A Fast 3D Particle Method for Simulations of Buoyant and Reacting Flows

by

Fabrice Schlegel

M.S. MATMECA Engineering School
Bordeaux, France, 2005

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[June 2007]
May 2007

Signature of Author . . . . [illegible]
Department of Mechanical Engineering
May 12, 2007

Certified by. . . . . [illegible]
Ahmed F. Ghoniem
Ronald C. Crane (1972) Professor
Thesis Supervisor

Accepted by. . . . . [illegible]
Lallit Anand
Chairman, Department Committee on Graduate Students

A Fast 3D Particle Method for Simulations of Buoyant and Reacting Flows

by

Fabrice Schlegel

Submitted to the Department of Mechanical Engineering
on 21 May 2007, in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering

Abstract

This thesis describes progress in several areas related to three dimensional vortex methods and their application to multiphysics problems. The first is the solution of a generic scalar transport equation by advecting and diffusing the scalar gradient along a particle trajectory and onto a mesh, respectively, and recovering the scalar values using a Biot-Savart-like summation. The second is the accurate, high-resolution calculation of the velocity gradient using a fast treecode, which avoids using kinematic relations between the evolution of the gradients and the distortion of the flow map. The same tree structure is used to compute all the variables of interest and those required during the integration of the governing equations. Next, we apply our modified interpolation kernel algorithm for treating diffusion and remeshing to maintain long time accuracy. The coupling between the vorticity transport and that of a dynamic scalar, in this case the temperature or density in a gravitational field, is manifested by the generation of vorticity. We demonstrate the performance of the multiphysics algorithm by solving a number of buoyant and reacting flow problems.

Thesis Supervisor: Ahmed F. Ghoniem
Title: Ronald C. Crane (1972) Professor

Acknowledgements

Thanks are due first to my academic advisor Professor Ahmed F. Ghoniem of the MIT Reacting Gas Dynamics Laboratory for his continued support on this project and his insightful comments and advice.
I would also like to thank him, as well as Iraj Mortazavi, Professor of the Bordeaux I University for making my first internship at MIT possible. This experience has not only enriched my background in the area, but it has also sparked new interest in further research. Finally, I would like to thank Professor Ghoniem for his support for my admission for a Masters Degree in Mechanical Engineering at MIT at the end of my internship.

I am very thankful to my parents, Sylvie Thierry and Laurent Schlegel for their love and encouragement. Special thanks are due to my girlfriend, Duygu Sar, and my two sisters, Carole and Marine Schlegel, for their unconditional affection and emotional support.

Finally, I would like to thank the entire Reacting Gas Dynamics Laboratory, especially Daehyun Wee and Jean-Christophe Nave for their guidance in the project as well as Murat Altay for helping to make my transition to Boston easier. In addition, I am thankful for the support of the administrative staff in particular Lorraine M. Rabb.

Contents

List of Figures

Introduction

Lagrangian vortex methods [1, 2] are tools for computing complex unsteady fluid flows at high Reynolds numbers. While they have other advantages, such as the relaxation of the CFL condition and the suppression of numerical diffusion, one of their most interesting features is the fact that they are based on the discretization of vorticity. Especially in unconfined and semi-confined flows, a typical computational domain must extend to a size that incorporates regions where the primary variables, i.e., velocity and pressure, may deviate very slightly from their uniform distribution. This can result in an unmanageable computational effort in 3D or would require complex non-uniform grids that cluster around zones of high gradients and transition to coarser meshes closer to uniform zones. Vorticity, on the other hand, is derived from the curl of the velocity field, and can be described by computational elements contained in a smaller fraction of the total volume of the flow field. As the result, the computational elements are utilized more efficiently. Lagrangian transport of vorticity guarantees that its evolution in space and time is well resolved.

The extension of this idea to general transport problems has been suggested and implemented in several contexts. In this methodology, when solving for the transport of a scalar variable, one discretizes the gradients of the scalar field, instead of the scalar field itself. The evolution of the scalar field is hence determined by solving the corresponding transport equation for its gradients. The advantage of this approach is identical to that described in the previous paragraph for vortex methods. Since the gradients can be described by computational elements confined to a small fraction of the total volume of the domain, one can utilize the discrete elements more efficiently.

These ideas were first described for 1D problem in Ghoniem and Oppeneim for modelling diffusion processes [3]. Further developments were suggested and implemented more generically in Ghoniem and Sherman [4]. Anderson [5] extended the concept of gradient transport to convection in 2D, and to buoyant flows. A conservative formulation of that construction was suggested by Ghoniem *et al.* [6], called the transport element method, and used it for the simulation of mixing in shear flows. Krishnan and Ghoniem [7] extended the transport element method to nearly inviscid buoyant flows in 2D. They studied a two-dimensional Rayleigh-Taylor flow evolving under the action of gravity across a large temperature gradient, i.e., without the Boussinesq approximation. A reacting flow version of transport element methods was also proposed by Soteriou and Ghoniem [8, 9] to investigate the dynamics of two-dimensional reacting shear layers. Soteriou *et al.* [10] applied transport element methods to planar buoyant plumes simulations.

Three-dimensional transport element method was proposed by Knio and Ghoniem, and was used to simulate the evolution of a periodic shear layer [11, 12]. The construction of this method was, however, based on rather a complicated internal coordinate system inside each computational element, which made the implementation difficult. Essentially, the construction was based on a kinematic relation between the evolution of the local gradients and that of the distortion of the material elements. To implement this kinematic relation, one needs to evolve the underlying flow map, that is, both the location of the field particles and their connectivity. Dahm and Tryggvason [36,37] rely on assigning scalar-valued gradients to segment elements, but these segments that define a direction are still connected.

In this thesis, we resurrect the basic concepts of the transport element methods (TEM) in the context of three-dimensional multi-physics problems, where the vorticity field and the scalar field are coupled by baroclinicity, but simplify its implementation using a number of new ideas. This is achieved by assigning vector-valued gradients to particles with no connectivity. The scheme is equipped with a multi-purpose adaptive tree-code, which enables fast and accurate evaluation of various quantities required for the simulation, i.e., velocity, velocity gradients, and scalar distribution. Accurate and fast evaluation of velocity gradients enables us to solve the scalar gradient evolution without complex internal coordinate systems, with negligible loss of conservation properties. The capability of the scheme is demonstrated in various three-dimensional buoyant and reacting flows, especially the buoyant jet-in-crossflow case.

The thesis is organized as follows. Buoyancy-driven and reacting flows are studied in Section 2 and 3 respectively. Section 4 is dedicated to the simulation of the buoyant jet and our multi-purpose adaptive tree-code is presented in detail in Section 4. Conclusions are given in Section 6.

1. Lagrangian Simulation of Buoyant Flows

In this chapter, we consider the evolution of thermals. Single thermal spheres have been intensively studied and their behavior is now well known. An important theoretical and experimental work has been done by Scorer [17, 18], Wang [19], Lin [20], Turner [21, 22], and Escudier and Maxworthy [23]. Numerical simulations on thermals have been initially performed by Andersdon [5], and Marcus and Bell [24] in 2D. More generally, two-dimensional studies of the Rayleigh-Taylor instability were conducted by Baker *et al.* [25], Kerr [26], Tryggvason [27] and Zufiria [28, 29], using vortex methods. Three-dimensional simulations of buoyant bubbles were realized by Brecht and Ferrante [31] in the inviscid limit using a vortex-in-cell code. Walther and Koumoutsakos [32] extended the particle methods in 3D to the viscous case. We first present the evolution of a single thermal sphere, and then nonlinear interactions between two thermal spheres will be shown.

The governing equations are given in part 1 and the numerical algorithm is presented in part 2. Part 3 is dedicated to the simulation of a vortex ring. Buoyancy-driven flows are studied in part 4.

1.1. Governing equations

To demonstrate the capability of our transport element method, we study buoyancy-driven flows in $\mathbf{R}^3$. Using the Boussinesq approximation, the Navier-Stokes equation is given as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = \nu\Delta\mathbf{u} - \frac{1}{\rho}\nabla p - \mathbf{g}_\mathbf{r}\beta(T - T_\infty) \qquad (1)$$

$$\nabla\cdot\mathbf{u} = 0 \qquad (2)$$

where $\mathbf{u}$ is velocity, p is pressure, β is the volumetric thermal expansion coefficient of the fluid, and $\mathbf{g}_\mathbf{r}$ is the gravitational acceleration. T_∞ is the temperature of the environment, ∇ and Δ are the gradient and Laplacian operators. The temperature field, T, is governed by the following advection-diffusion equation:

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T = \alpha\Delta T \qquad (3)$$

Here, ν is the kinematic viscosity, and α is the thermal diffusivity.

We normalize Eq.(1), (2) and (3) by choosing a reference length L, which can be defined by a characteristic geometric length scale of the problem. The corresponding reference flow speed is given by $U = \sqrt{g_r L}$, where $g_r{}^2 = \mathbf{g}_\mathbf{r}\cdot\mathbf{g}_\mathbf{r}$. We also choose a reference temperature, T_0, which is different from T_∞. Then, with the following normalization: $\tilde{\mathbf{x}} = \mathbf{x}/L$, $\tilde{\mathbf{u}} = \mathbf{u}/U$, $\tilde{t} = t/(L/U)$, $\tilde{\theta} = (T - T_\infty)/(T_0 - T_\infty)$, $\tilde{p} = p/(\rho U^2)$, and $\tilde{\mathbf{g}}_\mathbf{r} = \mathbf{g}_\mathbf{r}/g_r$, we obtain

$$\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{t}} + \tilde{\mathbf{u}} \cdot \tilde{\nabla} \tilde{\mathbf{u}} = \frac{1}{\mathrm{Re}} \tilde{\Delta} \tilde{\mathbf{u}} - \tilde{\nabla} \tilde{p} - \frac{\mathrm{Gr}}{\mathrm{Re}^2} \tilde{\mathbf{g}}_\mathbf{r} \tilde{\theta}, \tag{4}$$

$$\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0, \tag{5}$$

$$\frac{\partial \tilde{\theta}}{\partial \tilde{t}} + \tilde{\mathbf{u}} \cdot \tilde{\nabla} \tilde{\theta} = \frac{1}{\mathrm{Pr}\,\mathrm{Re}} \tilde{\Delta} \tilde{\theta}. \tag{6}$$

Naturally, $\tilde{\nabla} = \frac{1}{L} \nabla$ and $\tilde{\Delta} = \frac{1}{L^2} \Delta$. The system is governed by three dimensionless parameters, i.e., the Reynolds number, $\mathrm{Re} = UL/\nu$, the Prandtl number, $\mathrm{Pr} = \nu/\alpha$, and the Grashof number, $\mathrm{Gr} = g_r \beta (T_0 - T_\infty) L^3/\nu^2$. In the following, all the variables are understood as being normalized in this form, and the tilde over each dimensionless variable is omitted.

Taking the curl of Eq.(4), we obtain the vorticity-velocity formulation for buoyant flows:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathbf{u} \cdot \nabla \boldsymbol{\omega} = \boldsymbol{\omega} \cdot \nabla \mathbf{u} + \frac{1}{\mathrm{Re}} \Delta \boldsymbol{\omega} + \frac{\mathrm{Gr}}{\mathrm{Re}^2} \mathbf{g}_\mathbf{r} \times \nabla \theta, \tag{7}$$

where $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, and $\frac{\mathrm{Gr}}{\mathrm{Re}^2} \mathbf{g}_\mathbf{r} \times \nabla \theta$ is the baroclinic source term for vorticity generation. Using the Helmholtz decomposition, the velocity field can be separated as follows:

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_p, \tag{8}$$

where $\mathbf{u}_\omega$ is the vortical velocity field, and $\mathbf{u}_p$ is the potential velocity field. The potential velocity is added to satisfy the normal velocity boundary conditions. In $\mathbf{R}^3$, where no apparent boundary exists, $\mathbf{u}_p$ is set to 0, and $\mathbf{u} = \mathbf{u}_\omega$. On the other hand, given a distribution of vorticity within a domain D, the vortical velocity in $\mathbf{R}^3$ is determined using the Biot-Savart law:

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_\omega(\mathbf{x},t) = -\frac{1}{4\pi} \int_D \frac{(\mathbf{x} - \mathbf{x}') \times \boldsymbol{\omega}(\mathbf{x}',t)}{|\mathbf{x} - \mathbf{x}'|^3} d\mathbf{x}'. \tag{9}$$

The set of equations, Eq.(6), (7) and (9), provides a complete description of the flow.

1.2. Numerical algorithm

Our approach uses Lagrangian particles as computational elements. The vorticity field is discretized into Lagrangian computational elements, or particles, with weights $\mathbf{W}_i(t)$, and locations $\boldsymbol{\chi}_i(t)$ such that:

$$\boldsymbol{\omega}(\mathbf{x},t) \approx \sum_i^{N_\omega} \mathbf{W}_i(t) f_\delta(\mathbf{x}-\boldsymbol{\chi}_i(t)) \tag{10}$$

N_ω is the number of vortex elements. The core function $f_\delta(\mathbf{r})$ is obtained from a reference function $f(r)$ by $f_\delta(\mathbf{r}) = \delta^{-3} f(|\mathbf{r}|/\delta)$. In this work, the reference function f is the low-order algebraic core function [16]:

$$f(r) = \frac{3}{4\pi}\frac{1}{(1+r^2)^{5/2}}. \tag{11}$$

To simulate buoyant flows, we need to additionally solve the transport equation of energy or temperature, Eq.(6). Since the baroclinic source term in Eq.(7) depends on the gradient of θ, instead of using the scalar values as weights and computing their gradients locally, we use the scalar gradients as the weights of the corresponding computational elements. The advantage of using the gradients as weights is that the support of the gradient is smaller than that of the scalar itself, and hence the computational elements can be distributed over a smaller fraction of the domain. For instance, a hot sphere can be represented by discretizing the spherical shell between the hot interior and the cold exterior, while no elements are used in the temperature domains inside or out.

Such a method is generally referred as a transport element scheme. In the context of the current problem, we discretize the gradient of θ as follows:

$$\mathbf{g}(\mathbf{x}) = \nabla\theta(\mathbf{x}) \approx \sum_i^{N_g} \mathbf{G}_i(t) f_\delta(\mathbf{x}-\boldsymbol{\zeta}_i(t)). \tag{12}$$

N_g is the number of transport elements. By taking gradient of Eq.(6), we get the governing equation for $\mathbf{g}$.

$$\frac{\partial \mathbf{g}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{g} = -(\nabla\mathbf{u})^t \cdot \mathbf{g} + \frac{1}{\Pr\mathrm{Re}}\Delta\mathbf{g} \tag{13}$$

The solution of the equations of motion is expressed in terms of the instantaneous locations, i.e., $\boldsymbol{\chi}_i$ and $\boldsymbol{\zeta}_i$, and weights, i.e., $\mathbf{W}_i$ and $\mathbf{G}_i$, of these elements.

The numerical solution of Eq.(7) and Eq.(13) is obtained through an operator splitting. The computational time step is split in three substeps, i.e., convection substep, generation substep, and diffusion substep. During each substep, we solve the following equations:

Convection substep: $\dfrac{D\boldsymbol{\omega}}{Dt}=\dfrac{\partial\boldsymbol{\omega}}{\partial t}+\mathbf{u}\cdot\nabla\boldsymbol{\omega}=\boldsymbol{\omega}\cdot\nabla\mathbf{u}$, (14)

$$\frac{D\mathbf{g}}{Dt}=\frac{\partial\mathbf{g}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{g}=-(\nabla\mathbf{u})^t\cdot\mathbf{g}, \tag{15}$$

Generation substep: $\dfrac{\partial\boldsymbol{\omega}}{\partial t}=\dfrac{\mathrm{Gr}}{\mathrm{Re}^2}\mathbf{g}_r\times\nabla\theta$, (16)

Diffusion substep: $\dfrac{\partial\boldsymbol{\omega}}{\partial t}=\dfrac{1}{\mathrm{Re}}\Delta\boldsymbol{\omega}$, (17)

$$\frac{\partial\mathbf{g}}{\partial t}=\frac{1}{\Pr\mathrm{Re}}\Delta\mathbf{g}. \tag{18}$$

During the convection substep, the solution of Eq.(14) and Eq.(15) is obtained by integrating the following equations:

$$\frac{d\chi_i}{dt}=\mathbf{u}(\chi_i,t), \tag{19}$$

$$\frac{d\zeta_i}{dt}=\mathbf{u}(\zeta_i,t), \tag{20}$$

$$\frac{d\mathbf{W}_i}{dt}=\mathbf{W}_i(t)\cdot\nabla\mathbf{u}(\chi_i,t), \tag{21}$$

$$\frac{d\mathbf{G}_i}{dt}=-\left(\nabla\mathbf{u}(\zeta_i,t)\right)^t\cdot\mathbf{G}_i(t). \tag{22}$$

The integration of these equations is performed using a second-order predictor-corrector scheme.

During the generation substep, Eq.(16) for the baroclinic generation of vorticity is integrated using a first-order scheme. We need to introduce additional vorticity, where nontrivial baroclinicity exists. This is achieved by generating one new vortex element at the location of each transport element, using the following expression for each *i*th transport element:

$$\mathbf{W}_i^g=\left(\frac{\mathrm{Gr}}{\mathrm{Re}^2}\mathbf{g}_r\times\mathbf{G}_i(t)\right)\Delta t \text{ for } 1\le i\le N_g. \tag{23}$$

The vorticity field is updated by adding these new vortex elements to the existing vortex elements.

$$\boldsymbol{\omega}(\mathbf{x})=\sum_{i=1}^{N_\omega}\mathbf{W}_i f_\delta(\mathbf{x}-\chi_i)+\sum_{i=1}^{N_g}\mathbf{W}_i^g f_\delta(\mathbf{x}-\zeta_i). \tag{24}$$

After the update is finished, N_ω is augmented by N_g.

During the diffusion substep, Eq.(17) and Eq.(18) are solved by using a modified interpolation kernel [13]. The existing field is interpolated over a new set of elements, whose location is selected to satisfy certain requirements. Through the interpolation process, the vorticity field is updated such that the strength of the new elements are:

$$\mathbf{W}_j(t+\Delta t)=\sum_i f_{ij}^{\omega}\mathbf{W}_i(t)\,, \tag{25}$$

where f_{ij}^{ω} is the redistribution fraction from the *i*th vortex element to the *j*th grid point, which in the current implementation is described by a uniform Cartesian grid with the grid size Δx as shown in Figure 1. f_{ij}^{ω} is obtained by using the interpolation kernel Λ_3:

$$f_{ij}^{\omega}=\Lambda_3\left(\frac{x_j-x_i}{\Delta x};c_\omega\right)\Lambda_3\left(\frac{y_j-y_i}{\Delta x};c_\omega\right)\Lambda_3\left(\frac{z_j-z_i}{\Delta x};c_\omega\right), \tag{26}$$

where

$$\Lambda_3(\xi;c)=\begin{cases}1-2c^2+|\xi|(3c^2-1/2)-\xi^2+|\xi|^2/2 & |\xi|<1\\ (2-|\xi|)(\frac{1}{6}(3-|\xi|)(1-|\xi|)+c^2) & 1\le|\xi|<2\\ 0 & 2\le|\xi|\end{cases} \tag{27}$$

Here $c_\omega=\sqrt{\mathrm{Re}^{-1}\Delta t}/\Delta x$, which represents the ratio between the diffusion length scale and Δx. As shown in [11], in this modified interpolation kernel, during each interpolation step the second-order moments are increased by the amount required to simulate diffusion. In classical interpolation, the kernel preserves these second-order moments.

In a similar way, we update the gradient field,

$$\mathbf{G}_j(t+\Delta t)=\sum_i f_{ij}^{g}\mathbf{G}_i(t)\,, \tag{28}$$

where f_{ij}^{g} is the redistribution fraction from the *i*th transport element to the *j*th grid point. f_{ij}^{g} is obtained by using Eq.(26), but with $c_g=\sqrt{\mathrm{Pr}^{-1}\,\mathrm{Re}^{-1}\Delta t}/\Delta x$ in place of c_ω.

At the end the interpolation, time is advanced to $t+\Delta t$, and that completes the entire step. By the end of the time step, the fields are again expressed with Eq.(10) and Eq.(12), where i runs over all the grid points with nontrivial values of $\mathbf{W}_i$ and/or $\mathbf{G}_i$. Note that the problem of Lagrangian distortion is resolved within the diffusion substep, since the new particles are uniformly distributed at the end of the substep.

During the convection substep, we need to evaluate $\mathbf{u}$ and $\nabla\mathbf{u}$ at the location of each computational element. A naïve implementation of this process leads to an expensive operation, whose cost scales as $O(N^2)$. The recovery of θ from $\mathbf{g}$, which is necessary during post-processing, also requires similar set of operations. To reduce the computational cost of these tasks, we use a multi-purpose adaptive tree-code, which is described in Appendix A.

Also we note that, for isothermal flows, Eq.(6) is redundant, and the source term for vorticity generation in Eq.(7) drops out. In this hydrodynamic limit, the computational algorithm reduces to the standard vortex element scheme, where only convection and diffusion, i.e., Eq.(14) and Eq.(17), of vorticity is implemented. In the following sections, we first describe some results at this hydrodynamic limit, and then present the results of buoyant flow simulations.

1.3. Evolution of a vortex ring

In the following, we examine the accuracy and convergence of our algorithm. We first apply the algorithm at the hydrodynamic limit to perform the simulation of a vortex ring. The evolution of a viscous vortex ring was studied with an axisymmetric spectral calculation by Stanaway *et al.* [14], and the result was later reproduced in a vortex calculation by Wee and Ghoniem [13]. The initial vorticity distribution of the ring core is given by

$$\omega_\phi = \frac{K}{\pi}\frac{\Gamma}{a^2}\exp\left[-K\left(\frac{R^2}{a^2}+\frac{r^2}{a^2}-\frac{2Rr}{a^2}\sin\theta\right)\right] \qquad (29)$$

with $r=\sqrt{x^2+y^2+z^2}$, $\tan\theta=\frac{\sqrt{x^2+y^2}}{y}$ and $K=\frac{(2.24182)^2}{4}$. The core radius is chosen to be $a/R = 0.35$. The initial ring radius, $R(0)$, and its initial circulation, $\Gamma(0)$, are unity. Its evolution is computed for a Reynolds number $\mathrm{Re}\equiv\Gamma/\nu=500$.

The numerical parameters are chosen as follows: the time step for the highest resolution simulation is Δt=0.15 and the grid size for the diffusion substep is $\Delta x = 0.025$. Because of diffusion, the vorticity support expands and the number of particles grows in time. To control the number of particles, particles with its strength below a cutoff value are deleted after each diffusion substep. The cutoff value for deletion is chosen to be $|\omega dV|_{del}=10^{-11}$.

The simulation is initialized by computing by the vorticity distribution on a Cartesian grid with a grid size $\Delta x = 0.025$ and a cutoff value $|\omega dV|_{del}=10^{-11}$. The number of vortex elements at the beginning of the simulation is around 1,900,000.

The results are reported in terms of the following dimensionless variables. The dimensionless time is given by

$$\bar{t}=\frac{\nu^2}{I_0/\rho}, \qquad (30)$$

where I_0 represents the linear impulse of the vortex ring. The dimensionless speed of the vortex ring centroid is defined by

$$\bar{U}=U_c\frac{(I_0/\rho)^{\frac{1}{2}}}{\nu^{\frac{3}{2}}}. \qquad (31)$$

where U_c is the ring centroid velocity measured in the computational units.

We present the vorticity contours at $\bar{t} = 6.75\times10^{-5}$, 7.48×10^{-5}, 8.21×10^{-5}, 9.06×10^{-5}, 10.03×10^{-5} and 11.85×10^{-5}. The vorticity contours shown in Figure 2 match well those previously reported [13, 14]. Even the subtle tail structures at $\bar{t} = 10.03\times10^{-5}$ are well captured.

The predicted location of the vortex ring centroid velocity shown in Figure 3 also compares well with the results of previous calculations. We note that the current calculation which was performed using the full three dimensional representation of the ring structure matches more closely the two-dimensional spectral calculation results obtained by Stanaway *et al.* [14] than those reported in [13]. The ring maintains its two-dimensionality during the simulation and hence it is possible to compare our three-dimensional results with the two-dimensional results. The results of Wee and Ghoniem were obtained using a 20 degrees section to reduce the number of computational elements. The current results were obtained for the full 360 degrees ring representation. The number of vortex elements at the end of the simulation is around 3,500,000.

The use of our more efficient vortex-particle algorithm, instead of the vortex filament algorithm in [13], allows us to perform simulations with a smaller grid size. As a consequence, we have better accuracy and can accommodate up to 5 millions particles even with a serial implementation.

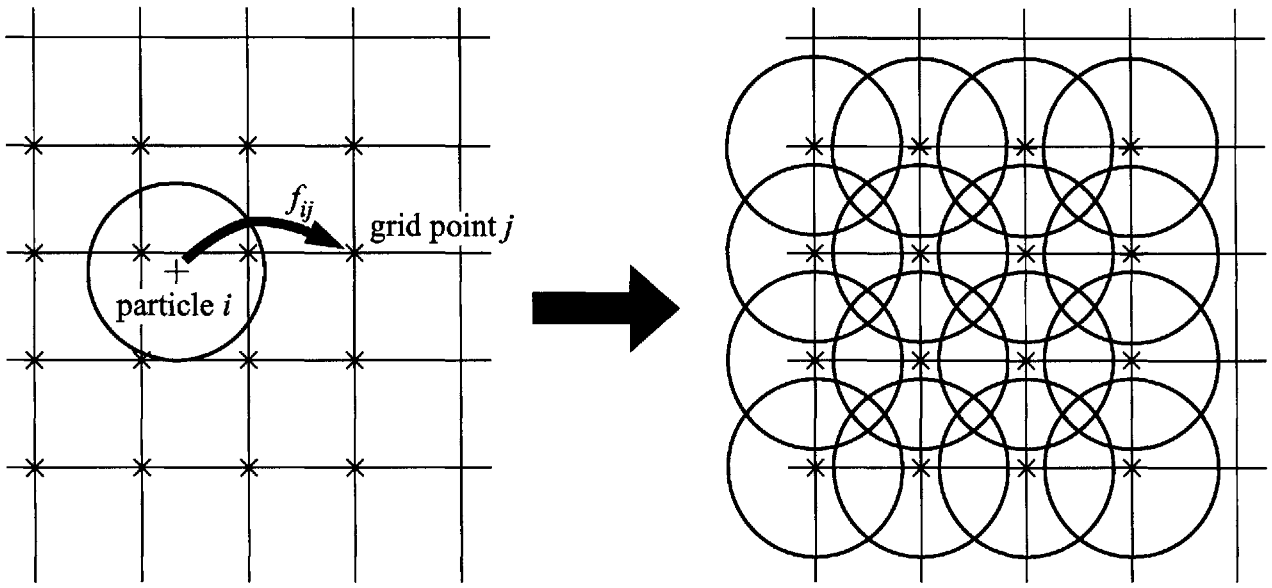

Figure 1: Schematic illustration of a diffusion substep. The strength of each particle is interpolated onto its nearest neighboring grid points (left). After finishing interpolation for all the particles, each grid point containing nontrivial strength is converted into a particle (right).

