Vorticity Structure and Evolution in a Transverse Jet with New Algorithms for Scalable Particle Simulation

by

Youssef Mohamed Marzouk

S.B., Massachusetts Institute of Technology (1997)
S.M., Massachusetts Institute of Technology (1999)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

© Massachusetts Institute of Technology 2004. All rights reserved.

Author .....
Department of Mechanical Engineering
7 May 2004

Certified by .....
Ahmed F. Ghoniem
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUL 20 2004
LIBRARIES

BARKER

Vorticity Structure and Evolution in a Transverse Jet with New Algorithms for Scalable Particle Simulation

by

Youssef Mohamed Marzouk

Submitted to the Department of Mechanical Engineering
on 7 May 2004, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

Transverse jets arise in many applications, including propulsion, effluent dispersion, oil field flows, V/STOL aerodynamics, and drug delivery. Furthermore, they exemplify flows dominated by coherent structures that cascade into smaller scales, a source of many current challenges in fluid dynamics. This study seeks a fundamental, mechanistic understanding of the relationship between the dispersion of jet fluid and the underlying vortical structures of the transverse jet—and of how to develop actuation that optimally manipulates their dynamics to affect mixing.

We develop a massively parallel 3-D vortex simulation of a high-momentum transverse jet at large Reynolds number, featuring a discrete filament representation of the vorticity field with local mesh refinement to capture stretching and folding and hairpin removal to regularize the formation of small scales. A novel formulation of the vorticity flux boundary conditions rigorously accounts for the interaction of channel vorticity with the jet boundary layer. This formulation yields analytical expressions for vortex lines in near field of the jet and suggests effective modes of unsteady actuation at the nozzle. The present computational approach requires hierarchical N-body methods for velocity evaluation at each timestep, as direct summation is prohibitively expensive. We introduce new clustering algorithms for parallel domain decomposition of N-body interactions and demonstrate the optimality of the resulting cluster geometries. We also develop compatible techniques for dynamic load balancing, including adaptive scaling of cluster metrics and adaptive redistribution of their centroids. These tools extend to parallel hierarchical simulation of N-body problems in gravitational astrophysics, molecular dynamics, and other fields.

Simulations reveal the mechanisms by which vortical structures evolve; previous computational and experimental investigations of these processes have been incomplete at best, limited to low Reynolds numbers, transient early-stage dynamics, or Eulerian diagnostics of essentially Lagrangian phenomena. Transformation of the cylindrical shear layer emanating from the nozzle, initially dominated by azimuthal vorticity, begins with axial elongation of its lee side to form sections of counter-rotating vorticity aligned with the jet trajectory. Periodic rollup of the shear layer

accompanies this deformation, creating arcs carrying azimuthal vorticity of alternating signs, curved toward the windward side of the jet. Following the pronounced bending of the trajectory into the crossflow, we observe a catastrophic breakdown of these sparse periodic structures into a dense distribution of smaller scales, with an attendant complexity of tangled vortex filaments. Nonetheless, spatial filtering of this region reveals the persistence of counter-rotating streamwise vorticity. We further characterize the flow by calculating maximum direct Lyapunov exponents of particle trajectories, identifying repelling material surfaces that organize finite-time mixing.

Thesis Supervisor: Ahmed F. Ghoniem
Title: Professor of Mechanical Engineering

Acknowledgments

Thanks are due first to my advisor, Ahmed Ghoniem, for his many years of support, motivation, and insightful guidance. Since convincing me to stay at MIT for graduate school seven years ago, he has made the ensuing time incredibly worthwhile. I have benefited from his clear thinking and technical intuition, and I am grateful for his mentorship, patience, and friendship. I would also like to thank the members of my thesis committee—John Bush, George Haller, and Jacob White—for their enthusiasm, insight, and support. I feel privileged to have involved each of them in my thesis work.

I would like to acknowledge my colleagues at the Reacting Gas Dynamics Lab, beginning with the "old guard" who welcomed me when I first joined the lab: Jean-Pierre Hathout, Issam Lakkis, Adrin Gharakhani, and Shankar Subramaniam. Sung-bae Park, Adam Wachsman, and Susanne Essig aren't old enough to be in the old guard, but they too have helped make my time in the lab truly collegial. I have enjoyed a fruitful collaboration and many insightful discussions with Daehyun Wee; I hope these do continue. And over the past year, I've also been fortunate to spend time with the "new guard" who will take over the office upon my departure: Zia Sobhani, Ray Speth, Murat Altay.

I would also like to acknowledge the help of Leslie Regan and Joan Kravit in the ME graduate office, who have kept everything running smoothly for me and so many other ME graduate students. Thanks are also due to Julie Rioux and Lorraine Rabb.

The support of many friends helped me through graduate school and has enriched life immeasurably along the way. It's rare that I get to acknowledge my friends in writing, but I can think of few factors that have been more important, so here goes. Han Shu brought camaraderie to late nights in the office and late-night dinners and has been the source of many inspiring technical discussions. Emma Dentinger conceived of our informal Cambridge thesis support group, which grew to include fellow dissertation-writers Jane Sohn and Eric Statz; one has to experience graduate school to fully understand it. I was gratified when Tarek El Aguizy came back to MIT just in time for the two of us to finish our degrees simultaneously. Charley

Able has been an enduring friend; Marcelo Targino, Damani Walton, Andy Wheeler, Misha Davidson, Rich Diaz, Elsa Zuniga, Dan Dobbs and many others have filled my time in Boston and Cambridge—particularly the last two years living at BDC—with memorable experiences along the winding path to responsible adulthood. I'll miss the community we all shared.

I bear an immense debt of gratitude to my parents, who have been constant, patient, and supportive through a long educational career and all the "real life" that accompanies it. I also thank my sister Shaden for her shining example of professional commitment and for realistic, sage advice that I only occasionally heed.

From September 1997 to June 2002 I was generously supported by a graduate fellowship from the Fannie and John Hertz Foundation. This work was also supported by the US Department of Energy, Office of Science, Mathematical, Information, and Computational Sciences (MICS) Program. Computational support was provided by the National Energy Research Supercomputing Center (NERSC). I also thank Keith Lindsay and Robert Krasny for making the source of their serial treecode publicly available.

Contents

List of Figures

Chapter 1

Introduction

The mixing properties of the transverse jet—a jet issuing normally into a uniform crossflow—are important to a variety of applications. Transverse jets may function as sources of fuel in industrial furnaces, or as diluent jets for blade cooling or exhaust gas cooling in industrial or airborne gas turbines. Other industrial applications include pipe tee mixers and jets of oil and gas entering the flow in oil wells. Transverse jets have also been studied extensively for their relevance to V/STOL aerodynamics, roll control of missiles, and environmental problems such as pollutant dispersion from chimneys or the discharge of effluents into the ocean.

Enhancement of the mixing rate between jet and crossflow can lead to significant improvements in many performance aspects. In gas turbines, for instance, better transverse jet mixing is essential to achieving a wider range of operability, lower emissions, smaller size, and lower noise output. The ultimate objective of this work is to develop actuation strategies for the transverse jet that optimally manipulate the mixing rate between the jet fluid and the crossflow.

Effective actuation must be grounded in a clear understanding of flow structures, their dynamics, and their impact on mixing. The transverse jet presents many challenges in this regard. The near field of the flow is dominated by coherent vortical structures; further downstream, these structures exhibit a critical transition to smaller scales. Mechanisms underlying both the formation of vortical structures in the near field and their eventual breakdown are largely unresolved. This thesis focuses on *eluci-*

dating these mechanisms and characterizing their impact on transport and dispersion of jet fluid.

Our approach to a mechanistic understanding of the flow is computational. We develop a three-dimensional vortex simulation of the high-momentum transverse jet at large Reynolds number, featuring a discrete filament description of the vorticity field with local mesh refinement to capture stretching and folding and hairpin removal to regularize the formation of small scales. Previous computational and experimental investigations of vorticity dynamics in the transverse jet have been limited to low Reynolds numbers [70, 76], transient early-stage dynamics [30], or Eulerian diagnostics of essentially Lagrangian phenomena [132, 118]. Vortex methods thus provide an attractive framework for simulation of the transverse jet, first of all for their explicit link to the formation and dynamics of vortical structures in the flow. Vorticity introduced at the boundary is tracked through the flow field, providing a clear, mechanistic view of its evolution. Moreover, vortex methods are well-suited to high Reynolds number flows for their ability to simulate convection without numerical diffusion, through the advection of Lagrangian computational elements. Also, inherent in the grid-free nature of the method is a dynamic clustering of computational points only where they are needed, i.e., over the small support of the vorticity field.

The present simulations employ a new formulation of vorticity flux boundary conditions that rigorously describes the interaction of crossflow boundary layer vorticity with jet vorticity. Analysis of the flow is aided by extracting and examining the evolution of material lines and vorticity isosurfaces, streamlines, and trajectories; these efforts encompass comparison with both experiment and similarity analysis. We further characterize the flow by calculating maximal Lyapunov exponents directly from particle trajectories, identifying repelling surfaces that organize finite-time mixing.

A complementary component of this work focuses on the computational challenges presented by highly-resolved vortex particle simulations. The primary cost of vortex methods is incurred in evaluation of the vortical velocity field. Velocities induced by all the vortex elements must be evaluated at each vortex element through solution of a Poisson equation. The result is an N-body problem on an irregular particle distri-

bution of non-uniform density; similar N-body problems arise in astrophysics (e.g., gravitational cluster interactions) and molecular dynamics. Direct solution of these problems yields a computational complexity of $O(N^2)$, which becomes prohibitive for large N. Yet simulations with $N = 10^6$ or greater are necessary for resolution and scale.

Hierarchical methods for N-body problems, detailed in Chapter 4, reduce this computational complexity to $O(N \log N)$ or $O(N)$. Many sequential hierarchical methods have been developed [8, 50, 104], but parallel methods expose additional computational issues, chief among these the impact of domain geometry and cell geometry on computational cost. This thesis introduces new algorithms, based on k-means clustering, for partitioning of parallel hierarchical N-body interactions. We also develop new heuristics for dynamic load balancing, including adaptive scaling of cluster metrics and adaptive redistribution of cluster centroids. These techniques prove quite effective when applied to realistic massively parallel N-body simulations. Moreover, results suggest that clustering can replace traditional oct-tree partitioning schemes to optimize the computational efficiency of serial hierarchical N-body algorithms.

1.1 Physics of the transverse jet

Numerous experiments and computations over the past fifty years have addressed the trajectory, scaling, and structure of jets in crossflow [80]. In the following, we review some of these results and highlight unresolved areas, providing context for the flow physics explored by our vortex simulations.

1.1.1 Flow parameters and coherent structures

The structure of the flow field is primarily governed by the jet-to-crossflow momentum ratio

$$r = \left(\frac{\rho_j V_j^2}{\rho_\infty U_\infty^2} \right)^{1/2} \tag{1.1}$$

written here as an effective velocity ratio, where ρ_j and V_j are the density and mean velocity of the jet, while ρ_∞ and U_∞ are the density and velocity of the crossflow.

Another controlling parameter is the jet Reynolds number, defined as $Re_j = V_j d/\nu$, where d is the jet diameter and ν is the kinematic viscosity. While only a few studies have discussed the effect of Reynolds number *independently* of r, it is generally considered to have at best a secondary effect on the large-scale structures of the jet flow field [70, 132]. A comparison of experimental results obtained at different Re_j throughout the literature confirms this assertion; in fact, the similarity analysis in [58] invokes Reynolds number invariance. Trajectory correlations discussed in §1.1.2 below are independent of Reynolds number. Broadwell and Breidenthal [18] argue that the flame length of the transverse jet—a measure of its mixing rate—is independent of Re_j above a certain critical value, a behavior typical of free shear flows.

Numerous studies report the presence of large-scale "coherent" vortical structures in the flow field [112, 58]. Experimental observations by [44] identify four such structures in the transverse jet, shown schematically in Figure 1-1: jet shear layer vortices; "wake vortices" arising from interaction between the jet and the channel wall boundary layer; horseshoe vortices that wrap around the jet exit; and a counter-rotating vortex pair that forms as the jet bends into the crossflow, persisting far downstream.

Jet shear layer vortices result from Kelvin-Helmholtz instability of the cylindrical shear layer shed from the edge of the jet orifice. A water-tunnel dye visualization of these vortices, adapted from [94], is shown in Figure 1-2. Kelso *et al.* [70] report that shear layer roll-up is limited to the upstream side of the jet for smaller Re, while for crossflow Reynolds number $Re_{cf} \equiv U_\infty d/\nu > 1000$ large scale roll-up occurs along the entire perimeter of the shear layer. These structures are analogous to the vortex ring structures typically observed in free jets. Chapter 3 of this thesis will characterize the evolution of the jet shear layer in greater detail.

The horseshoe vortices develop close to the wall, wrapping around the nascent column of jet fluid. These vortices form as the wall boundary layer upstream of the jet encounters an adverse pressure gradient and separates [44]. Several studies have elucidated the complex streamline topology of the horseshoe system [73, 71, 70];

these studies also observe periodicity in the formation and roll-up of the vortices. Different parameter regimes may see this unsteadiness coupling with the dynamics of wake vortices and with intermittent separation in the jet nozzle [71, 73]. Nonetheless, Kelso *et al.* [70] assert that "the horseshoe vortex system seems to play only a minor role in the overall structure" of transverse jet flow field.

Downstream of the orifice, an alternating pattern of upright or "wake" vortices extends from the wall to the bending jet. These vortices have been studied extensively using various flow-visualization techniques [44, 70] along with hot-wire anemometry [102]. Fric and Roshko [44] demonstrate that though the wake vortex pattern is reminiscent of Kármán shedding behind a solid cylinder, its origins are fundamentally different. Vorticity is not shed from the jet column into the wake; rather, the wake vortices consist of vorticity generated in the crossflow boundary layer. "Separation events" in the wall boundary layer downstream of the jet form upright vortices that lift fluid from the wall toward the main flow of the jet. For larger r, however, the wake structures connecting the wall to the jet fluid are weakened [112]; Fric and Roshko [44] find that the wake vortices are most orderly at $r = 4$, then diminish in strength as r increases.

For very low r, e.g., $r < 1$, interactions of the jet fluid with the wall boundary layer downstream of the orifice result in a qualitatively different flow structure; this regime is examined in [60] for a series of circular, elliptical, and rectangular nozzle geometries. We also note that *elevated* transverse jets have a different wake structure than jets flush with the wall. In this case, there is evidence that jet-wake vortices couple with similar Kármán-like structures in the wake of the stack [41].

The last of the four vortical structures listed above is the counter-rotating vortex pair (CVP). The CVP is a robust feature of the flow over all parameter ranges (e.g., r, Re) and has been a focus of numerous studies [30, 99, 66, 5, 4, 70, 76]. Broadwell and Breidenthal [18] argue that the impulse of the jet normal to the crossflow results in a streamwise counter-rotating vortex pair. This argument views the jet as a point source of momentum, and does not explain the vorticity transformation mechanisms that actually create the CVP in the near field. (Previous work aimed at understanding

these mechanisms is reviewed in §1.1.3; we elucidate these mechanisms more fully in Chapter 3.) Though the CVP is present in the mean flow, it has significant unsteady components [102] and its instantaneous structure may be strongly asymmetric [112].

With regard to the design of actuation strategies, many of the dynamic characteristics of the transverse jet are unknown. For free jets or co-flowing jets, for instance, the jet natural modes are well-known, and actuation typically consists of exciting the jet at a corresponding frequency or harmonic. Analogous modes for the transverse jet and their relation to jet properties largely have yet to be determined [30]. A few studies have characterized dominant frequencies in the wake of the unforced jet [73, 44]. Experimental studies of pulsed jets in crossflow [65, 43, 88] have identified pulsing frequencies and duty cycles or characteristic temporal pulse widths [87] that maximize jet penetration into the crossflow. These actuations tend to create discrete vortex rings that propagate deep into the crossflow [20], a flow structure that is qualitatively different from that of the unforced jet. The relationship between optimal pulsing and the transverse jet's preferred modes or shear layer dynamics has not been rigorously examined, however, particularly over a range of r.

1.1.2 Trajectories, length scales, and similarity analysis

The trajectory of the transverse jet has long been the subject of experimental measurements and analytical predictions. Many experimental correlations can be collapsed to power-law form [80, 58]:

$$\frac{y}{rd} = A\left(\frac{x}{rd}\right)^B \tag{1.2}$$

Values reported in the literature are in the range $1.2 < A < 2.6$ and $0.28 < B < 0.34$ [58]. Pratte and Baines [99] report $A = 2.05, B = 0.28$ for $r = 5$ to $r = 35$; the 0.28 exponent of this power-law fit has been corroborated in computational simulations [131]. Variation in the coefficients A and B may stem from different definitions of the jet trajectory. Some researchers use the streamline emanating from the center of the orifice; others use the locus of maximum velocity on the centerplane, and still others use the locus of maximum scalar concentration. Kamotani and Greber [66] note that

the trajectory based on maximum local velocity penetrates 5–10% deeper into the flow than the trajectory based on scalar concentration. Another source of scatter in the correlations is the determination of r, since the jet velocity U_j is not perfectly uniform at the jet exit. Hasselbrink and Mungal [58] argue that the velocity ratio r is best defined in the integral sense, as an average momentum flux per unit area of the jet.

Analytical predictions of the jet trajectory are pursued in [117, 18, 67]. Of particular interest are approaches based on the kinematics of the vorticity field. The inviscid model of Broadwell and Briedenthal [18] models the lift associated with a counter-rotating vortex pair in the far field, obtaining a 1/3 power law in rd-coordinates. Karagozian [67] also obtains a 1/3 power law, using a two-dimensional model of a viscous vortex pair.

All of the above trajectories—whether based on experimental measurements or analytical models—are strictly valid only in the far field of the jet. A recent similarity analysis by Hasselbrink and Mungal [58] provides a more precise delineation of "far field" and "near field" in this context. The authors argue that the transverse jet contains two regions of intermediate-asymptotic similarity. In the far field, for $y/rd \gg 1$, the centerline trajectory follows a 1/3 power law:

$$\frac{y_c}{rd} = \left(\frac{3}{c_{ew}} \frac{x_c}{rd} \right)^{1/3} \tag{1.3}$$

where c_{ew} is a far field entrainment coefficient. In the near field, for $y/d \gg 1, x/rd \ll 1$ the centerline trajectory obeys a 1/2 power law:

$$\frac{y_c}{rd} = \left(\frac{2}{c_{ej}} \frac{x_c}{rd} \right)^{1/2} \tag{1.4}$$

Here c_{ej} denotes a near-field entrainment coefficient. Intermediate-asymptotic behavior is confirmed via measurements of mean velocity and velocity-fluctuation profiles along the jet in [59].

Underlying the trajectory correlations and similarity analysis presented above are

several length scales that describe the jet scaling. Events near the orifice scale with the jet diameter d [29]. Away from this region, the most important global length scale [18, 67] is

$$L = \left(\frac{\dot{m}_j V_j}{\rho_\infty U_\infty^2} \right)^{1/2} \sim rd \tag{1.5}$$

where $\dot{m}_j$ is the mass flux of the jet. This rd-scaling underlies almost all the trajectory scaling laws. A third length scale is revealed by scalar concentration measurements in [112]; a branch point in the decay of centerline concentrations for various r collapses when normalized by $r^2 d$. Keffer and Baines [69] also collapse jet trajectory data for $r = 6$ to $r = 10$ using this length scale. Thus $r^2 d$ may play the role of an outer momentum length scale.

1.1.3 Vorticity generation and evolution

The transverse jet presents several subtle physical issues of relevance to mixing and dynamic response to actuation. Chief among these is the origin of the counter-rotating vortex pair (CVP). Differing accounts of the mechanism by which the counter-rotating vortices form still persist. Recent experimental work [70, 43, 76] suggests that the CVP is initiated just above the jet exit as jet shear layer vorticity folds onto itself and Kelvin-Helmholtz instability leads to a simultaneous roll-up. A water-tunnel dye visualization of the folding shear layer is shown in Figure 1-3. The resulting flow pattern can be interpreted as the tilting and folding of vortex rings as they are ejected from the nozzle, where the downstream side of each vortex ring is approximately aligned with the jet trajectory. Various other studies support this view [118, 20, 58, 29]. A different mechanism in [132] points to quasi-steady "hanging vortices" formed in the skewed mixing layers on lateral edges of the jet; the authors suggest that an adverse pressure gradient causes these vortices to break down into a weak CVP. Scalar concentration measurements in [112] indicate that CVP development is delayed with higher r.

Water-tunnel flow visualizations [70] suggest that the CVP also contains vorticity generated in the channel wall boundary layer. Though the relative magnitude of this

contribution must decrease with higher r, it has not been clear whether jet shear layer vorticity alone is sufficient to characterize the dynamics of the CVP. These questions will be addressed in the present work through careful construction of vorticity flux boundary conditions.

1.2 Objectives

The present modeling efforts focus on coherent vortical structures present in the main flow—the jet shear layer and the counter-rotating vortex pair—rather than those linked to the wall, as the former are most relevant to entrainment and mixing. We would like to capture the dynamics of these structures at high Reynolds number, to understand their formation mechanisms and follow them downstream as they mature. For simplicity, we focus on incompressible flow, and for relevance to mixing in engineered systems we consider $r \gg 1$.

The objectives of this work are as follows:

- To develop and validate a three-dimensional vortex simulation of the transverse jet, accurately incorporating vorticity generation mechanisms from first principles and capturing the formation and evolution of large-scale vortical structures.

- To characterize vortex dynamics in the high-momentum ($r \gg 1$) transverse jet. We seek a mechanistic understanding of the formation of coherent vortical structures and of their subsequent breakdown into small scales. To this end, develop appropriate methods for extracting material surfaces, integral quantities (e.g., streamlines and trajectories), and vorticity field diagnostics. Compare results to scaling laws and experiment.

- To analyze the mixing characteristics of the transverse jet using velocity data as well as recent tools for extracting distinguished Lagrangian structures [56]. Explore the impact of vortical structures—and their evolution—on mixing.

- To identify mechanisms for actuating the flow field and formulate boundary

conditions that describe these mechanisms. Where possible, develop reduced-order models for the response of the flow to actuation inputs.

- To develop numerical tools for fast, accurate high-resolution vortex simulation. These include local remeshing schemes to resolve the cascade to small scales on vortex filaments and hairpin removal to prevent a numerical over-proliferation of elements. These tools *also* include scalable parallel algorithms for hierarchical evaluation of vortical velocities, for which we will demonstrate applicability to general dynamic N-body problems.

More broadly speaking, this research seeks to develop the computational tools and the physical understanding required for control and optimization of mixing in three-dimensional vortical flows.

Chapter 2 begins with a discussion of vorticity transport in inviscid, incompressible flows, presenting three-dimensional vortex particle methods and details of our filament construction. This chapter also formulates vorticity flux boundary conditions for the transverse jet and extends the formulation to an analytical description of vorticity in the near field.

Chapter 3 presents simulation results revealing mechanisms of vorticity transformation in the transverse jet. We describe the formation and eventual breakdown of vortical structures, discussing our results in the context of earlier experimental, theoretical, and computational studies. We also use notions from dynamical systems to characterize the mixing properties of the jet, relating these properties to vortical structures.

Chapter 4 describes new clustering algorithms for partitioning and dynamically load-balancing parallel hierarchical N-body interactions. We evaluating the impact of these algorithms on computional speed and accuracy via a number of realistic test cases.

Conclusions and a sketch of future work are given in Chapter 5.

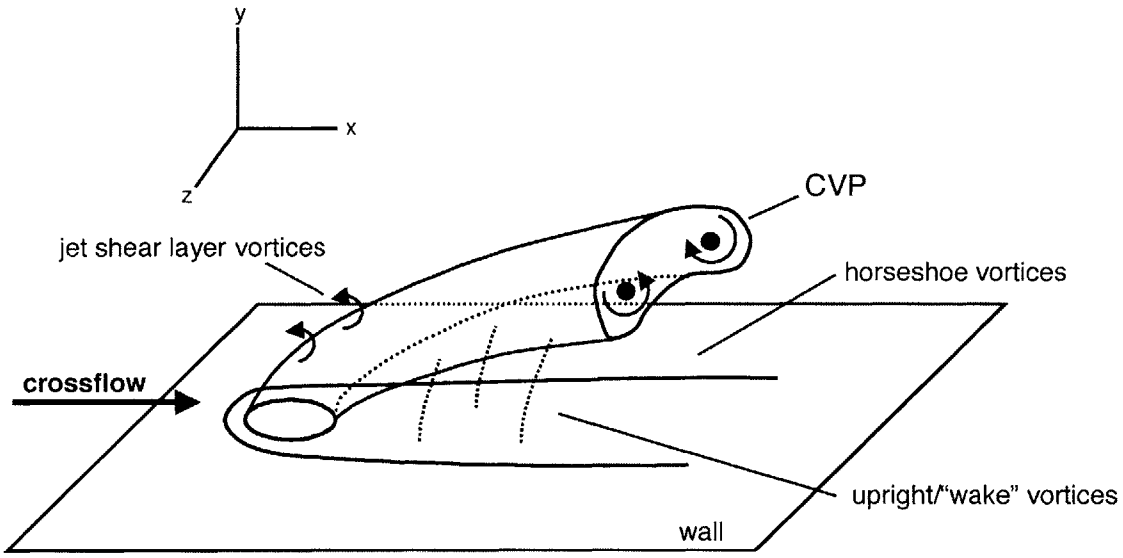


Figure 1-1: Coherent vortical structures in the transverse jet; schematic after Fric and Roshko [44].

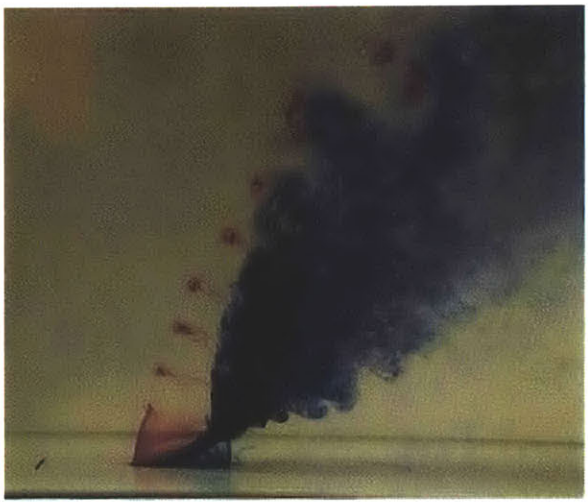


Figure 1-2: Water-tunnel dye visualization of a transverse jet at $r = 4.0$, $Re_j = 6400$. Blue dye is released from a circumferential slot 1.6 diameters upstream of the jet exit; red dye is released from a single port 0.06 diameters upstream of the jet exit. Reproduced from Perry, Kelso, and Lim [94].

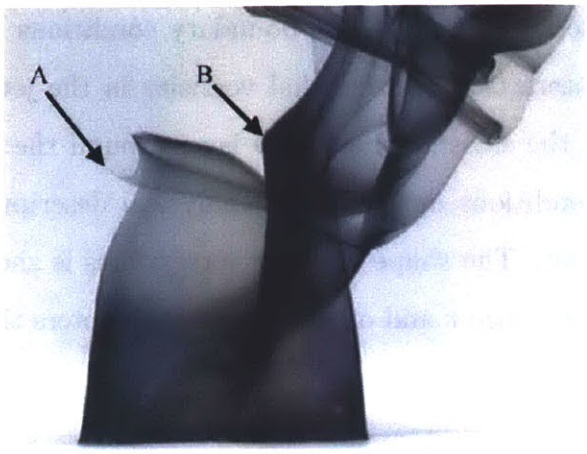


Figure 1-3: Water-tunnel dye visualization of a transverse jet at $r = 4.6$, $Re_j = 1600$, showing folding of cylindrical shear layer. Reproduced from Lim, New, and Luo [76].

Chapter 2

Vorticity Formulation for the Transverse Jet

This chapter presents governing equations for the flow field of the transverse jet and describes solution of these equations with a three-dimensional vortex particle method. Our interests center on incompressible transverse jets at high Reynolds number, and thus we focus on the inviscid transport of vorticity. The dominant role of inviscid dynamics is supported by experimental and computational evidence summarized in the preceding chapter.

We also formulate new vorticity flux boundary conditions for the transverse jet, accounting for the interaction of azimuthal vorticity in the jet boundary layer with spanwise vorticity in the crossflow boundary layer around the jet orifice. Derivation of these boundary conditions motivates an analytical description of vortex lines in the near field of the jet. The shape of these vortex lines is shown to depend on the jet-to-crossflow velocity ratio r and on additional parameters that describe actuation at the jet nozzle.

2.1 Numerical formulation

2.1.1 Governing equations

Equations of motion for inviscid, incompressible flow may be written in vorticity transport form, where $\boldsymbol{\omega} = \nabla \times \mathbf{u}$:

$$\frac{D\boldsymbol{\omega}}{Dt} = \boldsymbol{\omega} \cdot \nabla \mathbf{u} \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.2}$$

In this Lagrangian description, the right-hand side of (2.1) accounts for stretching and tilting of the vorticity as it is convected by the flow.

Using the Helmholtz decomposition of the velocity field, we write

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_p \tag{2.3}$$

where $\mathbf{u}_\omega$ is the curl of a vector potential ($\mathbf{u}_\omega = \nabla \times \boldsymbol{\psi}$) and $\mathbf{u}_p$ is the velocity of a potential flow ($\mathbf{u}_p = \nabla\phi$). It follows from (2.3) that the vector potential and the vorticity are related by a Poisson equation:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} = \nabla \times \nabla \times \boldsymbol{\psi} = -\nabla^2 \boldsymbol{\psi} \tag{2.4}$$
$$\text{where } \nabla \cdot \boldsymbol{\psi} = 0$$

The vector potential $\boldsymbol{\psi}$ is not uniquely determined by this system, and can always be chosen divergence-free by imagining an extension of the fluid to a domain where $\boldsymbol{\omega} \cdot \mathbf{n} = 0$ at the boundary [10]. Given a distribution of vorticity $\boldsymbol{\omega}$, the vortical velocity $\mathbf{u}_\omega$ may thus be recovered from the Biot-Savart law

$$\mathbf{u}_\omega(\mathbf{x}) = -\frac{1}{4\pi} \int_D \frac{(\mathbf{x} - \mathbf{x}') \times \boldsymbol{\omega}(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|^3} d\mathbf{x}' = \mathbf{K} * \boldsymbol{\omega} \tag{2.5}$$

Here $\mathbf{K}$ denotes the matrix-valued Biot-Savart kernel.

The above equations are closed by choosing a divergence-free potential velocity

field to satisfy a prescribed normal velocity $\mathbf{n} \cdot \mathbf{u}$ on the boundary of the given domain D:

$$\begin{aligned} \nabla^2 \phi &= 0 \\ \mathbf{n} \cdot \nabla \phi &= \mathbf{n} \cdot \mathbf{u} - \mathbf{n} \cdot \mathbf{u}_\omega \text{ on } \partial D \end{aligned} \tag{2.6}$$

Together, (2.1), (2.2), (2.3), (2.5), and (2.6) completely specify the motion of an incompressible, inviscid fluid [26].

2.1.2 Three-dimensional vortex particle methods

We formulate a three-dimensional vortex method for simulations of an unsteady, incompressible transverse jet at large Reynolds number.

Vortex methods are a computational approach to systems governed by the Navier-Stokes or Euler equations, employing a particle discretization of the vorticity field and transporting vorticity along particle trajectories [75, 72, 79, 32, 100]. Originally conceived of for high Reynolds number flows [22] and for flows dominated by vortex dynamics [103], these methods have received significant attention over the past thirty years, maturing into tools for direct simulation, supported by several convergence results and a rigorous error analysis [52, 53, 54, 11, 12, 3, 32].

The essence of a vortex method is the discretization of the vorticity field onto Lagrangian computational elements, or particles. In three dimensions, these particles have vector-valued weights $\boldsymbol{\alpha}_i(t) \equiv (\boldsymbol{\omega} dV)_i (t)$ and trajectories $\boldsymbol{\chi}_i(t)$.

$$\boldsymbol{\omega}(\mathbf{x}, t) \approx \sum_i^N \boldsymbol{\alpha}_i(t) f_\delta (\mathbf{x} - \boldsymbol{\chi}_i (t)) \tag{2.7}$$

The vorticity associated with each element is desingularized with a radially-symmetric core function $f_\delta (\mathbf{r})$ of radius δ, where $f_\delta (\mathbf{r}) \equiv \delta^{-3} f \left(\frac{|\mathbf{r}|}{\delta} \right)$. The function f must be smooth and rapidly decreasing, satisfying the same moment properties as the Dirac measure up to order $m > 1$ [13].

Given a regularized particle discretization of the vorticity field as in (2.7), the

Biot-Savart law (2.5) may be rewritten as follows:

$$\mathbf{u}_\omega(\mathbf{x}) = \sum_i^N \mathbf{K}_\delta\left(\mathbf{x}, \boldsymbol{\chi}_i\right) \boldsymbol{\alpha}_i \tag{2.8}$$

where the regularized kernel $\mathbf{K}_\delta$ results from componentwise convolution with the core function, $\mathbf{K}_\delta(\mathbf{x}) = \mathbf{K} * f_\delta(\mathbf{x})$. In the present simulations, we employ the Rosenhead-Moore kernel [77, 103]

$$\mathbf{K}_\delta\left(\mathbf{x}, \mathbf{x}'\right) = -\frac{1}{4\pi} \frac{\mathbf{x} - \mathbf{x}'}{\left(|\mathbf{x} - \mathbf{x}'|^2 + \delta^2\right)^{3/2}} \times \tag{2.9}$$

which corresponds to the low-order algebraic core function[1] [128]:

$$f(\rho) = \frac{3}{4\pi} \frac{1}{\left(\rho^2 + 1\right)^{5/2}} \tag{2.11}$$

Vortex methods solve the equations of motion via numerical integration for the particle trajectories $\boldsymbol{\chi}_i(t)$ and weights $\boldsymbol{\alpha}_i(t)$. Computing particle trajectories $\boldsymbol{\chi}_i(t)$ requires evaluation of the velocity at each particle at every timestep:

$$\frac{d\boldsymbol{\chi}_i}{dt} = \mathbf{u}(\boldsymbol{\chi}_i) \tag{2.12}$$

Three-dimensional vortex methods also require evaluation of velocity gradients to compute the vortex stretching term and thus the evolution of particle weights. For inviscid flows, ODEs for $\boldsymbol{\alpha}_i(t)$ follow directly from (2.1):

$$\frac{d\boldsymbol{\alpha}_i}{dt} = \boldsymbol{\alpha}_i \cdot \nabla \mathbf{u}\left(\boldsymbol{\chi}_i, t\right) \tag{2.13}$$

The right-hand side of (2.13) may be evaluated through explicit differentiation of

[1] Strictly speaking, this core function does not satisfy the moment condition

$$\int_0^\infty |f(\rho)|\, \rho^{2+r} d\rho < \infty \tag{2.10}$$

for $r > 0$, and thus the usual convergence results may not apply. However, it has been used effectively as a mollifier for numerous vortex particle simulations [77, 75]; based on this practical evidence, we employ it here.

the Biot-Savart kernel [3] or by application of a finite-difference operator [12]. The present calculations evalute vorticity stretch in the context of vortex filaments; more details on this construction will be provided in the next section.

A discussion of the various convergence results for inviscid vortex particle methods is beyond the scope of this thesis. However, it is important to note that error norms expressing convergence to smooth solutions of the Euler equations go to zero as the number of particles increases and the core size δ decreases, subject to the constraint that the typical interparticle spacing $h \rightarrow 0$ faster than δ; effectively, this requires the cores of neighboring particles to *overlap*. For convergence proofs and rigorous error analyses, the reader is referred to [52, 53, 54, 11, 12, 3, 31, 62]; reviews can be found in [79, 100, 32].

The Lagrangian vortex method provides an attractive model of the transverse jet, first of all for its explicit link to the formation and dynamics of vortical structures in the flow. Vorticity introduced at the boundary is tracked through the flow field, providing a clear, mechanistic view of its evolution. Moreover, because convection exactly corresponds to the advection of Lagrangian computational elements, the errors associated with vortex methods contain minimal numerical dissipation [32], rendering these methods well-suited to high Reynolds number flows. Finally, inherent in the grid-free nature of the method is a dynamic clustering of computational points only where they are needed, i.e., over the small support of the vorticity field.

Additional, important physics have been built on the foundation outlined above. New particle methods for solving the diffusion equation, coupled with viscous splitting, have extended the applicability of vortex methods to flows of finite Reynolds number [45, 34, 35, 109, 86], enabling direct simulation of the Navier-Stokes equations, including boundary layer phenomena. We also mention a range of techniques for dealing with complex boundaries [96, 97], as well as extensions to stratified flows [113], aeroacoustics [42], and reacting flows [114, 115, 74].

2.1.3 Vortex filament methods

Consider the evolution of a material line element $\boldsymbol{\delta\chi}$ in a velocity field $\mathbf{u}(\mathbf{x}, t)$:

$$\frac{D\boldsymbol{\delta\chi}}{Dt} = \boldsymbol{\delta\chi} \cdot \nabla \mathbf{u} \tag{2.14}$$

Comparing this equation with (2.1), we observe that in inviscid incompressible flows the motion of material lines corresponds to the evolution of vortex lines [10]. This is Helmholtz's theorem for inviscid incompressible flow; a material element coinciding with a vortex line remains on that vortex line for all time.

In a vortex *filament* scheme, the overall vorticity field is viewed as a collection of vortex filaments; each filament consists of a vorticity field concentrated on a curve $\mathbf{r}_j(p)$ that is either closed or extending to infinity. The circulation of each filament is constant at all cross sections and, in accordance with Kelvin's theorem, unchanged as the filament is transported by the flow. From a vortex element point of view, we can discretize the filament along its one-dimensional parameterization, writing:

$$\boldsymbol{\omega}_i dV_i = \Gamma_j \boldsymbol{\delta\chi}_i \tag{2.15}$$

where the circulation is indexed by the filament number j and is *constant in time.* On a given filament, connectivity should be maintained between neighboring vortex elements. In place of the original discretization in (2.7), we now have

$$\boldsymbol{\omega}(\mathbf{x}, t) \approx \sum_j \sum_i^{N_j} \Gamma_j \boldsymbol{\delta\chi}_i(t) f_\delta \left(\mathbf{x} - \boldsymbol{\chi}_i^c(t)\right) \tag{2.16}$$

Filament methods present several numerical advantages over ordinary three-dimensional vortex particle methods. They preserve the fundamental invariants of three-dimensional inviscid flow, conserving total circulation, impulse, and helicity, and maintaining the solenoidal nature of the vorticity field. They also allow efficient evaluation of vorticity stretch: rather than updating $\boldsymbol{\omega}_i dV_i$ explicitly according to (2.13), as in a vortex particle method, one must simply keep track of the deformation of the filament (i.e.,

$\delta\boldsymbol{\chi}_i(t)$), since vortex lines and material lines coincide.

It is possible to enforce only a *local* correspondence between vortex lines and material lines—that is, to initialize vortex elements so that they coincide with specific material lines without requiring that the elements be arranged along continuous closed (or infinite-extent) vortex lines. In this case, we do not have a true filament method; each vortex element may locally coincide with a different vortex line, and thus have a different Γ. However, we can still write $\boldsymbol{\omega}_i dV_i = \Gamma_j \delta\boldsymbol{\chi}_i$ and take advantage of the coincidence of vortex lines and material lines to evaluate vorticity stretch. Discretization of the vorticity field still takes the form of (2.16). In this construction, we refer to the computational elements as lying on "partial filaments" or "vortex sticks." These methods relinquish some of the unique conservation properties of filament methods, but they allow greater flexibility in the initialization of the vorticity.

Vortex sticks can be interpreted as ordinary vortex particles that employ a finite-difference stencil oriented in the direction of the local vorticity vector to evaluate the vorticity stretch term. However, they differ from ordinary vortex particles in that, like filament methods, they provide a straightforward facility for local remeshing, ensuring that core overlap is maintained along the direction of the vorticity. Remeshing and other forms of "filament surgery" will be described in §2.1.4.

The present computations describe each filament j—whether it is a partial filament or a true, closed filament—by a finite set of nodes $\{\boldsymbol{\chi}_i\}_j$. Rather than using these nodes to construct a piecewise linear description of filament geometry, we use a cubic spline interpolant to describe each space-curve [7]. Each filament $\mathbf{r}_j(p)$ is parameterized by the accumulated chord-length between nodes. That is,

$$\Delta p_i \equiv p_{i+1} - p_i = \left|\boldsymbol{\chi}_{i+1} - \boldsymbol{\chi}_i\right| \tag{2.17}$$

For simulations employing closed vortex filaments, we use a periodic cubic spline interpolant for each Cartesian component of $\mathbf{r}_j(p)$, eliminating any ambiguity in the specification of end conditions [33]. With vortex sticks or partial filaments, we specify "not-a-knot" end conditions for the cubic splines [33]. If a filament contains only three

nodes, quadratic interpolants are used; if only two nodes are present, we resort to linear interpolation.

With the vorticity field described in this manner, the mollified Biot-Savart law may be re-written as follows:

$$\mathbf{u}_\omega(\mathbf{x}) = -\frac{1}{4\pi}\sum_j \left(\Gamma_j \int \frac{(\mathbf{x}-\mathbf{r}(p))}{|\mathbf{x}-\mathbf{r}(p)|^3} \times \frac{d\mathbf{r}}{dp}\, \kappa_\delta\left(\mathbf{x}-\mathbf{r}(p)\right) dp \right) \tag{2.18}$$

where $\kappa_\delta(\mathbf{r}) \equiv \kappa(\frac{|\mathbf{r}|}{\delta})$,

$$\kappa(\rho) = 4\pi \int_0^\rho f(s)s^2 ds \tag{2.19}$$

and $f(s)$ is the core function in (2.16). Note that the mollified Biot-Savart kernel $\mathbf{K}_\delta$ in (2.9) is equal to

$$\mathbf{K}_\delta\left(\mathbf{x},\mathbf{x}'\right) = -\frac{\kappa_\delta\left(\mathbf{x}-\mathbf{x}'\right)}{\left|\mathbf{x}-\mathbf{x}'\right|^3}\left(\mathbf{x}-\mathbf{x}'\right) \times \tag{2.20}$$

The midpoint rule is used for quadrature of the Biot-Savart law as written above (2.18). This is analogous to applying the Biot-Savart law directly to the summation over vortex elements in (2.16), taking $\boldsymbol{\delta\chi}_i = d\mathbf{r}/dp|_{p_{i+1/2}}\,\Delta p_i$ and $\boldsymbol{\chi}_i^c = \mathbf{r}\left(p_{i+1/2}\right)$.

Nodes on each filament are advected by the velocity field, as specified in (2.12). A second-order predictor/corrector method is used for time integration of these ODEs (Euler predictor, trapezoidal corrector). Two criteria are used for timestep control during integration. The first estimates the position error in advecting each node and forces the maximum error estimate, over all the nodes, to be less than some fraction α of the core radius δ:

$$\max_i\left(\frac{err_{\boldsymbol{\chi}_i}}{\delta}\right) < \alpha \tag{2.21}$$

The second criterion requires that each element travel no further than a single core radius over the course of one timestep, in effect ensuring that remeshing of the filaments occurs frequently enough to maintain core overlap at all times:

$$\Delta t < \frac{\delta}{\max_i |\mathbf{u}(\boldsymbol{\chi}_i)|} \tag{2.22}$$

2.1.4 Mesh refinement and hairpin removal

Cubic spline representations for $\mathbf{r}_j(p)$ are recomputed from the advected nodes at each timestep. As the filaments stretch and fold in response to the strain field of the flow, a mesh refinement scheme must be implemented to ensure core overlap, as well as an adequately detailed description of filament geometry. When $|\boldsymbol{\delta\chi}_i|$ of a given element, as computed from the cubic spline representation, exceeds a given fraction of the core radius, e.g., 0.9δ, a new node is added at the midpoint of the element, thus splitting the element in two. The location of the new node is also computed from the cubic spline representation: $\boldsymbol{\chi}^{\text{new}} = \boldsymbol{\chi}_i^c = \mathbf{r}\left(p_{i+1/2}\right)$, and, after advection, the new node is used to compute subsequent spline interpolants.

One result of this mesh refinement scheme is a continuous (in fact exponential) growth in the number of vortex elements used to resolve the vorticity field, corresponding to the generation of smaller length scales in the flow. We employ filament-based hairpin removal algorithms to curb the numerical proliferation of small length scales [23, 24]. These algorithms directly remove small-scale folds in vortex lines (i.e. "hairpins"), yet have been demonstrated to preserve the dynamical characteristics of large-scale vortical structures and to conserve integral quantities of the flow, like kinetic energy and linear impulse [24]. By considering the statistical mechanics of vortex filaments at the inviscid limit, hairpin removal can be justified as a renormalization procedure [28, 25, 27].

Hairpins are identified by computing the angle between adjacent vortex elements; for simplicity, we use the chords between neighboring nodes to compute these angles. When the angle exceeds a certain maximum, given by the parameter $\cos(\theta)_{\text{min}}$, the pair of elements is replaced by its vector sum. More complicated rules are needed to deal with adjacent hairpins on a single filament. If the number n of adjacent hairpins is odd, we remove hairpins at the odd-numbered nodes, e.g., $\{1, 3, \ldots, n\}$. If the number of adjacent hairpins is even, we search for the sharpest hairpin and remove it along with its neighboring even interleaf. For instance, if the sharpest hairpin occurs at node m in a group of $m < n$ adjacent hairpins, we remove hairpins

at nodes $i \in \{\ldots, m-2, m, m+2, \ldots\}$, for $1 \leq i \leq n$. Hairpin removal may also require multiple passes; removing one set of hairpins may leave, or even produce, another. Our current implementation does not search for hairpins between elements on *different* filaments, however; thus explicit filament splitting and reconnection as described in [24] are not pursued.

In addition to splitting elements and removing hairpins, we merge small elements with their neighbors along a filament if the linear extent of the element becomes too small, e.g., for $|\boldsymbol{\delta\chi}_i| < 0.2\delta$. As with hairpin removal, we devise separate rules to contend with even and odd numbers of adjacent small elements. In practice, we find that hairpin removal is significantly more important than small-element merging in curbing the proliferation of vortex elements and in regularizing filament geometry.

2.1.5 Parallel implementation

The present computations are implemented on a massively parallel distributed memory computer using message passing, via the standard MPI libraries. Two sets of parallel calculations, with two corresponding data distributions, are employed in the code.

The first set of parallel calculations centers on the vortex filaments. All of the following tasks are performed in parallel: calculation of cubic spline representations for $\mathbf{r}_j(p)$ and of all the quantities derived from cubic spline representations (e.g. element centers, particle weights); mesh refinement along the filaments; hairpin removal; small element merging. Domain decomposition in this case is simply a block distribution of the filaments; each processor is assigned approximately an equal number of filaments on which to operate. Since filaments may vary significantly in size, load balance is not perfect. In this case, however, good load balance is not particularly important. When parallelized, filament-based calculations represent less than 5% of the overall computational expense of the code.

The second set of parallel calculations is significantly more complicated and computationally demanding. To speed the evaluation of the velocity field, we use an adaptive treecode developed by Lindsay and Krasny [77]. Parallel implementation of

this treecode, or indeed any hierarchical solver, presents a number of computational and geometric challenges. We introduce clustering algorithms for parallel domain decomposition in this context, as well as new heuristics for dynamic load balancing. Chapter 4 presents this development in detail.

Simulations reported in this thesis contain as many as 3.5×10^6 vortex elements.

2.2 Boundary conditions

We now discuss boundary conditions particular to the simulation of the transverse jet flow field—both normal-velocity boundary conditions and vorticity flux boundary conditions. In the subsequent expressions, all variables are made dimensionless by d, the jet diameter, and U_∞, the velocity of the uniform crossflow. The crossflow is directed in the positive x direction; the jet centerline is aligned with the y axis; and the z axis is in the spanwise direction. Except on the disc of the jet orifice, the x-z plane is taken to be a solid wall through which we enforce a no normal-flow boundary condition.

2.2.1 Normal-velocity boundary conditions

The jet outflow is represented by a semi-infinite cylindrical vortex sheet of radius $1/2$ extending from $y = 0$ to $y = -\infty$, with strength $\boldsymbol{\gamma} = 2r\hat{\mathbf{e}}_\theta$. The vorticity in this cylinder is mollified by a core function *identical* to that used with the computational vortex elements. This matching is crucial. An unmollified cylindrical vortex sheet, or, equivalently, a uniform distribution of potential sources over the jet orifice with surface source strength $2r$, yields a singularity in the radial velocity at the nozzle edge when paired with the computational vortex elements. This singularity was noted, and left uncorrected, in [30]. We discretize the vortex sheet using vortex particles with core radius, axial spacing, and azimuthal spacing identical to the vortex particles introduced into the flow, as shown in Figure 2-1.

The crossflow velocity is given by the potential $\phi_\infty = x$. Image vortex elements are used to model the behavior of vorticity in the semi-infinite domain, i.e., to enforce

no-flow through the channel wall $y = 0$. Writing the vorticity in the domain componentwise $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$, the image vorticity has components $\boldsymbol{\omega}_{\text{img}} = (-\omega_x, \omega_y, -\omega_z)$.

2.2.2 Boundary generation of vorticity

Vorticity produced in the jet boundary layer (i.e., in the pipe below the $y = 0$ plane) is represented by a single sheet of azimuthal vorticity. Introducing this vorticity into the flow every Δt_{noz} time units, we divide it among n_θ vortex elements distributed along the edge of the jet nozzle, where $\Delta\theta = 2\pi/n_\theta$. These elements have weight

$$(\boldsymbol{\omega}_i dV_i)_o = -\frac{r^2}{4}\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_\theta \tag{2.23}$$

where $\hat{\mathbf{e}}_\theta$ is the tangential unit vector in the x-z plane.

Upstream of the jet, vorticity produced in the channel wall boundary layer initially points in the negative spanwise ($-\hat{\mathbf{z}}$) direction. Our interest lies in the interaction of this vorticity with the jet flow immediately around the nozzle edge; in particular, we wish to model channel wall vorticity carried upward by the jet, as this is the vorticity that will affect the evolution of the jet trajectory over the range of r. Thus we do not attempt to resolve events in the wall boundary layer away from the jet nozzle, as these have a diminished role in determining jet dynamics for $r > 1$ [112].

By considering the slip of crossflow velocity over the edge of the jet orifice, or, alternatively, the penetration of crossflow velocity into the jet fluid at the wall, we now derive perturbations to the vortex element strengths given in (2.23). These perturbations, due to channel wall boundary layer vorticity, are $O(r)$ rather than $O(r^2)$ [83].

First, consider the slip of crossflow velocity over the edge of the jet orifice. In polar coordinates (r, θ) centered at the origin of the x-z plane, the radial component of this slip velocity is canceled locally by an azimuthal vortex sheet of strength $\boldsymbol{\gamma} = -\cos\theta\hat{\mathbf{e}}_\theta$. These vortex sheets are shed a distance $r\Delta t_{noz}/2$ into the flow every timestep. Again dividing this vorticity over $n_\theta = 2\pi/\Delta\theta$ elements distributed along the nozzle edge,

we obtain

$$(\boldsymbol{\omega}_i dV_i)_1 = -\frac{r}{4}\cos\theta_i \Delta t_{noz} \Delta\theta \hat{\mathbf{e}}_\theta \tag{2.24}$$

for the vortex element strengths due to this interaction.

Next we observe that crossflow velocity does not penetrate into the jet at $y = 0$. This requires a velocity discontinuity in the $\hat{\mathbf{e}}_\theta$ direction, which corresponds to a vortex sheet of strength $\boldsymbol{\gamma} = \sin\theta\hat{\mathbf{e}}_y$. Thus the interaction of crossflow vorticity with the jet results in wall-normal vorticity; this idea is confirmed heuristically by considering the tilting of a spanwise material line that encounters either spanwise extremity of the jet. Again, we expect these vortex sheets to be shed at the local flow velocity, i.e., $r\Delta t_{noz}/2$ every timestep. Dividing the vorticity over elements along the nozzle edge, we obtain element strengths as:

$$(\boldsymbol{\omega}_i dV_i)_2 = \frac{r}{4}\sin\theta_i \Delta t_{noz} \Delta\theta \hat{\mathbf{e}}_y \tag{2.25}$$

A final constraint arises from kinematic considerations. In cylindrical coordinates, for vorticity confined to a sheet emanating from the nozzle edge, the solenoidality constraint on the vorticity field $\nabla \cdot \boldsymbol{\omega} = 0$ requires

$$\frac{\partial \omega_y}{\partial y} = -2\frac{\partial \omega_\theta}{\partial \theta} \tag{2.26}$$

Each new set of vortex elements represents vorticity in the flow for $0 < y < r\Delta t_{noz}/2$. We thus introduce elements so that their centers lie at $y = r\Delta t_{noz}/4$. Summing jet and channel-wall boundary layer contributions to vortex element strengths and enforcing (2.26), we obtain the following expression for the total strength of the vortex elements introduced at each timestep:

$$\begin{aligned}\boldsymbol{\omega}_i dV_i = &\left(-\frac{r^2}{4} - \frac{r}{4}\cos\theta_i\right)\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_\theta \\ &+ \left(\frac{r}{4}\sin\theta_i - \frac{r^2\Delta t_{noz}}{8}\sin\theta_i\right)\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_y\end{aligned} \tag{2.27}$$

It is worthwhile to contrast the present vorticity formulation with other vortex

models of the transverse jet. Our earlier computational effort [82] neglected vorticity in the crossflow boundary layer, focusing only on the evolution of jet azimuthal vorticity; this approximation is discussed therein and its effect will be noted in the results section below. Another recent vortex filament simulation of the unforced transverse jet [30] enforces a no-slip boundary condition along the channel wall by modifying the uniform crossflow with a cubic boundary layer profile near the wall. This boundary layer profile corresponds to a finite vorticity, yet this vorticity is not allowed to evolve, i.e., to obey the dynamics of equation (2.1), nor is it carried into the main flow by the jet.

2.2.3 Closed vortex filaments in the near field

As a further modeling step, we provide a description of continuous, closed vortex filaments representing the vorticity field derived above. For the purposes of detailed simulation, closed vortex filaments are numerically convenient; they guarantee that the numerical representation of the vorticity field remains solenoidal and conserve many fundamental invariants of three-dimensional inviscid flow [32]. Knowledge of filament geometry also provides a deeper physical understanding of the flow, particularly in the context of filament folding mechanisms shown to be responsible for formation of the counter-rotating vortex pair [82, 43] (see Chapter 3).

From the derivation detailed above, we can write the vorticity field near the nozzle, i.e., for $0 < y \ll 1$, as:

$$\begin{aligned}\boldsymbol{\omega}_i dV_i = &\left(-\frac{r^2}{4} - \frac{r}{4}\cos\theta_i\right)\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_\theta \\ &+ \left(\frac{r}{4}\sin\theta_i - y\frac{r}{2}\sin\theta_i\right)\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_y \\ &= \Gamma_i \boldsymbol{dl}_i(\theta, y)\end{aligned} \tag{2.28}$$

Dividing by the desired filament circulation $\Gamma = \frac{r^2}{2}\Delta t$, we seek integral curves of the vector field

$$\boldsymbol{l}(\theta, y) = -\left(1 + \frac{\cos\theta}{r}\right)\hat{\mathbf{e}}_\theta + \frac{\sin\theta}{r}(1 - 2y)\,\hat{\mathbf{e}}_y \tag{2.29}$$

The resulting space curve $\boldsymbol{\zeta}(s) = p(s)\hat{\mathbf{e}}_\theta + q(s)\hat{\mathbf{e}}_y$ can be obtained as the solution of two ODEs:

$$\begin{aligned} \frac{dp}{ds} &= -\left(1 + \frac{\cos p}{r}\right) \\ \frac{dq}{ds} &= \frac{\sin p}{r}\left(1 - 2q\right) \end{aligned} \tag{2.30}$$

A solution of these coupled ODEs can in fact be written analytically. Choosing initial conditions so that $(\theta, y) = (\pi, 0)$ is a point on $\boldsymbol{\zeta}(s)$, we have:

$$p(s) = -2\arctan\left(\mathcal{I}\sqrt{\frac{r+1}{r-1}}\right) \tag{2.31}$$

$$q(s) = \frac{2r}{(r-1)^2(r+1)^2}\left(r - \frac{1-\mathcal{I}^2}{1+\mathcal{I}^2}\right)^2 + \frac{2\mathcal{I}^2\left(\mathcal{I}^2 r + r - 1\right)}{(r-1)^2\left(1+\mathcal{I}^2\right)^2} \tag{2.32}$$

where

$$\mathcal{I} \equiv \tan\left(\frac{s}{2}\sqrt{1 - \frac{1}{r^2}}\right) \tag{2.33}$$

and the solution is $\left(2\pi/\sqrt{1-1/r^2}\right)$-periodic in the parameter s. For larger r, the resulting vortex filament is flat and ring-like, as jet azimuthal vorticity dominates; for smaller r, the vortex filament is more "kinked" in the y-direction. The geometry of the initial vortex filament entering the flow is shown in Figure 2-2 for $r = 5$ and in Figure 2-3 for $r = 15$.

