5

Conclusions

We have implemented an evaporation model in a high-order Eulerian-Lagrangian solver and verified the results in with D^2-law and analytical calculations of the physics. We consider one dimensional shock interacting with a cloud of particles, that case has been studied and verified by Jacobs and Don [5] based on the work of Boiko *et al.* [15].

To demonstrate the validation of the WENO-Z code with the evaporation model included, we have compared the original WENO-Z scheme in the one dimensional solution presented by Jacobs and Don with the new WENO-Z with the evaporation simulation deactivated. In particular, the Mach number used is $M_s = 2.8$ with the shock at $x_s = 0$. A reflective boundary conditions is imposed on both ends of the shock tube. A cloud of particles with a volume concentration of 4% is initialize in the interval [0, 0.2981]. The particle response time is $\tau_p = 3.929610^3$ and the density is $\rho_p = 1200$. We take the Reynolds number needed to compute the particle traces according to the documentation at $Re_{ref} = 1.7630x10^6$.

These results are presented in Figure 5.1 and shown that the new scheme obtain a pressure profile slightly higher in the shock wave interaction, this solution is valid and the difference is caused by the new procedure to obtain parameters as Prandtl number or response time.

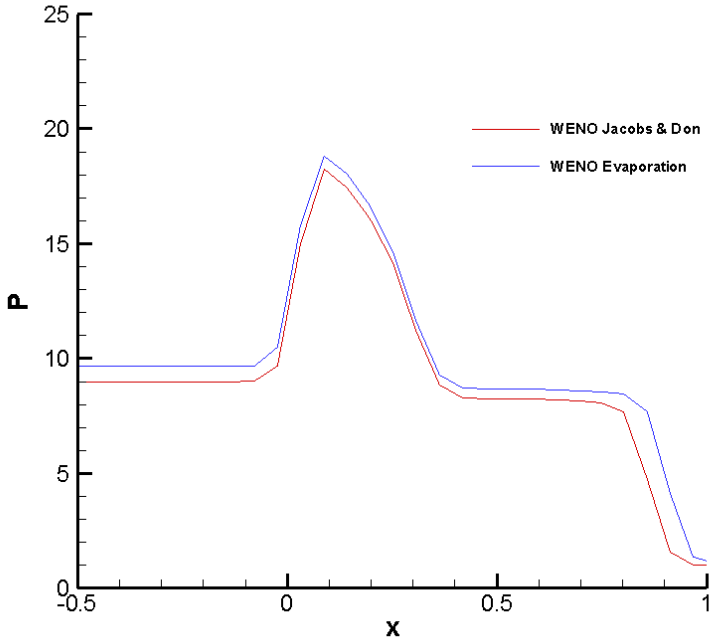


Figure 5.1: Pressure profile at time $t_{=}0.275$ for the original WENO-Z method and the new WENO-Z with the evaporation model deactivated.

For the simulation discussed next, in order to compare the behavior of the shock wave with the could of particles for both the evaporation subroutine included and deactivated, a sock tube problem is setup inte domain $x \in [-5, 6]$ with length, $L = 11$. The state of the sock flow is

$$[\rho_R, u_R, p_R] = [1, 0, 1]. \tag{5.1}$$

The post-shock state can be computed via the well-known Rankine-Hygoniot relations for given Mach number M_s. Here, we use $M_s = 2.8$ with the shock at $x_s = 0$. A reflective boundary conditions is imposed on both ends of the shock tube.

A cloud of particles with a volume concentration of 4% is inialitez in the interval [0, 0.2981]. The particle response time is $\tau_p = 3.929610^3$ and the density is $\rho_p = 730kg/m^3$, corresponding to the values from d^2-law verification. We take the Reynolds number needed to compute the particle traces according to the documentation at $Re_{ref} = 1.7630x10^6$. The droplet size is setup considering a diameter, $d_p = 200\mu m$ and a initial temperaturel, $T_p = 437.7K$.

A left moving shock is generated when the shock hits the cloud of particles shortly after the time $t > 0$. Moreover, an expansion fan is formed at the rear end of the cloud after the shock has passed the cloud of particles.

From Figures [5.2, 5.3 and 5.4] we can extract that the final state of the gas behind the could of particles is determined by irreversible losses of the gas flow in the sock wave and in the cloud of particles due to friction and heat exchange with particles. In the case of evaporation this loses are lower than the convectional simulation, the increase of the pressure is produced by the contribution of the evaporation droplet. The evaporation droplet contributes energy to the continuum phase and the increase of pressure is directly related to the increase of velocity after the sock wave.

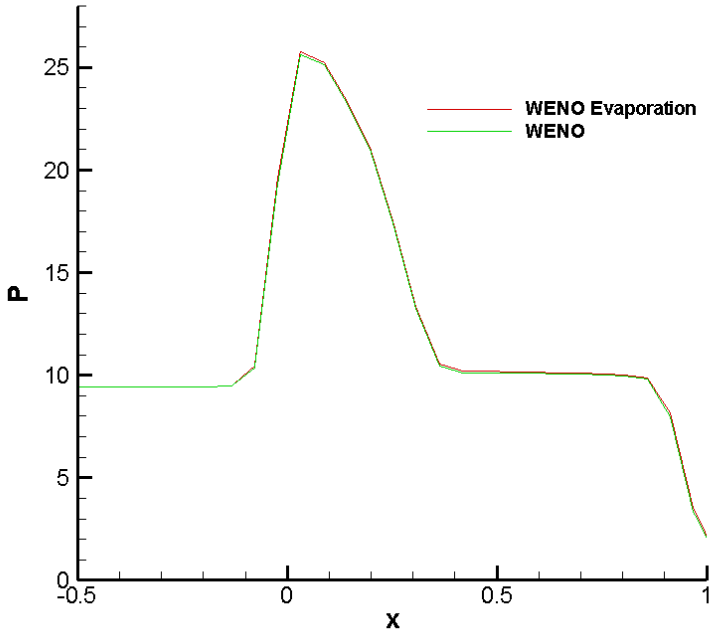
All the initial parameters of the particle and gas phase are summarized in Tables 5.1 and 5.2.