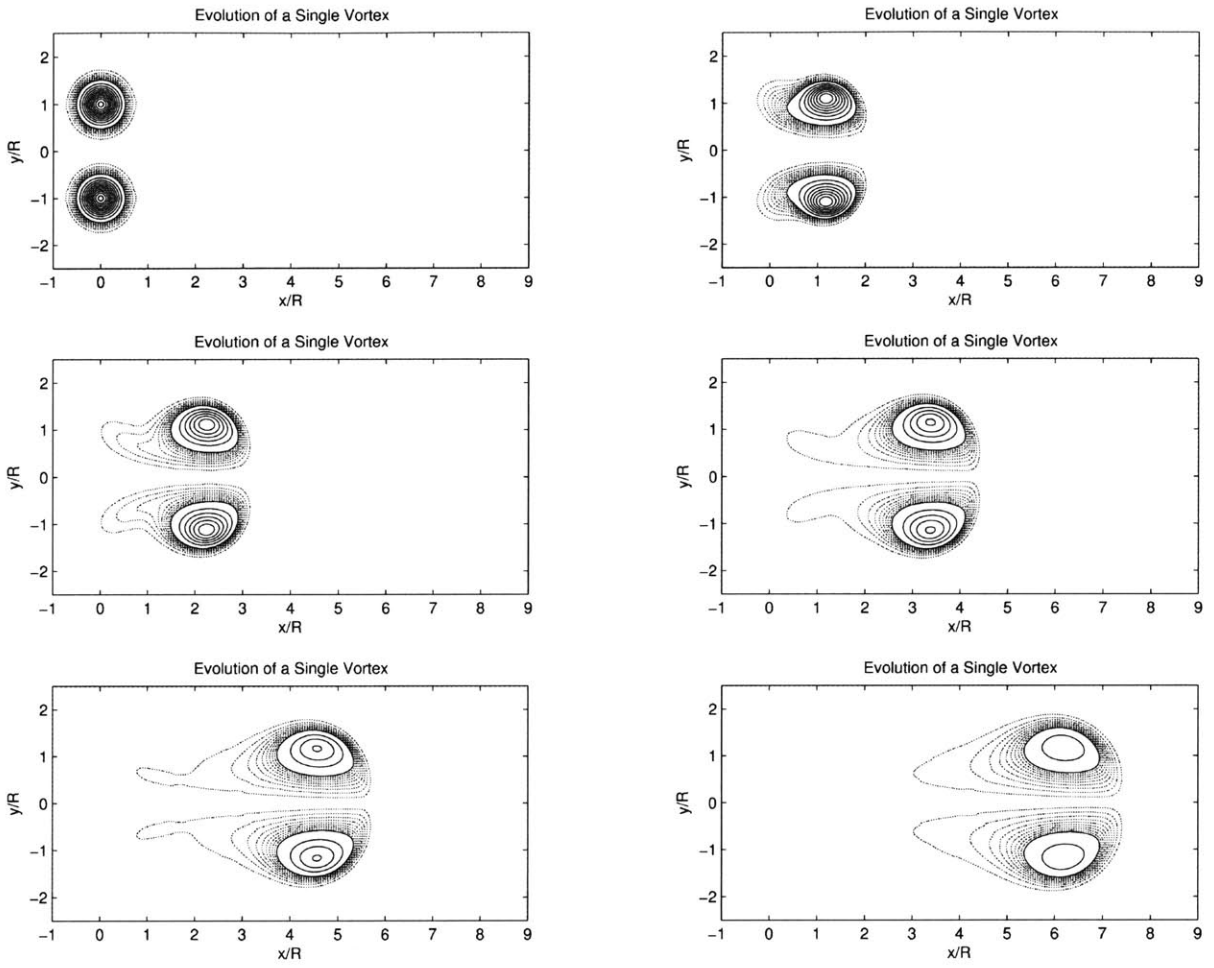

Figure 2: Vorticity contours of the evolution of a single vortex ring for the times $t^{-} = 6.75\times 10^{-5}$, 7.48×10^{-5}, 8.21×10^{-5}, 9.06×10^{-5}, $10.03\times\ 10^{-5}$ and 11.85×10^{-5}. The vorticity difference between two solid lines is ten times higher as the vorticity difference between two dashed lines.

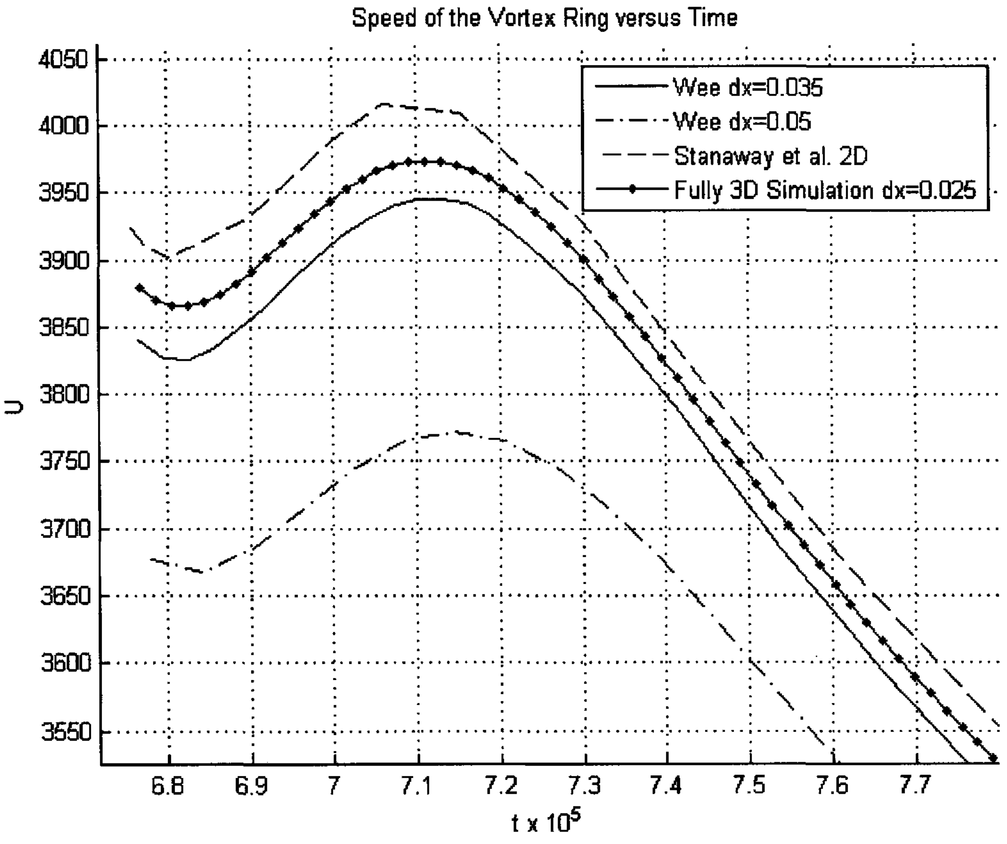

Figure 3: The vortex ring velocity for the highest resolution fully 3-D simulation (solid curve) using the current vortex particle algorithm. The results obtained in 2-D by Stanaway *et al.* [14] are plotted in dashed lines, and those obtained by Wee and Ghoniem [13] for a 20 degrees section simulation performed in parallel are plotted in solid lines.

1.4. Buoyancy-driven flows

As mentioned earlier, the development of a fast tree-code to compute the velocity and its gradient enable us to resurrect the idea of the transport element methods (TEM), in which the gradients of primitive variables are used as weights for computational particles, instead of the primitive variables themselves. In the following, we demonstrate the capability of our combined strategy, i.e., vortex element/transport element scheme, by computing buoyancy-driven flows. We first present the evolution of a single thermal sphere, and then nonlinear interactions between two thermal spheres will be shown.

1.4.1. Evolution of a thermal sphere

A sphere of hot air is placed in relatively cold ambient atmosphere, such that its center is initially at the origin. The radius of the sphere is used as the reference length scale for normalization, i.e., R=1. Dynamically, the difference in temperature between the hot and cold air drives the sphere against gravity through buoyancy. This phenomenon can be kinematically described by the baroclinic generation of vorticity around the surface of the sphere.

The initial temperature profile is defined by the error function, i.e., $\theta(\mathbf{x}) = \frac{1}{2}\text{erfc}\left(\frac{|\mathbf{x}|-R}{\delta_T}\right)$, where δ_T is the thickness of the temperature transition layer. The gradient profile is given by a Gaussian distribution, $\nabla\theta = -\frac{1}{\sqrt{\pi}\delta_T}\exp\left[-\left(\frac{|\mathbf{x}|-R}{\delta_T}\right)^2\right]\mathbf{e}_r$, where $\mathbf{e}_r$ represents the radial unit vector. The profiles are shown in Figure 4. As described above, using gradients as weights for the particles, we only need to cover the support of the gradient. Gravity is pointed in the negative y-direction.

The parameters are chosen as follows:

$$\text{Pr} = \nu/\alpha = 1, \tag{32}$$

$$\text{Re} = \frac{U^* R}{\nu} = \frac{R\sqrt{gR}}{\nu} = 1000, \tag{33}$$

$$\text{Gr} = \frac{\rho_\infty g_r R^3 \Delta\rho}{\mu^2} \approx \frac{g_r R^3 \beta \Delta T}{\nu^2} = 5\times10^5, \tag{34}$$

$$\text{Gr}/\text{Re}^2 = \frac{1}{2}. \tag{35}$$

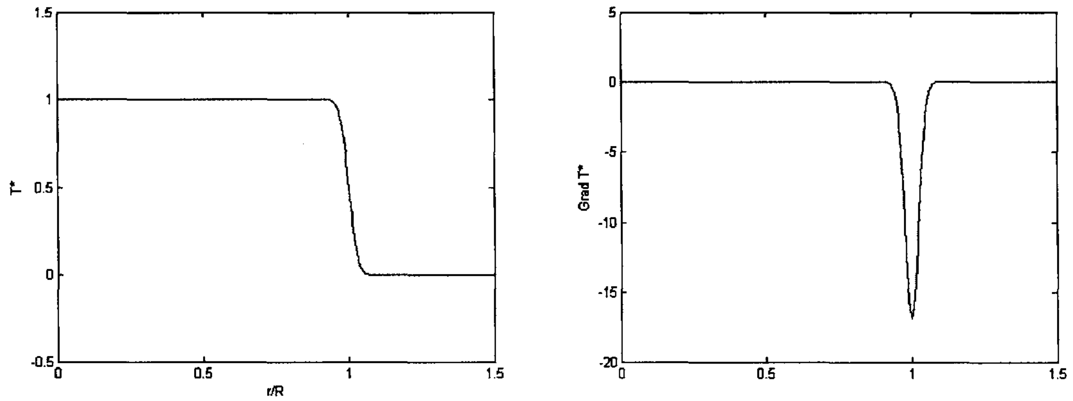

Figure 4: View on a radial cut, of the value of the temperature distribution on the left (*erfc*), and the value of the temperature gradient distribution on the right (*Gaussian*), at t=0s.

The thickness of the temperature transition region is given by $\delta_T = 1/30$,i.e., the temperature difference is allowed to spread over 1/30 of the initial radius of the sphere before we start the simulations. The grid size for diffusion is $\Delta x = 0.05$ and the time step size is $\Delta t = 0.125$. Figure 5 shows the evolution of both the temperature and the vorticity on the left hand side and the right hand side respectively. The temperature is recovered from the gradient elements, using the method described in the appendix for fast summation over gradient elements:

$$\theta(\mathbf{x}) = -\sum_j \mathbf{K}_\delta(\mathbf{x},\zeta_j)\cdot\mathbf{G}_j \tag{36}$$

As expected, the sphere is driven against gravity through buoyancy. The vorticity generated on both sides of the sphere rolls up forming a complex ring structure.

A convergence study is performed by repeating the same simulations for different grid sizes, Δx=0.025, 0.035, 0.05 and 0.1. The corresponding time step size is determined by $c_\omega = c_g = 2^{-1/2}$, which represents the ratio between the diffusion length scale and the grid size. The temperature is computed on a two-dimensional Cartesian grid, $\Delta x = 0.025$, by performing our fast summation over all the computational elements. The temperature centroid is defined as

$$y_T = \frac{\int \theta\, y\, dV}{\int \theta\, dV}. \tag{37}$$

The position of the buoyant sphere temperature centroid is plotted in Figure 6 for different resolutions. The error is defined as

$$\left\| Error \right\|_{L_2} = \sqrt{\frac{1}{N^2} \sum_{t=0}^{N\Delta t} \left(y_T(t, \Delta x) - y_T(t, \Delta x = 0.025) \right)^2}\,, \qquad (38)$$

where N is the number of time steps. The highest resolution simulation, $\Delta x = 0.025$, was compared with the three other simulations obtained using coarser grids. The order of convergence is 1.45. Note that we use a second order scheme for the convection step, while the diffusion and the baroclinic generation of vorticity are first order. In all cases the flows remains essentially two-dimensional.

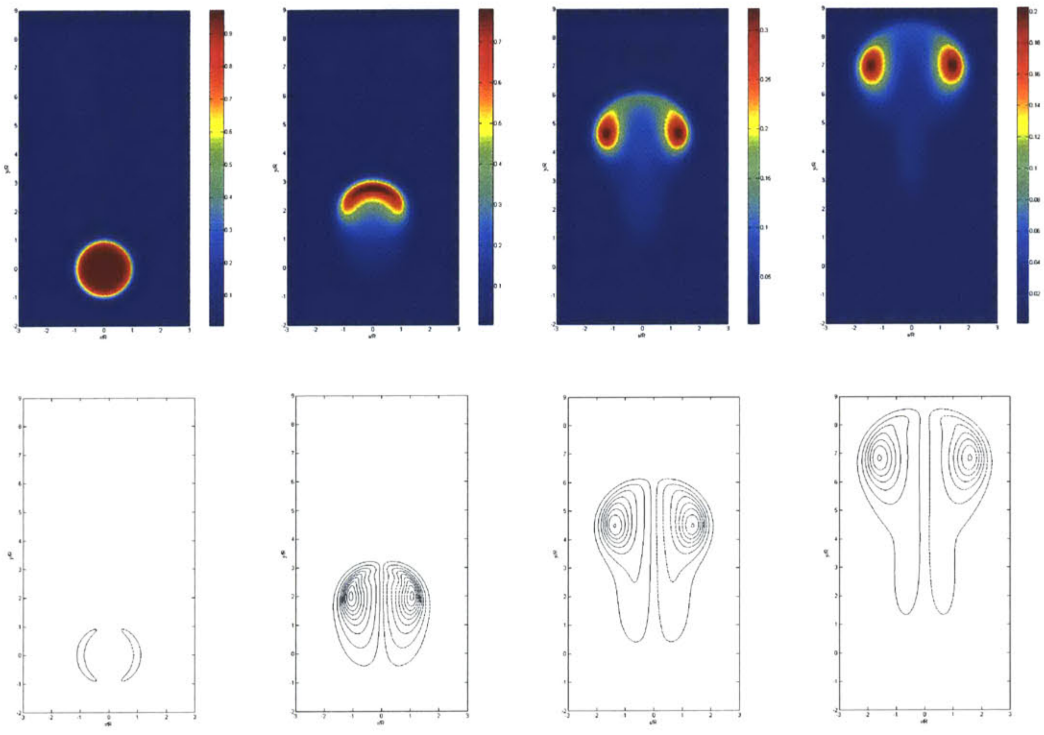

Figure 5: Evolution of the buoyant sphere, 2D-cut (3D simulation); temperature contours on the top and vorticity contours on the bottom, for the times $\overline{t}$ =0.125, 2.4, 4.8 and 7.2. The two first contour values are 0.1 and 0.33, then their values vary linearly with an increment of 0.33.

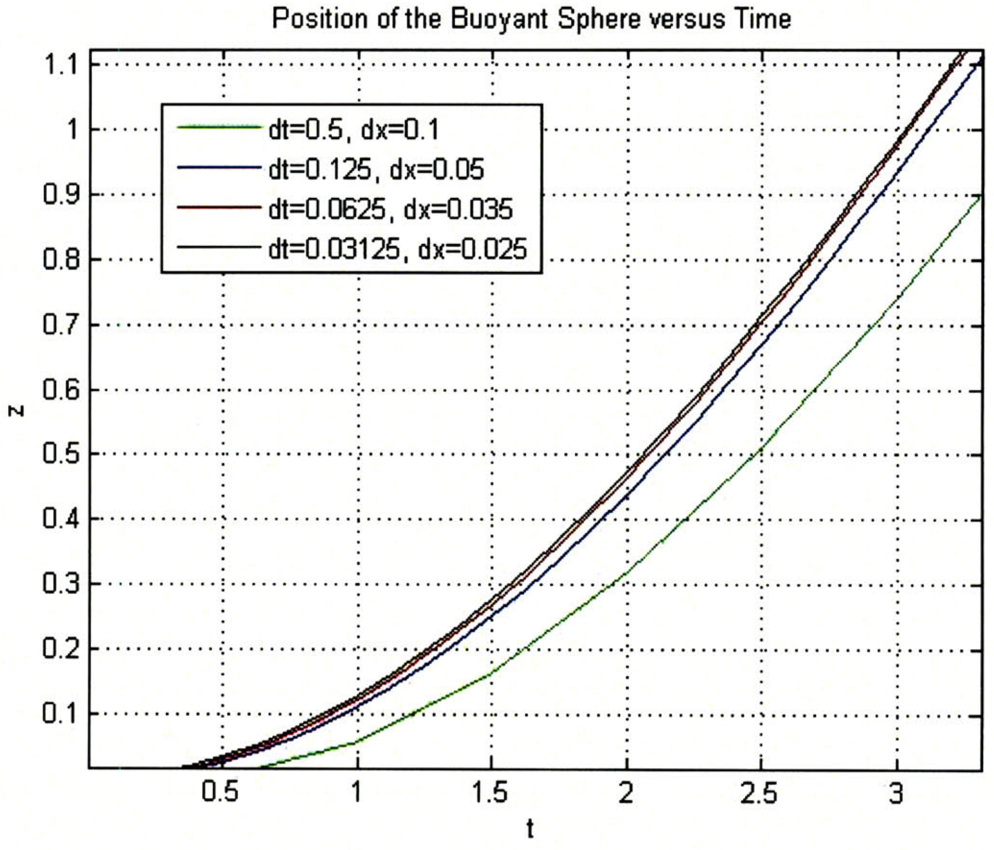

Figure 6: Position of the temperature center of the buoyant sphere for different resolutions.

1.4.2. Interactions of two thermal spheres

1.4.2.1. Side-by-side interaction

Figure 7 shows another sample calculation used to demonstrate the three-dimensional capabilities of the code: two hot-air spheres are evolving under a gravity field in the z-direction. In this case the two spheres are initially placed side-by-side. We again use the same parameters that we used in the single buoyant sphere simulation but allow Δx to grow in time such that $\Delta x = 0.035$ for $t \in [0,3.75]$, $\Delta x = 0.05$ for $t \in [3.75,12.37]$ and $\Delta x = 0.07$ for $t > 12.37$. The corresponding time step size is determined by $c_\omega = c_g = 2^{-1/2}$. Figure 7 and Figure 8 show 3D plots of the temperature contour, $\theta = 0.3$, and the vorticity isosurface, $|\omega| = 1.2$. These figures show that, due to diffusion, the distinction between the two spheres is lost and a continuous complex structure is formed a short distance into their vertical rise. The distortion of the temperature isosurfaces because of the mutual interactions between the two sphere leads to the formation of a complex tangle of vorticity structures. The results are plotted in two dimensional cuts across the vertical plane of the initial centers of the two spheres in Figure 9 and Figure 10 respectively, to show the strong distortion of the temperature contours, followed by the inter-diffusion between the neighbors spheres and the corresponding vorticity field in that plane.

The vortical structures formed due to the interaction between generation, convection and diffusion are well illustrated in Figure 8. Here, we observed two distinct rings whose individual structures resemble those observed in the single sphere simulations only at the very early stages. As these rings evolve, a mutual distortion of the overall structure is observed starting at t=1.3 in Figure 8. This distortion is most pronounced in the contortion of the initial two rings upwards, where they intersect close to the anti-symmetry plane, and the formation of two new crescent shaped structures between these rings. Although each hot sphere leads to the formation of a single vortex ring, as seen before, and the two spheres initially form two side-by-side vortex rings, the similarity ends here, as explained next.

It is interesting to notice that the underlying physics is here much different from what is observed in the case of the interaction of two vortex rings. The general behavior of two colliding side-by-side vortex rings is well known and can be seen in [13]. In the case of the side-by-side ring propagation, the following features are observed. At the early stages, before the inner vorticity cores inter-diffuse and values with opposite signs annihilate each other, or before the inner cores connect, the downward motion near the anti-symmetry plane pushes the inner cores downward with respect to the outer cores. At the later stages, however, after the two inner cores connect and their vorticity dissipates by inter-diffusion, the strength of the inner cores becomes weaker that those associated with the outer cores. The motion is now reversed and the inner cores move upwards with respect to those of the outer cores.

In the case of the side-by-side hot spheres, the two rings that form early in the evolution are contorted upwards near the anti-symmetry plane from the very early stages, as seen at t=3.3. In this case, the reconnection between the two hot spheres reduces the vorticity generation rate in the inner cores below that at the outer cores. This is shown by the empty zone at t=1.3. Moreover, the diffusion of the opposite signs vorticity generated within the inner cores further weakens their impact with respect to that of outer cores. The formation of relatively uniform temperature zones near the anti-symmetry plane is shown in Figure 9 as the two spheres connect/diffuse. Meanwhile, higher temperature zones persist at the outer cores even at the later stages. The impact of this distortion on the vorticity field is seen in Figure 10, where from the early stages of the simulation, the outer vorticity cores are larger and the inner cores are driven upwards from $t \geq 0$.

Parallel to the formation and distortion of the two side-by-side rings as the two hot spheres rise, we observe the formation of two crescent shaped vortical structures that "hang" below the two rings as they propagate upwards. These two crescents form as the two sphere interconnect from below, as seen in Figure 9 at t=3.3. The baroclinic generation associated with the bridge between the two initial spheres is consistent with the temperature distribution within this bridge, as shown in Figure 11. Note that the vorticity forms at the interface between the hot fluid originally in the spheres and the cold fluid outside.

To quantify the vorticity within these crescent shaped structures and show their impact on the velocity field, we plot their vorticity distribution and the total velocity on a y-z plane located half way between the original two spheres in Figure 12. The plot shows that these two structures contribute significantly to the upward motion at the anti-symmetry plane. It is interesting to observe that, at the late times, the vorticity associated with the original two rings decay, while that contained in these two crescents persist. This is confirmed by Figure 8, at t=17.1.

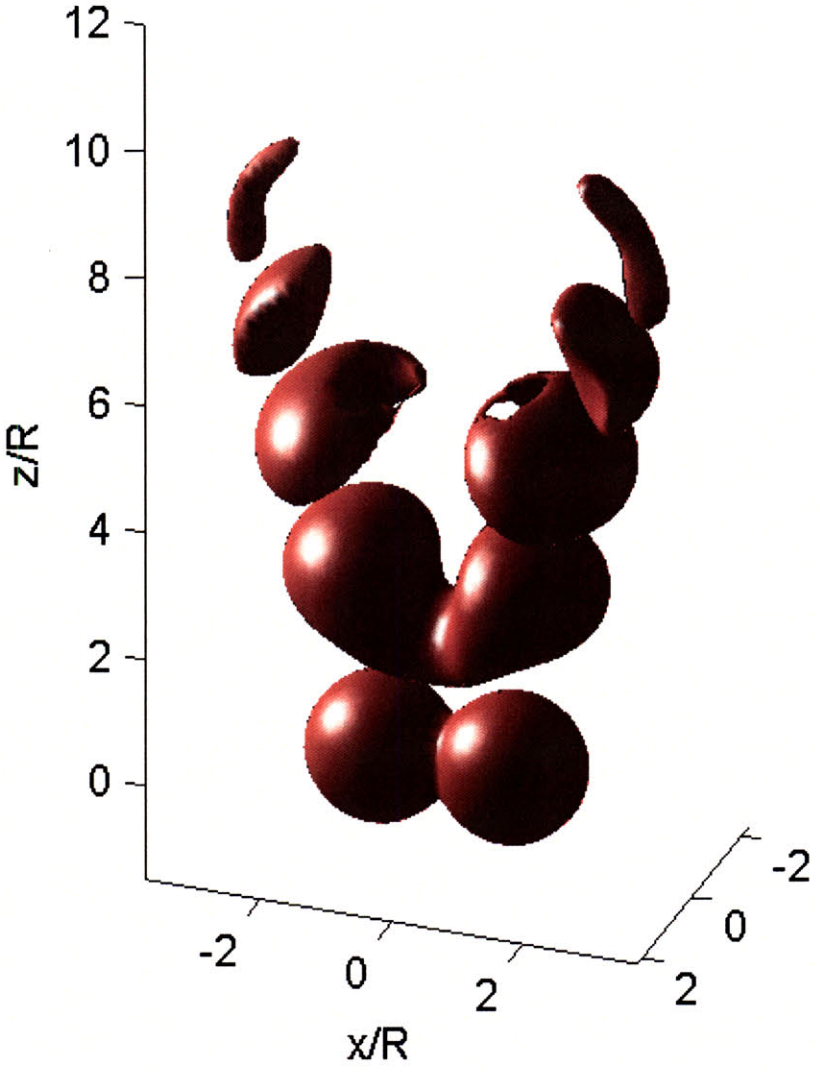

Figure 7: Temperature isosurface, $\theta = 0.3$, of the evolution in time of two thermal spheres of initial radius R=1, for $\bar{t}$ =0, 6.5, 10.5, 14.5 and 19.

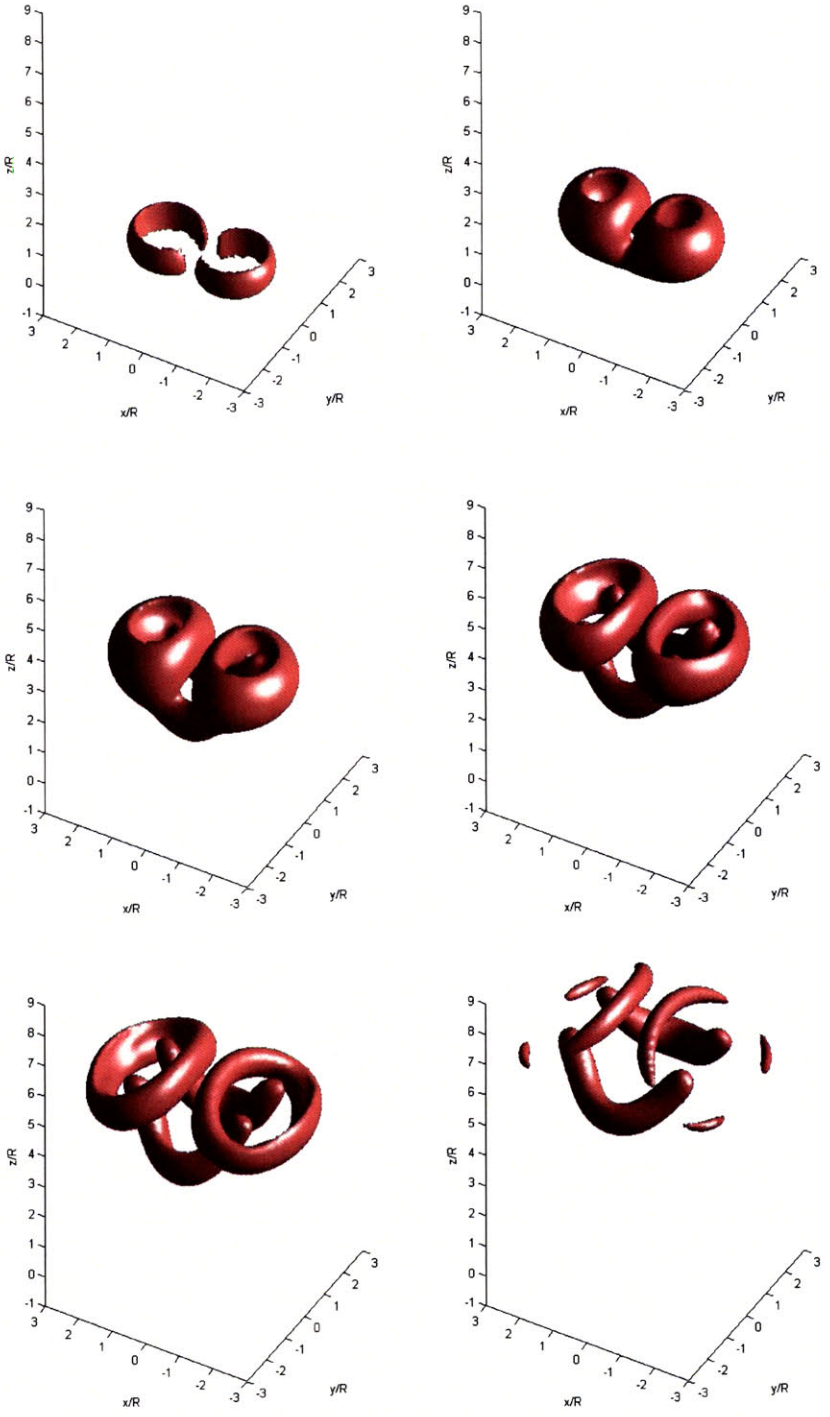

Figure 8: Vorticity isosurface, $|\omega| = 1.2$, of the evolution in time of two thermal spheres of initial radius R=1, for $\overline{t}$ =1.3, 3.3, 6.9, 8.9, 11.9 and 17.1.

