Simulation of Vortex Interactions With a Solid Wall Using Adaptive Mesh Refinement

by

Kristopher Rowe

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

One feature that is common to many fluid flows is that phenomena of interest often occur at disparate length scales, whether it be vortices interacting with a boundary layer, or shear instabilities on an internal gravity wave. It has been demonstrated in many studies that when performing computer simulations of fluid flows, one must ensure that sufficient resolution is used to capture the smallest scale features of the flow. If the smallest scale features of the flow occur in a small subset of the problem domain, however, much of the computational resources used for a simulation will be wasted where they are not needed. In order to address these kinds of problems, a class of algorithms known as adaptive mesh refinement (AMR) seek to use grid resolution only where it is needed. Upon a coarse base grid, areas of a fluid flow where small scale features occur are identified, and a hierarchy of successively finer grids is build until sufficient resolution is obtained. We give a thorough review of the adaptive mesh refinement algorithm for the incompressible Navier-Stokes equations presented in Martin, Colella, and Graves (2008) and connect their techniques to the literature for finite volume methods. The performance and scalability of their algorithm on a commodity computer cluster is studied in order to systematically choose optimal grid parameters. This algorithm is then used to perform a number of simulations of vortices interacting with a viscous boundary layer. Following Clercx and Bruneau (2006), the interaction of a vortex dipole with a solid wall is modelled: a problem which has been suggested as a difficult physical benchmark for incompressible Navier-Stokes solvers due to the resolution needed to obtain the correct behaviour for the flow. The interaction of a vortex ring with a solid wall is also simulated for a variety of Reynolds numbers. The results of these simulations are shown agree well with those seen in laboratory experiments. A loop-structured secondary vortex ring is formed which undergoes a topologically complex interaction with the initial vortex ring, ultimately leading to the breakdown and dissipation of both vortex rings. Emphasis is placed on the performance of AMR when compared to a traditional single grid model, and subsequently, the ability of AMR methods to model fluid flows using direct numerical simulation at higher Reynolds numbers than were previously possible.

Acknowledgements

Thanks to the Applied Numerical Algorithms Group at Lawrence Berkeley National Laboratory for all of their help with Chombo and for hosting my in the summer of 2013. A special thanks in particular to Hans Johansen, Dan Martin, and Dan Graves.

Thanks to Kevin Lamb for help with the editing of the thesis and for sticking with me throughout the project even when it looked like things had reached a dead end on several occasions.

Thanks to Marek Stastna for all of your help with the editing of the thesis and for many insightful discussions involving fluids, numerical analysis, and the ugly inner workings of the university.

Lastly, thanks to my wife Larissa for putting up with all of the late nights and my long absences while trying to get the thesis finished.

This research was funded in part by the NSERC Canadian Graduate Scholarship program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There has been a renewed interest in recent years in vortex interactions with a solid wall. As an example, a review article in the Annual Review of Fluid Mechanics on the "Dynamics and Instabilities of Vortex Pairs" [30] was published in January of this year. The article surveys work that has been done to date on the stability of vortex pairs in two and three dimensions. Of particular interest to the Aerospace industry is vortex pairs interacting with the ground at high velocities. A review of the structure of the wing-tip vortices that are generated by transport aircraft is given in [42] and also [13]. The structure of the wake behind subsonic transport aircraft involve a number of vortex pairs, which have instabilities in three dimensions. At airports many of these structures can persist over runways for long periods of time, creating a hazard to planes which are taking off or landing [42].

In addition to the Aerospace industry, there are those in the turbulence modelling community who have an interest in the modelling of vortical structures near solid walls. Contrary to what many fluid dynamicists first believed, the structure of turbulent boundary layers is actually quite organized, containing hairpin and other vortex structures. A review of the formation and organization of hairpin vortex structures in turbulent flows is given in [2]. To date most of the computer simulations of vortex interactions with solid walls have either been at very low Reynolds numbers or else have used Large Eddy Simulation. This is due to the significant amount of resolution needed to completely model the viscous boundary layer that forms in these types of flows. To try and overcome this obstacle, we will turn to methods which seek to apply high resolution only where it is needed in a computer simulation. First, however we will review the governing equations of fluid mechanics and the dynamics of vortex motions.

1.1 Equations of Motion

In this section we will examine the equations of motion for a constant density and viscosity fluid. These primitive variable equations can then be used to derive equations governing the dynamics of vortex motion, from which a number of immediate and important results follow. We will then analyze how to compare geometrically similar flows for their important ratios of parameters and see how this is related to the theory of boundary layers for a viscous fluid near a solid wall. The mechanism for vorticity generation in the viscous boundary layer will prove to be central to the phenomena we will study later.