Name	Simbol	Value
Particle Density	ρ_{part}	$730kg/m^3$
Particle Response Time	τ_p	3.929610^3
Reynolds Number	Re	1.763010^6
Temperature	T_{part}	$437.7K$
Droplet size	d_p	$200\mu m$

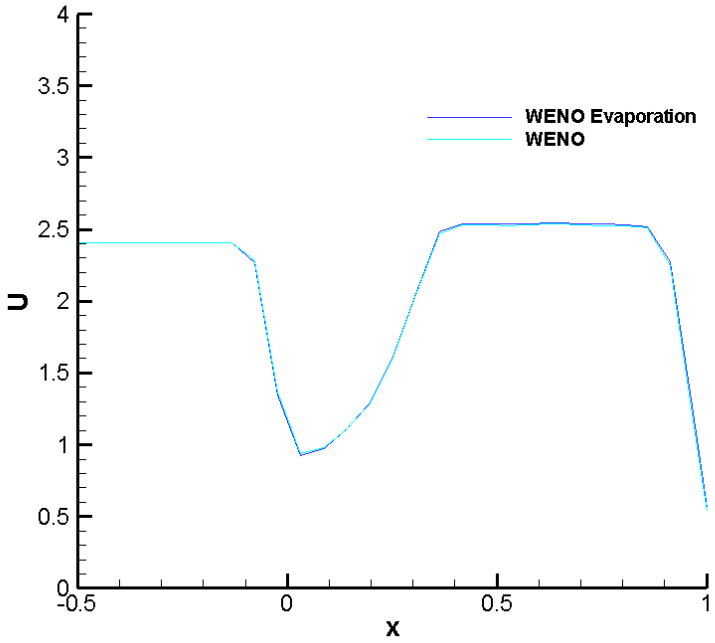
Table 5.1: Initial particle parameters for the simulation of evaporation model

Name	Simbol	Value
Mach Number	M_s	2.8
Density	ρ_f	$1291kg/m^3$
Shock Initialization	x_s	0
Shock tube domain	x	$[-5, 6]$

Table 5.2: Initial Fluid parameters and Shock Wave for the evaporation model simulation

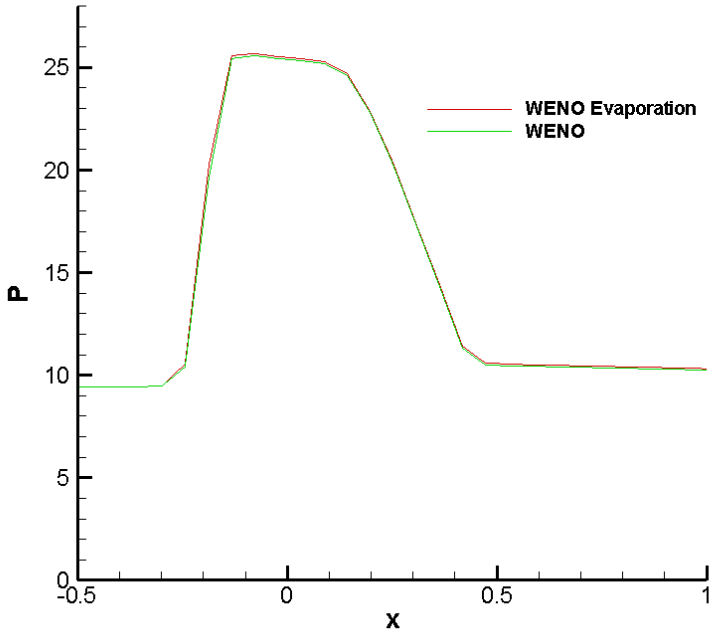


(a) Pressure at t_1=0.275

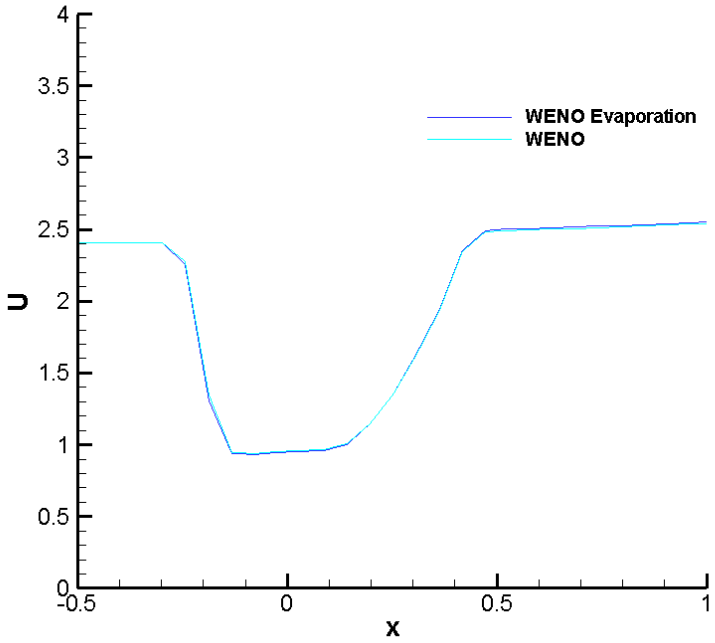


(b) Velocity at t_1=0.275

Figure 5.2: Pressure profile (a) and Velocity profile (b) at time $t_1 = 0.275$ for WENO-Z mehtod and WENO-Z method including evaproation model.