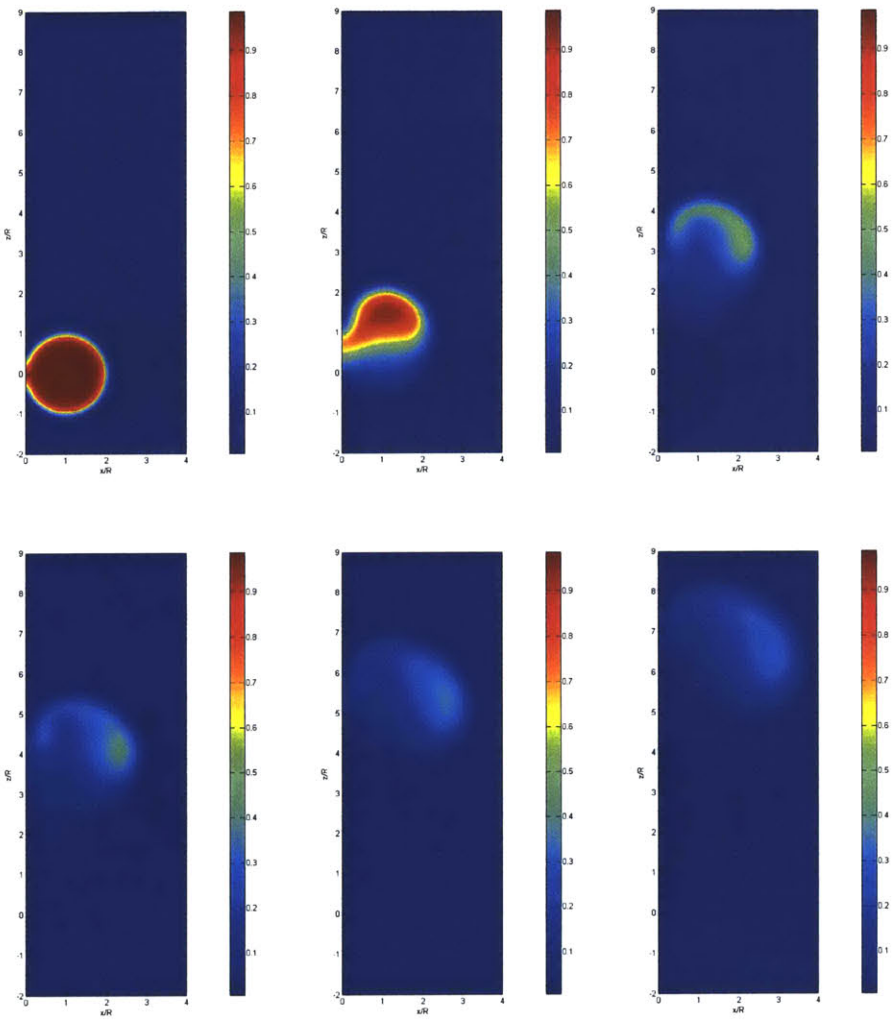

Figure 9: Temperature contours of the evolution in time of two thermal spheres of initial radius R=1, for $\overline{t}$ =0, 3.3, 6.9, 8.9, 11.9 and 15.1.

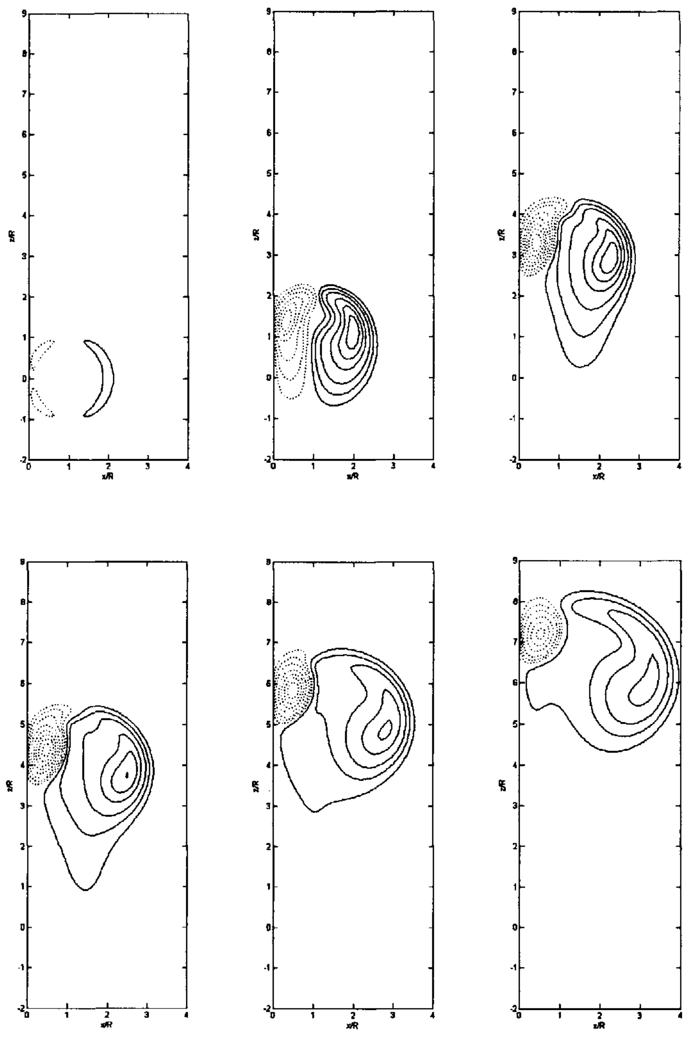

Figure 10: Vorticity contours of the evolution in time of two thermal spheres of initial radius R=1, for $\overline{t}$ =0.125, 3.3, 6.9, 8.9, 11.9 and 15.1. The three first contour values are 0.1, 0.25 and 0.5, then their values vary linearly with an increment of 0.5.

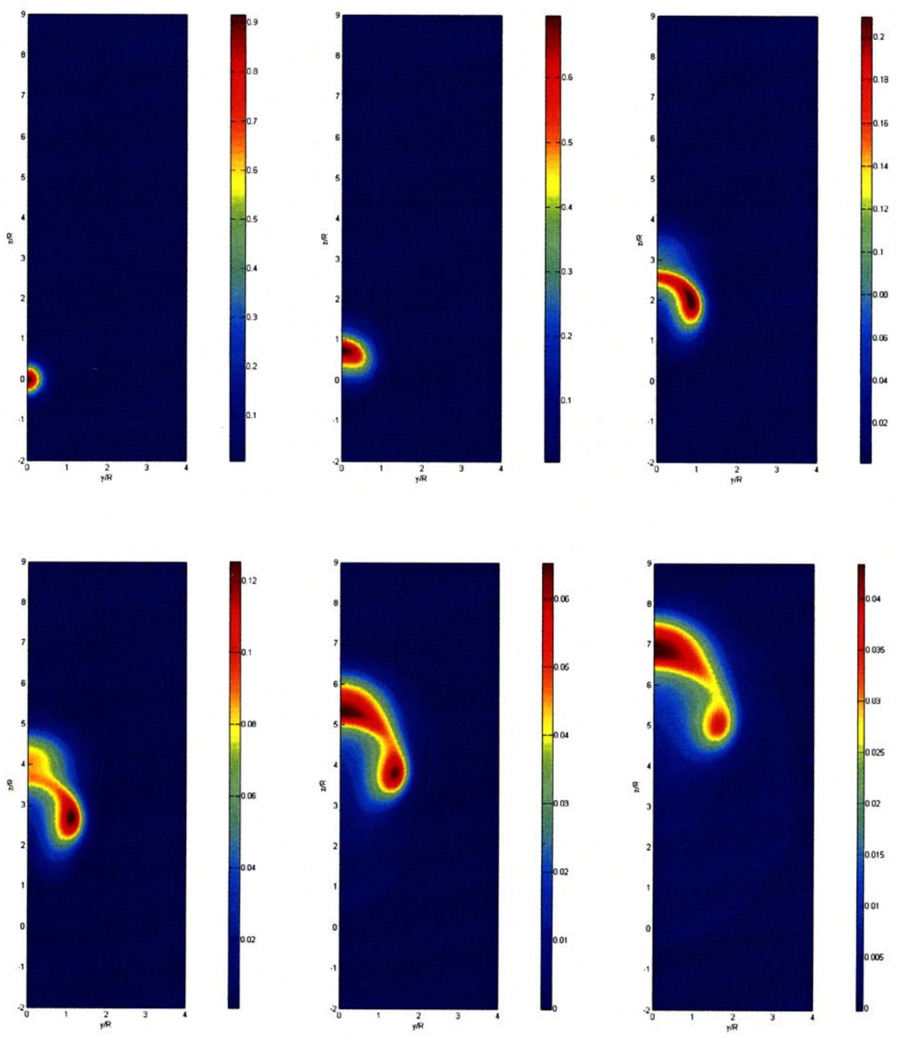

Figure 11: Temperature contours of the evolution in time of two thermal spheres in the y-z plane at x=0, for $\bar{t}$ =0, 3.3, 6.9, 8.9, 11.9 and 15.1.

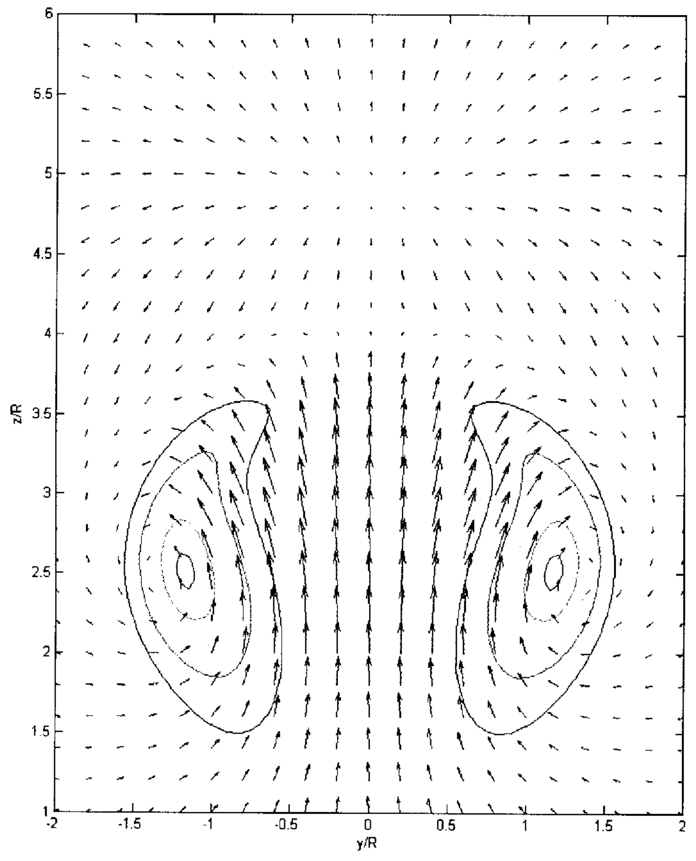

Figure 12: Velocity field induced by the vortical structures in the yz-plane at x=0 at time $\bar{t}$ =8.9. The contours correspond to the vorticity norm, their values vary linearly from 0.5 to 2.5, with an increment of 0.5.

1.4.2.2. Two spheres with different sizes

Another calculation that demonstrates the three-dimensional capability of our method is that shown in Figure 13 and Figure 14, where initially the upper sphere has a radius of 1.5 and the lower one has a radius of 1. To manifest the three-dimensionality, the centers of the two spheres are shifted in the lateral direction with respect to the vertical (gravity) direction. Hence, the upper is centered at (0,0,3), and the lower one at (0.5,0,0). We use the same numerical parameters that we used in the single buoyant sphere simulation. The temperature isosurface, θ=0.3, and vorticity isosurface, $|\omega|$=1.4, are plotted in 3D in Figure 13 and 14 respectively. The temperature and vorticity contours are plotted in Figure 15 and 16. The two spheres are initially at the same temperature. Initially, the vorticity generated on the sides of both spheres roll up forming a vortex ring, as it was the case for the buoyant sphere case. However, soon after the formation of the initial vorticity, the two newly formed rings exhibit the traditional vortex ring leap frogging mechanism. The initial eccentricity augments this motion.

The results show that the asymmetry introduced by the eccentricity persists, as manifested by the motion of the smaller sphere towards the left while it passes through the larger sphere. The material in the smaller sphere is drawn into a long thin structure by the stronger vortical structure formed by the larger sphere. Meanwhile, as that structure "punches" through the larger sphere, it forces more of its fluid to move towards the left side.

A convergence study is performed by repeating the same simulations for different grid sizes, Δx=0.035, 0.05, 0.07 and 0.1. The corresponding time step sizes are determined by $c_\omega = c_g = 2^{-1/2}$. The temperature centroid is defined in (37) . The position of the buoyant spheres temperature center is plotted in Figure 17. The error in the temperature centroid is defined in (38) . The order of convergence is found to be 1.45. It is similar to what was found previously.

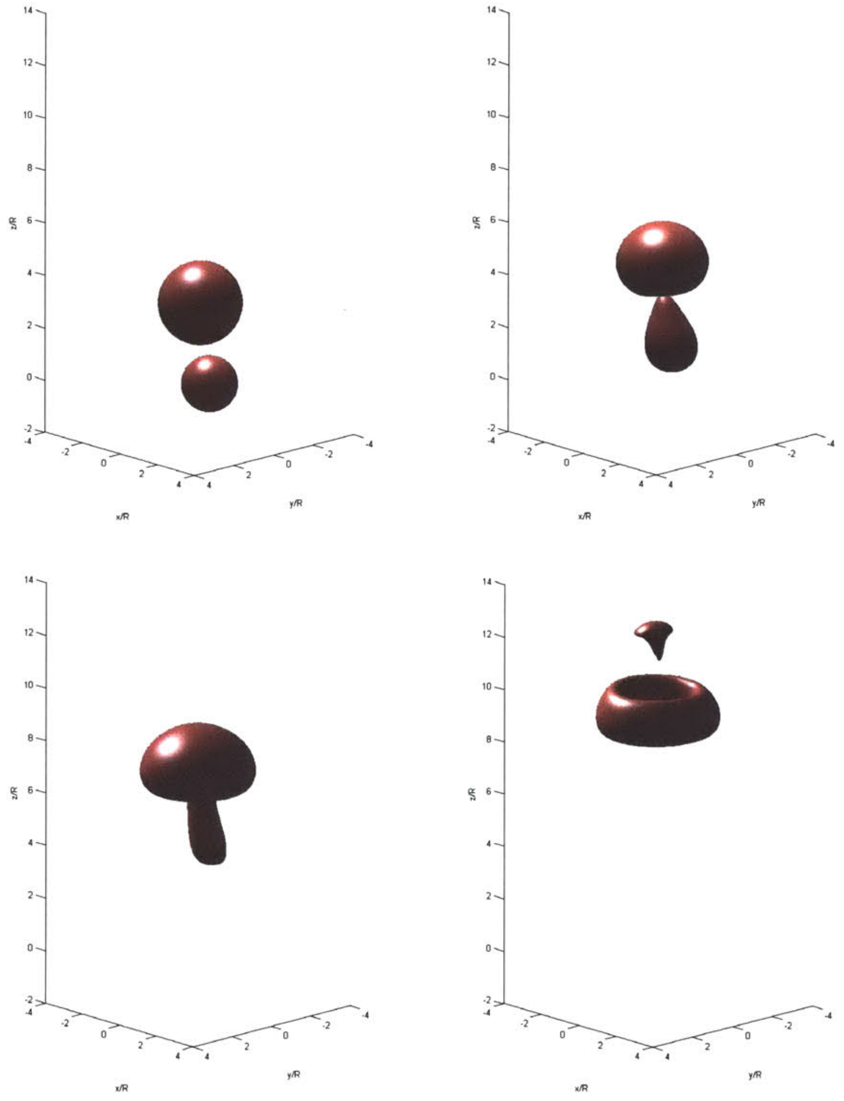

Figure 13: Evolution of two thermal spheres; temperature isosurface, $\theta = 0.3$, for $\bar{t}$ =0, 3.5, 7 and 10.5. The upper sphere with a radius of 1.5 centered at (0,0,3), and the lower one with a radius of 1 centered at (0.5,0,0). The two spheres are initially at the same temperature.

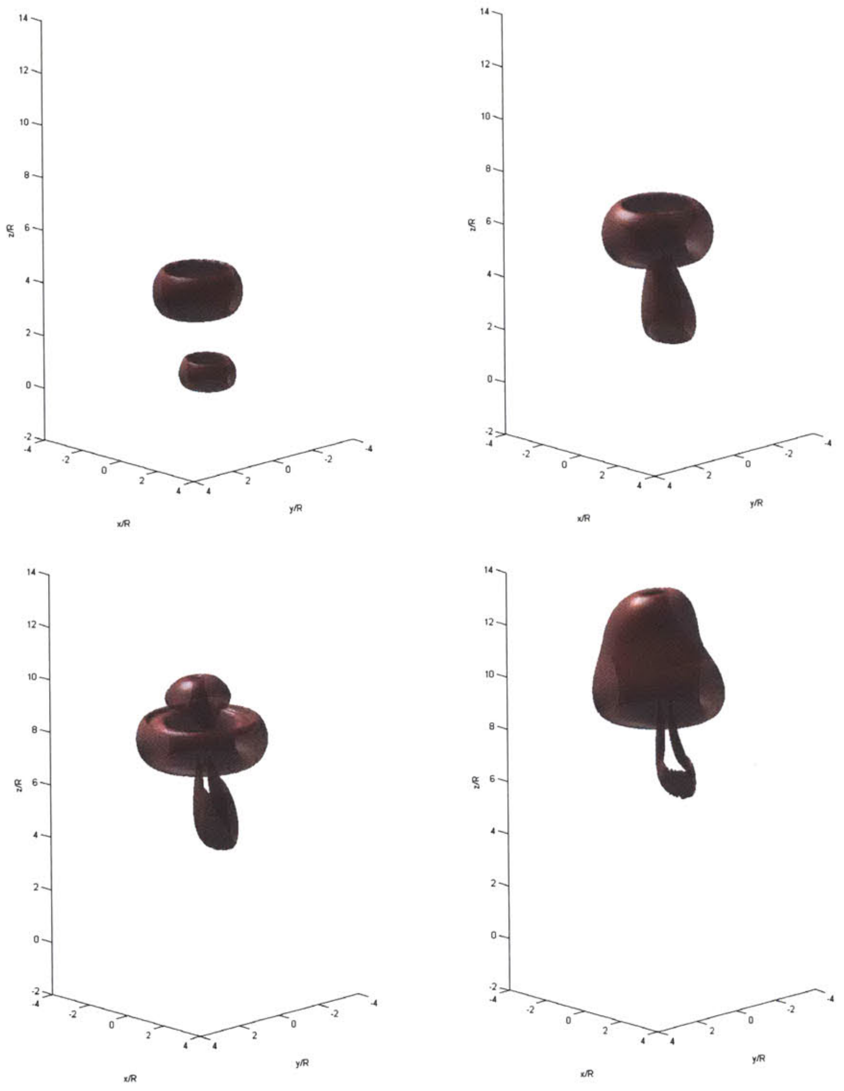

Figure 14: Evolution of two thermal spheres; vorticity isosurface, $|\boldsymbol{\omega}|=1.4$, for $\overline{t}$ =2.5, 5.5, 8.5 and 11.25.

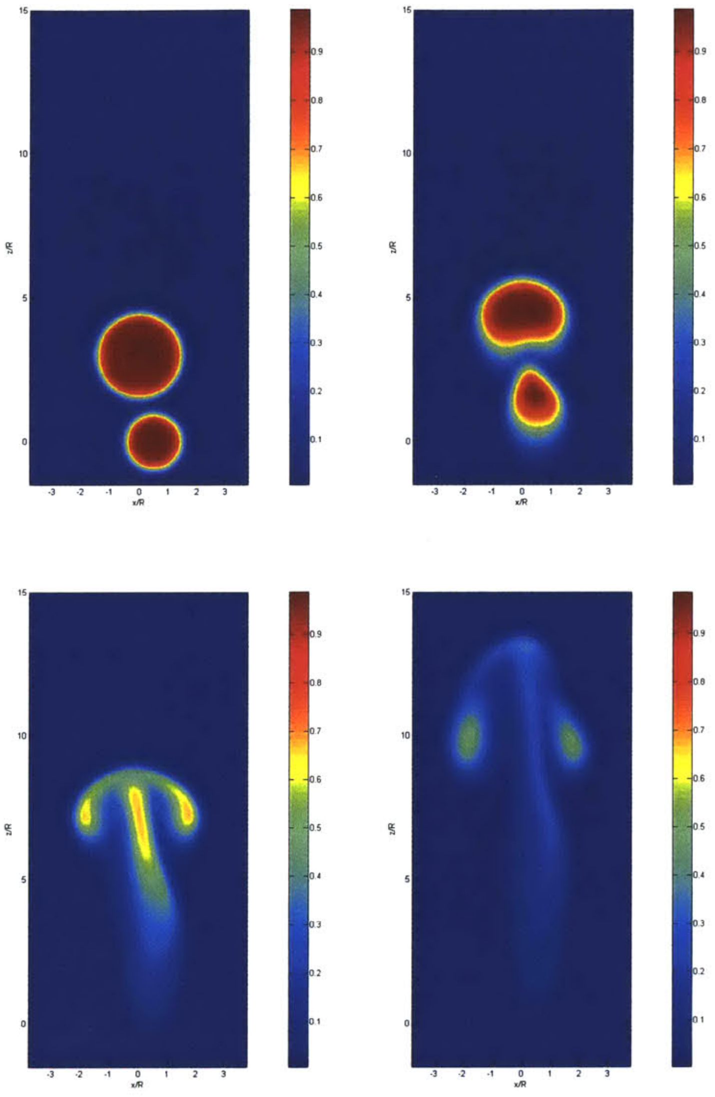

Figure 15: Evolution of two thermal spheres; temperature contours, for $\overline{t}$ =0, 3.5, 7 and 10.5. The upper bubble with a radius of 1.5 centered at (0,0,3), and the lower one with a radius of 1 centered at (0.5,0,0). The two bubbles are initially at the same temperature.

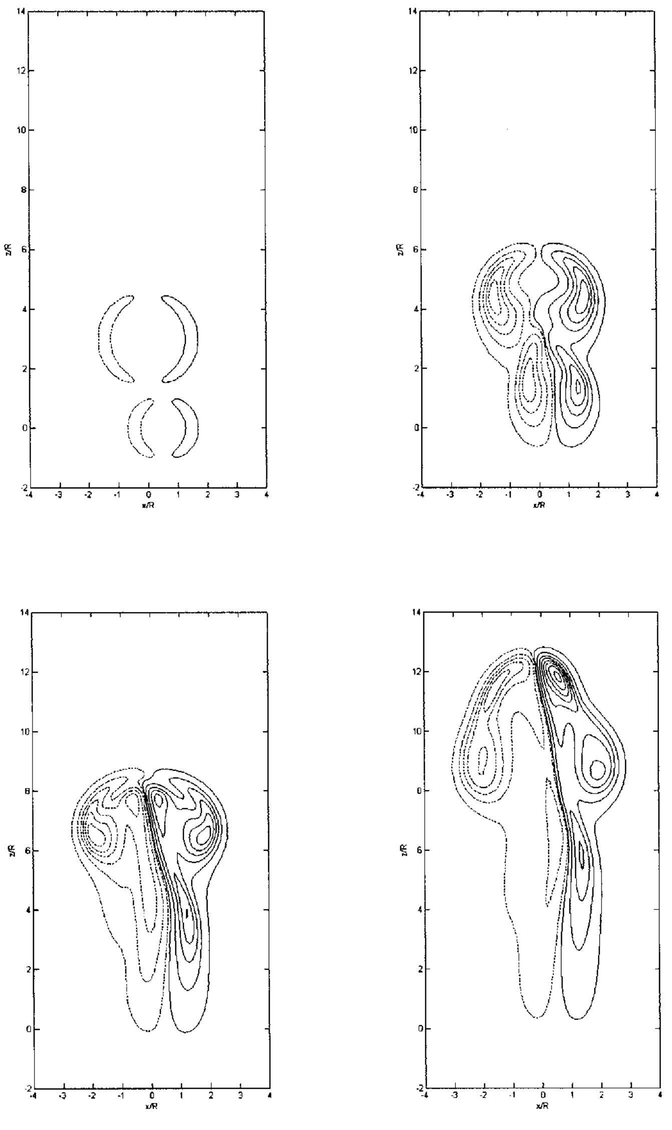

Figure 16: Evolution of two thermal spheres; vorticity contours for $\bar{t}$=0, 3.5, 7 and 10.5. The two first contour values are 0.1 and 0.5, then their values vary linearly with an increment of 0.5.

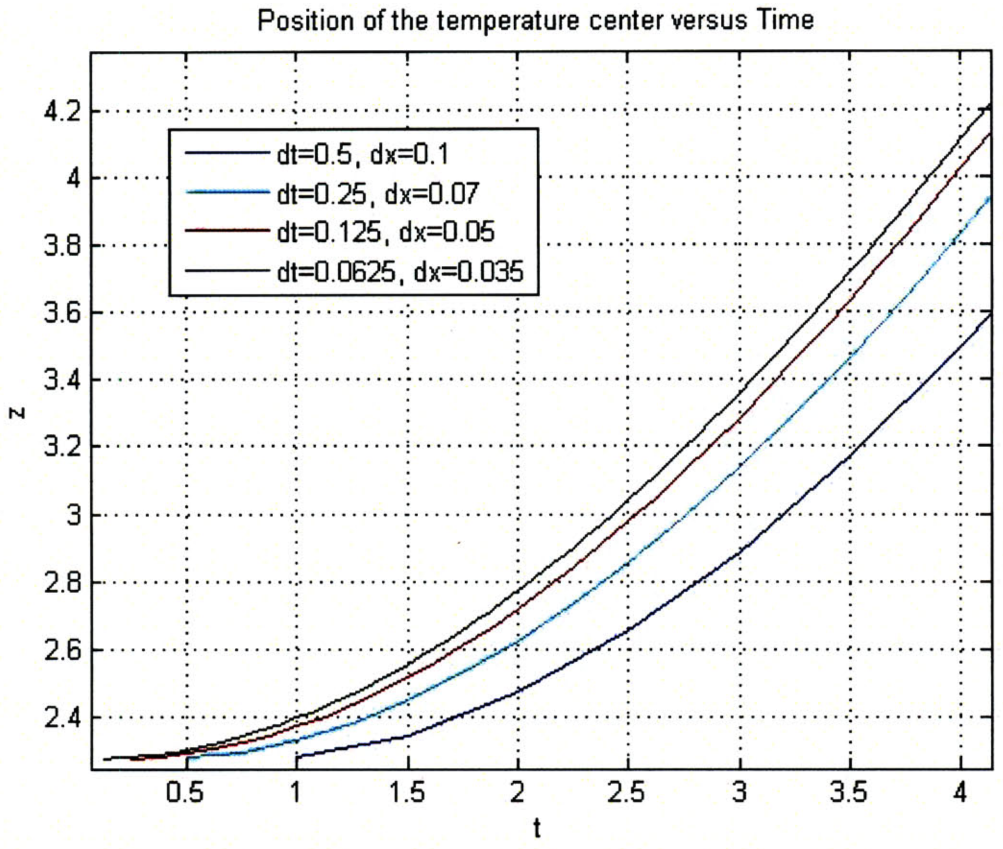

Figure 17: Position of the temperature center of the buoyant spheres for different resolutions.

2. Lagrangian Simulation of Combustion

Accurate and efficient computational algorithms for the simulation of high Reynolds number turbulent reacting flows with fast chemical reactions are valuable for the study of turbulence-combustion interactions in engineering systems utilized in automotive, aerospace and utility industries, as well as in problems related to safety and environmental concerns. In this section, a simulation of a diffusion-controlled combustion problem for Lewis number, $L_e = 1$, is developed using a Schwab-Zeldovich formulation [35]. This is an extension of our previous work that utilized the TEM with the Schwab-Zeldovich variable as transported scalar. The vorticity and energy equation are coupled through the density and its gradients. Buoyancy will play a major role since it is the only source of vorticity generation and the density change in time will be a source of volumetric expansion.