1.1.1 Conservation Laws

The **Navier-Stokes equations**[1] are a set of conservation laws which describe the evolution of a *Newtonian Fluid*[2]. We will also make the assumptions that our fluid is incompressible[3] and has constant density and viscosity. The form the Navier-Stokes equations take under these assumptions is

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{1.2}$$

where $\mathbf{u}$ is the fluid velocity, p the fluid pressure per unit mass[4], and ν is the kinematic viscosity. The first equation (really three equations) expresses the conservation of momentum, and the second equation expresses the conservation of mass. In addition to these two equations, we need initial and boundary conditions to completely define our problem.

One observation about the system (1.1) - (1.2) is that it contains no evolution equation for the pressure. Additionally, the mathematical theory for the Navier-Stokes equations tells us that we cannot prescribe initial and boundary conditions for both the pressure and the velocity. We will see later that all of this creates a great deal of difficulty when trying to solve these equations numerically.

[1] See [29] for a derivation.

[2] A Newtonian fluid is one whose viscous stress tensor is linearly proportional to its rate of strain tensor.

[3] This approximation amounts to filtering out high speed sound waves which have no effect on the remaining dynamics.

[4] The density can be thought of as being set to unity, or else divided through and absorbed into the pressure term.

Boundary Conditions

The velocity and pressure on the domain boundary must be compatible with our conservation laws. By considering an infinitesimally small fluid element on a solid boundary, one can derive the following boundary conditions for the velocity [29]:

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0 \tag{1.3}$$

$$\mathbf{u} \cdot \hat{\mathbf{t}} = 0 \tag{1.4}$$

where $\hat{\mathbf{n}}$ is the vector normal to the boundary, and $\hat{\mathbf{t}}$ is any vector tangent to the boundary. The first condition says that there can be no fluid flow through the solid boundary. The second conditions arises from the inclusion of viscosity and is referred to as the **no-slip** condition as it prohibits fluid from flowing immediately adjacent to the solid boundary. Since we can form a complete coordinate system using vectors normal and tangent to the boundary, it is possible to combined these two conditions into a single condition that states

$$\mathbf{u} = 0 \tag{1.5}$$

on boundaries. Notice we have not prescribed the pressure on the boundary. Instead we will use the normal component of the momentum equation to determine the pressure boundary conditions. Taking the dot product of (1.1) with the normal vector and applying (1.3) gives

$$\nabla p \cdot \hat{\mathbf{n}} = \frac{\partial p}{\partial \hat{\mathbf{n}}} = 0. \tag{1.6}$$

Another boundary condition which we will use is the **no-shear** or symmetry boundary condition, which retains (1.3) but replaces the no slip condition with

$$\frac{\partial (\mathbf{u} \cdot \hat{\mathbf{t}})}{\partial \hat{\mathbf{n}}} = 0. \tag{1.7}$$

This says that there can be no shear production at the domain boundary. Another interpretation comes from the method of images. In this sense the boundary condition can be thought of as extending the fluid to include its mirror image reflected across the domain boundary. Notice that since we have used the same no normal flow conditions as above that the pressure boundary condition will still be given by (1.7).

Lastly, we will occasionally use **periodic** boundary conditions for three dimensional simulations. If we have a rectangular domain $[0, L] \times [0, W] \times [0, H]$, then requiring that the flow be periodic in the z-directions says that

$$\mathbf{u}(x, y, 0, t) = \mathbf{u}(x, y, H, t) \tag{1.8}$$

and equivalently for the pressure.

1.1.2 Vorticity

Vorticity is a scalar (vector) field in two (three) dimensions given by the curl of the velocity field

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} \tag{1.9}$$

and is a local measure of how quickly the fluid is rotating about an axis at a particular point in space [39]. The axis of rotation is given by the direction of $\boldsymbol{\omega}$ and the strength of the rotation is given by half of its magnitude. An important and related quantity is the **circulation** around a closed curve $\mathcal{C}$

$$\Gamma = \oint_{\mathcal{C}} \mathbf{u} \cdot d\mathbf{s}. \tag{1.10}$$

Assuming that the curve $\mathcal{C}$ is the boundary of a smooth orientable surface, $\mathcal{S}$, Stokes theorem gives the relationship between vorticity and circulation as

$$\Gamma = \iint_{\mathcal{S}} \boldsymbol{\omega} \cdot \hat{\mathbf{n}} dS. \tag{1.11}$$

Hence, the circulation is the total flux of the vorticity through the surface $\mathcal{S}$.