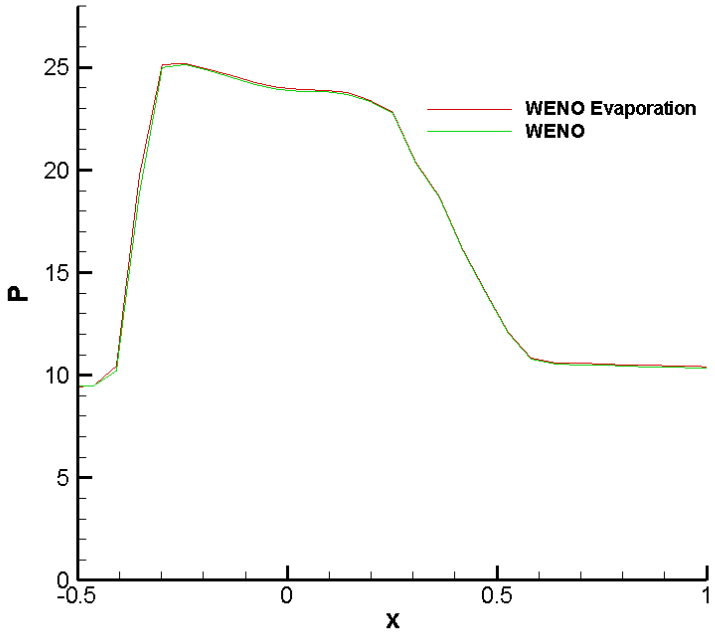


(a) Pressure at t_2=0.55

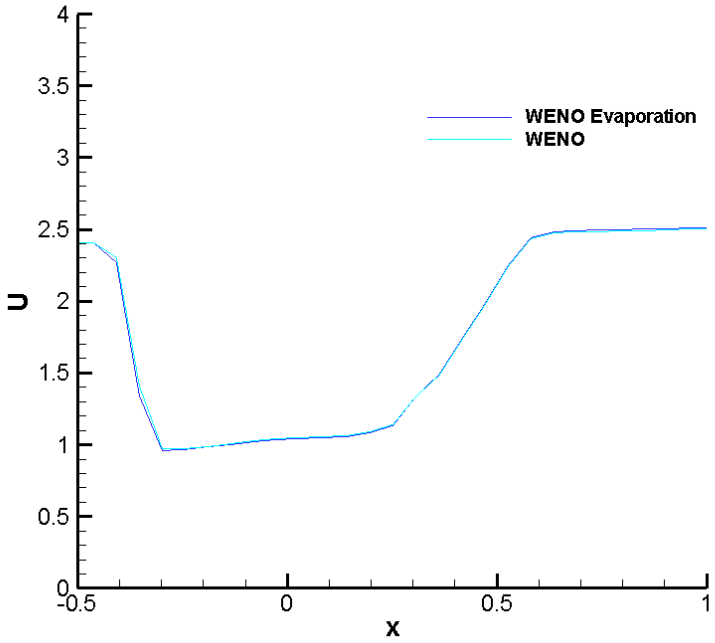


(b) Velocity at t_2=0.55

Figure 5.3: Pressure profile (a) and Velocity profile (b) at time $t_2 = 0.55$ for WENO-Z mehtod and WENO-Z method including evaproation model.



(a) Pressure at t_3=0.825



(b) Velocity at t_3=0.825

Figure 5.4: Pressure profile (a) and Velocity profile (b) at time $t_3 = 0.825$ for WENO-Z mehtod and WENO-Z method including evaproation model.

CHAPTER 6

Future Work

In future work, capabilities of the numerical numerical model will be extended to include species conservation law in source

- Include species conservation law in source term.
- The simulation with different particle elements.
- Simulate the behavior with different shock types.
- Include combustion and chemical reaction.

BIBLIOGRAPHY

[1] S. Osher A. Harten, B. Engquist and Chakravarthy. Uniformly high order essentially non-oscillatory. *J. Comp. Phys*, 71:231–303, 1987.

[2] C.W. Shu and S. Osher. Efficient impletation of essentially non-oscillatory sock capturing schemes. *J. Comp. Phys*, 83:32–78, 1989.

[3] G.S. Jiang and C.W. Shu. Efficient implementation of weighted eno schemes. *J. Comp. Phys*, 126:202–228, 2000.

[4] D. Balsara and C.W. Shu. Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy. *J. Comp. Phys*, 160:405–452, 2000.

[5] G.B. Jacobs and W.S. Don. A high orden weno-z finite differences based particle-source-in-cell method for computation of particle-laden flows with shocks. *J. Comp. Physics*, 2008.

[6] K. Harstad nad J. Bellan R.S. Miller. Evaluation of equilibrium and non-equilibrium evaporation models for many-droplet gas-liquid flow simulations. *International Jouernal of Multiphase Flow*, 24:1025–1055, 1998.

[7] F. Mashayek. Droplet-turbulence interaction in low mach number homogeneous shear two-phase flows. *J. Fluid Mech*, 376:163–203, 1998.

[8] D.B. Spalding. The combustion of liquid fuels. in: Proceedings of the fourth symposium (international) on combustion. *Combustion Institute, Baltimore*, MD:847–864, 1953.

[9] M. P. Sharma C. T. Crowe and D. E. Stock. *Multiphase Flows with Droplets and Particles*. Press LLC, Boca Raton, FL, 1998.

[10] J. Meijerink and B. Jacobs. A weno-z based eulerian-lagrangian code for simulation of shocked flows laden with evaporating droplets. *American Institute of Aeronautics and Astronautics*, 2010.

[11] C.K. Birdsall and A.B. Langdon. *Plasma physics via computer simulation*. McGraw-Hill, Inc., 1985.

[12] S. Natsuhiko H. Abe and R. Itatani. High-order spline interpolations in the particle simulation. *J. Comp. Phys*, 63:247–267, 1997.

[13] B. Costa R. Borges, M. Carmona and W.S. Don. An improved weighted essentially non-oscilatory scheme for huperbolic conservation laws. *J. Comp. Phys.*, 227(6):3101–3211, 2009.

[14] Stephen R. Turns. *An Indtrocution to Combustion: Concepts and applications*. McGraw-Hill ., 2000.