The vortex filament construction here can be generalized to arbitrary perturbations to the primary jet azimuthal vorticity—that is, arbitrary vorticity actuations at the nozzle edge.

$$\begin{aligned} \boldsymbol{\omega}_i dV_i = &\left(-\frac{r^2}{4} + f(\theta)\right)\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_\theta \\ &+ \left(g(\theta) - 2y f'(\theta)\right)\Delta t_{noz}\Delta\theta\hat{\mathbf{e}}_y \end{aligned} \tag{2.34}$$

Here, $f(\theta)$ is a perturbation to the azimuthal vorticity and $g(\theta)$ is perturbation to the axial vorticity; both functions could be $O(r)$. Once again, solution of simple ODEs, analogous to (2.30), yields the geometry of the closed vortex filaments entering the

flow at the jet orifice. This construction provides a compact, physically revealing description of key actuation inputs. The shape and circulation of filaments entering the flow depends explicitly on the distribution of axial and azimuthal vorticity along the nozzle edge and on the jet-to-crossflow velocity ratio r.

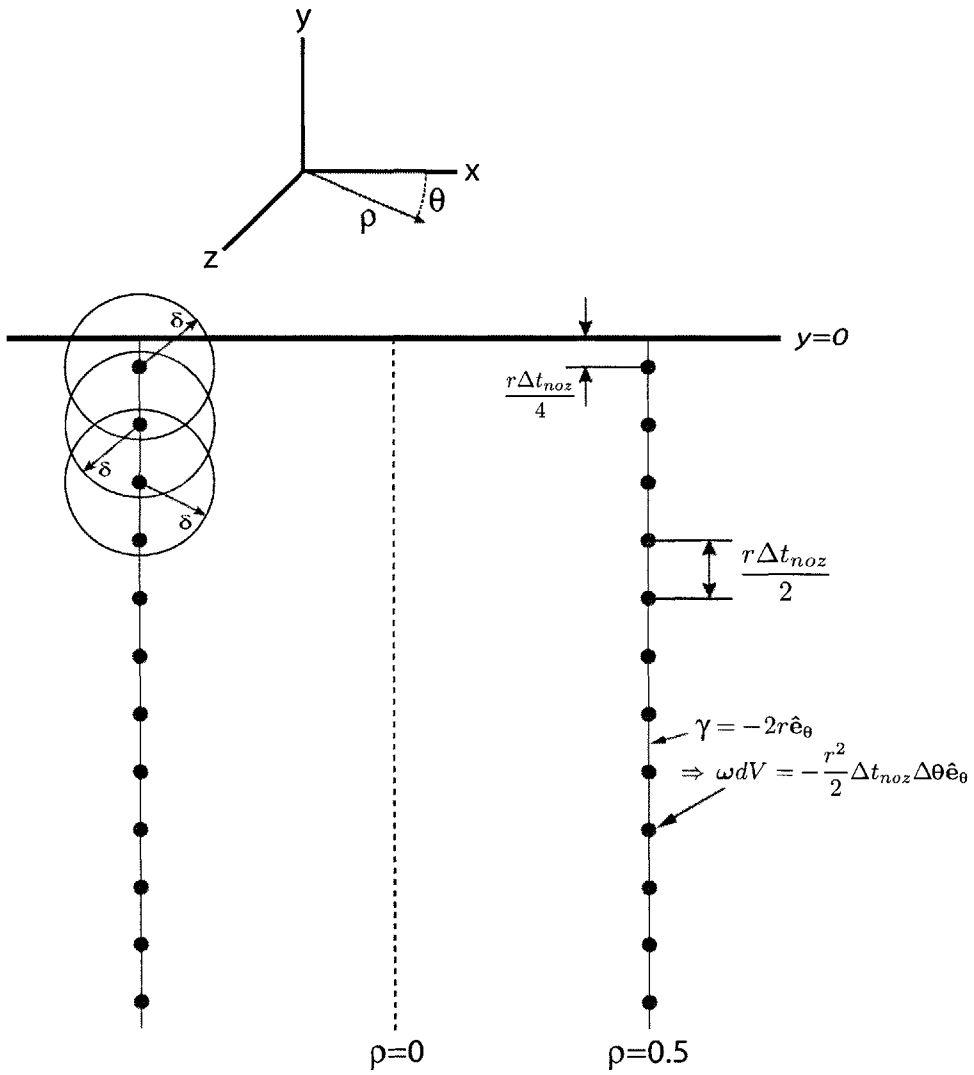


Figure 2-1: Discretization of the cylindrical vortex sheet representing the jet outflow for $y > 0$.

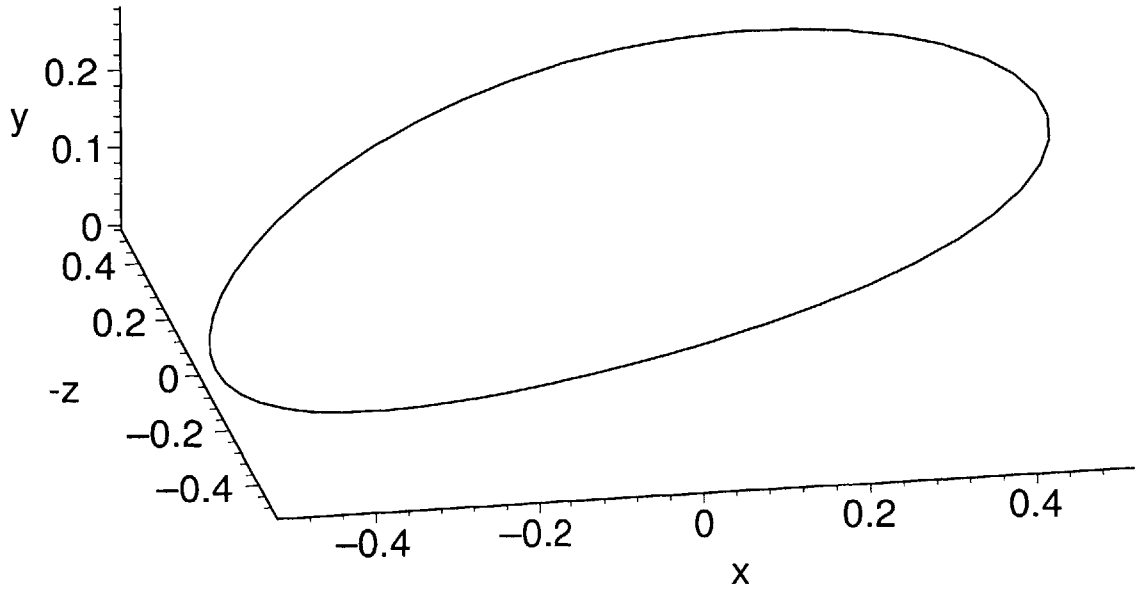


Figure 2-2: Initial geometry of closed vortex filaments in the transverse jet, for $r = 5$.

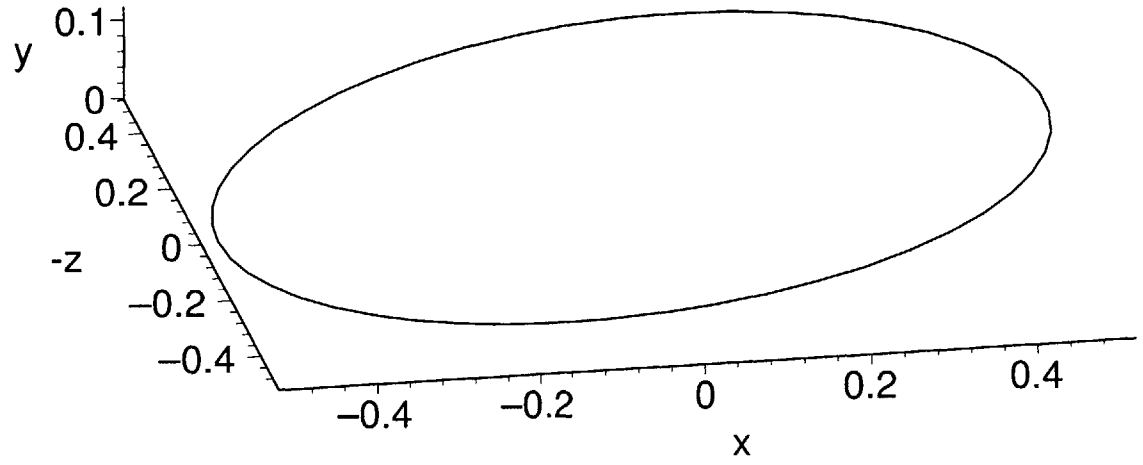


Figure 2-3: Initial geometry of closed vortex filaments in the transverse jet, for $r = 15$.

Chapter 3

Results: Vorticity Dynamics

We now present the results of vortex simulations of the spatially evolving transverse jet, at velocity ratios r ranging from 5 to 10. Our goal is to understand the structure of the vorticity field and to elucidate the mechanisms giving rise to this structure. While the literature has seen much emphasis on the formation of the counter-rotating vortex pair (CVP), we find that this process is tightly coupled to dynamics involving unsteady transformations of the initially cylindrical jet shear layer into a rich and varied set of vortical structures, and a concurrent "cascade" of large length scales into small scales as the jet evolves downstream.

Analysis of the flow is aided by extracting and examining the evolution of material lines carrying vorticity, instantaneous and averaged vorticity isosurfaces, streamlines, and trajectories. These efforts encompass comparison with similarity theory, other computational results, and experiment. We will also examine the impact of the vorticity flux boundary conditions derived in the previous chapter and comment on the validity of analytical expressions for the near-field vortex lines.

Finally, we compute the field of maximal finite-time Lyapunov exponent associated with particle trajectories for selected regions and time intervals in the $r = 7$ jet. This field, calculated directly from particle paths, identifies repelling material surfaces that organize finite-time mixing.

3.1 Numerical parameters

Numerical parameters for the vortex simulations were chosen as follows: Overall spatial resolution is governed by the core radius δ, chosen to be 0.1. The number of elements discretizing vorticity introduced along the nozzle edge (e.g., the initial azimuthal resolution) is $n_\theta = 64$. Axial resolution depends on the timestep between successive filament introductions, Δt_{noz}. We keep the distance $r\Delta t_{noz}$ relatively constant for different choices of r. Thus, for the $r = 7$ jet we put $\Delta t_{noz} = 0.01$; for $r = 10$ we put $\Delta t_{noz} = 0.0075$; and for $r = 5$ we put $\Delta t_{noz} = 0.0125$.

The length threshold for splitting elements is fixed at 0.9δ, while the length threshold for merging small elements along a filament is 0.2δ. We fix the cutoff for hairpin removal at $\cos(\theta)_{\min} = 0.0$. The error tolerance parameter α controlling the integration timestep in (2.21) is set to 0.01.

A series of numerical convergence studies were performed to justify the above choices of numerical parameters. Filament geometries were observed to be unaffected by further reduction of the error tolerance parameter α. The jet trajectory as well as the shape and location of large-scale vortical structures in the flow were unaffected by relaxation of the hairpin removal cutoff $(\cos\theta)_{\min}$. Similarly, the trajectories, vortical structures, and dynamical processes reported below were preserved under further refinement of the spatial resolution parameters δ, n_θ, and Δt_{noz}, suggesting that the present simulations are well-resolved at the chosen values.

3.2 Vorticity transformation mechanisms

3.2.1 Vortex filament evolution

We first examine successive snapshots of computational vortex filaments in transverse jets at various jet-to-crossflow velocity ratios r. Filament geometries provide a clear overview of the vortical structures in the starting jet and of the dynamical processes that accompany formation of these structures. Figure 3-1(a)-(e) shows filaments in 3-D perspective for the $r = 5$ jet. Times $\tilde{t}$ corresponding to each snapshot are

normalized by d/U_∞, the crossflow convective time scale. Figure 3-2(a)-(e) shows analogous snapshots for the $r = 7$ jet and Figure 3-3(a)-(e) shows filaments in the $r = 10$ jet. Times for the various snapshots were chosen to cover the entire computed evolution of each jet and to allow comparison of different jets at identical crossflow convective times $\tilde{t}$. Times were also selected to show the r-dependence of jet evolution at identical *jet* convective times $t/(d/rU_\infty)$. For instance, the $r = 5$ jet at $\tilde{t} = 2.40$, Figure 3-1(c), is at the same jet convective time as the $r = 10$ jet at $\tilde{t} = 1.20$, shown in Figure 3-3(c).

Several important features are apparent in these figures. The most obvious is that the jet penetrates more deeply into the domain for larger r. The envelope of the $r = 10$ jet is more upright than that of the $r = 7$ and $r = 5$ jets, and although all the jets are deflected by the crossflow in the positive $x-$direction, the larger-r jet begins significant tilting much later in its evolution—later in the sense of both jet convective time and wall-normal distance y/d.

Next, all three cases show roll-up of the jet shear layer, indicative of the expected Kelvin-Helmoltz instability. Shear layer roll-up is indicated by the axial grouping-together of vortex filaments and is particularly visible on the upstream (windward) side of the jet. On the lee side of each jet, a more complex out-of-plane distortion of the vortex filaments accompanies the roll-up. This distortion holds the key to the development of counter-rotating vorticity and will be explored more fully in the next subsection. As the jets evolve further, filaments stretch and fold in response to flow strain and are continually remeshed with a growing number of nodes. The growing number of computational elements thus reflects the mechanism by which smaller length scales are generated in the flow. At all values of r, large-scale vortical structures undergo a critical transition at the head of each jet, breaking up to form a "mushroom cloud" dominated by complex, shorter-length-scale vortical interactions. This transition is accompanied by more pronounced bending into the crossflow. While this far-field region appears solid black in the present black-and-white plots, it contains structure which we will elucidate with additional diagnostics (see §3.2.3 and §3.2.4).

We also note that, though the jets continue to evolve downstream, the near field envelope of each jet seems to mature. Comparing Figure 3-2(d) and Figure 3-2(e), for instance, the jet envelope seems unchanged for $x/d < 2.0$. At this stage, the *initial* orientation of the jet is normal to the wall for all three values of r.

3.2.2 Ring folding and counter-rotating vorticity

The out-of-plane distortion of vortex filaments may be analyzed more carefully by following the evolution of individual segments of the jet shear layer. Consider first the $r = 7$ jet: we count nine distinct roll-up events on the windward side of the jet as it evolves from $\tilde{t} = 0$ to $\tilde{t} = 2.60$. The fate of the ring-like vortical structures formed in the first two roll-ups differs qualitatively from that of subsequent structures. This is not surprising, as the earlier rings initially interact with a very different vorticity field than the later rings. The first vortex ring, shown forming at head of the jet in Figure 3-2(a), encounters no vorticity downstream. Vorticity later introduced into the flow is affected significantly by vorticity already in the field; while this vorticity also forms ring-like structures, these structures are stretched and deformed by existing vorticity as they in turn affect the evolution of pre-existing and subsequent vortical structures.

Let us examine this process step-by-step, remaining focused on the $r = 7$ jet, though we will later show that the evolution of vorticity is similar for all the values of r considered. The first roll-up, occurring at the head of the shear layer in Figure 3-2(a), gives rise to a vortex ring that remains strong in the flow for subsequent times. This ring can be seen near the head of the jet in Figure 3-2(b) for instance, 0.60 convective times later, though it has pulled additional vorticity through its center. The ring stays relatively planar as it tilts into the crossflow, inducing deformations of the vorticity-carrying material around it that contribute to the complicated structure in the mushroom cloud.

Next, in Figure 3-4 we show only those vortex elements that were introduced between $\tilde{t} = 0.45$ and $\tilde{t} = 0.52$. As detailed in Chapter 2, elements take the form of locally-defined vortex filaments or "sticks." Though they primarily carry azimuthal

vorticity, these filaments actually represent multiple components of vorticity whose relative strengths vary along the azimuthal coordinate in accordance with (2.27). These filaments grow in length and are remeshed independently of each other in response to flow-induced stretch. Their initial arrangement, however, is essentially along a ring, and the filaments collectively maintain this coherence as they evolve. Thus it is meaningful to speak of the geometric transformations of a vorticity-carrying material "ring" when describing the collective evolution of this group of vortex elements.

The segment of the shear layer shown in Figure 3-4(a) participates in the second roll-up on the windward side of the $r = 7$ jet. But the evolution of the entire material ring is significantly more complex than a single roll-up. This evolution is traced in Figure 3-4(b)–Figure 3-4(e) with four snapshots, equally spaced in time. The shear layer first deforms out-of-plane, as shown in Figure 3-4(b); here, the downstream (lee) side of the shear layer has stretched upwards to form a tongue-like structure. This deformation can be attributed to velocity induced by the preceding vortex ring; above and slightly downstream of the filaments shown, the first ring induces an upward velocity on the lee side of the ensuing shear layer. In Figure 3-4(c), the tongue-like structure becomes more pronounced and the filaments group together; the shear layer is rolling up, and the roll-up centers on the filaments we have selected here. At the very top of the tongue-like structure, however, a new deformation is present: vortex elements are curving towards the windward side of the jet. This deformation is due at least in part to the vorticity carried by lower portions of the filaments. Vertical arms of the tongue-like structure carry counter-rotating vorticity—pointing upwards for $z > 0$ and downwards for $z < 0$. Material elements between the counter-rotating arms are transported backwards (in the negative x-direction), normal to the vorticity. As the ring evolves further, its upper portion flattens and the entire structure takes the form of two vortex arcs connected by vertical arms. While the arms are approximately aligned with the jet trajectory in Figure 3-4(d), they curve against the crossflow in Figure 3-4(e); this final stage of deformation is unique to this shear layer segment, due to interactions with the first vortex ring, and is not repeated as the jet near-field matures.

Subsequent segments of the shear layer undergo a series of deformations reminiscent of those just described, but settle into a repeating pattern. This pattern may be summarized by tracing the evolution of two complementary groups of elements. Figure 3-5 is representative of the first group, showing vortex elements introduced into the flow between $\tilde{t} = 1.09$ and $\tilde{t} = 1.14$, in five successive snapshots each separated by 0.20 time units. The start of upward deformation on the lee side of the shear layer is shown in Figure 3-5(b). Upward deformation becomes more pronounced in Figure 3-5(c). In contrast to the shear layer segment considered in the preceding paragraph, it is the lee side of the present shear layer segment that rolls up most strongly; the windward side remains sheet-like. Vertical arms form below the lee-side roll-up on either side, but their length is at most one diameter. As with the earlier tongue-like structure, the vertical arms carry counter-rotating vorticity essentially aligned with the jet trajectory. In Figure 3-5(d) the lee side roll-up begins curving towards the windward side of the jet; at this stage, the lee-side roll-up is a vortex arc in its own right. Deformation of this arc is consistent with the orientation of the counter-rotating vorticity; it is likely the result of velocity induced both by counter-rotating vorticity on the present vertical arms (i.e., below the roll-up) and by counter-rotating vorticity produced by earlier segments of the shear layer, not shown in these figures but situated above the roll-up. (We will address coupling between the deformation of different shear layer segments later.) Figure 3-5(e) shows that the lee-side vortex arc has curved more completely towards the windward boundary of the jet while the rest of the shear layer segment's vorticity has become somewhat more convoluted. Note that the vortex arc carries vorticity pointing in the negative z direction, opposite in sign to the vorticity that has remained on the windward side of the shear layer segment.

Figure 3-6 is representative of the second group of vortex elements. This figure traces evolution of the shear layer segment immediately following that of Figure 3-5; its elements were introduced into the flow between $\tilde{t} = 1.16$ and $\tilde{t} = 1.22$. This shear layer segment is also pulled upwards on its lee side, as shown in Figure 3-6(b), but it rolls up most strongly on its *windward* side. This shear layer segment

in fact participates in the 6th roll-up on the windward side of the jet. Now it is the lee side that, though deformed out-of-plane, remains more sheet-like—see Figure 3-6(c). Vertical arms of counter-rotating vorticity still develop as the lee side is pulled upwards, however. The arms themselves show a tight grouping of vortex elements, clearly visible in Figure 3-6(d); these rolled-up vertical arms coincide with the vertical arms in of the first group, in Figure 3-5. Figure 3-6(d) shows that lee-side elements still get swept towards the windward boundary of the jet, just without undergoing roll-up. Lee-side elements appear slightly more disorganized in Figure 3-6(e); some of the elements seem to be winding around the primary vertical arms.

Shear layer segments in both groups thus exhibit strong similarities in their evolution. Each segment transforms into two arcs contributing opposite signs of vorticity to the jet's windward boundary and connected by vertical arms of counter-rotating vorticity. The two groups differ in whether it is the lee side or the windward side of the shear layer that rolls up more strongly. The two shear layer segments we selected are complementary because they coincide in space after their transformations are complete. In other words, the upper vortex arc of the first group is surrounded by disorganized elements of the second group; and the lower vortex arc of the second group is surrounded by disorganized windward-side elements of the first.

This analysis is consistent with recent experiments of Lim, New, and Luo [76] in which water-tunnel dye visualizations identify "vortex loops" carrying opposite signs of vorticity that result from folding of the cylindrical shear layer. (See Figure 1-3.) Our simulation results allow us to trace the origins of these loops and note how both lee and windward loops alternately evolve from identical vorticity-carrying material rings.

Assembling successive segments of the shear layer affords additional insight into the mechanisms underlying the observed ring-deformation. Figure 3-7 shows four successive segments of the shear layer at $\tilde{t} = 1.56$, before significant deformation has occurred. The segments are color-coded; the gray segment is the "first group" of vortex elements analyzed above, while the red segment is the "second group." The green and blue segments show vorticity introduced into the flow for $\tilde{t} \in [1.23, 1.26]$ and

$\tilde{t} \in [1.28, 1.35]$ respectively. Now consider the shape of these shear layer segments at $\tilde{t} = 2.00$. Figure 3-8(a) shows groups 1 and 2 together; it is clear that these deformed shear layer segments coincide in space, with group 1 contributing to the lee-side roll-up and group 2 contributing to the windward roll-up as described above. Figure 3-8(b) adds the next shear layer segment to the picture. This segment begins the transition to next group of roll-ups; like other segments, it folds into two vortex arcs, but its vertical extent is much smaller. Its lee-side vorticity is not attracted to the vortex arc at the top of the figure; rather, it initiates a *new* lee-side vortex arc in between the windward and lee arcs of segments 1 and 2. Figure 3-8(c) adds the fourth shear layer segment. This segment rolls up strongly on its lee side, strengthening the new vortex arc; its windward side, though not rolled up strongly, points out the site of a new windward roll-up, at the bottom of the figure. In keeping with this alternating pattern, the subsequent segment of the shear layer will reinforce the new windward roll-up. An alternate perspective view of all four shear layer segments at $\tilde{t} = 2.00$ is given in Figure 3-9. Figure 3-9(b) clearly shows the winding of vortex filaments around the arms of counter-rotating vorticity.

The periodic shear layer deformation mechanism just described is not unique to the $r = 7$ jet. In fact, we observe the same transformation of vorticity-carrying material rings into arcs connected by vertical arms of counter-rotating vorticity in the $r = 5$ and $r = 7$ jets, along with the same alternating pattern of lee and windward roll-ups. Figure 3-10 shows the folding of a shear layer segment in the $r = 5$ jet, with strong roll-up on the lee side. Figure 3-11 shows the complementary segment of the shear layer immediately following, which rolls up strongly on its windward side. Figure 3-12 shows an analogous shear layer segment in the $r = 10$ jet, featuring strong lee roll-up; transformation and windward roll-up of the complementary shear layer segment is traced in Figure 3-13.

3.2.3 Vorticity isosurfaces

We turn our attention from transformations of the jet shear layer to direct examination of the vorticity field. The vorticity $\boldsymbol{\omega}(\mathbf{x}, t)$ is computed on nodes of a regular

grid by summation over all the vortex elements, as specifed by (2.16). Grid spacing is $h = 0.05$; this 3-D mesh is then used to create vorticity isosurfaces and contours.

Figure 3-15 shows isosurfaces of vorticity magnitude $\|\boldsymbol{\omega}\|_2 = 40.0$ for the $r = 7$ jet at three successive times, $\tilde{t}$=1.8, 2.1, and 2.4. Analogous isosurfaces are computed for the two other jets. Figure 3-14 shows $\|\boldsymbol{\omega}\|_2 = 28.0$ isosurfaces for the $r = 5$ jet at $\tilde{t}$=2.4 and 3.0; Figure 3-16 shows $\|\boldsymbol{\omega}\|_2 = 60.0$ isosurfaces for the $r = 10$ jet at $\tilde{t}$=1.5 and 1.8.

Examination of these isosurfaces confirms the presence of several key vortical structures. Roll-up of the shear layer into vortex rings is clearly visible at all values of r. These rings immediately deform upwards on the lee side of the jet. Note that because isosurfaces only highlight regions of high vorticity, they cannot strictly be interpreted as material lines; that is, isosurfaces may not reflect the continuity of the material rings discussed in §3.2.2. Nonetheless, these figures all show arms of vorticity aligned with the jet trajectory on the lee side of the jet; these arms seem connected to vortex rings on the windward side, particularly among the first several roll-ups. This vorticity structure is consistent with the material deformations described in the previous section. The configuration of the vorticity arms—pulled upwards into the center of the vortex rings immediately above them—strengthens our hypothesis that vortical structures already in the flow induce the initial axial stretching of the jet shear layer on its lee side.

The relatively organized and periodic vortical structures in the intermediate field of the jet undergo a sudden breakdown into smaller length scales—4–5 diameters from the nozzle for $r = 10$, 3–4 diameters from the jet nozzle for $r = 7$, and slightly closer for $r = 5$. In the vortex filament plots, this breakdown was manifested by a complex tangle of computational elements; here we observe a dense field of small, nearly fragmented isosurfaces. This transition is accompanied by more pronounced bending into the crossflow. Some larger-scale structures—e.g., interacting vortex rings along the top edge of jet—are still visible, but the field is dominated by small length scales. Further downstream (e.g., for $x/d > 4.0$ at $r = 7$) there are artifacts of jet startup, such as a tongue of vortex-carrying fluid collapsing onto the centerplane. But

the present simulations have continued long enough to show that the breakdown into small scales is a persistent feature of the flow.

Returning to the intermediate field, we note a change in the spatial periodicity of rings on the windward side of the shear layer as the jet penetrates into the crossflow, upstream of the transition to small scales. Mechanisms discussed in §3.2.2 suggest that vortex arcs carrying opposite-sign azimuthal vorticity are driven towards the windward boundary of the jet. Confirmation of this mechanism is presented in Figures 3-17 and 3-18 which contour vorticity magnitude isosurfaces with values of spanwise vorticity ω_z at successive times, for $r = 5$ and $r = 7$. Negative spanwise vorticity, indicated by darker shading, originates on the lee side of the jet. As the jet penetrates, lee-side vortex arcs produced by roll-up and deformation of the shear layer find their way to the windward side. The resulting pattern is of vortex arcs carrying alternating signs of azimuthal vorticity, curved along the windward boundary of the jet.

While merging of opposite-sign vortex arcs has been proposed in [43] and was observed in our earlier, more dissipative, simulations [82], we do not explicitly observe merging here. Opposite-sign vortex arcs certainly approach each other more closely as the jet evolves; it is possible that the subsequent breakup into small scales may obscure any large-scale merging. It is also reasonable to expect that the mechanism and location of any merging may depend on Reynolds number; this dependence is currently under investigation [125].

Mechanisms proposed in §3.2.2 predict that vortex arms aligned with the jet trajectory should carry counter-rotating vorticity. In the near field of the jet, a significant component of this counter-rotating vorticity will be oriented vertically, and thus we show isosurfaces of vorticity magnitude contoured by ω_y in Figure 3-19. This image clearly reveals tubes of counter-rotating vorticity on the lee side of the jet, tilting into the cross-stream. Regions of high vorticity also seem to wind around the CVP arms; this phenomenon is particularly visible in Figure 3-15(a) and Figure 3-14(b).

A crucial feature of counter-rotating vorticity in the near field is that, though it results from periodic processes (e.g., roll-up and deformation of the shear layer), it

persists in the mean field [30, 102, 132]. We investigate this feature for the $r = 7$ jet by computing the time-average of the vorticity field over one cycle of shear layer roll-up. Shear layer roll-up in the near field is strongly periodic, with a period $\tilde{T}$ of approximately 0.18. This corresponds to a jet Strouhal number $St \equiv fd/(rU_\infty)$ of 0.8, a value confirmed by measurements of velocity spectra at selected points near the shear layer. (Further measurements of velocity spectra at different r and in different regions of the flow are underway [85].) We compute the averaged vorticity field using realizations incrementally separated by $\Delta t = 0.02$. Figure 3-20 shows the isosurface of vorticity magnitude $\|\bar{\boldsymbol{\omega}}\| = 40$ where $\bar{\boldsymbol{\omega}}$ is the mean vorticity in the interval $\tilde{t} \in [2.31, 2.49]$. The isosurface is contoured by the mean vertical vorticity $\bar{\omega}_y$. Two arms of counter-rotating vorticity are clearly present in the near field; the remaining structures—the periodic vortex arcs of alternating sign on the windward side of the jet—have disappeared in the mean field. Widening the interval over which averaging is performed to $[2.23, 2.49]$ reveals a continued CVP section further along the trajectory, in Figure 3-21(a). Since the length of this interval is not a multiple of near-field roll-up period, however, the near field CVP is more cleanly isolated in Figure 3-20.

Higher isosurface values can expose the CVP cores even more clearly. Figure 3-21(b) contours the isosurface of vorticity magnitude $\|\bar{\boldsymbol{\omega}}\| = 48$, showing that the CVP is really the dominant feature of the *mean* vorticity field upstream of the breakup into smaller scales. Also, it is important to emphasize that the counter-rotating structures are present at earlier times in the simulation as well. Figure 3-22 shows the isosurface $\|\bar{\boldsymbol{\omega}}\| = 52$, where time-averaging has been performed over the interval $\tilde{t} \in [1.71, 1.89]$. The higher cutoff shrinks the isosurfaces even further, but clearly isolates tubes of counter-rotating vorticity aligned with the jet trajectory.

In each of the four preceding figures, note that very little wall-normal vorticity is present in the first diameter of the trajectory. This observation is consistent with the mechanisms described in §3.2.2, in which axial deformation of the shear layer gradually re-orients initially azimuthal vorticity to the wall-normal direction.

3.2.4 Streamwise vorticity and transition to small scales

The dramatic breakdown of organized vortical structures into smaller scales as the transverse jet bends into the crossflow has been observed elsewhere [132], but mechanisms for this breakdown are not immediately clear. Classical instabilities of anti-parallel vortex tubes may be excited as counter-rotating vortex arcs approach each other on the windward boundary of the jet [98, 127]. More generally, the wild stretching and folding of vortex filaments to form small scales can be linked to short-wave instabilities, excited when the distance between filaments is comparable to or smaller than a core size [121, 28]. These mechanisms and their relevance to the transverse jet merit further investigation.

The complex structure of the far field, observed in vorticity isosurface plots (Figures 3-14–3-18) is also visible on instantaneous slices of streamwise vorticity. Our interest in streamwise vorticity is motivated by the traditional picture of the transverse jet far-field, in which a counter-rotating vortex pair with compact cores slowly travels away from the wall and spreads as it persists downstream [18, 67]. This picture, based on integral arguments or ensemble-averaged measurements of scalar concentration, vorticity, or velocity, is admittedly over-simplified. The instantaneous structure of the jet cross section is far more complex—asymmetric, meandering in the spanwise and wall-normal directions, and dominated by small scales [112, 41, 102, 88]. Figures 3-23 and 3-24 bear this out, showing slices of streamwise vorticity on a series of x/d-planes, for $r = 5$ and $r = 7$. The field varies significantly from plane to plane, and significant co-mingling of positive and negative vorticity is evident.

Though the vorticity field appears quite unorganized, it may yet have an underlying structure; we would like to extract a signature of this structure if it is present. Following the approach suggested by Yuan *et al.* [132], we low-pass filter the instantaneous streamwise vorticity on each plane in Figures 3-23 and 3-24. We construct a two-dimensional low pass filter with corner wavenumber $k_y = k_z = \pi/d$, where d is the jet diameter. Results of the filtering are shown in Figures 3-25 and 3-26. Organized counter-rotation is evident in the filtered vorticity field, with strong regions of

positive ω_x for $z > 0$ and vice versa, as expected. The maximum streamwise vorticity magnitude in the filtered field is approximately 8 times lower than in the unfiltered field; again, this result is consistent with [132].

The instantaneous vorticity in Figures 3-23 and 3-24 is slightly asymmetric about the centerplane ($z = 0$). Asymmetry persists under filtering (see Figures 3-25 and 3-26) and is more pronounced in the $r = 7$ case than in the $r = 5$ case. Interpretation of this asymmetry and its origins is rather subtle. Though our vorticity-flux boundary conditions are symmetric and the crossflow is uniform, symmetry is not explicitly enforced elsewhere in the computation. In fact the clustering partition, with its random seed of initial centroids (see Chapter 4) does not respect symmetry at all. As a result, the velocity approximation error is distributed asymmetrically in space. These errors, in turn, may cause initially symmetric vortex particle locations and weights to evolve asymmetrically.

Numerical mechanisms, however, cannot be separated from the underlying flow physics. In a simulation, various forms of numerical error (e.g., approximation error and roundoff error from finite-precision arithmetic) are always present; similarly, no experimental setup can be free of physical noise, surface roughness, or asymmetry at length scales smaller than measurement or machining accuracy. Questions of symmetry are intimately linked to how the flow amplifies or dampens these perturbations. In general, issues of whether the transverse jet is ultimately "symmetric" or "asymmetric" remain unresolved. Smith and Mungal [112] performed a series of wind tunnel experiments in which symmetry of the experimental setup and flow uniformity were carefully controlled, yet they report instantaneous and even *ensemble-averaged* PLIF images of the jet cross-section that are not symmetric. Spatially filtered streamwise vorticity contours reported in [132] are also asymmetric, though in this computational study, the incoming pipe velocity profiles were extracted from a temporally-evolving "turbulent flow" simulation which may have had instantaneous asymmetry. In general, several researchers have observed asymmetry in averaged profiles of the transverse jet, increasing with r and with downstream distance [66, 112].

Analysis of the filtered vorticity field raises another interesting issue—the impact

of Reynolds number. Our earliest vortex simulations of this flow employed core expansion and consequently were quite dissipative in the far field; despite the coarseness of those calculations, counter-rotating streamwise vorticity was immediately evident in instantaneous transverse planes [81, 82]. Some of our current work (beyond the scope of this thesis) focuses on accurate finite-Reynolds number simulations of the transverse jet using vorticity redistribution [125]; here the breakdown of organized structures into small scales remains very much in evidence, but it may be that the filtered vorticity field carries a proportionally larger portion of the energy. An LES study by Yuan *et al.* [132] at a crossflow Reynolds number ($Re = U_\infty d/\nu$) of 1050 shows results similar to the present data. Water-tunnel experiments for extremely low Reynolds number (Re=21–78), however, show the jet fluid can completely bifurcate into tubes aligned with counter-rotating vorticity [63].

3.3 Boundary conditions and jet trajectories

Figures 3-27 and 3-28 show instantaneous velocity vectors and streamlines on the centerplane $z = 0$ at $t = 3.20$, for simulations with $r = 7$. The simulation in Figure 3-27 introduces vortex elements containing only jet azimuthal vorticity, i.e., with strengths given by equation (2.23). The simulation in Figure 3-28 introduces vortex elements that additionally account for the interaction of channel wall vorticity with the jet, i.e., with strengths given by equation (2.27). Contours indicate the total velocity magnitude $\|\mathbf{u}\|_2$. In both simulations, the time interval for introducing new filaments at the nozzle was fixed at $\Delta t_{noz} = 0.02$.

The comparison in these two figures clearly illustrates the effect of nozzle-edge vorticity on the near field trajectory of the jet. Neglecting vorticity in the jet channel wall boundary layer results in a jet initially angled downstream from the vertical, inconsistent with experimental observation. Modeling the interaction of channel wall vorticity with the jet, however, results in a jet trajectory initially normal to the wall, matching experimental observations and correlations [58].

An interesting feature of the centerplane velocity field in Figure 3-28 is the presence

of a node just downstream of the jet nozzle. In three dimensions this correponds to fluid being swept forward around the jet, toward the centerplane. Hasselbrink and Mungal [59] confirm the presence of this node in PIV measurements. By continuity, the lee side of the jet shear layer is initially subject to a compressive strain rate ($\partial w/\partial z < 0$) as it is stretched upwards.

Three-dimensional streamlines, shown at $\tilde{t} = 2.00$ in Figures 3-29 and 3-29, provide a more complete context for features on the centerplane. Crossflow fluid near the wall is swept around the nascent jet fluid, consistent with our boundary conditions, and into the centerplane downstream of the jet. Water-tunnel dye visualizations by Kelso, Lim and Perry [70] revealed very similar flow patterns, corroborated in other studies [108]. On the lee side of the jet, some of the crossflow fluid is pulled strongly upwards into the region of counter-rotating vorticity, while other streamlines continue in the streamwise direction; there is likely a separatrix in the streamline pattern delineating the two behaviors.

Additional validation may be obtained may be comparing numerical results with correlations and scaling laws for the jet trajectory. Hasselbrink and Mungal [58] perform an extensive scaling analysis of the transverse jet and derive an analytical expression for the near-field trajectory, where the trajectory is defined as the mean streamline emanating from the center of the jet:

$$\frac{y_c}{rd} = \left(\frac{2}{c_{ej}} \frac{x_c}{rd} \right)^{1/2} \tag{3.1}$$

Here c_{ej} denotes a near-field entrainment coefficient; we use the value $c_{ej} = 0.32$ as suggested by [101]. This analytical trajectory is shown in Figure 3-31, along with the instantaneous center streamlines obtained from simulations at $r = 5$ and $r = 7$. We plot in rd-coordinates suggested by similarity analysis and often used in experimental correlations [80, 69, 117]. Initial agreement between the simulation and the scaling-law model is good, although the instantaneous streamlines exhibit wiggles around periodic vortical structures, as expected. Moderate deviations downstream may be due to a variety of factors. For one, the near-field scaling law in (3.1) transitions

to a different 1/3 power-law trajectory for the jet far-field, and it is not clear where this transition should occur, and how this location should depend on r. Also, while the 1/2 exponent in (3.1) results from a well-founded series of similarity assumptions and other approximations, experimentalists have reported a range of different values, typically from 0.28 to 0.34. Finally, it is important to note that the downstream section of each numerical trajectory represents a jet envelope that is still evolving downstream in time, and that vortical artifacts of jet startup have yet to convect far away enough to have negligible effect.

The ring-folding mechanism discussed in §3.2.2 motivates an interesting connection to the trajectories and scaling analysis. Jet centerline trajectories in Figure 3-31 match Hasselbrink and Mungal's "near field" 1/2 power-law trajectory quite well until a certain critical value of y/d (or y/rd); then, the trajectories continue with a shallower penetration into the flow. (This is particularly visible for the $r = 7$ results, at $y/rd = 0.65$.) The same similarity analysis yields a 1/3 power-law for the far field, which does indeed correspond to shallower trajectories:

$$\frac{y_c}{rd} = \left(\frac{3}{c_{ew}} \frac{x_c}{rd} \right)^{1/3} \tag{3.2}$$

Here c_{ew} is a far field entrainment coefficient; a value of $(3/c_{ew})^{1/3} = 2.1$ is suggested in [59].

The folding of vortex rings suggests a mechanism governing the transition from near to far fields. Before rings have folded completely, the jet is more upright, dominated by periodic structures derived from deformations of the cylindrical shear layer; after the rings fold, we observe a cascade to small scales and an underlying counter-rotating vorticity. Folding comprises the key topological change in the evolution of vorticity field, replacing one set of vorticity dynamics with another. It is possible that this demarcation of the vorticity dynamics bears some correspondence to the near- and far-field jet behavior obtained by intermediate asymptotic similarity, and that the folding of vortex rings to form the counter-rotating vortex pair provides a mechanistic explanation of the transition. While the folded states depicted in Fig-

ures 3-5–3-6 are obtained at different times than the $r = 7$ trajectory in Figure 3-31, the y/rd coordinate at which lee-side vortex arcs reach the windward side of the jet roughly corresponds to the point at which the jet centerplane trajectory departs from the near-field power law. This possiblity bears further investigation.

A more complete analysis of jet trajectories requires longer-time simulations to achieve a stationary state for $y/rd \gg 1$ and to compute the resulting mean velocity field. Continuing the present vortex simulations to longer time is more computationally feasible at finite Reynolds number; finite-Re simulations employing vorticity redistribution and remeshing are currently underway [125].

3.4 Near-field vortex lines

A confirmation of the analytical model for closed vortex filaments in the near field of the transverse jet is presented in Figure 3-32. Here, the solid curves are vortex lines of the numerical vorticity field—i.e., lines obtained by numerical integration of the vorticity of an $r = 7$ jet at $\tilde{t} = 1.40$, i.e., a time by which the vorticity field several diameters above the jet nozzle has matured. Dashed lines are obtained from integration of the ODE system in (2.30). Agreement is quite good. Slight discrepancies may be due in part to the finite spatial resolution of the numerical vorticity field, here obtained for $\delta = 0.05$, $h/\delta < 1$, compared to the continuous field used to derive the analytical filaments.

3.5 Direct Lyapunov exponent calculations

An alternative approach to characterizing the flow seeks Lagrangian coherent structures. Many approaches to coherent structures—indeed myriad *definitions* of coherent structure—have been proposed [55, 57, 126, 19, 64]; we do not attempt to review these here. Instead, we note that the utility of many of these criteria is restricted in the present context because the turbulent transverse jet is an aperiodic flow for which we necessarily have only a finite-time interval of data.

Here we take the approach suggested in [56] and view coherent structures as linearly unstable material lines or surfaces. A straightforward means of extracting stable/unstable material structures is to directly calculate their effect on particle paths. Consider the deformation tensor $\mathbf{F}$:

$$\mathbf{F}^{\mathrm{T}} = \nabla_{\mathbf{x}_0} \mathbf{x}\left(t, t_0, \mathbf{x}_0\right) \tag{3.3}$$

Here $\mathbf{x}(t, t_0, \mathbf{x}_0)$ is the flow map, i.e., the trajectory followed by the particle that is at point $\mathbf{x}_0$ at time t_0. $\mathbf{F}$ thus describes deformation at $(\mathbf{x}, t)$ with respect to initial conditions. For $t > t_0$, the largest singular value of $\mathbf{F}$, $\sigma_1(\mathbf{F})$, gives the maximum *relative* growth of an infinitesimal vector at $\mathbf{x}_0$ advanced by the flow map over the interval $[t, t_0]$. Equivalently, this is the maximum length $\mathcal{E}_{t_0}^{t}(\mathbf{x}_0)$ of a unit vector advected by the linearized flow map:

$$\mathcal{E}_{t_0}^{t}(\mathbf{x}_0) = \sigma_1\left(\mathbf{F}\right) = \sqrt{\lambda_{\max}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F}\right)} \tag{3.4}$$

where $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ is known as the right Cauchy-Green strain tensor [91]. The largest finite-time Lyapunov exponent associated with $\mathbf{x}(t, t_0, \mathbf{x}_0)$ is thus [56]

$$\Lambda(t, t_0, \mathbf{x}_0) = \frac{1}{2(t - t_0)} \ln\left(\lambda_{\max}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F}\right)\right) \tag{3.5}$$

We calculate Λ in forward time ($t > t_0$) using a dense initial grid of particles. These particles are advected by the same second-order predictor-corrector scheme used to advect the nodes of vortex filaments. The velocity at each particle is calculated directly from the Biot-Savart law (2.8). Derivatives in $\mathbf{F}$ are approximated with central differences; the maximal eigenvalue of each symmetric matrix $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ is calculated using a rational variant of the QR method. Local maxima of the resulting field $\Lambda_{t_0}^{t}(\mathbf{x}_0)$ are *repelling* coherent structures.

Direct calculation of Λ as described above is susceptible to numerical error. Particle paths diverging from each other exponentially fast will yield exponentially-growing errors in discrete approximations to components of $\mathbf{F}$. However, this issue can be sur-

mounted for finite time by choosing a sufficiently dense initial grid [56]. We focus our calculations on the near field of the $r = 7$ jet, releasing particles at $\tilde{t}_0 = 2.00$, a time when vorticity dynamics in the near field of the jet have matured. We run cases corresponding to two initial grids: a lower-resolution case with a uniform grid spacing of 0.040 and a higher-resolution case with uniform grid spacing of 0.025; both of these values are normalized by the jet diameter d.

Figures 3-33 and 3-34 show contours of maximal direct Lyapunov exponent (DLE) $\Lambda_{\tilde{t}_0}^{\tilde{t}}(\mathbf{x}_0)$ on planes of constant z/d. We restrict our attention to $z/d < 0$ since the flow is essentially symmetric in this region. The lower-resolution case, Figure 3-33, continues the calculation to $\tilde{t} = 2.30$. The higher-resolution case, Figure 3-34, continues to $\tilde{t} = 2.35$ but restricts the grid of initial conditions to $y/d > 1.5$ in order to save computational time. In both cases, continuing the calculations significantly further in time led to degradations in the sharpness of the structures, perhaps a result of numerical error.

An interesting series of repelling structures is revealed as the planes slice through the near field of the transverse jet. For $y/d < 3.0$, roll-up of the shear layer is a central feature of the vorticity dynamics. On both the lee and windward sides of the jet, we observe a telescoping series of repelling lines. Lines on the windward side are quite vertical near the nozzle and increasingly S-shaped further into the flow. Lines on the lee side flatten somewhat as they recur along the jet trajectory. The periodic structure of these repelling lines suggests that they delineate regions of fluid that participate in different roll-ups along the shear layer. Figure 3-35 lends credence to this hypothesis by plotting vortex filaments at $\tilde{t} = 2.00$ over a spanwise slice of DLE contours. It is important to keep in mind that the DLE field reflects dynamics over the entire interval $[2.00, 2.30]$, while the vortex filaments provide only an instantaneous snapshot of the flow structure at $\tilde{t} = 2.00$. Thus we should not expect features to correspond exactly between the two. In regions where roll-up has already occurred at $\tilde{t}_0$ (e.g., for $y/d > 1.2$), however, repelling surfaces seem to the bound cores of high vorticity. These surfaces reflect the *future* action of vortex arcs on the surrounding fluid. Focusing on the lee side for $2 < y/d < 3$, we also note that flattening and inward

migration of the lee-side repelling surfaces matches movement of lee-side vortex arcs toward the windward boundary.

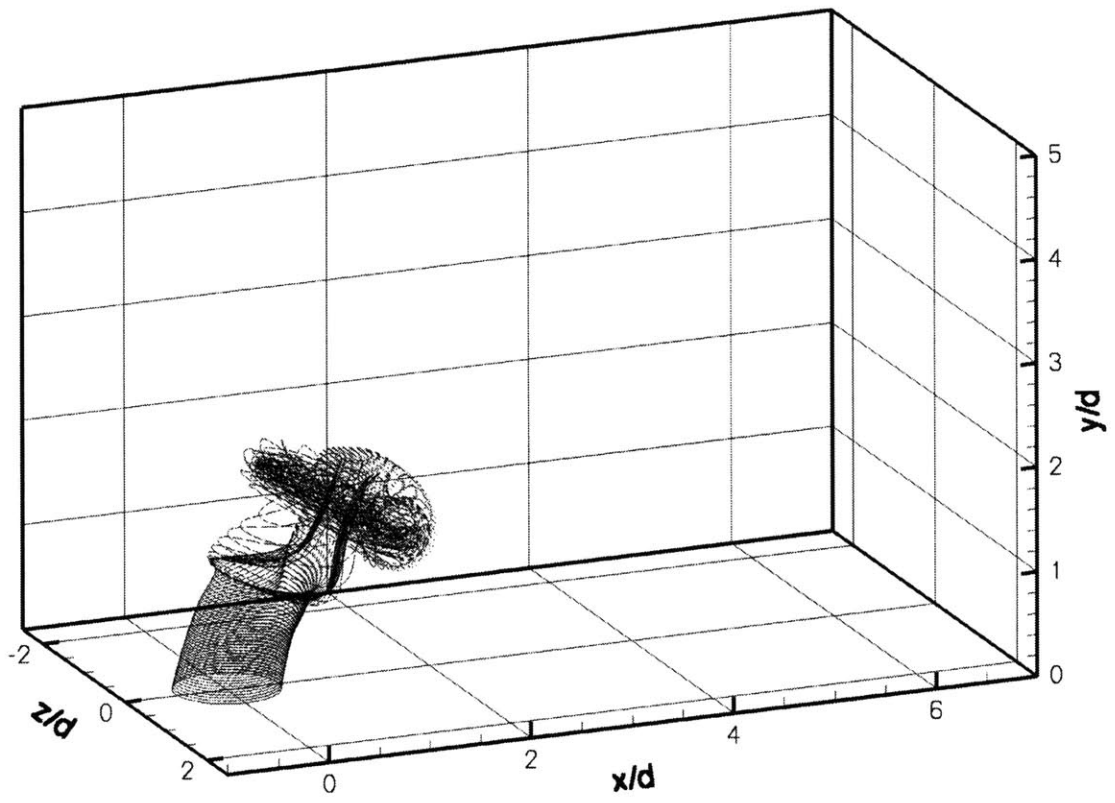
Returning to Figures 3-33 and 3-34, we note that spanwise slices for which $z/d < -0.5$ are beyond the jet column, and thus they intersect a different set of repelling surfaces than those discussed above. The structure in these planes is more difficult to interpret, but does reflect the widening of the jet that follows formation of counter-rotating vorticity.

Axial slices of Λ reveal another crucial dynamical feature of the near field. Figure 3-36 shows DLE contours on planes of constant y/d. (Zero values at the centerplane for subfigures (b)-(e) are an artifact of the plotting routine and should be ignored.) Part (a) shows a circular repelling surface surrounding the jet column near the nozzle; subsequent slices, moving away from the wall, show this circle transform into the traditional kidney-shaped cross section associated with counter-rotating vorticity. This transformation seems to continue in layers; the repelling line furthest upstream declines in strength until it is replaced by the next line, and so on. Layers mirror the periodicity of the roll-up and deformation process that creates successive vertical sections of counter-rotating vorticity, described in §3.2.2. As the axial slices move away from the wall, the repelling surfaces move further downstream and spread further in the spanwise direction.

Figures 3-37–3-39 show isosurfaces of Λ in three dimensions, with and without overlaid vortex filaments. A relatively large value of Λ was chosen so that isosurfaces would delineate local maxima on either side, but no attempt was made to rigorously extract local maxima in three dimensions. This limitation, plus limitations of numerical resolution, give some of the surfaces a rough appearance. Nonetheless the structure amplifies that of the planar contour plots. In particular, the repelling lines identified on spanwise planes wrap around the jet to form layered shells of repelling surfaces. The shape of repelling surfaces on the sides of the transverse jet suggests a helical winding of fluid through counter-rotating tubes.

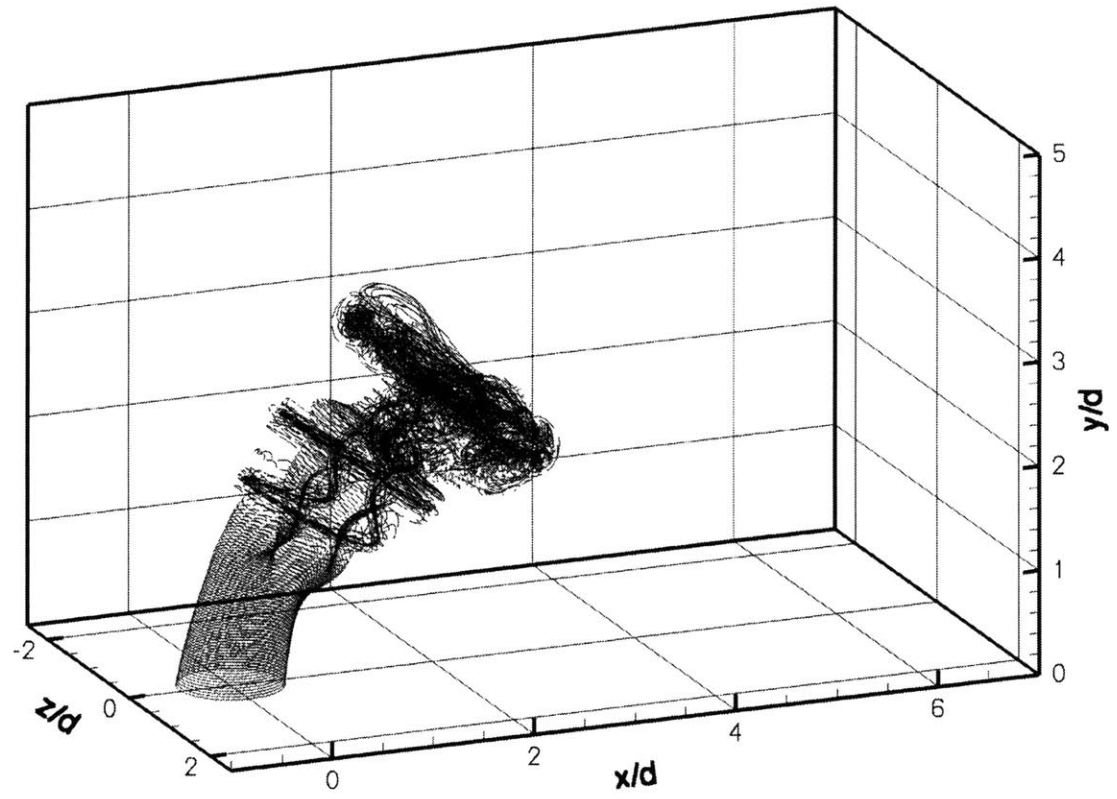
It would be interesting to continue these DLE calculations further in time and apply them to the more complex far-field regions of the flow, particularly after the

breakdown into smaller scales. Doing so may require a significantly denser grid of initial particles, however, and thus carry great computational expense. Alternatively, we may have more success with techniques that employ velocity gradient information along particle trajectories to compute hyperbolicity time [56]. We also note that *attracting* coherent structures, computed for $\tilde{t} < \tilde{t}_0$, would complete the present picture of transport in the near field of the jet; these structures tend to correspond with readily observable features of the flow, and may thus be easier to interpret.