The integral curves of the vorticity field are know as **vortex lines**. The set of all vortex lines passing through a closed curve is referred to as **vortex tube**. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are the cross sections at two points on a vortex tube, and $\mathcal{V}$ is the volume contained between them, then it follows from the divergence theorem that

$$\iint_{\mathcal{A}_1} \boldsymbol{\omega} \cdot \hat{\mathbf{n}} dS - \iint_{\mathcal{A}_2} \cdot \hat{\mathbf{n}} dS = \iiint_{\mathcal{V}} \nabla \cdot \boldsymbol{\omega} dV = 0 \tag{1.12}$$

and hence that the flux of the vorticity is constant along the vortex tube [39]. Combining this with (1.10) gives that the circulation is constant throughout the vortex tube. Therefore, we can use the circulation as a convenient measure of the strength of a vortex tube. Finally, **vortex filament** is any vortex tube which is immediately surround by irrotational fluid.

Kelvin's Circulation Theorem And The Helmholtz Laws

Using the definitions above and the equations of motion, a number of important results can be derived. **Kelvin's Circulation Theorem** states the circulation around a material

curve is conserved following the flow for an ideal fluid (i.e. inviscid, conservative body forces).

$$\frac{D\Gamma}{Dt} = 0. \tag{1.13}$$

For a proof of this theorem see [39]. Three important results which follow from this are known as the **Helmholtz Laws**[5]:

I The strength of a vortex filament does not vary with time during the fluid motion.

II A vortex filament cannot end in the fluid: it must form a closed loop or else end at the domain boundaries.

III Fluid particles originally free of vorticity remain free of vorticity for all times.

A last lemma to the circulation theorem is that viscosity provides a source of circulation.

The Vorticity Equation

Consider the momentum equation (1.1). We can rewrite the advection term in the form

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \nabla\left(\frac{1}{2}\mathbf{u}\cdot\mathbf{u}\right) - \mathbf{u}\times(\nabla\times\mathbf{u}) \tag{1.14}$$

Taking the curl of (1.1) then gives

$$\nabla\times\frac{\partial\mathbf{u}}{\partial t} + \nabla\times\left(\nabla\left(\frac{1}{2}\mathbf{u}\cdot\mathbf{u}\right) - \mathbf{u}\times(\nabla\times\mathbf{u})\right) = -\nabla\times\nabla p + \nabla\times\nu\nabla^2\mathbf{u}$$

$$\Rightarrow \quad \frac{\partial\boldsymbol{\omega}}{\partial t} + \mathbf{0} + \nabla\times\mathbf{u}\times\boldsymbol{\omega} = \mathbf{0} + \nu\nabla^2\boldsymbol{\omega} \tag{1.15}$$

$$\tag{1.16}$$

Using the identity

$$\nabla\times\mathbf{u}\times\boldsymbol{\omega} = (\nabla\cdot\boldsymbol{\omega} + \boldsymbol{\omega}\cdot\nabla)\,\mathbf{u} - (\nabla\cdot\mathbf{u} + \mathbf{u}\cdot\nabla)\,\boldsymbol{\omega} \tag{1.17}$$

$$= (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} - (\mathbf{u}\cdot\nabla)\boldsymbol{\omega} \tag{1.18}$$

[5]Helmholtz derived these independently and before Kelvin proved the circulation theorem [39].

where the terms containing the divergence vanish due to (1.2) and the fact that the divergence of the curl of a vector field is zero, equation (1.17) can be written in the form

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega}. \tag{1.19}$$

This is known as the **vorticity equation**. Notice that the left hand side of (1.19) is the rate of change of $\boldsymbol{\omega}$ following the flow. Therefore the terms on the right hand size of (1.19) represent the production or dissipation of vorticity. We will examine more closely the physical meaning of each of these terms.

Vortex Tilting and Stretching

The first term on the right hand side of (1.19) represents vortex stretching and tilting. Consider an axisymmetric vortex oriented with the z-axis:

$$\boldsymbol{\omega} = \Omega(r, z)\hat{\mathbf{z}} \tag{1.20}$$

Assuming the fluid is inviscid, then *initially* the vorticity evolution in cylindrical coordinates is

$$\frac{D\omega_z}{Dt} = \Omega \frac{\partial u_z}{\partial z} \tag{1.21}$$

$$\frac{D\omega_r}{Dt} = \Omega \frac{\partial u_r}{\partial z}. \tag{1.22}$$

where we have linearized the first term on the right hand side of (1.19). If u_z varies with z, then this corresponds to a stretching in the z-direction. Equation (1.21) says that this **vortex stretching** results in an increase or decrease in the vorticity ω_z along the same axis. Similarly, if u_r varies with z, then equation (1.22) says that there is a resulting increase or decrease in the vorticity ω_r in the direction normal to the vortex axis. This corresponds to **vortex tilting**. Notice that in two dimensions it is impossible for there to be an variation in the z-direction, hence the stretching and tilting terms vanish. These are thus truly three dimensional effects.