[15] V.M. Boiko et al. Interaction of a shock wave with a cloud of particles. *Combustion, explosion, and Shock Waves*, 32(2):191–203, 1996.

[16] C.K. Law and H.K. Law. Quasi-steady diffusion flame theory with varaible specific heat and transportation coefficients. *Comb. Sci. Tech.*, 12:207–216, 1976.

[17] M.C. Yuen and L.W. Chen. On drag of evaporating liquid droplet. *Comb. Sci. Tech.*, 14:147–154, 1976.

[18] G.M. Harpole. Droplet evaporation in high temperature environments. *J. Heat Transfer*, 103:86–91, 1981.

[19] B. Abramzon and W.A. Sirignan. Droplet vaporization model for spray combystion calculations. *J. Heat Transfer*, 32(9):1605–1457, 1989.

APPENDIX

A - Main modifications in scrip of the FORTAN code

A - Main modifications in scrip of the FORTAN code

Main modifications in the code.

A.0.1 Particle Stuff

```
integer             :: N_Frame        = 10
integer             :: Shock_Type     = 0
character(LEN=1)    :: Variable_Type  = 'C'
integer             :: Shock_Profile  = 0
REALTYPE            :: Gamma          = 1.40d0
REALTYPE            :: Mach           = 3.0d0
REALTYPE            :: P_Ratio        = 1.0d0
REALTYPE            :: Wave_Number    = 1.0d0
REALTYPE            :: Amplitude      = 0.2d0
REALTYPE            :: Begin_Time     = ZERO
REALTYPE            :: Final_Time     = 1.0d 4
integer             :: Step           = 0
REALTYPE            :: Time           = ZERO
integer             :: Restart        = 0
integer             :: Central_Order  = 6
integer             :: Filter_Choice  = 1
integer             :: Filter_Order   = 6
REALTYPE            :: Filter_Viscosity = 0.2
integer             :: IEVAP = 0 ! Evaporation 1=ON, 0=OFF


REALTYPE, dimension(3)        :: Q_Left, Q_Right
REALTYPE, ALLOCATABLE         :: Source1(:,:)
REALTYPE, ALLOCATABLE         :: Stats_P(:,:)
INTEGER                       :: Nstats = 11

integer  :: i, j

REALTYPE :: Shock_Location, Shock_Speed, Shock_Position

REALTYPE :: dt, dt_Original
REALTYPE :: CPU_Begin, CPU_End, CPU_Start

logical  :: Save_Indicator = .TRUE.
REALTYPE :: Movie_Resolution = 0.10d0
REALTYPE :: Next_Save_Time = 0.0d0

REALTYPE ::      x_Left,    x_Right
REALTYPE ::    Rho_Left,  Rho_Right
REALTYPE ::      U_Left,    U_Right
REALTYPE ::      P_Left,    P_Right
```

```
REALTYPE ::      T_Left,    T_Right
REALTYPE ::      C_Left,    C_Right

REALTYPE :: MW_Gas_1=ZERO, Rho_Gas_1=ZERO, T_Pre_Shock=ZERO

!
! Particle Stuff
!
!INTEGER, PARAMETER                    :: Npart = 0
INTEGER, PARAMETER                    :: Npart = 86000
REALTYPE, dimension(Npart)            :: Xpart, Upart, Tpart, f1, f2
REALTYPE, dimension(Npart)            :: f3, f4, deltaY, A1, A2
REALTYPE, dimension(Npart)            :: Ufluid, Tfluid, Pfluid, dpdxfluid
REALTYPE, dimension(Npart)            :: Rhofluid, Mpart, P_B
REALTYPE, dimension(Npart)            :: PD, Taup, T_B, T_WB, Sc, Pr
INTEGER                               :: np, update_particle
REALTYPE                              :: Source_factor
REALTYPE                              :: Stokes, dxpart
REALTYPE                              :: TAUP0, RHOPref, re, PD0, PM0
REALTYPE                              :: CRE,CF2, pi, Rhofp, RHOP
REALTYPE                              :: Ufluct
REALTYPE                              :: Mref, Tref, Uref, R, MWG, MWP, T_particle

!
! Input Data
!
call Input

call PS_Shock_Specification (Shock_Type, Variable_Type,                  &
                             Mach, P_Ratio, MW_Gas_1, Gamma, Shock_Speed, &
                             Rho_Left , U_Left , P_Left , T_Left ,       &
                             Rho_Gas_1, U_Right, P_Right, T_Pre_Shock,   &
                             Final_Time, x_Left, x_Right, Shock_Location, &
                             C_Left, C_Right, Q_Left, Q_Right)


!
! Setup Operators
!

call Domain_Setup
call Index_Setup
call Allocate_Variables
call Operator_Setup

ALLOCATE(Source1(N0:N5,NV))
ALLOCATE(Stats_P(N0:N5,Nstats))
Source1 = 0.0d0
Stats_P = 0.0d0

!
! Setup Initial Shock Profile
!
```

```
call Initial_Condition

!
! Setup Initial Particle Conditions
!
       Source_factor = 86000.0d0/DBLE(Npart)
       !Source_factor = 3.18310d5/DBLE(Npart)
       !Source_factor = 3.1831d7/DBLE(Npart)

      pi      = 3.14159265358979 32d0
      TAUP0   = 3.9296e03                          ! particle response time
      !TAUP0   = 1.7845d03                         ! particle response time
      !TAUP0   = 2.1413d3

      RHOPref = 1290.0d0                          ! Ref non dimensional particle density
      !RHOP     = 7.41670d3                        ! non dimensional particle density
      RHOP     =  730.0d3/RHOPref                  ! Decane   for D2 Laq comprobatio

      re      =  1.7638e06                        ! fluid Reynolds number
      !re      =  9.6503d05
      !re      = 1.7638d6

      PD0     = dsqrt(18.0d0*TAUP0/(re*RHOP))      ! particle diameter
      PM0     = pi*RHOP*PD0**3 /6.d0               ! particle mass

      Rhofp   = 1.0d0 !0.4232d0/RHOPref
      CRE     = re*Rhofp*PD0



!! Reference Values and values for Evaporation model
      Mref        = 1.0d0           ! Refence Mach number
      Tref        = 10.0d0          ! K
      T_particle  = 437.7           ! K
      R           = 8.314410d7      ! erg/K
      MWG         = 28.970d0        ! g/mol, Molar weight, Air
      MWP         = 142.0d0         !
      Uref   = Mref*SQRT(Gamma*R/MWG*Tref)

       dxpart = (1.55d 2/52d 3)/DBLE(Npart 1)
      DO np=1,Npart
        Xpart(np) = ((0.0d 2)/52.0d 3) + DBLE(np 1)*dxpart
      ENDDO



     call Interpolate_Fluid_To_Particle_ENO(Xpart, Upart, Tpart, Npart, Q,  &
                                            Ufluid, Tfluid, f1, f2,dpdxfluid, &
                                            T_B, T_WB, Sc, Pr)

      DO np=1,Npart
        !Upart(np) = Ufluid(np)*2.0
        CALL RANDOM_NUMBER (Ufluct )
```

```
      Ufluct = 0.5d0
      Upart(np) = Ufluid(np) + ((Ufluct 0.5d0)*2.0d0)*0.2d0
      !Tpart(np) = Tfluid(np)
      Tpart(np) = T_particle/Tref
      Mpart(np) = PM0 !1.5922d 5  !1.9902d 6  !1.5922d 5  !1.2d 4
      PD(np)    = PD0 !(6.0d0*Mpart(np)/(PI*RHOP))**(ONE/THREE)
    ENDDO
```