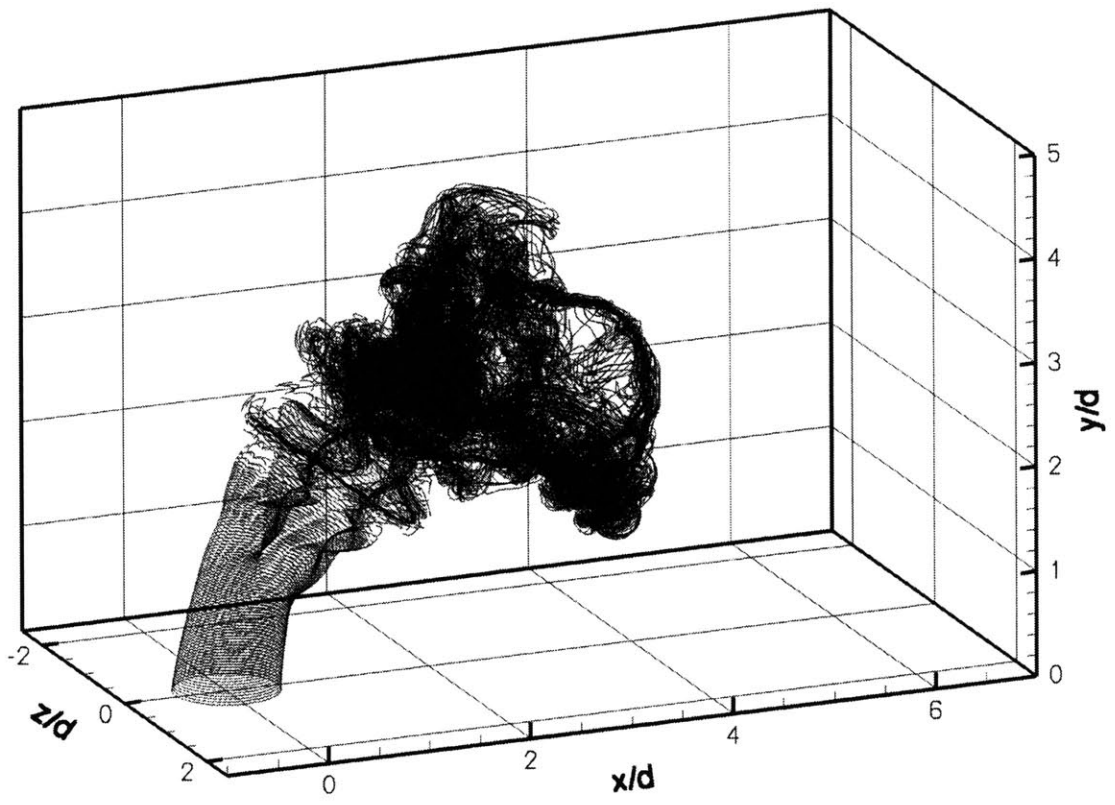
(a) $\tilde{t} = 1.20$

Figure 3-1: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 5$.



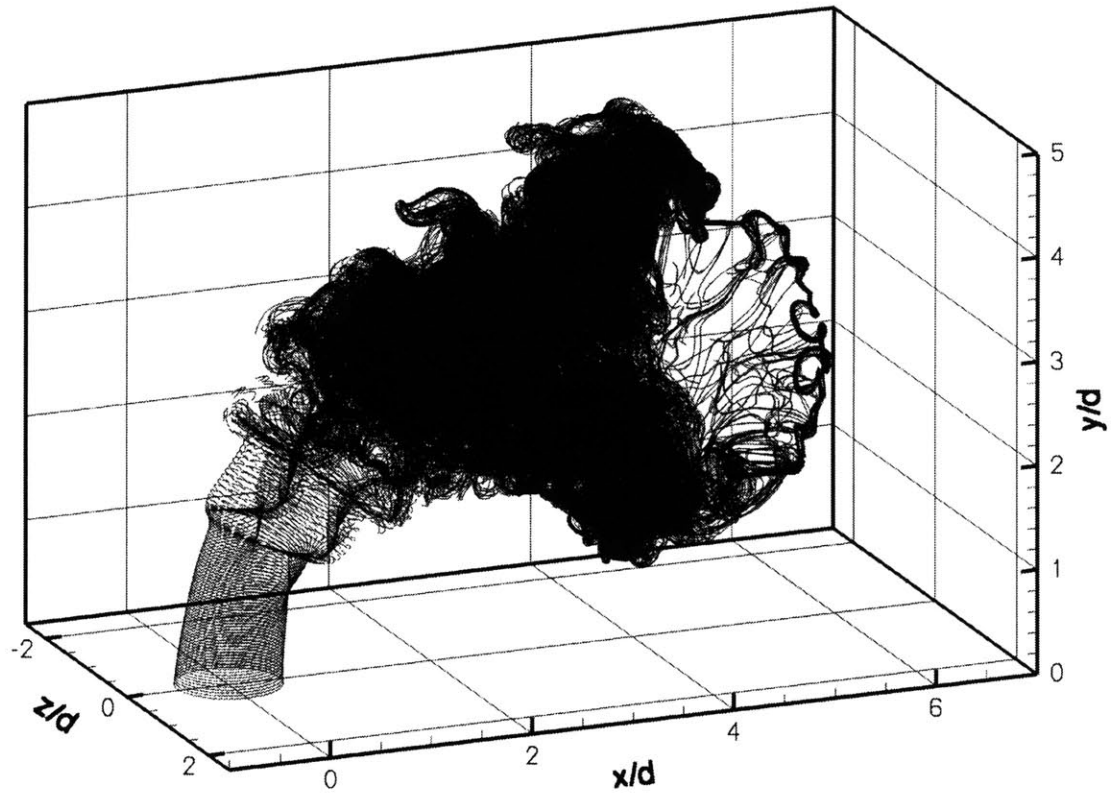
(b) $\tilde{t} = 1.80$

Figure 3-1: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 5$ (con't).



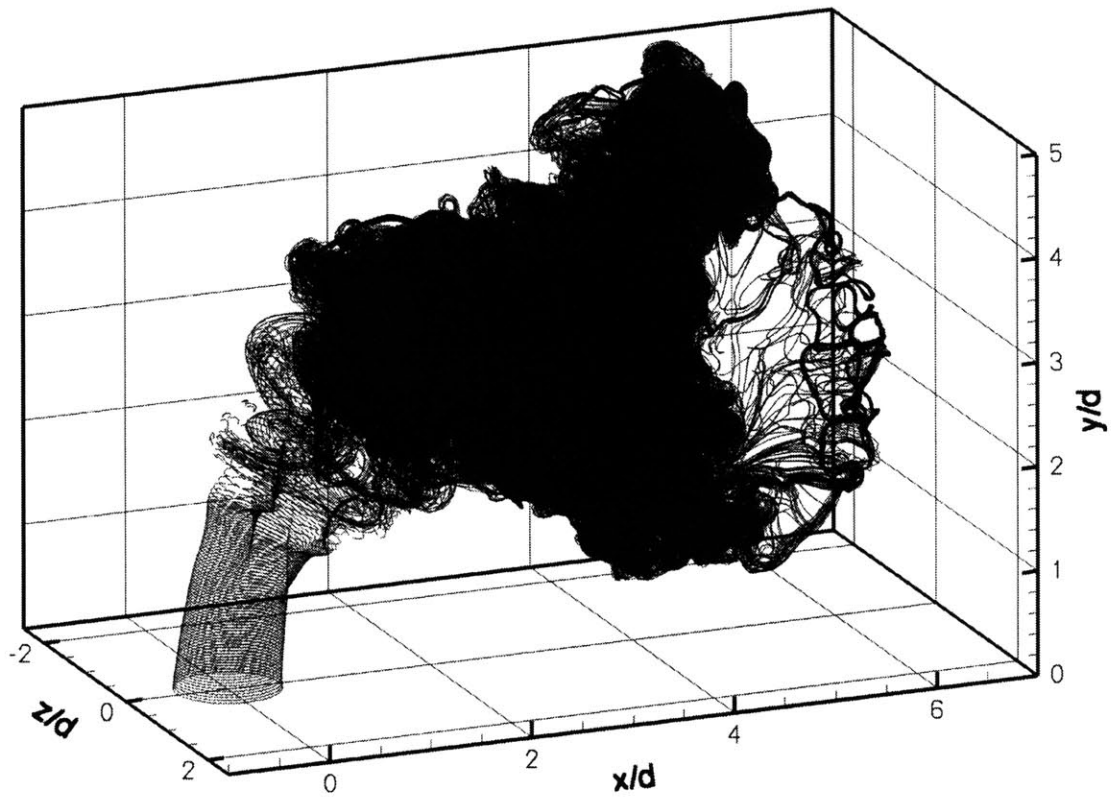
(c) $\tilde{t} = 2.40$

Figure 3-1: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 5$ (con't).



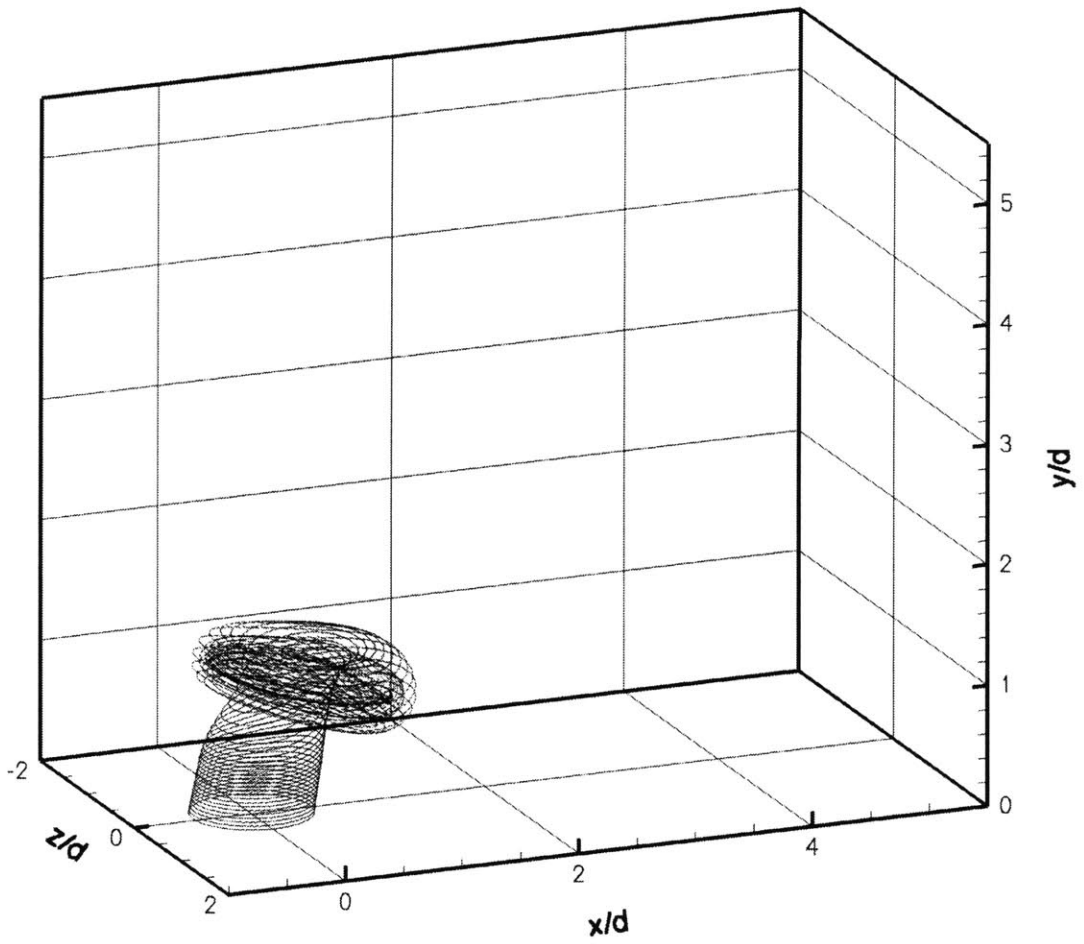
(d) $\tilde{t} = 3.00$

Figure 3-1: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 5$ (con't).



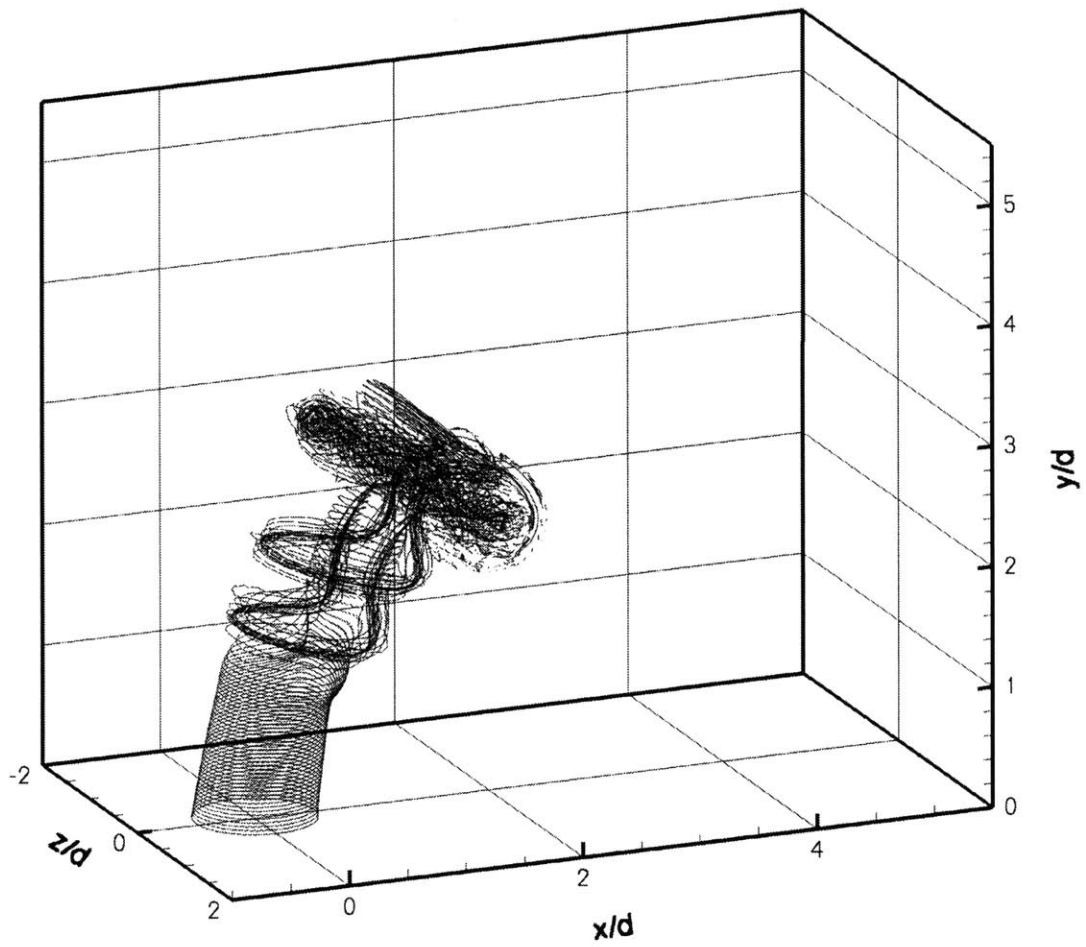
(e) $\tilde{t} = 3.30$

Figure 3-1: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 5$ (con't).



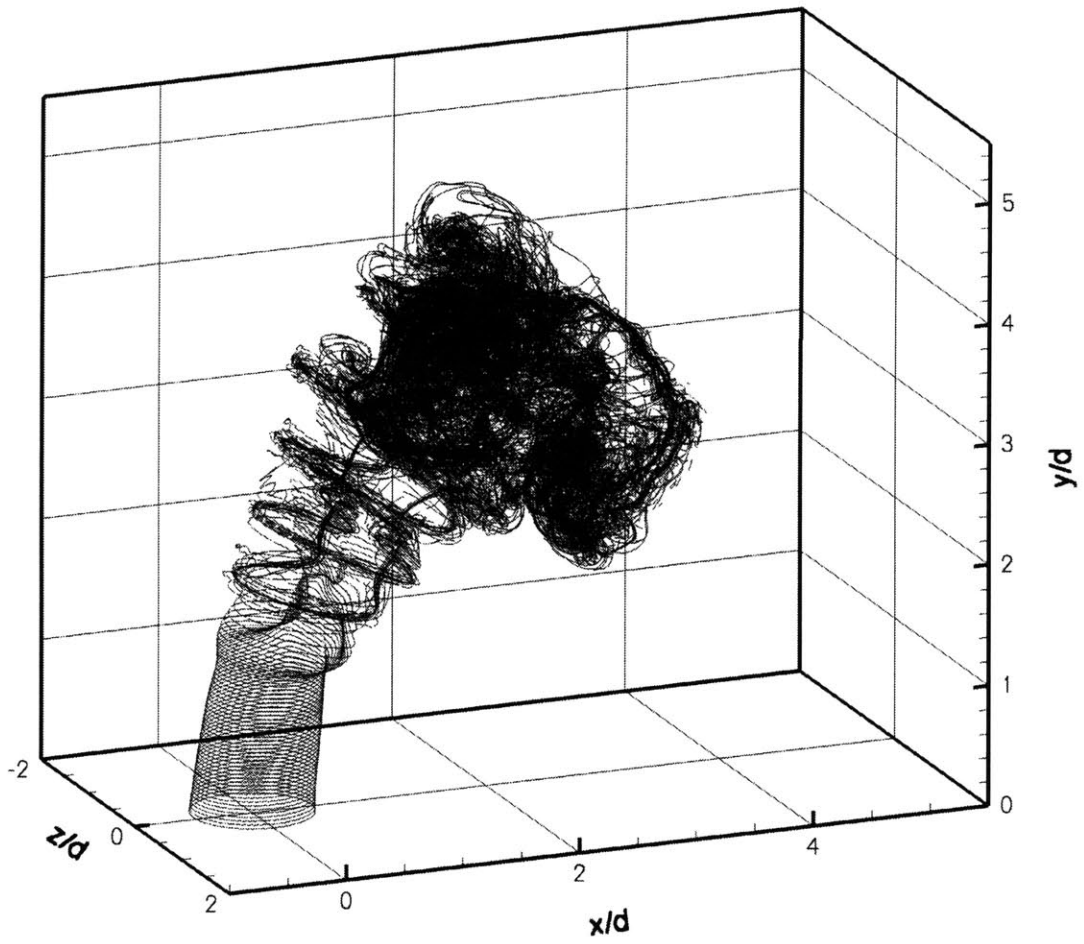
(a) $\tilde{t} = 0.60$

Figure 3-2: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 7$.



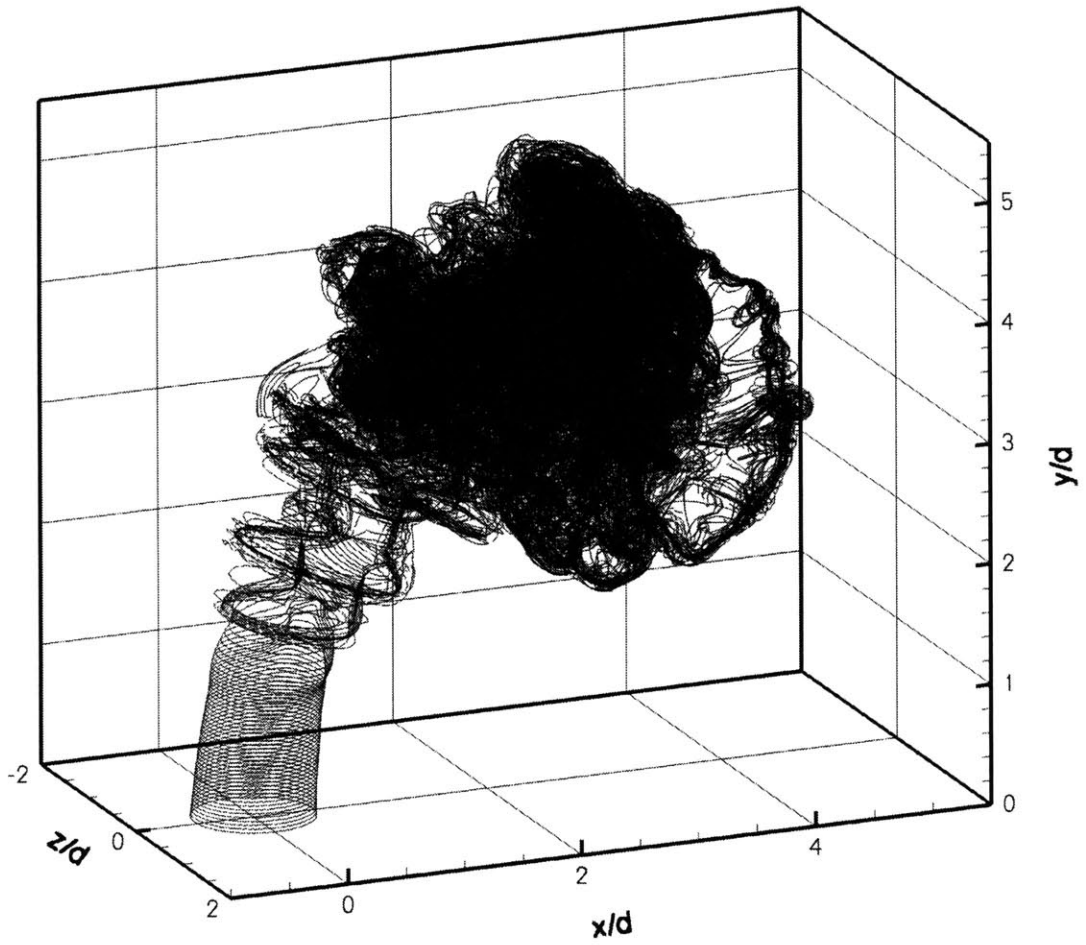
(b) $\tilde{t} = 1.20$

Figure 3-2: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 7$ (con't).



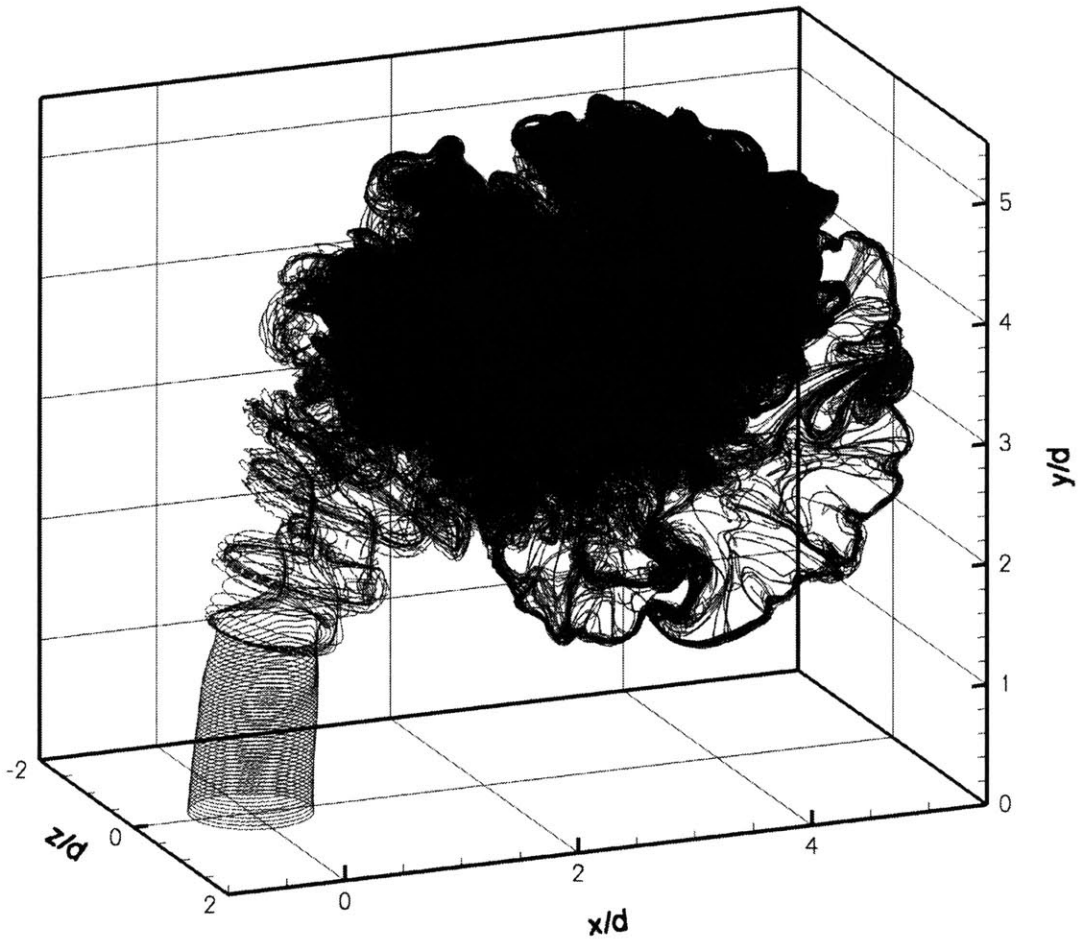
(c) $\tilde{t} = 1.80$

Figure 3-2: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 7$ (con't).



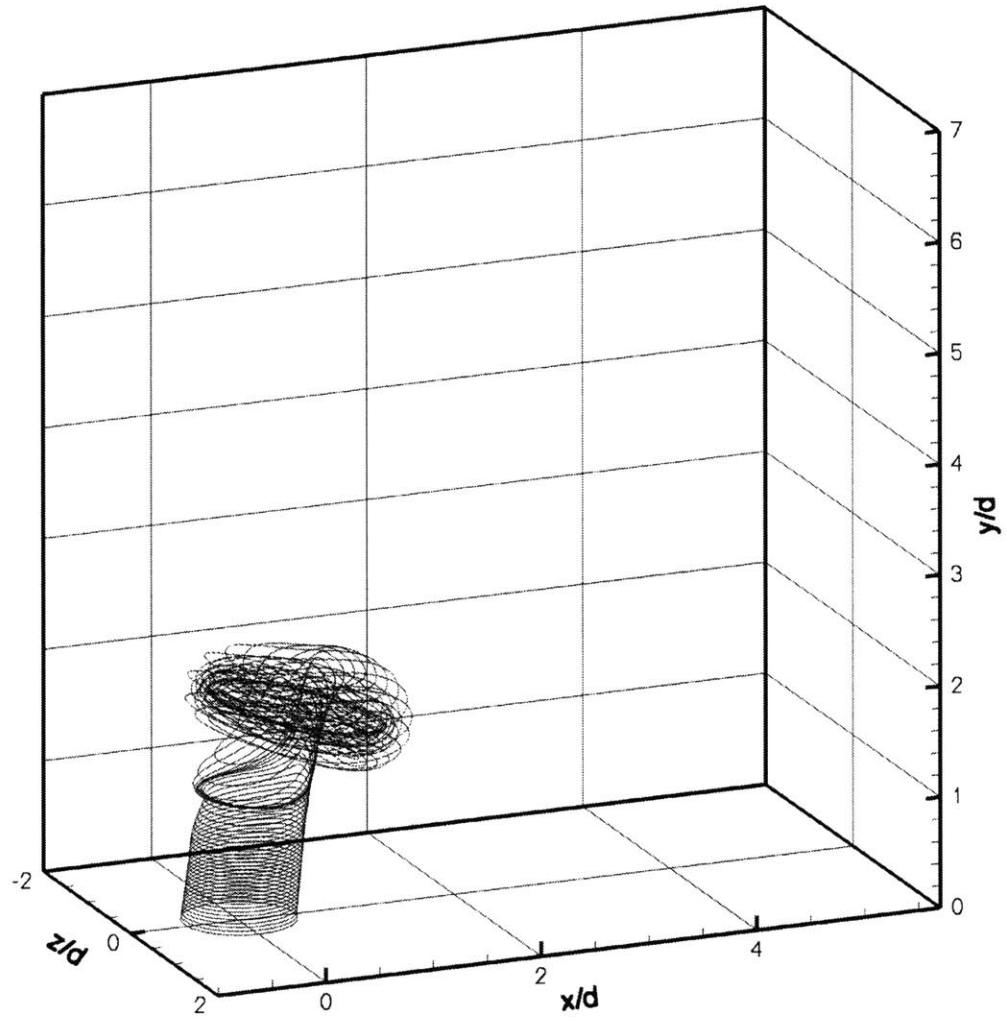
(d) $\tilde{t} = 2.10$

Figure 3-2: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 7$ (con't).



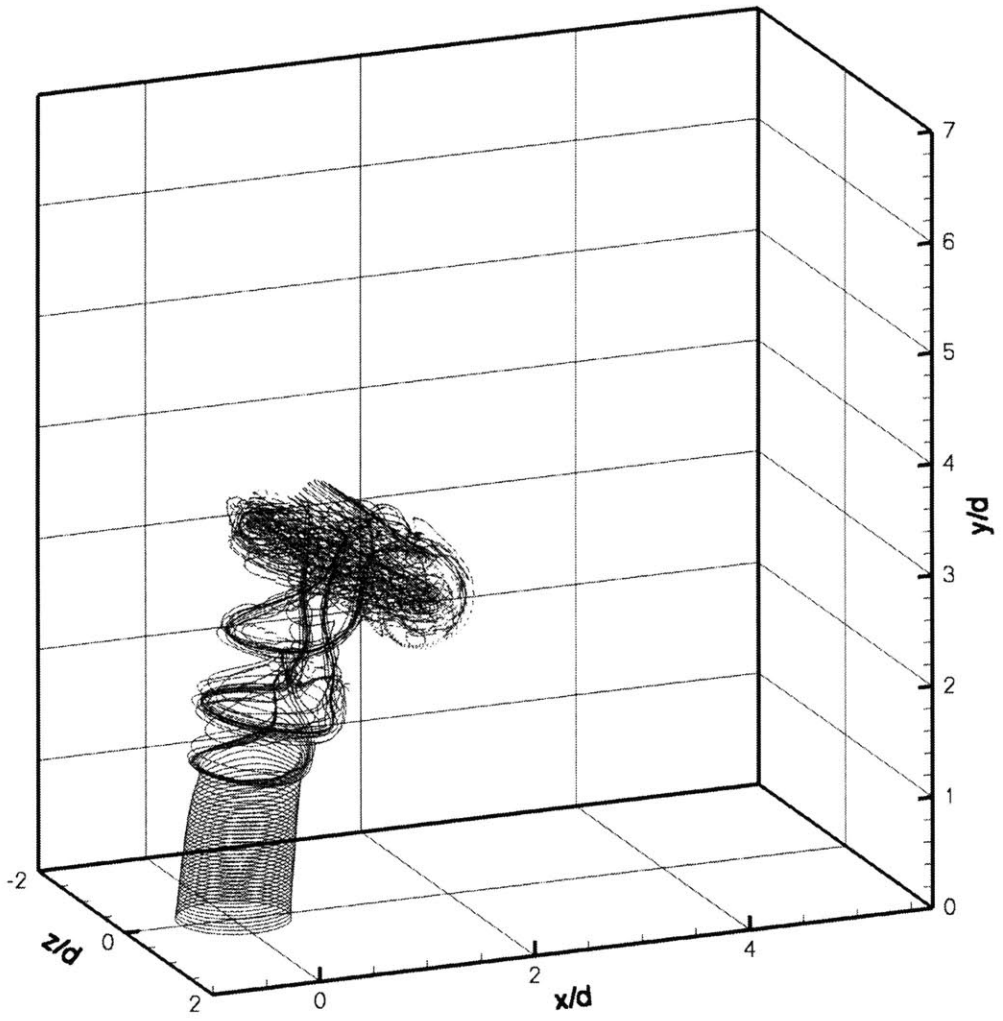
(e) $\tilde{t} = 2.40$

Figure 3-2: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 7$ (con't).



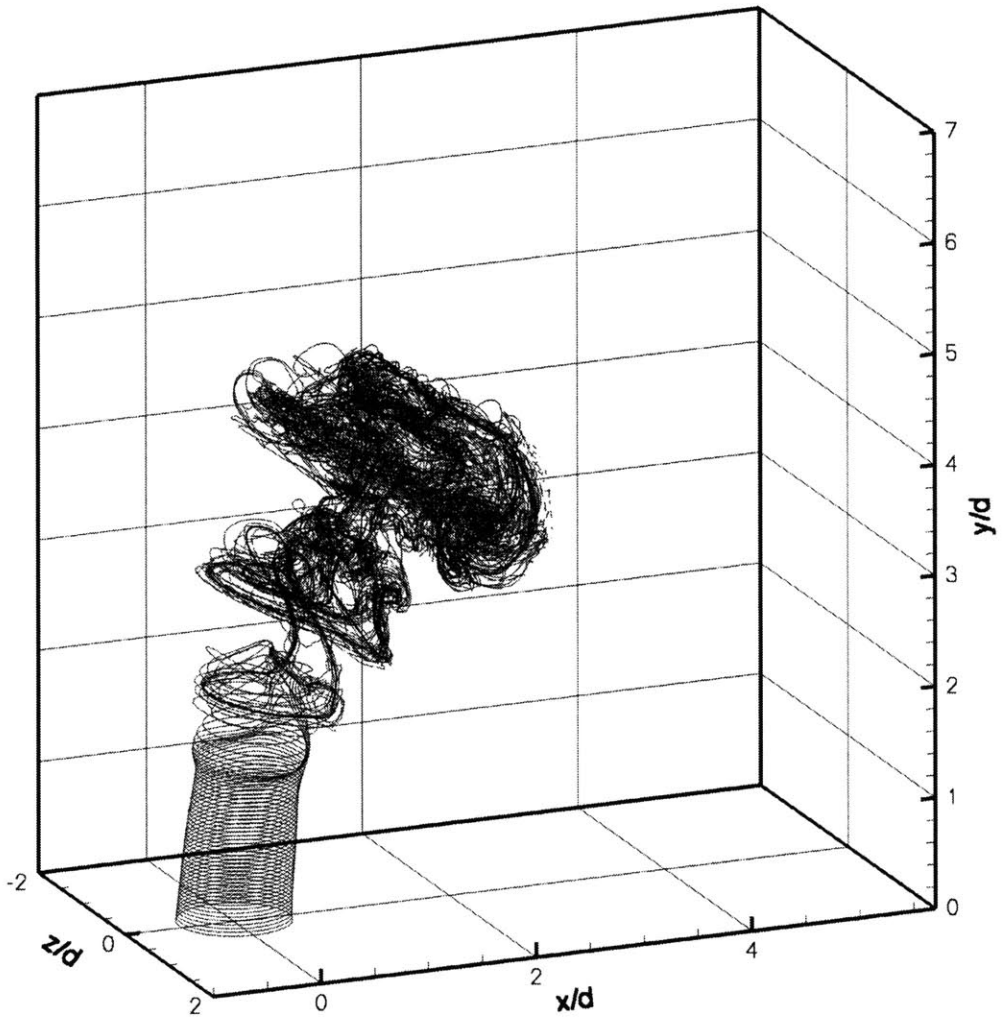
(a) $\tilde{t} = 0.60$

Figure 3-3: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 10$.



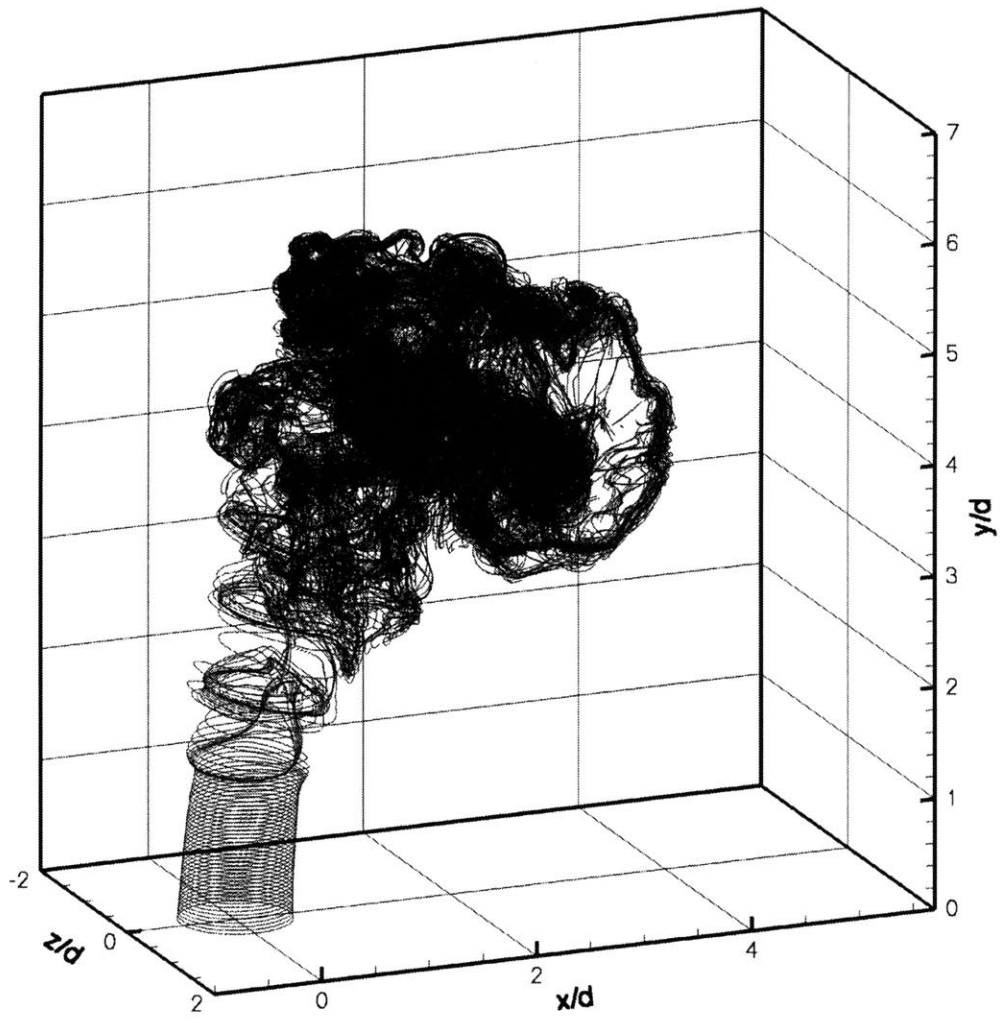
(b) $\tilde{t} = 0.90$

Figure 3-3: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 10$ (con't).



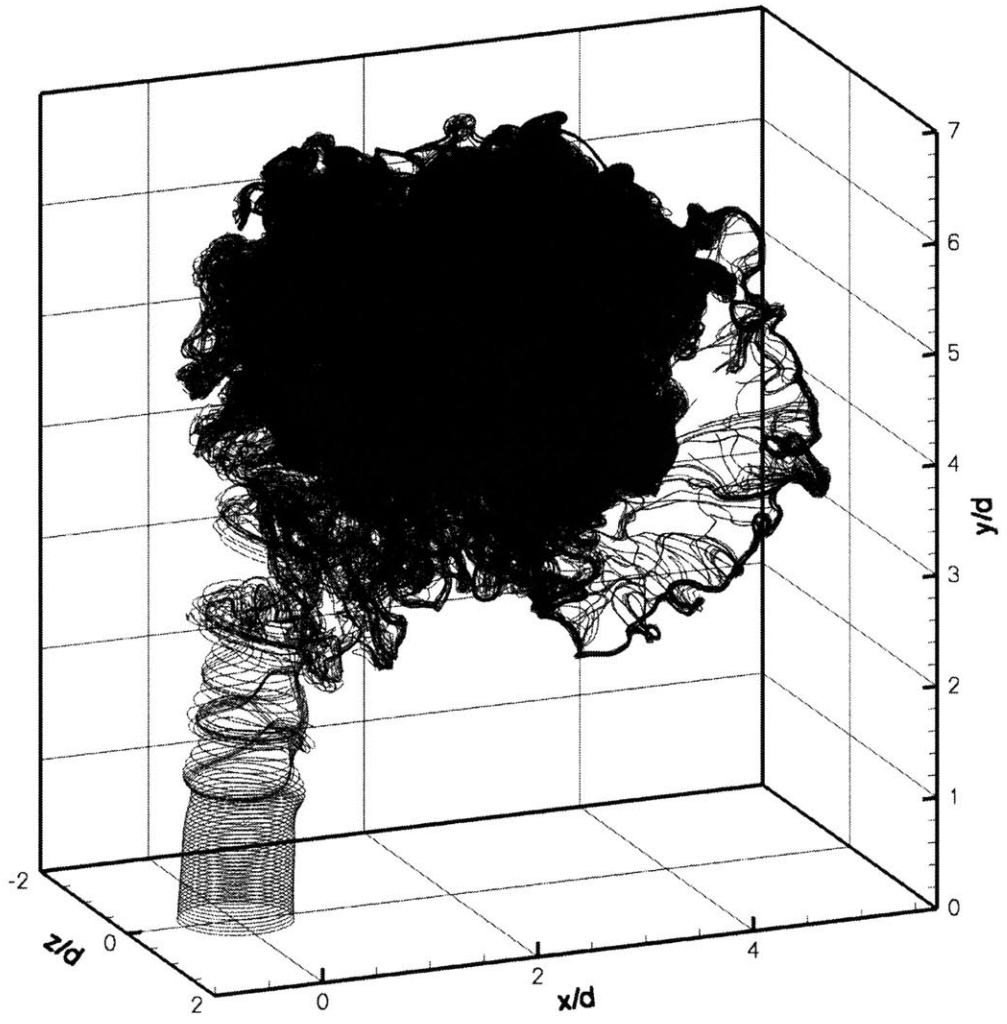
(c) $\tilde{t} = 1.20$

Figure 3-3: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 10$ (con't).



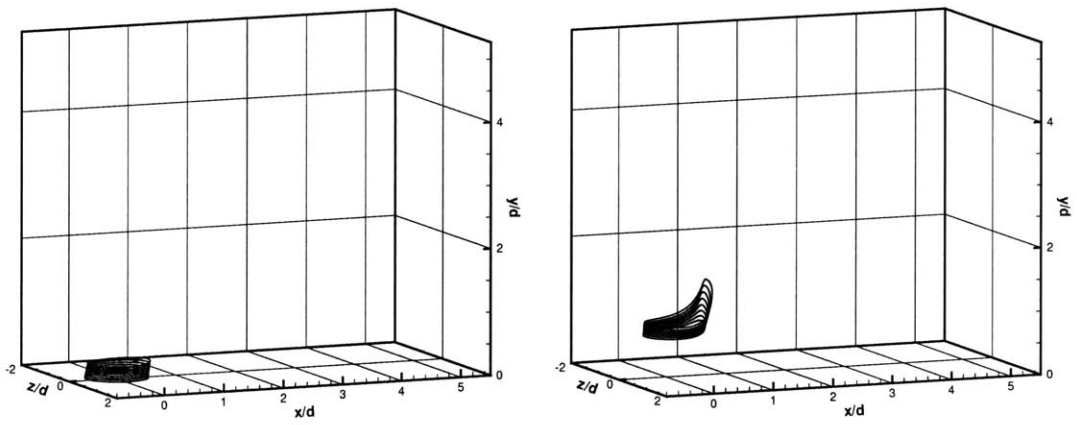
(d) $\tilde{t} = 1.50$

Figure 3-3: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 10$ (con't).

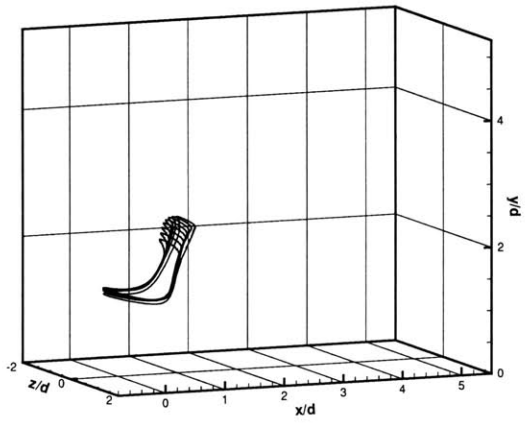


(e) $\tilde{t} = 1.80$

Figure 3-3: Perspective view of vortex filaments in the evolving transverse jet at five successive times, for $r = 10$ (con't).

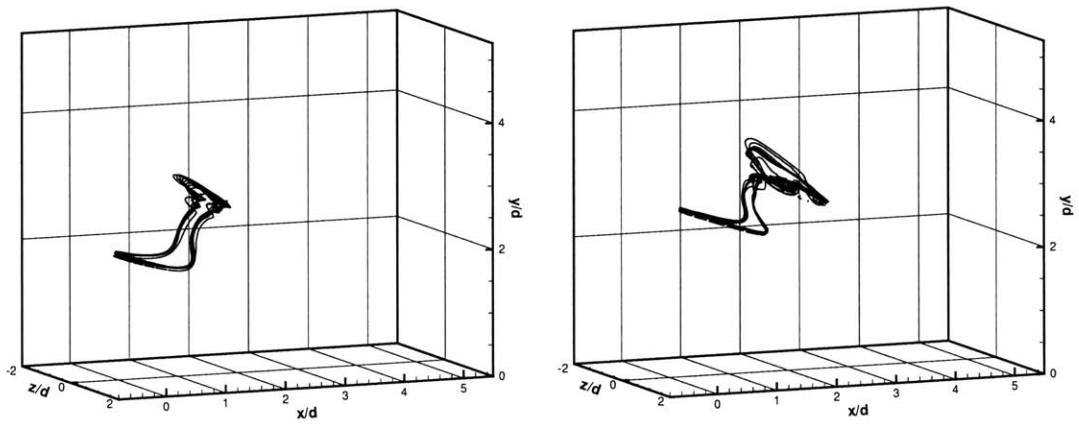


(a) $\tilde{t} = 0.52$ (b) $\tilde{t} = 0.72$



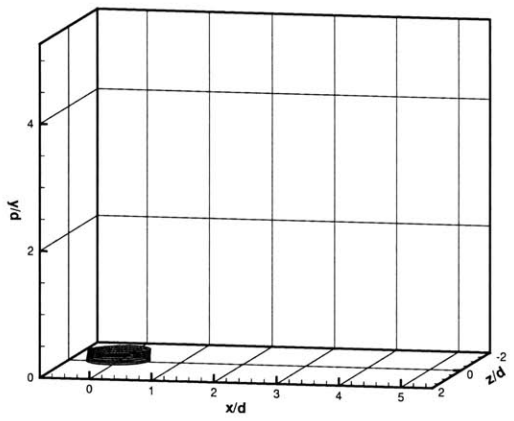
(c) $\tilde{t} = 0.92$

Figure 3-4: Vortex ring folding in the transverse jet, for $r = 7$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [0.45, 0.52]$, corresponding to the second shear layer roll-up.

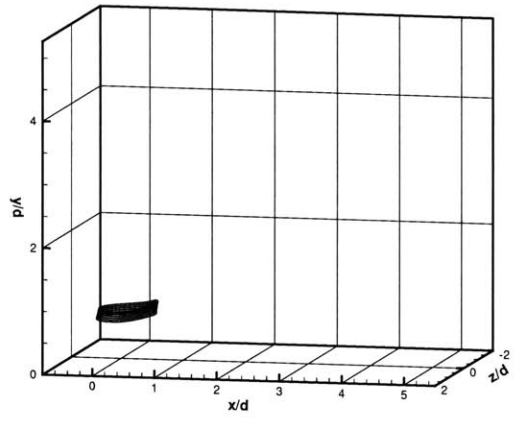


(d) $\tilde{t} = 1.12$ (e) $\tilde{t} = 1.32$

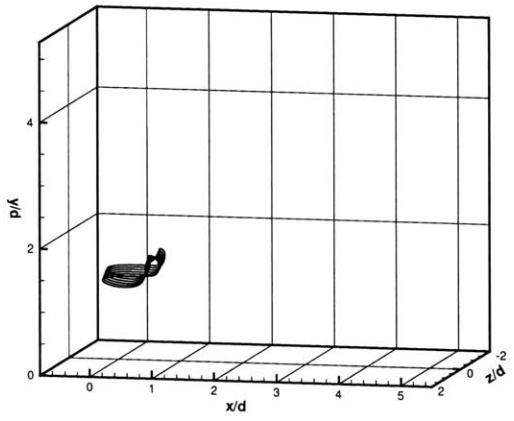
Figure 3-4: Vortex ring folding in the transverse jet, for $r = 7$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [0.45, 0.52]$, corresponding to the second shear layer roll-up (con't).



(a) $\tilde{t} = 1.14$

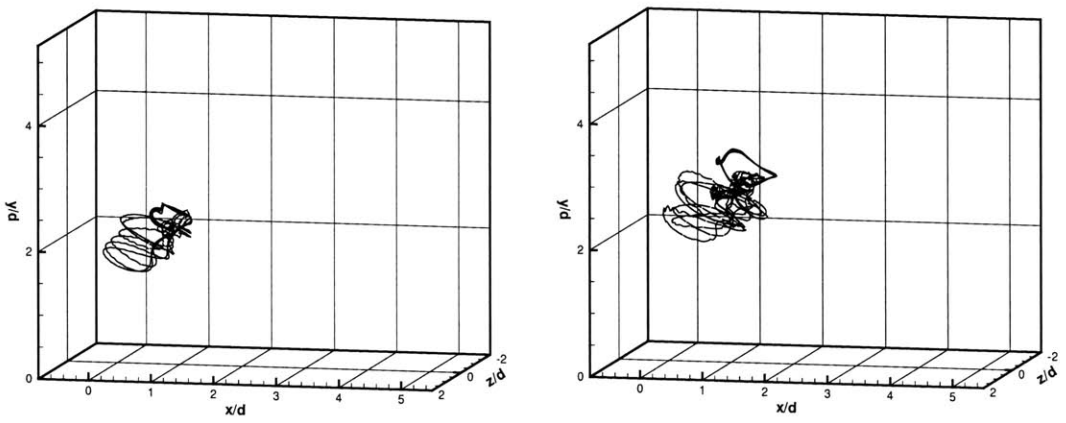


(b) $\tilde{t} = 1.34$



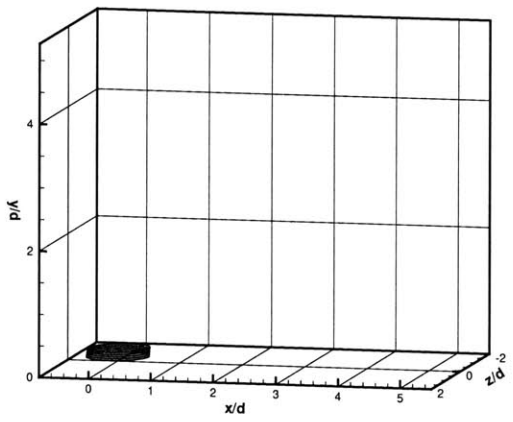
(c) $\tilde{t} = 1.54$

Figure 3-5: Vortex ring folding in the transverse jet, for $r = 7$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [1.09, 1.14]$, corresponding to the 6th lee-side roll-up.

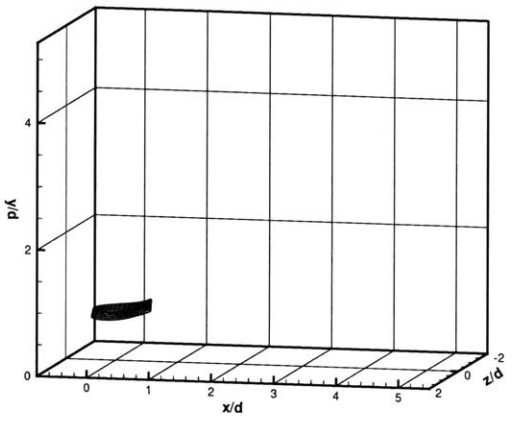


(d) $\tilde{t} = 1.74$ (e) $\tilde{t} = 1.94$

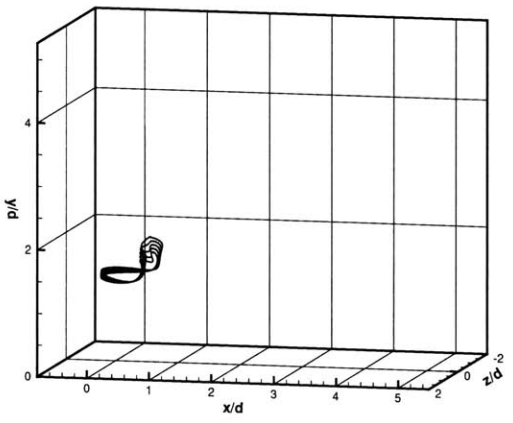
Figure 3-5: Vortex ring folding in the transverse jet, for $r = 7$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [1.09, 1.14]$, corresponding to the 6th lee-side roll-up (con't).



(a) $\tilde{t} = 1.22$

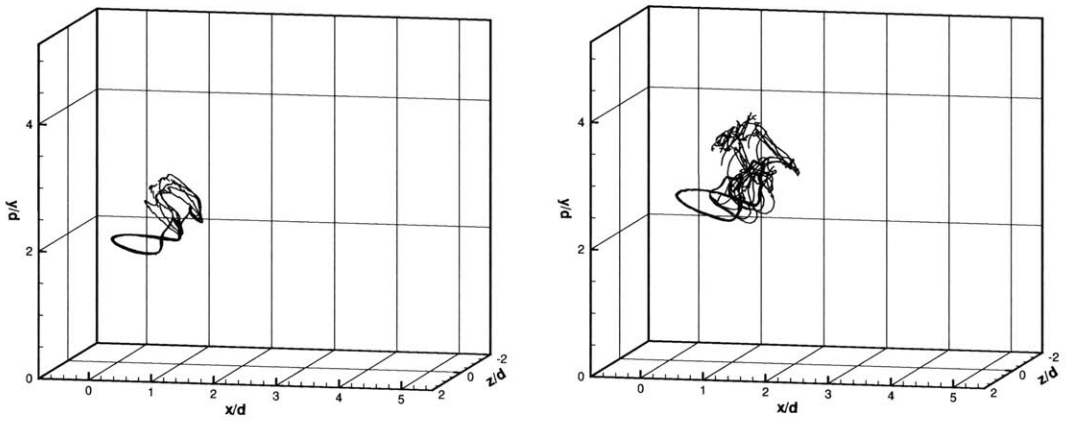


(b) $\tilde{t} = 1.42$



(c) $\tilde{t} = 1.62$

Figure 3-6: Vortex ring folding in the transverse jet, for $r = 7$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [1.16, 1.22]$, corresponding to the 6th windward-side roll-up.



(d) $\tilde{t} = 1.82$ (e) $\tilde{t} = 2.02$

Figure 3-6: Vortex ring folding in the transverse jet, for $r = 7$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [1.16, 1.22]$, corresponding to the 6th windward-side roll-up (con't).

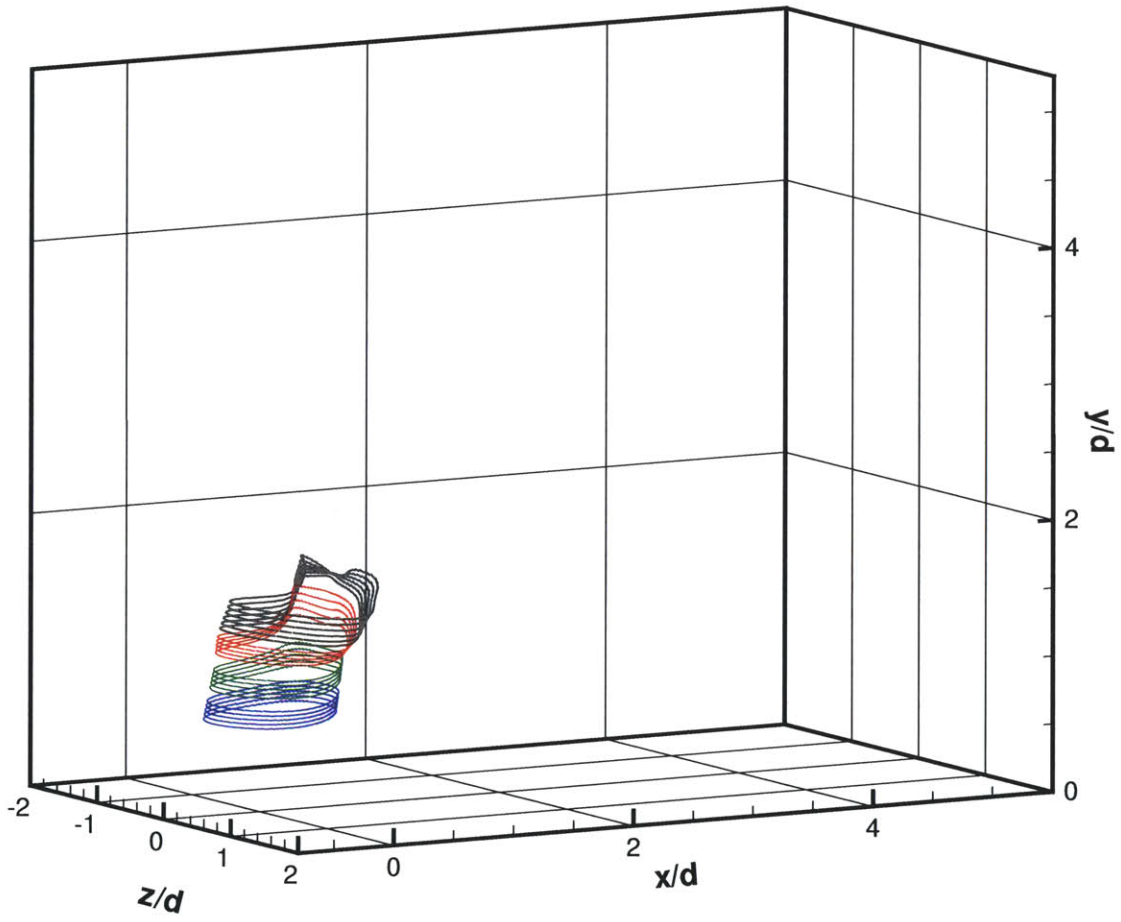
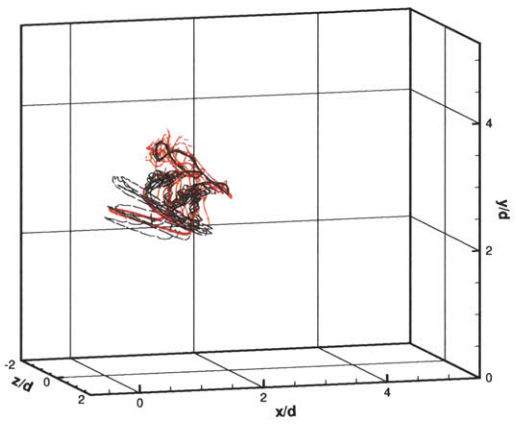
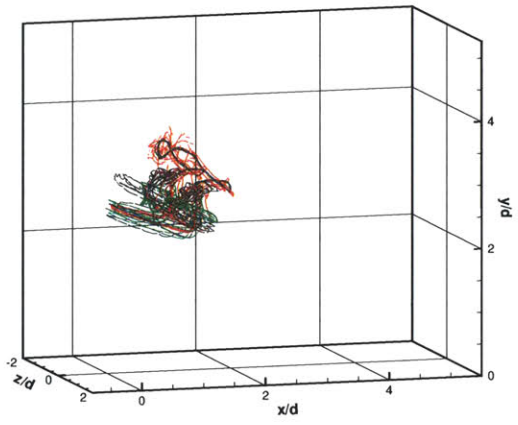


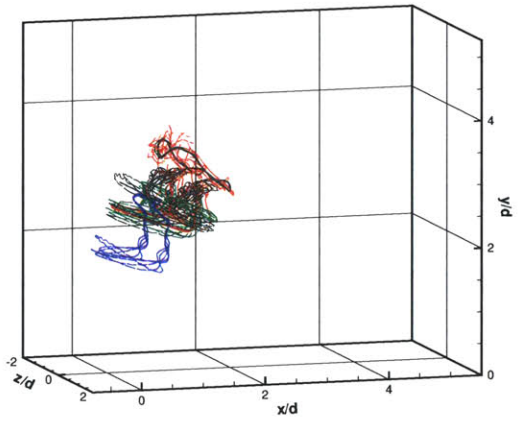
Figure 3-7: Four successive shear layer segments in the $r = 7$ jet, color-coded, shown at $\tilde{t} = 1.56$.