Diffusion of Vorticity

In two dimensions, the stretching/tilting term in the vorticity equation (1.19) vanishes and we are left with

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega = \nu \nabla^2 \omega \tag{1.23}$$

This is an *advection-diffusion equation* and hence the role of viscosity is to diffuse vorticity. Consider a point vortex of the form

$$\omega = \Gamma_0 \, \delta(\mathbf{x}). \tag{1.24}$$

for some constant, Γ_0. The similarity solution to (1.23) with this initial condition is

$$\omega = \frac{\Gamma_0}{4\pi\nu t} \mathrm{e}^{-\frac{r^2}{4\nu t}}. \tag{1.25}$$

where r is the radial distance from the origin. Hence, after an infinitesimally small amount of time our point vortex becomes a vortex of finite extent and finite strength. This solution is known as the **Lamb-Oseen vortex** and plays an important role in the study of vortex dynamics [39]. One can readily show from (1.25) that the circulation of the Lamb-Oseen vortex around a circle of radius r centred at the origin is

$$\Gamma = \Gamma_0(1 - \mathrm{e}^{-\frac{r^2}{4\nu t}}) \tag{1.26}$$

and the tangential velocity along this circle is given by

$$u_\theta = \frac{\Gamma}{2\pi r} = \frac{\Gamma_0}{2\pi r}\left(1 - \mathrm{e}^{-\frac{r^2}{4\nu t}}\right). \tag{1.27}$$

Kinetic Energy and Enstrophy

The **total kinetic energy** *per unit mass* of a velocity field $\mathbf{u}$ is the quantity

$$K(t) = \iiint \frac{1}{2}\|\mathbf{u}\|^2 dV \tag{1.28}$$

and is a measure of the amount of motion present within the fluid. For unforced, constant density, viscous flows, the total kinetic energy of a flow is always decreasing. Additionally, the rate at which the total kinetic energy decreases can tell us something about how efficiently viscous forces are acting on the flow.

An analogous quantity for the vorticity field $\boldsymbol{\omega}$ is the **total enstrophy** given by

$$\Omega(t) = \iiint \frac{1}{2}\|\boldsymbol{\omega}\|^2 dV. \tag{1.29}$$

This is a measure of the amount of energy present in the form of vortical motions in the fluid. While enstrophy is dissipated by viscous forces, it is also produced by the stretching and intensification of large scale vortices. Additionally, the rate of dissipation of total kinetic energy is proportional to the total enstrophy. We will use both the total kinetic energy and total enstrophy to characterize and compare different flows.

1.1.3 Dimensional Analysis

A fundamental concept in the study of fluid dynamics is that of **dynamic similarity**. Loosely dynamic similarity is the idea that geometrically similar fluid flows with the same boundary conditions behave in the same way. In a more technical sense, appealing to the Buckingham Pi Theorem, we know that since there are a finite number of dimensional variables and parameters involved in a flow, and there are a finite number of dimensional units, then there can only be a finite number of independent dimensionless variables and parameters [12]. These dimensionless variables are formed using the products of powers of our dimensional variables.

For example, suppose that a large ship of length L can travel at a maximum speed U in the open ocean, and that water has a kinematic viscosity of ν. Since L has units of length, $\mathcal{L}$, and U has units of length per time, $\mathcal{L}\mathcal{T}^{-1}$, and the viscosity has units $\mathcal{L}^2\mathcal{T}^{-1}$, the only *dimensionless* combination of these variables that we can form is

$$Re = \frac{UL}{\nu}. \tag{1.30}$$

This is called the **Reynolds Number** and it is one of the most important dimensionless parameters in the study of fluid mechanics.

Suppose that we wish to model the viscous drag on the large ship by towing a small model ship in a wave tank in a laboratory. If we want to ensure that the dynamics are the same as for the large ship then we will need to ensure the laboratory flow has the same Reynolds number. Since the scale model ship has a much smaller length, and the water has the same viscosity, then the Reynolds number tells us we will need to tow the model ship at a much fast velocity for the dynamics to be the same.