2.1. Formulation

The governing equations of the reacting flow problem are the conservation of mass, momentum, chemical species, energy, and the equation of state [35]. We apply the 'infinite chemistry rate' approximation, i.e., the fuel and the oxidizer never coexist and the burning rate is determined by the rate at which the fuel and the oxidizer mix by diffusion. In this case, a single conserved scalar can be defined,

$$s = \frac{\eta_i - \eta_{i,o}}{\eta_{i,f} - \eta_{i,o}} \tag{39}$$

where $\eta_{i,f}$ and $\eta_{i,o}$ are the value of η_i in the streams carrying the fuel and the oxidizer, and i=1, 2, with,

$$\eta_1 = Y_o - \varphi Y_f \tag{40}$$

and

$$\eta_2 = \theta - \frac{Q_f}{1+\varphi} Y_p \tag{41}$$

where Y_k is the mass fraction of a chemical species. The indices f, o and p are respectively for fuel, oxidizer and product. The variable θ represents the normalized temperature, φ the mass stoichiometry and Q_f the normalized enthalpy of reaction. Finally, if we assume that the Lewis number L_e=1, the normalized conserved scalar is governed by the following sourceless convection-diffusion equation,

$$\frac{\partial s}{\partial t} = \frac{1}{Pe} \Delta s \tag{42}$$

where $s \to 1$ correspond to the fuel side and $s \to 0$ correspond to the oxidizer side. The distribution of all the reacting species, the reaction front location and the temperature field can be recovered from this conserved scalar value.

The conserved scalar gradient is needed to compute the baroclinic generation of vorticity, therefore we follow the evolution of the gradient, ∇s, instead of the evolution of the scalar, s. Differentiating Eq.(42), this yields,

$$\frac{D(\nabla s)}{Dt} = -(\nabla u)^{T} \cdot (\nabla s) + \nabla s (\nabla . u) + \alpha \Delta (\nabla s) \tag{43}$$

The Prandtl number is unity, i.e., the viscous diffusion rate ν equal the thermal diffusion rate α and $P_e = R_e P_r = R_e = \frac{VR}{\nu} = \frac{\sqrt{gR}R}{\nu}$. With R=1 and the normalized gravity, $g = 1$, we obtain $P_e = \frac{1}{\alpha}$.

Propagating the gradient, we do not loose accuracy or CPU time in differentiation, and the code remains fully Lagrangian. The value of s is recovered using our multi-purpose treecode

$$s(\mathbf{x},t) = \sum_{i=1}^{n} (\nabla s(t)_i dV_i) \cdot \nabla G_\delta (\mathbf{x} - \mathbf{x}_i(t)) \tag{44}$$

using the Green's function,

$$\nabla G_\delta (\mathbf{x}) = \frac{(x,y,z)}{4\pi (r^2 + d^2)^{\frac{3}{2}}}. \tag{45}$$

Once the scalar field is recovered, we compute the distribution of all the reacting species (fuel, product, oxidizer and diluents), the temperature and the density field. For a computational point located on the oxidizer side, we have $\eta_1 > 0$, and the following equations apply:

$$Y_o = \eta_1; \quad Y_f = 0; \quad \frac{dY_o}{ds} = \eta_{1,f} - \eta_{1,o}; \quad \frac{dY_f}{ds} = 0$$

Else, if it is located on the fuel side, we have $\eta_1 < 0$, and the following equations apply:

$$Y_o = 0; \quad Y_f = -\frac{\eta_1}{\varphi}; \quad \frac{dY_o}{ds} = 0; \quad \frac{dY_f}{ds} = \frac{\eta_{1,o} - \eta_{1,f}}{\varphi}$$

Then we compute the mass fraction of diluent,

$$Y_d = Y_{d,o} + (Y_{d,f} - Y_{d,o}) * s$$

the mass fraction of product,

$$Y_p = 1 - Y_o - Y_f - Y_d$$

the temperature,

$$T = \eta_2 + \frac{Q_f}{(1+\varphi)} Y_p$$

the derivatives of the mass fractions,

$$\frac{dY_d}{ds} = Y_{d,f} - Y_{d,o}$$

$$\frac{dY_p}{ds} = -\frac{dY_o}{ds} - \frac{dY_f}{ds} - \frac{dY_d}{ds}$$

the derivative of the temperature by respect to the scalar value

$$\frac{dT}{ds} = \eta_{2,f} - \eta_{2,o} + \frac{Q_f}{(1+\varphi)} \frac{dY_d}{ds}$$

the average molar mass of the mixture M_i

$$M_i = \frac{Y_o}{W_o} + \frac{Y_f}{W_f} + \frac{Y_d}{W_d} + \frac{Y_p}{W_p}$$

the density,

$$\rho = \frac{1}{T * M_i}$$

and the quantity $\frac{1}{\rho}\frac{d\rho}{ds}$ necessary to the computation of the baroclinic generation of vorticity

$$\frac{1}{\rho}\frac{d\rho}{ds} = -(\frac{1}{T}\frac{dt}{ds} + \frac{1}{M_i}(\frac{1}{W_o}\frac{dY_o}{ds} + \frac{1}{W_f}\frac{dY_f}{ds} + \frac{1}{W_d}\frac{dY_d}{ds} + \frac{1}{W_p}\frac{dY_p}{ds})).$$

2.2. Algorithmic Description

As in the buoyant case, the present solutions are obtained through an operator splitting. The computational time step is split in three substeps, i.e., convection substep, generation substep, and diffusion substep. However, the combustion problem requires the additional knowledge of the Lagrangian derivatives of the density and the velocity field, $\frac{D\rho}{Dt}$ and $\frac{Du}{Dt}$ respectively. Since diffusion is done on a regular grid, the path of the particles is broken and the computation of these derivatives becomes tricky. The algorithm used to compute these three substeps, as well as these Lagrangian derivatives, is explained in details in the following sections.

2.2.1. Convection

The following equations are solved using a second order Predictor/Corrector scheme,

$$\frac{D\omega}{Dt} + \omega(\nabla . u) = \omega . \nabla u \tag{46}$$

$$\frac{D(\nabla s)}{Dt} = -(\nabla u)^{T} \cdot (\nabla s) + \nabla s(\nabla . u) \tag{47}$$

The velocity field **u** has two components: vortical velocity from the vorticity field and expansion velocity from the non-trivial divergence or expansion source. Each of them can be recovered from the data provided by the previous diffusion/generation substep by using our multi-purpose adaptive tree-code. The vortical velocity in $\mathbf{R}^3$ is determined using the Biot-Savart law, Eq.(9), and the expansion velocity and its gradient are obtained from the mass conservation $\nabla . u = \varepsilon$, with $\varepsilon = \frac{-1}{\rho}\frac{D\rho}{Dt}$. They are computed using the treecode algorithm as follow,

$$\mathbf{u}_{\exp}(x) = \int \mathbf{K}_{\delta}(\mathrm{x},\mathrm{x}')\varepsilon(\mathrm{x}) \tag{48}$$

and

$$\nabla \mathbf{u}_{\exp}(x) = \int \nabla_{\mathrm{x}} \mathbf{K}_{\delta}(\mathrm{x},\mathrm{x}')\varepsilon(\mathrm{x}) . \tag{49}$$

The variable $\varepsilon = \frac{-1}{\rho}\frac{D\rho}{Dt}$ is computed using the fact that the Lagrangian derivative of the density, $\frac{D\rho}{Dt}$, is zero during convection and, consequently, non-zero only during the diffusion/generation substep. Thus, the density ρ is evaluated before and after the diffusion/generation substep at the scalar elements location after the diffusion substep; the numerical differentiation in time of these two values give us $\frac{D\rho}{Dt}$.

2.2.2. Diffusion/Generation

In this substep, the following equations are solved

$$\frac{\partial \omega}{\partial t} = \nu \Delta \omega + \frac{\nabla \rho}{\rho} \times (g - \frac{Du}{Dt}) \tag{50}$$

$$\frac{\partial (\nabla s)}{\partial t} = a \Delta (\nabla s) \tag{51}$$

We start by diffusing both vortex and scalar elements

$$\frac{\partial \omega}{\partial t} = \nu \Delta \omega \tag{52}$$

$$\frac{\partial (\nabla s)}{\partial t} = a \Delta (\nabla s) \tag{53}$$

Then, we need to compute the Lagrangian derivative of the velocity before the generation step. Since the vorticity is diffused on a Cartesian grid during the diffusion substep, we first need to calculate the elements velocity at their initial positions, before the convection substep, then we advect and diffuse the particles, and we compute the new velocity at the particle location that was saved before the diffusion. That way we follow the particle path and the differentiation in time of the two values give us the Lagrangian derivative of the velocity $\frac{Du}{Dt}$.

Finally, we generate new vortex elements through baroclinic generation of vorticity

$$\frac{d(\omega dV)_i}{dt} = \frac{1}{\rho} (\frac{\partial \rho}{\partial s}) (\nabla s dV)_i \times (g - \frac{Du}{Dt}) \tag{54}$$

where the variables ρ and $\frac{\partial \rho}{\partial s}$ are functions of s. The value of s is recovered from ∇s at scalar particle locations using our treecode algorithm. Subsequently, baroclinic vorticity is evaluated at the scalar particles locations and the vorticity field is updated by adding these new vortex elements to the existing vortex elements.

The diffusion scheme is the same as the one used previously for the simulation of buoyant flows.

2.2.3. Prediction/Correction Type Approach

We could have stopped here and simply apply the next convection substep, but we are left with vortices at both the location of the vortex particles (existing vortex elements) and the location of the scalar particles (new vortex elements obtained through baroclinic generation). To reduce the number of vortex elements, we diffuse the vorticity that has just been generated from the scalar field during the baroclinic generation substep, and the

vorticity coming from the vortex elements field before the diffusion step. This gives us a new vorticity field on the grid. This field will be used in the next convection substep.
In the buoyant jet-in-crossflow calculation, this problem is solved by having only one kind of element, carrying both the vorticity and the scalar gradient information. This way, the generation step does not augment the number of computational elements.

2.3. Results

2.3.1. Methane Ring

This numerical experiment is initialized with a axisymmetric methane ring (CH_4), with R=1. The stoichiometric chemical reaction corresponding to the complete combustion of hydrogen H_2 and carbon C, is

$$CH_4 + 2(O_2 + 3.76N_2) \rightarrow CO_2 + 2H_2O + 7.52N_2 \qquad (55)$$

The mass stoichiometry for this simulation is φ=4, the normalized enthalpy of reaction is $Q_f = 80$, and the Reynolds number is $R_e = 100$. The initial distribution of fuel, oxidizer, product and diluents concentrations is given in Figure 18. The mixture fraction varies from zero on the air side to one on the fuel side. The boundary conditions are:

On the fuel side: $Y_f = 1$, $Y_o = 0$, $Y_p = 0$ and $Yd = 0$.

On the air side: $Y_f = 0$, $Y_o = 0.2$, $Y_p = 0$ and $Yd = 0.8$.

The molar masses are $M_f = 16$, $M_o = 32$, $M_d = 28$ and $M_p = 26.62$. The initial vorticity is zero and the initial non-dimensional temperature is θ=1 on both the fuel side and the air side and the initial thickness of the Gaussian for s is $\delta_s = 0.2$. The core radius for the vorticity distribution is $\delta = 0.1$, the grid size for diffusion is $\Delta x = 0.05$ and the time step is $\Delta t = 0.125$.

Temperature and vorticity contours are plotted in Figure 19. The temperature, as well as the vorticity produced through baroclinic generation, is first symmetric around the r/R=1 axis. At this point, the flow is dominated by diffusion, not convection. Due to diffusion, the vorticity spreads across the z-axis and dissipates by inter-diffusion. The negative vorticity $(r/R \leq 0)$ becomes less important in absolute value than the positive vorticity and the structure rises and rotates clockwise. This behavior is similar to the one observed in 1.4.2.1 in the two side-by-side buoyant spheres problem when the two inner cores connect, their vorticity dissipates by inter-diffusion and the strength of the inner cores becomes weaker that those associated with the outer core.

The contours of the conserved scalar s value and the distribution of fuel, oxidizer, product and diluents concentrations are plotted in Figure 20. The distribution of the concentrations is plotted for an arbitrary line going through the centroid of the scalar field. The burning rate is determined by the rate at which the fuel and the oxidizer mix by diffusion. Most of the fuel is consumed at the early stage; thus, no chemical reaction occurs afterwards and the temperature distribution is only governed by diffusion and convection. The convection is due to the vorticity and volumetric expansion. The impact of volumetric expansion can be seen through the fact that the mean center of temperature is moving away from the axisymmetry axis during the simulation (Figure 19). Finally, Figure 19 and Figure 20 show that the regions of large vorticity correspond to the one with high scalar gradients, as expected.

The results of this simulation for R_e=50 were compared to those obtained by Lakkis [35] for the same simulation to qualitatively validate the code. The following results are obtained at a Reynolds number $R_e = 100$.

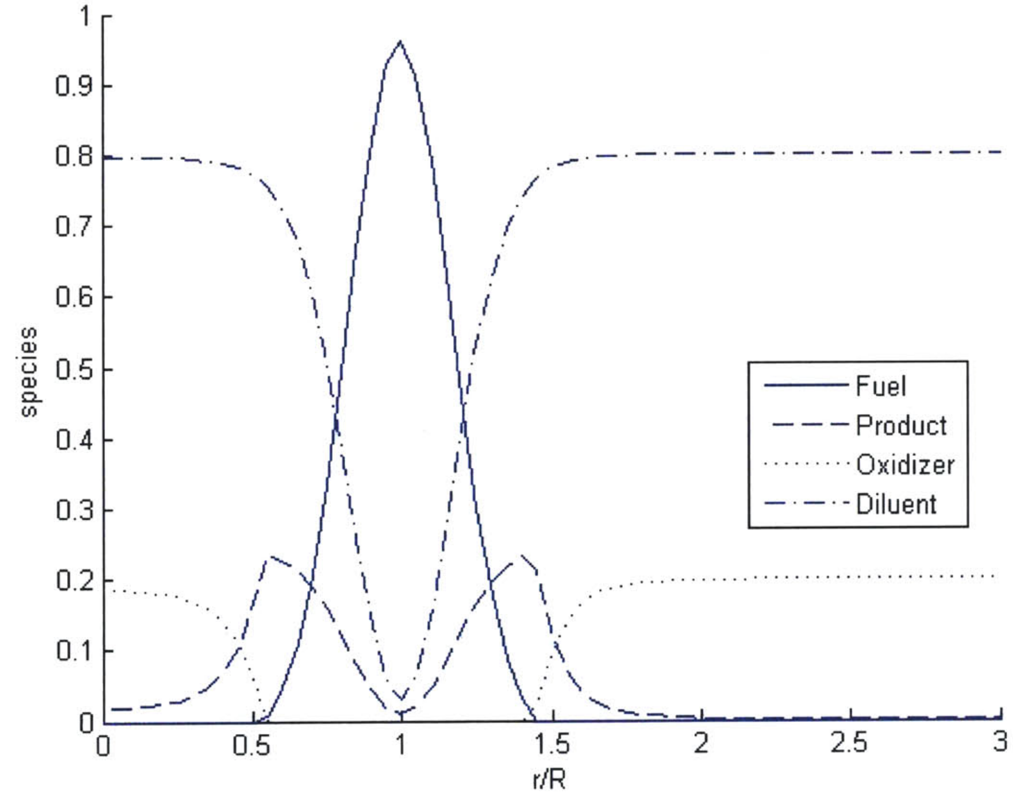

Figure 18: Initial distribution of fuel, oxidizer, product and diluents concentrations for the reacting flow problem.

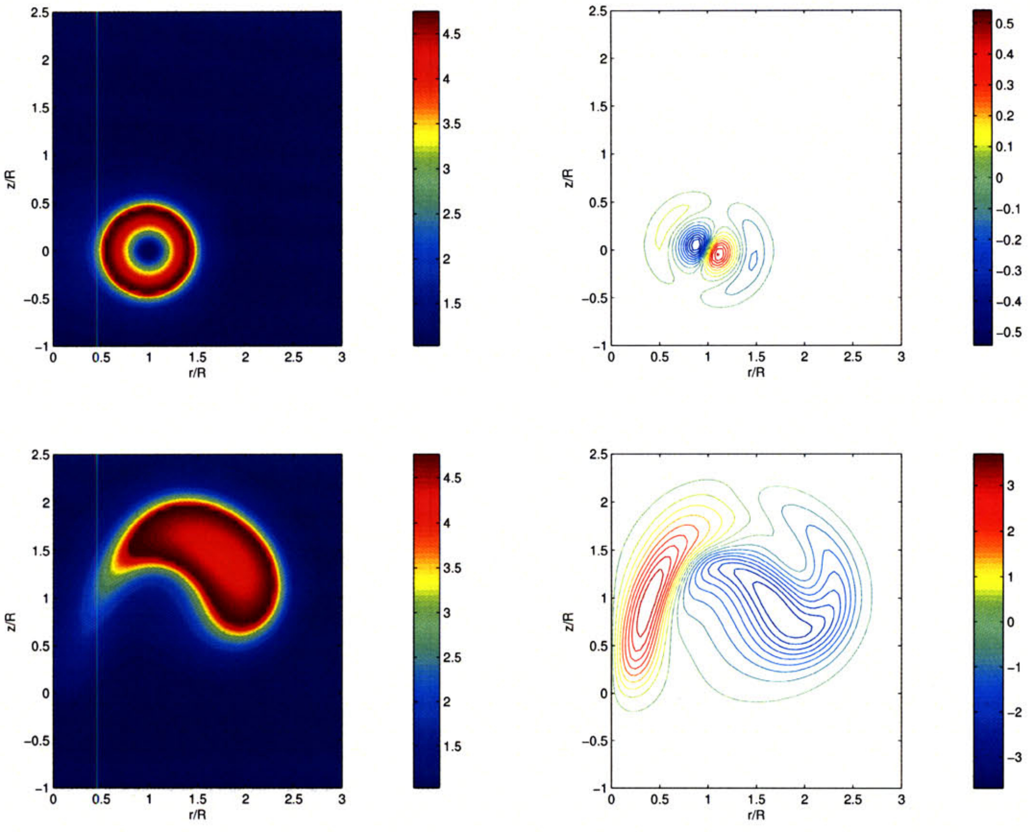
Figure 19: Evolution of a methane ring with a Gaussian distribution of fuel for the times $\bar{t}$=0 and 3.75; temperature contours are plotted on the left and vorticity contours are plotted on the right.

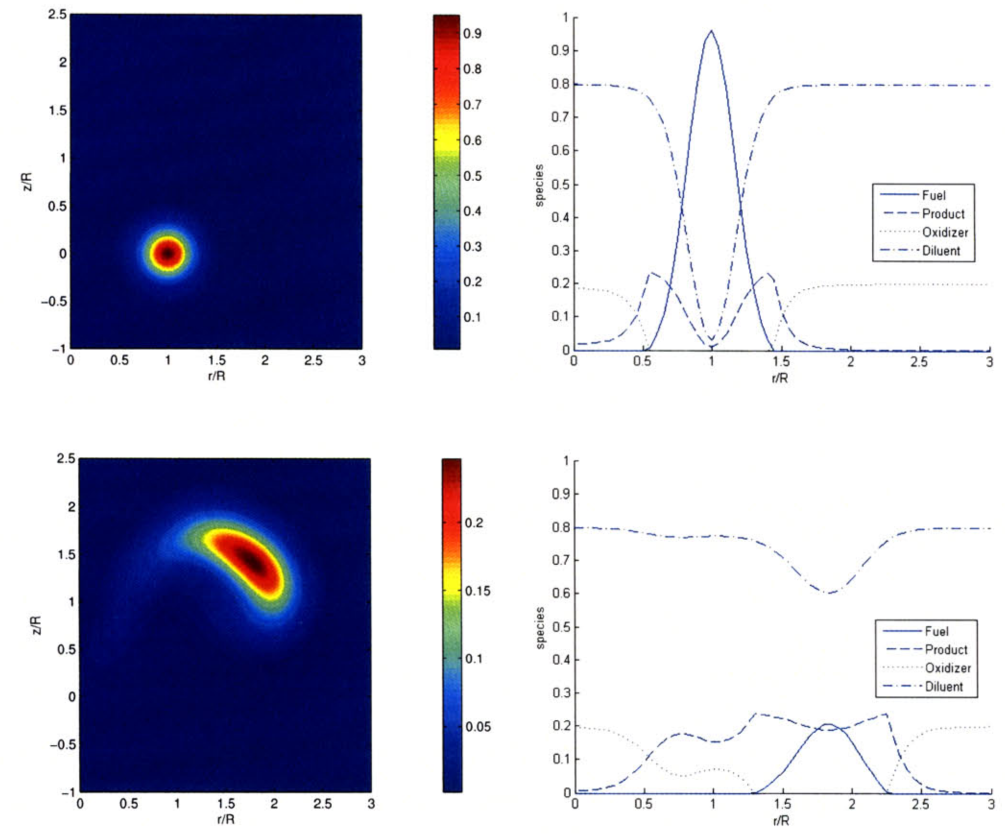

Figure 20: Evolution of a methane ring with a Gaussian distribution of fuel for the times $\bar{t}$=0 and 3.75; the conserved scalar value s is plotted on the left, and the distribution of fuel, oxidizer, product and diluents concentrations is plotted on the right.

2.3.2. Methane Ring with a Gaussian Distribution of Fuel

In this section, the evolution of a methane sphere (CH_4) with a Gaussian distribution of fuel is presented. The stoichiometric chemical reaction is shown Eq.(55). Again, the mass stoichiometry for this simulation is φ=4 and the normalized enthalpy of reaction is $Q_f = 80$. The initial distribution of fuel, oxidizer, product and diluents concentrations is given in [35]. The mixture fraction varies from zero on the air side to one on the fuel side. The boundary conditions are:

On the fuel side: $Y_f = 1$, $Y_o = 0$, $Y_p = 0$ and $Yd = 0$.

On the air side: $Y_f = 0$, $Y_o = 0.2$, $Y_p = 0$ and $Yd = 0.8$.

The molar masses are $M_f = 16$, $M_o = 32$, $M_d = 28$ and $M_p = 26.62$. The initial vorticity is zero and the initial non-dimensional temperature is θ=1 on both the fuel side and the air side and the initial thickness of the Gaussian for s is $\delta_s = \frac{1}{3}$. The core radius for the vorticity distribution is $\delta = 0.1$, the grid size for diffusion is $\Delta x = 0.05$ and the time step is $\Delta t = 0.125$. The following results are obtained at a Reynolds number $R_e = 100$.

The behavior of the sphere is similar to the one observed in 1.4.1 for the evolution of the buoyant sphere. In addition to buoyancy major role, the effects of volumetric expansion coming from the density change in time can also be observed in this problem. Temperature and vorticity contours are plotted in Figure 21. The temperature and the vorticity produced through baroclinic generation are axisymmetric around the z-axis. The flow is first dominated by diffusion. Most of the fuel is consumed at the early stage; thus the temperature distribution is then mostly governed by diffusion and convection. The contours of the conserved scalar s value and the distribution of fuel, oxidizer, product and diluents concentrations are plotted in Figure 22. Again, the distribution of the concentrations is plotted for an arbitrary line going through the centroid of the scalar field. The impact of volumetric expansion can be seen through the fact that the center of temperature is moving away from the axisymmetry axis.

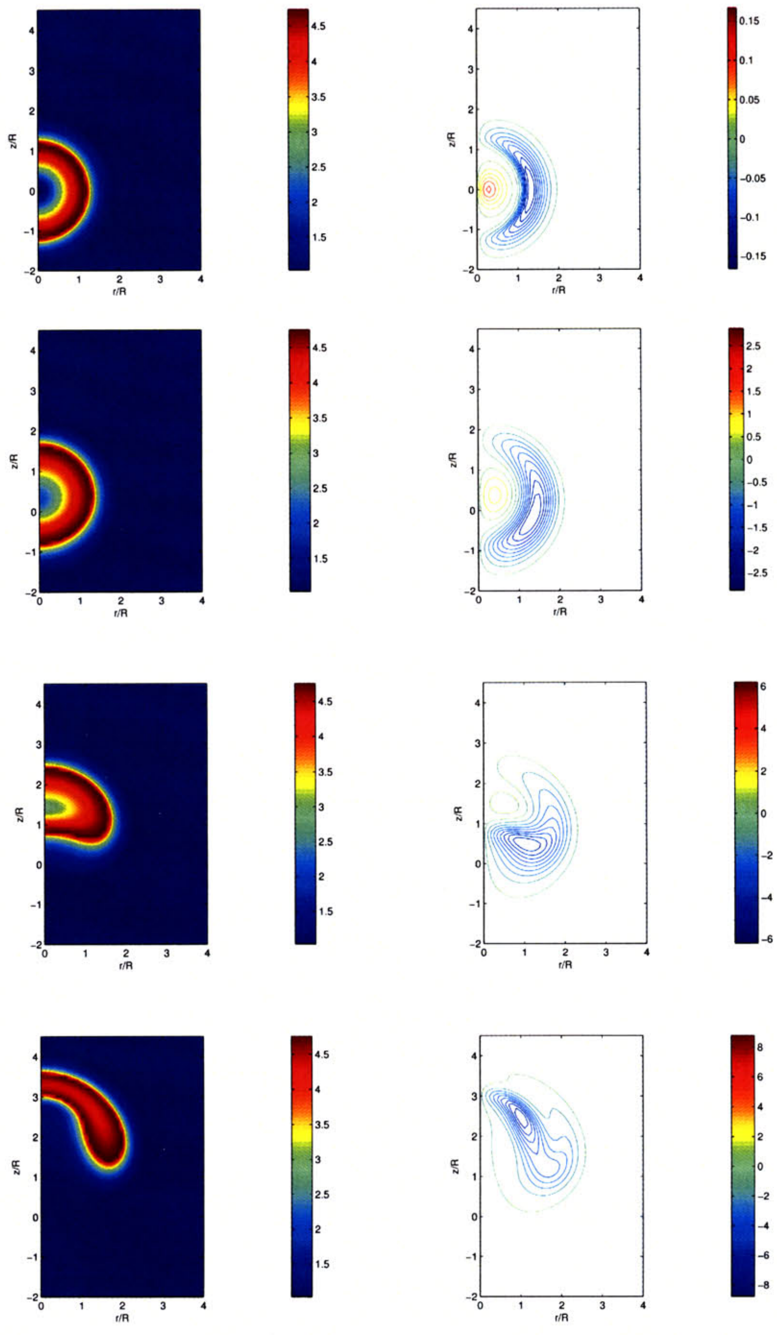

Figure 21: Evolution of a methane sphere with a Gaussian distribution of fuel for the times $\bar{t}$=0, 0.75, 1.5 and 2.75; temperature contours are plotted on the left and vorticity contours are plotted on the right.