(a) Groups 1 and 2.

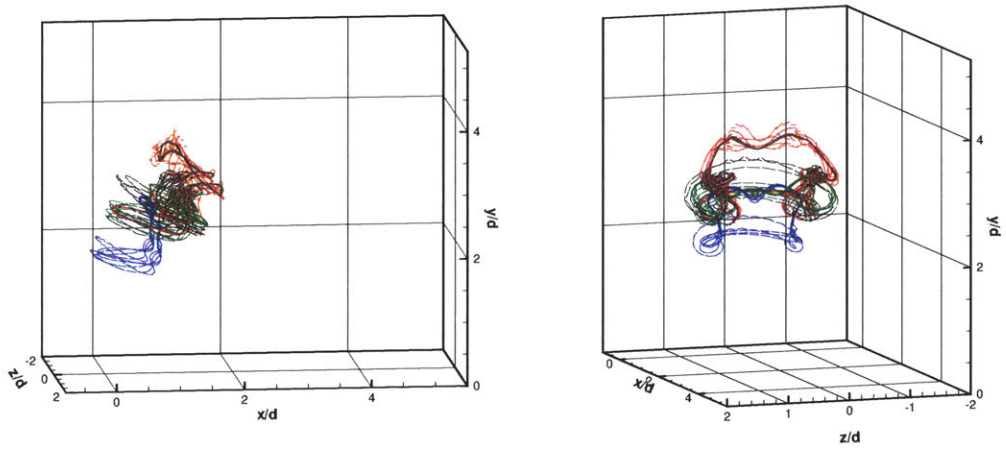


(b) Groups 1, 2, and 3.



(c) Groups 1, 2, 3, and 4.

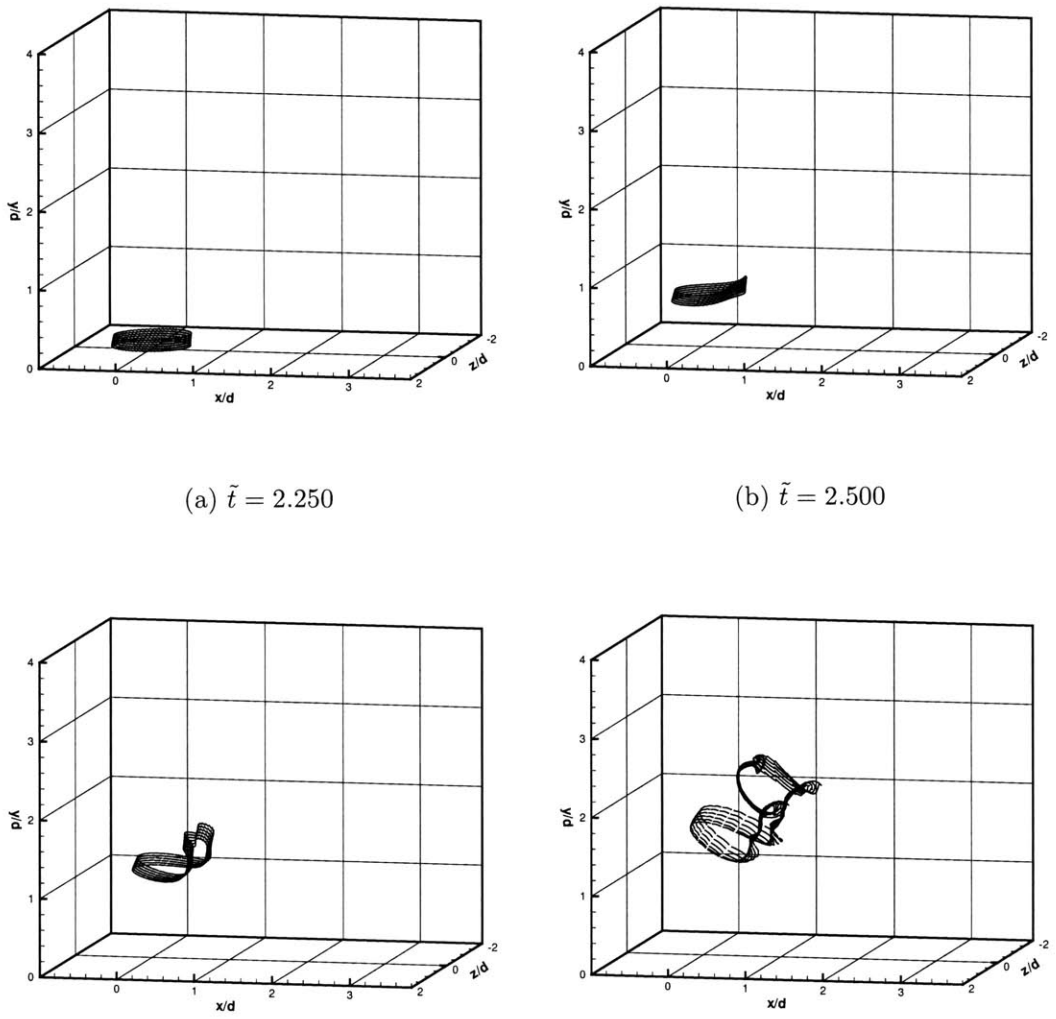
Figure 3-8: Shear layer segments from Figure 3-7 shown at $\tilde{t} = 2.00$; $r = 7$.



(a) Side view.

(b) Downstream view.

Figure 3-9: Shear layer segments from Figure 3-8 shown at $\tilde{t} = 2.00$; alternate perspective view



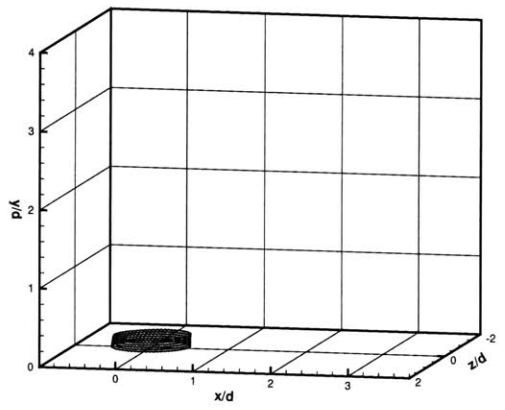
(a) $\tilde{t} = 2.250$

(b) $\tilde{t} = 2.500$

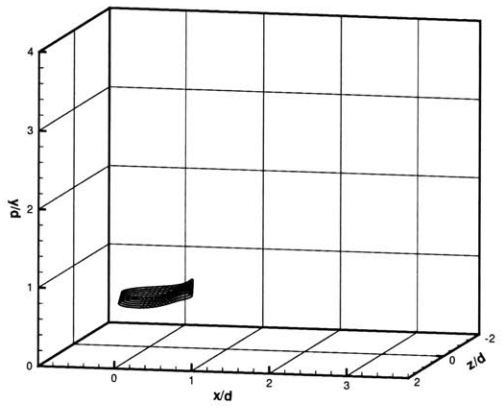
(c) $\tilde{t} = 2.750$

(d) $\tilde{t} = 3.000$

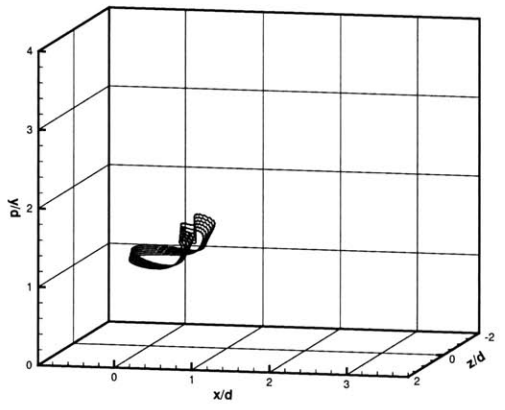
Figure 3-10: Vortex ring folding in the transverse jet for $r = 5$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [2.175, 2.2375]$.



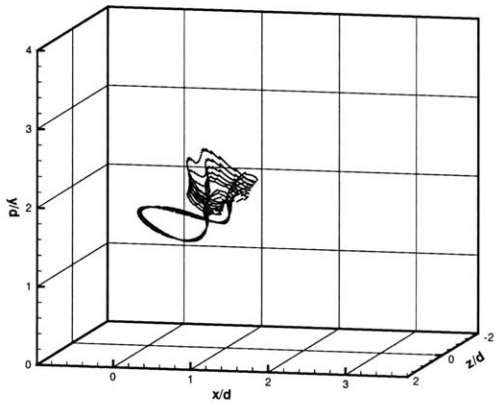
(a) $\tilde{t} = 2.325$



(b) $\tilde{t} = 2.575$

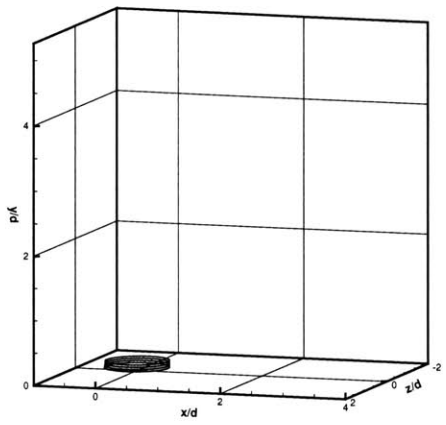


(c) $\tilde{t} = 2.825$

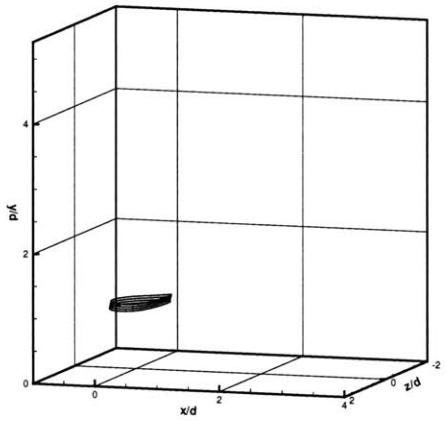


(d) $\tilde{t} = 3.075$

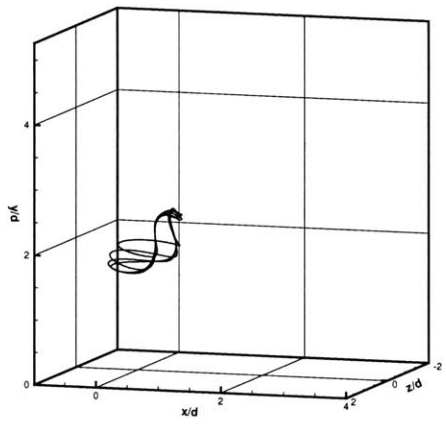
Figure 3-11: Vortex ring folding in the transverse jet for $r = 5$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [2.2625, 2.325]$.



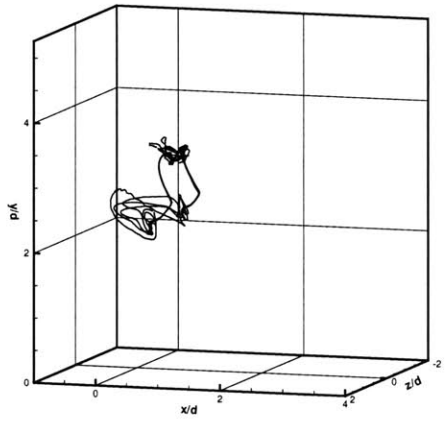
(a) $\tilde{t} = 1.170$



(b) $\tilde{t} = 1.365$

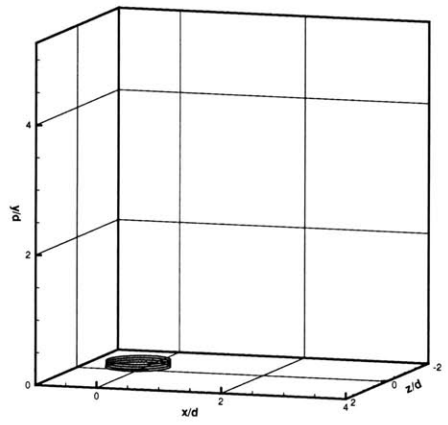


(c) $\tilde{t} = 1.560$

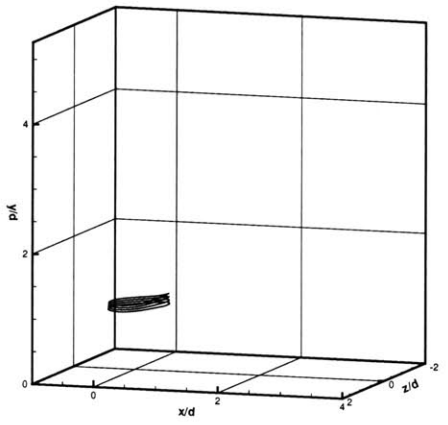


(d) $\tilde{t} = 1.755$

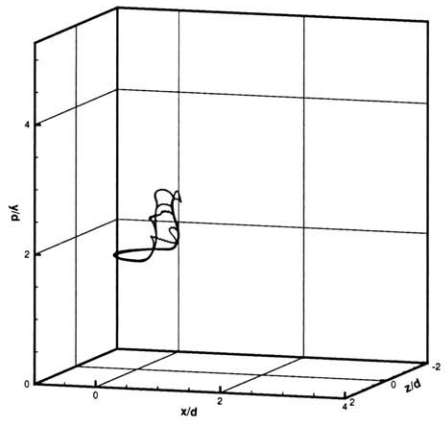
Figure 3-12: Vortex ring folding in the transverse jet for $r = 10$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [1.1400, 1.1625]$.



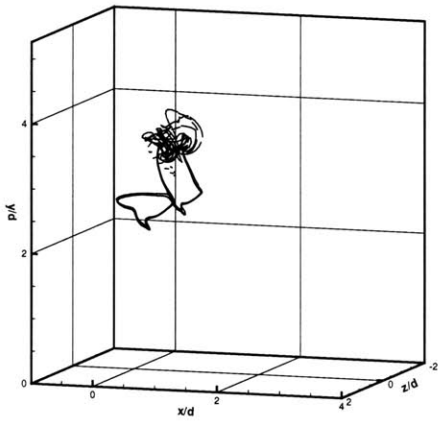
(a) $\tilde{t} = 1.200$



(b) $\tilde{t} = 1.395$

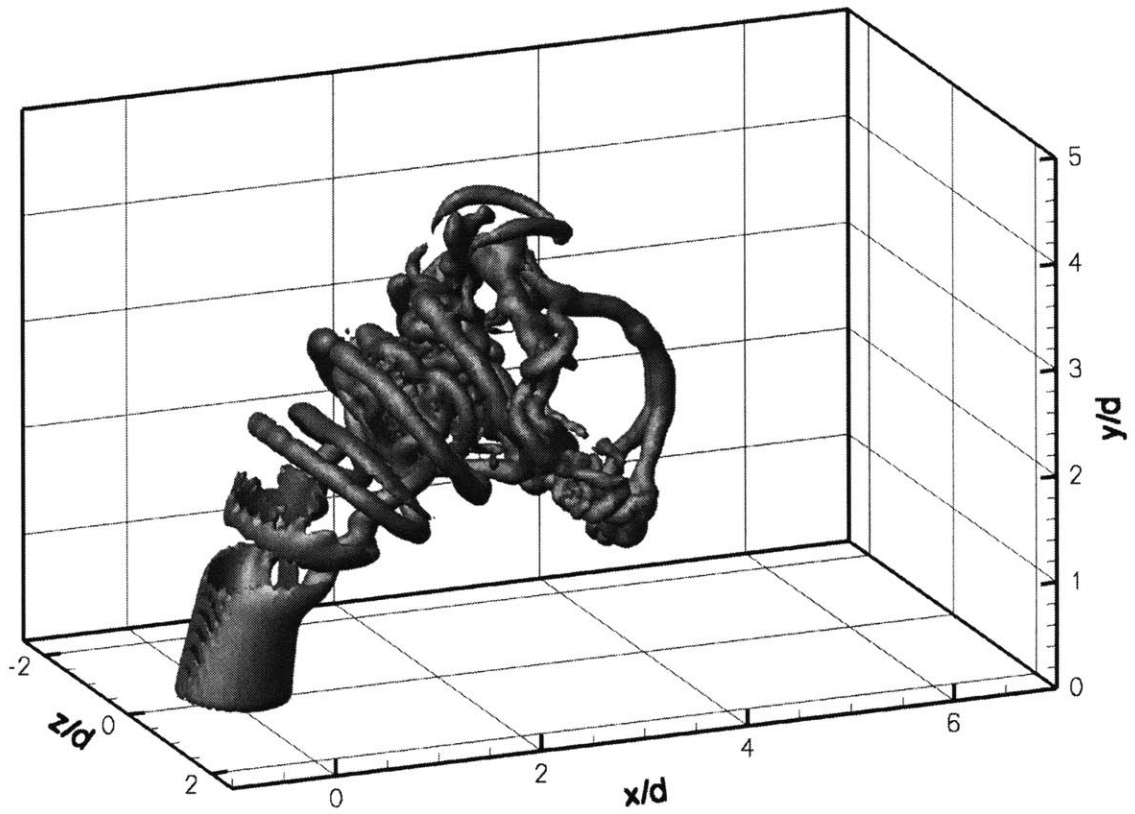


(c) $\tilde{t} = 1.590$



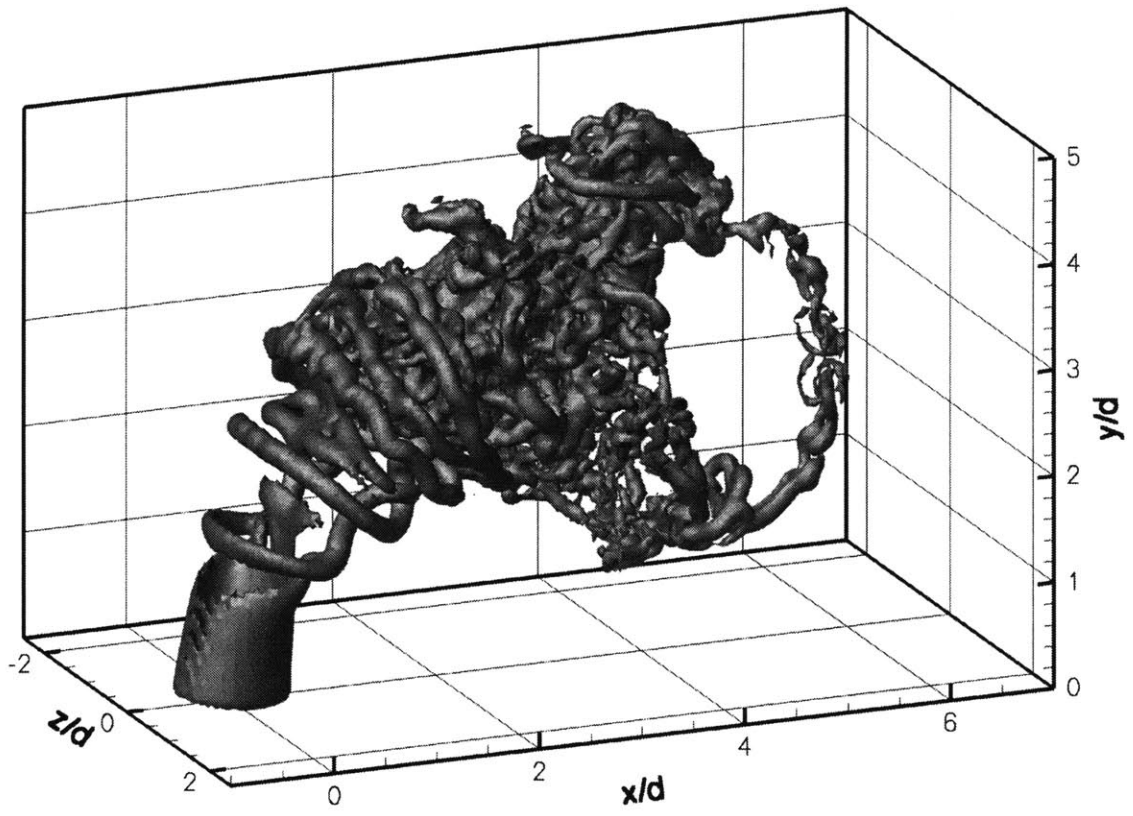
(d) $\tilde{t} = 1.785$

Figure 3-13: Vortex ring folding in the transverse jet for $r = 10$. Snapshots follow the evolution of vortex elements introduced for $\tilde{t} \in [1.1700, 1.1925]$.



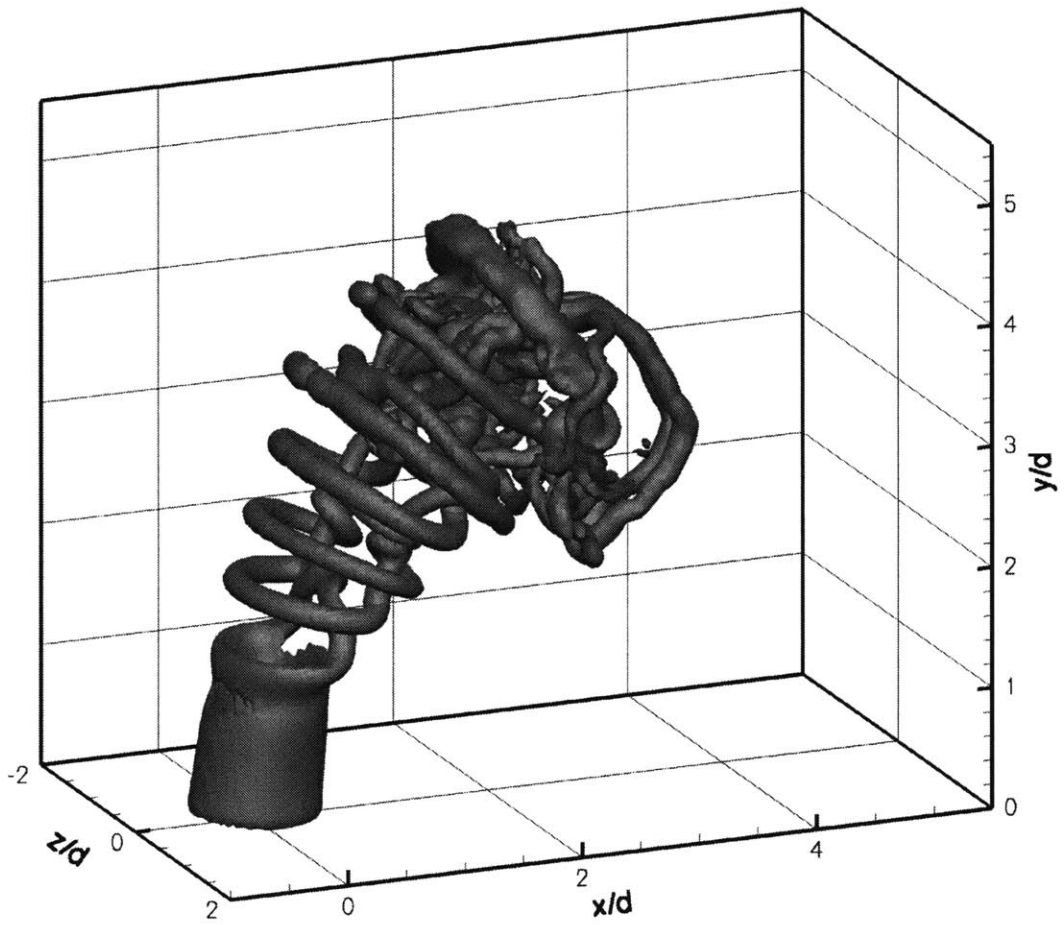
(a) $\tilde{t} = 2.40$

Figure 3-14: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 28.0$ in the $r = 5$ jet, at two successive times.



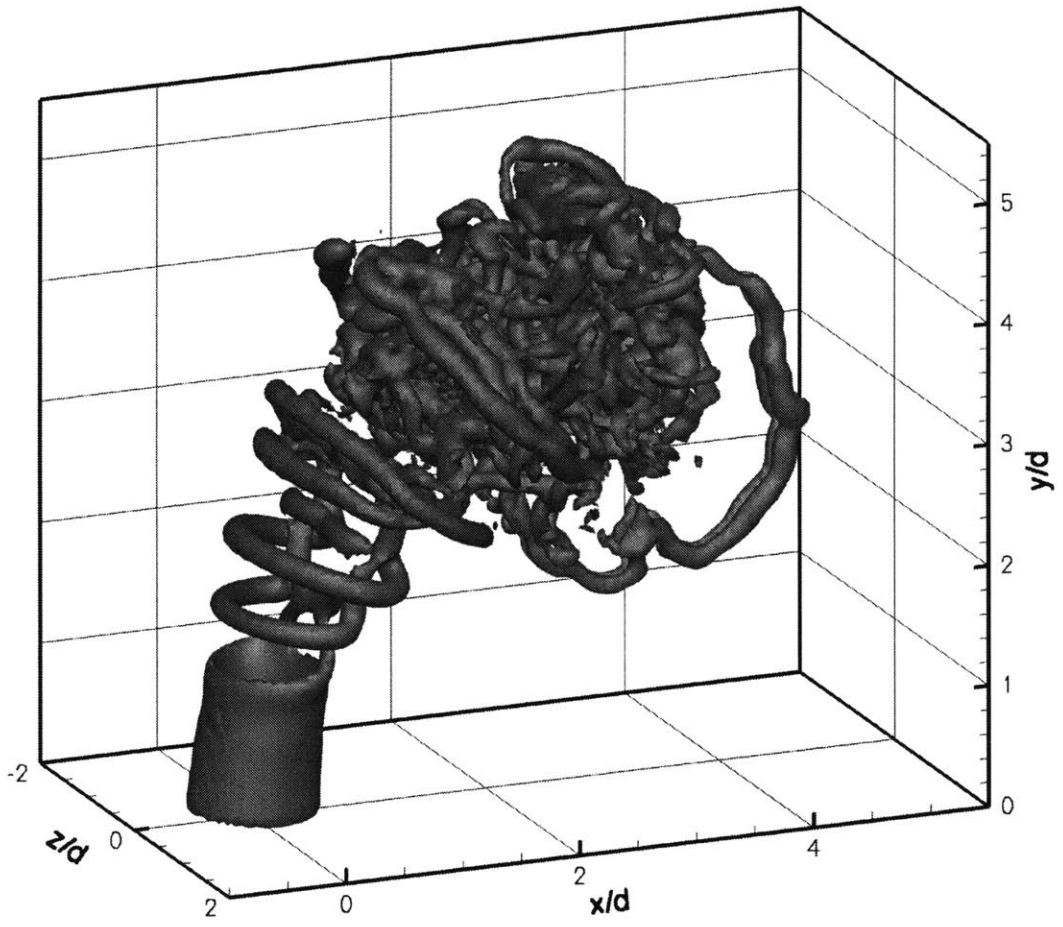
(b) $\tilde{t} = 3.00$

Figure 3-14: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 28.0$ in the $r = 5$ jet, at two successive times (con't).



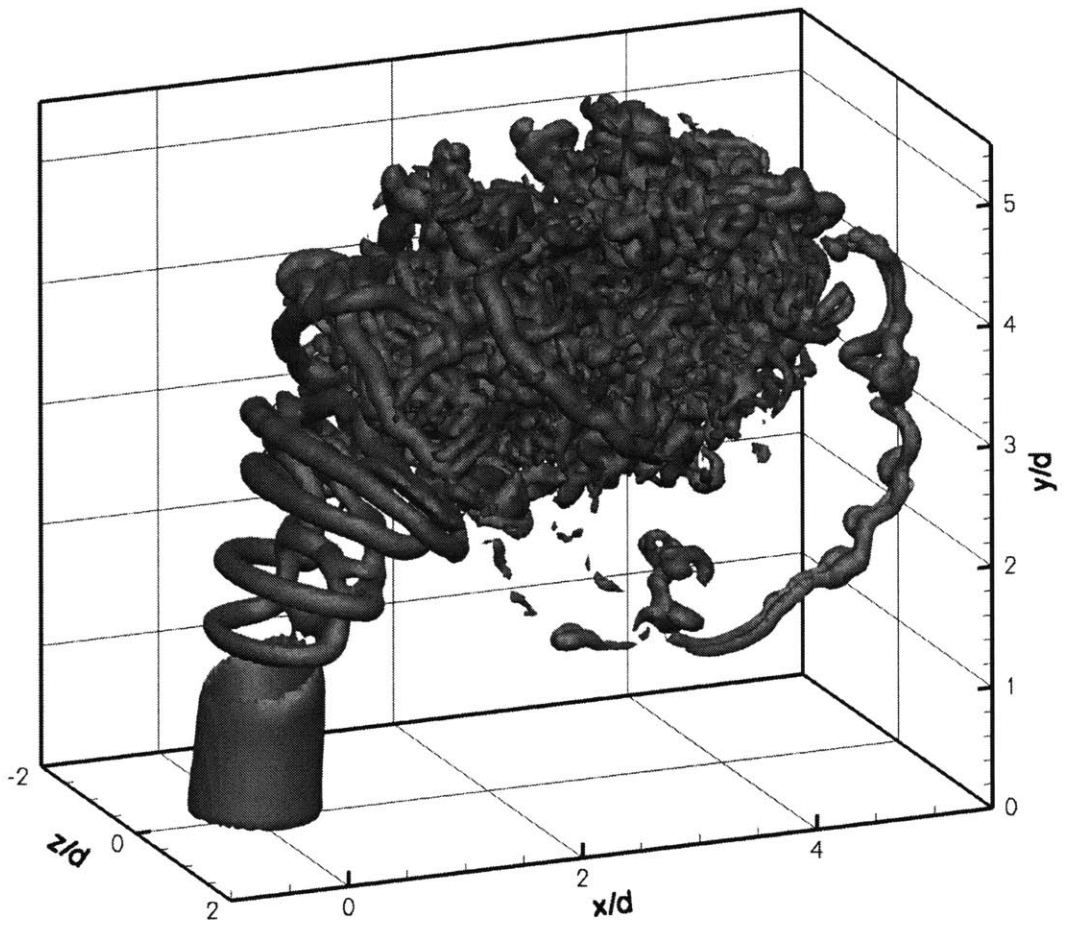
(a) $\tilde{t} = 1.80$

Figure 3-15: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 40.0$ in the $r = 7$ jet, at three successive times.



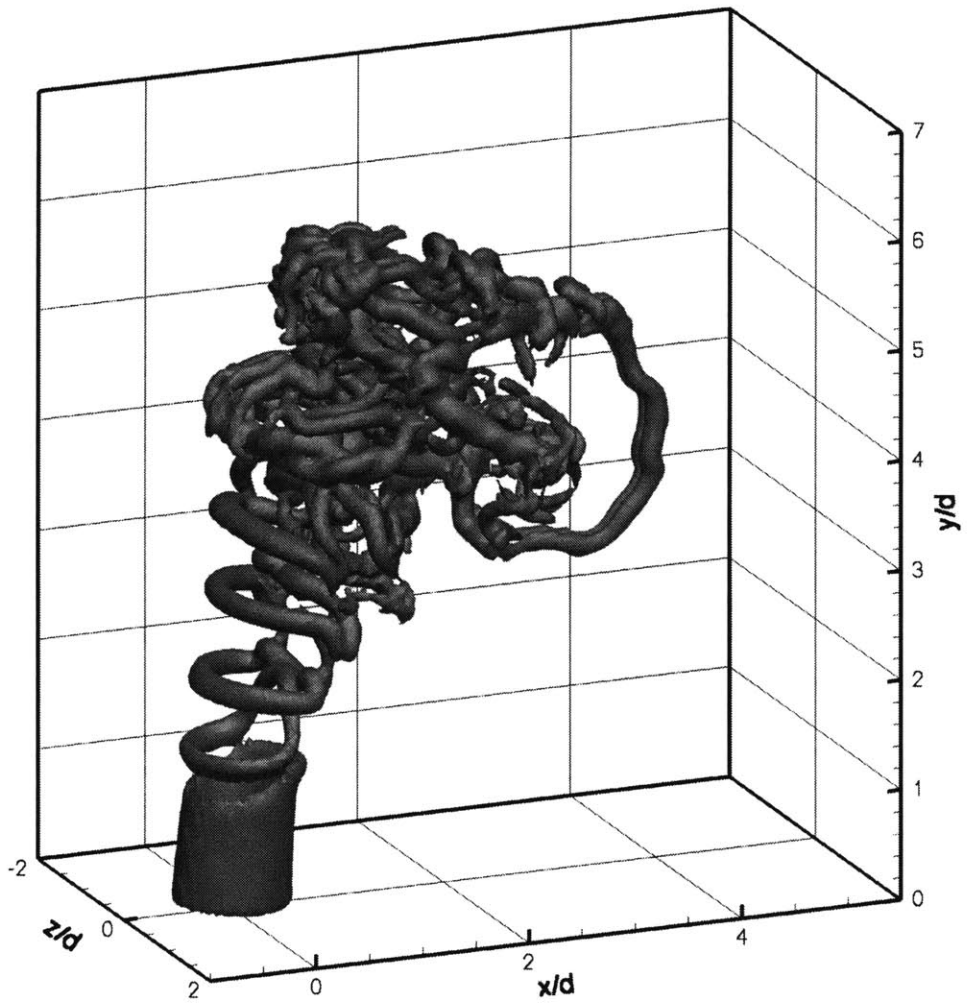
(b) $\tilde{t} = 2.10$

Figure 3-15: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 40.0$ in the $r = 7$ jet, at three successive times (con't).



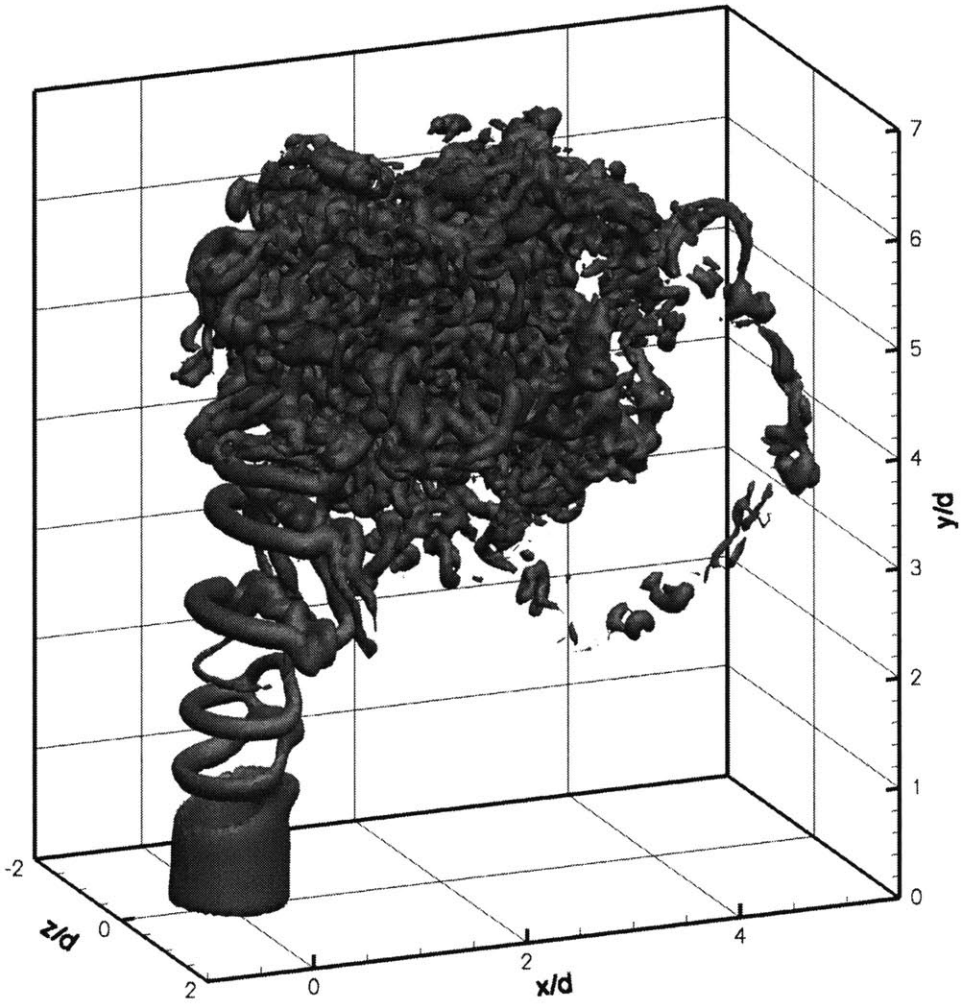
(c) $\tilde{t} = 2.40$

Figure 3-15: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 40.0$ in the $r = 7$ jet, at three successive times (con't).



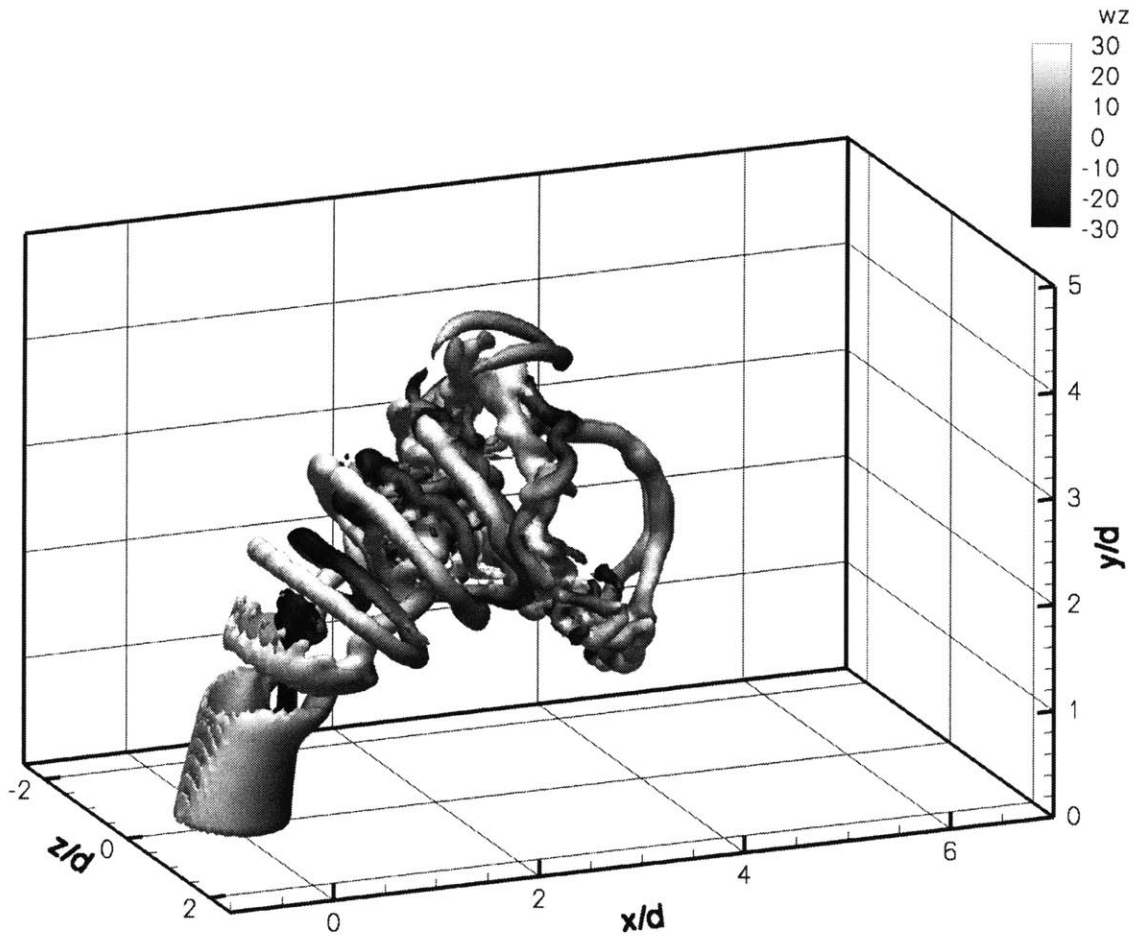
(a) $\tilde{t} = 1.50$

Figure 3-16: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 60.0$ in the $r = 10$ jet, at two successive times.



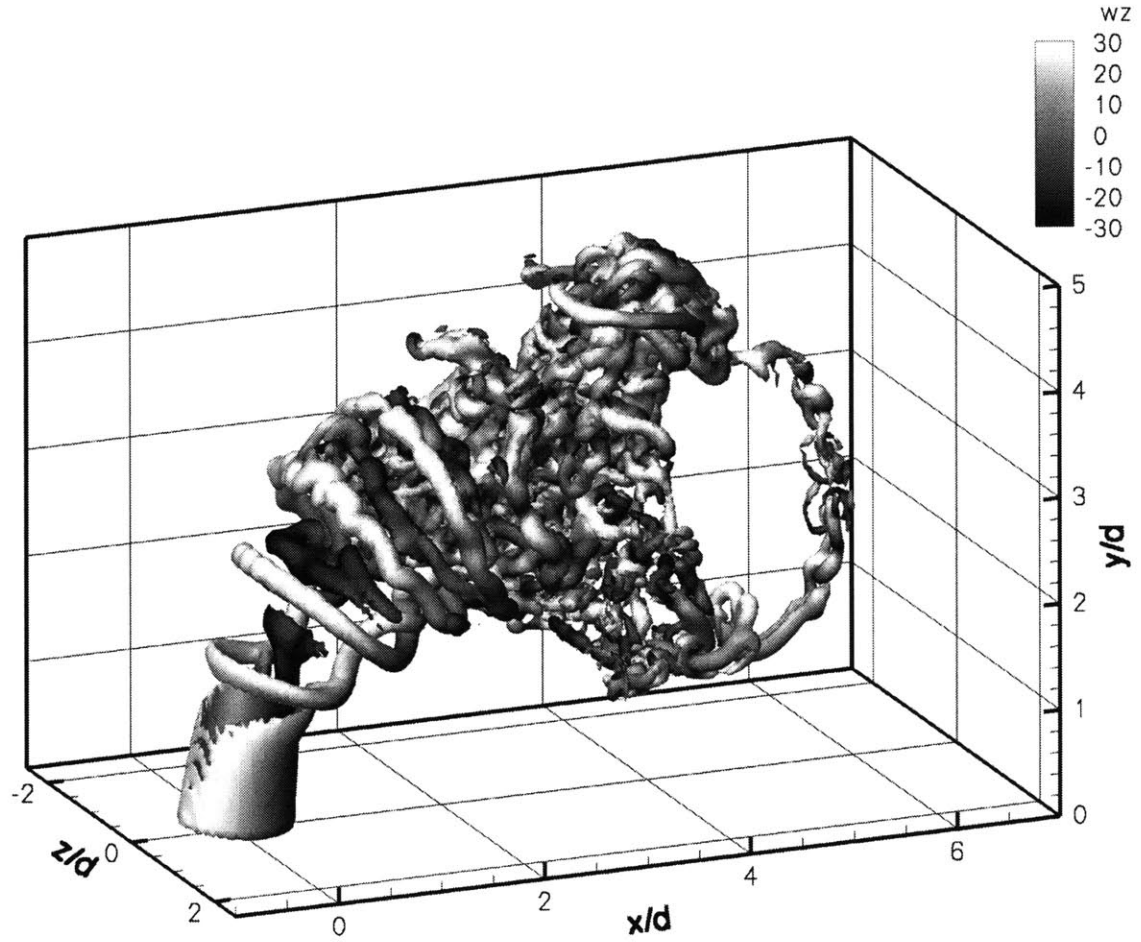
(b) $\tilde{t} = 1.80$

Figure 3-16: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 60.0$ in the $r = 10$ jet, at two successive times (con't).



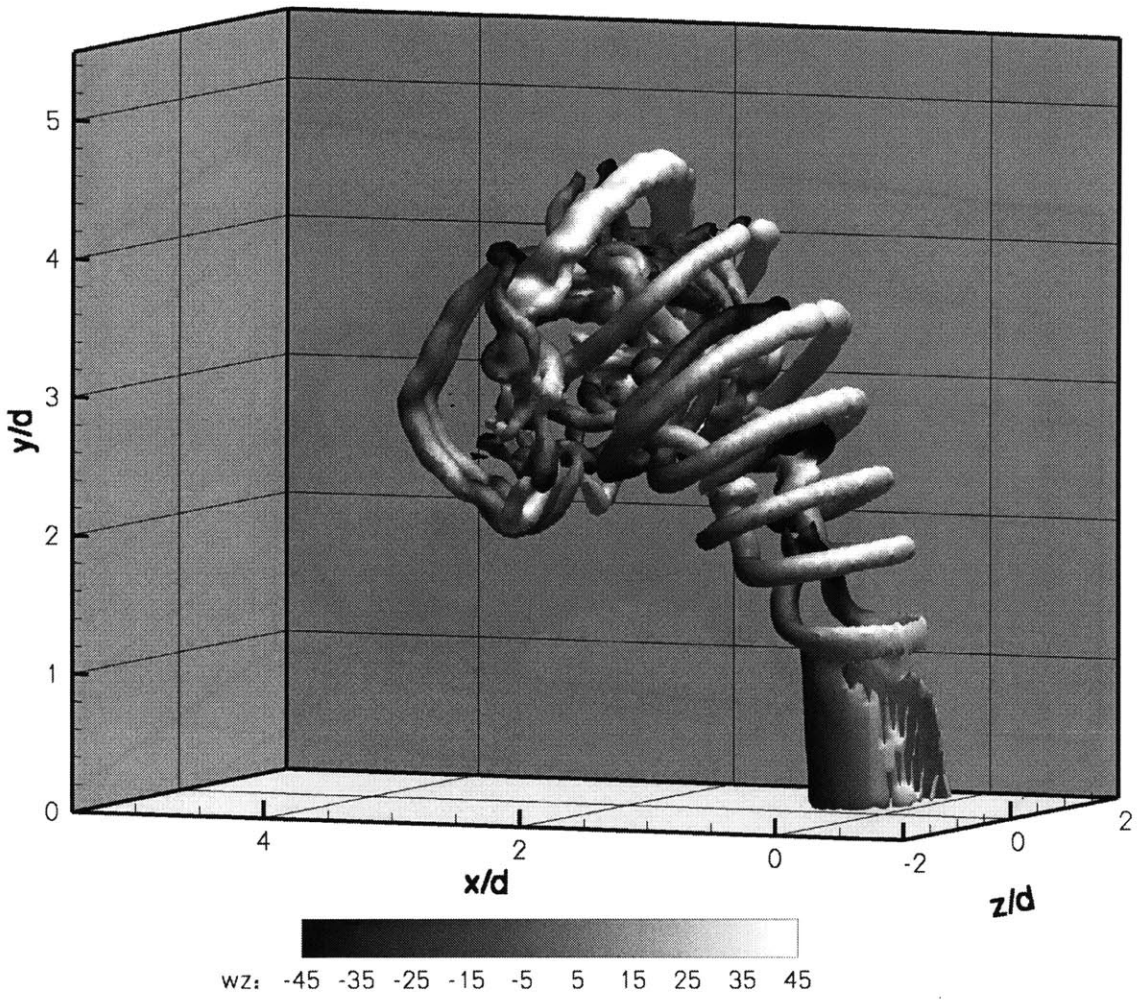
(a) $\tilde{t} = 2.40$

Figure 3-17: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 28.0$ contoured by spanwise vorticity ω_z at two successive times; $r = 5$.



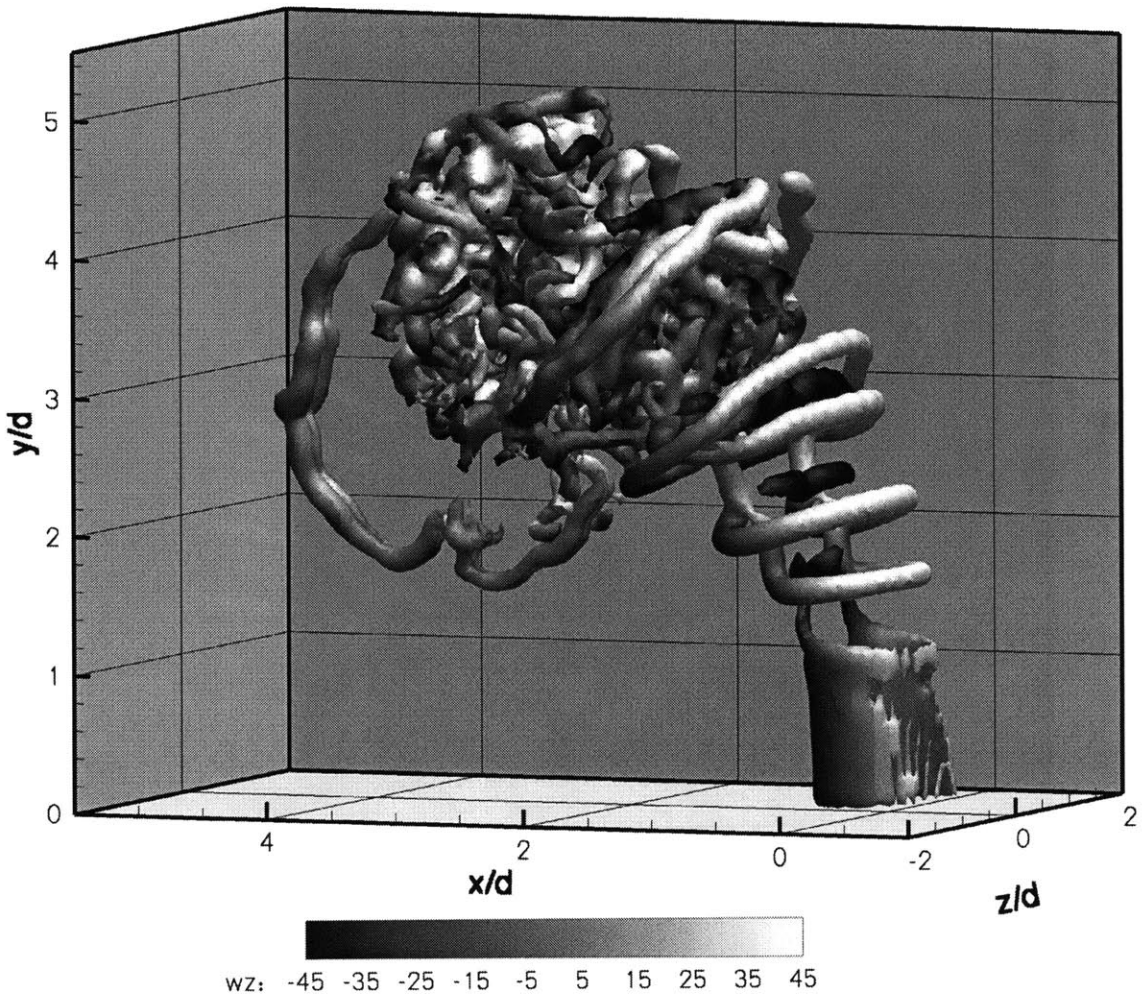
(b) $\tilde{t} = 3.00$

Figure 3-17: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 28.0$ contoured by spanwise vorticity ω_z at two successive times; $r = 5$ (con't).



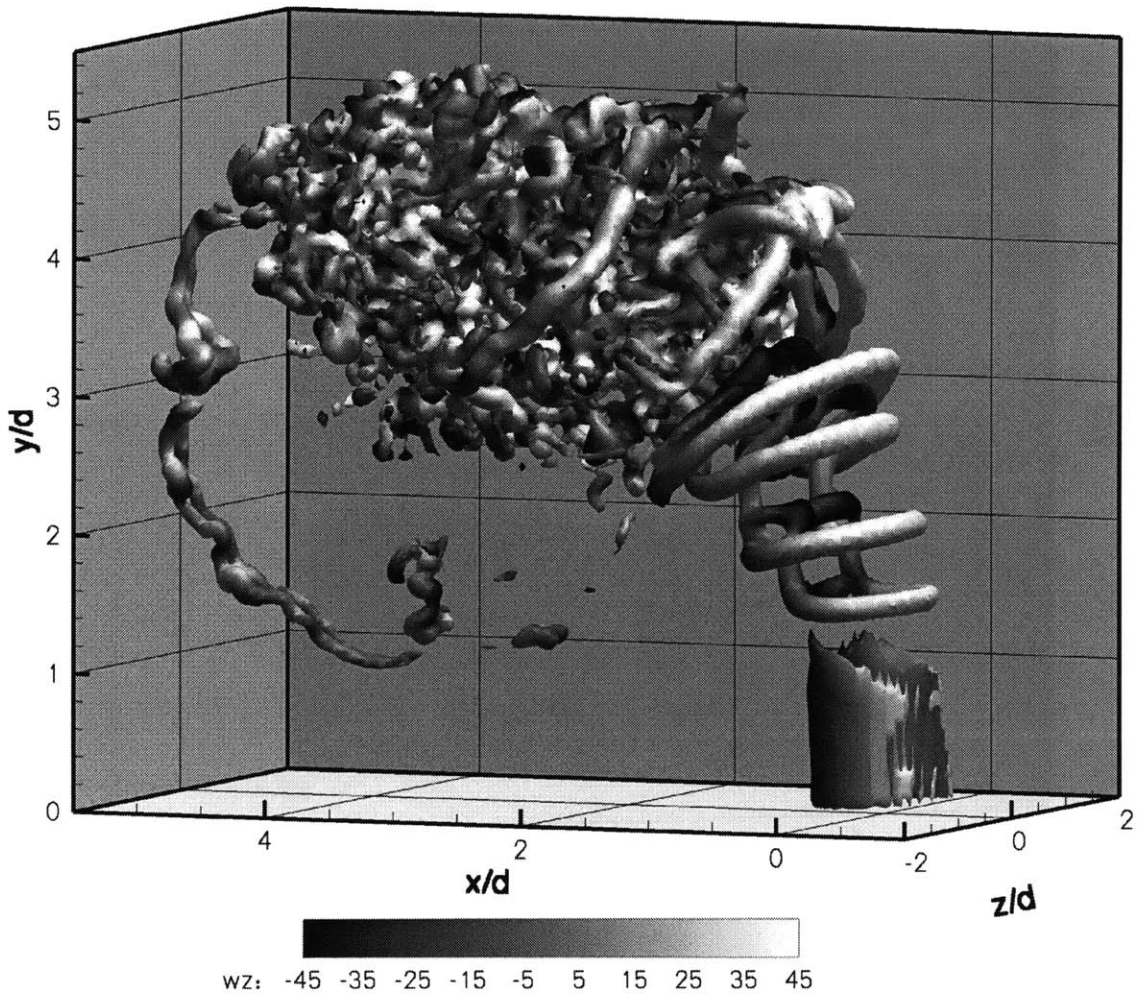
(a) $\tilde{t} = 1.80$

Figure 3-18: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 45.0$ in the $r = 7$ jet contoured by spanwise vorticity ω_z, at three successive times.