A.0.2 Runge Kutta

```
Subroutine Runge_Kutta(Xpart, Upart, Npart, Ufluid, Tfluid,dpdxfluid)

  implicit none

  integer   :: Runge_Kutta_Stage
  integer   :: i,j
  REALTYPE  :: Time_n, Time_Now

  REALTYPE, dimension(N0:N5,NV) :: Q1, D_Flux, Source
  !REALTYPE, dimension(N0:N5,NV) :: Q1, D_Flux

  INTEGER                       :: Npart
  REALTYPE, dimension(Npart)    :: Xpart,Upart, Ufluid, Tfluid
  REALTYPE, dimension(Npart)    :: f1,f2,dpdxfluid, f3, f4, deltaY
  REALTYPE, dimension(Npart)    :: Xpart1, Upart1, Tpart1, Mpart1

  Time_n = Time dt

    Q1 = Q ; D_Flux = ZERO

 ! Source = ZERO
! Stage 1 :

  Runge_Kutta_Stage = 1 ; Time_Now = Time_n + dt

     call Interpolate_Fluid_To_Particle_ENO(Xpart, Upart, Tpart, Npart, Q,  &
                                            Ufluid, Tfluid, f1, f2,dpdxfluid,  &
                                            T_B, T_WB, Sc, Pr)


     call Particle_Evaporation (Xpart, Upart, Tpart, Npart, Q, Ufluid,     &
                                Tfluid, f3, f4, dpdxfluid, deltaY, A1, A2, &
                                PD, Taup, Mpart)


     call Weigh_Particle_To_Grid(Xpart, Upart, Tpart, Npart, Ufluid, Tfluid, &
                                 f1, f2, Source, f3, f4, deltaY, Mpart)

     call Fluxes          (Q , D_Flux)


    DO np=1,Npart
```

```
      Xpart1(np) = Xpart(np) + dt*Upart(np)
      Upart1(np) = Upart(np) + dt*(f1(np)*(Ufluid(np) Upart(np))/Taup(np))
      Tpart1(np) = Tpart(np) + dt*(f2(np)*(Tfluid(np) Tpart(np)) &
                   f3(np)*deltaY(np))/Taup(np)
      Mpart1(np) = Mpart(np)    dt*f4(np)*deltaY(np)*sqrt(Taup(np))
    ENDDO
    if (update_particle == 0) Source = 0.0
    Q1     = Q     + dt*(D_Flux + Source)

  call Boundary_Condition (Q1)

! Stage 2 :

  Runge_Kutta_Stage = 2 ; Time_Now = Time_n + dt/2


    call Interpolate_Fluid_To_Particle_ENO(Xpart1, Upart1, Tpart1,  Npart, Q, &
                                           Ufluid, Tfluid, f1, f2,dpdxfluid,  &
                                           T_B, T_WB, Sc, Pr)

     call Particle_Evaporation (Xpart1, Upart1, Tpart1, Npart, Q1, Ufluid, &
                                Tfluid, f3, f4, dpdxfluid, deltaY, A1, A2, &
                                PD, Taup, Mpart1)

     call Weigh_Particle_To_Grid(Xpart1, Upart1, Tpart1,  Npart, Ufluid, Tfluid, &
                                 f1, f2, Source, f3, f4, deltaY, Mpart)

    call Fluxes              (Q1, D_Flux)

    DO np=1,Npart
      Xpart1(np) = (THREE*Xpart(np) + Xpart1(np) + dt*Upart1(np))/FOUR
      Upart1(np) = (THREE*Upart(np) + Upart1(np) + dt*(f1(np)*(Ufluid(np)  &
                                Upart1(np))/Taup(np)     dpdxfluid(np)/RHOPref) )/FOUR
      Tpart1(np) = (THREE*Tpart(np) + Tpart1(np) + dt*(f2(np)*(Tfluid(np)   &
                                Tpart1(np)) f3(np)*deltaY(np))/Taup(np))/FOUR
      Mpart1(np) = (THREE*Mpart(np) + Mpart1(np)   dt*f4(np)*deltaY(np)* &
                                sqrt(Taup(np)))/FOUR
    ENDDO
    if (update_particle == 0) Source = 0.0d0
    Q1     = (THREE*Q + Q1 + dt*(D_Flux + Source))/FOUR

  call Boundary_Condition (Q1)

! Stage 3:

  Runge_Kutta_Stage = 3 ; Time_Now = Time_n + dt


    call Interpolate_Fluid_To_Particle_ENO(Xpart1, Upart1, Tpart1,  Npart, Q, &
                                           Ufluid, Tfluid, f1, f2,dpdxfluid,  &
                                           T_B, T_WB, Sc, Pr)

     call Particle_Evaporation (Xpart1, Upart1, Tpart1, Npart, Q1, Ufluid, &
                                Tfluid, f3, f4, dpdxfluid, deltaY, A1, A2, &
```

```
                                    PD, Taup, Mpart1)

      call Weigh_Particle_To_Grid(Xpart1, Upart1, Tpart1,  Npart, Ufluid, Tfluid, &
                                  f1, f2, Source, f3, f4, deltaY, Mpart)

  call Fluxes              (Q1, D_Flux)

    DO np=1,Npart
      Xpart(np) = (Xpart(np) + TWO*Xpart1(np) + TWO*dt*Upart1(np))/THREE
      Upart(np) = (Upart(np) + TWO*Upart1(np) + TWO*dt*(f1(np)*(Ufluid(np)    &
                   Upart1(np))/Taup(np)    dpdxfluid(np)/RHOPref) )/THREE
      Tpart(np) = (Tpart(np) + TWO*Tpart1(np) + TWO*dt*(f2(np)*(Tfluid(np)    &
                  Tpart1(np)) f3(np)*deltaY(np))/Taup(np))/THREE
      Mpart(np) = (Mpart(np) + TWO*Mpart1(np)    TWO*dt*f4(np)*deltaY(np)* &
                           sqrt(Taup(np)))/THREE

    ENDDO
    if (update_particle == 0) Source = 0.0d0


    Q       = (Q    + TWO*Q1 + TWO*dt*(D_Flux + Source))/THREE

  call Boundary_Condition (Q )

  Source1(:,2:3) = Source(:,2:3)

!if (Time > 0.1) stop
!write(*,*) 'stop_here'
!stop

END Subroutine Runge_Kutta
```