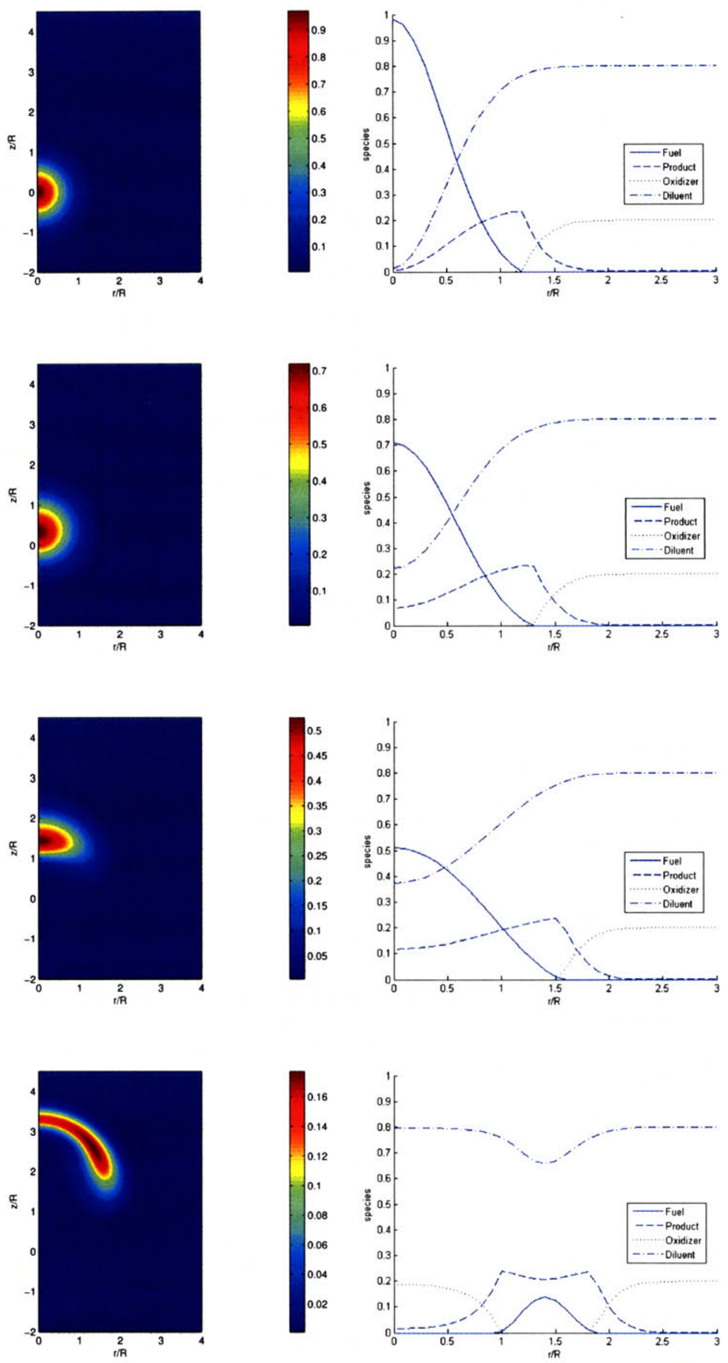

Figure 22: Evolution of a methane sphere with a Gaussian distribution of fuel for the times $\bar{t}$=0, 0.75, 1.5 and 2.75; the conserved scalar value s is plotted on the left, and the distribution of fuel, oxidizer, product and diluent concentrations is plotted on the right.

2.4. Conclusion

Our algorithm is efficient enough to reach the convergence in the case of the methane sphere with a Gaussian distribution of fuel, but it needs to be parallelized and modified for the sharp interface case to be able to perform more accurate simulations. In fact, simulations of the evolution of a methane sphere with a sharp interface between the fuel side and the oxidizer have been performed for $R_e = 100$, but we were not able to reach the convergence for this case. This issue can be addressed using the same strategy as the one described for the jet problem. In fact, the buoyant jet-in-crossflow algorithm is implemented on a parallel distributed memory computer using MPI and it contains only one kind of element, carrying both the vorticity and the scalar gradient information. This way, the CPU time decreases with the number of processors used for the simulation, the generation step does not augment the number of computational elements and more importantly, the number of particles that is advected at every time step is divided by two.

3. Application to the Buoyant Jet Problem

Transverse jets do not only represent a canonical example of a flow composed of multiple coherent vortical structures, their mixing properties are also very important to a variety of industrial applications, e.g., fuel sources in industrial furnaces or as primary or dilution air jets in gas turbine (Figure 23). They have also been studied for environmental problems such as pollutant dispersion from chimneys or the discharge of effluents into the ocean.

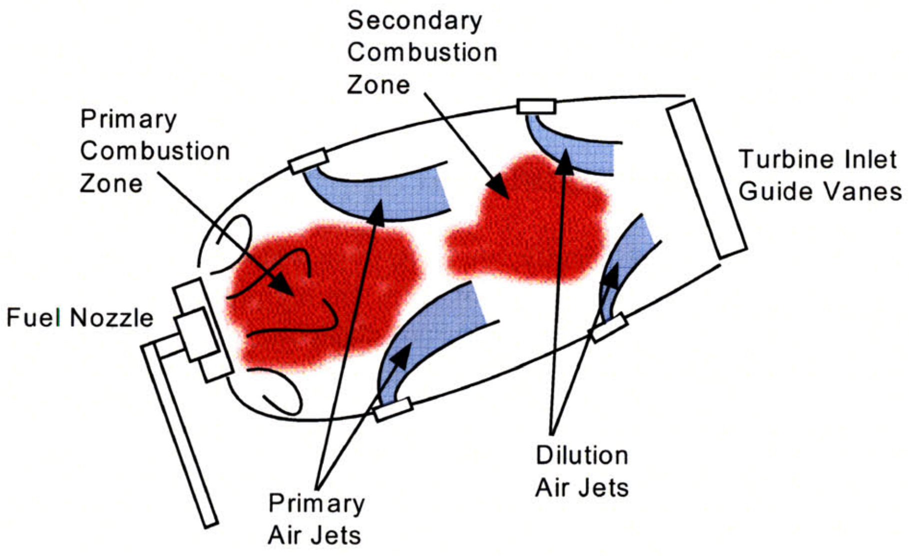

Figure 23: Mixing in gas turbines.

Previous numerical investigation [15] has focused on elucidating the mechanisms underlying the formation of organized vortical structures in the near field and the subsequent breakdown of these structures into small scales. Here, we will extend our previous work on thermals to the buoyant transverse jet. Our aim is now to understand how buoyancy will affect the vortical structure.

In this chapter, two major modifications made to the previous TEM code will also be explained. First, an efficient strategy for its parallelization has been implemented; also, the new algorithm contains only one kind of element, carrying both the vorticity and the scalar gradient information.

3.1. Numerical Formulation

3.1.1. Parallel Implementation

All the previous simulations on thermals were performed in serial on a single processor since we were able to get results in a reasonable amount of time, generally one or two days. The present computations however are implemented on a parallel shared or distributed memory computer using MPI, the standard message passing libraries. In fact, a typical simulation contains around 5 millions vortex elements. The domain decomposition is achieved through clustering algorithms, as well as heuristic methods for dynamic load balancing, as described in [34], in which an efficiency of 98% on 1024 processors was observed on a realistic distribution of 1.2 millions of vortex particles. The clustering provides us a partition of the source particles and then a local oct-tree is constructed in each cluster, the number of clusters corresponding to the number of processors. The output produced by each processor, the vortical velocity for instance, is summed at each target.

3.1.1.1. Computational Approach

To parallelize the algorithm efficiently, we have to provide an equal workload for each processor and also avoid the duplication of work among processors. Thus, the domain decomposition will be achieved through k-means clustering algorithms. K-means takes a set of N observations $\{x_i\}$ in d-dimensional space as input and partitions the set into k clusters with centroids $\{ y_1, \ldots, y_k \}$, where k is prescribed.

The partition is chosen to minimize the cost function

$$J = \sum_{i=1}^{N} \min_{k'}(| x_i - y_{k'} |^2 \omega_i) = \sum_{j=1}^{k} \sum_{i}^{N_j} | x_i - y_{k'} |^2 \omega_i ,$$

where ω_i represent the vorticity magnitude. This way, each particle is assigned to the nearest centroid and the centroid positions are chosen to minimize the weighted within-cluster sum of the squared Euclidean distances. A complete description of the algorithm can be found in [34].

Although each processor has a copy of all the particles in memory, this is not necessary when using TEM. In fact, contrary to Vortex Filament Methods, there is no need to preserve an ordering or connectivity between neighboring elements.

3.1.1.2. Dynamic load balancing

The time required to compute target velocities depends on many parameters, like the distance between the cluster centroid and the target particle. The closer they are, the longest it will take to evaluate the velocities (the ratio between direct summation and Taylor approximation in the multi-purpose treecode will be bigger). Thus, equipartition of particles in each domain does not ensure load balance. To ensure a good load balance, we use three heuristic methods found in [34]. The first one entails the introduction of a scaling factor s_k into the weighted k-means cost function

$$J = \sum_{i=1}^{N} \min_{k'}(s_{k'} \,|\, x_i - y_{k'} \,|^2 \, \omega_i)$$

where the scale factor $s_{k'}$ is updated based on each cluster deviation from the mean source evaluation time $\bar{t}^n = \sum_k t_n^k / k$,

$$s_{k'}^{n+1} = s_{k'}^n (1 + \alpha \tanh(\beta \frac{t_n^k - \bar{t}^n}{\bar{t}^n})).$$

In other words, each particle is assigned to the centroid from which the scaled distance is the smallest.

The second heuristic method consists in splitting the highest-cost cluster in two at the end of each time step. The two resulting converged centroids are used as initial guesses for the next clustering iteration. To keep the same number of clusters, the lowest-cost cluster is deleted at each time step.

Finally, a last heuristic method consists in reseeding the cluster centroid if the load imbalance, defined as $(\max_k t_k)/\bar{t}$, is bigger than a user-defined parameter, typically 1.5.

3.1.2. Algorithm Optimization

Our new algorithm using the TEM contains only one set of element, carrying both the vorticity and the scalar gradient information. In fact, during the convection substep, the solution of Eq.(14) and Eq.(15) is obtained by integrating Eq.(19), Eq.(20), Eq.(21) and Eq.(22). Using only one kind of particle, Eq.(20) becomes redundant and the velocity gradient do not need to be computed at two different particle locations to solve Eq.(21) and Eq.(22). Velocity gradients are now computed once at a single location and both equations are solved. Thus, the number of particles that are advected at every time step is divided by two approximately, for Prandtl number $P_r \approx 1$.

Moreover, additional vorticity is introduced where nontrivial baroclinicity exists during the generation step. This was previously achieved by generating one new vortex element at the location of each transport element. Once the update was finished, the number of vortex elements N_ω was augmented by the number of transport elements N_g. In our new algorithm, scalar information is simply converted into vorticity information for each particle. Thus, the number of computational elements does not augment during the generation step.

3.1.3. Boundary Conditions

In the present computations, the jet is aligned with the y axis and the crossflow, $U_\infty(y) = 1$ for $y \geq 0$, is directed in the positive x direction; the z axis is the spanwise direction. The x-z plane is taken to be a solid wall through which we enforce a no normal-flow boundary condition, i.e., the wall is impermeable, $\vec{u}(x, y_w, t).\vec{n} = 0$, except for the disc of the jet orifice. We call d the jet diameter and r the velocity ratio between the jet and the crossflow.

3.1.3.1. Wall

The effect of the wall is taken care of by the image method, with

$$\begin{cases} x_{image} = \ x \\ y_{image} = -y \\ z_{image} = \ z \end{cases} \text{ and } \begin{cases} \vec{\omega}.\vec{x}_{image} = -\vec{\omega}.\vec{x} \\ \vec{\omega}.\vec{y}_{image} = \ \vec{\omega}.\vec{y} \\ \vec{\omega}.\vec{z}_{image} = -\vec{\omega}.\vec{z} \end{cases}$$

Each vortex has an image vortex placed on the other side of the wall, which participates in the velocity and velocity gradient integration. The effect of these image vortices is that of an inviscid wall.

The fact that the velocity normal to the wall is zero implies that no elements should cross it. However, this can happen during the diffusion, or even during the convection, due to the discrete nature of the simulation. To avoid this problem, vortex elements that cross the walls are reflected in the domain. This yields $y_{new} = -y_{cross}$ and $\vec{\omega}.\vec{y}_{new} = -\vec{\omega}.\vec{y}_{cross}$. In the jet calculation, the same elements are used for all the transported properties. Thus, the crossing of a transport element is taken care of in a similar fashion to that of the vortex elements, they are simply reflected. This yields $y_{new} = -y_{cross}$ and $\nabla T.\vec{y}_{new} = \nabla T.\vec{y}_{cross}$.

When post-processing the data in order to recover the temperature field from the gradients, the image method is used, with

$$\begin{cases} x_{image} = \ x \\ y_{image} = -y \\ z_{image} = \ z \end{cases} \text{ and } \begin{cases} \nabla T.\vec{x}_{image} = \ \nabla T.\vec{x} \\ \nabla T.\vec{y}_{image} = -\nabla T.\vec{y} \\ \nabla T.\vec{z}_{image} = \ \nabla T.\vec{z} \end{cases}$$

Each gradient has an image placed on the other side of the wall, which participates in the temperature integration. The image method prescribes the temperature at the jet exit to unity.

3.1.3.2. Symmetry Plane

Due to the symmetric nature of the problem, the simulation is performed in the z-negative domain. The algorithm starts by computing the influence of the particles in this domain on themselves, then it computes the influence of the particles across the z=0 symmetry plane on the previous one, with:

$$\begin{cases} x_{image} = x \\ y_{image} = y \\ z_{image} = -z \end{cases}, \quad \begin{cases} \vec{\omega}.\vec{x}_{image} = -\vec{\omega}.\vec{x} \\ \vec{\omega}.\vec{y}_{image} = -\vec{\omega}.\vec{y} \\ \vec{\omega}.\vec{z}_{image} = \vec{\omega}.\vec{z} \end{cases} \text{ and } \begin{cases} \nabla T.\vec{x}_{image} = \nabla T.\vec{x} \\ \nabla T.\vec{y}_{image} = \nabla T.\vec{y} \\ \nabla T.\vec{z}_{image} = -\nabla T.\vec{z} \end{cases}$$

The contributions of the computational elements on themselves, the virtual particles across the symmetry plane, their images and the jet are added before solving Eq.(19), Eq.(21) and Eq.(22).

3.1.3.3. Jet Outflow

3.1.3.3.1. Vorticity

The jet outflow is represented by a semi-infinite cylindrical vortex sheet of radius $R = 0.5$ extending from $y = 0$ to $y = -\infty$, with the strength $2r\hat{\mathbf{e}}_\theta$ [15]. The vorticity in this cylinder is mollified by the same core function as the one used for vortex elements. Numerically, vortex elements are placed on the cylinder every $\frac{r\Delta t}{2}$ in the negative y-direction, starting at $y = -\frac{r\Delta t}{4}$ and ending at y_{MAX}, a user-defined parameter, typically $y_{MAX} = -5$. These elements have weight [15]

$$(\omega dV) = -\frac{-r^2}{4}\Delta t \Delta\theta \hat{\mathbf{e}}_\theta$$

where $\hat{\mathbf{e}}_\theta$ is the tangential unit vector in the *x-z* plane and $\Delta\theta = \frac{2\pi}{n_\theta}$, n_θ being the number of vortex distributed along the circumference of the cylinder, i.e., the azimuthal discretization.

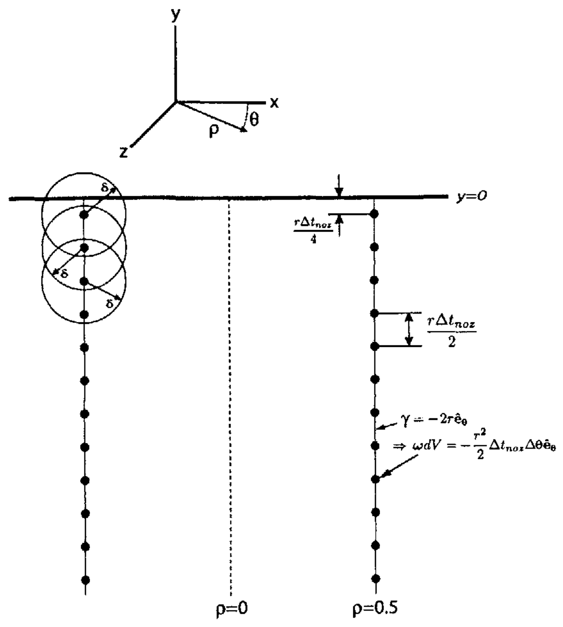
y
x
z
ρ
θ
δ
y=0
$\frac{r\Delta t_{noz}}{4}$
$\frac{r\Delta t_{noz}}{2}$
$\gamma = -2r\hat{e}_\theta$
$\Rightarrow \omega dV = -\frac{r^2}{2}\Delta t_{noz}\Delta\theta\hat{e}_\theta$
ρ=0
ρ=0.5

Figure 24: Discretization of the cylindrical vortex sheet representing the jet ouflow [15].

Also, to compensate for the vorticity produced in the jet boundary layer at the nozzle exit, a single vortex sheet is introduced in the flow at every time step at $y = \frac{r\Delta t}{4}$ with the strength

$$
\begin{aligned}
(\omega dV) &= \left(-\frac{r^2}{4} - \frac{r}{4}\cos(\theta) \right) \Delta t \Delta\theta \hat{\mathbf{e}}_\theta \\
&+ \left(\frac{r}{4}\sin(\theta) - \frac{r^2 \Delta t}{8}\sin(\theta) \right) \Delta t \Delta\theta \hat{\mathbf{e}}_y
\end{aligned}
$$

A complete description of this vortex sheet can be found in [15].

3.1.3.3.2. Scalar Field

As for the boundary generation of vorticity, a single sheet of scalar gradients is introduced in the flow at every time step. We first tried a "first order" implementation with a single sheet of gradients introduced at $y = \frac{r\Delta t}{4}$ with the strength $(\nabla T dV) = -Rh\Delta\theta\hat{\mathbf{e}}_r$, where h represents the time step multiplied by the vertical velocity at the location of the particle. This implementation of the boundary conditions is not accurate enough. In fact, even with a small time step, the gradients will not be aligned in the negative $\hat{\mathbf{e}}_r$-direction, especially not at the early stage, when the jet is forming.

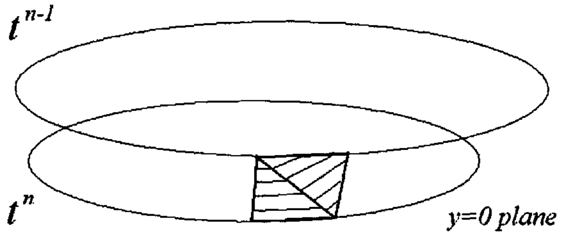

Figure 25: Insertion of gradient elements near the nozzle.

A much better implementation of these conditions is done by letting evolve a ring of Lagrangian points with *R*=0.5 placed in the *xz*-plane at *y*=0 during the previous time step. Thus, this ring is advected using the 2nd order Runge-Kutta scheme. As shown in Figure 25, the surface between the two rings is then discretized by $2n_\theta$ triangular elements. An element is placed at the barycenter of each triangle; its strength is given by the area δA of the triangle and its direction by the normal,

$$(\nabla T dV) = -(T_{Jet} - T_\infty)\delta A\vec{n}$$

with T_{Jet}=1 and T_∞=0, this yields

$$(\nabla T dV) = -\delta A\vec{n}$$

where $\vec{n}$ is the normal pointing outwards of the triangle surface.

3.2. Results

3.2.1. Numerical Parameters

Numerical parameters were chosen as follow: the core radius δ is chosen to be 0.1 and the number of elements discretizing vorticity and temperature gradients introduced along the nozzle, i.e. the azimuthal resolution, is n_θ=128. The axial resolution depends on the time step Δt. The product between the time step Δt and the jet velocity *r* is chosen to be less or equal to the grid size. For $r = 5$ we have $\Delta t = 0.01$ and $\Delta x = 0.035$. Thus, the grid size is 29 times smaller than the jet diameter. All the following simulations were performed at a Reynolds number $\mathrm{Re}_c = 300$ based on the crossflow, a Reynolds number $\mathrm{Re}_{Jet} = 1500$ based on the jet velocity, and a Grashof number such that $Gr/\mathrm{Re}_c^2 = 5$, and, consequently, $Gr/\mathrm{Re}_{Jet}^2 = 1$. Because of diffusion, the vorticity support expands and the number of particles grows in time. To control the number of particles, particles with its strength below a cutoff value are deleted after each diffusion substep. The cutoff value for deletion is chosen to be $|\omega dV|_{del} = 10^{-8}$ and $|\nabla T dV|_{del} = 10^{-8}$. The number of elements at the end of the simulation is around 500,000 for the non-buoyant case and 750,000 for the buoyant case.

3.2.2. Passive Scalar Transport

Figure 26, 27 and 28 show the vorticity isosurfaces, $|\omega| = 15$, of the non-buoyant transverse jet of initial radius R=0.5 and velocity ratio r=5, at $\bar{t}$=3. The jet is contoured by spanwise vorticity ω_z. These figures clearly show the development of ring-like vortical structures, the Kelvin-Helmholtz's instability, and the formation of two counter rotating vortex pairs (CVP, figure 26). Figure 30 presents the vorticity contours and streamlines in the xy-plane at z=0. A complete description of the flow physics can be found in [15]. Vorticity contours and streamlines are plotted in a 2D cut in the xy-plane at z=0 to show the jet trajectory at $\bar{t}$=3. The distortion of the temperature field because of diffusion, ring-like vortical structures and the CVP can be observed in Figure 30 and 31, in which the temperature contours and temperature isosurface, $\theta = 0.2$, of the evolution in time of the jet are plotted at $\bar{t}$=0.75, 1.5, 2.25 and 3.

These results have been compared with those obtained by Wee and Ghoniem in [39] for the validation of the vorticity field. Further discretization refinements left the trajectories and vortical structures unchanged, suggesting that this problem is well resolved. The temperature field evolution is well resolved for $\bar{t} \leq 3$, but a better discretization, i.e., a smaller grid size, is required during the diffusion step to push the simulation further. In fact, smaller scale vortical structures appear after $\bar{t} = 3$. A convergence study is performed by repeating the same simulations for different grid sizes, Δx=0.025, 0.035 and 0.05. The number of elements at $\bar{t} = 3$ of the finest simulation is around 1,350,000. The temperature field is computed on a three-dimensional Cartesian grid, $-1.5 \leq x \leq 4.5$, $0 \leq y \leq 5$ and $-2 \leq z \leq 0$, by performing our fast summation over all the computational elements. The temperature integral, plotted in Figure 32, is then obtained by performing a summation over all the grid points. This plot shows the convergence of the method. In fact, the temperature integral slope should be constant, as it is the case for the two finest simulations. We note a slow decrease in their temperature integral for $\bar{t} \geq 2.5$, but this is mainly due to the fact that the domain of integration of the temperature field is finite.

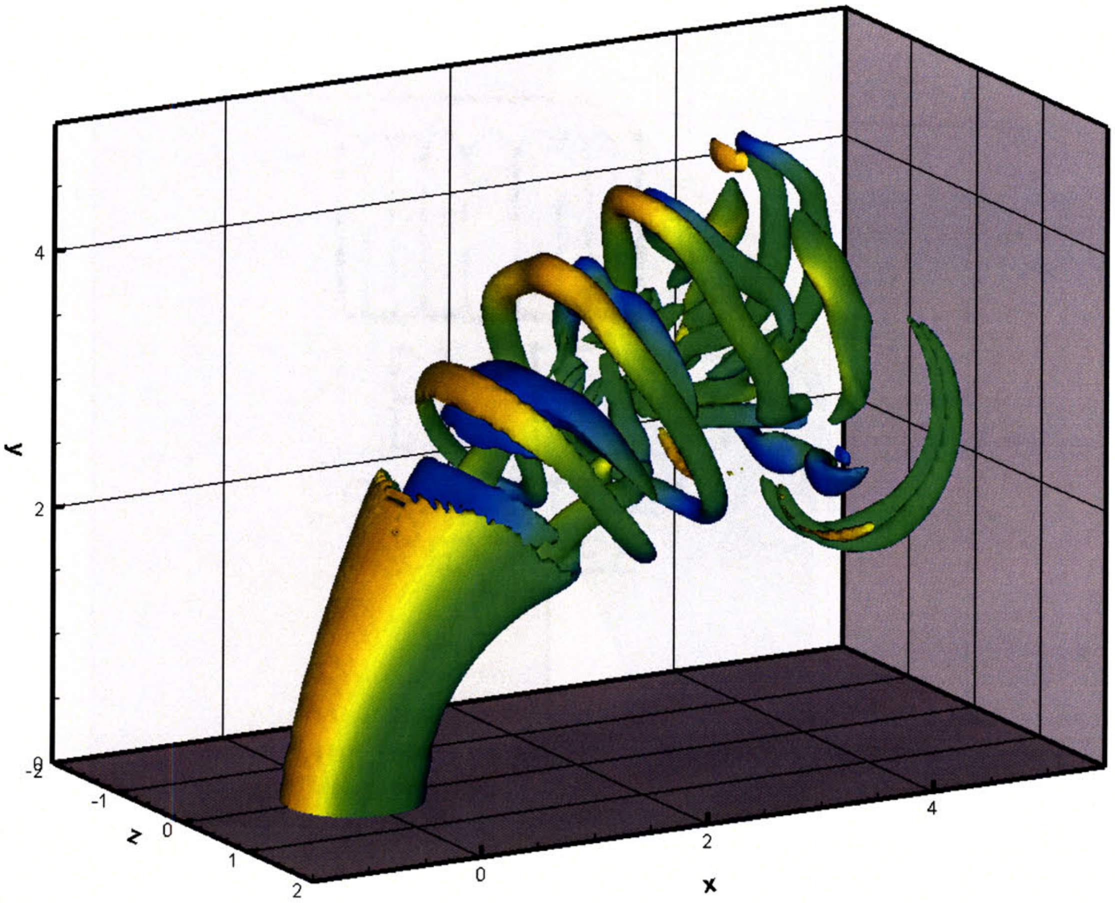

Figure 26: Vorticity isosurface, $|\omega| = 15$, of the non-buoyant transverse jet of initial radius R=0.5, r=5, at $\bar{t}$=3. Jet contoured by spanwise vorticity ω_z.

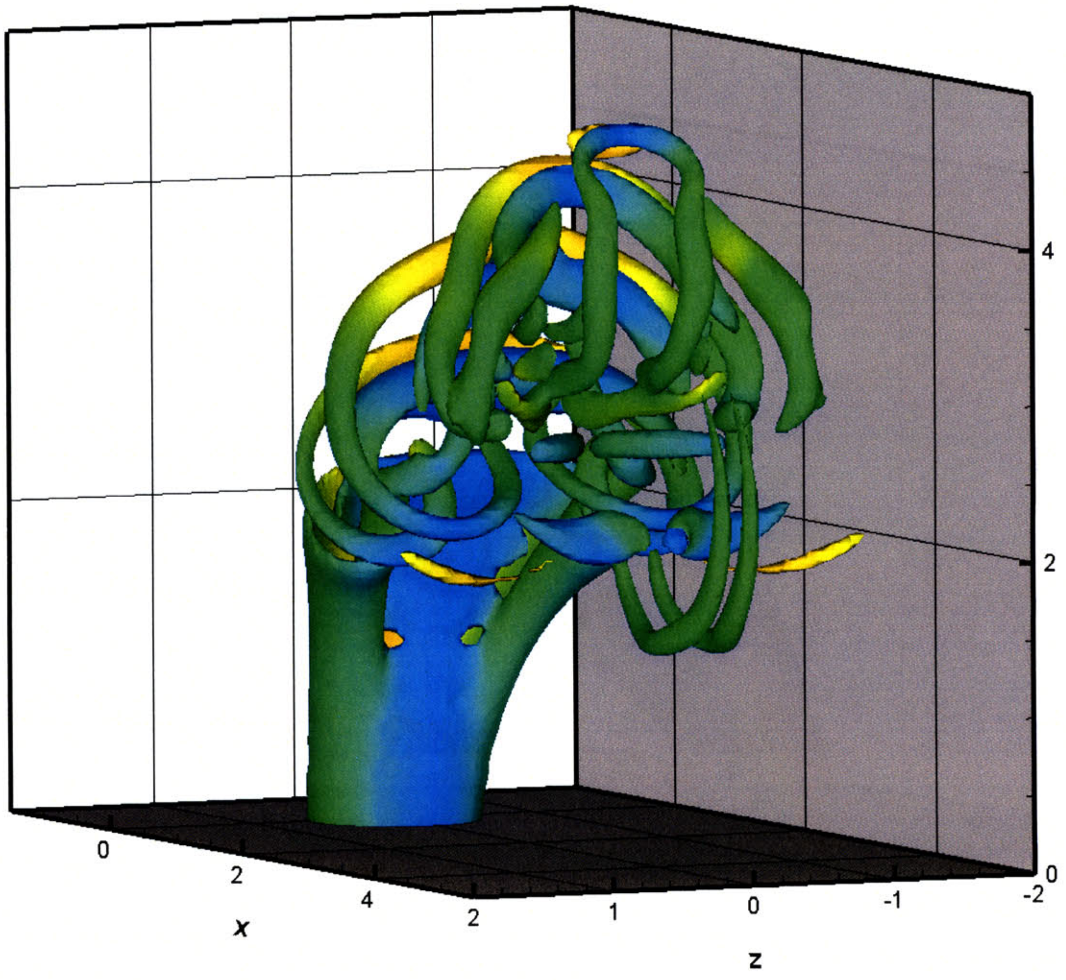

Figure 27: Vorticity isosurface, $|\omega| = 15$, of the non-buoyant transverse jet of initial radius *R*=0.5, r=5, at $\bar{t}$ =3. Jet contoured by spanwise vorticity ω_z.