(b) $\tilde{t} = 2.10$

Figure 3-18: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 45.0$ in the $r = 7$ jet contoured by spanwise vorticity ω_z, at three successive times (con't).



(c) $\tilde{t} = 2.40$

Figure 3-18: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 45.0$ in the $r = 7$ jet contoured by spanwise vorticity ω_z, at three successive times (con't).

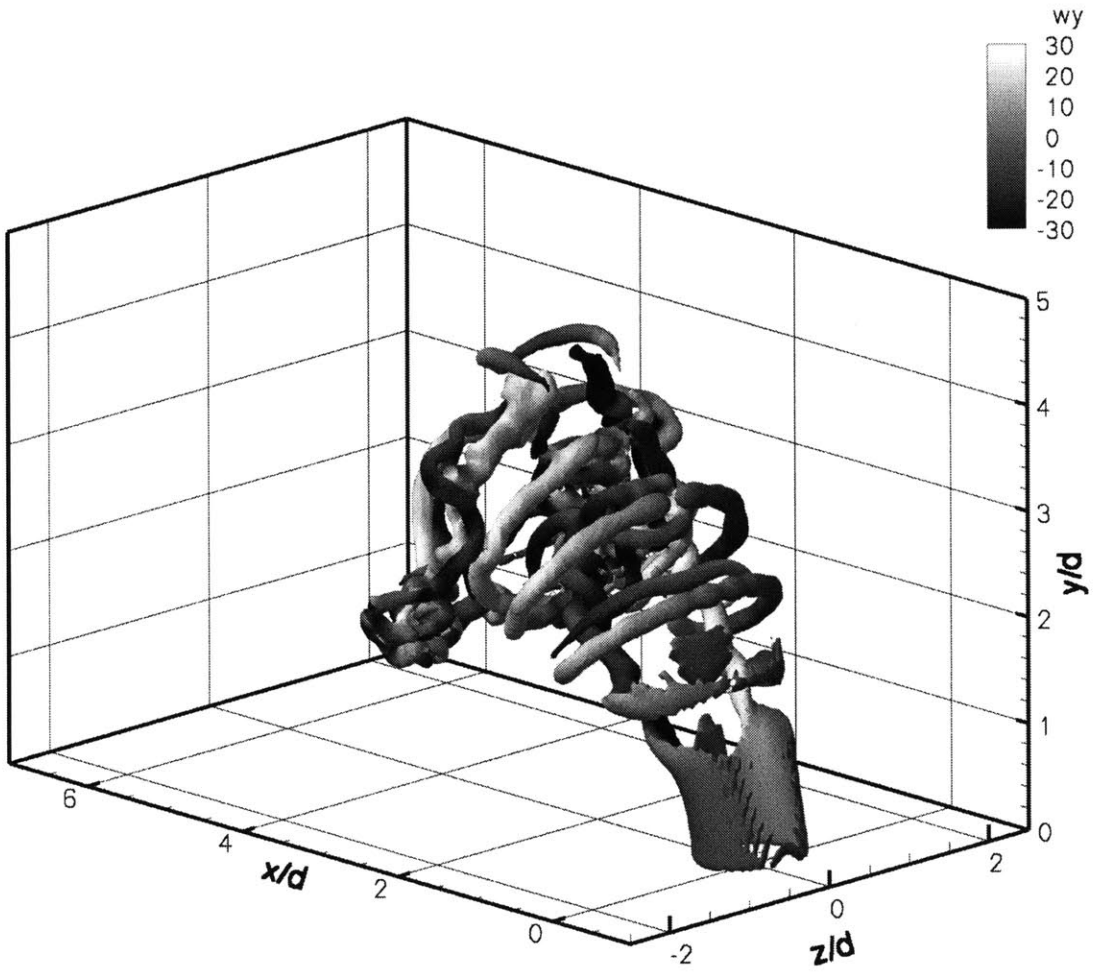


Figure 3-19: Vorticity magnitude isosurface $\|\boldsymbol{\omega}\|_2 = 30.0$ contoured by wall-normal vorticity ω_y in the $r = 5$ jet at $\tilde{t} = 2.40$.

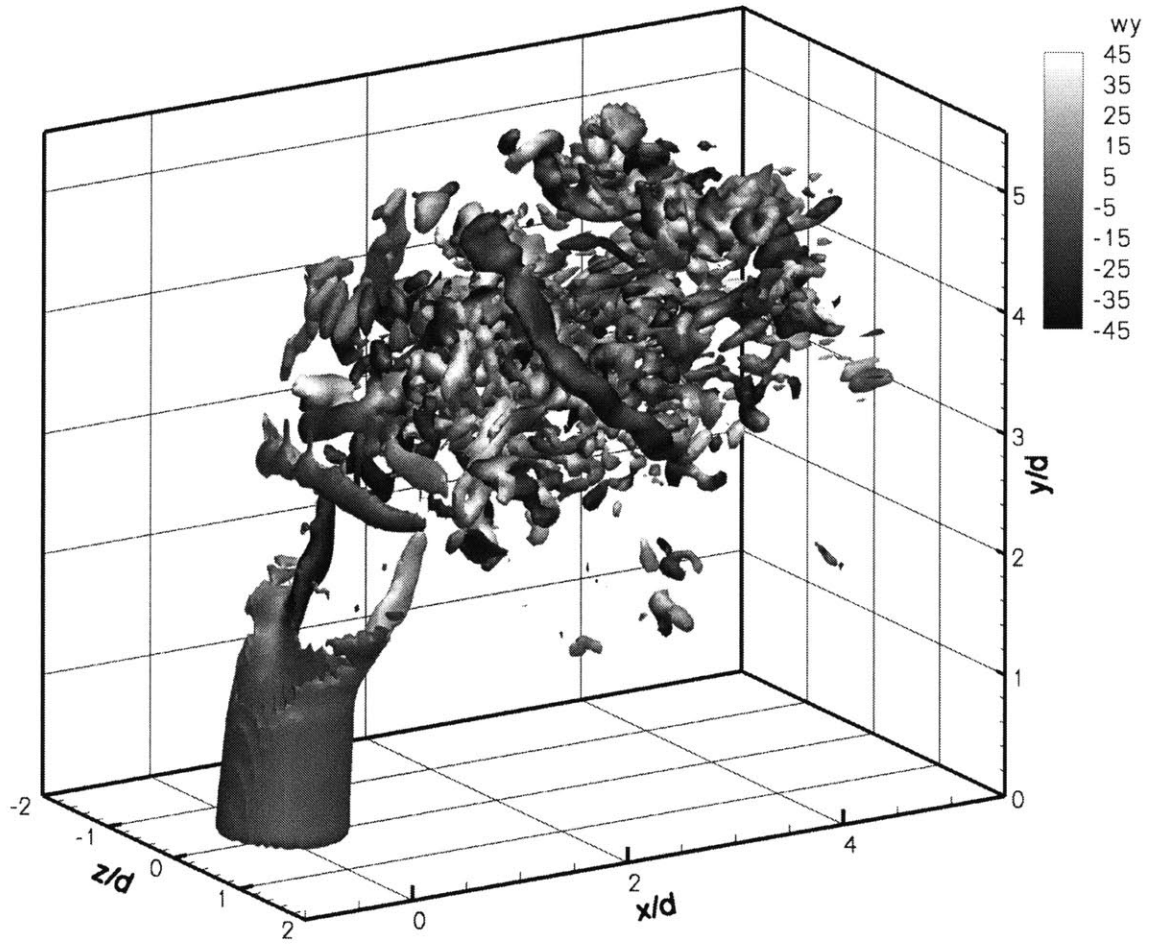
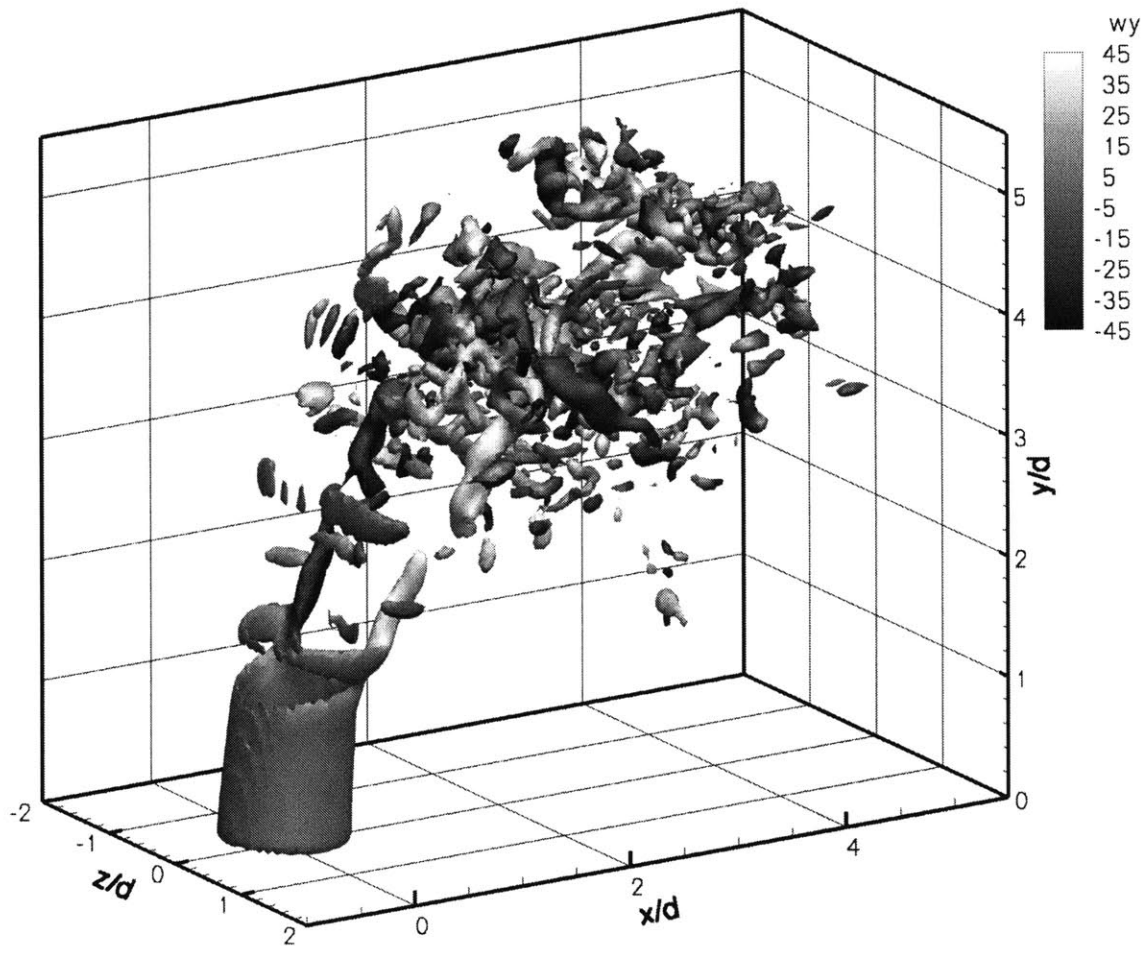
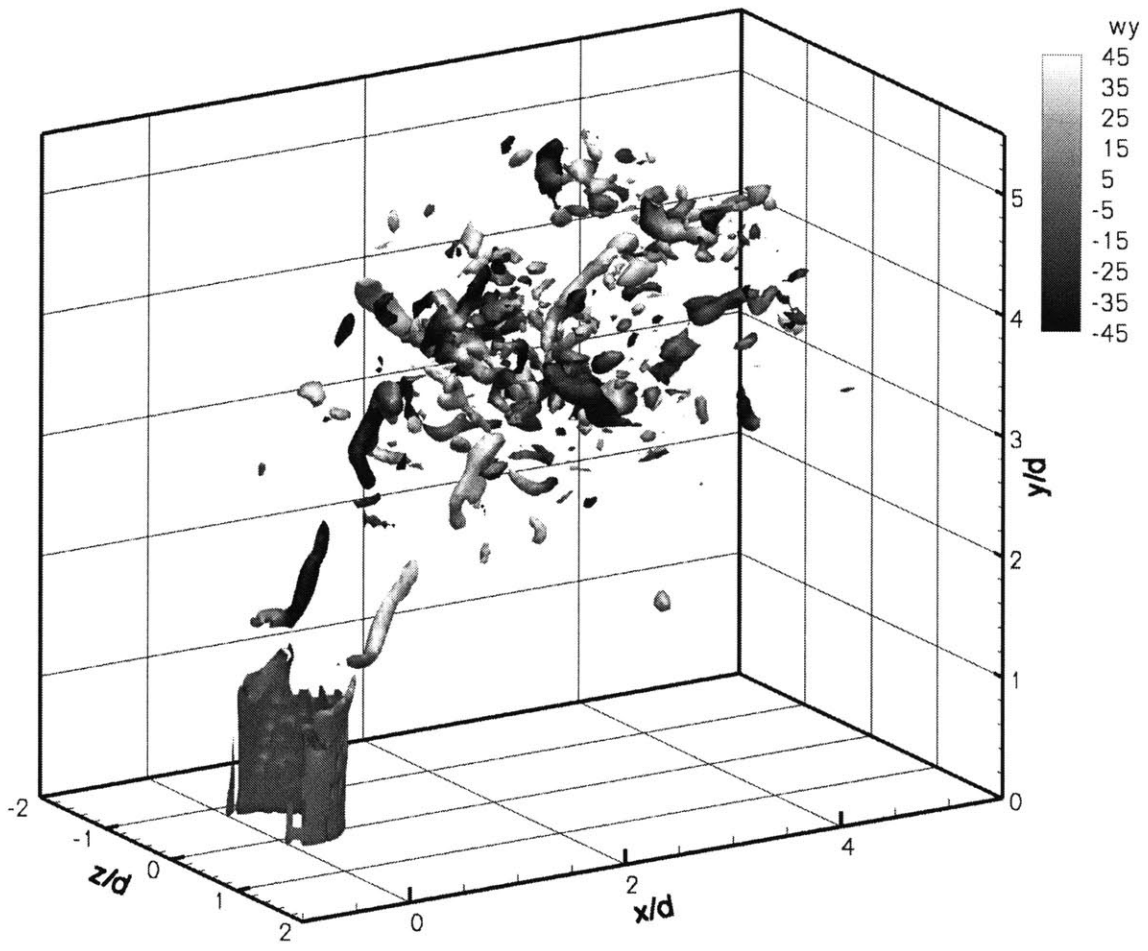


Figure 3-20: Isosurface of $\|\bar{\boldsymbol{\omega}}\|_2 = 40.0$ contoured by the mean wall-normal vorticity $\bar{\omega}_y$ in the $r = 7$ jet; $\bar{\boldsymbol{\omega}}$ is the mean vorticity field over the interval $\tilde{t} \in [2.31, 2.49]$, which corresponds to a single cycle of shear layer roll-up.



(a) $\|\bar{\omega}\|_2 = 40.0$

Figure 3-21: Isosurfaces of $\|\bar{\boldsymbol{\omega}}\|_2$ contoured by mean wall-normal vorticity $\bar{\omega}_y$ in the $r = 7$ jet; $\bar{\boldsymbol{\omega}}$ is the mean vorticity field over the interval $\tilde{t} \in [2.23, 2.49]$.



(b) $\|\bar{\boldsymbol{\omega}}\|_2 = 48.0$

Figure 3-21: Isosurfaces of $\|\bar{\boldsymbol{\omega}}\|_2$ contoured by mean wall-normal vorticity $\bar{\omega}_y$ in the $r = 7$ jet; $\bar{\boldsymbol{\omega}}$ is the mean vorticity field over the interval $\tilde{t} \in [2.23, 2.49]$ (con't).

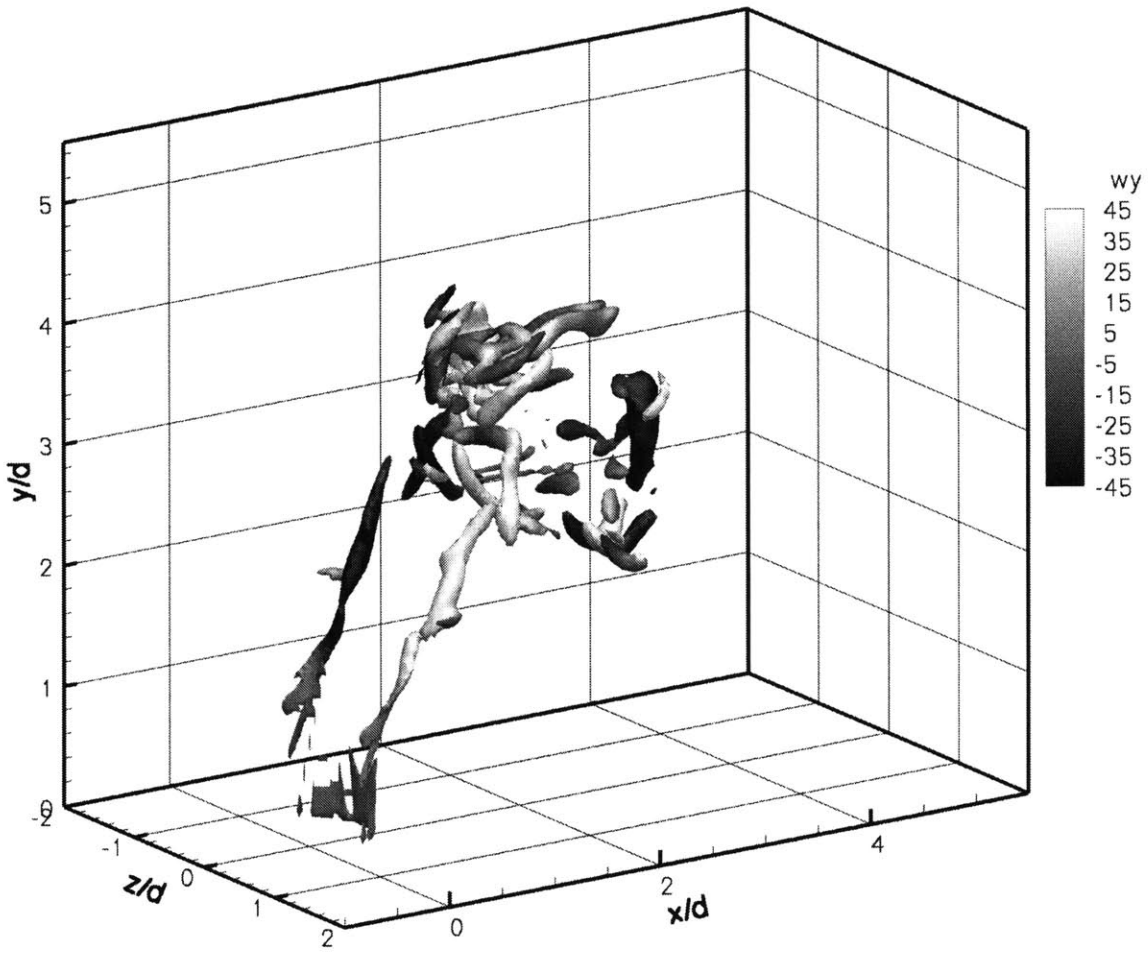
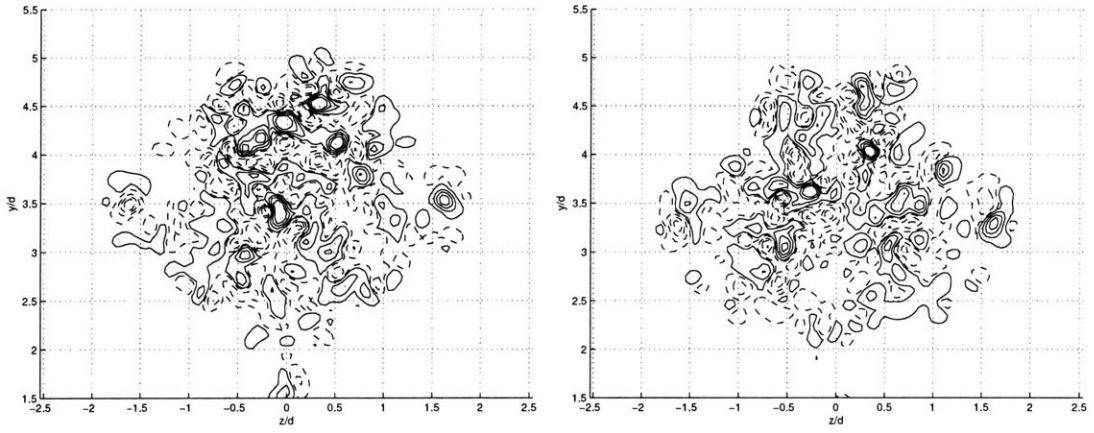
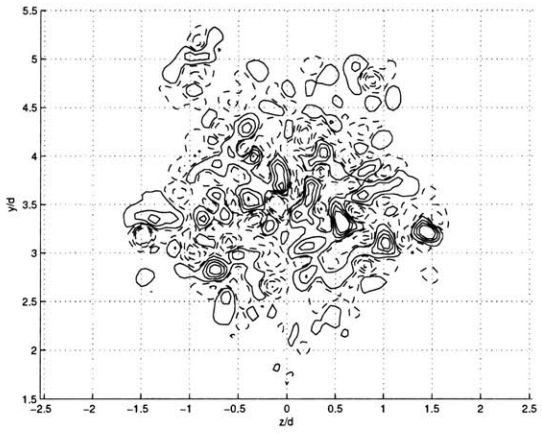


Figure 3-22: Isosurface of $\|\bar{\boldsymbol{\omega}}\|_2 = 52.0$ contoured by mean wall-normal vorticity $\bar{\omega}_y$ in the $r = 7$ jet; $\bar{\boldsymbol{\omega}}$ is the mean vorticity field over the interval $\tilde{t} \in [1.71, 1.89]$, which corresponds to a single cycle of shear layer roll-up.



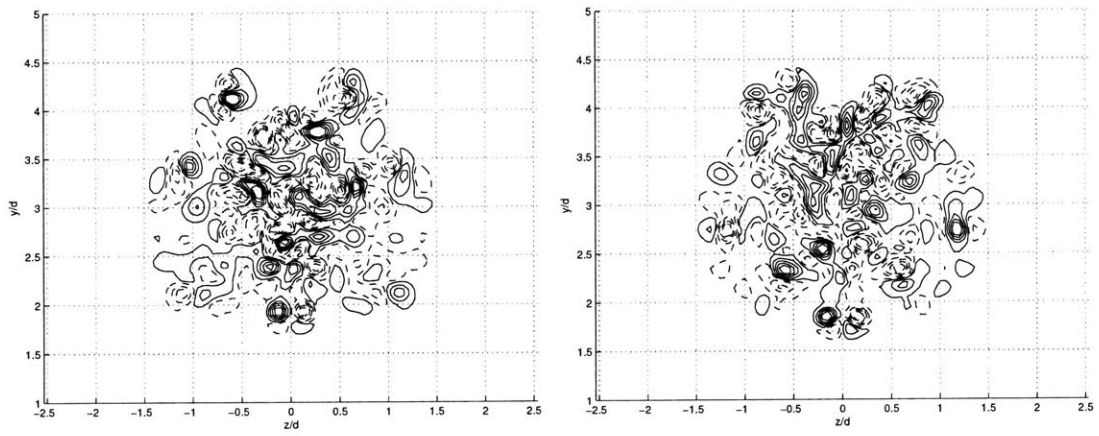
(a) $x/d = 2.25$

(b) $x/d = 2.50$



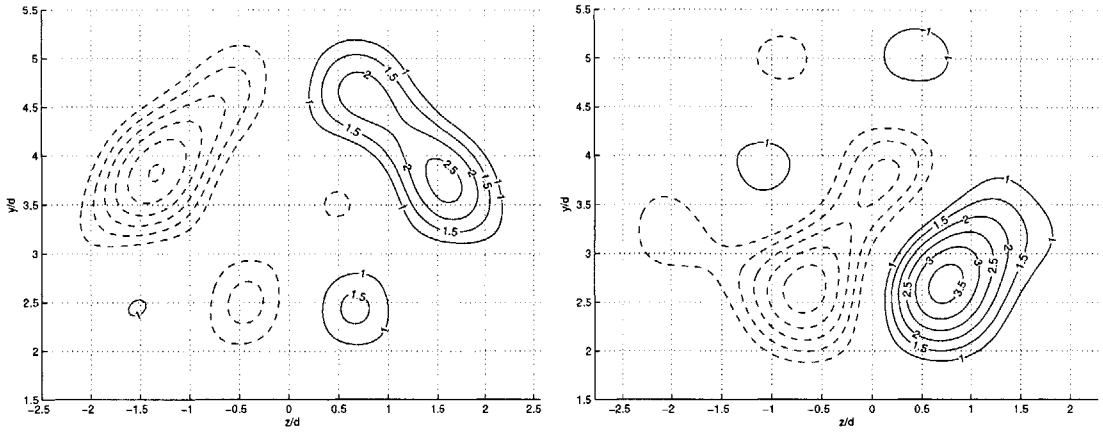
(c) $x/d = 2.75$

Figure 3-23: Transverse slices of streamwise vorticity at $\tilde{t} = 2.40$ in the $r = 7$ jet. Solid contours indicate positive values $\omega_x = 5, 25, 45, 65$; dashed contours indicate the corresponding negative values.



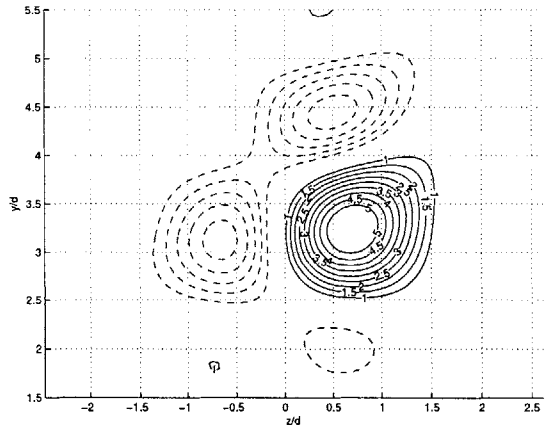
(a) $x/d = 2.25$ (b) $x/d = 2.50$

Figure 3-24: Transverse slices of streamwise vorticity at $\tilde{t} = 3.00$ in the $r = 5$ jet. Solid contours indicate positive values $\omega_x = 3, 13, 33, 43$; dashed contours indicate the corresponding negative values.



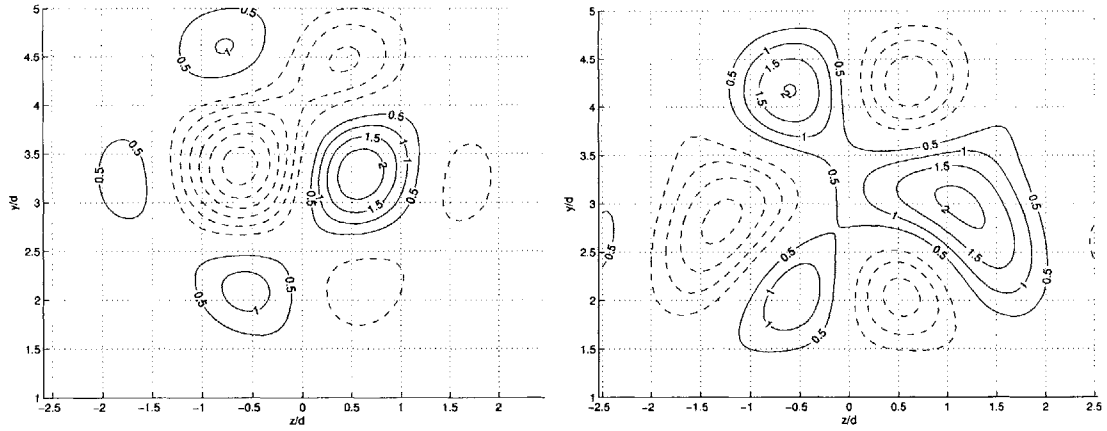
(a) $x/d = 2.25$

(b) $x/d = 2.50$



(c) $x/d = 2.75$

Figure 3-25: Transverse slices of filtered streamwise vorticity at $\tilde{t} = 2.40$ in the $r = 7$ jet. Positive values are indicated by the labeled solid contours; dashed contours are the negative counterparts.



(a) $x/d = 2.25$ (b) $x/d = 2.50$

Figure 3-26: Transverse slices of filtered streamwise vorticity at $\tilde{t} = 3.00$ in the $r = 5$ jet. Positive values are indicated by the labeled solid contours; dashed contours are the negative counterparts.

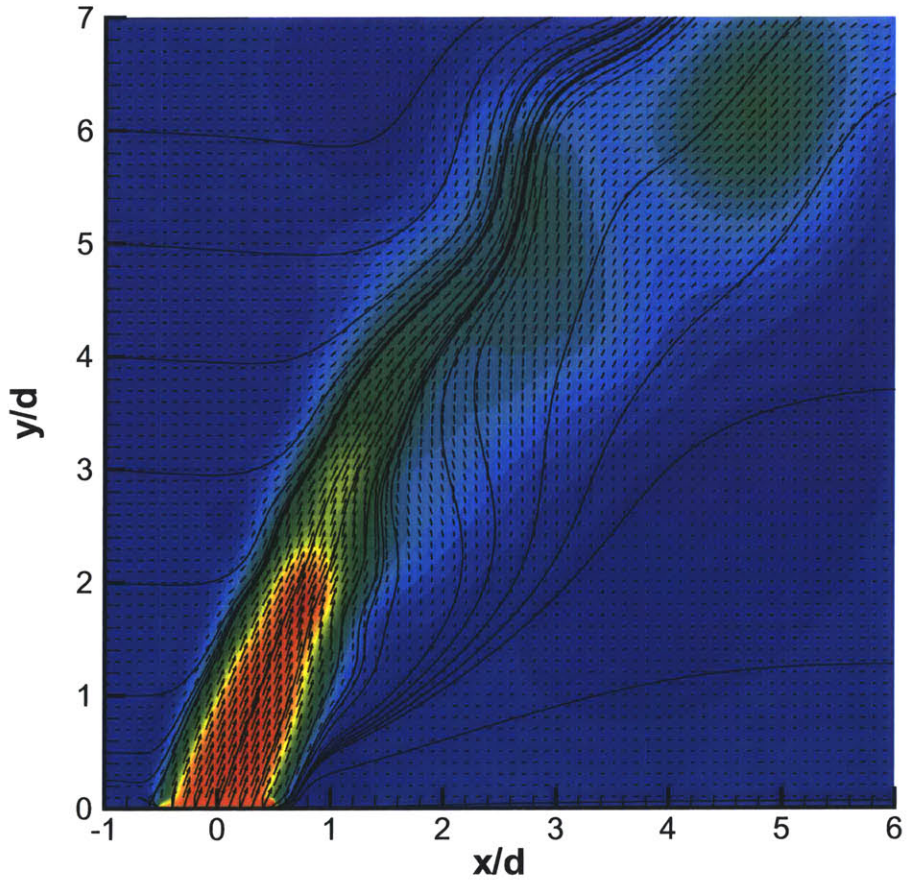


Figure 3-27: Velocity streamlines in the centerplane $z = 0$ at $\tilde{t} = 3.20$, introducing only jet azimuthal vorticity. Contours indicate total velocity magnitude.

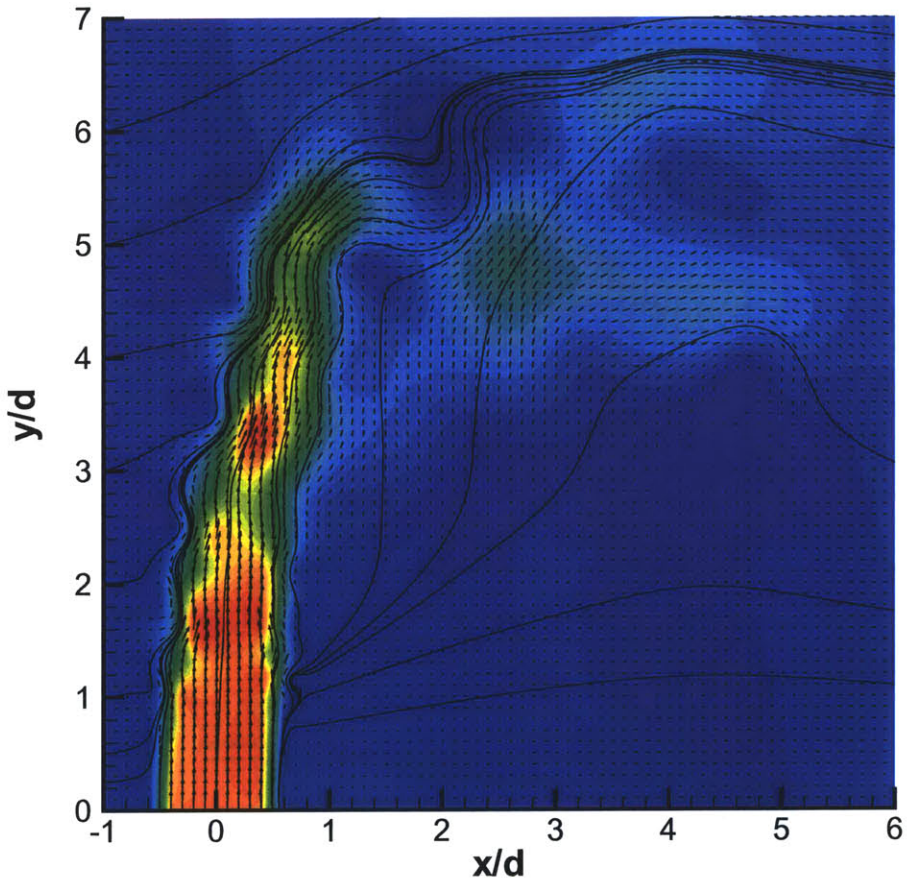


Figure 3-28: Velocity streamlines in the centerplane $z = 0$, at $\tilde{t} = 3.20$, using the vorticity-flux boundary condition in (2.27). Contours indicate total velocity magnitude.

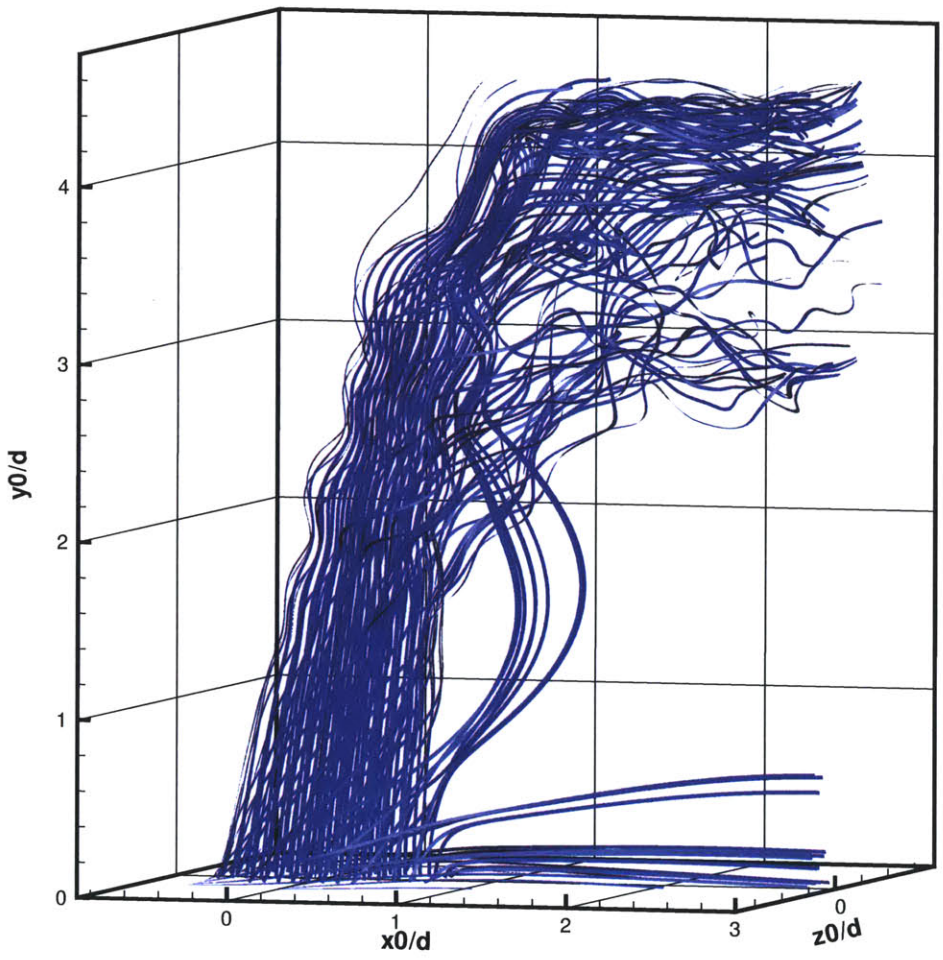


Figure 3-29: Three-dimensional velocity streamlines in the $r = 7$ jet at $\tilde{t} = 2.00$; spanwise view.

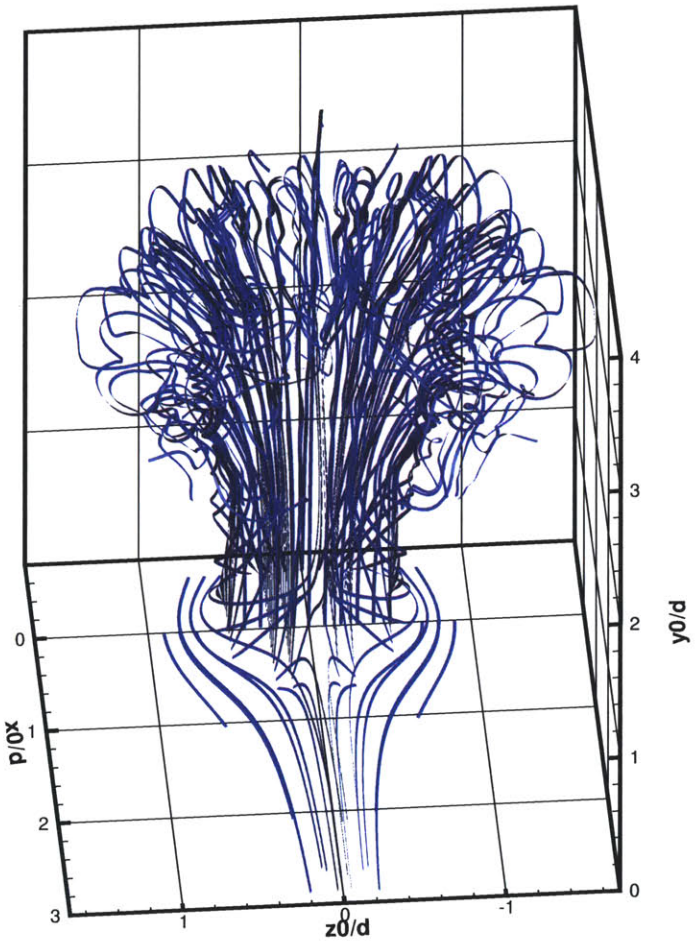


Figure 3-30: Three-dimensional velocity streamlines in the $r = 7$ jet at $\tilde{t} = 2.00$; view from downstream.

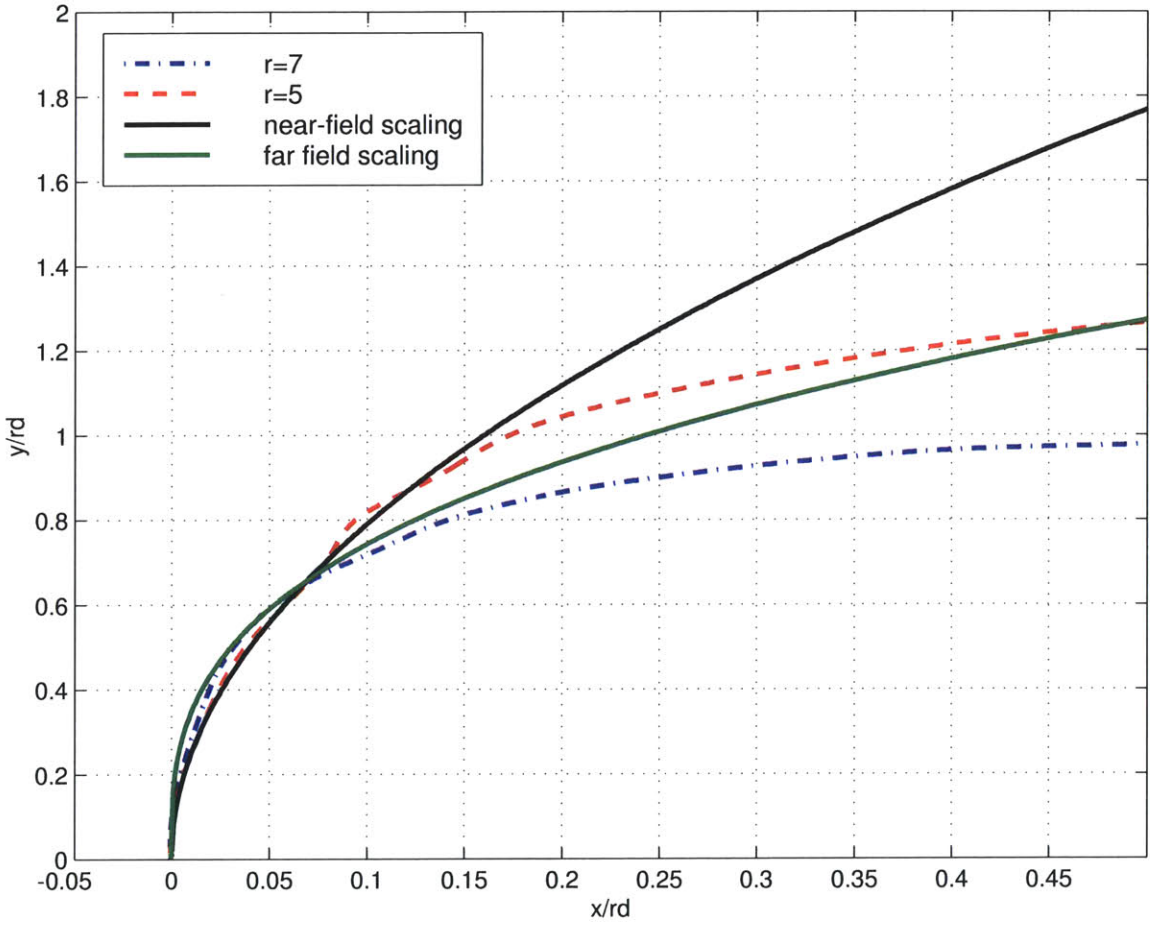
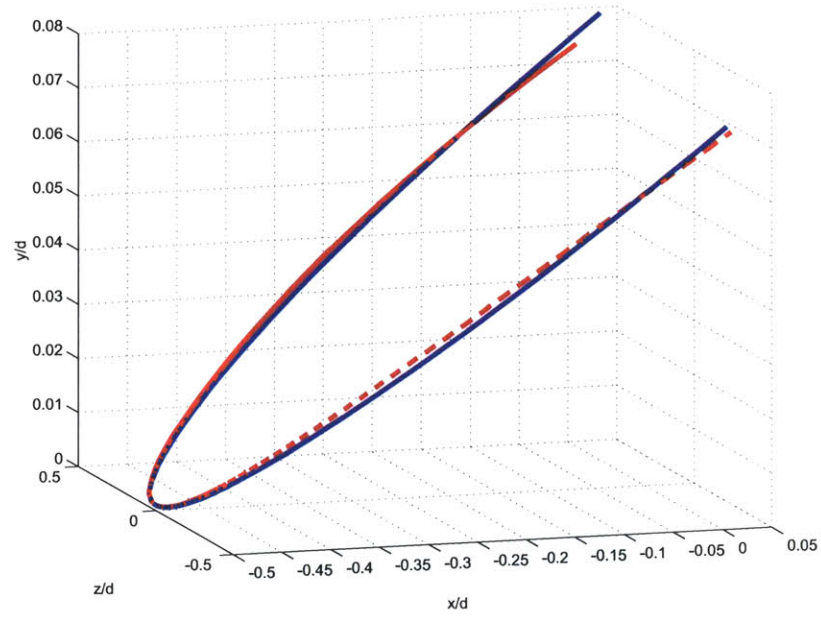
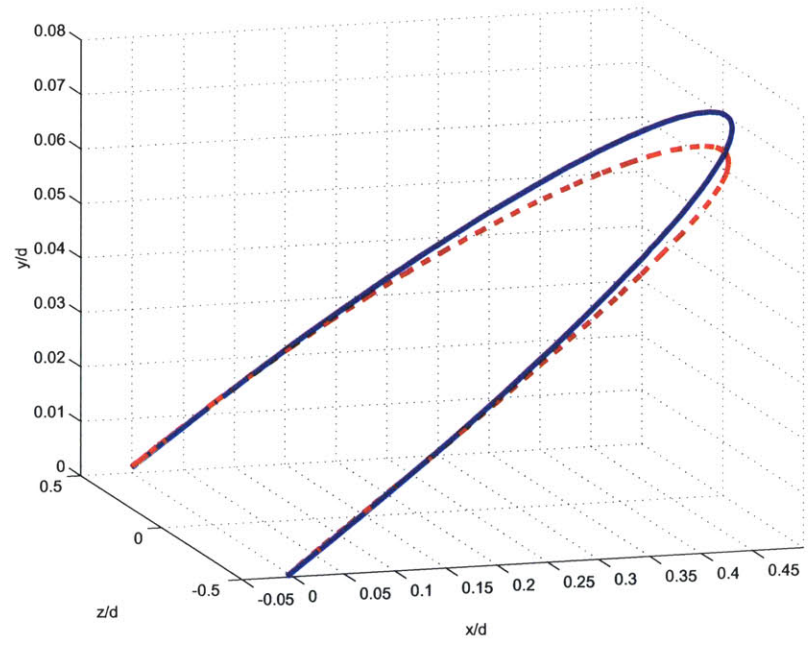


Figure 3-31: Instantaneous jet-center streamlines for an $r = 5$ jet at $\tilde{t} = 4.7$ and an $r = 7$ jet at $\tilde{t} = 3.1$, compared to trajectories obtained from similarity analysis.

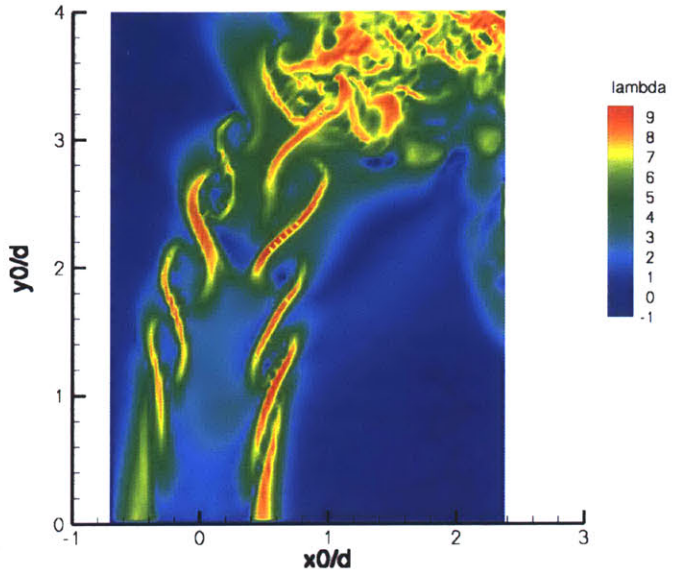


(a) Back half of jet orifice.

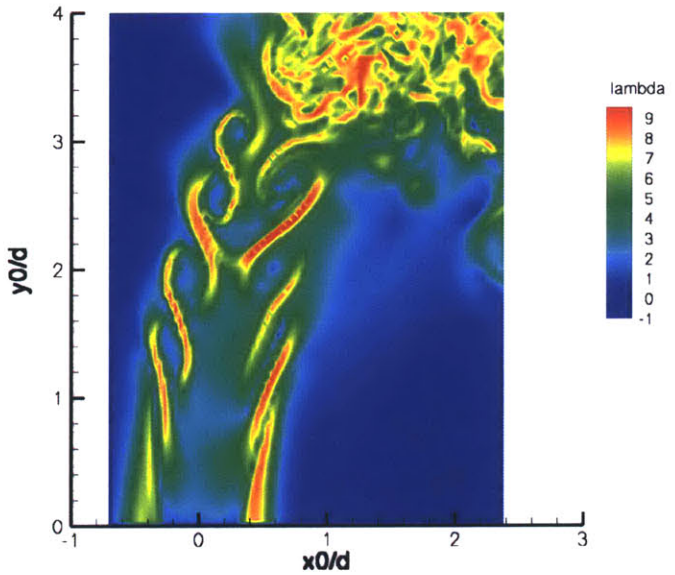


(b) Front half of jet orifice.

Figure 3-32: Comparison of analytical and numerical vortex lines in the near field of the transverse jet, $r = 7$. Solid lines are integral curves of the numerical vorticity field at $\tilde{t} = 1.4$; dashed lines are computed from the analytical expression derived in §2.2.3.

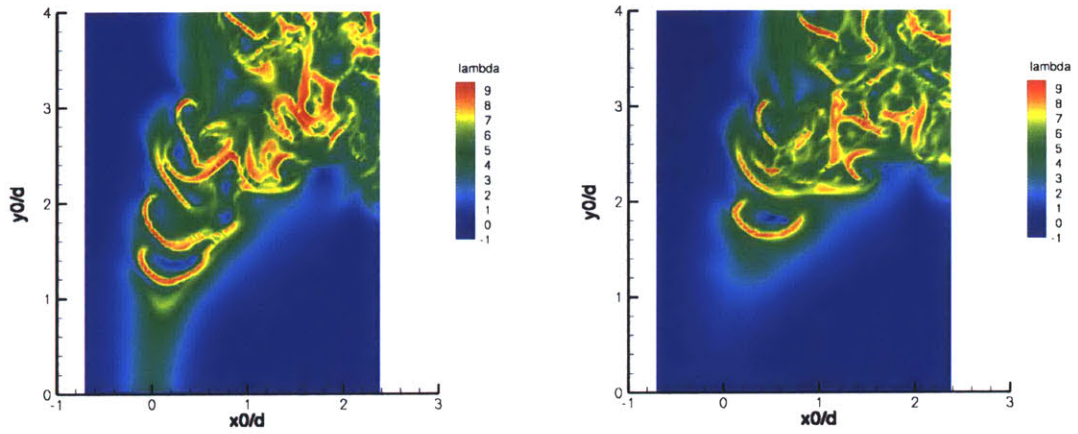


(a) $z/d = -0.02$



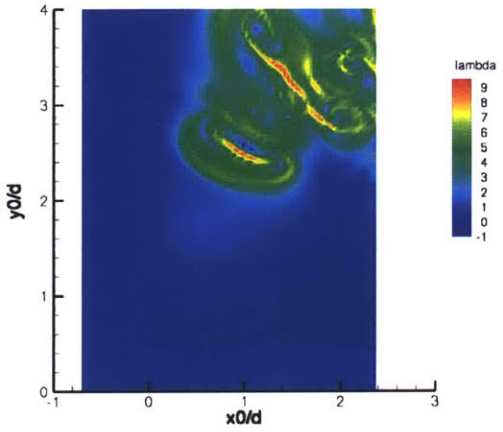
(b) $z/d = -0.22$

Figure 3-33: Contours of maximal direct finite-time Lypapunov exponent $\Lambda(\mathbf{x}_0)$ on spanwise planes of the $r = 7$ jet. Values are computed in forward time over the interval $[\tilde{t}_0, \tilde{t}] = [2.00, 2.30]$.



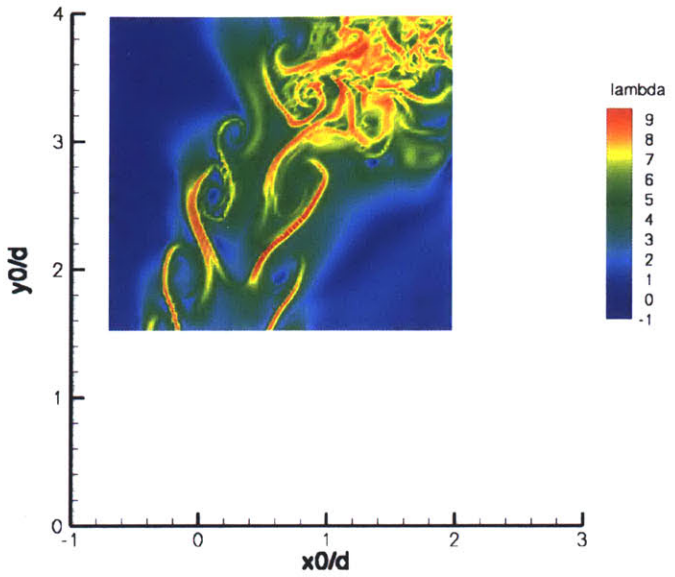
(c) $z/d = -0.62$

(d) $z/d = -0.82$

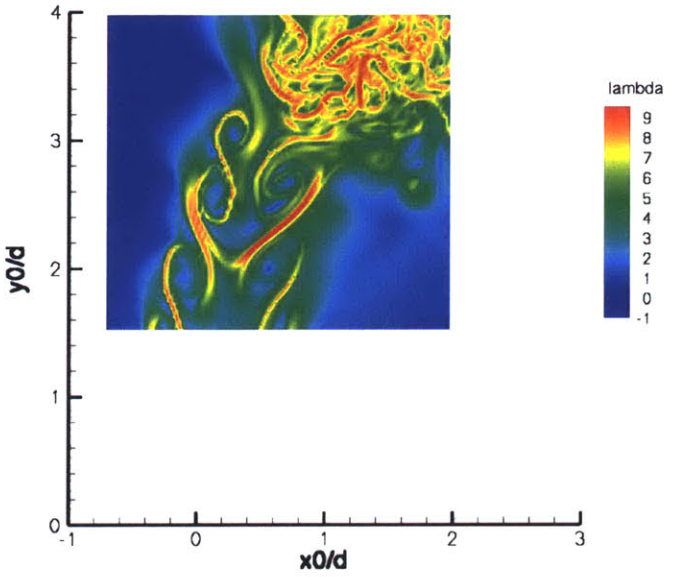


(e) $z/d = -1.20$

Figure 3-33: Contours of maximal direct finite-time Lypapunov exponent $\Lambda(\mathbf{x}_0)$ on spanwise planes of the $r = 7$ jet. Values are computed in forward time over the interval $[\tilde{t}_0, \tilde{t}] = [2.00, 2.30]$ (con't).

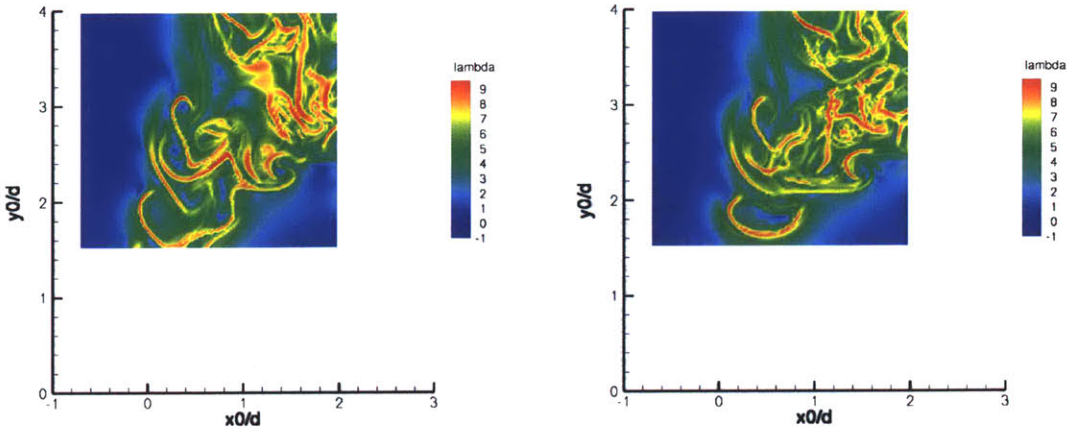


(a) $z/d = -0.025$



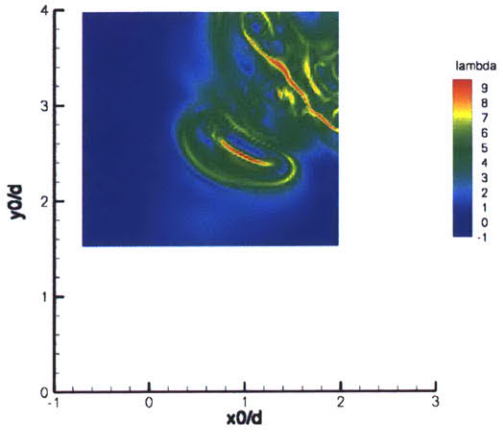
(b) $z/d = -0.225$

Figure 3-34: Higher-resolution contours of maximal direct finite-time Lypapunov exponent $\Lambda(\mathbf{x}_0)$ on spanwise planes of the $r = 7$ jet. Values are computed in forward time over the interval $[\tilde{t}_0, \tilde{t}] = [2.00, 2.35]$.