A.1 Weigh Particle To Grid

```
Subroutine Weigh_Particle_To_Grid (Xpart, Upart, Tpart, Npart, Ufluid, Tfluid, &
                                   f1, f2, Source, f3, f4, deltaY, Mpart)

  INTEGER                              :: Npart
  REALTYPE, dimension(Npart)           :: Xpart, Upart, Tpart, Mpart
  REALTYPE, dimension(Npart)           :: Ufluid, Tfluid, f1, f2, f3, f4, deltaY
  REALTYPE, dimension(N0:N5,NV)        :: Source
  REALTYPE                             :: domain_length,dist1,dist2
  REALTYOE                            :: weight,weight1,   xfirst
  INTEGER                              :: npart_loc, np

  domain_length =  x(N5)    x(N0)
  xfirst        =  x(N0)

  Source          = 0.0
  DO np = 1, Npart
    npart_loc             = INT((Xpart(np) xfirst)/domain_length*DBLE(N5 N0)) + N0
    dist1                 = (Xpart(np) x(npart_loc))/dx
```

```
    dist2                  = 1.0 dist1
    weight                 = Mpart(np)*f1(np)*(Ufluid(np) Upart(np))/Taup(np)
    Source1(npart_loc,1)    = Source1(npart_loc,2)    + Mpart(np)*dist2
    Source1(npart_loc+1,1) = Source1(npart_loc+1,2) + Mpart(np)*dist1
    Source(npart_loc,2)    = Source(npart_loc,2)    + weight*dist2
    Source(npart_loc+1,2) = Source(npart_loc+1,2) + weight*dist1
    weight                 = weight*Upart(np)
    weight1                = Mpart(np)*(f2(np)*(Tfluid(np) Tpart(np))  &
                             f3(np)*deltaY(np))/Taup(np)/(gamma 1.0)
    Source(npart_loc,3)    = Source(npart_loc,3)    + weight*dist2 + weight1*dist2
    Source(npart_loc+1,3) = Source(npart_loc+1,3)  + weight*dist1 + weight1*dist1
  ENDDO

  Source =  Source*Source_factor/dx
  Source1  =  Source1*Source_factor/dx
!stop
!
END Subroutine Weigh_Particle_To_Grid
```