Another useful purpose of dimensionless parameters is to classify different regimes for a given fluid flow. For example, in the study of the flow of fluids around a circular cylinder, the flow has fundamentally different behaviour for different Reynolds numbers. For $Re < 4$ the flow around the cylinder is smooth and the dynamics are simply described. For $80 < RE < 200$, a trail of vortices known as the *Von Karmen vortex street* form an oscillatory pattern downwind from the cylinder. Finally, for $Re \gg 200$, the wake behind the cylinder becomes very chaotic.

In general, fluid flows at lower Reynolds numbers tend to be smooth and fluid parcels follow parallel paths which do not intersect. This is known as **laminar flow**. For large Reynolds numbers, fluid flow tends to be very disorganized and there is an appearance of many small scale structures. This is known **turbulent flow**.

The terms "low Reynolds number" and "high Reynolds number" are often used loosely in the literature and there is not always a hard dynamics cutoff with which to classify the point at which a flow changes regimes. This is further complicated when a fluid flow involves multiple length and time-scales which one could choose to define the Reynolds numbers. As a result, one must be careful when comparing the results of different studies and the claims made therein. Most importantly, whenever it is possible to define different Reynolds numbers using different length and timescales it must be understood how to convert between them so that flows can be correctly categorized.

Reynolds Numbers for Vortex Dynamics

In the study of vortex dynamics, there are a number of different Reynolds number which can be define. We will now examine some of them and then show how they are related for certain kinds of flows which will be studied later. When defining all of the Reynolds numbers we will assume that we are concerned with a vortex filament of initial diameter $2R_0$ and circulation Γ_0, whose initial translational velocity is U_s, in water with kinematic viscosity ν.

The **translational Reynolds number** of the vortex filament can be defined by taking the characteristic velocity to be the translational velocity of the filament, and the characteristic length scale to be its diameter. Then

$$Re_{\text{tr}} = \frac{2U_s R_0}{\nu} \tag{1.31}$$

In a laboratory setting, this is the easiest Reynolds number to use when categorizing flows. If for example the apparatus used generates vortices of a fixed length and a municipal water source is used, then to increase the translational Reynolds number, one must increase the velocity with which generated vortices travel. In a computational setting, one can fix the length and translation speed of the prescribed initial conditions and then vary ν to change the translation Reynolds number. This will ensure that phenomena of interest will occur at roughly the same physical time in each case, making them easier to track.

The second Reynolds number which is commonly used in vortex dynamics is the **vortex or circulation Reynolds number**. As mentioned previously, the circulation of a vortex filament can be used to characterize the strength of the flow around it. Since circulation has units of $\mathcal{L}^2\mathcal{T}^{-1}$, then

$$Re_\Gamma = \frac{\Gamma_0}{\nu}. \tag{1.32}$$

In a laboratory setting, this Reynolds number is used less commonly because it is difficult to measure the circulation of a vortex filament. In a computational setting it is seen often as the initial condition can be prescribed exactly, and circulation can be calculated easily using (1.11).

Lastly, the **bulk or integral-scale Reynolds number** is defined by using the root-mean-square of the velocity field over the entire domain, $\mathcal{D}$,

$$U_{\text{rms}} = \sqrt{\frac{1}{Vol(\mathcal{D})} \iiint_{\mathcal{D}} \|\mathbf{u}\|^2 \, dV} \tag{1.33}$$

as well as the domain length, L_D to obtain

$$Re_{\mathcal{I}} = \frac{U_{\text{rms}} L_D}{\nu} \tag{1.34}$$

Consider a scenario in two dimensions where two Lamb-Oseen vortices, (1.25), of equal strength and opposite circulation, and initial radius a, are placed with their centres a distance b apart. The vortices will form a **vortex dipole** which translates in a direction perpendicular to the line which their two centres lie on. It can be shown [37] that if $\frac{a}{b} < \frac{1}{2}$ then in the inviscid limit the dipole will translate with a speed of

$$U_s = \frac{\Gamma_0}{2\pi b}. \tag{1.35}$$

This leads to a translational Reynolds number of

$$Re_{\text{tr}} = \frac{2U_s R_0}{\nu} = \frac{a\Gamma_0}{2\pi b\nu}. \tag{1.36}$$

To compare this to the vortex Reynolds number, we can compute the ratio

$$\frac{Re_\Gamma}{Re_{\text{tr}}} = \frac{2\pi b}{a}. \tag{1.37}$$

Given the assumption that $\frac{a}{b} < \frac{1}{2}$, this implies that $Re_\Gamma > 4\pi Re_{\text{tr}}$ for the translating Lamb-Oseen dipole, i.e. there is an order of magnitude difference between the vortex Reynolds number and the translational Reynolds number for this system. A similar situation will arise later when we study vortex rings.