(c) $z/d = -0.600$

(d) $z/d = -0.800$



(e) $z/d = -1.200$

Figure 3-34: Higher-resolution contours of maximal direct finite-time Lypapunov exponent $\Lambda(\mathbf{x}_0)$ on spanwise planes of the $r = 7$ jet. Values are computed in forward time over the interval $[\tilde{t}_0, \tilde{t}] = [2.00, 2.35]$ (con't).

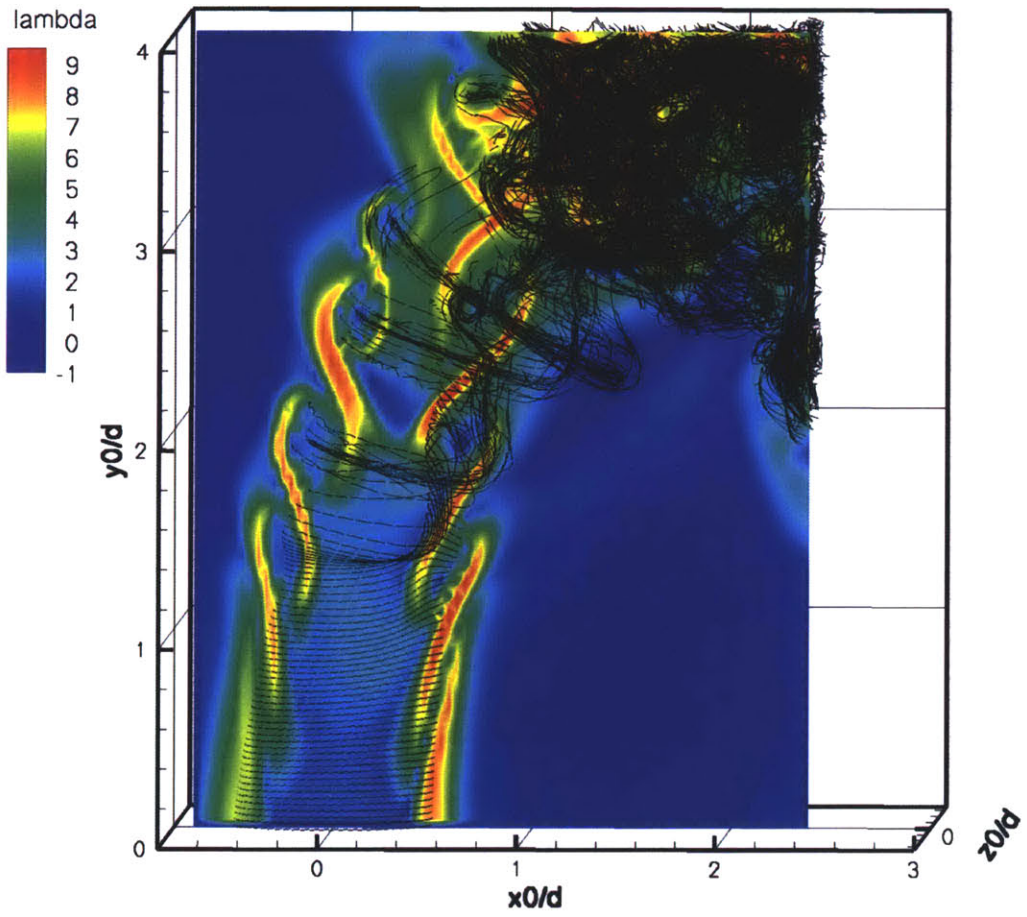
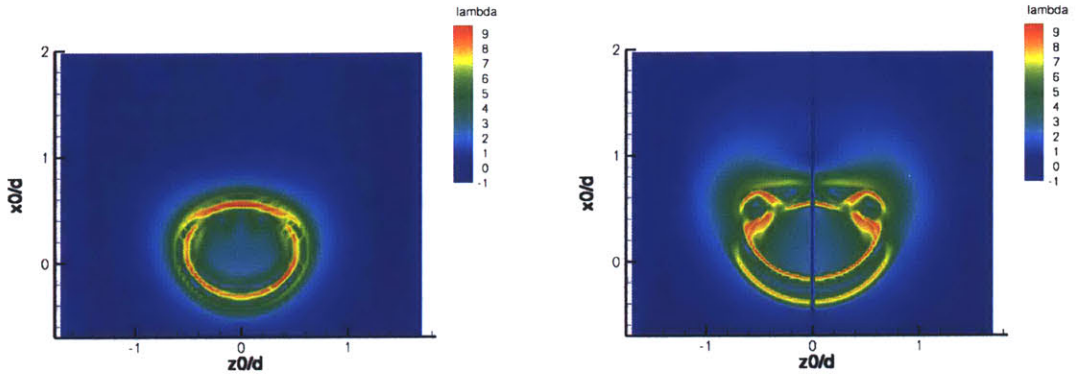


Figure 3-35: DLE contours at $z/d = -0.025$ on the interval $[2.00, 2.30]$; superimposed on the contours is a perspective view of vortex filaments at $\tilde{t} = 2.00$; $r = 7$.



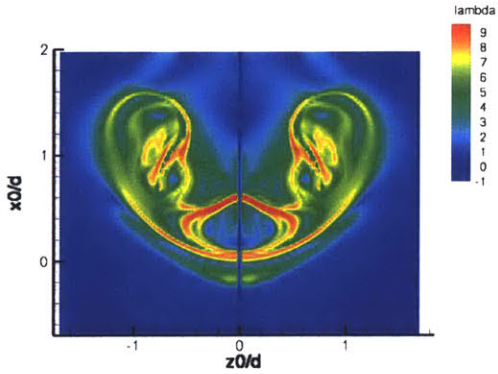
(a) $y/d = 1.000$

(b) $y/d = 1.525$

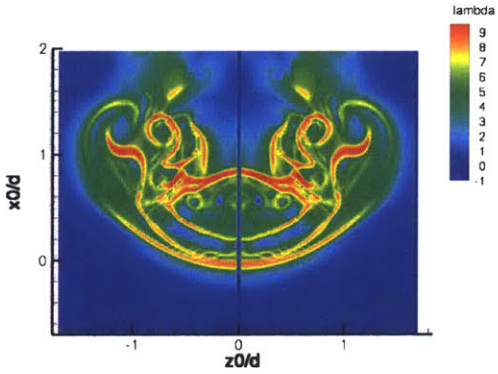
(c) $y/d = 1.825$

(d) $y/d = 2.025$

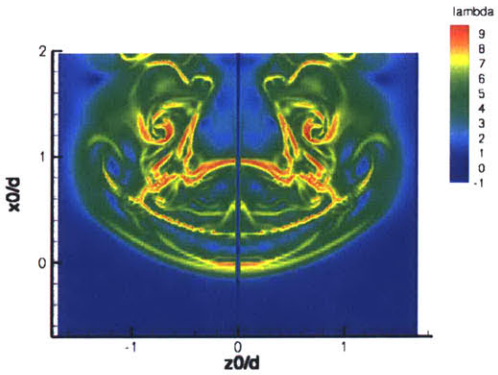
Figure 3-36: Contours of maximal direct finite-time Lypapunov exponent $\Lambda(\mathbf{x}_0)$ on axial planes of the $r = 7$ jet. Values are computed in forward time over the interval $[\tilde{t}_0, \tilde{t}] = [2.00, 2.35]$.



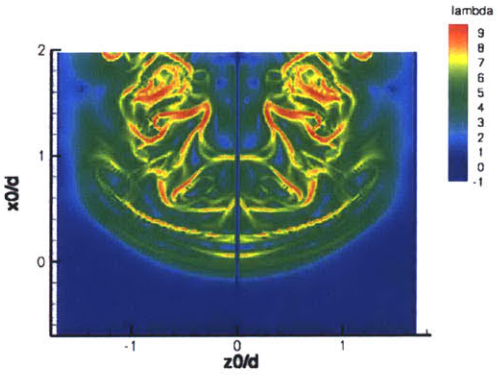
(e) $y/d = 2.200$



(f) $y/d = 2.425$

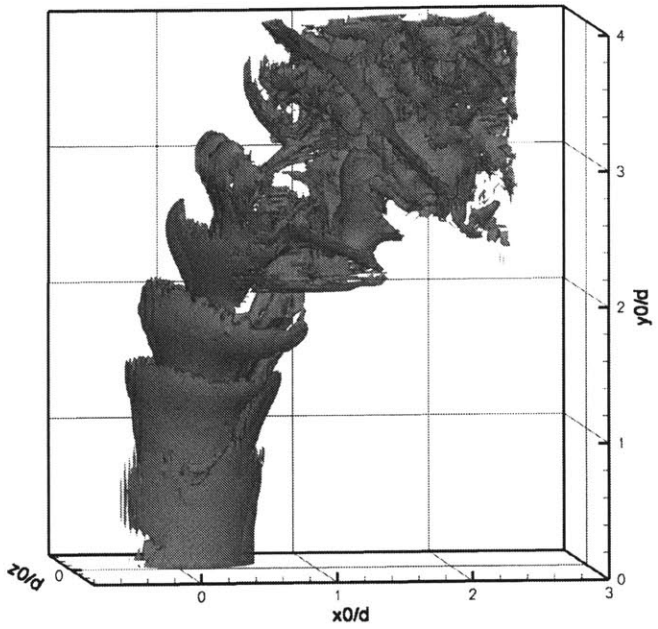


(g) $y/d = 2.625$

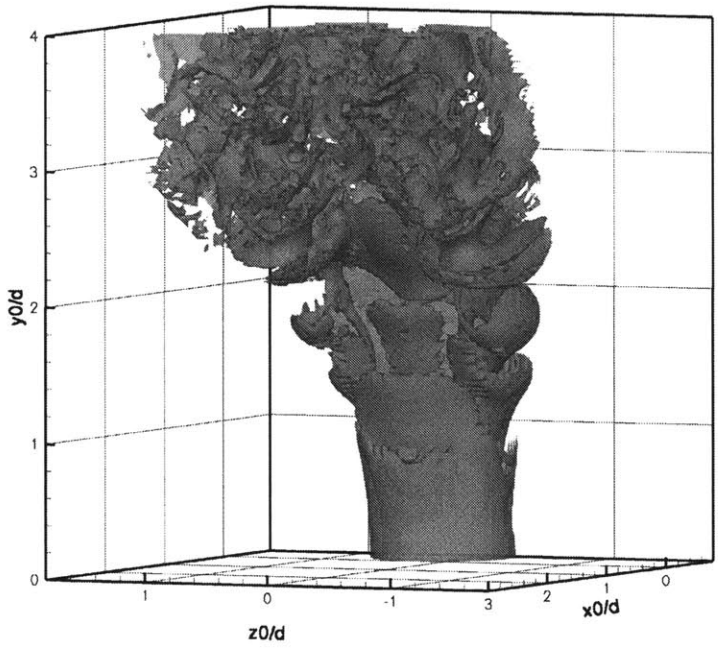


(h) $y/d = 2.750$

Figure 3-36: Contours of maximal direct finite-time Lypapunov exponent $\Lambda(\mathbf{x}_0)$ on axial planes of the $r = 7$ jet. Values are computed in forward time over the interval $[\tilde{t}_0, \tilde{t}] = [2.00, 2.35]$ (con't).



(a) Side view.



(b) Downstream view.

Figure 3-37: DLE isosurfaces $\Lambda = 7.2$ on the interval $[2.00, 2.30]$, plotted at $\tilde{t}_0 = 2.00$; $r = 7$.

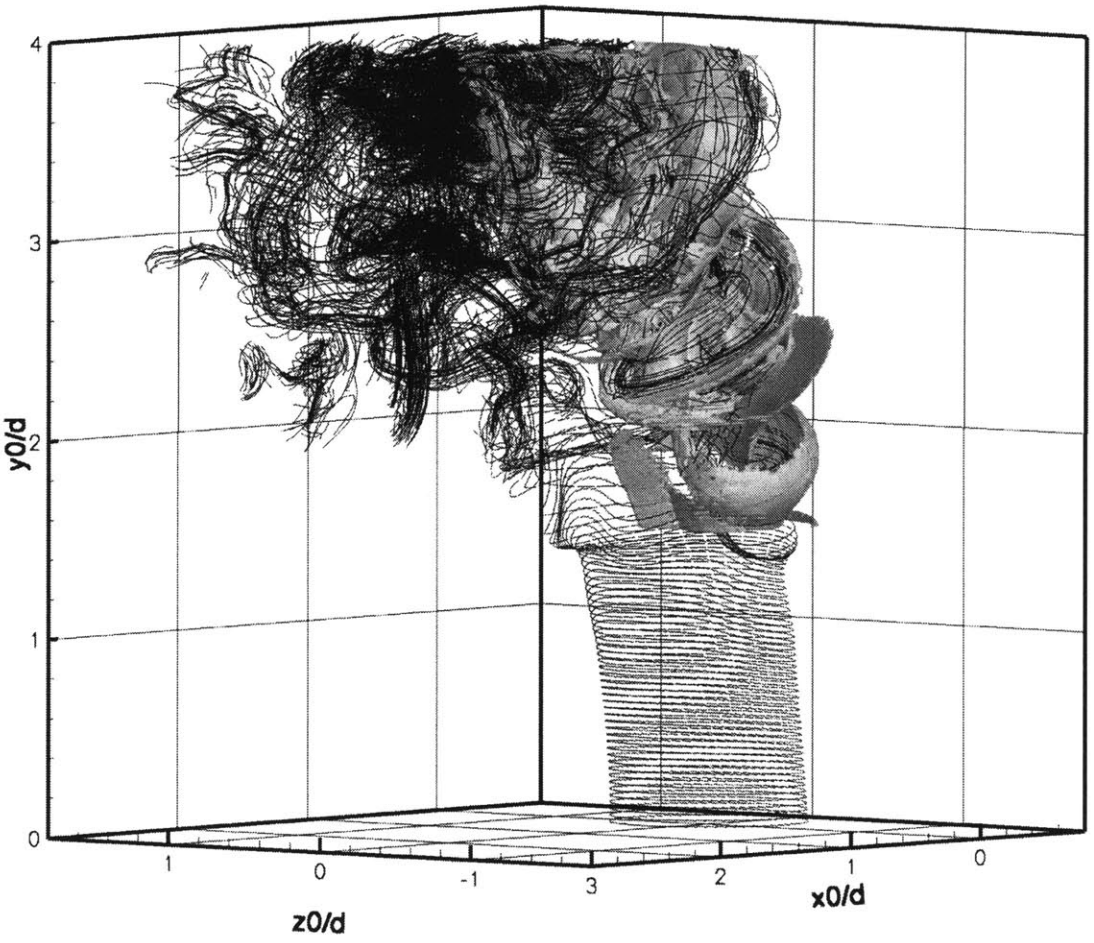
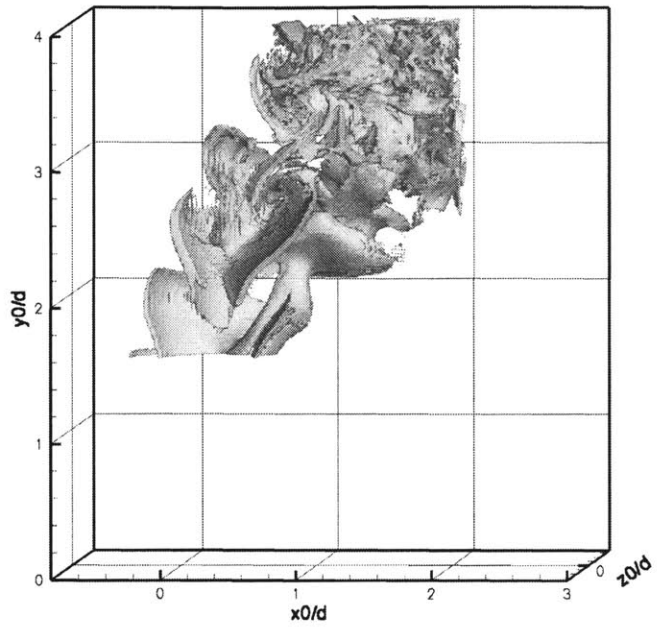
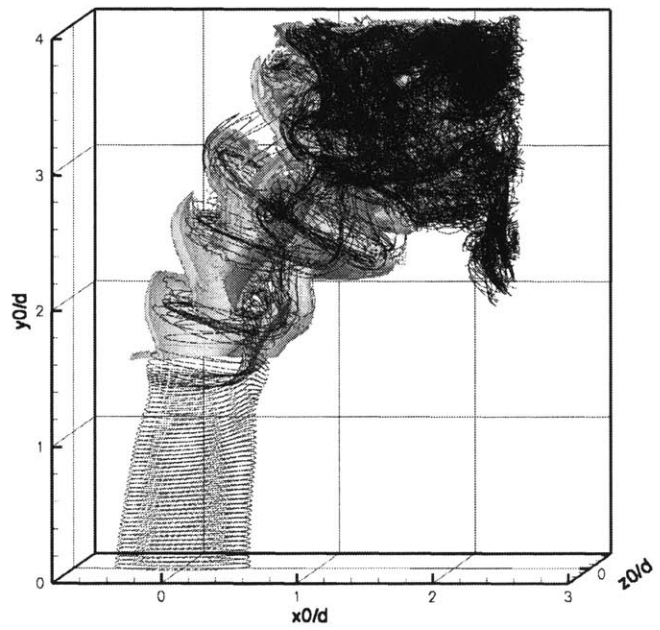


Figure 3-38: DLE isosurfaces $\Lambda = 7.2$ on the interval $[2.00, 2.35]$, plotted at $\tilde{t}_0 = 2.00$; lee-side view, $r = 7$.



(a) Isosurface only.



(b) Isosurface and vortex filaments at $\tilde{t} = 2.00$.

Figure 3-39: DLE isosurfaces $\Lambda = 7.2$ on the interval $[2.00, 2.35]$, plotted at $\tilde{t}_0 = 2.00$; side view, $r = 7$.

Chapter 4

K-means Clustering for Dynamic Partitioning of Hierarchical N-body Interactions

A number of complex physical problems can be approached through N-body simulation. High-Reynolds number flows, computed with vortex methods as detailed in the preceding chapters, are but one example. Other important problems range from gravitational astrophysics and cosmology [39] to smoothed particle hydrodynamics, molecular dynamics, non-Newtonian flows [130], and electrodynamics [49].

In all these applications, a dense system of pairwise particle interactions leads to a computational cost of $O(N^2)$, which is prohibitive for large N. Fast summation algorithms that reduce this cost to $O(N \log N)$ or $O(N)$ are necessary to achieve resolution and scale. Typically these methods must contend with irregular particle distributions of non-uniform density; in dynamic N-body problems, the algorithms also face a particle distribution that evolves in time. Large, realistic physical problems require efficient implementation of these algorithms on massively parallel distributed memory computer architectures.

The present work employs hierarchical methods for fast summation. Hierarchical methods construct approximations for the influence of a cluster of particles and, where possible, use these approximations to replace pairwise particle interactions

with a smaller number of particle-cell or cell-cell interactions. Based on the latter, these methods may be classifed into treecodes (particle-cell interactions) [6, 8] and fast multipole methods (cell-cell interactions) [50]. The focus here is on treecodes; a more complete background on hierarchical methods is provided in §4.1.2.

The "quality" of spatial partitioning is central to the performance of a hierarchical method. The spatial partition determines cell moments and cell proximities (including neighbor relationships), and thus controls the number and order of particle-cell interactions necessary to achieve a given level of accuracy. For an efficient parallel implementation, one also must devise a spatial partition that is compatible with distributing hierarchical interactions over many processors.

In the following, we introduce new algorithms, based on k-means clustering, for partitioning parallel hierarchical N-body interactions. The advantages of cluster partitions stem from their *geometric* properties. K-means partitions optimize cluster moments and other quantities that control the error bounds of a treecode, and thus reduce the computational cost of N-body approximations. The clustering procedure is inherently adaptive—an important feature for non-uniform distributions of particle positions and weights—and itself may be parallelized efficiently. All these features are preserved as the number of processors is scaled. Alternative algorithms for spatial partitioning of parallel treecodes—namely orthogonal recursive bisection (ORB) [123, 39] or the hashed-oct-tree (HOT) algorithm [122]—do not yield similar geometric properties.

We demonstrate the parallel performance of clustering by constructing a parallel treecode for vortex particle simulations, based on the serial variable-order treecode developed by Lindsay and Krasny [77]. For simplicity, we do not focus on distributed data and the communications algorithms required to fetch non-local cell data efficiently. On a modern computer, locations, weights, and cell moments for up to 10^7 particles will fit on one processor's memory, so this problem becomes less important. We also note that the spherical domain geometries favored by clustering minimize surface area-to-volume ratios often associated with communications overhead [14], and thus may be advantageous to any distributed data implementations we develop

in future work.

This chapter also presents new heuristics for dynamically load balancing cluster partitions. These techniques include dynamic scaling of cluster metrics and adaptive redistribution of cluster centroids. Load balance is always an issue in N-body problems with non-uniform particle distributions, but a unique impediment to load balance in the present context is the continual introduction of *new* vortex elements. As detailed in §2.1.4 and Chapter 3, new element introduction is crucial to simulation of turbulent flow: resolving the stretching of vortical structures and the resulting breakdown of the flow into small scales requires a continual remeshing of vortex filaments. We demonstrate the performance of load-balanced clustering on full three-dimensional simulations of the transverse jet, identical to those considered in Chapter 3.

4.1 Background

We begin by reviewing the fundamentals of vortex particle methods for fluid dynamics, illustrating how the formulation gives rise to a classical N-body problem. We then discuss hierarchical solvers that have been developed for efficient solution of such N-body problems in serial code.

4.1.1 Vortex methods and N-body problems

Vortex methods are a computational approach to systems governed by the Navier-Stokes or Euler equations, employing a particle discretization of the vorticity field and transporting vorticity along particle trajectories [75, 72, 100, 79, 32]. The reader is referred to Chapter 2, §2.1, for a complete exposition of vortex methods. Here, we simply highlight the connection between vortex methods and more general N-body problems.

Given a regularized particle discretization of the vorticity field:

$$\boldsymbol{\omega}(\mathbf{x},t) \approx \sum_{i}^{N} \boldsymbol{\alpha}_i(t) f_\delta\left(\mathbf{x} - \boldsymbol{\chi}_i\left(t\right)\right) \tag{4.1}$$

we can write the vortical velocity $\mathbf{u}_\omega$ at any point $\mathbf{x}$ as follows:

$$\mathbf{u}_\omega(\mathbf{x}) = \sum_i^N \mathbf{K}_\delta\left(\mathbf{x}, \boldsymbol{\chi}_i\right) \boldsymbol{\alpha}_i \tag{4.2}$$

where $\mathbf{K}_\delta$ is the regularized Biot-Savart kernel. Vortex particles here have vector-valued weights $\boldsymbol{\alpha}_i(t)$ and positions $\boldsymbol{\chi}_i(t)$.

Vortex methods solve the inviscid equations of motion via numerical integration for the particle trajectories $\boldsymbol{\chi}_i(t)$ and weights $\boldsymbol{\alpha}_i(t)$. Computing particle trajectories $\boldsymbol{\chi}_i(t)$ requires evaluation of the velocity at each particle at every timestep. As each vortex element induces a velocity on every other vortex element, this is an N-body problem; direct evaluation of (4.2) at every element yields a computational cost of $O(N^2)$. For large numbers of particles, this clearly can be prohibitively expensive.

The $O(N^2)$ bottleneck is not unique to vortex methods; indeed, it is a feature inherent to N-body problems in a variety of contexts, whether the result of summation or quadrature (as in equations 2.5 or 2.8) is a velocity, a force, or a potential. Gravitational N-body simulations are an essential tool in astrophysics, where they are used to study galaxy dynamics and cosmological structure formation [104, 123]. Here, as in vortex methods, large N is essential to resolve fine features and the necessary large scales [39]. N-body problems are also encountered in smoothed particle hydrodynamics [46, 61] and plasma physics. Coulomb potentials and other, more complicated short-range potentials give rise to N-body problems in molecular dynamics [37, 38], with increasingly important biological applications [16, 107]. Overcoming the $O(N^2)$ bottleneck is thus essential to progress across a variety of scientific fronts.

4.1.2 Hierarchical methods

Hierarchical methods for N-body problems construct approximations for the influence of a cluster of particles and, where possible, use these approximations to replace particle-particle interactions with a smaller number of particle-cluster or cluster-cluster interactions. The construction of these approximations is typically organized by a recursive tree structure. Treecodes, introduced for gravitational problems by

Appel [6] and Barnes-Hut [8], organize a group of N particles into a hierarchy of nested cells, e.g., an oct-tree in three dimensions. At subsequent levels of the tree, each "parent" cell is divided into smaller "child" cells representing finer spatial scales.

Treecodes have found wide application in particle methods. The original Barnes-Hut (BH) algorithm employs an oct-tree with a monopole moment calculated at each cell. Tree construction proceeds until leaf nodes each contain only a single particle. The tree is traversed once for every particle using a divide-and-conquer strategy of particle-cell interactions; if the monopole approximation at a given cell cannot provide the force on the target particle to a sufficient level of accuracy, the contribution of the cell is replaced by the contribution of its child cells. The total computational cost scales as $O(N \log N)$. Variations on this treecode algorithm have been numerous; broadly speaking, these differ in terms of physics—i.e., the kernel describing the influence of each particle [122, 38, 16, 49]—and in the *type* and *order* of series approximation used to describe the influence of a cluster [120, 2, 77]. Other variations encompass adaptive features of the tree construction [9] and more sophisticated error estimates [104, 48].

Fast multipole methods (FMM), introduced by Greengard and Rokhlin [50, 51], employ additional analytical machinery to translate the centers of multipole expansions and to convert far-field multipole expansions into local expansions, reducing the total operation count to $O(N)$. Like many BH-type codes, these codes also use higher-order approximations, typically a multipole expansion involving spherical harmonics in three dimensions [116, 95], although other schemes have been proposed [2, 21, 16].

Lindsay and Krasny have introduced a BH-type treecode with many adaptive features well-suited to vortex particle methods [77]. This serial code provided a convenient platform on which to develop and test the clustering and load-balancing algorithms described in this paper, so we will review its essential features. The Lindsay-Krasny (LK) code organizes particles in an oct-tree. Adaptive features of the tree include nonuniform rectangular cells that shrink to fit their contents at every level of the tree and a leaf size parameter N_0 below which a cell is not divided.

The velocity induced by each particle is given by the Rosenhead-Moore kernel, a regularized form of the Biot-Savart kernel; in vortex methods, this regularization is also known as the low-order algebraic smoothing [128].

$$\mathbf{K}_\delta(\mathbf{x},\mathbf{x}') = -\frac{1}{4\pi}\frac{\mathbf{x}-\mathbf{x}'}{(|\mathbf{x}-\mathbf{x}'|^2+\delta^2)^{3/2}}\times \tag{4.3}$$

Because this kernel is not harmonic, it cannot be expanded in a classical multipole series; instead the treecode employs a Taylor expansion in Cartesian coordinates to approximate the influence of each cell at a target point $\mathbf{x}$:

$$\mathbf{u}(\mathbf{x}) \approx \sum_{i=1}^{N_c}\sum_{\mathbf{k}}^{\|\mathbf{k}\|<p} \frac{1}{\mathbf{k}!}\mathbf{D}_\mathbf{y}^\mathbf{k}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c)(\mathbf{y}_i-\mathbf{y}_c)^\mathbf{k}\times\boldsymbol{\alpha}_i \tag{4.4}$$

$$\approx \sum_{\mathbf{k}}^{\|\mathbf{k}\|<p} \mathbf{a}_\mathbf{k}(\mathbf{x},\mathbf{y}_c)\times\mathbf{m}_\mathbf{k}(c) \tag{4.5}$$

where $\mathbf{y}_c$ is the coordinate of the cell centroid, N_c is the number of particles in the cell, $\mathbf{y}_i$ are their coordinates, $\mathbf{k}=(k_1,k_2,k_3)$ is an integer multi-index with all $k_i \geq 0$,

$$\mathbf{a}_\mathbf{k}(\mathbf{x},\mathbf{y}_c) = \frac{1}{\mathbf{k}!}\mathbf{D}_\mathbf{y}^\mathbf{k}\mathbf{K}_\delta(\mathbf{x},\mathbf{y}_c) \tag{4.6}$$

is the $\mathbf{k}$th Taylor coefficient of the Rosenhead-Moore kernel at $\mathbf{y}=\mathbf{y}_c$ and

$$\mathbf{m}_\mathbf{k}(c) = \sum_{i=1}^{N_c}(\mathbf{y}_i-\mathbf{y}_c)^\mathbf{k}\boldsymbol{\alpha}_i \tag{4.7}$$

is the $\mathbf{k}$th moment of cell c about its center. Taylor expansions are computed to variable order p up to a user-specified maximum order of approximation p_{max}; a typical choice is $p_{max}=8$. A recurrence relation for the Taylor coefficients $\mathbf{a}_\mathbf{k}$ allows them to be computed cheaply (each coefficient in $O(1)$ operations) for each particle-cluster interaction. Cell moments, on the other hand, are computed as needed for each cell then stored for use in subsequent interactions. Unlike other vortex particle treecodes, the LK treecode incorporates the regularization of the kernel directly into the expansion [129, 105].

The velocity at each target particle is evaluated using a "divide-and-conquer" strategy governed by a user-specified accuracy parameter ϵ and an error estimate derived from the approximation error for the vector potential:

$$\epsilon \geq \frac{M_p(c)}{4\pi R^{p+1}} \tag{4.8}$$

where

$$M_p(c) = \sum_{i}^{N_c} |\mathbf{y}_i - \mathbf{y}_c|^p |\boldsymbol{\alpha}_i| \tag{4.9}$$

is the pth absolute moment of cell c and $R = (|\mathbf{x} - \mathbf{y}_c|^2 + \delta^2)^{1/2}$ is the regularized distance between the target particle and the cell center. Velocity evaluation for each target particle begins at the root cell and proceeds recursively. For each cell c encountered, the code computes the minimum order of approximation p that satisfies inequality (4.8). If this $p < p_{max}$, the particle-cell interaction is evaluated for the cell c; otherwise the velocity evaluation descends the hierarchy and sums the velocities induced by the children of cell c. This procedure is modified by a run-time choice between Taylor expansion and direct evaluation at each cell, which may become active at lower levels of the hierarchy; if the estimated time for direct summation is smaller than the estimated time for Taylor expansion, the former method is used to compute the influence of the cell.

Some of our ongoing work [125] develops recurrence relations for other regularizations of the Biot-Savart kernel, including the higher-order algebraic smoothing proposed in [128].

4.2 Computational approach

Efficient parallelization of an algorithm depends on avoiding the duplication of work among processors, ensuring equal workload at each processor, and minimizing additional costs, such as the time for inter-processor communication and domain decomposition. In the case of treecodes for particle methods, a good spatial decomposition is essential to all of the goals just mentioned.

4.2.1 Clustering

We propose a new approach for parallel domain decomposition of vortex particles, based on *k-means clustering* of the particle coordinates. Clustering procedures are essential tools for multivariate statistical analysis, data mining, and unsupervised machine learning [1, 40]; k-means clustering [78] is a classical algorithm in these contexts. In the new context of domain decomposition, however, we develop a variant of the k-means algorithm yielding a partition with many desirable properties.

K-means takes a set of N observations $\{\mathbf{x}_i\}$ in d-dimensional space as input and partitions the set into k clusters with centroids $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$, where k is prescribed. The partition is chosen to minimize the cost function

$$J = \sum_{i=1}^{N} \min_{k'} \left(|\mathbf{x}_i - \mathbf{y}_{k'}|^2 |\boldsymbol{\alpha}_i| \right) = \sum_{j=1}^{k} \sum_{i}^{N_j} |\mathbf{x}_i - \mathbf{y}_j|^2 |\boldsymbol{\alpha}_i| \tag{4.10}$$

In other words, each observation is assigned to the nearest centroid, and the centroid positions are chosen to minimize the within-cluster sum of squared Euclidean distances. In our implementation, for reasons that will be made clear below, we weigh each particle's contribution by its vorticity magnitude $|\boldsymbol{\alpha}_i|$. K-means results in a *flat* or non-hierarchical clustering, in contrast to other clustering algorithms that construct hierarchical partitions, either from the bottom up (agglomerative) or the top down (divisive) [40].

The k-means algorithm can be viewed as an iterative optimization procedure for the cost function defined in (4.10), beginning with a choice of centroids $\{\mathbf{y}_{i=1\ldots k}\}$ and iteratively updating them to reduce J. K-means will find a local minimum of J, and thus the solution may depend on the initial choice of centroids; the problem of finding a global minimum is in fact NP-complete. We implement a "batch" version of the k-means algorithm, in which all the particles are assigned to centroids before the centroids are updated, at each iteration. This is in contrast to the "online" approach, in which the centroid locations are updated as each particle is individually classified [17]. In either case, the resulting classification boundaries are the Voronoi tesselation

of the cluster centroids, and thus they bound convex subsets of $\mathbb{R}^d$.

An outline of the algorithm, using variable particle weights, is as follows:

Algorithm 4.1 (K-means Clustering).

initialize N, k, $\mathbf{y}_1, \ldots, \mathbf{y}_k$

do $l = 1$ *to* l_{max}

assign each particle $\mathbf{x}_i$ *to cluster* $k_i^* = \underset{k'}{\operatorname{argmin}} \left(|\mathbf{x}_i - \mathbf{y}_{k'}|^2\right)$

put each $\mathbf{y}_k = \dfrac{\sum_i^{N_k} |\boldsymbol{\alpha}_i| \mathbf{x}_i}{\sum_i^{N_k} |\boldsymbol{\alpha}_i|}$

recompute $J^{(l)}$

until $J^{(l-1)} - J^{(l)}$ *small*

return *centroids* $\mathbf{y}_1, \ldots, \mathbf{y}_k$ *and memberships* $\{k_i^*\}_{i=1\ldots N}$

The computational complexity of this algorithm is $O(NkdT)$, where T is the number of iterations. It is straightforward to implement k-means clustering in parallel, however, and we do so using the parallel implementation proposed by Dhillon and Modha [36]. Since we typically seek a number of clusters k equal to the number of processors, an ideal parallelization reduces the computational complexity to $O(NdT)$. This scaling is what find in practice, as will be shown in the Results section. Recent work has demonstrated new algorithms for fundamentally accelerating k-means clustering, using kd-trees to reduce the number of nearest-neighbor queries, for computational times sublinear in N [93]. We do not pursue these approaches here, but note that they may become useful in ensuring that the time for parallel domain decomposition ($O(N)$ with parallel k-means) remains a small fraction of the time required for velocity evaluation ($O(N/k \log N)$ in the ideal case) for very large k.

It is worthwhile to note that the partition of vortex elements resulting from the clustering procedure may bear no relation to the data structure elsewhere used to represent the vortex particles in memory. This is particularly relevant to vortex filament methods, in which an ordering, or connectivity, between neighboring elements must be preserved. In this case, it may be necessary to maintain separate data structures or attribute lists, one appropriately representing filament connectivity, and

another encoding the flat k-means partition, which considers the vortex elements as a set of completely independent particles.

4.2.2 Towards optimal geometry of clusters

Clusters resulting from k-means procedure are certainly well-localized, in the sense of cost function (4.10). But it is important to consider why data locality is important to hierarchical N-body solvers, and precisely why poor data locality may negatively impact performance.

In the following discussion we make a distinction between "source" particles and "target" particles. Targets are the points at which the velocity or force is computed; sources are the particles, or quadrature points, inducing the velocity or force. In most situations—e.g., a vortex blob code or a gravitational N-body simulation—the source sets and target sets are exactly the same, and each particle simply takes turns in either role. For simplicity, in the following discussion we let the set of source particles be identical to the set of target particles.[1]

A given domain decomposition method admits many schemes for parallelizing the treecode calculations. Consider first the possibility of using a global tree. This process may be driven by parallelizing over targets. By this we mean that the domain decomposition scheme yields a certain partition of particles and that each processor is responsible for computing the influence of the entire domain on the particles assigned to it by partition. A global tree is thus constructed on each processor, but only in structure: Cells subdividing the root cell are included only if they will be requested during tree traversal for velocity evaluation. In the LK treecode, cell-moment calculations make up the majority of tree construction cost. Because velocity is evaluated only for the assigned particles, the resulting set of tree cells with filled-in moments (to varying order) is in fact the *locally essential tree* [123, 15]. Each locally essential tree *is* a global tree, describing the influence of the entire domain on the assigned

[1] In a vortex filament code, the two sets may differ slightly depending on the quadrature rule used along the filament coordinate; the targets (filament nodes) may be staggered with respect to the sources (element centers). In this case, however, the displacement between members of the source and target sets is less than half a core size and relatively negligible in discussing cluster geometry.

target particles.

The topology of the locally essential tree and the methods by which it is constructed depend strongly on the domain decomposition scheme and on the overall parallel implementation. Section §4.2.4 discusses these parallel implementation issues in detail.

Regardless of the global tree's topology, the fundamental issues of geometry in domain decomposition are the same. Contrast cases of good and bad partition over targets, illustrated in Figure 4-1 for two processors. In the worst case, case (a), the partitions are interleaved; that is, the convex hull of particles assigned to one processor overlaps with the hull of points on another processor. Even if this is not the case—consider, for instance, long and narrow non-overlapping domains as in Figure 4-1(b)—the locally essential trees on different processors can still overlap strongly. This means that computations evaluating the influence of cells deeper in the hierarchy and at higher order are duplicated across processor domains, and parallel efficiency will be poor. To minimize the overlap of locally essential trees, one should minimize the surface-area-to-volume ratio of each domain, favoring spherical, non-overlapping partitions as in Figure 4-1. Note that in this discussion, the concept of "overlap" characterizes not the spatial extent of trees, but the duplication of locally essential cells at each level of the tree hierarchy.

An alternative approach to parallelization avoids constructing a global tree (i.e., a locally essential tree), and instead can be viewed as "parallelizing over sources." The flat partition resulting from k-means clustering is perhaps better suited to this approach. Here, the domain decomposition scheme is applied to the particles and a local adaptive oct-tree is constructed in each processor's domain, as shown schematically in Figure 4-2. The target particles are left unorganized, and each processor computes the influence of its source tree on the entire set of target particles. Global reduction operations then sum the contribution of each processor to the velocity at every target.

With this approach, the geometric considerations governing good domain decomposition are analogous to those described before. Cases of good and bad source

partition are shown schematically in Figure 4-2 for three processors. Again, the worst case partition is one in which source points belonging to the three processors are interleaved. Though these sets of points are interleaved in space, they belong to distinct local trees, and thus their influence must be approximated with up to three times as many particle-cluster interactions as necessary (k times in the case of k interleaved partitions). But the situation persists even in the case of non-overlapping partitions. Consider the partition shown in Figure 4-2(b). The source domains are long and narrow, and they constrain the shape of each local oct-tree accordingly. One can enumerate Nk pairs of source domains interacting with target particles; few of these pairs specify targets that are well-separated from sources. The closer a target particle lies to a source domain, the more expensive the interaction; evaluation of the velocity induced on the target will descend to cells deeper in the local source hierarchy and/or employ higher-order expansions. The relative lack of well-separated domain-target pairs is equivalent to noting that the surface-area-to-volume ratio of each source domain is large, compared to the partition in Figure 4-2(c). Here, the domains are compact, and the domain boundaries are more nearly spherical. As a result, velocities at the targets may be computed with fewer particle-cluster interactions. This is illustrated schematically for a target particle in the center of the particle distribution.

The qualitative discussion above emphasizes the critical role of *partition geometry* in N-body problems, whether the partition is used to separate target particles or to fix the root cells of local source trees. For identical sets of particle distributions and weights, the partition geometry directly determines the number and order of particle-cluster interactions necessary to evaluate the velocity at each particle, summed over all domains.

In this work, we use k-means clustering to construct a partition of the source particles, "parallelizing over sources" as described above. A local oct-tree is then constructed from each processor's assigned particles and the velocities induced by each processor's source tree are summed at each target. The root cell of each source tree is thus a k-means cluster. Because these clusters minimize the cost function J

in (4.10), their boundaries define tightly-localized, convex sets, as discussed in §4.2.1. Based on the preceding discussion, this partition geometry will favor smaller numbers and lower orders of particle-cluster interactions, for greater parallel efficiency.

For clarity, we have focused the discussion in this section on computational time, not on the communication time associated with retrieving non-local cell or particle data. In fact, our current implementation keeps copies of all the particle positions and weights on each processor; on a modern computer, this allows for problem sizes of up to $N = 10^7$ and thus does not constrain the present vortex simulations. But we concentrate our discussion as stated above for more fundamental reasons. First, we view the relationship of partition geometry to computational time as more fundamental, inherent to the accuracy and error bounds of multipole expansions—e.g., the distance from a target to a cell center, the magnitude of the multipole moments of the cell. Communication time is typically more implementation-dependent—it can depend on the hardware interconnect or on latency-hiding features of the message passing architecture—and in N-body problems is usually much smaller. Secondly, computational time and communication time are not really separable issues. Minimizing the number of particle-cluster interactions has the simultaneous benefit of minimizing time spend on interprocessor communication, sending and receiving the cell or particle data. Also, k-means clustering yields relatively spherical domains, which in turn minimize the surface area-to-volume ratios often associated with communication overhead [14].

Considerations of optimal source geometry can be made yet more precise. The numerator in the error criterion of the LK treecode, (4.8), is the p-th absolute moment of cell c, $M_p(c)$. Note the correspondence between this sum and the cost function in (4.10). Our vorticity-weighted k-means algorithm finds a partition yielding a local minimum of $\sum_k^K M_2(c_{0,k})$ where $c_{0,k}$ are the resulting K root cells. While this flat partition cannot minimize individual absolute moments $M_2(c_{0,k})$, it will tend to make each one small. Furthermore, while the minimization of $\sum M_p$ strictly occurs for $p = 2$, $\sum M_p$ will generally remain small for other values of p, with possible exception of pathological cases.

Error estimates containing absolute moments of the form $M_p(c)$ are not limited to the LK treecode. In fact, they are a general feature of multipole expansions [104, 122]. For multipole expansions of the singular Biot-Savart kernel in spherical harmonics, Winckelmans et al [129, 122] report the following error bound:

$$e_p(\mathbf{x}) \leq \frac{1}{(d-b)^2}\left[(p+2)\frac{B_{p+1}}{d^{p+1}} - (p+1)\frac{B_{p+2}}{d^{p+2}}\right] \tag{4.11}$$

where $e_p(\mathbf{x})$ is the L_2 error on $\mathbf{u}_\omega$ at the evaluation point $\mathbf{x}$, $d = |\mathbf{x} - \mathbf{y}_c|$, b is the radius of the smallest sphere centered at x_c and containing all the particles in the cell, and

$$B_p(c) = \int_c \|\mathbf{x}' - \mathbf{y}_c\|^p \, \|\boldsymbol{\alpha}'\| \, d\mathbf{x} \approx M_p(c) \tag{4.12}$$

In this case, the error thus depends not only on the cell moment but on the effective cell radius b, another quantity that will generally be small in a k-means partition.

In all these cases, error bounds or error criteria directly control the computational cost of the treecode by affecting the order of expansion (in a variable-order code) and the choice of cells used to sum the velocity at each target. When the inequalities in (4.8) and (4.11) cannot be met, evaluation of the velocity on a target particle must proceed at higher order or descend to the children of the cell; in the latter case, a single particle-cluster interaction may be replaced with up to eight interactions. (In the limit, tree descent typically devolves into direct summation; criterion for this depend on the structure of the particular treecode.) Reducing the cell moment $M(c)$ and cell radius b allows an error criterion to be met for *smaller* cell-to-target distances R or d, avoiding the need to increase the order of expansion p or descend further into the hierarchy.

In the present implementation, k-means clustering determines the configuration of each root cell. The geometries of child cells in the local hierarchy are strongly influenced by the root, however. In other words, it is reasonable to expect a good root cell geometry to maintain small cell moments and radii several levels into the hierarchy. We explore the impact of the depth of local trees on performance in the Results section.

As described in §4.1.2, the serial LK treecode uses a user-specified accuracy parameter ϵ to control the velocity error at each target point, due to all the sources. In order to maintain the same global error specification in the present implementation, a global accuracy parameter ϵ must be distributed to each processor's local tree. We take the following approach, patterned after the fractional distribution of ϵ from parent cells to child cells inside the LK oct-tree [77].

$$\epsilon(c) = \frac{M_0(c)}{M_0}\epsilon \tag{4.13}$$

In other words, the global error parameter is distributed to each k-means cluster c in proportion to its total vorticity magnitude.

A few other approaches to domain decomposition of treecodes will be reviewed in Section 4.2.4.

4.2.3 Dynamic load balancing

While k-means partition yields domain geometries that favor reduced computational cost, this partition comes with no guarantee of load balance. Load balance is a difficult issue in N-body problems. Irregular particle distributions, often with a wide range of particle densities, are typical and thus equipartition (ensuring the same number of particles in each domain) does not ensure load balance. Contrast, for instance, the cost of computing the velocity at a target that is well-separated from other particles to the cost of computing the velocity at a target in a densely populated region; clearly, more particle-cluster interactions must be used to compute the latter [122]. Indeed, it is difficult to define or estimate a spatial distribution of "per-particle-cost" *a priori*, as this quantity depends both on the particle distribution and on how the distribution is partitioned.[2]

[2]The concept of "per-particle-cost" is perhaps most meaningful when considering each particle's role as a target, for then the time or number of interactions required for velocity evaluation at each particle is directly measurable, though even this measurement may be disrupted by one-time costs like the fill-in of multipole moments or the retrieval of faraway data to build locally essential trees. When considering partition over sources, the "per-particle-cost" becomes somewhat more ill-defined as the influence of each particle is replaced by multipole expansions of source cells. Allowing variable orders of expansion could make this per-particle cost even less consistent.

In dynamic N-body problems, particle locations and weights change in time, and thus it is advantageous to repartition the domain and re-balance computational loads as the particle distribution evolves. Vortex particle methods render the load balancing process more challenging because of local mesh refinement; at each timestep, new particles are introduced throughout the domain in order to maintain resolution and core overlap [72, 75, 82]. The spatial distribution of the newly inserted particles is itself highly irregular and hard to predict.

To address these difficulties, we develop several heuristics for the dynamic load balancing of k-means clusters. The first of these introduces scaling factors s_k into the weighted k-means cost function along with a rule to adapt their values in time:

$$J = \sum_{i=1}^{N} \min_{k'} \left(s_k |\mathbf{x}_i - \mathbf{y}_{k'}|^2 |\boldsymbol{\alpha}_i| \right) \tag{4.14}$$

The factors s_k scale the squared Euclidean distance between each cluster centroid and the surrounding particles, and thus modify the assignment at each k-means iteration. Each particle is assigned to the centroid from which its *scaled* distance is smallest. The space around each centroid is effectively "zoomed" in or out by the scaling factor. The resulting classification boundaries still define convex sets, but the cutting planes are no longer equidistant from the nearest two centroids, as in a Voronoi tesselation; rather they are shifted closer to the centroid with larger s_k. Figure 4-3 illustrates the geometry of these scaled k-means clusters.

To dynamically load balance the cluster populations, we update the scaling factors s_k at the start of each timestep. In general, the adaptation rule for scaling factors should express dependence on the previous timestep's scalings s_k^n as well as the times t_k^n spent evaluating the influence of each cluster's particles on the whole domain. Here the superscript n denotes the preceding time layer and $n+1$ is the current time.

$$s_k^{n+1} = f\left(s_k^n, t_k^n; k = 1 \ldots K\right) \tag{4.15}$$

In the present implementation, we choose a simple case of this adaptation rule, mul-

tiplicatively updating each s_k based on each cluster's deviation from the mean source evaluation time $\bar{t}^n = \sum_k t_k^n / k$.

$$s_k^{n+1} = s_k^n \left(1 + \alpha \tanh \left(\beta \frac{t_k^n - \bar{t}^n}{\bar{t}^n} \right) \right) \tag{4.16}$$

In this sense, we are using the *relative time* required to evaluate the velocity due to all the sources in clusters at the previous timestep as an estimate of the relative time required by a similarly-composed cluster at a nearby position in space. Cluster boundaries and centroids will change from timestep to timestep, as will the actual memberships—due to movement of particles, evolution of particle weights, and new particle introduction—but these changes should be incremental.

After each iteration, s_k will increase for a "high-cost" cluster and vice-versa. Expensive clusters will thus lose particles to their neighbors, while clusters with below-average t_k will incrementally become more attractive. The parameters α and β can be tuned; we find that $\alpha = 0.1$ and $\beta = 2$ give good performance. We also impose safety bounds to avoid unreasonable values of the scaling factors; after each application of (4.16) we require that $0.25 < s_k < 4.0$.

A new k-means partition is computed at each timestep, immediately after updating the cluster scalings. At the first timestep, all the s_k are set to unity and initial guesses for the centroids $\mathbf{y}_k$ are randomly chosen from among the particle locations. At subsequent timesteps, converged centroid locations from the preceding step are used as initial guesses for the current k-means partition. As a result, later applications of k-means converge much more quickly than the first. In practice, three or four k-means iterations are sufficient to achieve convergence for k-means processes that are initialized with the preceding step's centroids.

A second heuristic for load balancing the k-means domain decomposition involves modifying the centroid initializations themselves. As discussed in §4.2.1, the k-means algorithm yields only a local optimum, and thus the final partition may be quite dependent on the initial $\mathbf{y}_k$. We take advantage of this dependence by *splitting* the centroid of the highest-cost cluster at the end of each timestep. We select the particles

that were assigned to the cluster with maximum t_k^n and partition them with a local application of k-means, putting $k = 2$. The two resulting converged centroid locations are used as initial guesses for the full k-means iterations that partition the entire domain. Since the total number of centroids must remain constant (and equal to K), the centroid of the lowest-cost cluster is removed from the initialization list. Aided by these successive splittings, the centroid distribution will adapt itself to the particle distribution over time.

A final heuristic enables the load balancing scheme to recover from situations in which the random initial choice of centroids may be poor. If the load imbalance, defined as $(\max_k t_k)/\bar{t}$, exceeds a chosen safety threshold for two timesteps in a row, the centroids are "reseeded"—in other words, new centroid initializations are randomly chosen from among the current particle locations and the scaling factors s_k are all set to unity. For most of the runs reported herein, we set the threshold imbalance to 1.5 and observe that this level of imbalance is rarely encountered. (For more details, see section §4.3.5 below.)

4.2.4 Other frameworks for treecode parallelization

The preceding sections introduced k-means clustering as a new tool for the partition of hierarchical N-body methods. Other frameworks for parallelizing treecodes have been developed in the literature, however, and it is worthwhile to contrast their approaches to domain decomposition, load balancing, and interprocessor communication with the present k-means-based implementation.

Beginning with a parallel BH-type code developed by Warren and Salmon for astrophysical N-body simulations [123], there have emerged a family of parallel treecodes employing orthogonal recursive bisection (ORB) for domain decomposition [39, 15, 111]. ORB recursively partitions the computational domain into rectangular cells. At each level of the hierarchy, the domain is bisected along its longest coordinate dimension. The result of this partition is a binary tree with each leaf node corresponding to the domain of a single processor; for a tree with p levels, there are 2^{p-1} processors.

Load balance in ORB partition is controlled by positioning each bisecting plane

so that equal amounts of computational work lie on either side. Computational work is estimated on a per-particle basis, usually by counting the number of interactions necessary to evaluate a particle's velocity at the previous timestep.[3] Schemes have been devised for incrementally updating these bisector positions to maintain load balance [15]; other codes recompute the ORB partition at each timestep [123, 39].

Following ORB partition, velocity evaluation in all these codes proceeds by parallelization over targets, as described in the first half of §4.2.2. That is, each processor is responsible for evaluating the influence of the whole domain on its own particles. Each processor constructs a *local* BH tree in its own domain. The root nodes of these trees, corresponding to the leaf nodes of the ORB tree, are all shared among processors. Then, each processor builds a unique locally essential tree, i.e., imports the cells of non-local BH trees required to evaluate the velocity on local particles. Rather than transmitting this data as needed during tree traversals for velocity evaluation, these codes use simple multipole acceptability criteria (MAC) [8, 9, 123] to construct locally essential trees *a priori*. This process is typically organized by sender-driven communication; the owner of an ORB domain determines which of its cells may be essential to other ORB leaf nodes and sends appropriate multipole data. This is only possible for simple MACs, like the original cell-opening criterion of Barnes and Hut or variations thereof [8]. More complex and accurate error bounds like those in Equations 4.11 and 4.8 preclude the a priori construction of locally essential trees, particularly for variable-order treecodes [122].

Consistent with our description in §4.2.2, the locally essential tree is effectively a pruned global tree, incorporating the influence of the entire domain on the local particles. In [123] and [39], this tree is hybrid in structure—a binary tree on top (due to ORB) partition, and an oct-tree below the ORB leaf nodes. Bhatt *et al.* [15] go further by explicitly constructing a global oct-tree, resolving levels between the local

[3]Per-particle cost estimates are possible because velocity evaluation is parallelized over targets, as will be described below; and typically, the order of multipole expansion is fixed, so per-particle costs will remain more consistent. The "granule" of parallel partition (a target particle) can be directly associated with a cost, since velocity evaluation involves target particle-source cluster interactions. New particle introduction, however, will disrupt these estimates; this issue has not been addressed in the cited codes.

oct-trees; this process is rendered more difficult by adaptive features in the BH trees, like variable-size leaf nodes.

In contrast to these ORB-based codes, the hashed oct-tree (HOT) code of Warren and Salmon directly performs domain decomposition on the bodies of a global, distributed oct-tree [124, 105, 122]. Particle coordinates are mapped to 64-bit keys; the mapping is designed so that keys can identify not just particles (i.e., leaf nodes of the tree) but higher nodes of the tree. A hashing function maps keys to cell data, e.g., multipole moments and cell centers. In contrast to the pointer-based tree traversals employed above, hashing is designed to allow easier access to non-local cell data.

Domain decomposition in the HOT code proceeds by sorting the body key ordinates. Sorting these keys amounts to constructing a space-filling curve passing through all the particles with Morton ordering [106]. The curve is then partitioned into K segments, one for each processor, using estimates of per-particle cost to ensure that segments represent equal work. Branch nodes—the smallest oct-tree cell containing all the particles on a particular processor's segment of the curve—are then shared among processors to build the upper levels of the global oct-tree. Because it is the oct-tree itself that is partitioned, this stage of the algorithm avoids many of the complications of the ORB scheme [47].

Morton ordering preserves reasonable spatial locality, but is not ideal in this regard; the sorted list still contains spatial discontinuities that may be spanned by a single processor domain. These discontinuities can lead to inefficiencies in velocity evaluation [124], particularly in light of the discussion in §4.2.2. The hashed oct-tree scheme also requires that leaf nodes contain single particles, unlike adaptive treecodes that allow variable leaf-node size [15]. Also, cells of the hashed oct-tree are uniform rectangular prisms, and cannot shrink to fit their data, an adaptive feature that was found to improve the efficiency of the present LK treecode [77].

As in the ORB codes, velocity evaluation proceeds by parallelizing over targets. To allow the use of data-dependent error criteria like that in (4.11) the HOT code does not construct a locally essential tree *a priori*. Instead, each processor requests cell data from other processors as needed while evaluating the velocity on its own

particles. A complicated system of communication lists is used to hide the latency of requesting faraway data.