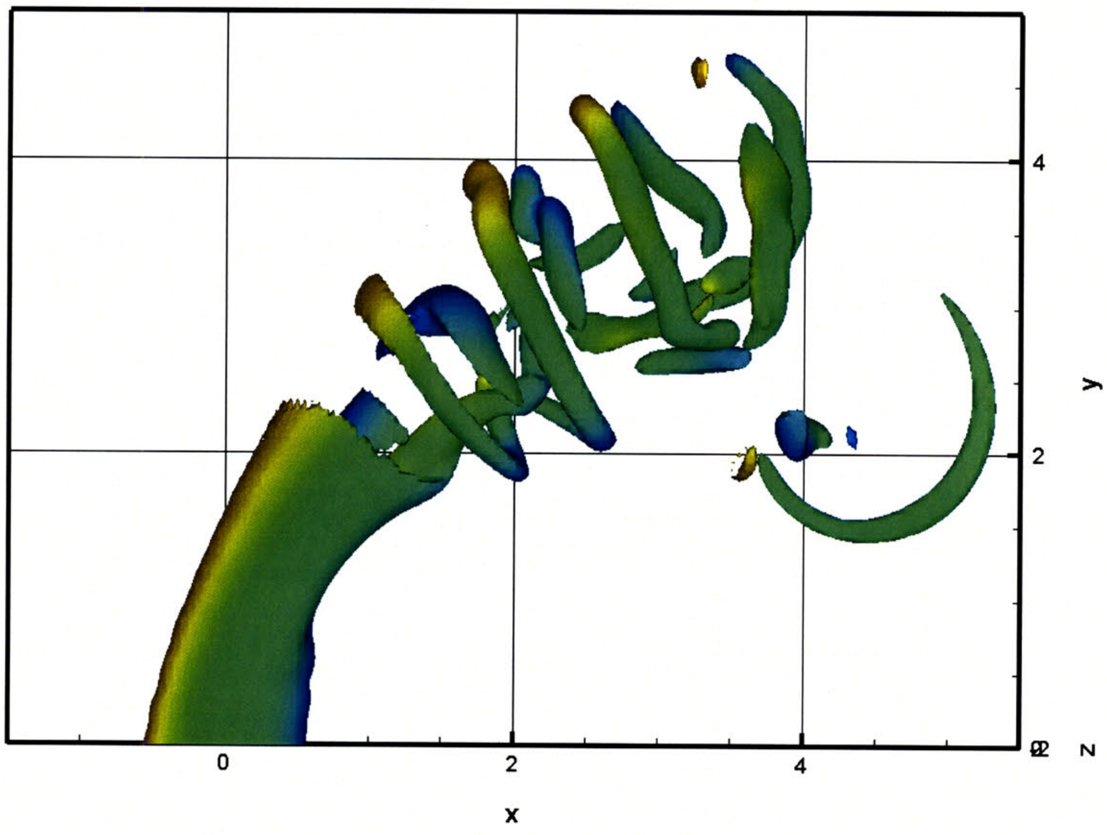

Figure 28: Vorticity isosurface, $|\omega| = 15$, of the non-buoyant transverse jet of initial radius R=0.5, r=5, at $\bar{t}$=3. Jet contoured by spanwise vorticity ω_z.

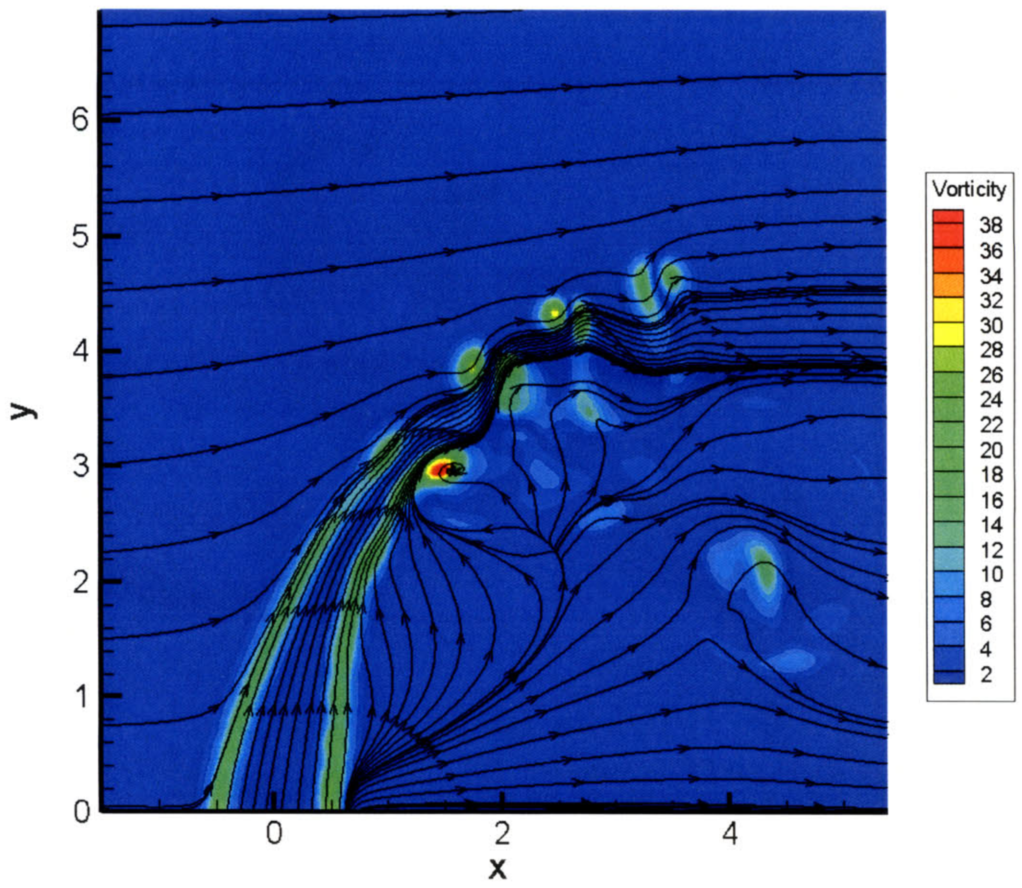

Figure 29: Vorticity contours and streamlines of the non-buoyant transverse jet of initial radius R=0.5, r=5, at $\bar{t}$=3, 2D cut in the *xy*-plane at *z*=0 (3D Simulation).

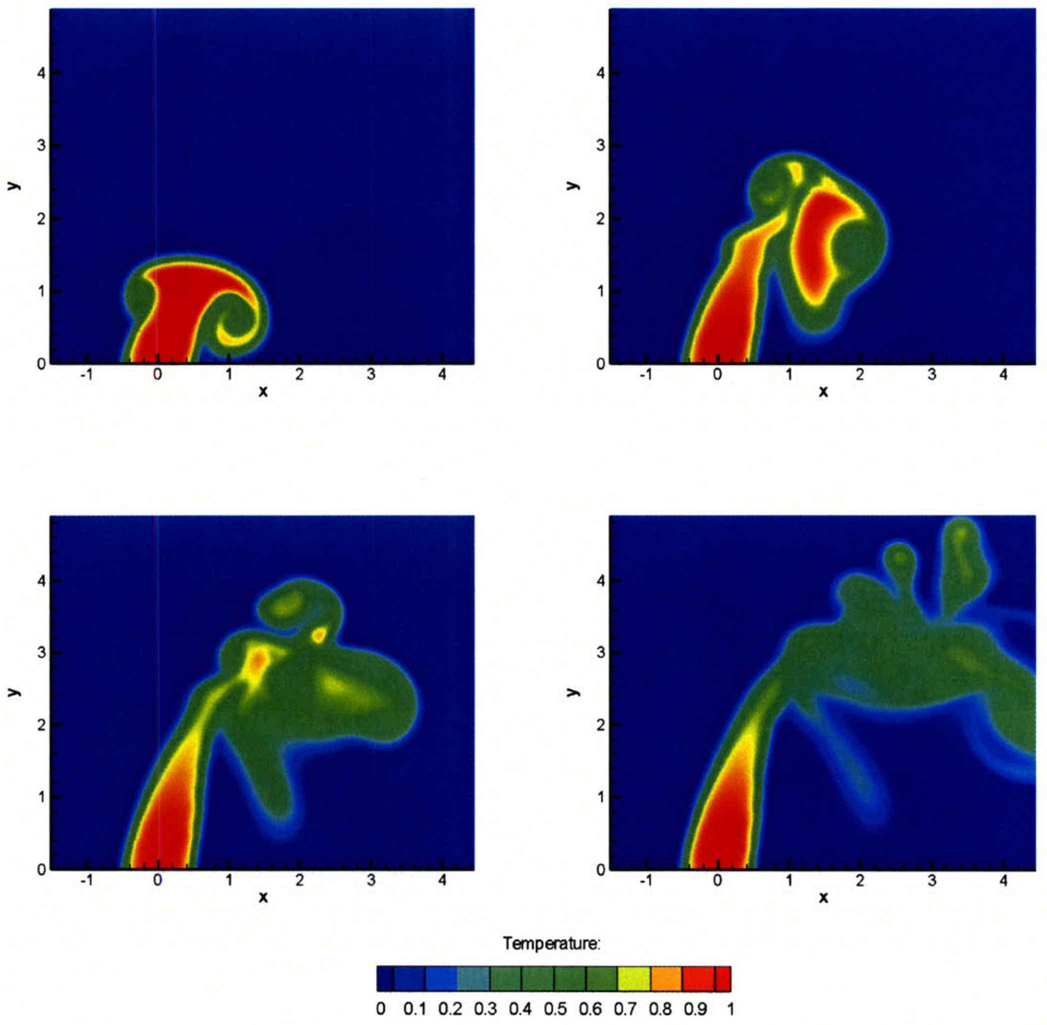

Figure 30: Temperature contours of the evolution in time of the non-buoyant transverse jet of initial radius *R*=0.5, r=5, at $\bar{t}$ =0.75, 1.5, 2.25 and 3, 2D cuts in the *xy*-plane at *z*=0 (3D Simulation).

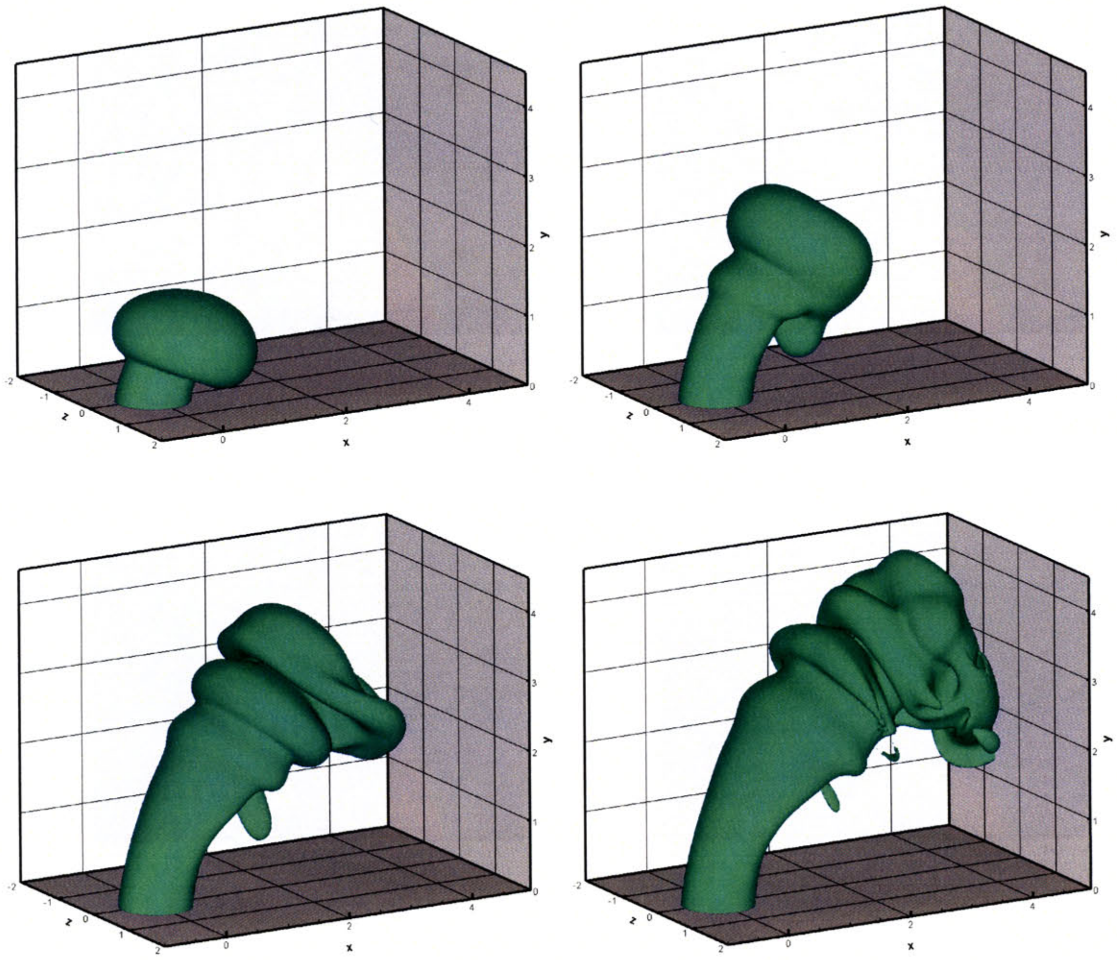

Figure 31: Temperature isosurfaces, $\theta = 0.2$, of the evolution in time of the non-buoyant transverse jet of initial radius R=0.5, r=5, at $\bar{t}$ =0.75, 1.5, 2.25 and 3.

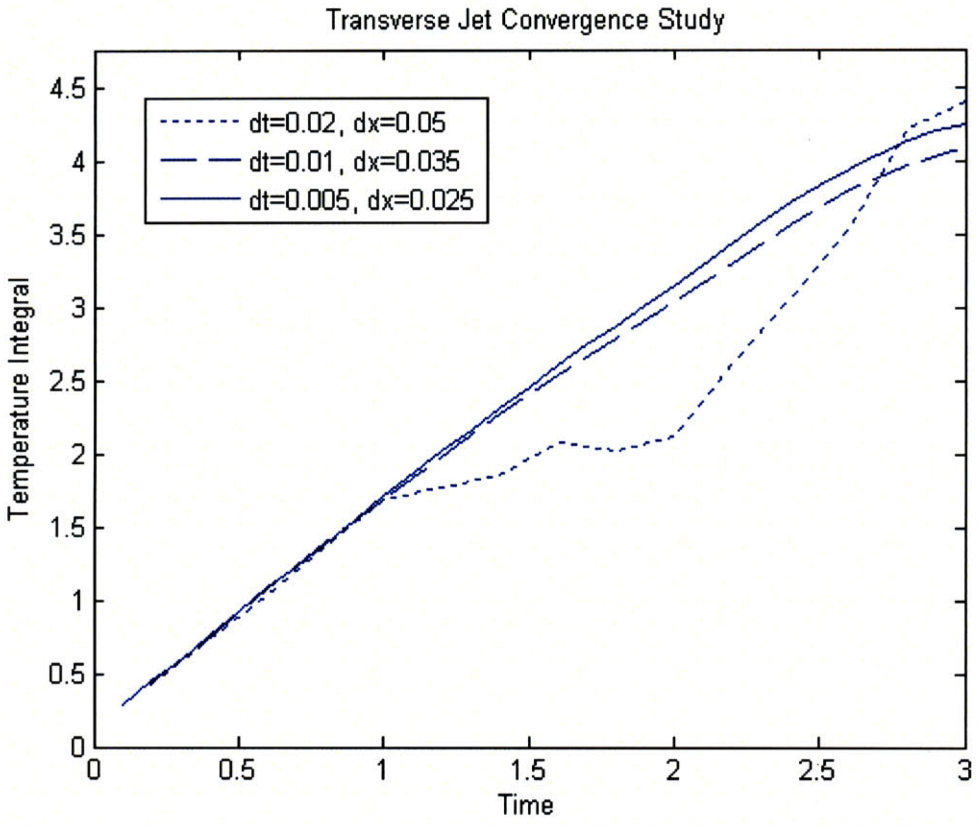

Figure 32: Temperature integral of the non-buoyant transverse jet of initial radius *R*=0.5, r=5, for different resolutions.

3.2.3. Buoyant Transverse Jet

Figure 33, 34 and 35 show a comparison between the non-buoyant and buoyant jet-in-crossflow isosurfaces, $|\omega| = 15$, at $\bar{t}$=3. The jet is contoured by spanwise vorticity ω_z. Again, the development of ring-like vortical structures and the CVP can be observed. The plot shows that buoyancy significantly contributes to the upward motion of hot fluid. In fact, due to the density difference between the jet and the crossflow fluids, the trajectory of the buoyant jet is not as tilted as the one of the non-buoyant jet. Vorticity is added at the temperature gradient locations due to baroclinic generation and it allows the buoyant jet to penetrate deeper in the crossflow. It is even more interesting to observe vorticity isosurfaces, $|\omega| = 20$, of both cases in Figure 35 in 3D, in which only two KH rings and the CVP can be seen at $\bar{t}$=3 for the non-buoyant case.

Temperature isosurfaces, $\theta = 0.2$, 0.3 and 0.4, of the transverse non-buoyant and buoyant jet are plotted in Figure 37, 38 and 39 under different views at $\bar{t}$=3. These figures show first that, due to the baroclinic generation of vorticity, the hot fluid rises faster to the top, but also that, the hot liquid mixes faster with the crossflow fluid due to the presence of more vortical structures on the back side of the jet.

To quantify the vorticity difference between the non-buoyant and the buoyant case and show the impact of buoyancy on the temperature field and the jet trajectory, we plot the 2D cuts in the *xy*-plane at *z*=0 of the temperature and vorticity contours in Figure 40. We note that the trajectories are similar for $y \leq 2D$, before the jet vorticity diffuses and before vorticity generated through baroclinicity has a significant effect on the flow field. But buoyancy plays an important roles and, consequently, the buoyant jet has much stronger vortical structures (core of the jet $y \leq 2D$, KH rings and CVPs (Figure 35)). Its behavior is, however, still close to the behavior of the non-buoyant jet for $y \leq 4D$. Qualitatively speaking, for $y \leq 4D$, the vortical structures of the buoyant case with a velocity ratio *r*=5 correspond to the ones of the non-buoyant case for a velocity ratio *r*=7 or 8.

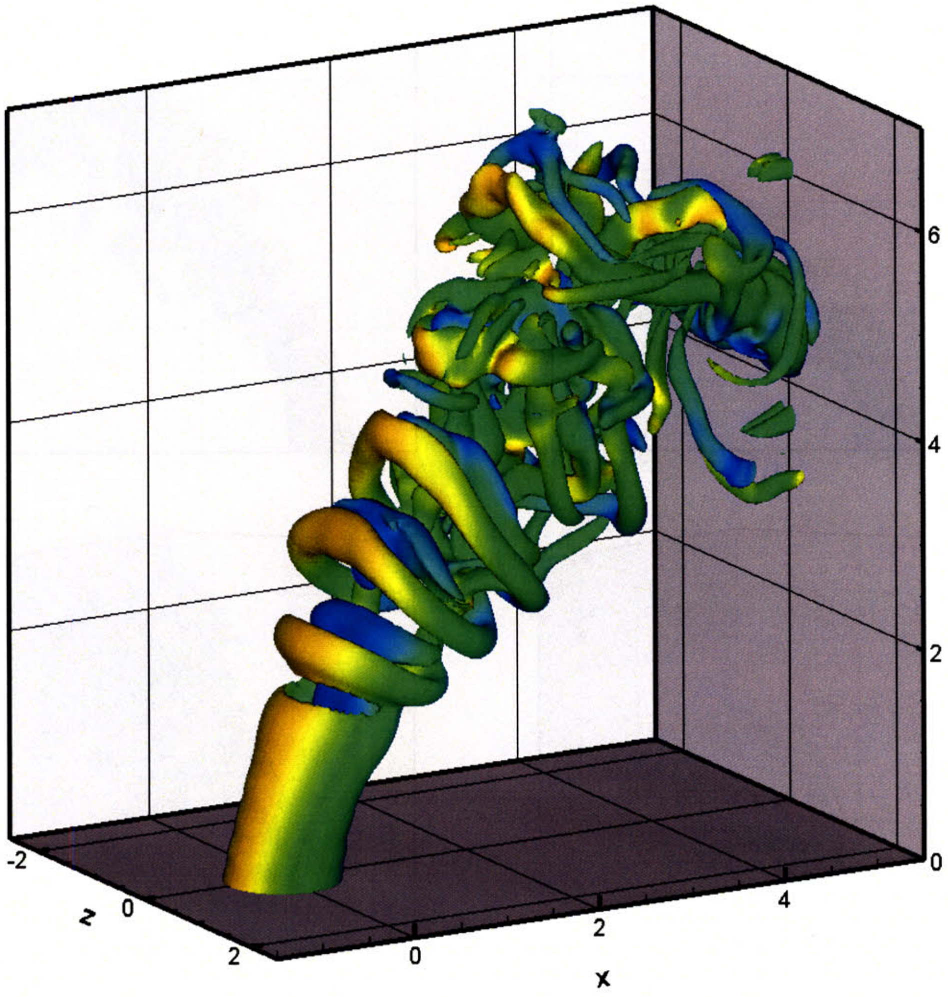

Figure 33: Vorticity isosurface, $|\omega| = 15$, of the buoyant transverse jet of initial radius R=0.5, r=5, at $\bar{t}$=3. Jet contoured by spanwise vorticity ω_z.

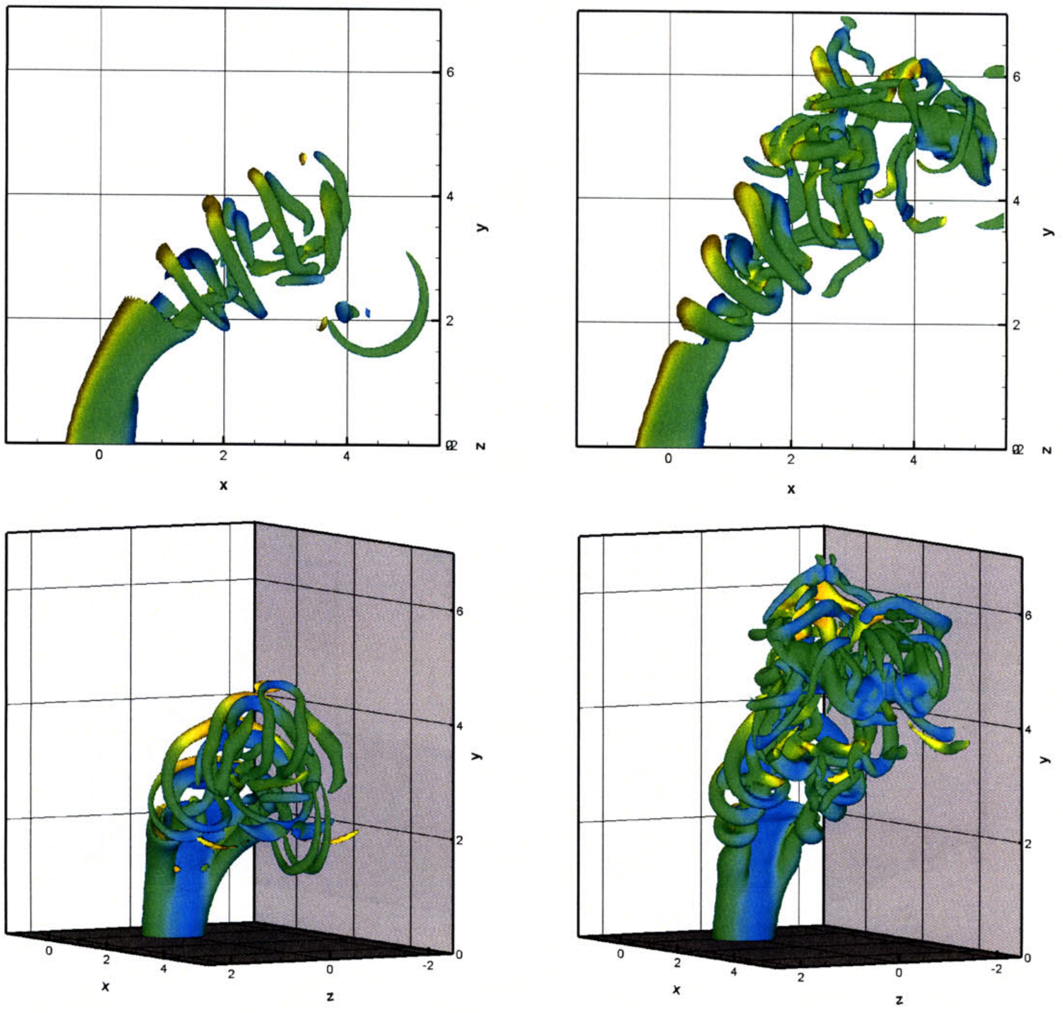

Figure 34: Vorticity isosurface, $|\omega| = 15$, of the transverse non-buoyant (left) and buoyant (right) jet of initial radius R=0.5, r=5, at $\bar{t}$ =3.

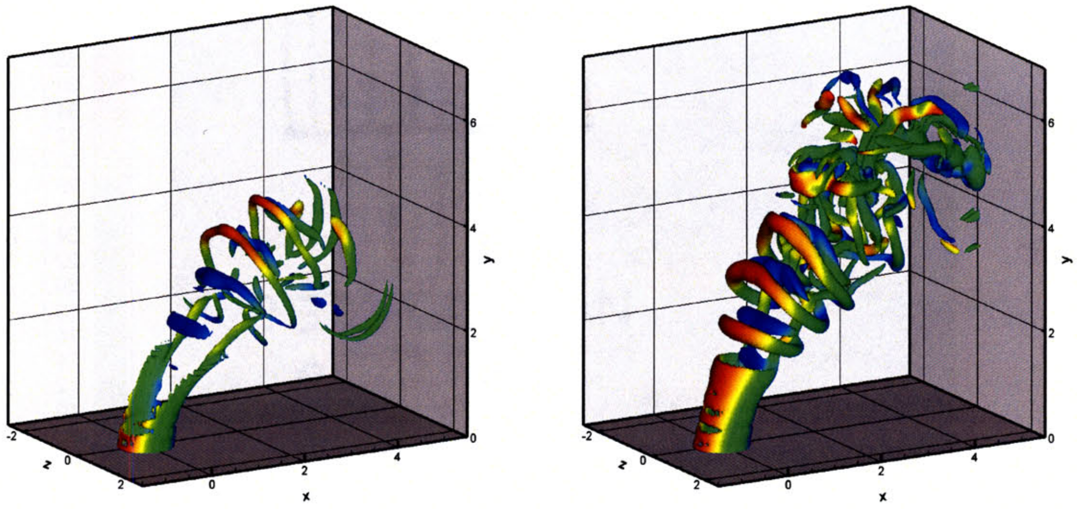

Figure 35: Vorticity isosurface, $|\omega| = 20$, of the transverse non-buoyant (left) and buoyant (right) jet of initial radius *R*=0.5, r=5, at $\overline{t}$ =3.