The Viscous Boundary Layer

For large Reynolds numbers, it can be shown through scaling analysis [29] that the effects of viscous stress are negligible throughout the fluid flow except at solid boundaries. At solid boundaries viscous effects become very important in a thin layer known as the **viscous boundary layer**. The thickness, δ, of this layer varies like

$$\delta \sim \frac{1}{\sqrt{Re}} \tag{1.38}$$

which implies that it will become increasingly thin as the Reynolds number increases. The viscous boundary layer plays an important role in the simulation of vortex-wall interactions and resolving it in computer simulations will require significant computational resources.

1.2 Numerical Methods for Partial Differential Equations

While some exact solutions to the Navier-Stokes equations exist in special cases, a general solution will probably never be known. In order to make any progress in the modelling of vortex interactions with a wall we will need to find approximate solutions to the Navier-Stokes equations using sophisticated numerical analysis techniques. The solution domain for a problem is discretized onto a computational grid and the solution is advanced forwards in time using an iterative algorithm. Numerical solutions are tested for convergence and compared to physical experiments to ensure their validity and accuracy.

1.2.1 Adaptive Mesh Refinement

While many classical numerical methods for PDEs use a single uniform grid over the entire computational domain, if phenomena of interest occupy a small subset of the domain, then much of this resolution will be wasted where it is not needed. Adaptive mesh refinement (AMR) techniques seek to increase the resolution of the computational domain only where it is needed. Beginning with a coarse base mesh, cells are tagged for refinement where it is required. The resolution can be increased in a number of ways, some of which included subdividing a tagged grid cell locally, or overlaying the coarse grid with a finer computational grid. The previous method is known as **cell-based** AMR while the latter is known as **structured** AMR. In this work **block structured** AMR, which is defined by a hierarchy of logically rectangular grids, will be used.

Chombo: Software for Adaptive Solutions to Partial Differential Equations

Developing software for PDE based scientific applications can be a daunting task, one which is even further complicated by the use of AMR. It is good practice therefore to use existing software when possible to save on human and financial resources. Further, if software is open source then it can often be customized to meet the demands of a particular application[6]. Chombo [1] is a software library for implementing finite difference and finite volume methods for the solution of partial differential equations using block-structured adaptive mesh refinement. Developed by the Applied Numerical Algorithms Group (ANAG) at Lawrence Berkeley National Laboratory, Chombo is written in C++ and Fortran, and uses the MPI framework for parallelism. A bit mapping data structure is used to manage the grid hierarchy and extensive use of C++ templating allows for customization. Chombo has proven to be highly scalable to over 10^5 cores and has been used to develop a wide variety of scientific applications. Additionally, a number of already developed applications come with the standard Chombo repository. In the present work the AMRINS code developed by Dan Martin of ANAG is improved upon and used extensively.

Other AMR Software

There are a number of other AMR software libraries which are freely available. They vary widely in their complexity and ease of use. We list a sampling of them below.

- **SAMRAI**: Structured Adaptive Mesh Refinement Software[27]. This library was developed at Lawrence Livermore National Laboratory. It is intended to be used for large scale multi-physics simulations. The library is built using C++ and MPI, and uses HDF5 for I/O. The developers of SAMRAI claim it scales to over 10^5 cores. A few of the application SAMRAI has been used for include supersonic flow, magnetohydrodynamics, and multiphase flows.

- **Gerris**: A comprehensive software package developed by Stéphane Popinet at Université Pierre et Marie Curie which includes both libraries for AMR as well as complete applications [38]. The incompressible Navier-Stokes solver in Gerris uses the same finite volume method as [7]. In addition to AMR, Gerris can also handle complicated domain boundaries using an embedded boundary type method and is parallelized through the MPI framework. The type of AMR used by Gerris is cell-based, and uses quadtree or octree data structures. There is also a free surface algorithm

[6] Barring any licensing restrictions.

in the Navier-Stokes solver. A few examples of the application Gerris has been used for are Rayleigh-Taylor instabilities, flow around an aerofoil, and tsunami run-up modelling on a three-dimensional beach.

- **Paramesh**: A library from Druxel Univeristy and NASA Goddard Space Flight Center to handle the grid hierarchy and operations for AMR applications. Built using Fortran 90 and C, Paramesh uses the MPI library to handle parallelism. Paramesh uses block structured AMR and quadtree or octree data structures. A few examples of applications Paramesh has been used for are supersonic flows and magnetohydrodynamics. Unfortunately Paramesh is no longer has technical support and is no longer funded.