A.1.1 Interpolate Fluid To Particle ENO

```
Subroutine Interpolate_Fluid_To_Particle_ENO(Xpart, Upart, Tpart,  Npart, &
                  Q, Ufluid, Tfluid, f1, f2,dpdxfluid,  T_B, T_WB, Sc, Pr)

  INTEGER, PARAMETER                 :: norder  = 5
  INTEGER                            :: Npart
  INTEGER                            :: np, npart_loc, nloc1,p1,i,n,is,m,k
  INTEGER, DIMENSION(N0:N5)          :: left
  REALTYPE, dimension(Npart)         :: Xpart, Upart, Tpart, T_B, T_WB, Sc, Pr
  REALTYPE, dimension(Npart)         :: Ufluid, Tfluid, f1, f2, dpdxfluid
  REALTYPE, dimension(N0:N5,NV)      :: Q
  REALTYPE, dimension(N0:N5,norder+1):: c1
  REALTYPE, dimension(norder+1)      :: c,x1
  REALTYPE, dimension(N0:N5)         :: dpdx,p
  REALTYPE                           :: rep
  REALTYPE                           :: domain_length
  REALTYPE                           :: xa, xb, ua, ub, Ta, Tb
  REALTYOE                           :: rhoa, rhob, xfirst, pa, pb
  REALTYPE                           :: map,cd, f1temp, dist1, som
  REALTYPE                           :: hx(norder), cx(norder),Qpart(norder)
  REALTYPE                           :: argexp

!
! Determine divided differences
  DO i=N0+norder,N5 norder
     x1(:) = x(i:i+norder)
     c(:)  = Q(i:i+norder,2)
     DO p1=1,norder 1
        DO k=norder,p1+1, 1
           c(k) = (c(k) c(k 1))/(x1(k) x1(k p1))
        ENDDO
```

```
      c1(i,p1) = c(k)
     ENDDO
  ENDDO

!
! determine left most interpolating point with ENO approach
  DO i=N0+norder,N5
    is = i
    DO m=2,norder-1
      IF (abs(c1(is-1,m)) < abs(c1(is,m))) THEN
          is = is - 1;
      ENDIF
    ENDDO
    left(i) = is;
  ENDDO

  DO n=1,norder
    cx(n) = (n-1)*dx
  ENDDO

  DO n=N0,N5
    p(n) = (Q(n,3) - 0.5*Q(n,2)**2/Q(n,1))*(gamma-1.0)
  ENDDO

  call PS_Diff_WENO (1, WENO_Order, N0, N5, N2, N3, dx, p, dpdx)

  domain_length = x(N5) - x(N0)
  xfirst        = x(N0)
  DO np = 1, Npart
!
     npart_loc          = INT((Xpart(np) - xfirst)/domain_length*DBLE(N5 - N0)) + N0
     nloc1              = left(npart_loc)

     dist1              = (Xpart(np) - x(npart_loc)) + (npart_loc - nloc1)*dx

    DO n=1,norder
        CALL polyn1(n,dist1,norder,cx,hx(n))
    ENDDO
!
    DO i = 1,3
      som = 0.0
      DO n=1,norder
          som = som + Q(nloc1 + n - 1,i)*hx(n)
      ENDDO
      Qpart(i) = som

  ENDDO
     Rhofluid(np)       = Qpart(1)
     Ufluid(np)         = Qpart(2)/Qpart(1)
     Tfluid(np)         = (Qpart(3)/Qpart(1) - Ufluid(np)**2/2.0)*(gamma-1.0)*gamma
     Pfluid(np)         = Tfluid(np)*Qpart(1)/gamma

   !
    som = 0.0
```

```
      DO n=1,norder
           som = som + dpdx(nloc1 + n    1)*hx(n)
      ENDDO
      dpdxfluid(np)       = som
      !if (abs(dpdxfluid(np)) > 5.0) dpdxfluid(np) = 0.0

      rep                 = Rhofluid(np)*CRE*sqrt((Ufluid(np) Upart(np))**2)
      map                 = sqrt((Ufluid(np) Upart(np))**2)/sqrt(Tfluid(np)) + 1d 15

! Evaporation Model

      ! Boiling  (normalized) and Wet Bulb Temperature (K)
      T_B(np)    = 447.70d0/Tref
      T_WB(np)   = (137.0d0*((T_B(np)*Tref/373.150d0)**0.680d0)* &
                           log10(Tfluid(np)*Tref) 45.0d0)


      ! Prandtl and Schmidt (assuming Lewis number is unity)
      if (T_WB(np) >  600.0d0) Pr(np) = 0.647d0 + 5.5d 5*T_WB(np)
      if (T_WB(np) <= 600.0d0) Pr(np) = 0.815d0    4.958d 4*T_WB(np) + &
           4.514d 7*T_WB(np)**TWO
      Sc(np)     = Pr(np)

      CF2     = 1.0d0/(3.0d0*Pr(np))



      !f1(np)              = (1.+.15*rep**(.687))/(1.)
      !f1temp              = (1.+.15*rep**(.687))/(1.)
      !f1(np)              = 1.0d0
      !f2(np)              = 1.0d0

      argexp              = max( 100.0d0, 0.43d0/(map**4.67d0))
      cd                  = ( 24.0d0 + 0.38d0*rep + 4.0d0*sqrt(rep))*(1.0+exp(argexp))
      f1(np)              = 3.0d0*cd/(4.0d0*18.0d0)*1.4d0
      f2(np)              = CF2*(2.+.6*sqrt(rep)*Pr(np)**(.33))/(1.)*1.4
      !if (np==100) write(*,*) 'check argexp', np,map, argexp, cd,f1(np)

      !if (rep> 20000) WRITE(6,*) 'Re_p is too large, change setup', rep

   ENDDO

END Subroutine Interpolate_Fluid_To_Particle_ENO
```