A different, more theoretical approach is taken by Teng [119]; his work analyzes the *communication graphs* of hierarchical N-body algorithms, including the BH scheme, and proposes algorithms for their load-balanced partition. The communication graph, defined on particles and oct-tree cells, represents the interactions between these objects during the execution of the N-body algorithm; the edge weights reflect communication requirements of each interaction. A good partitioning algorithm, in this analysis, yields an edge-partition of the communication graph into two disjoint graphs of equal (vertex-weighted) computational cost, while keeping the "cost"—the total weight of all edges removed—small. While this and other studies of graph partition algorithms [92, 68], including recursive bisection [110], provide useful theoretical results on the partition of oct-trees and other data structures, they do not address the problem of what the tree objects themselves should look like. Teng's algorithm only considers partitioning the cells of an existing, generic oct-tree. Yet cell geometry can have a profound effect on computation and communication costs, as discussed above and as will be demonstrated in the next section. Guided by these considerations, K-means clustering yields an entirely new class of geometric objects, forming an adaptive spatial partition of N-body interaction.

We re-emphasize the following points:

- More accurate, more complex error estimates preclude the a priori construction of locally essential trees. So do adaptive features, like a run-time choice between direct interaction and multipole expansion.

- Variable order expansion and adaptive features of the tree (like cells that shrink to fit, run-time choice between direct interaction and multipole expansion) make per-particle cost more difficult to define, even when parallelizing over targets. Moreover, the present implementation parallelizes over sources.

- ORB and sorted hash keys do not create optimal domain geometries. Domains can have poor shape, even spatial discontinuities; for instance see Figure 3 in

[39]. The hashed oct-tree construction further limits tree adaptivity.

- None of these studies show the performance of load balancing while new particles are continually being introduced, e.g., through filament remeshing. But we will do so below.

4.3 Results

In the following, we examine the performance of cluster partition of N-body interactions by a variety of measures—speed and parallel efficiency, error control, load balance, and particle-cluster interaction counts. In particular, we apply the k-means clustering and load balancing algorithms developed in §4.2 to the parallel hierarchical evaluation of vortical velocities in a vortex simulation of a transverse jet at high Reynolds number.

The mixing properties of the transverse jet—a jet issuing normally into a uniform crossflow—are important to a variety of engineering applications. Transverse jets may function as sources of fuel in industrial furnaces, or as diluent jets for blade cooling or exhaust gas cooling in industrial or airborne gas turbines. The transverse jet is a canonical example of a flow dominated by large-scale "coherent structures." Experimental observations by Fric and Roshko [44] identify four such structures in the transverse jet: jet shear layer vortices; "wake vortices" arising from interaction between the jet and the channel wall boundary layer; horseshoe vortices that wrap around the jet exit; and a counter-rotating vortex pair that forms as the jet bends into the crossflow, persisting far downstream. The evolution of these structures is inherently three-dimensional and characterized by topological changes in the vorticity field. Vortex methods are attractive in this context for their explicit link to the formation and dynamics of vortical structures in the flow.

Details of our vorticity formulation and a thorough analysis of the flow physics revealed by vortex simulation are presented in Chapters 2 and 3. Here, we merely summarize aspects of the simulation that are relevant to the parallel N-body problem. Vorticity entering the flow at each timestep is discretized with vortex particles that lie

on partial filaments [84]. Filaments result from our physically-motivated expression of vorticity flux boundary conditions, but also provide a convenient mechanism for local remeshing in response to flow strain. Filament geometries are described by cubic splines supported by a finite set of nodes. Nodes are advected by the local velocity field using a second-order predictor/corrector method with timestep control. Advecting the nodes accounts for deformation of the material lines and thus for stretching and tilting of the vorticity and the corresponding modification of element weights $\boldsymbol{\alpha}_i = \boldsymbol{\omega} dV_i$, since vortex lines and material lines coincide. When the length $|\boldsymbol{\delta\chi}_i|$ of a given element exceeds 0.9δ, where δ is the regularization radius in (2.9), a new node is added at the midpoint of the element, thus splitting the element into two connected elements and enforcing the core overlap condition along the filament [82].

In addition to local insertion of vortex elements/nodes, we implement hairpin removal algorithms to remove small-scale folds along vortex filaments; this process regularizes the formation of small scales and thus reduces the rate at which elements proliferate. We also merge neighboring elements along filaments whenever the linear extent of an element becomes too small. The result of all these operations on filament geometry is an incremental remeshing which modifies the vortex particle distribution—a distribution that is also being modified by advection and by the evolution of particle weights $\boldsymbol{\alpha}_i(t)$. On balance, local element insertion, hairpin removal, and small element merging result in a net positive introduction of elements. Thus, not only do elements enter the flow at the jet nozzle, but they are created throughout the domain. This is typical of three-dimensional vortex methods [72, 7], and corresponds to the turbulent cascade towards smaller length scales via stretching and folding of vortex lines.

4.3.1 Single-step performance and scaling

We first examine timings for a single evaluation of vortical velocities. A "single evaluation" involves calculation of the full N-body problem, computing the velocity induced by every vortex element on every other vortex element. Of course, a higher-order time integration scheme (e.g., a Runge-Kutta scheme) may require multiple

such evaluations in a single timestep.

Figure 4-4 shows a representative particle distribution from transverse jet simulation, containing $N = 157297$ particles. Each vortex particle is represented by a sphere with radius proportional to the norm of the particle's vector weight $\|\boldsymbol{\alpha}_i\|_2$. The crossflow is directed in the positive x direction; the jet centerline is aligned with the y axis; and the z axis is in the spanwise direction. Flow variables are made dimensionless by d, the nozzle diameter, and U_∞, the crossflow velocity. The jet orifice is thus a disc of diameter one centered at the origin of the x-z plane; the remainder of the x-z plane is a solid wall through which we enforce a no-flow boundary condition. The ratio of the jet velocity to the crossflow velocity, denoted by r, is 7.

A few comments on the particle distribution are in order. Vortex particles are introduced at the edge of the jet nozzle every Δt_{noz} time units; the number of particles introduced at each such instant is n_θ. Here we take $\Delta t_{noz} = 0.01$, $n_\theta = 64$, and the particle core radius $\delta = 0.1$. As particles enter the flow, they initially compose a cylindrical shear layer which rolls up 1–2 diameters above the jet. Roll-up of the shear layer is manifested by grouping of the particles into vortex rings; but as these rings form, they stretch and deform out-of-plane in a process that is intimately linked to the formation of a counter-rotating vortex pair aligned with the jet trajectory. Counter-rotating vorticity further disrupts the particle distribution as vortex filaments wind and stretch and as opposite-signed vorticities approach each other. The jet then bends further into the crossflow and the particle distribution—in terms of both locations and weights—becomes enormously complicated as smaller scales are generated and the particles fill the space. Again, a complete analysis of these vortical transformations is in Chapter 3.

We consider two different vortex element distributions, both drawn from these simulations of an evolving transverse jet—one with smaller N ($N = 261,481$) and one with larger N ($N = 1,164,184$). The larger case is reached approximately 0.4 convective time units after the smaller one, and thus represents not only more particles but a particle distribution that has evolved slightly further downstream and developed more small scales.

For each of these two cases, we compare three partitioning schemes. The first, labeled "block distribution," is simply a block partition of vortex element arrays. Within the arrays, elements are arranged in order of (1) *where* they appear along a filament and (2) *when* the filament was introduced into the flow. Since elements on successive filaments will share somewhat similar trajectories, this distribution preserves some data locality, as shown in Figure 4-5(a). Nonetheless, some interleaving is present. Performance of this "naive" partition is not expected to be good, but it is a convenient and straightforward partition to compute for purposes of comparison. Each domain contains essentially the same number of particles, N/k.

The second partitioning scheme is k-means clustering as presented in §4.2.1, with no attempt at correcting the load balance. Thus all the scaling factors s_k are set to 1.0, and the iterations for cluster centroids are allowed to begin from a random initial seed of particle locations. The third partitioning scheme employs k-means clustering but adds the load balance heuristics developed in §4.2.3. The precise procedure for obtaining this partition is as follows: First, k-means clustering is performed with unity scalings and random initial seed. Then, the velocity is evaluated and cluster timings t_k^n are obtained. The highest-cost cluster is split and the scalings s_k are updated with one application of (4.16), then the partition is re-computed. In other words, the difference between the second and third partitioning schemes is one iteration of the load balance heuristics.

Two exceptions to this procedure are the $k = 1024$ "scaled clusters" cases in Figures 4-6–4-9, identified by the filled-in square symbol in each figure. Timings for these cases were obtained from an actual, dynamic vortex element simulation, and hence are the result of many load-balancing iterations applied to an evolving particle distribution. These cases serve to illustrate the realistic performance of load-balanced clustering in a dynamic simulation.

For each particle distribution, with each of the three partitioning schemes, we scale the number of processors from 1 to 1024, choosing $k \in \{1, 16, 64, 128, 512, 1024\}$. Figures 4-6(a) and 4-6(b) show the total time of velocity evaluation in each of these cases. We set the treecode accuracy parameter $\epsilon = 10^{-2}$ and the leaf size parameter

$N_0 = 512$. Velocity evaluation times reflect both the time necessary to evaluate the velocity at each vortex particle in parallel *and* the overhead of interprocessor communication (i.e., for global reduction operations after every processor has finished evaluating the velocity induced by sources in its domain). This can be broken down, using the notation of the previous section, as $T = \max_k t_k + t_{\text{comm}}$. From these figures, it is clear that the block distribution results in slower velocity evaluations than either of the cluster distributions. Load-balanced clustering results in faster evaluations than plain k-means clustering, particularly for $k > 128$. Regardless of the partition scheme, adding more processors leads to faster evaluations (not a surprising result).

It is instructive to cast the timing data in terms of speedup S, where

$$S = \frac{T_{\text{serial}}}{T_{\text{parallel}}} \tag{4.17}$$

These data are shown in Figure 4-7. Ideal speedup is equal to k. Clearly, the block distribution performs poorly compared to the clustered distributions, yielding a speedup of less than 200 when using 1024 processors. Scaled k-means outperforms plain k-means, especially for $k > 128$. An additional gain in speedup is seen in the the scaled $k = 1024$ cases, denoted with solid squares in Figures 4-7(a) and 4-7(b); recall that these partitions result from successive load-balance iterations, whereas scaled cases with $k \neq 1024$ are only one load balance iteration away from their unscaled counterparts. In general, the speedup of scaled k-means clusters in the larger N case is quite close to the ideal speedup. This comparison is better distilled by plotting parallel efficiency P, defined as follows

$$P = \frac{T_{\text{serial}}}{k * T_{\text{parallel}}} \tag{4.18}$$

and shown in Figure 4-8. Parallel efficiency of block distribution falls off rapidly at relatively small k and approaches a value below 20% in both the large N and small N cases. With the cluster partitions, the parallel efficiency observed in the large N case is significantly better than in the small N case. This may be due to the proportionally smaller communication cost of the former, or may point to some subtle

interaction between the granularity N/k and the maximum leaf node size N_0. In any case, problem sizes of $N > 10^6$ are much more representative and computationally demanding targets for parallel hierarchical methods, particularly for large $k \approx 10^3$. In this case, we observe parallel efficiencies consistently above 85% for scaled k-means clusters. The most realistically load-balanced case, $k = 1024$, shows a remarkable parallel efficiency of 98%.

The load imbalances underlying the single-step timings just presented are shown in Figure 4-9. We define load imbalance I as follows:

$$I = \frac{\max_k t_k}{\bar{t}} \tag{4.19}$$

Several features are worth noting. First, block distribution shows the best overall load balance, with an imbalance below 1.3 in all cases of k and N. Although every domain in a given block distribution has essentially the same number of particles, per-particle cost is not uniform, as discussed in §4.2.3, and thus some imbalance will be present. Nonetheless, this imbalance is relatively small, and illustrates the fact that good load balance is no guarantee of parallel efficiency. *Domain geometry* has a key role in determining parallel efficiency; and thus all the cluster partitions, though they may exhibit larger load imbalances, show vastly better parallel performance than the block partition.

Load imbalance with plain k-means clustering, shown by the dash-dotted line in Figures 4-9(a) and 4-9(b), tends to increase with the number of processors, although jaggedness in this curve reflects the fact that, without any attempts at controlling the relative cluster populations, load balance in the clustered case depends on the random initial seed. After all, it is the highest-cost cluster that determines the load imbalance. One application of the load-balancing heuristics reduces the imbalances to those observed on the solid line, for $k \neq 1024$. Successive load-balancing iterations, even though they are performed on a dynamically evolving particle distribution, reduce the imbalance even further—to approximately 1.3 in both $k = 1024$ cases. Again, we emphasize that this value is typical of the imbalance observed in full simulations.

We will comment further on the performance of successive load-balance iterations in §4.3.5.

Examination of Figure 4-8(b) and Figure 4-9(b) together motivates an additional observation. Consider the scaled k-means partitions for large N: while these cases have load imbalances of 1.3–1.7, they have parallel efficiencies above 85%. In particular, consider the $k = 1024$ case, with its load imbalance of 1.355 and parallel efficiency of 98%. If the load balance were further improved, the parallel efficiency would clearly be higher than 100%. Suppose, for instance, that the load imbalance in this case could somehow be reduced to 1.0, and suppose further that perfecting the load balance would not shift the mean cluster time, $\bar{t}$, or change the communication overhead T_{comm}. Then, using a simple breakdown of computational costs,

$$P = \frac{T_{\text{serial}}}{k * T_{\text{parallel}}} \approx \frac{T_{\text{serial}}}{k\left(T_{\text{comm}} + I * \bar{t}\right)} \tag{4.20}$$

we would find a parallel efficiency of 130%. Without communication overhead, we would observe a speedup of 1536—a "parallel efficiency" of 150%. While this situation seems entirely hypothetical, it illustrates that our actual parallel partition performs better than the load balance would lead one to expect. Why is this the case? There may be some gains in speed due to better use of cache in the parallel computation. But a factor that cannot be overlooked is the difference in the geometry of the hierarchical partition between the serial and parallel cases. The serial case has no k-means clusters; instead it has a single adaptive oct-tree covering the entire domain. Clusters partition the domain differently, and it may be that the resulting hybrid partition—with k-means clusters serving as root cells of small oct-trees—is more efficient than even the original oct-tree.

4.3.2 Particle-cluster interaction counts

While the timings reported in the preceding section are the ultimate practical measure of performance, additional insight into the effect of geometry on computational cost may be gained by counting particle-cluster interactions.

The possibility raised at the close of the previous section—that k-means clustering may provide a better partition of the domain— has implications beyond parallel decomposition. It is difficult to explore this possibility with measures of computational time alone, however, as factors like communication overhead, along with memory and cache access patterns, will color the timing data in a hardware-dependent fashion.

Figure 4-10 shows the number of source particles evaluated at each order of expansion p or with direct summation, for a single computation of vortical velocities. We use the $N = 1164124$ particle distribution presented earlier and consider three different partitions: (1) block distribution with 1024 domains, (2) k-means clustering with $k = 1024$ clusters, and (3) a single adaptive oct-tree. The last partition is simply the serial ($k = 1$) case in §4.3.1, employing all the adaptive oct-tree features discussed in [77].

Particle-cluster interactions are counted as follows. Each time the velocity induced by a source cell of n_c particles is evaluated at some order p, n_c is added to a counter corresponding to that p. Of course, every target particle interacts with N source particles through expansions at different levels of the tree, and thus the sum of the ordinates on each curve in Figure 4-10 is equal to N^2. The curves differ, however, in their distribution. The block partition yields a very large number of direct interactions, accounting for its poor parallel efficiency. The global oct-tree shows that the largest number of particle participate in order $p = 7$ interactions, and that very few cells are expanded at $p \leq 5$. The k-means partition, by contrast, shows lower orders of expansion. Some particles are expanded at order $p = 4$, and the distribution peaks at $p = 6$. Lower-order expansions are less expensive—both on a per-cell basis, because the number of terms at each order is $O(p^3)$—and on a per-particle basis, because the initial calculation of higher-order moments may be avoided. K-means thus yields a partition on which hierarchical velocity evaluations may be performed more efficiently, at lower computational expense.

Figure 4-10 very clearly shows the impact of geometry on computation cost—in particular, how geometry directly controls the number and order of particle-cluster interactions necessary to evaluate velocity within a given accuracy. This confirms the

mechanisms discussed in §4.2.2 and carries implications for the geometric constructions on which hierarchical methods are built, transcending issues of serial or parallel implementation. Future work that extends these ideas will be discussed in §5.2.1.

4.3.3 Error control

While noting substantial gains in parallel performance, it is important to verify that the accuracy of velocity evaluation is well-controlled with the present algorithm. We compute the exact velocity at each target particle $\mathbf{u}_i^{dir}$ using direct summation and compare these values to those obtained with the parallel treecode, using both block and cluster partitions. This comparision is performed for two cases of N: $N = 261,481$ (the small-N case used in the previous two sections) and $N = 102,505$. Because of the high cost of direct summation, performing this comparison for much larger N would have been computationally prohibitive. The treecode velocities were obtained for two different values of the accuracy parameter, $\epsilon = 10^{-2}, 10^{-4}$. Cluster partition is performed without splitting of high-cost clusters or scaling, as load balance is not expected to have much effect on velocity error.

Figure 4-11 shows the absolute error in velocity as a function of k, for $k = 1, 16, 64, 256$. This error is defined as the maximum, over all the target particles, of the velocity error magnitude:

$$e_{\text{abs}} = \max_i \left\| \mathbf{u}_i - \mathbf{u}_i^{dir} \right\|_2 . \tag{4.21}$$

We find that this error is very well-controlled as the velocity evaluation is further parallelized—it remains at or below the serial ($k = 1$) error for all k, with only one exception ($k = 16$ and $N = 102505$). Block distribution seems to produce smaller errors than cluster distribution as k increases. With either partition, the reduction in error with higher k is more pronounced for the $\epsilon = 10^{-2}$ case than the $\epsilon = 10^{-4}$ case. Similar trends are observed with the relative magnitude of the velocity error,

defined as

$$e_{\text{rel}} = \max_i \left(\frac{\left\| \mathbf{u}_i - \mathbf{u}_i^{dir} \right\|_2}{\left\| \mathbf{u}_i^{dir} \right\|_2} \right). \tag{4.22}$$

and shown in Figure 4-12. The fact that errors are smaller in the parallel cases than in the serial case suggests that we are over-constraining the error. By distributing fractions of the global accuracy parameter ϵ to all clusters, as in Equation (4.13), we are in a sense replacing one constraint with k independent constraints. As a result, the bound on the total error is too conservative. A more sophisticated distribution of the error parameter to clusters—one that keeps e_{abs} or e_{rel} from declining with higher k—could conceivably further reduce the time of parallel velocity evaluation, for additional gains in computational efficiency.

4.3.4 K-means performance and scaling

The calculation of a k-means partition carries its own computational cost, and it is desirable, for an effective parallel domain decomposition, that this cost (1) remain small relative to the actual cost of evaluating the N-body interaction and (2) scale well with problem size. Figure 4-13 shows the time per iteration of the k-means algorithm as a function of the number particles N for different values of k. In this context k is both the number of clusters *and* the number of processors performing the clustering. As a result, the overall computational complexity of a single k-means step, $O(Nkd)$, is divided by k, and we should see $O(N)$ scaling. This is borne out in the figure, as the lines corresponding to different k lie on top of each other and are relatively indistinguishable. Thus, the parallel efficiency of our parallel k-means implementation is very near 100%. The absolute time per iteration is quite small compared to the velocity evaluation times in Figure 4-6 for similar N, and, for a given k, the $O(N)$ scaling of k-means is asympotically smaller than the $O(N \log N)$ scaling expected of a BH-type treecode. Converging to a k-means partition may require several iterations; in fact we set $l_{max} = 7$ when starting from a random seed of centroids and $l_{max} = 4$ otherwise. But the total cost of these iterations is still small compared to the cost of velocity evaluation, especially when one considers that

higher-order time integration schemes like Runge-Kutta will perform several velocity evaluations with a single k-means partition.

4.3.5 Dynamic load balance

The performance of the load-balancing algorithms developed in §4.2.3 is best demonstrated in a dynamic N-body simulation; here we use the transverse jet simulation, complete with evolving vortex particle locations and weights *and* new particle introduction throughout the domain.

First, we examine the distribution of processor times t_k for a single velocity evaluation step. We choose the $N = 1164184$ case used earlier, extracting this case from two simulations. One simulation employs the load-balancing algorithms—i.e., iterated scaling and split/merge of cluster centroids—and the other does not, instead reseeding the centroids at each step. The processor times t_k do not include any communication overhead; they consist of the time required by each processor to evaluate the influence of its source particles on the whole domain. These times are normalized by $\bar{t}$ and used to populate the histogram in Figure 4-14. Clearly, the distribution of processor times is much narrower in the load-balanced case. The maximum processor time determines the load imbalance, which is approximately 1.4.

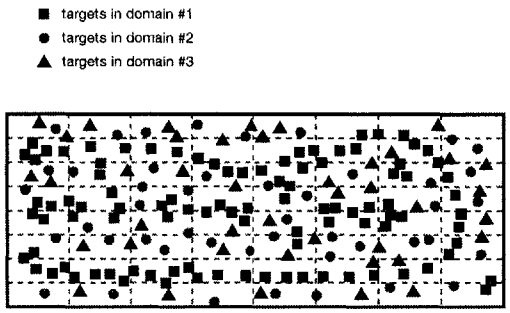
We can extend this analysis to successive velocity evaluations and thus find the averaged normalized load distribution on an evolving field of particles. Figure 4-15 shows normalized processor times for 36 successive velocity evaluations, with N growing from 10^6 to 2.5×10^6. The load-balanced simulation again shows a significantly narrower load distribution than the simulation performed with plain k-means clustering. The load distribution in the latter case has a long tail above the mean processor time; iterated scaling and split/merge in Figure 4-15(b) seem to control the extent of this tail quite effectively.

Figure 4-16 shows the load imbalance at each step of the simulation. Since we are using a second-order Runge-Kutta method for time integration, there are two values of the imbalance at every value of N, corresponding to two different velocity evaluations. The load imbalance with plain k-means ranges up to 2.5 and shows significant

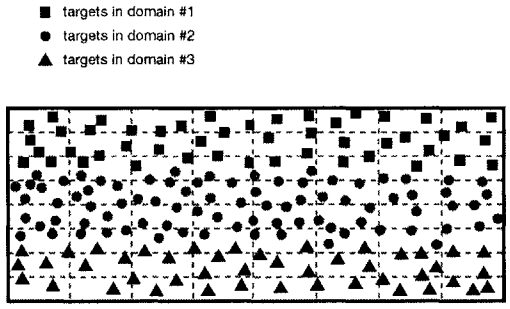
variation from step to step. In contrast, the load-balanced clustering maintains an imbalance well below 1.5 for much of the simulation, and has a better bounded step-to-step variation as well. The contrast in values of the load imbalance is particularly appreciable at large values of N.

Of course, the ultimate measure of performance is velocity evaluation time, to which load balance is a contributing factor. Figure 4-17 plots the total velocity evaluation time—including communication overhead—at every step of a simulation of the evolving transverse jet. The dashed line shows velocity evaluation times using plain k-means clustering; as suggested by the plots of load balance, this line is quite jagged, and its deviation from the load-balanced case becomes significant at large N. Introducting the load-balance heuristics results in a smoother profile of evaluation time versus N; at larger values of N, the computational savings, relative to the unbalanced case, can be as much as 50 seconds per evaluation. The hypothetical ideal performance is represented by the dash-dotted line, which shows the mean velocity evaluation time $\bar{t}$ plus communications overhead. This is the velocity evaluation time that could be achieved with perfect load balance. While there is some gap between this line and the actual load-balanced simulation at large N, the present simulations perform remarkably well relative to this ideal.

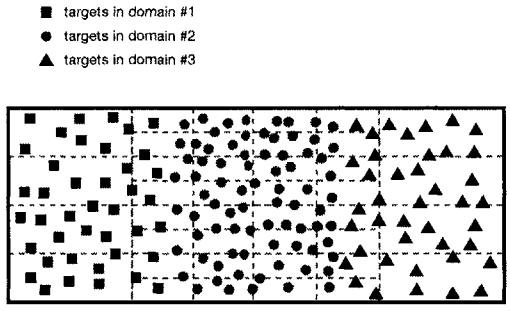
Figure 4-17 also shows a relatively favorable increase in velocity evaluation time versus N. The trend is best observed with either the load-balanced evaluation time or the mean evaluation time. While we cannot make strict conclusions about scaling, since the particle distribution is evolving as N increases, the mean velocity evaluation time shows a growth that appears somewhere between $O(N)$ and the $O(N \log N)$ that would be expected for a Barnes-Hut-type treecode.



(a) Interleaved.

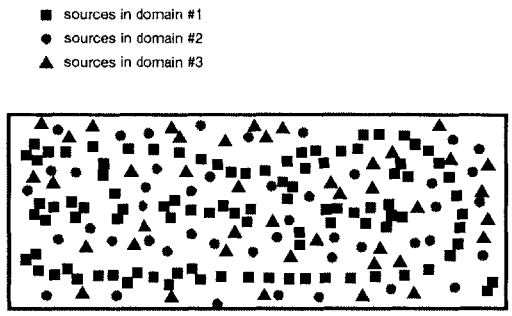


(b) Non-interleaved.

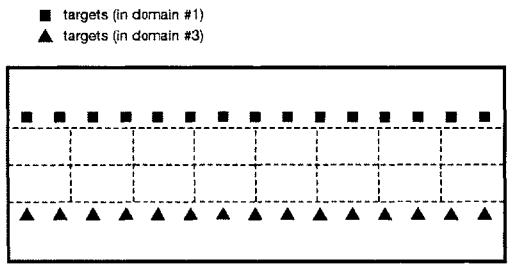


(c) Well-localized.

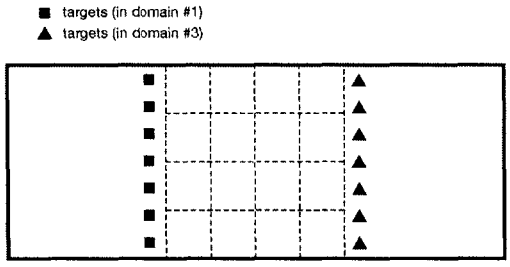
Figure 4-1: The geometry of partitions over target particles, illustrated with three processors. Dashed lines outline the locally essential tree for domain #2.



(a) Interleaved.



(b) Non-interleaved.



(c) Well-localized.

Figure 4-2: The geometry of partitions over source particles, illustrated with three processors. Dashed lines represent the local source tree for domain #2. Note that the quadtree in (b) employs an adaptive bisection to control the aspect ratio of its cells, but still demonstrates the larger number of particle-cluster interactions that accompany poor domain geometry.

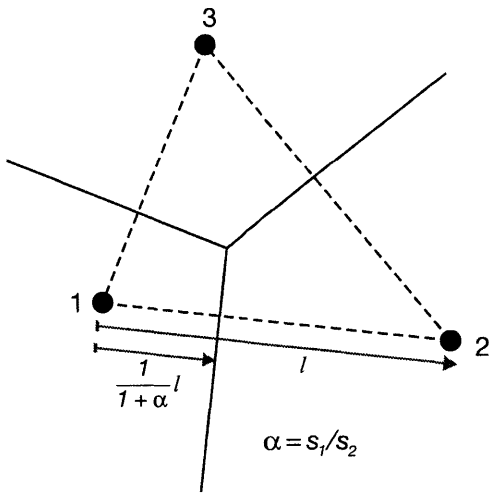


Figure 4-3: Two-dimensional schematic of scaled k-means cluster boundaries, with three clusters. The numered solid circles are cluster centroids. Clusters have scaling factors s_i; here, $s_1 > s_2$.

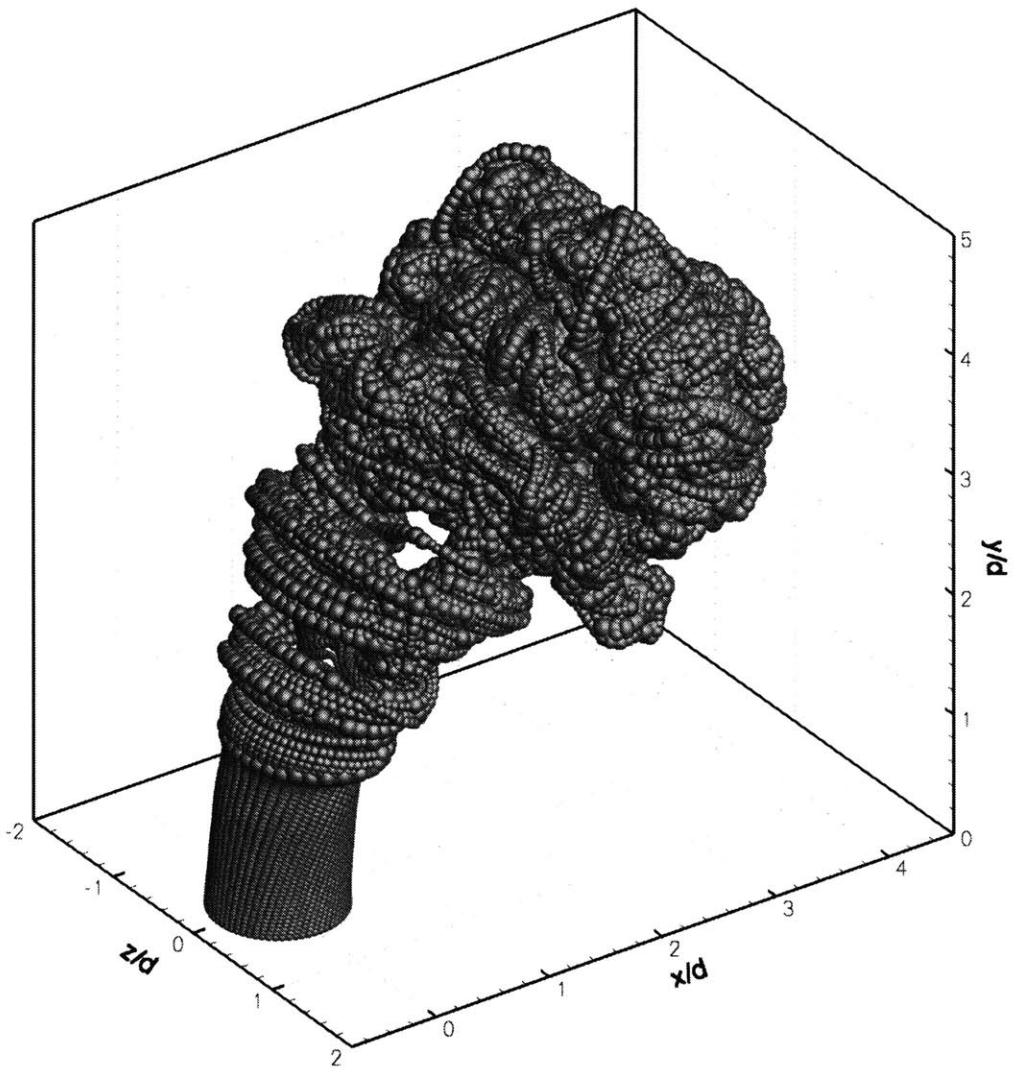
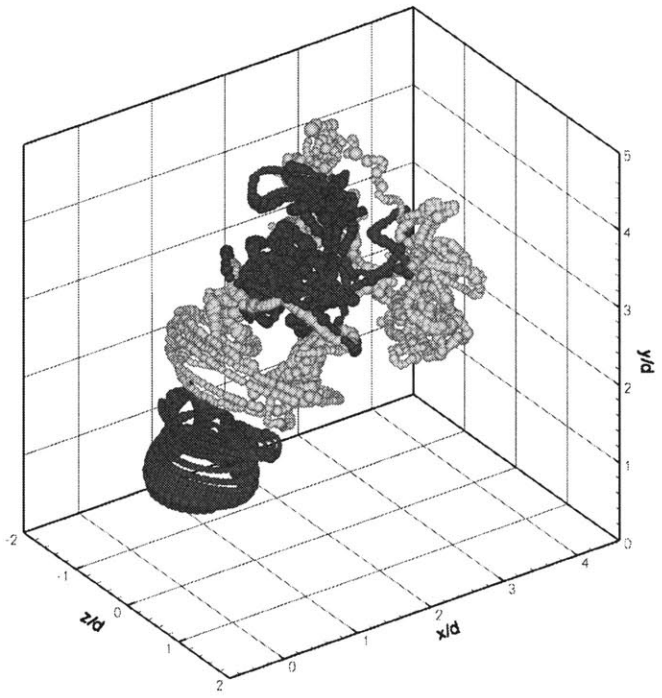
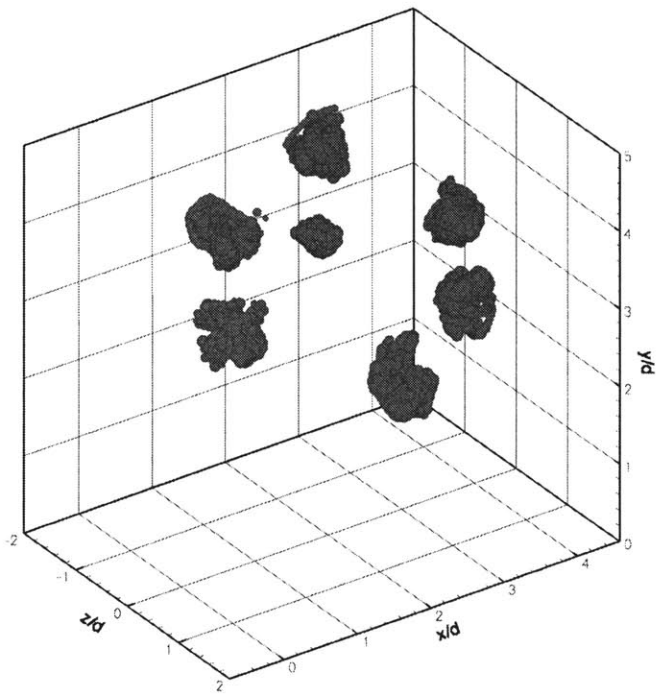


Figure 4-4: Vortex elements in the transverse jet at $t = 2.0$; $N = 157297$. Particle sizes are proportional to $\|(\boldsymbol{\omega} dV)_i\|_2$.

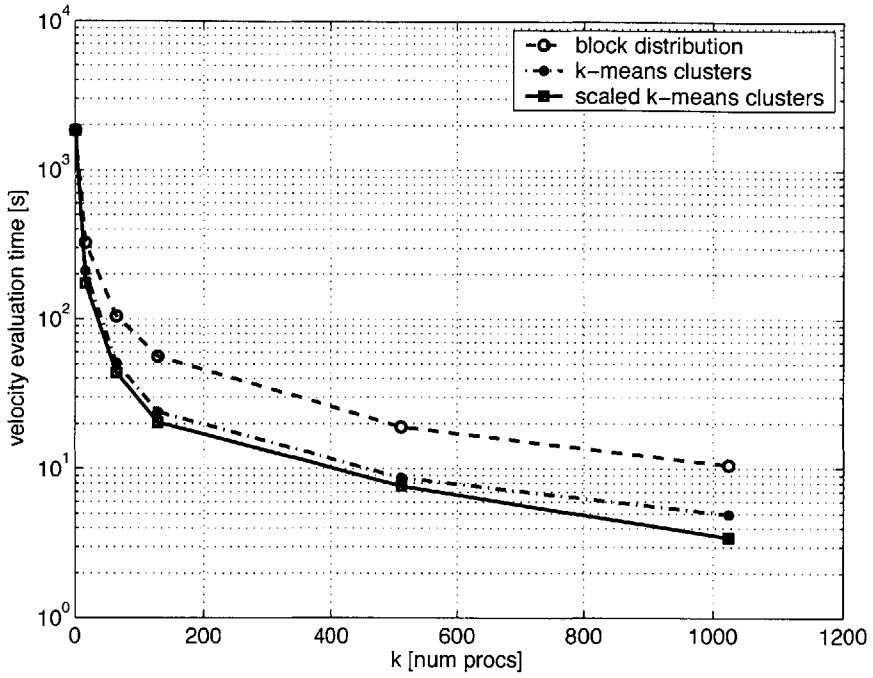


(a) Block partition, 4 domains.

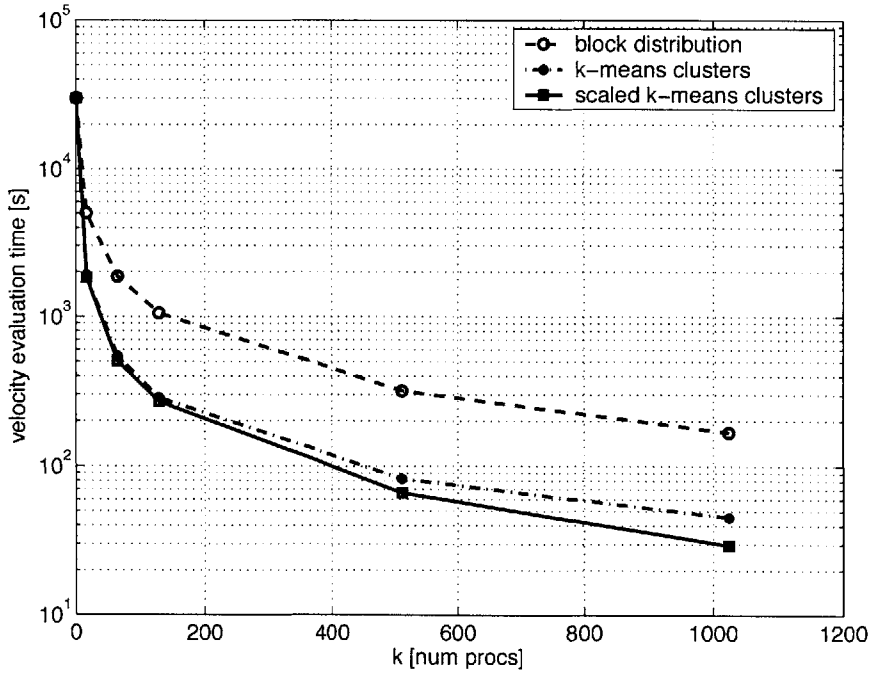


(b) Cluster partition, 7 domains.

Figure 4-5: Block and cluster partitions of vortex elements in the transverse jet at $t = 2.0$; $N = 157297$, $k = 128$.

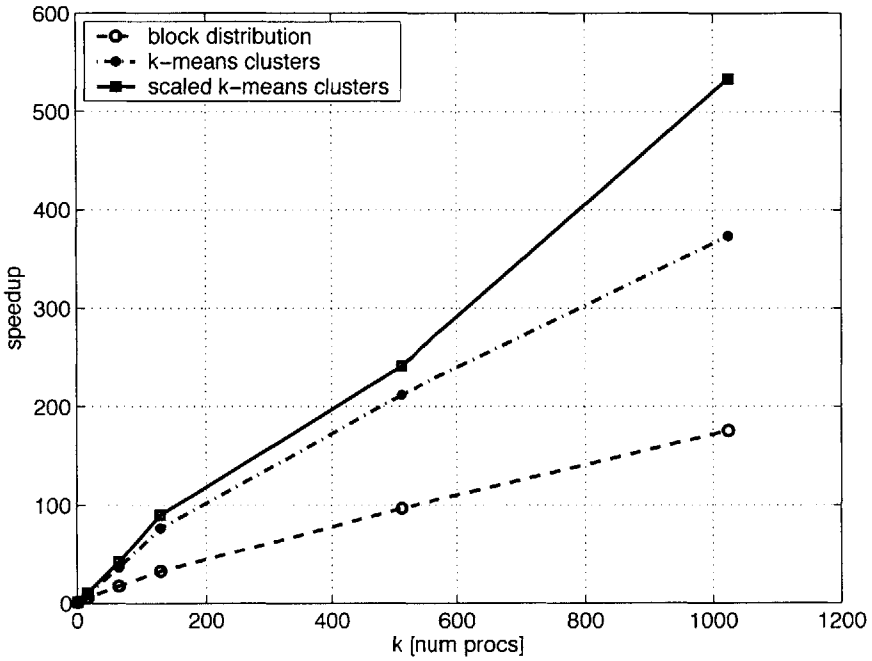


(a) $N = 261481$.

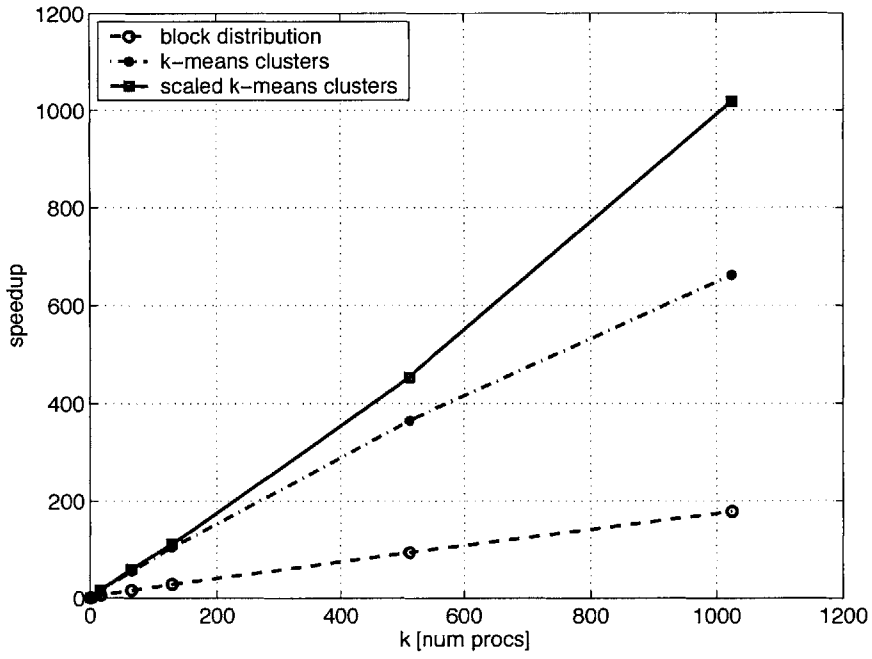


(b) $N = 1164184$.

Figure 4-6: Velocity evaluation time for the parallel treecode versus number of processors, testing three different domain decomposition schemes.

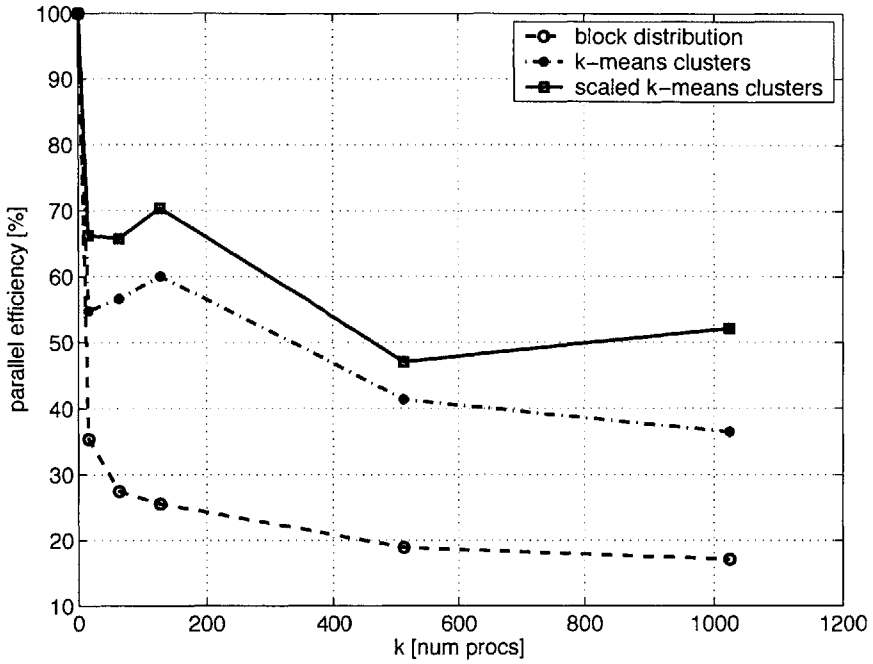


(a) $N = 261481$.

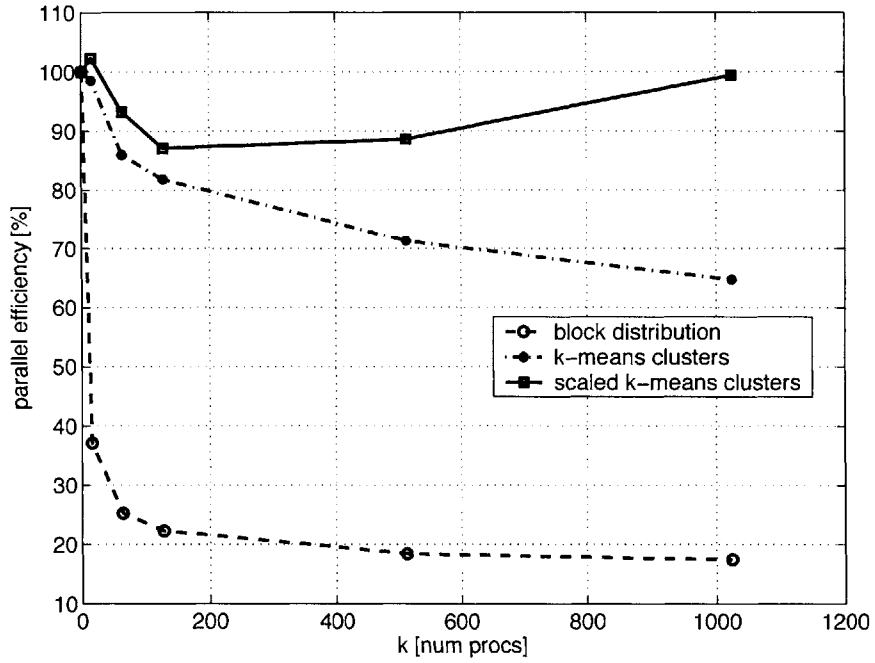


(b) $N = 1164184$.

Figure 4-7: Speedup for the parallel treecode versus number of processors, testing three different domain decomposition schemes.

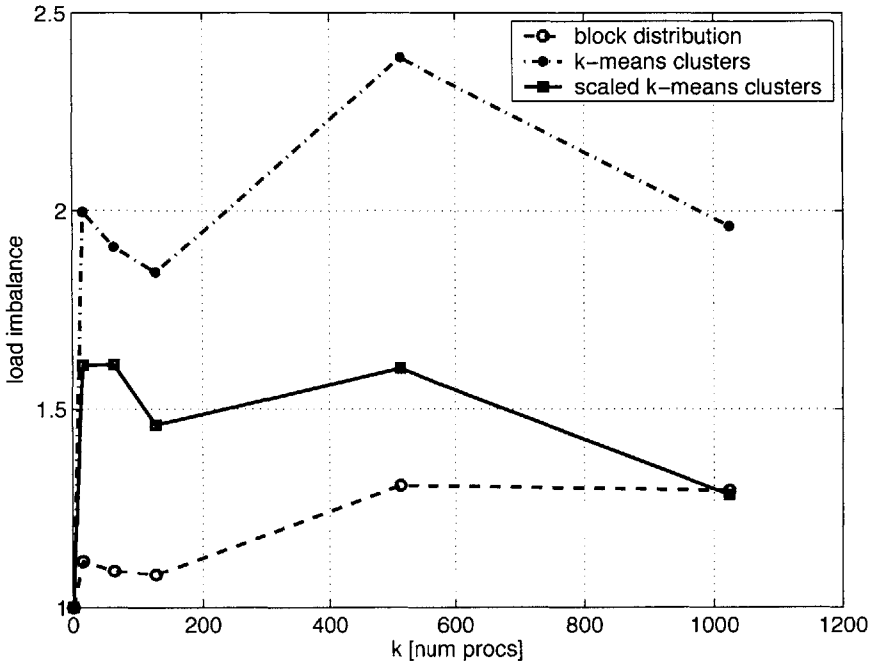


(a) $N = 261481$.

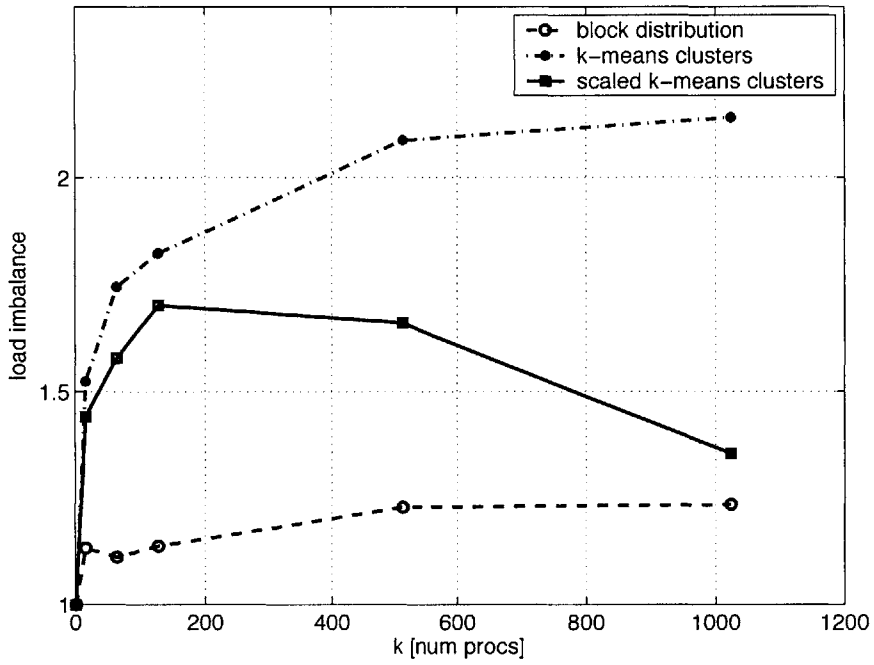


(b) $N = 1164184$.

Figure 4-8: Parallel efficiency versus number of processors, testing three different domain decomposition schemes.



(a) $N = 261481$.



(b) $N = 1164184$.

Figure 4-9: Load imbalance for each test case in Figures 4-6–4-8, testing three different domain decomposition schemes.

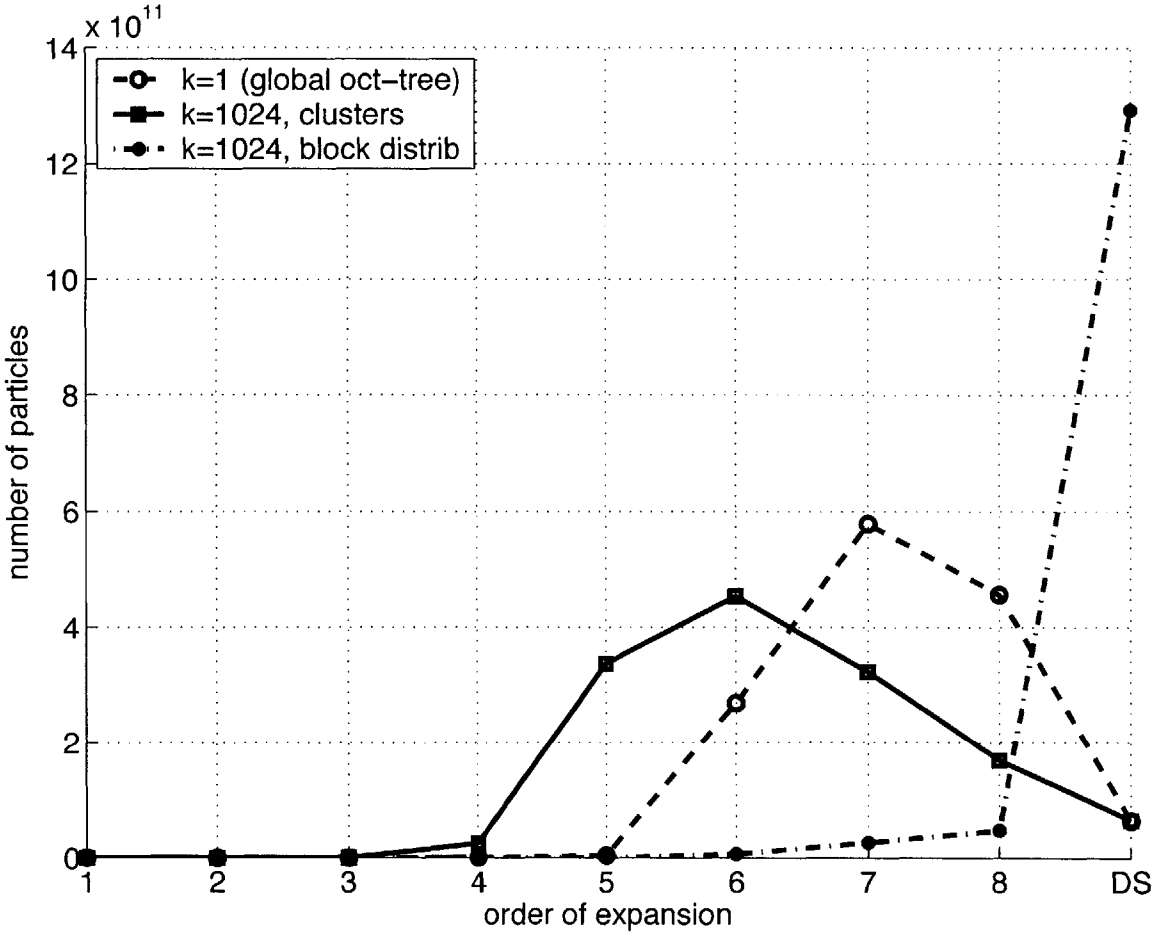
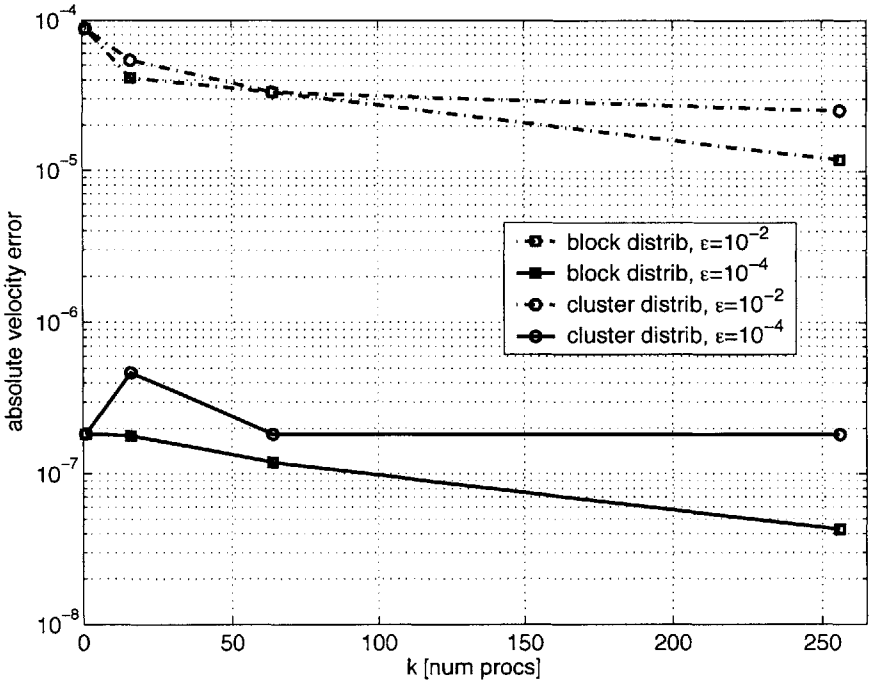
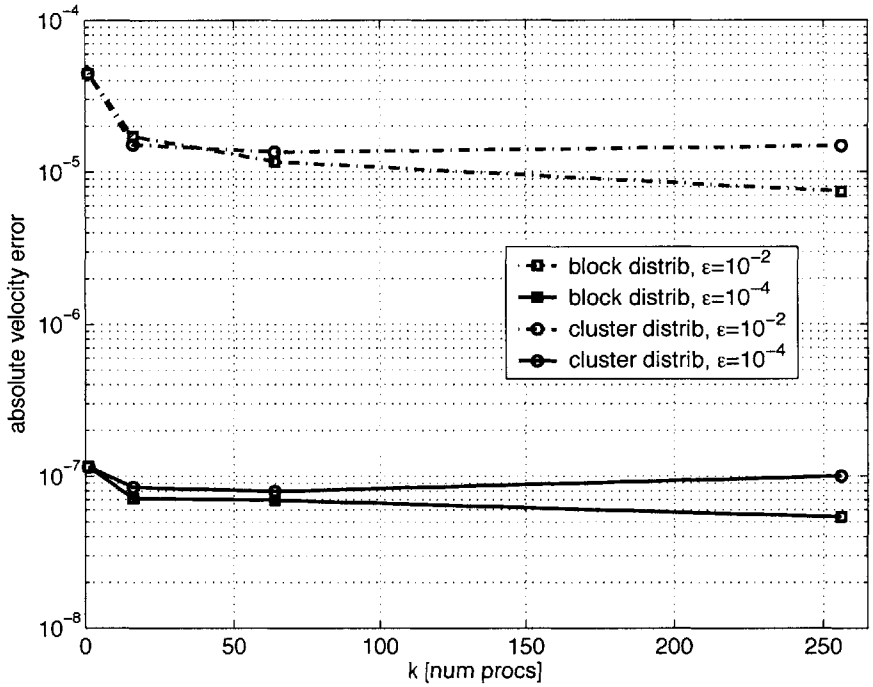


Figure 4-10: Number of particles evaluated at each order of expansion p or with direct summation, using different global partitions; $N = 1164184$ case.