- **Carpet**: An AMR library for the Cactus framework for the development of PDE based scientific applications. Carpet has options to develop applications in Fortan 77 and 90, as well as C, and C++. Parallelism is handled using the MPI framework and data I/O using HDF5.

- **AMRClaw**: An AMR library developed by Rene Leveque's group to work with CLAWPACK. The user must write their problem in a standard conservation law form and provide a Riemann solver to CLAWPACK. The algorithm used is Leveque's wave-propagation finite volume formulation. The developers of AMRClaw claim that it scales to over 10^5 cores.

1.3 High Performance Computing

In modern scientific applications, the resolution needed to perform computer simulations of interest is often so high that the simulations cannot be carried out on a single core[7]; thus, parallel processing has become a fact of life for the modern scientist. **High Performance Computing (HPC)** refers to the use of parallel processing to run advanced software applications more quickly, efficiently, and reliably that could be done on a single core computer. In previous decades high performance computing was only carried out by the largest and most well funded of research groups with access to large super computers. With the advent of inexpensive multicore processors, however, high performance

[7]We will refer to the number of *cores* used for a particular simulation as this is the most precise terminology and will lead to the least confusion. While colloquially one often uses the term processor in proxy for the term core, a given processor might have multiple cores so this usage is ultimately incorrect.

computing has become accessible to even the most humble of research groups. Additionally modern computing conglomerates such as *SciNet* and *SharcNet* facilitate collaboration and allow various universities to combine their computing resources and provide access to larger computer clusters. It is therefore necessary to understand what high performance computing can and cannot do for us and the various metrics we can use to measure how well our software is performing in an HPC environment.

1.3.1 Parallel Performance Metrics

The first and most intuitive metric when evaluating the parallel performance of software is the wall-clock time. The **Wall-Clock Time**, T_c is the amount of physical time that has elapsed during the execution of our parallel software on C cores. This is the metric we often care about the most as we would like our simulation data as quickly as possible so that we can conduct analysis. Simply using a larger number of cores to get results faster is not necessarily a prudent use of computing resources as using twice as many cores rarely results in a simulation finishing in half the time.

In light of this, the next metric that is useful to define is parallel speedup. The **Speedup**, S_c, obtained by using C cores is the ratio of wall-clock times for serial and parallel execution of the code and is given by

$$S_c = \frac{T_1}{T_c}.$$

In a perfect world the speed up of our code would be exactly the number of cores that we were using to run our software. This is called linear or *perfect* speedup. In practice however our speedup will often look like Figure 1.1 as there are parts of our code that cause bottlenecks. When it is not possible or feasible to run a computer simulation on a single core, we can calculate the speedup obtained by running on C cores over C' cores via the **Relative Speedup**

$$S_C^{C'} = \frac{T_{C'}}{T_C}.$$

Another useful time scale to define in addition to wall-clock time is the **CPU Time**. This is the sum total of the time used across all cores to perform a computer simulation and is simply given by the number of cores times the wall-clock time, CT_C. From this we can define the **Parallel Efficiency**, E_C of using C cores, which is the ratio of cpu times for serial and parallel execution of our code

$$E_C = \frac{T_1}{CT_C}.$$

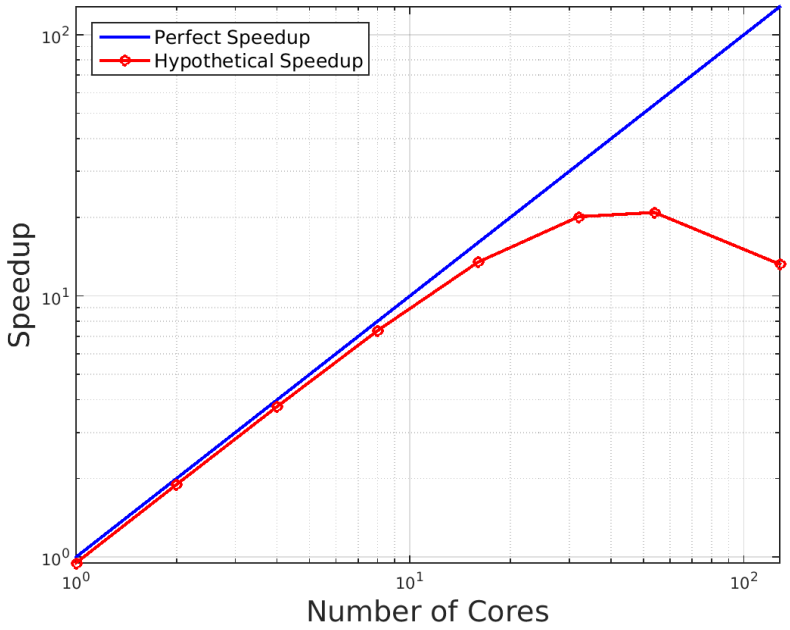

Figure 1.1: Hypothetical speedup graph (red) plotted beside the line of perfect speedup (blue).