A.1.2 Particle Evaporation (New Subroutine)

```
Subroutine Particle_Evaporation (Xpart, Upart, Tpart, Npart, Q, Ufluid, Tfluid, &
                                 f3, f4, dpdxfluid, deltaY, A1, A2, PD, Taup, Mpart)

    IMPLICIT NONE

    INTEGER                          :: Npart
```

```
   REALTYPE, dimension(Npart)       :: Xpart, Upart, Tpart, A1, A2, PD, Taup, Mpart
   REALTYPE, dimension(N0:N5,NV)    :: Q
   REALTYPE, dimension(Npart)       :: Ufluid, Tfluid, f1, f3, f4, dpdxfluid, deltaY

   INTEGER  :: np, npart_loc

   REALTYPE, parameter :: b1 = 24.0d0 , b2 = 0.380d0 , b3 = FOUR
   REALTYPE, parameter ::               b4 = 0.430d0 , b5 = 4.670d0
   REALTYPE, parameter :: d1 = TWO    , d2 = 0.60d0  , d3 = 0.330d0
   REALTYPE, parameter :: e1 = THREE/(FOUR*18.0d0)

   REALTYPE :: Yinf
   REALTYPE :: Pfp
   REALTYPE :: U2, Rep1, Map1, sRep
   REALTYPE :: Nu, Sh, Ys_eq, Ys, Bm, Bt, G
   REALTYPE :: Lk, beta, Ys_neq

   REALTYPE :: A3, CTAUP, CYS, CF3, CF4
   REALTYPE :: MWP, CPG, CVG, CPL, CPV, CVV
   REALTYPE :: MUref, Rhoref, Uref
   REALTYPE :: MU, LV, LV2
   REALTYPE :: R, CRe1, CD

   REALTYPE :: T_WB2


  R       = 8.314410d7                    ! erg/K
  Yinf    = 0.0d0

  DO np=1,Npart


!
! Physical parameters carier phase and liquid phase in CGS UNITS in the code:
! (G) continuum = gas/carrier phase
! (L) particle liquid phase
! (V) particle vapor and air mixture
!
! Air

! Heat capacities (erg/g/K for constant P, erg/cm3/K for constant V)
      CPG    = Pr(np)*(3.227d 3 + 8.3894d 5*T_WB(np)    1.958d 8*T_WB(np)**TWO)/  &
               (6.109d 6 + 4.604d 8*T_WB(np)    1.051d 11*T_WB(np)**TWO)
      CPG    = CPG*1.0d4
      CVG    = CPG    R/MWG
! Property evaluation
      Gamma = CPG/CVG
      !write(*,*) 'check_Gamma', Gamma, CPG, CVG, T_B, T_WB
      MU     = (6.10d 6 + 4.60d 8*(T_WB(np))    1.050d 11*(T_WB(np))**TWO)*TEN
!
! Decane
      !write(*,*) 'check_P_B', Rhofluid(np), T_B, Gamma, Mref
      P_B(np)    = Rhofluid(np)*T_B(np)/(Gamma*Mref**2)
```

```
      T_WB2 = T_WB(np)/1000.0d0
! Heat capacities (erg/g/K for constant P, erg/cm3/K for constant V)
      CPL   = 2.520d+7

      if (T_WB2 >  0.80d0) CPV = 0.0982d0 + 1.304d0*T_WB2   0.593d0*T_WB2**TWO &
                           + 0.101d0*T_WB2**THREE
     if (T_WB2 <= 0.80d0) CPV = 0.02547d0 + 1.377d0*T_WB2   0.4d0*T_WB2**TWO  &
                           + 0.113d0*T_WB2**THREE
      CPV   = CPV*4.1868d0*1000.0d0*1.0d+4
      CVV   = CPV R/MWP
! mass averaging of the gas mixture heat capacity
     !CPG   = CPG*(1 Yinf)+CPV*Yinf    ! not used since Yinf remains 0

! Property evaluation (Latent heat, erg/g)
      LV    = (3.9580d+4*(619.0d0 T_WB(np))**0.380d0)*1.0d+4
      LV2   = LV + (CPL CPV)*T_WB(np)


!
! constant coefficients used in particle equations

      A1(np) = LV/(CPG*Tref)          ! normalized Latent heat
      A2(np) = ONE                    ! A2=1 is used by Mashayek

      CTAUP  = (RHOP**(ONE/THREE)*re/18.0d0)*(SIX/PI)**(TWO/THREE)
      CYS    = Gamma*A1(np)/((Gamma ONE)*T_B(np))

      CF3    = A1(np)*A2(np)/(THREE*Sc(np))
      CF4    = PI*SQRT(18.0d0/RHOP)/(re**(1.50d0)*Sc(np))
      !write(*,*) 'check_CF4', CF3, A1(np), LV, CPG, T_WB(np)

      U2           = SQRT(ABS((Ufluid(np) Upart(np)))**2)
      CRe1         = re*Rhofluid(np)*(SIX/(PI*RHOP))**(1.0d0/3.0d0)
      Rep1         = Rhofluid(np)*CRe1*U2*Mpart(np)**(ONE/THREE)   ! particle re
      sRep         = SQRT(Rep1)

      Map1         = U2*Mref/SQRT(Tfluid(np)) + EPSILON(ONE)  ! Mach number
      Taup(np)     = CTAUP*Mpart(np)**(2.0d0/3.0d0)      !
      Sh           = TWO   !+d2*sRep*Sc(np)**(d3)              ! Sherwood nu
     !Nu           = TWO+d2*sRep*Pr(np)**(d3)                  ! Nusselt number
      Nu           = Sh


! Particle Diameter

      PD(np)       = (6.0d0*Mpart(np)/(PI*RHOP))**(ONE/THREE) ! variable particle diame
      !write(*,*) 'check_PD', np, PD(np), Mpart(np), Tpart(np)

! Evaporation model by Mashayek

      !Ys_eq       = (P_B(np)/Pfluid(np))*EXP(CYS*(ONE T_B(np)/Tpart(np)))
      Ys_eq       = EXP(CYS*(ONE T_B(np)/Tpart(np)))  ! used by Mashayek for slow evap.
      Ys          = Ys_eq
! for verification of the analtical solution take Y_s=const
```

```
        Ys          = 0.948766699d0
        !Ys          = 1.0d0
        Bm          = (Ys Yinf)/(ONE Ys)

        !write(*,*) 'check_Ys', np, Ys_eq, T_B, Tpart(np), Al(np)

        deltaY(np) = Bm
        deltaY(np) = Ys Yinf                    ! used by Mashayek for slow evap.

        CD          = (b1 + b2*Rep1 + b3*SRep)*(ONE+EXP( b4/(Map1**b5)))


      SELECT CASE (IEVAP)
        CASE (0)
        f3(np)      = 0
        f4(np)      = 0
        CASE (1)
        f3(np)      = CF3*Rhofluid(np)*Sh
        f4(np)      = CF4*Rhofluid(np)*Sh
      END SELECT
        !write(*,*) 'check_evap', f3(np), f4(np), Rhofluid(np)
        !write(*,*) 'check_f3', CF3, Rhofluid(np), Sh, f3(np)
        !write(*,*) 'check_f4', Rep1, f4(np), CF4, Rhofluid(np), Sh
    ENDDO
  !stop

END Subroutine Particle_Evaporatio
```