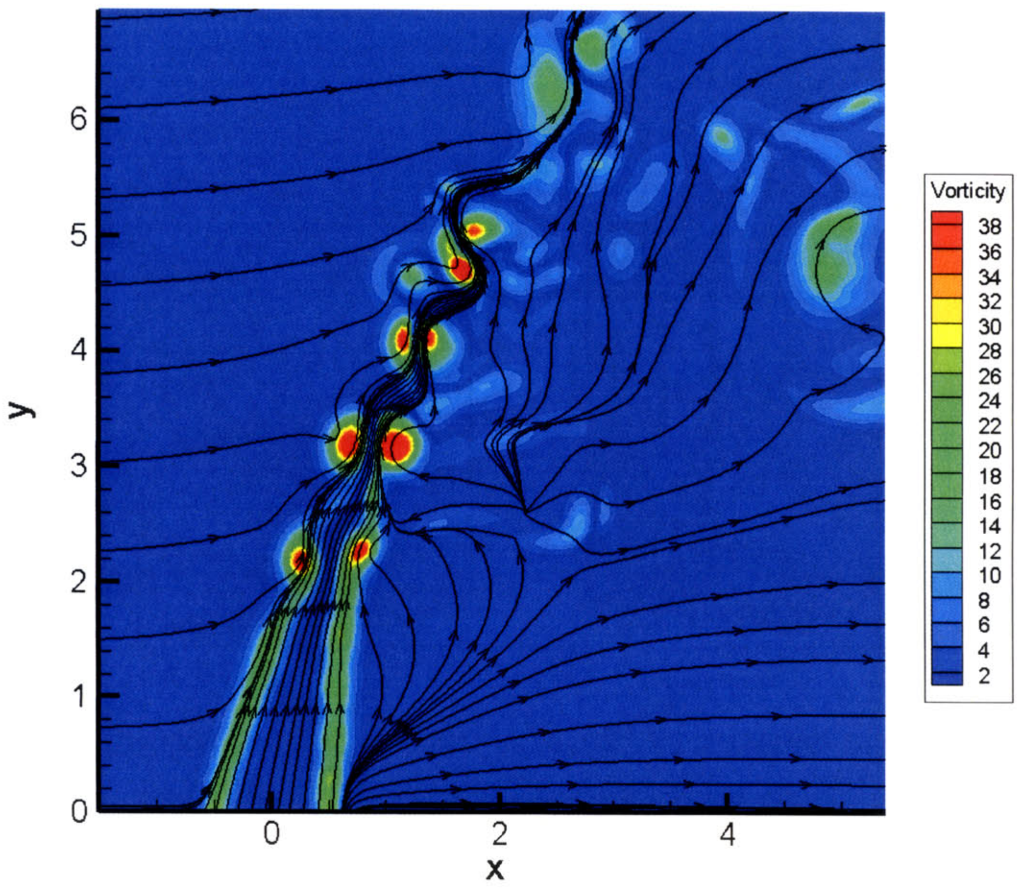

Figure 36: Vorticity contours and streamlines of the non-buoyant transverse jet of initial radius R=0.5, r=5, at $\bar{t}$=3, 2D cut in the *xy*-plane at *z*=0 (3D Simulations).

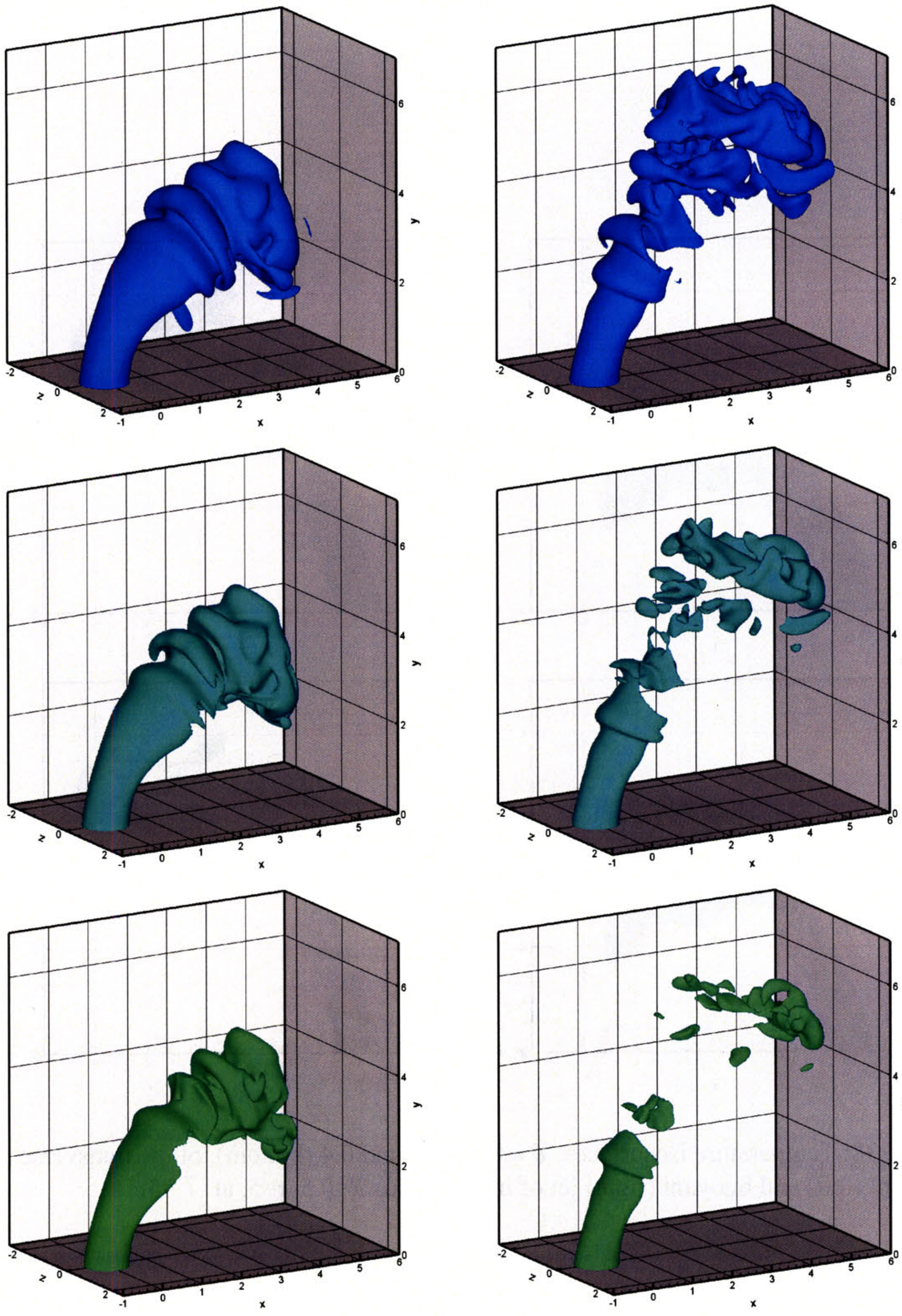

Figure 37: Temperature isosurfaces, $\theta = 0.2$ (top), 0.3 (middle) and 0.4 (bottom), of the transverse non-buoyant (left) and buoyant (right) jet of initial radius R=0.5, r=5, at $\bar{t}$ =3.

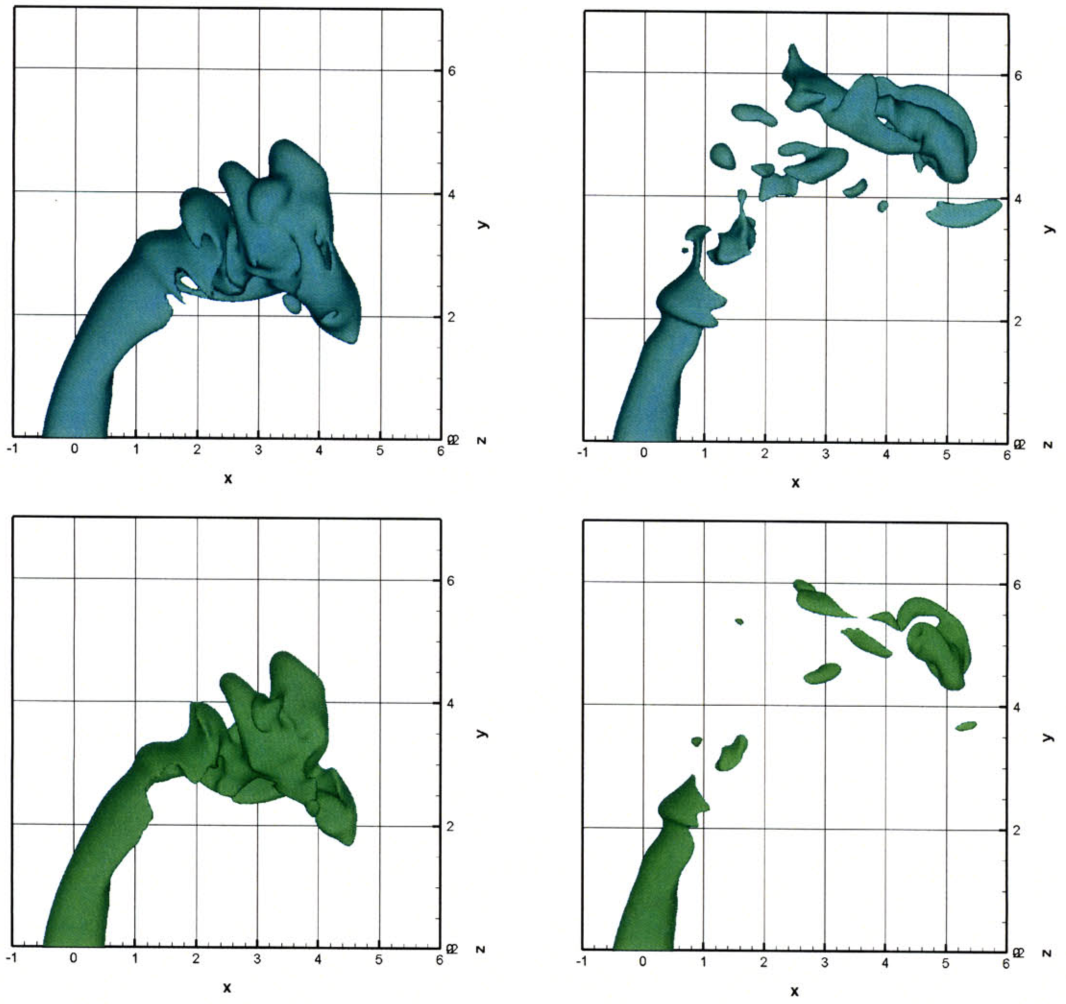

Figure 38: Temperature isosurfaces, $\theta = 0.3$ (top) and 0.4 (bottom), of the transverse non-buoyant (left) and buoyant (right) jet of initial radius R=0.5, r=5, at $\bar{t}$ =3.

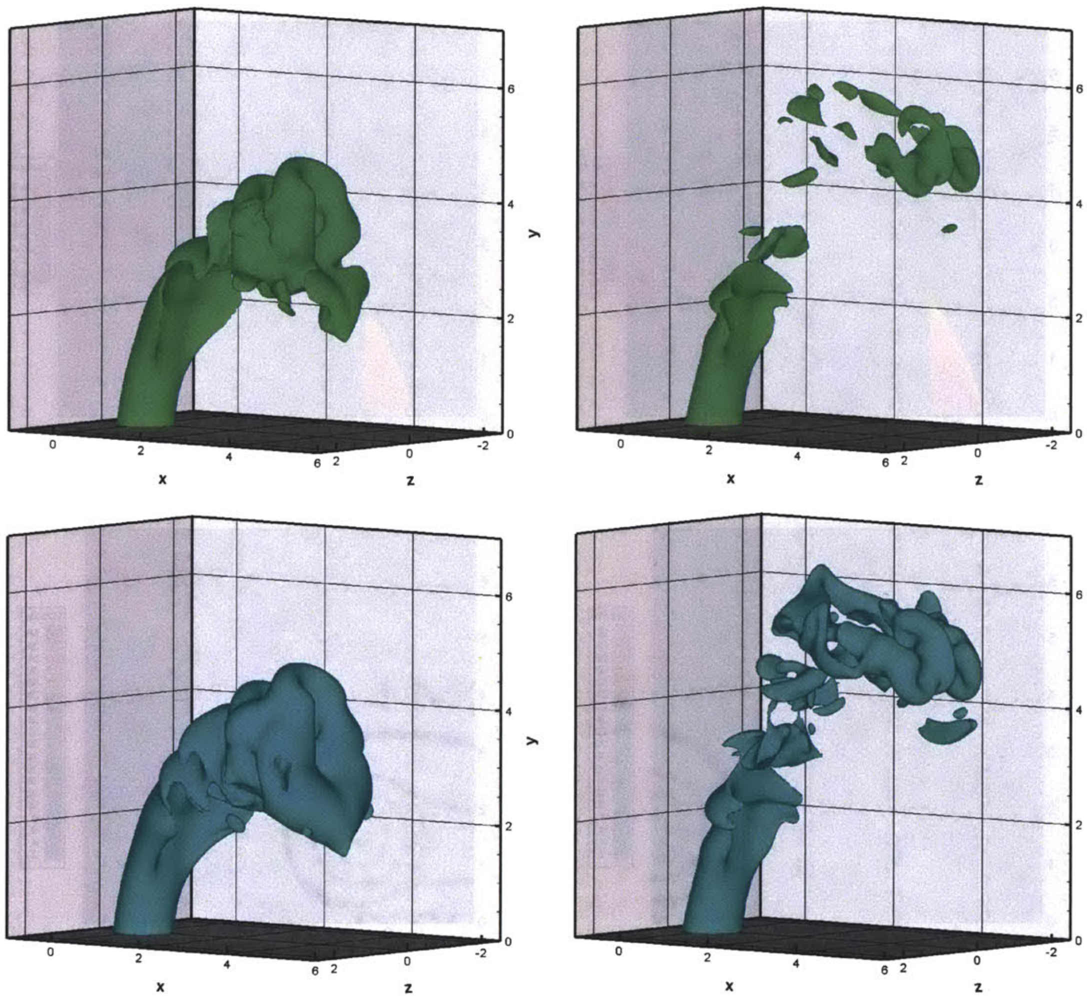

Figure 39: Temperature isosurfaces, $\theta = 0.3$ (top) and 0.4 (bottom), of the transverse non-buoyant (left) and buoyant (right) jet of initial radius R=0.5, r=5, at $\bar{t}$=3.

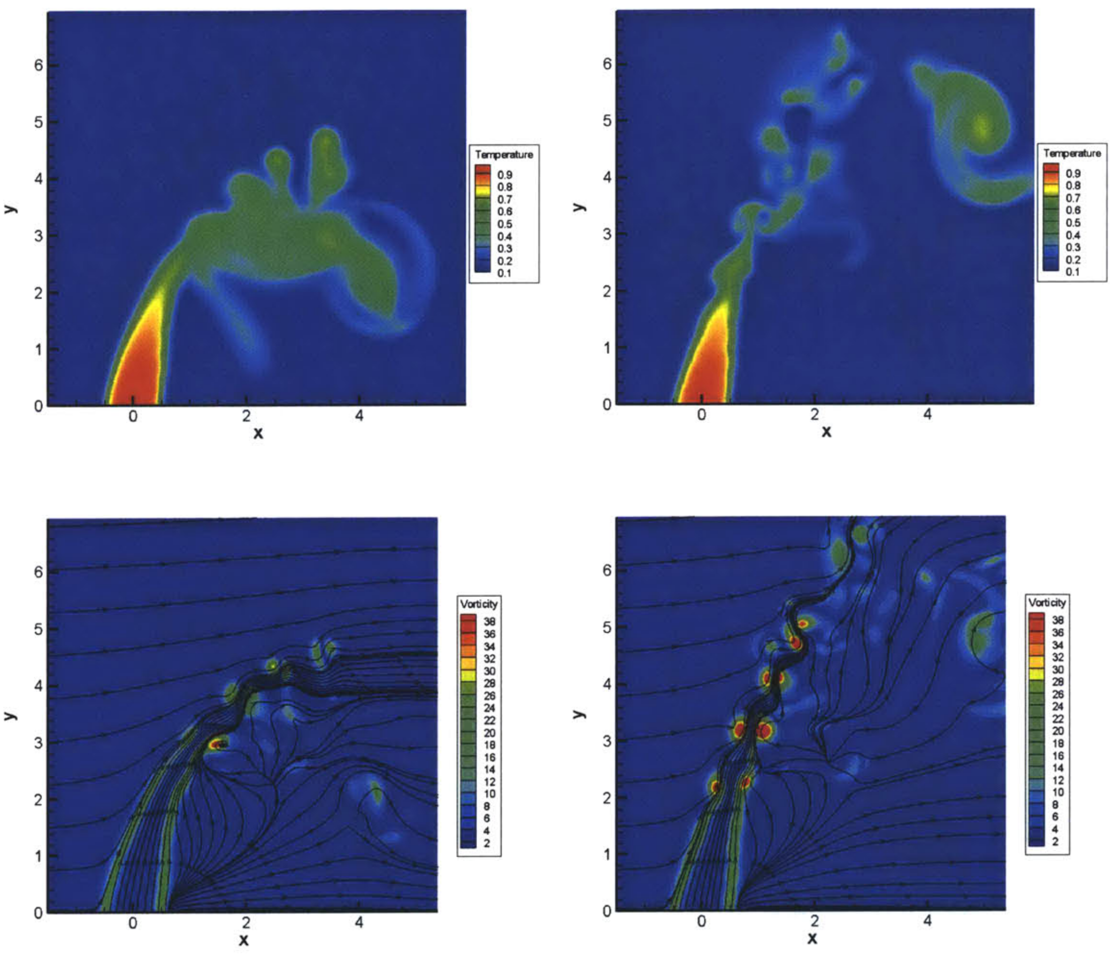

Figure 40: Temperature (top) and Vorticity (bottom) contours of the transverse non-buoyant (left) and buoyant (right) jet of initial radius R=0.5, r=5, at $\overline{t}$=3, 2D cut in the *xy*-plane at z=0 (3D Simulations).

4. Multi-Purpose Adaptive Treecode

4.1. Problem Definition

Lagrangian simulations using vorticity or gradients as particle weights require the solution of several differential equations that model the convection and source terms in the original conservation equations, to update the field. One after needs to compute the vortical velocity, $\mathbf{u}_\omega$, from the vorticity field, $\boldsymbol{\omega}$. One also needs to obtain the expansion velocity, $\mathbf{u}_\varepsilon$, from the divergence field, ε, generated by the volumetric expansion of material elements due to, e.g., a chemical reaction. In the case of using gradients for the primitive variable, e.g. in the transport element methods, there is also a need to recover the conserved scalar field, s (or θ in the main texts), from the information on its gradient, $\mathbf{g}$. Additionally, the evolutions of the weights of vortex elements and/or transport elements rely on the information regarding the gradient of the velocity field. Thus, it is necessary to compute $\nabla\mathbf{u}_\omega$ and $\nabla\mathbf{u}_\varepsilon$ simultaneously from $\boldsymbol{\omega}$ and ε directly.

In summary, the problem can be restated as follows. During each time step of the simulation, it is necessary to invert one or many of the following equations:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}_\omega, \qquad (56)$$

$$\varepsilon = \nabla \cdot \mathbf{u}_\varepsilon, \qquad (57)$$

$$\mathbf{g} = \nabla s. \qquad (58)$$

That is, knowing $\boldsymbol{\omega}$, ε, and $\mathbf{g}$, we need to calculate $\mathbf{u}_\omega$, $\mathbf{u}_\varepsilon$, and their gradients, as well as s. $\boldsymbol{\omega}$, ε, and $\mathbf{g}$ are all discretized into Lagrangian computational elements or particles:

$$\boldsymbol{\omega}(\mathbf{x},t) \approx \sum_{j=1}^{N} \mathbf{W}_j(t) f_\delta(\mathbf{x} - \boldsymbol{\chi}_j(t)), \qquad \text{(vortex elements; 59)}$$

$$\varepsilon(\mathbf{x},t) \approx \sum_{j=1}^{N} E_j(t) f_\delta(\mathbf{x} - \boldsymbol{\chi}_j(t)), \qquad \text{(divergence elements; 60)}$$

$$\mathbf{g}(\mathbf{x},t) \approx \sum_{j=1}^{N} \mathbf{G}_j(t) f_\delta(\mathbf{x} - \boldsymbol{\chi}_j(t)), \qquad \text{(transport elements; 61)}$$

The weights $\mathbf{W}_j$, E_j, and $\mathbf{G}_j$ correspond to the vorticity, divergence, and gradient assigned to each computational element, i.e., $\mathbf{W}_j = [\boldsymbol{\omega} dV]_j$, $E_j = [\varepsilon dV]_j$, and $\mathbf{G}_j = [\mathbf{g} dV]_j$. $\boldsymbol{\chi}_j$ is the location of the jth particle. f_δ is a desingularized radially symmetric core function of radius δ, given by $f_\delta(\mathbf{x}) \equiv \delta^{-3} f(|\mathbf{x}|/\delta)$. The function f must be smooth and rapidly decaying at infinity. In this work, we use the low-order algebraic core function [16,33].

$$f(\rho) = \frac{3}{4\pi}\frac{1}{(1+\rho^2)^{5/2}}. \qquad (62)$$

In $\mathbf{R}^3$, the solution of (56) is given by the Biot-Savart law.

$$\mathbf{u}_\omega(\mathbf{x}) = -\frac{1}{4\pi}\int \frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|^3} \times \boldsymbol{\omega}(\mathbf{y})\,d\mathbf{y}\,. \tag{63}$$

With (59), (63) can be rewritten as follows:

$$\mathbf{u}_\omega(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j) \times \mathbf{W}_j\,, \tag{64}$$

where $\mathbf{K}_\delta$ is the Rosenhead-Moore kernel

$$\mathbf{K}_\delta(\mathbf{x},\mathbf{y}) = -\frac{1}{4\pi}\frac{\mathbf{x}-\mathbf{y}}{(|\mathbf{x}-\mathbf{y}|^2+\delta^2)^{3/2}}\,. \tag{65}$$

In a similar way, one can invert (60) and (61) as follows:

$$\mathbf{u}_\varepsilon(\mathbf{x}) = -\sum_{j=1}^{N} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\,E_j\,, \tag{66}$$

$$s(\mathbf{x}) = -\sum_{j=1}^{N} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\cdot \mathbf{G}_j\,. \tag{67}$$

The velocity gradients are obtained by taking the gradients of (64) and (66).

$$\nabla \mathbf{u}_\omega(\mathbf{x}) = \sum_{j=1}^{N} \nabla_{\mathbf{x}} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j) \times \mathbf{W}_j\,, \tag{68}$$

$$\nabla \mathbf{u}_\varepsilon(\mathbf{x}) = -\sum_{j=1}^{N} \nabla_{\mathbf{x}} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\,E_j\,. \tag{69}$$

The summations in Eq.(64), (66), (67), (68), and (69) necessitate the evaluation of particle-particle interactions, whose cost scales as $O(N^2)$, where N is the number of computational elements. The corresponding computational load grows up quickly as we increase the problem size, we must perform these operations using an adaptive tree-code that limits the computational load roughly within $O(N \log N)$. In the following section, we describe the adaptive

4.2. Overview of the tree-code

Our multi-purpose tree-code is based on the work in [33]. Particles are divided into a nested set of clusters by constructing a tree, and particle-particle interactions are replaced by a smaller number of particle-cluster interactions. The tree construction starts with the root cell containing all the particles. The cell on the next level is obtained by bisecting one of the cells at the current level in one of the three coordinate directions. When every terminal cell in the tree contains a number of particles smaller than the smallest leaf size, N_0, which is predefined by the user, the process terminates and returns the tree structure.

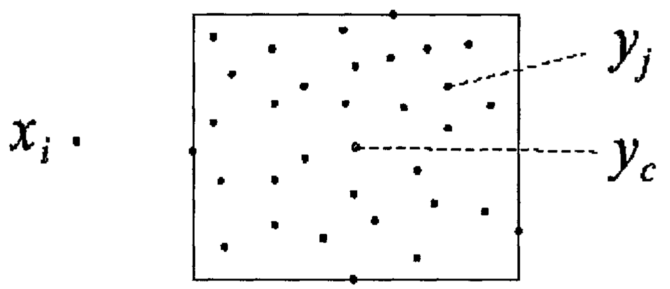

Figure 41: Particle-cluster interactions, where y_c denotes the cell center of the cluster c, and x_i is the point where the quantities are computed.

Once the tree is constructed, (64), (66), (67), (68), and (69) are rewritten in the following form:

$$\mathbf{u}_\omega(\mathbf{x}) = \sum_c \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x},\chi_j) \times \mathbf{W}_j , \tag{70}$$

$$\nabla \mathbf{u}_\omega(\mathbf{x}) = \sum_c \sum_{j=1}^{N_c} \nabla_{\mathbf{x}} \mathbf{K}_\delta(\mathbf{x},\chi_j) \times \mathbf{W}_j , \tag{71}$$

$$\mathbf{u}_\varepsilon(\mathbf{x}) = -\sum_c \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x},\chi_j) E_j , \tag{72}$$

$$\nabla \mathbf{u}_\varepsilon(\mathbf{x}) = -\sum_c \sum_{j=1}^{N_c} \nabla_{\mathbf{x}} \mathbf{K}_\delta(\mathbf{x},\chi_j) E_j , \tag{73}$$

$$s(\mathbf{x}) = -\sum_c \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x},\chi_j) \cdot \mathbf{G}_j , \tag{74}$$

where c denotes a cluster containing N_c particles. The particle-cluster interactions are evaluated either by using Taylor approximation or by direct summation, following the same strategy described in [33].

The procedure uses a complex combination of theoretical error estimates and empirical computational time estimates to determine the best order of the approximation and the best size of the cluster. In fact, the algorithm starts with the root cell containing all the particles. On each level the cells are uniform cubes; the cells on one level are obtained by bisecting the cells on the previous level in the three coordinate directions. Finally, the tree is adapted to the particles distribution by leaving undivided any cell containing fewer than a user-specified number of particles N_0. The procedure determines then the minimum order p satisfying the accuracy criterion, and a run time choice is made between Taylor expansion and direct summation. If approximation is faster, and the required order smaller than a user-specified parameter, the Taylor approximation is performed. If direct summation is faster or high-order approximation is required, then direct summation is performed if the cluster c is a leaf $(N_c \le N_0)$. Otherwise, the code descends to the next level of the tree and recursively calls itself for each child $\hat{c}$ of cluster c.

The accuracy parameter is taken to be,

$$\hat{\varepsilon} = \frac{M_0(\hat{c})}{M_0(c)}\varepsilon$$

where

$$M_0(c) = \sum_{j=1}^{N_c} |\,\omega_j\,|$$

is the total weight of the particles in cluster c. Thus, the accuracy parameter is distributed to children in proportion to their weight. A parallel implementation of the same algorithm that uses k-means clustering to distribute the load among a number of processors is documented in [34].

4.3. Taylor Approximation

To derive a Taylor approximation for a particle-cluster interaction, $\mathbf{K}_\delta(\mathbf{x},\mathbf{y})$ in (70) is expanded in a Taylor series with respect to $\mathbf{y}$, around the cluster center $\mathbf{y}_c$, such that

$$\begin{aligned}\sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\times \mathbf{W}_j &= \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c + (\boldsymbol{\chi}_j - \mathbf{y}_c))\times \mathbf{W}_j \\ &= \sum_{j=1}^{N_c}\sum_{\mathbf{k}} \frac{1}{\mathbf{k}!} D_y^{\mathbf{k}} \mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c)\ (\boldsymbol{\chi}_j - \mathbf{y}_c)^{\mathbf{k}} \times \mathbf{W}_j \qquad (75) \\ &= \sum_{\mathbf{k}} \mathbf{a}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)\times \mathbf{m}_{\mathbf{k}}^{\omega}(c).\end{aligned}$$

Here, $\mathbf{a}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)$ is the **k**th Taylor coefficient of $\mathbf{K}_\delta(\mathbf{x},\mathbf{y})$ at $\mathbf{y}=\mathbf{y}_c$:

$$\mathbf{a}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c) = \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c), \qquad (76)$$

and $\mathbf{m}_{\mathbf{k}}^{\omega}(c)$ is the **k**th moment of the vortex elements in cluster c about its center $\mathbf{y}_c$:

$$\mathbf{m}_{\mathbf{k}}^{\omega}(c) = \sum_{j=1}^{N_c} (\boldsymbol{\chi}_j - \mathbf{y}_c)^{\mathbf{k}}\,\mathbf{W}_j. \qquad (77)$$

$\mathbf{k} = (k_1,k_2,k_3)$ is an integer multi-index with $k_i \ge 0$, and $\mathbf{k}! = k_1!k_2!k_3!$. For $\mathbf{x}\in\mathbf{R}^3$, $\mathbf{x}^{\mathbf{k}}$ is interpreted in a standard way, i.e., $x_1^{k_1}x_2^{k_2}x_3^{k_3} \in \mathbf{R}$. The infinite series in (75) is approximated by a finite sum,

$$\sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\times \mathbf{W}_j \approx \sum_{|\mathbf{k}|\le p} \mathbf{a}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)\times \mathbf{m}_{\mathbf{k}}^{\omega}(c), \qquad (78)$$

where $|\,\mathbf{k}\,| = k_1 + k_2 + k_3$. The order of the approximation, p, must be chosen so that the error due to the truncation remains small.