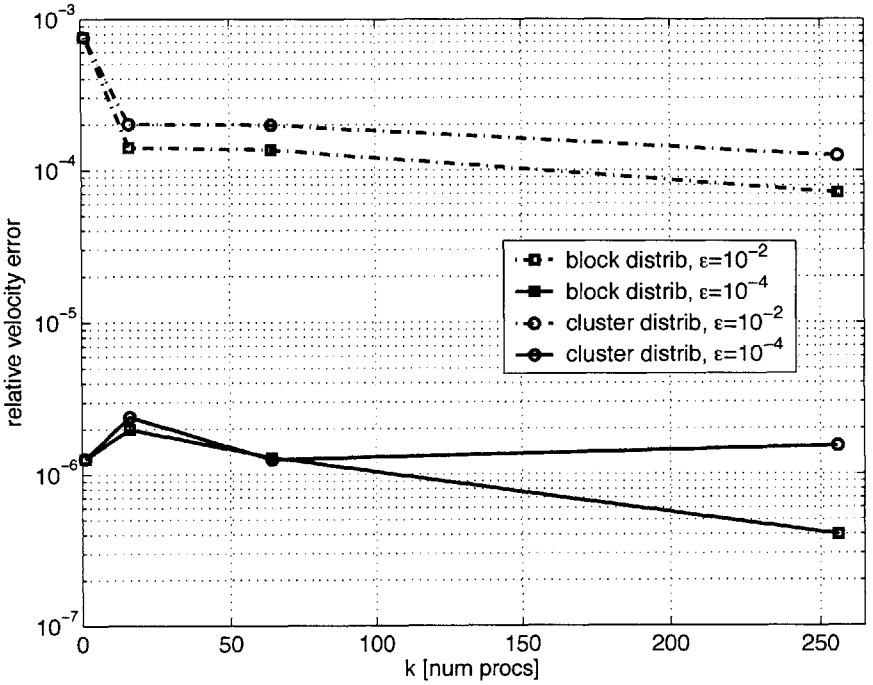


(a) $N = 102505$.

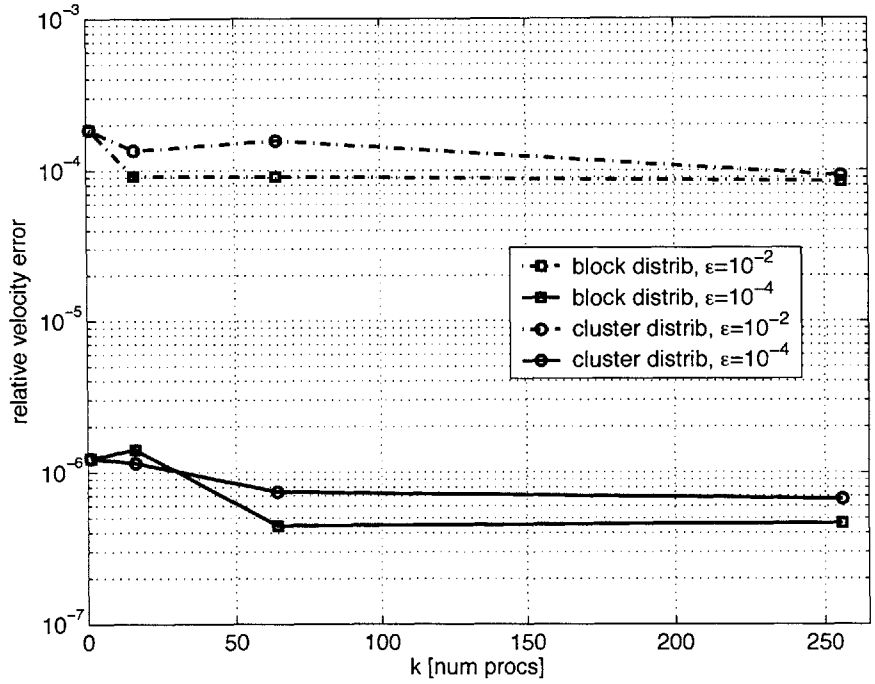


(b) $N = 261481$.

Figure 4-11: Velocity magnitude error versus number of processors, for $\epsilon = 10^{-2}$, 10^{-4}, testing block and cluster partitions.



(a) $N = 102505$.



(b) $N = 261481$.

Figure 4-12: Relative velocity magnitude error versus number of processors, for $\epsilon = 10^{-2}$, 10^{-4}, testing block and cluster partitions.

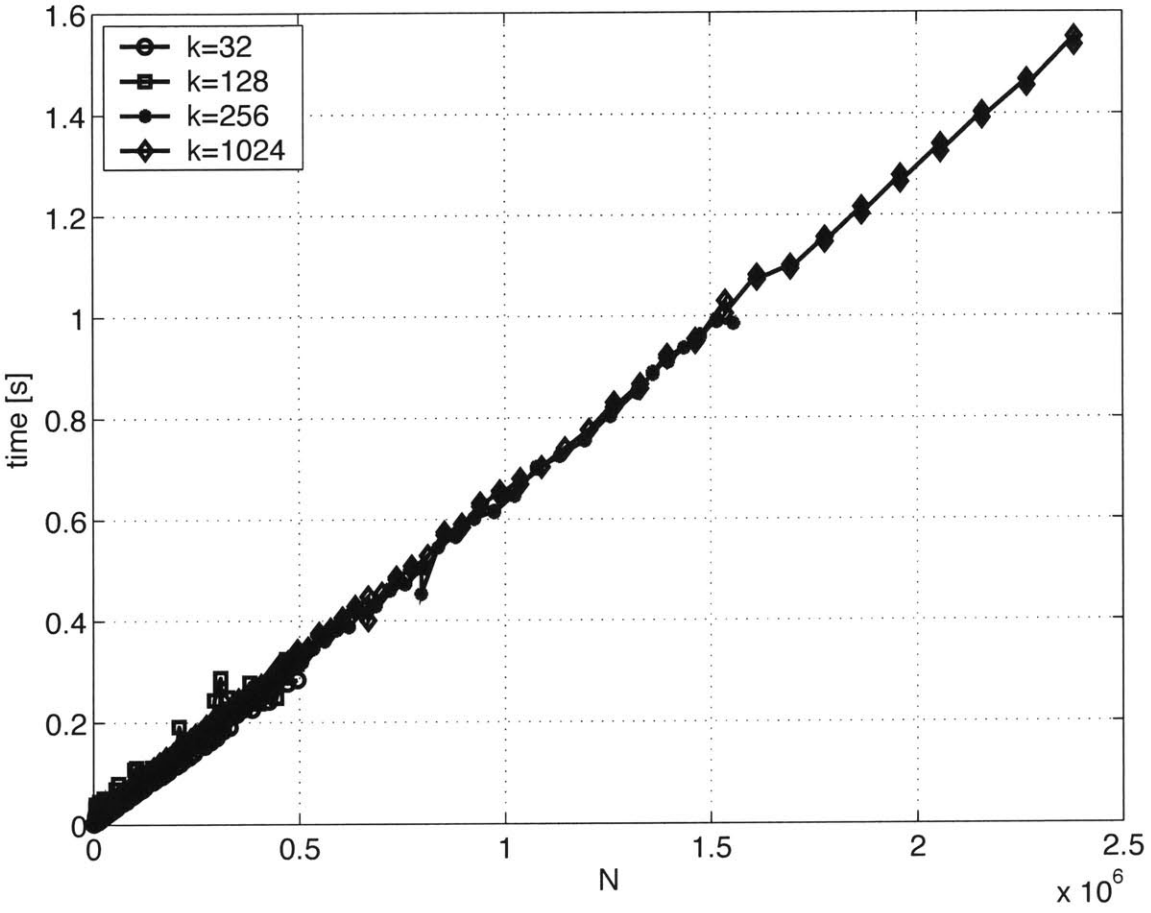
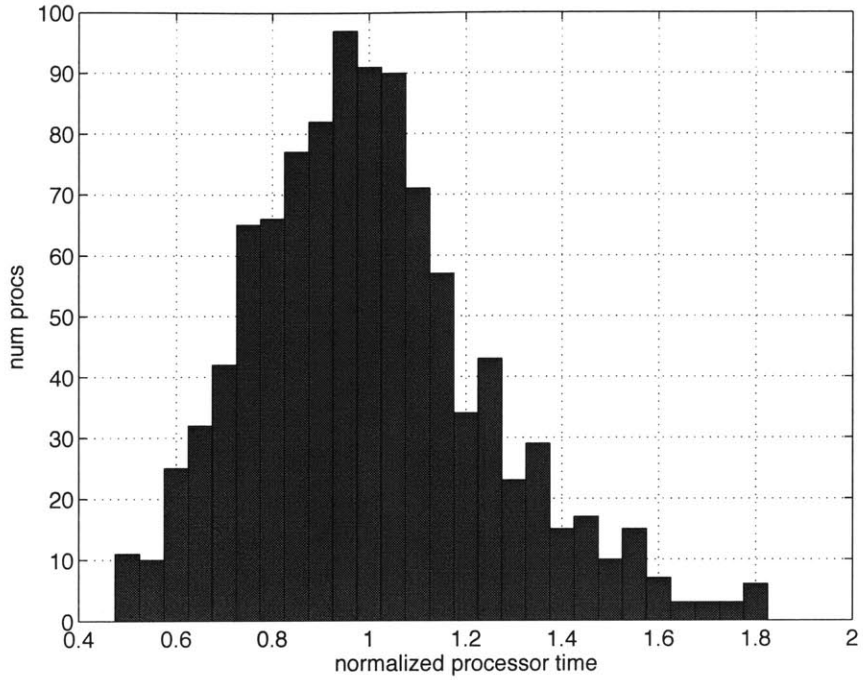
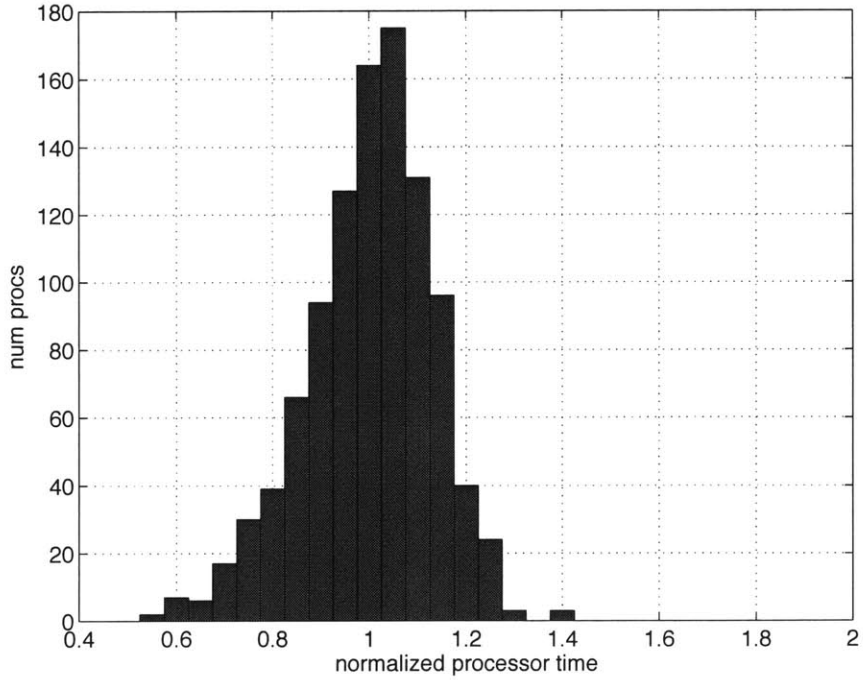


Figure 4-13: Time per iteration of parallel k-means clustering.

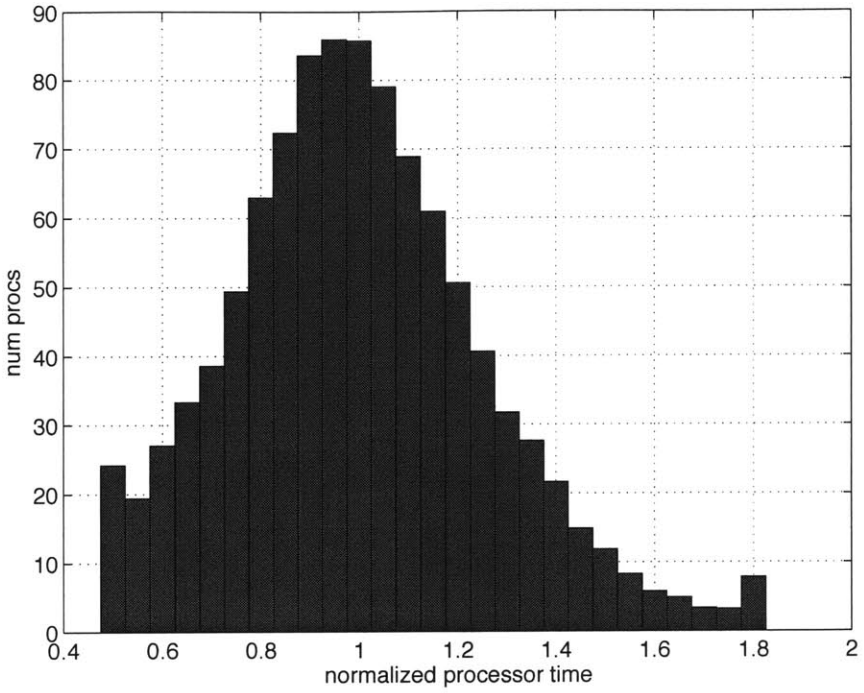


(a) Clustering, no scaling or split/merge.

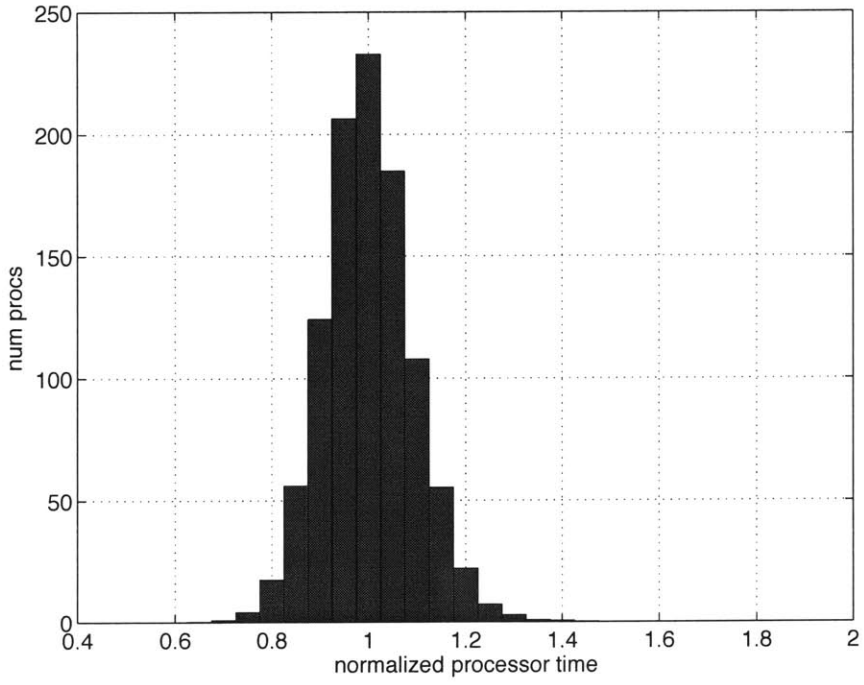


(b) Clustering, with iterated scaling and split/merge.

Figure 4-14: Distribution of normalized processor times for a test case with $N = 1164184$, $k = 1024$, drawn from a simulation of an evolving transverse jet.



(a) Clustering, no scaling or split/merge.



(b) Clustering, with iterated scaling and split/merge.

Figure 4-15: Normalized processor times in a simulation of an evolving transverse jet, averaged over 36 successive velocity evaluations, for N growing from 10^6 to 2.5×10^6.

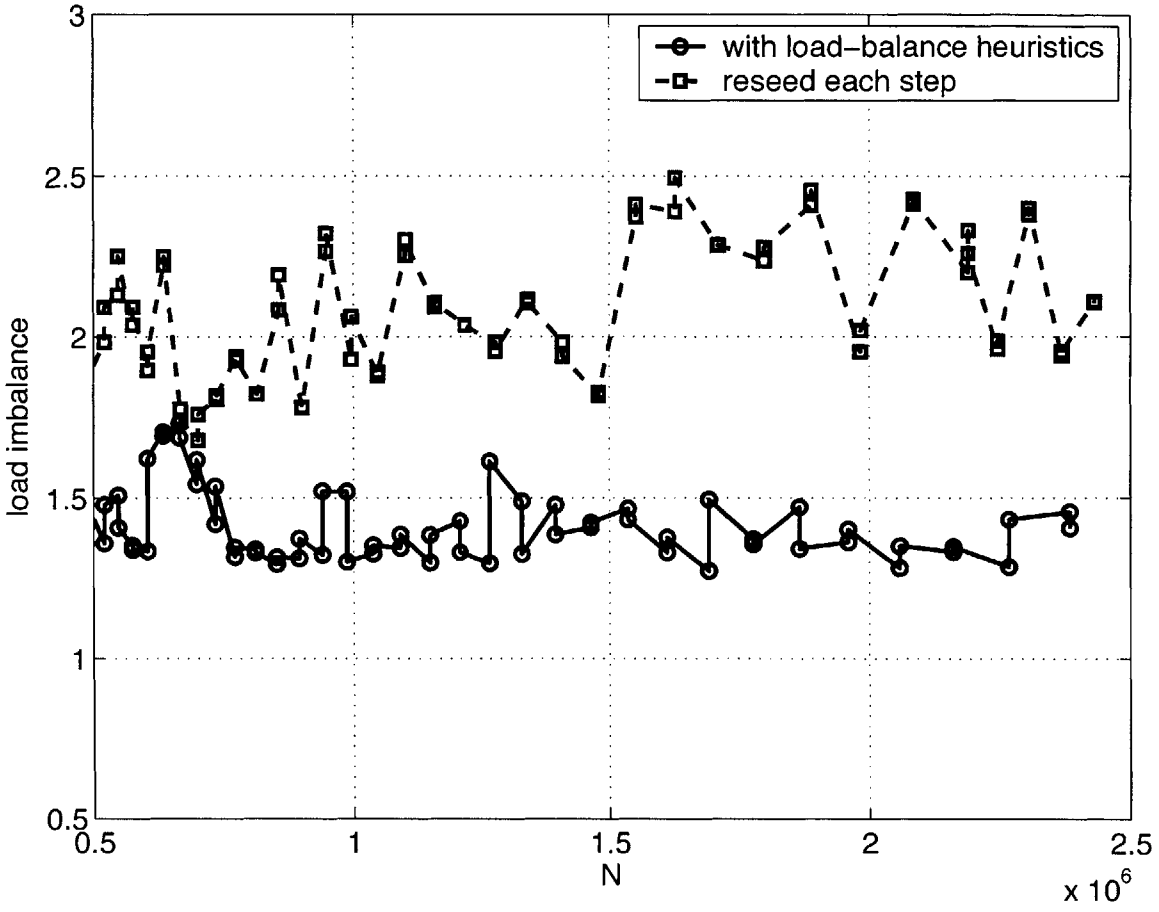


Figure 4-16: Load imbalance versus N for an evolving transverse jet, showing the effect of load-balancing heuristics.

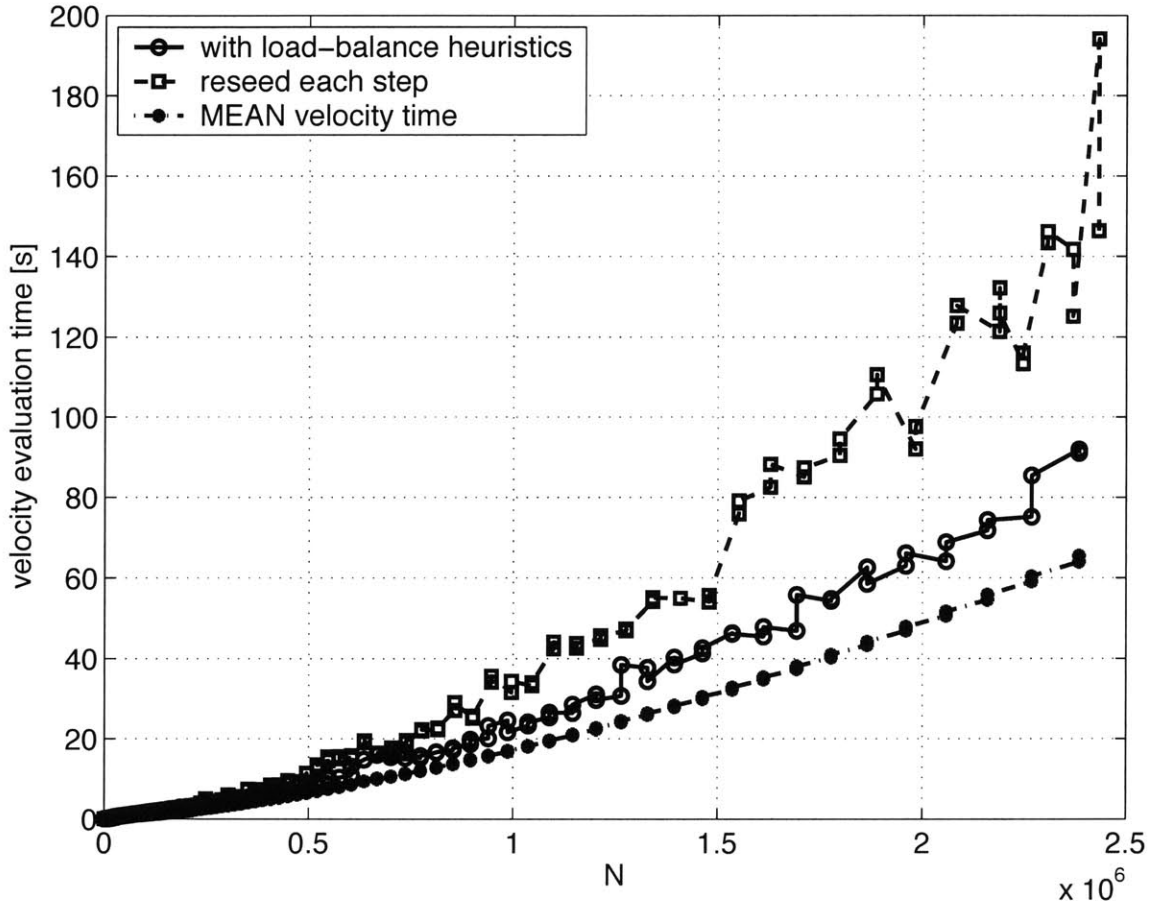


Figure 4-17: Velocity evaluation time versus N for an evolving transverse jet. Mean velocity time (shown with asterisks) represents the case of best possible load balance.

Chapter 5

Conclusions and Future Work

5.1 Vorticity dynamics in the transverse jet

This thesis has sought a detailed, mechanistic understanding of vorticity dynamics in the transverse jet. Transverse jets arise in many applications, including propulsion, effluent dispersion, oil field flows, V/STOL aerodynamics, and drug delivery. More fundamentally, they are a canonical example of a flow composed of coherent vortical structures. Our investigation has centered 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. We have also sought to characterize the impact of vortical structures on the transport and mixing characteristics of the flow.

We develop a massively parallel 3-D vortex simulation of a high-momentum transverse jet at large Reynolds number, featuring a discrete filament representation of the vorticity field with local mesh refinement to capture stretching and folding and hairpin removal to regularize the formation of small scales. A novel formulation of the vorticity flux boundary conditions, detailed in Chapter 2, carefully accounts for the interaction of channel vorticity with the jet boundary layer. We demonstrate that this interaction is essential in predicting the near-field jet trajectory and in obtaining agreement with scaling laws. Our formulation also yields analytical expressions for vortex lines in near field of the jet, which are confirmed in comparisons with numerical

simulations.

Results presented in Chapter 3 capture the key vortical structures in the transverse jet and, more importantly, reveal the mechanisms by which vortical structures evolve. Previous computational and experimental investigations of these processes have been incomplete at best, limited to low Reynolds numbers, transient early-stage dynamics, or Eulerian diagnostics of essentially Lagrangian phenomena. Our results resolve transformations of the cylindrical shear layer emanating from the nozzle. Initially dominated by azimuthal vorticity, the lee side of the shear layer is elongated axially by existing vortical structures to form arms of counter-rotating vorticity aligned with the jet trajectory. Periodic roll-up of the shear layer accompanies this deformation, creating vortex arcs on the lee and windward sides of the jet. Counter-rotating vorticity then drives lee-side vortex arcs toward the windward boundary where they form a pattern of azimuthal vorticity of alternating sign. Following the pronounced bending of the trajectory into the crossflow, we observe a catastrophic breakdown of these sparse periodic structures into a dense distribution of smaller scales, with an attendant complexity of tangled vortex filaments. Nonetheless, spatial filtering of this region reveals the underlying persistence of counter-rotating streamwise vorticity.

A range of diagnostic tools facilitates insight into the flow physics. The vorticity transformation mechanisms described above are elucidated via time-resolved tracking and continual remeshing of material lines and shear layer segments. Instantanous isosurfaces of vorticity magnitude, contoured by different vector components, reveal the overall vorticity structure. We also compute velocity spectra in the near field of the jet, then examine the mean vorticity field over single cycles of shear layer roll-up. Though the near field of the jet is dominated by deformation and periodic roll-up of the shear layer, the resulting counter-rotating vorticity is a pronounced feature of the mean field; in turn, the mean counter-rotation exerts substantial influence on the deformation of the shear layer.

We further characterize the flow by calculating maximum direct Lyapunov exponents of particle trajectories, identifying repelling material surfaces that organize finite-time mixing. We visualize these surfaces simultaneously with vortical struc-

tures in the flow. Spatially periodic repelling surfaces bound the jet fluid before it bends strongly into the crossflow; these surfaces begin and end near cores of high vorticity. Plan views of the repelling surfaces reveal an initially circular jet boundary that spreads and deforms into a kidney shape as fluid penetrates into the crossflow.

5.1.1 Future work

Our analysis of vorticity dynamics in the transverse jet, as well as the limitations and advantages of the numerical methods used to obtain our results, suggest a number of avenues for future work. We briefly outline these areas below:

1. **Continuing analysis of the flow physics:** While results in Chapter 3 have offered fundamental insights and answered long-standing questions about the physics of transverse jet, they have also exposed many interesting open issues. First among these is the cascade to small scales that accompanies jet bending into the crossflow, described in §3.2.4. It may be fruitful to seek an understanding of this breakdown in terms of vortical instabilities.

   Also, while inviscid dynamics play a dominant role in the transverse jet, the fine-scale structure of the far field raises a host of questions related to Reynolds number. (We have noted these questions at specific points in Chapter 3.) In particular, to what extent does the mean/instantaneous structure of the far-field CVP depend on Re? Is merging of opposite-signed vortex arcs on the windward side of the jet more complete at lower Re? Does Re affects the near-field deformation mechanism of the shear layer? An ongoing effort in developing diffusion and remeshing algorithms for particle simulation has enabled vortex simulations of the transverse jet at finite Reynolds number [125]; we intend to employ these new computational tools to address these questions.

   We would also like to characterize the natural dynamics (St) of the flow more completely. While we have measured Stassociated with shear layer roll-up at a particular r, different regions of the flow may exhibit different dominant frequencies; merging or pairing of vortices may contribute to these delineations, for

example. A thorough analysis of flow spectra—and of their dependence on flow parameters like r and the shear layer thickness—is absent from the literature.

Finally, longer-time simulations of the starting transverse jet will yield mean trajectories that are stationary over wider regions of the flow. Achieving a stationary state for $y/rd \gg 1$ will allow better comparison with far-field trajectory correlations, along with further investigation of the relationship between topological changes in the vorticity field and demarcation between near- and far-field regions of intermediate-asymptotic similarity.

2. **Flow actuation:** The broader context of this work, as we have emphasized in Chapter 1, is the development of actuation strategies that optimally manipulate vortical structures to affect mixing. This thesis has focused on the physics of the unforced flow rather than on actuation, yet we may make the following observation: Figures 3-27 and 3-28 show significantly different jet trajectories, yet the vorticity entering the flow in Figure 3-28 essential adds only axial and azimuthal perturbations to the primary jet vorticity. In particular, the relative magnitude of these perturbations is $O(1/r)$, approximately 10%. The resulting difference in trajectories suggests that small axial and azimuthal vorticity perturbations can serve as useful actuation inputs.

   We plan to study the physics of the actuated flow with a focus on particular actuation inputs. We will examine the role of jet pulsing—i.e., varying the frequency, amplitude, and duty cycle of $r(t)$. Pulsed actuation has been explored in some experimental studies [87, 43, 65], but the relationship between optimal pulsing and the transverse jet's preferred modes or shear layer dynamics remains unexamined. In addition, we are generalizing the results of Figure 3-28 to examine more general sets of perturbations to the vorticity field along the nozzle edge. These actuations correspond to choosing the functions $f(\theta)$ and $g(\theta)$ in (2.34), and may be realized through the use of tabs [133] or vortex generators along the nozzle edge. Continuous filament models as derived in §2.2.3 will be employed as illustrative physical models and in numerical simulation.

Non-circular nozzle shapes—elliptical orifices, for example [89, 90]—may also have a useful impact on the flow field. The effect of these shape modifications on the vorticity entering the flow is clear, suggesting an interesting examination of the ensuing vorticity dynamics.

3. **Particle methods, diffusion, and subgrid modeling:** On the purely numerical side, this work is motivating new algorithmic developments for vortex methods. As mentioned earlier, we are developing novel approaches for simultaneous diffusion and remeshing of vortex particles [125], based on vorticity redistribution [109].

   More open is the question of subgrid modeling with vortex methods—analyzing and generalizing the role of hairpin removal methods described in §2.1.4. The energy spectrum obtained with hairpin removal has not been analyzed; this fundamental question is even more interesting in light of the transverse jet's observed breakdown into small scales. How can one construct more general subgrid models for vortex particles? To what extent can these models capture the inertial range?

   Related to this are questions of optimal inviscid remeshing: What is the optimal distribution of Lagrangian computational points representing a smooth field containing a range of length scales? Work in this area should have important implications for the numerical modeling of multi-scale phenomena.

5.2 K-means clustering for hierarchical N-body interactions

The high-resolution vortex particle simulations described in Chapters 2 and 3 of this thesis present a number of computational challenges. Chief among these—in terms of computational cost—is the N-body problem of evaluating vortical velocities at every particle. Direct summation is prohibitively expensive for large N and thus we employ a hierarchical method, specifically an adaptive treecode based on Taylor expansions

of the Rosenhead-Moore kernel (2.9) in Cartesian coordinates [77]. This method must contend with an irregular, time-evolving particle distribution of non-uniform density and with the continual introduction of new vortex particles throughout the domain. Moreover, the size of our problem requires that we implement the hierarchical method in parallel on a distributed-memory machine.

This thesis introduces new algorithms, based on weighted k-means clustering, for partitioning and dynamic load balancing of N-body interactions. Good spatial partitioning is central to the performance of a hierarchical method. We demonstrate that the number of particle-cluster interactions and the order at which they are performed is directly affected by partition geometry, and that the relationship between partition geometry and computational cost is expressed in the error bounds of various cluster approximations. Weighted k-means creates well-localized convex domains and minimizes a sum of cluster moments, reducing the cost of the computing the N-body problem.

We also introduce heuristics for dynamic load balancing that are compatible with k-means; these include iterative scaling of cluster sizes and adaptive redistribution of cluster centroids.

Application of load-balanced k-means partition to the parallel hierarchical evaluation of vortical velocities results in outstanding parallel efficiencies; velocity evaluation errors, on the other hand, are maintained at or below their serial values. On a realistic particle distribution of 1.2 million particles (obtained from the transverse jet) we observe a parallel efficiency of approximately 98% on 1024 processors; naive approaches to domain decomposition show parallel efficiencies below 20% on the same problem. In simulations of the evolving transverse jet, we find that load imbalance is typically maintained below 1.5. Additionally, we find that (1) load balance provides no guarantee of parallel efficiency, and (2) with k-means partition, the parallel efficiency of the hierarchical method is better than the load imbalance would suggest.

The utility of these clustering algorithms extends beyond vortex particle methods to N-body problems in a variety of fields.

5.2.1 Future work

The performance of parallel cluster partitioning and the accompanying dynamic load balancing techniques, while very good, suggests a number of extensions and improvements.

First, one may explore alternative means of obtaining load-balanced cluster partitions. Load balancing in this problem may be cast as a constrained optimization problem. Optimal geometry—i.e., minimization of the cost function J in (4.10) with all the scalings s_k set to unity—must be subject to a constraint ensuring equal computational cost for each cluster. One way to approach this constraint is to add a penalty term of the form $\gamma(N_k - \bar{N})^2$ or $\gamma(t_k - \bar{t})^2$ to the k-means cost function. How best to solve this optimization problem—in other words, how this new term would modify (or invalidate) the k-means algorithm—will require some thought.

Second, a logical step forward for our parallel treecode is to extend the implementation to distributed data—that is, to no longer store copies of all the particle locations and weights on all processors. Doing this will add additional communication steps to the current parallel framework, but eliminate any realistic constraints that memory may place on problem size. We may be able to take advantage of the k-means partition in designing the necessary algorithms. Mapping a point in space to the domain that contains it in $\log k$ time could easily be accomplished with hierarchical clustering (see below); we also may be able to use the Voronoi tesselation of cluster centroids to do the same.

Finally, and most fundamentally, the load balance, parallel efficiency, and interaction counts reported in Chapter 4 together suggest that k-means partition should be used for more than just parallel domain decomposition. The parallel efficiency of k-means is better than its load balance would suggest because it creates a different overall partition geometry. As shown in Figure 4-10, more source particles have their influence computed at lower order in the parallel case (a hybrid k-means + local oct-tree partition) than in the serial case (an adaptive global oct-tree partition). This result is not surprising, given the geometric optimality of k-means clusters. Results

thus suggest that *hierarchical* k-means clustering could replace traditional oct-tree partitioning schemes to optimize the computational efficiency of serial hierarchical N-body algorithms. The opportunities for adaptivity in this context are enormous. The number of child clusters within each parent cell is not constrained in any way, and could be locally optimized at each node of the tree. In the parallel context, hierarchical clustering may also offer a simpler means of load-balancing by *localizing* competition among centroids for particles. The improved interaction counts in Figure 4-10 may just scratch the surface of potential gains.

Bibliography

[1] Michael R. Anderberg. *Cluster analysis for applications.* Academic Press, New York,, 1973.

[2] C. R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing*, 13(4):923–947, 1992.

[3] Christopher Anderson and Claude Greengard. On vortex methods. *SIAM Journal on Numerical Analysis*, 22:413–440, 1985.

[4] J. Andreopoulos. On the structure of jets in a cross-flow. *Journal of Fluid Mechanics*, 157:163–197, 1985.

[5] J. Andreopoulos and W. Rodi. Experimental investigation of jets in a cross-flow. *Journal of Fluid Mechanics*, 138:93–127, 1984.

[6] A. W. Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6:85–103, 1985.

[7] W. T. Ashurst and Eckart Meiburg. Three-dimensional shear layers via vortex dynamics. *Journal of Fluid Mechanics*, 189:87–116, 1988.

[8] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[9] J. E. Barnes. A modified tree code: Don't laugh, it runs. *Journal of Computational Physics*, 87(1):161–170, 1990.

[10] G. K. Batchelor. *An introduction to fluid dynamics.* Cambridge University Press, Cambridge; New York, 1973.

[11] J. T. Beale and A. Majda. Vortex methods: 1. Convergence in three dimensions. *Mathematics of Computation*, 39(159):1–27, 1982.

[12] J. T. Beale and A. Majda. Vortex methods: 2. Higher-order accuracy in two and three dimensions. *Mathematics of Computation*, 39(159):29–52, 1982.

[13] J. T. Beale and Andrew Majda. High-order accurate vortex methods with explicit velocity kernels. *Journal of Computational Physics*, 58:188–208, 1985.

[14] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation: numerical methods.* Prentice Hall, Englewood Cliffs, N.J., 1989.

[15] S. Bhatt, P. Liu, V. Fernandez, and N. Zabusky. Tree codes for vortex dynamics: Application of a programming framework. In *Workshop on Solving Irregular Problems on Parallel Machines, International Parallel Processing Symposium, Santa Barbara, CA*, 1995.

[16] A. H. Boschitsch, M. O. Fenley, and W. K. Olson. A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions. *Journal of Computational Physics*, 151(1):212–241, 1999.

[17] Léon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithm. In *Advances in Neural Information Processing Systems*, volume 7, Denver, 1995. MIT Press.

[18] J. E. Broadwell and R. E. Breidenthal. Structure and mixing of a transverse jet in incompressible-flow. *Journal of Fluid Mechanics*, 148:405–412, 1984.

[19] B. J. Cantwell. Organized motion in turbulent flow. *Annual Review of Fluid Mechanics*, 13:457–515, 1981.

[20] Y. K. Chang and A. D. Vakili. Dynamics of vortex rings in cross-flow. *Physics of Fluids*, 7(7):1583–1597, 1995.

[21] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155(2):468–498, 1999.

[22] A. J. Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57:785–796, 1973.

[23] A. J. Chorin. Hairpin removal in vortex interactions. *Journal of Computational Physics*, 91:1–21, 1990.

[24] A. J. Chorin. Hairpin removal in vortex interactions II. *Journal of Computational Physics*, 107:1–9, 1993.

[25] A. J. Chorin and O. H. Hald. Vortex renormalization in three space dimensions. *Physical Review B*, 51(17):11969–11972, 1995.

[26] A. J. Chorin and Jerrold E. Marsden. *A Mathematical Introduction to Fluid Mechanics.* Springer-Verlag, 1993.

[27] Alexandre Chorin. Microstructure, renormalization, and more efficient vortex methods. *ESAIM Proceedings: Vortex Flows and Related Numerical Methods II*, 1:1–14, 1996.

[28] Alexandre Joel Chorin. *Vorticity and turbulence.* Springer-Verlag, New York, 1994.

[29] S. L. V. Coelho and J. C. R. Hunt. The dynamics of the near-field of strong jets in crossflows. *Journal of Fluid Mechanics*, 200:95–120, 1989.

[30] L. Cortelezzi and A. R. Karagozian. On the formation of the counter-rotating vortex pair in transverse jets. *Journal of Fluid Mechanics*, 446:347–373, 2001.

[31] G. H. Cottet. A new approach for the analysis of vortex methods in two and three dimensions. *Annales De L'Institut Henri Poincare—Analyse Non Lineaire*, 5(3):227–285, 1988.

[32] George-Henri Cottet and Petros D. Koumoutsakos. *Vortex Methods: Theory and Practice.* Cambridge University Press, 2000.

[33] Carl de Boor. *A Practical Guide to Splines.* Springer-Verlag, 1978.

[34] P. Degond and S. Masgallic. The weighted particle method for convection-diffusion equations: 1. The case of an isotropic viscosity. *Mathematics of Computation*, 53(188):485–507, 1989.

[35] P. Degond and S. Masgallic. The weighted particle method for convection-diffusion equations: 2. The anisotropic case. *Mathematics of Computation*, 53(188):509–525, 1989.

[36] Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In M. J. Zaki and C.-T. Ho, editors, *Large-Scale Parallel Data Mining*, volume 1759 of *Lecture Notes in Artificial Intelligence*, pages 245–260. Springer-Verlag, 2000.

[37] Z. H. Duan and R. Krasny. An Ewald summation based multipole method. *Journal of Chemical Physics*, 113(9):3492–3495, 2000.

[38] Z. H. Duan and R. Krasny. An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *Journal of Computational Chemistry*, 22(2):184–195, 2001.

[39] J. Dubinski. A parallel tree code. *New Astronomy*, 1(2):133–147, 1997.

[40] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification.* Wiley, New York, 2001.

[41] Olivier S. Eiff and James F. Keffer. On the structures in the near-wake region of an elevated turbulent jet in a crossflow. *Journal of Fluid Mechanics*, 333:161–195, 1997.

[42] J. D. Eldredge, T. Colonius, and A. Leonard. A vortex particle method for two-dimensional compressible flow. *Journal of Computational Physics*, 179(2):371–399, 2002.

[43] A. Eroglu and R. E. Breidenthal. Structure, penetration, and mixing of pulsed jets in crossflow. *AIAA Journal*, 39(3):417–423, 2001.

[44] T. F. Fric and A. Roshko. Vortical structure in the wake of a transverse jet. *Journal of Fluid Mechanics*, 279:1–47, 1994.

[45] A. F. Ghoniem and F. S. Sherman. Grid-free simulation of diffusion using random-walk methods. *Journal of Computational Physics*, 61(1):1–37, 1985.

[46] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(2):375–389, 1977.

[47] A. Grama, V. Kumar, and A. Sameh. Scalable parallel formulations of the barnes-hut method for n-body simulations. *Parallel Computing*, 24(5-6):797–822, 1998.

[48] A. Grama, V. Sarin, and A. Sameh. Improving error bounds for multipole-based treecodes. *SIAM Journal on Scientific Computing*, 21(5):1790–1803, 2000.

[49] L. Greengard. Fast algorithms for classical physics. *Science*, 265(5174):909–914, 1994.

[50] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[51] L. Greengard and V. Rokhlin. The rapid evaluation of potential fields in three dimensions. *Lecture Notes in Mathematics*, 1360:121–141, 1988.

[52] O. Hald and V. M. Delprete. Convergence of vortex methods for Euler's equations. *Mathematics of Computation*, 32(143):791–809, 1978.

[53] O. H. Hald. Convergence of vortex methods for Euler's equations 2. *SIAM Journal on Numerical Analysis*, 16(5):726–755, 1979.

[54] O. H. Hald. Convergence of vortex methods for Euler's equations 3. *SIAM Journal on Numerical Analysis*, 24(3):538–582, 1987.

[55] G. Haller. Finding finite-time invariant manifolds in two-dimensional velocity fields. *Chaos*, 10(1):99–108, 2000.

[56] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D*, 149(4):248–277, 2001.

[57] G. Haller and G. Yuan. Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D*, 147(3-4):352–370, 2000.

[58] Ernest F. Hasselbrink and M. G. Mungal. Transverse jets and jet flames. Part 1: Scaling laws for strong transverse jets. *Journal of Fluid Mechanics*, 443:1–25, 2001.

[59] Ernest F. Hasselbrink and M. G. Mungal. Transverse jets and jet flames. Part 2: Velocity and OH field imaging. *Journal of Fluid Mechanics*, 443:27–68, 2001.

[60] B. A. Haven and M. Kurosaka. Kidney and anti-kidney vortices in crossflow jets. *Journal of Fluid Mechanics*, 352:27–64, 1997.

[61] L. Hernquist and N. Katz. TreeSPH—a unification of SPH with the hierarchical tree method. *Astrophysical Journal Supplement Series*, 70(2):419–446, 1989.

[62] T. Y. Hou and J. Lowengrub. Convergence of the point vortex method for the 3-D Euler equations. *Communications on Pure and Applied Mathematics*, 43(8):965–981, 1990.

[63] P. Huq and M. R. Dhanak. The bifurcation of circular jets in crossflow. *Physics of Fluids*, 8(3):754–763, 1996.

[64] A. K. M. F. Hussain. Coherent structures and turbulence. *Journal of Fluid Mechanics*, 173:303–356, 1986.

[65] H. Johari, M. Pacheco-Tougas, and J. C. Hermanson. Penetration and mixing of fully modulated turbulent jets in crossflow. *AIAA Journal*, 37(7):842–850, 1999.

[66] Y. Kamotani and I. Greber. Experiments on a turbulent jet in a cross flow. *AIAA Journal*, 10:1425–1429, 1972.

[67] A. R. Karagozian. An analytical model for the vorticity associated with a transverse jet. *AIAA Journal*, 24:429–436, 1986.

[68] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[69] J. F. Keffer and W. D. Baines. The round turbulent jet in a cross-wind. *Journal of Fluid Mechanics*, Vol. 15:481–496, 1962.

[70] R. M. Kelso, T. T. Lim, and A. E. Perry. An experimental study of round jets in cross-flow. *Journal of Fluid Mechanics*, 306:111–144, 1996.

[71] R. M. Kelso and A. J. Smits. Horseshoe vortex systems resulting from the interaction between a laminar boundary-layer and a transverse jet. *Physics of Fluids*, 7(1):153–158, 1995.

[72] Omar M. Knio and Ahmed F. Ghoniem. Numerical study of a three-dimensional vortex method. *Journal of Computational Physics*, 86:75–106, 1990.

[73] A. Krothapalli, L. Lourenco, and J. M. Buchlin. Separated flow upstream of a jet in a cross-flow. *AIAA Journal*, 28(3):414–420, 1990.

[74] I. Lakkis and A. F. Ghoniem. Axisymmetric vortex method for low-Mach number, diffusion-controlled combustion. *Journal of Computational Physics*, 184(2):435–475, 2003.

[75] Anthony Leonard. Computing three-dimensional incompressible flows with vortex elements. *Annual Review of Fluid Mechanics*, 17:523–559, 1985.

[76] T. T. Lim, T. H. New, and S. C. Luo. On the development of large-scale structures of a jet normal to a cross flow. *Physics of Fluids*, 13(3):770–775, 2001.

[77] Keith Lindsay and Robert Krasny. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *Journal of Computational Physics*, 172(2):879–907, 2001.

[78] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[79] Andrew Majda and Andrea L. Bertozzi. *Vorticity and incompressible flow.* Cambridge University Press, Cambridge; New York, 2001.

[80] R. J. Margason. Fifty years of jet in cross flow research. In *Computational and Experimental Assessment of Jets in Cross Flow*, volume CP-534. AGARD, 1993.

[81] Youssef M. Marzouk and Ahmed F. Ghoniem. Three-dimensional vortex filament simulation of a transverse jet. *Bulletin of the American Physical Society*, 46(10), 2001.

[82] Youssef M. Marzouk and Ahmed F. Ghoniem. Mechanism of streamwise vorticity formation in a transverse jet. In *40th Aerospace Sciences Meeting and Exhibit*, number AIAA-2002-1063. AIAA, January 2002.

[83] Youssef M. Marzouk and Ahmed F. Ghoniem. Vorticity generation mechanisms and correct boundary conditions for transverse jet simulation. In K. J. Bathe, editor, *Second MIT Conference on Computational Fluid and Solid Mechanics*, pages 1020–1023. Elsevier, 2003.

[84] Youssef M. Marzouk and Ahmed F. Ghoniem. Vorticity formulation for an actuated jet in crossflow. In *42nd Aerospace Sciences Meeting and Exhibit*, number AIAA-2004-0096. AIAA, January 2004.

[85] Youssef M. Marzouk and Ahmed F. Ghoniem. Vorticity structure and evolution in strong transverse jets. In preparation, 2004.

[86] S. Mas-Gallic. The diffusion velocity method: A deterministic way of moving the nodes for solving diffusion equations. *Transport Theory and Statistical Physics*, 31(4-6):595–605, 2002.

[87] R. T. M'Closkey, J. M. King, L. Cortelezzi, and A. R. Karagozian. The actively controlled jet in crossflow. *Journal of Fluid Mechanics*, 452:325–335, 2002.

[88] S. Narayanan, P. Barooah, and J. M. Cohen. Dynamics and control of an isolated jet in crossflow. *AIAA Journal*, 41(12):2316–2330, 2003.

[89] T. H. New, T. T. Lim, and S. C. Luo. Elliptic jets in cross-flow. *Journal of Fluid Mechanics*, 494:119–140, 2003.

[90] T. H. New, T. T. Lim, and S. C. Luo. A flow field study of an elliptic jet in cross flow using dpiv technique. *Experiments in Fluids*, 36(4):604–618, 2004.

[91] J. M. Ottino. *The kinematics of mixing: stretching, chaos, and transport.* Cambridge University Press, Cambridge; New York, NY, 1989.

[92] Manish Parashar and James C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, pages 604–613, 1996.

[93] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. In *KDD-99, Proceedings of the Fifth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, pages 277–281, 1999.

[94] A. E. Perry, R. M. Kelso, and T. T. Lim. Topological structure of a jet in a cross flow. In *Computational and Experimental Assessment of Jets in Cross Flow*, volume CP-534. AGARD, 1993.

[95] H. G. Petersen, D. Soelvason, J. W. Perram, and E. R. Smith. The very fast multipole method. *Journal of Chemical Physics*, 101(10):8870–8876, 1994.

[96] P. Ploumhans and G. S. Winckelmans. Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *Journal of Computational Physics*, 165(2):354–406, 2000.

[97] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Application to the sphere at Re = 300, 500, and 1000. *Journal of Computational Physics*, 178(2):427–463, 2002.

[98] Dhoorjaty S. Pradeep and Fazle Hussain. Core dynamics of a coherent structure: a prototypical physical-space cascade mechanism. In Julian C. R. Hunt and J. C. Vassilicos, editors, *Turbulence structure and vortex dynamics*, pages 54–82. Cambridge University Press, Cambridge, 2000.

[99] B. D. Pratte and W. D. Baines. Profiles of the round turbulent jet in a cross flow. *J. Hydraul. Div, ASCE*, 92:53–64, 1967.

[100] E. G. Puckett. Vortex methods: An introduction and survey of selected research topics. In Max D. Gunzburger and Roy A. Nicolaides, editors, *Incompressible computational fluid dynamics: trends and advances*, pages 335–407. Cambridge University Press, Cambridge; New York, 1993.

[101] F. P. Ricou and D. B. Spalding. Measurements of entraiment by axisymmetrical turbulent jets. *Journal of Fluid Mechanics*, 11:21–32, 1961.

[102] A. Rivero, J. A. Ferre, and F. Giralt. Organized motions in a jet in crossflow. *Journal of Fluid Mechanics*, 444:117–149, 2001.

[103] L. Rosenhead. The formation of vortices from a surface of discontinuity. *Proc. Royal Society A*, 134:170–192, 1931.

[104] J. K. Salmon and M. S. Warren. Skeletons from the tree-code closet. *Journal of Computational Physics*, 111(1):136–155, 1994.

[105] J. K. Salmon, M. S. Warren, and G. S. Winckelmans. Fast parallel tree codes for gravitational and fluid dynamical N-body problems. *International Journal of Supercomputer Applications and High Performance Computing*, 8(2):129–142, 1994.

[106] Hanan Samet. *The design and analysis of spatial data structures.* Addison-Wesley series in computer science. Addison-Wesley, Reading, Mass., 1990.

[107] T. Schlick, R. D. Skeel, A. T. Brunger, L. V. Kale, J. A. Board, J. Hermans, and K. Schulten. Algorithmic challenges in computational molecular biophysics. *Journal of Computational Physics*, 151(1):9–48, 1999.

[108] J. U. Schluter and T. Schonfeld. LES of jets in cross flow and its application to a gas turbine burner. *Flow Turbulence and Combustion*, 65(2):177–203, 2000.

[109] S. Shankar and L. van Dommelen. A new diffusion procedure for vortex methods. *Journal of Computational Physics*, 127(1):88–109, 1996.

[110] H. D. Simon and S. H. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, 1997.

[111] J. P. Singh, C. Holt, T. Totsuka, A. Gupta, and J. Hennessy. Load balancing and data locality in adaptive hierarchical N-body methods—barnes-hut, fast multipole, and radiosity. *Journal of Parallel and Distributed Computing*, 27(2):118–141, 1995.

[112] S. H. Smith and M. G. Mungal. Mixing, structure, and scaling of the jet in crossflow. *Journal of Fluid Mechanics*, 357:83–122, 1998.

[113] M. C. Soteriou and A. F. Ghoniem. Effects of the free-stream density ratio on free and forced spatially developing shear layers. *Physics of Fluids*, 7(8):2036–2051, 1995.

[114] M. C. Soteriou and A. F. Ghoniem. 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, 1995.

[115] M. C. Soteriou and A. F. Ghoniem. On the effects of the inlet boundary condition on the mixing and burning in reacting shear flows. *Combustion and Flame*, 112(3):404–417, 1998.

[116] James H. Strickland and Roy S. Baty. A pragmatic overview of fast multipole methods. In James Renegar, Michael Shub, and Stephen Smale, editors, *The Mathematics of Numerical Analysis: 1995 AMS-SIAM Summer Seminar in Applied Mathematics*, pages 807–830, Providence, R.I., August 1996. American Mathematical Society.

[117] J. Sucec and W. W. Bowley. Prediction of the trajectory of a turbulent jet injected into a crossflowing stream. *Journal of Fluids Engineering*, pages 667–673, 1976.

[118] R. I. Sykes, W. S. Lewellen, and S. F. Parker. On the vorticity dynamics of a turbulent jet in a cross-flow. *Journal of Fluid Mechanics*, 168:393–413, 1986.

[119] Shang-Hua Teng. Provably good partitioning and load balancing algorithms for parallel adaptive N-body simulation. *SIAM Journal on Scientific Computing*, 19(2):635–656, 1998.

[120] L. van Dommelen and E. A. Rundensteiner. Fast, adaptive summation of point forces in the two-dimensional poisson equation. *Journal of Computational Physics*, 83(1):126–147, 1989.

[121] H. Y. Wang. Short wave instability on vortex filaments. *Physical Review Letters*, 80(21):4665–4668, 1998.

[122] M. S. Warren and J. K. Salmon. A portable parallel particle program. *Computer Physics Communications*, 87(1-2):266–290, 1995.

[123] Michael S. Warren and John K. Salmon. Astrophysical N-body simulations using hierarchical tree data structures. In *Proceedings of Supercomputing '92*, pages 570–576. IEEE, 1992.

[124] Michael S. Warren and John K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Proceedings of Supercomputing '93*, pages 12–21. IEEE, 1993.

[125] D. H. Wee, Y. M. Marzouk, and A. F. Ghoniem. In preparation, 2004.

[126] J. Weiss. The dynamics of enstrophy transfer in two-dimensional hydrodynamics. *Physica D*, 48(2-3):273–294, 1991.

[127] C.H.K. Williamson, T. Leweke, and G. D. Miller. Fundamental instabilities in spatially-developing wing wakes and temporally-developing vortex pairs. In Julian C. R. Hunt and J. C. Vassilicos, editors, *Turbulence structure and vortex dynamics*, pages 83–103. Cambridge University Press, Cambridge, 2000.

[128] G. S. Winckelmans and A. Leonard. Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. *Journal of Computational Physics*, 109(2):247–273, 1993.

[129] G. S. Winckelmans, J. K. Salmon, M. S. Warren, A. Leonard, and B. Jodoin. Application of fast parallel and sequential tree codes to computing three-dimensional flows with the vortex element and boundary element methods. *ESAIM Proceedings: Vortex Flows and Related Numerical Methods II*, 1:225–240, 1996.

[130] Lexing Ying, George Biros, and Denis Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.

[131] L. L. Yuan and R. L. Street. Trajectory and entrainment of a round jet in crossflow. *Physics of Fluids*, 10(9):2323–2335, 1998.

[132] L. L. Yuan, R. L. Street, and J. H. Ferziger. Large-eddy simulations of a round jet in crossflow. *Journal of Fluid Mechanics*, 379:71–104, 1999.

[133] K. B. M. Q. Zaman and J. K. Foss. The effect of vortex generators on a jet in a cross-flow. *Physics of Fluids*, 9(1):106–114, 1997.

SS68-83