For the particle-cluster interactions in (71), a similar series with different set of Taylor coefficients is developed.

$$\begin{aligned}\sum_{j=1}^{N_c}\nabla_{\mathbf{x}}\mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\times\mathbf{W}_j &= \sum_{j=1}^{N_c}\nabla_{\mathbf{x}}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c+(\boldsymbol{\chi}_j-\mathbf{y}_c))\times\mathbf{W}_j \\ &= \sum_{j=1}^{N_c}\sum_{\mathbf{k}}\frac{1}{\mathbf{k}!}D_y^{\mathbf{k}}\nabla_{\mathbf{x}}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c)\ (\boldsymbol{\chi}_j-\mathbf{y}_c)^{\mathbf{k}}\times\mathbf{W}_j \\ &= \sum_{\mathbf{k}}\mathbf{c}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)\times\mathbf{m}_{\mathbf{k}}^{\omega}(c),\end{aligned} \tag{79}$$

where $\mathbf{c}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)$ is the **k**th Taylor coefficient of $\nabla_{\mathbf{x}}\mathbf{K}_\delta(\mathbf{x},\mathbf{y})$ at $\mathbf{y}=\mathbf{y}_c$:

$$\mathbf{c}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)=\frac{1}{\mathbf{k}!}D_{\mathbf{y}}^{\mathbf{k}}\nabla_{\mathbf{x}}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c), \tag{80}$$

which yields a three-by-three matrix for each set of $(\mathbf{x},\mathbf{y}_c)$. Again, the infinite sum is truncated into a finite one up to an appropriate order p:

$$\sum_{j=1}^{N_c}\nabla_{\mathbf{x}}\mathbf{K}_\delta(\mathbf{x},\boldsymbol{\chi}_j)\times\mathbf{W}_j \approx \sum_{|\mathbf{k}|\le p}\mathbf{c}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)\times\mathbf{m}_{\mathbf{k}}^{\omega}(c). \tag{81}$$

To evaluate either (78) or (81), we need the Taylor coefficients, i.e., $\mathbf{a}_{\mathbf{k}}$ or $\mathbf{c}_{\mathbf{k}}$. An efficient method to obtain $\mathbf{a}_{\mathbf{k}}$ was proposed in [33]. The Rosenhead-Moore kernel (65) is given by the gradient of the Plummer potential:

$$\mathbf{K}_\delta(\mathbf{x},\mathbf{y})=-\nabla_{\mathbf{y}}\phi_\delta(\mathbf{x},\mathbf{y}), \tag{82}$$

where

$$\phi_\delta(\mathbf{x},\mathbf{y})=\frac{1}{4\pi}\frac{1}{(|\mathbf{x}-\mathbf{y}|^2+\delta^2)^{1/2}}. \tag{83}$$

We set the **k**th Taylor coefficient of $\phi_\delta(\mathbf{x},\mathbf{y})$ at $\mathbf{y}=\mathbf{y}_c$ as

$$b_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)=\frac{1}{\mathbf{k}!}D_{\mathbf{y}}^{\mathbf{k}}\phi_\delta(\mathbf{x},\mathbf{y}_c). \tag{84}$$

Then, $\mathbf{a}_{\mathbf{k}}$ is related to $b_{\mathbf{k}}$ as follows [33]:

$$\begin{aligned}\mathbf{a}_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c) &= \frac{1}{\mathbf{k}!}D_{\mathbf{y}}^{\mathbf{k}}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c) \\ &= \frac{1}{\mathbf{k}!}D_{\mathbf{y}}^{\mathbf{k}}\left(-\nabla_{\mathbf{y}}\phi_\delta(\mathbf{x},\mathbf{y}_c)\right) \\ &= -\frac{1}{\mathbf{k}!}D_{\mathbf{y}}^{\mathbf{k}}\left(\sum_{i=1}^{3}\mathbf{e}_i D_{\mathbf{y}}^{\mathbf{e}_i}\phi_\delta(\mathbf{x},\mathbf{y}_c)\right), \\ &= -\frac{1}{\mathbf{k}!}\sum_{i=1}^{3}\mathbf{e}_i D_{\mathbf{y}}^{\mathbf{k}+\mathbf{e}_i}\phi_\delta(\mathbf{x},\mathbf{y}_c) \\ &= -\sum_{i=1}^{3}\mathbf{e}_i\,(k_i+1)\,b_{\mathbf{k}+\mathbf{e}_i}(\mathbf{x},\mathbf{y}_c)\end{aligned} \tag{85}$$

where $\mathbf{e}_i$ is the ith Cartesian-basis vector. Therefore, to compute $\mathbf{a}_\mathbf{k}$, it is sufficient to obtain $b_\mathbf{k}$. The calculation of $b_\mathbf{k}$ is performed recursively to reduce the computational load, using the following formula [33]:

$$|\mathbf{k}|R^2 b_\mathbf{k} - (2|\mathbf{k}|-1)\sum_{i=1}^{3}(\mathbf{x}-\mathbf{y})\cdot\mathbf{e}_i b_{\mathbf{k}-\mathbf{e}_i} + (|\mathbf{k}|-1)\sum_{i=1}^{3} b_{\mathbf{k}-2\mathbf{e}_i} = 0, \tag{86}$$

for $|\mathbf{k}|\geq 1$, where $b_0(\mathbf{x},\mathbf{y}) = \phi_\delta(\mathbf{x},\mathbf{y})$, $b_\mathbf{k}(\mathbf{x},\mathbf{y}) = 0$ if any $k_i < 0$, and $R^2 = |\mathbf{x}-\mathbf{y}|^2 + \delta^2$.

To utilize the same machinery, we develop a similar relation for $\mathbf{c}_\mathbf{k}$ here:

$$\begin{aligned}
\mathbf{c}_\mathbf{k}(\mathbf{x},\mathbf{y}_c) &= \frac{1}{\mathbf{k}!}D_\mathbf{y}^\mathbf{k}(\nabla_\mathbf{x}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c)) \\
&= \frac{1}{\mathbf{k}!}D_\mathbf{y}^\mathbf{k}(\nabla_\mathbf{y}\nabla_\mathbf{y}\phi_\delta(\mathbf{x},\mathbf{y}_c)) \\
&= \frac{1}{\mathbf{k}!}D_\mathbf{y}^\mathbf{k}\sum_{i=1}^{3}\mathbf{e}_i D_\mathbf{y}^{\mathbf{e}_i}\sum_{j=1}^{3}\mathbf{e}_j D_\mathbf{y}^{\mathbf{e}_j}\phi_\delta(\mathbf{x},\mathbf{y}_c) \\
&= \frac{1}{\mathbf{k}!}\sum_{i=1}^{3}\sum_{j=1}^{3}\mathbf{e}_i\mathbf{e}_j D_\mathbf{y}^{\mathbf{k}+\mathbf{e}_i+\mathbf{e}_j}\phi_\delta(\mathbf{x},\mathbf{y}_c) \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\mathbf{e}_i\mathbf{e}_j\frac{(\mathbf{k}+\mathbf{e}_i+\mathbf{e}_j)!}{\mathbf{k}!}b_{\mathbf{k}+\mathbf{e}_i+\mathbf{e}_j}(\mathbf{x},\mathbf{y}_c)
\end{aligned} \tag{87}$$

with

$$\frac{(\mathbf{k}+\mathbf{e}_i+\mathbf{e}_j)!}{\mathbf{k}!} = \begin{cases}(k_i+1)(k_i+2) & \text{for } i=j \\ (k_i+1)(k_j+1) & \text{otherwise}\end{cases} \tag{88}$$

Therefore, just as $\mathbf{a}_\mathbf{k}$, $\mathbf{c}_\mathbf{k}$ can be obtained from $b_\mathbf{k}$. The only difference is that we need to evaluate $b_\mathbf{k}$ up to one order higher than that required for $\mathbf{a}_\mathbf{k}$. The additional cost of one more order does not matter much in most calculations, since most of particle-cluster interactions are dealt with at relatively low orders, namely, $p \leq 6$. As clearly seen in (87), $\mathbf{c}_\mathbf{k}$ is symmetric in its indices, i and j, and we thus evaluate only six components instead of all the nine components separately.

A similar construction has been developed for the particle-cluster interactions in (72), (73), and (74), that is,

$$\sum_{j=1}^{N_c}\mathbf{K}_\delta(\mathbf{x},\chi_j)E_j \approx \sum_{|\mathbf{k}|\leq p}\mathbf{a}_\mathbf{k}(\mathbf{x},\mathbf{y}_c)m_\mathbf{k}^\varepsilon(c), \tag{89}$$

$$\sum_{j=1}^{N_c}\nabla_\mathbf{x}\mathbf{K}_\delta(\mathbf{x},\chi_j)E_j \approx \sum_{|\mathbf{k}|\leq p}\mathbf{c}_\mathbf{k}(\mathbf{x},\mathbf{y}_c)m_\mathbf{k}^\varepsilon(c), \tag{90}$$

where

$$m_\mathbf{k}^\varepsilon(c) = \sum_{j=1}^{N_c}(\chi_j-\mathbf{y}_c)^\mathbf{k}E_j, \tag{91}$$

and

$$\sum_{j=1}^{N_c}\mathbf{K}_\delta(\mathbf{x},\chi_j)\cdot\mathbf{G}_j \approx \sum_{|\mathbf{k}|\leq p}\mathbf{a}_\mathbf{k}(\mathbf{x},\mathbf{y}_c)\cdot\mathbf{m}_\mathbf{k}^h(c), \tag{92}$$

where

$$\mathbf{m}_{\mathbf{k}}^{h}(c)=\sum_{j=1}^{N_c}(\chi_j-\mathbf{y}_c)^{\mathbf{k}}\mathbf{G}_j\,. \tag{93}$$

Since all of these relations only require the evaluation of the same coefficients, namely, $\mathbf{a}_{\mathbf{k}}$ and $\mathbf{c}_{\mathbf{k}}$, the process can be efficiently integrated in a single tree-code.

4.4. Results for the velocity gradient case

In this section, we examine the accuracy and CPU time of the tree code fast summation (Taylor approximation) in comparison with the direct summation using one data file coming from the interaction of two vortex ring.
Two separate ring are placed in the unbounded three-dimensional space (Figure 42), where s denotes the initial distance between the two centers. Each ring consists of a single closed filament, which is subdivided in 64 elements with $\sigma = 0.1$, where σ is the radius of the cutoff function. The ring has a unit circulation and a unit radius, i.e. $\Gamma = 1$ and $r = 1$. The Reynolds number is defined as $\mathrm{Re}=\Gamma/\nu$.
The data file is taken at t=4.8s. At this time the number of elements was 163,521 (Figure 42). The velocity gradient is computed on a regular grid with $x/d \in \{-2.5;2.5\}$, $y/d \in \{-2.5;0.5\}$ and $z/d \in \{-2;2\}$. This grid contains 20580 points.
The order of approximation depends on the accuracy parameter, the exact solution was computed using direct summation and an approximation was computed using the tree code for three values of the accuracy parameter, $\varepsilon = 10^{-1}, 10^{-2}, 10^{-3}$.

The norm of the gradient matrix $\mathbf{A}_{i,j}$ is defined as $\mathbf{A}_{i,j} = \sqrt{\sum_{i,j=1}^{3} a_{i,j}^{2}}$. The recorded relative error is the maximum value over all grid points of the relative error. We also computed the maximum value over all points of the gradient trace. Indeed, due to incompressibility, a velocity gradient matrix should have zero trace.

Accuracy	DS	10^{-1}	10^{-2}	10^{-3}
Computing time (s)	755.89	20.17	25.96	41.87
Max relative error		3.65E-01	3.04E-02	2.50E-03
Max trace value		7.21E-16	6.66E-16	5.55E-16

For three values of the accuracy parameter, $\varepsilon = 10^{-1}, 10^{-2}, 10^{-3}$, the trace of the velocity gradient matrix is about 10^{-16} (the code was implemented in double precision). As observed in the tabular, with the accuracy parameter $\varepsilon = 10^{-2}$ for instance, the relative error is about 3 % and the CPU time is only 3.4 % of the direct summation time. Figure 43 shows the number of interactions for each case. The number of interactions is the number of cells multiply by average of particles per cell at each order. For $\varepsilon = 10^{-2}$, most of the Taylor approximation are performed with an order $p = \{4,5,6\}$.

Hence, the velocity gradient of each particle is evaluated by an adaptive treecode algorithm based on Taylor approximation in Cartesian coordinates. The necessary Taylor coefficients are computed by a recurrence relation. The algorithm uses a divide-and-conquer strategy to evaluate the particle velocity gradient: the tree consists of non-uniform rectangular clusters adapted to the particle distribution. For each particle-cluster interaction, the order of approximation is chosen adaptively and a run-time choice is made between Taylor approximation and direct summation. Tests were performed to check the algorithm's accuracy and efficiency. The exact solution was computed using direct summation and checked using the existing tree code which computes the velocities and finite differences.

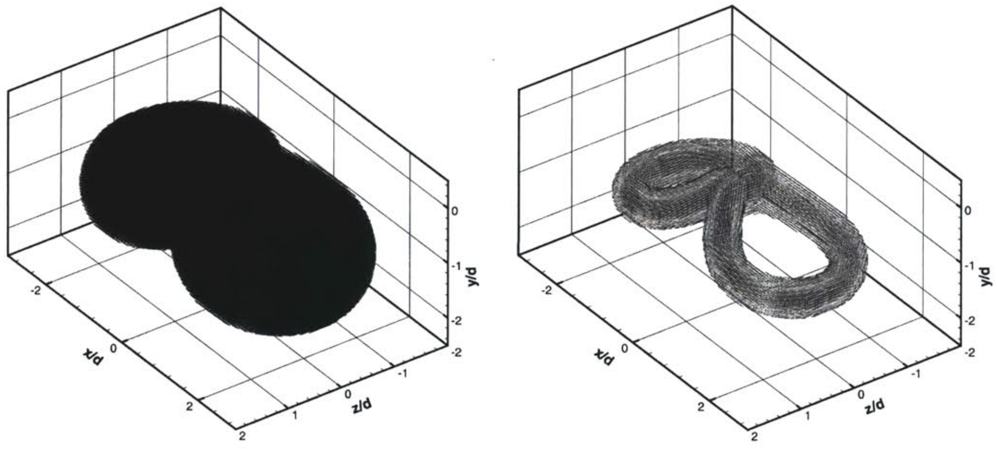

Figure 42: View of vortex element distribution at t=4.8s. Only element with $|\omega_i dV_i| \leq 10^{-4}$ are plotted left, and all the elements are plotted right.

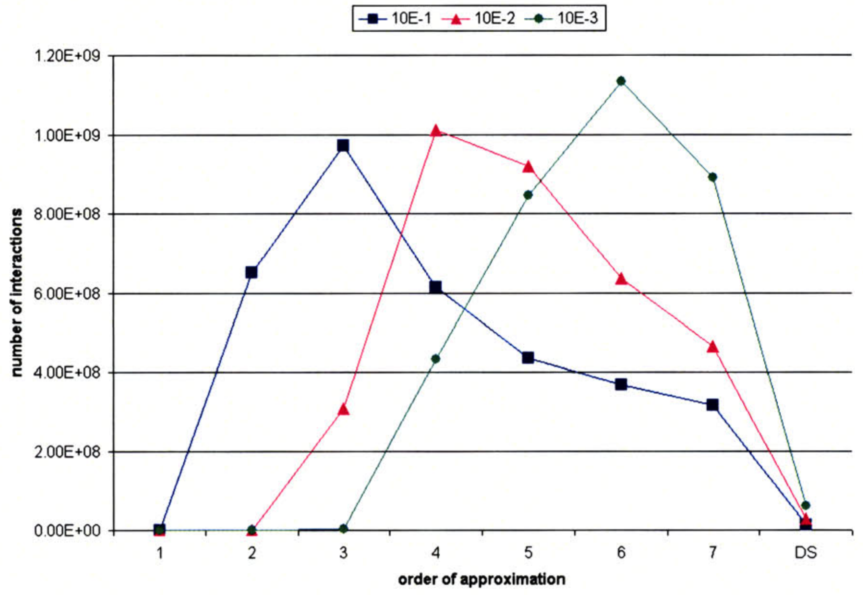

Figure 43: Number of interactions with targets points in the tree code as a function of the order of expansions in Taylor approximation. If direct summation is faster or high-order approximation is required, then direct summation is performed if the cluster c is a leaf $(N_c \leq N_0)$. Here $N_0 = 128$.

5. Conclusion

5.1. Transport Element Method

Vortex methods have been used in the simulation of high Reynolds number complex flows, especially when fast transitions and strong distortion of the underlying vortical structure of the flow are expected. The Lagrangian, self-adaptive nature of the calculations makes it possible to resolve strong gradients wherever and whenever they arise, while maintaining a coarse resolution when uniform zones continue to exist. For this reason, vortex methods have been particularly successful in resolving the evolution of shear layers. In cases when the flow is driven by body forces, that is, when the vorticity is continuously generated by the interaction between the density gradients and the pressure gradients, it is important that the vorticity source term is also evaluated accurately. Since flow gradients are involved in computing the source terms, it is important to apply compatible schemes in simulating the flow dynamics and the evolution of scalar gradients. The method presented here achieves this compatibility.

Solution of a number of buoyancy driven generic flows in three dimensions demonstrates the success of the method in resolving the temperature gradients and the corresponding vorticity structures, and in particular show its convergence. Complex vortical structures that arise in the early and late stages were reproduced.

The method shows sub quadratic convergence for the buoyant spheres because the source terms integration is first order. However, this is not an inherent limitation and one can improve the convergence order by applying second order integration for the source terms, as in the jet-in-crossflow case, in which the vortex and gradients elements have been reduced to a single kind of particle, carrying both the vorticity and temperature information. Second order interpolation kernels are also available for diffusion, and high order splitting can be used.

5.1.1. Future Work

The present algorithm can be optimized by computing the maximum time step allowed for every time step. We can either use $\|\omega\|_{max}^{-1}$, determine the maximum element strain as in Bradly et al [38] or compute the acceleration term as done by Brecht and Ferrante [30] when using a leapfrog method. In our case, a 2nd order Runge-Kutta scheme is used, so the best way to solve this issue might be to base the time step on a maximum element strain criterion.

5.2. Jet-in-Crossflow

The Transport Element Method has been used to investigate the buoyancy effects on the transverse jet problem. The non-buoyant and buoyant cases were compared by giving a complete description of the flow fields. The combination of vortex methods, the TEM and our multipurpose adaptive treecode allow us to perform accurate and time efficient simulations of the jet, the major difficulty in this problem being the accurate description and implementation of the boundary conditions at the nozzle exit for both the vorticity and the scalar gradient introduction.

5.2.1. Future Work

The present paper suggests a number of avenues for future work, the first being the development of the transverse reacting jet. Its implementation should be pretty straightforward now that both the buoyant transverse jet and the reacting algorithm have been developed. In fact, as it is done for the reacting algorithm, the scalar gradient in the transverse jet code will simply be replaced by the gradient of the Schwab-Zeldovich variable. The scalar field will be computed every time step using the treecode and the distribution of all the reacting species, the reaction front location and the temperature field will be recovered from this conserved scalar value. The density change in time will be a source of volumetric expansion.

The resulting code will be two or three times slower than the current jet-in-crossflow algorithm, but it is still acceptable since the current algorithm only required 10 hours on 3 CPUs to compute the evolution of the buoyant jet presented in 1.1.1 until $t = 4s$. The new algorithm will require a bigger amount of memory though.

Secondly, it seems really important to formulate a better model for vorticity introduction and the in-pipe flow structure. In fact, with the current implementation, we are not able to capture the hovering vortex structures in the pipe and a single vortex sheet is introduced in the flow at every time step to compensate for the vorticity produced in the jet boundary layer at the nozzle exit, the slip of the crossflow velocity over the jet orifice and the fact that the crossflow velocity does not penetrate into the jet at $y=0$. Wee and Ghoniem provide a complete description of the vorticity introduction mechanisms and shows that the wall boundary layer separation is critical in understanding jet behaviors in [39]; however, in-pipe vortical structures are still not captured.

A better analysis of the flow physics should also be realized, for both the buoyant and the future reacting jet. When studying the vorticity field, the stretching rate

$$\frac{D}{Dt}\left(\frac{1}{2}|\omega|^2\right) = \frac{\omega.(\omega.\nabla\mathbf{u})}{|\omega|^2}$$

is generally analyzed.

The same way, the "squeezing rate"

$$\frac{D}{Dt}\left(\frac{1}{2}|\vec{g}|^2\right)=\frac{-\vec{g}.(\vec{g}.\nabla\mathbf{u})}{|\vec{g}|^2}$$

should also be outputted and analyzed; $\vec{g}$ represents the transported scalar gradient here.

Many efforts have been made on elucidating the mechanisms underlying the formation of organized vortical structures in the near field and the subsequent breakdown of these structures into small scales for the non-buoyant case. We have now to understand how buoyancy will affect this vortical structure, how it will affect the mixing rate between the fluid coming from the jet and the ambient fluid. Finally, we have to understand how buoyancy will change the entrainment of the ambient fluid.

References

[1] A.J. Majda, A.L. Bertozzi, *Vorticity and Incompressible Flow*, Cambridge University Press, Cambridge, MA, 2002.

[2] G. H. Cottet and P. D. Koumoutsakos, *Vortex Methods: Theory and Practice*, Cambridge University Press, London, 2000.

[3] A. F. Ghoniem and A. K. Oppenheim. *Numerical-Solution for the Problem of Flame Propagation by the Random Element Method.* AIAA Journal. Vol.22, No.10, 1984, 1429-1435.

[4] A. F. Ghoniem and F. S. Sherman. *Grid-Free Simulation of Diffusion Using Random-Walk Methods*. Journal of Computational Physics **61**, No.1, 1985, 1-37.

[5] C.R. Anderson, *A vortex method for flows with slight density variations*, Journal of Computational Physics **61** (1985) 417.

[6] A. F. Ghoniem, G. Heidarinejad and A. Krishnan. *Numerical-Simulation of a Thermally Stratified Shear-Layer Using the Vortex Element Method.* Journal of Computational Physics **79**, No.1 (1988) 135-166.

[7] A. Krishnan and A.F. Ghoniem , *Simulation of the rollup and mixing in Rayleigh Taylor flow using the vortex/transport element method.* Journal of Computational Physics **99** (1992), 1–27.

[8] Soteriou, MC; Ghoniem, AF. 1995. *On the application of the infinite reaction rate model in the simulation of the dynamics of exothermic mixing layers*. Combustion Science and technology 105 (4-6): 377-397.

[9] Soteriou, MC; Ghoniem, AF. 1998. *On the effects of the inlet boundary condition on the mixing and burning in reacting shear flows*. Combustion and Flame 112 (3): 404-417.

[10] Soteriou, M. C., Dong, Y. and Cetegen, B. M., *Lagrangian simulation of the unsteady near field dynamics of planar buoyant plumes*, Physics of Fluids, Vol. 14, No. 9, pp. 3118-3140, 2002.

[11] Knio, OM; Ghoniem, AF. 1991. *3-Dimensional Vortex Simulation of Rollup and Entrainment in a Shear-Layer*. Journal of Computational Physics **97** (1): 172-223

[12] Knio, OM; Ghoniem, AF. 1992. *The 3-Dimensional Structure of Periodic Vorticity Layers under Nonsymmetrical Conditions*. Journal of Fluid Mechanics 243: 353-392.

[13] D. Wee and A. F. Ghoniem. *Modified interpolation kernels for treating diffusion and remeshing in vortex methods.* Journal of Computational Physics **213** (2006) 293-263.

[14] S. Stanaway, B. Cantwell, and P. Spalart, *A numerical study of viscous vortex rings using a spectral method*, TM 101004, NASA, 1988 SK Stanaway

[15] Y. M. Marzouk, D. Wee and A. F. Ghoniem. *Simulations of high Reynolds number transverse jets and analysis of the underlying vortical structures.* 43rd AIAA Aerospace Sciences Meeting; Reno, NV; 10-14 January 2005.

[16] Wickelmans, G.S. and Leonard. *Contributions to vortex particles methods for 3-dimensional incompressible unsteady flows.* Journal of Computational Physics **109**,247(1993).

[17] R. S. Scorer, *Experiments on convection of isolated masses of buoyant fluid*, J. Fluid Mech, 2, 583.(1957)

[18] R. S. Scorer, *Natural Aerodynamics.* Pergamon Press, London.(1958)

[19] C. P. Wang, *Motion of an isolated buoyant thermal*, Phys. Fluids, **14**, 1643 (1971). 3.

[20] S Lin, L. Tsang, and C. P. Wang, *Temperature Field Structure in Strongly Heated Buoyant Thermals,* Phys. Fluids **15**(12), 2118 (1972).

[21] J.S. Turner, *Buoyant vortex rings.* Proc. Roy. Soc. London Ser. A, 239, 61(1957).

[22] J.S. Turner, *Buoyancy Effects in Fluids, Cambridge University Press,* Cambridge(1973).

[23] M.P. Escudier and T. Maxworthy, *On the motion of turbulent thermals. J. Fluid Mech.* **61,** 541-552(1973).

[24] D. L. Marcus and J. B. Bell, *The structure and evolution of the vorticity and temperature fields in thermals*, Theoret. Comput. Fluid Dynamics 3: 327-344 (1992)

[25] G. R. Baker, D. I. Meiron, and S. A. Orszag, *Vortex simulations of the Rayleigh–Taylor instability*, Phys. Fluids **23**(8), 1485 (1980).

[26] R. M. Kerr, *Simulation of Rayleigh–Taylor flows using vortex blobs*, J. Comput. Phys. **76**, 48 (1988).

[27] G. Tryggvason, Numerical *simulations of the Rayleigh-Taylor instability*, J. Comput. Phys. **75**, 253 (1988).

[28] J. A. Zufiria, *Vortex-in-cell simulation of bubble competition in Rayleigh–Taylor instability, Phys. Fluids*, **31**(11), 3199 (1988).

[29] J. A. Zufiria, *Linear analysis of the vortex-in-cell algorithm applied to Rayleigh–Taylor instability*, J. Comput. *Phys.* **80**, 387 (1989).

[30] S. G. Brecht and J. R. Ferrante, *Vortex-in-cell calculations in three dimensions*, Comput. Phys. Commun. **58**, 25 (1990).

[31] S. H. Brecht and J. R. Ferrante, *Vortex-in-cell simulation of buoyant bubbles in three dimensions*, Phys. Fluids *A* **1**(7), 1166 (1989).

[32] J. H. Walther and P. Koumoutsakos, *Three-Dimensional Vortex Methods for Particle-Laden Flows with Two-Way Coupling*, Journal of Computational Physics **167,** 39–71 (2001).

[33] K. Lindsay and R. Krasny. *A particle method and adaptative treecode for vortex sheet motion in three-dimensional flow*. Journal of Computational Physics **172,** 879-907(2001).

[34] Y. M. Marzouk and A. F. Ghoniem. *K-means clustering for optimal partitioning and dynamic load balancing of parallel hierarchical N-body simulations.* Journal of Computational Physics. **207**, 493-528(2005).

[35] I. Lakkis and A. F. Ghoniem. *Axisymmetric vortex method for low-Mach number, diffusion controlled combustion.* Journal of Computational Physics **184**, 435-475(2003).

[36] G. Tryggvason, W. J. A. Dahm. *An integral method for mixing, chemical reactions, and extinction in unsteady strained diffusion layers*. Combustion and Flame 83 (1991) 207-220.

[37] C. H. H. Chang, W. J. A. Dahm, G. Tryggvason. *Lagrangian model simulations of molecular mixing, including finite rate chemical reactions, in a temporally developing shear layer.* Phys. Fluids A 3 (5) (1991) 1300-1311.

[38] M. Brady, A. Leonard, D. I. Pullin, *Regularized vortex sheet evolution in three dimensions*, J. Comput. Phys. 146 (1998) 520-545.

[39] D. Wee, PhD Thesis, *Lagrangian Simulation of Transverse Jets with a distribution-based Diffusion Scheme*, Massachusetts Institute of Technology, Cambridge, MA, 2007.