If our code could achieve perfect speedup, then it would have an efficiency of $E_C = 1$ for all numbers of cores. In practice however our efficiency often decreases monotonically to zero for large numbers of cores. Analogously we can define the **Relative Efficiency**, $E_C^{C'}$ of using C cores over C' cores to be the ratio of the cpu times

$$E_C^{C'} = \frac{C'T_{C'}}{CT_C}.$$

1.4 Computer Systems

Three computer systems were used at various times to carry out the computer simulations that will be presented.

- **Orca**: One of the HPC clusters maintained by SHARCNET. The 32 "kglamb" nodes are composed of two 8 core, 2.7 GHz Intel E5-2690 (first generation) processors, with 20MB cache, and either 64 GB or 128 GB of memory per node. This equals a total

of 512 available cores. The interconnect is QDR InfiniBand. The Intel v12.1.3 C++ and Fortran compilers were used on this system, along with the OpenMPI v1.6.2 MPI library, which was compiled with the same compilers. HDF5 v1.8.11 was used for file I/O.

- **Hood**: One of the clusters owned by the fluids group in the Applied Mathematics Department at the University of Waterloo. Hood is composed of 4 NUMA nodes, each with an 8 core, 2.4 GHz Intel Xenon E5-4640 (first generation) processor and 88GB of memory. This equals a total of 32 available cores. The Intel v15.0.0 C++ and Fortran compilers were used on this system, along with the OpenMPI v1.8.4 MPI library, which was compiled with the same compilers. HDF5 v1.8.13 was used for file I/O on this system.

- **Boogaloo**: A desktop tower owned by professor Marek Stastna in the Applied Math Department at the University of Waterloo. Boogaloo is composed of 1 node with two 6 core, 2.6 GHz Intel E5-2630 processors and 98 GB of memory. With hyperthreading enabled this equals a total of 24 available virtual processors. The GNU v4.8.4 C++ and Fortran compilers were used, along with the OpenMPI v1.6.5 MPI library, which was compiled with the same compilers. HDF5 v1.8.13 was used for file I/O on this system.

1.5 Thesis Organization and Contributions

The remainder of the thesis is divided into two major parts. Chapter 2 gives an extensive review of the algorithm used by the AMRINS code. The projection method of [7] is reviewed in detail and tied into the broader context of finite volume methods. The details of the block-structured AMR used by Chombo are reviewed and then all of the parts are tied together to give a complete AMR algorithm for the incompressible Navier-Stokes equations.

In Chapter 3, the results of three different sets of computer simulations are presented. First, a series of tests is conducted to examine the performance of the AMRINS code and to determine optimal grid parameters for the AMR scheme. In the second section, a series of experiments involving the interaction of a two dimensional vortex dipole with a solid wall are simulated. Simulations are conducted over a wide range of Reynolds numbers and these results are compared to benchmark values from the literature. In the last section of this chapter, the interaction of a three dimensional vortex ring with a solid wall are simulated.

The last chapter of the thesis gives a brief overview of a number of potential paths the research presented here could take in the future.

The author's contributions to the field which are contained within this thesis include, but are not limited to:

- Significant scale and performance analysis of the Chombo AMRINS code on a commodity cluster system which lead to a number of code improvements. These include
  - The removal of crippling memory leaks
  - Changes to the multigrid algorithm and multigrid parameter tuning to ensure robust convergence under a variety of circumstances
  - Analysis of optimal grid generation parameters
- The benchmarking of the Chombo AMRINS code against the difficult Clercx and Bruneau vortex dipole problem.
- The simulation of the Clercx and Bruneau dipole problem to higher Reynolds numbers than have been done previously.
- The simulation of a vortex ring wall interaction to higher Reynolds numbers than have been possible previously.
- Analysis of the kinetic energy and enstrophy time-series, as well as the bottom shear stress of a vortex ring wall interaction.

All of these results are novel and warrant publication in a peer-reviewed academic journal.

Chapter 2

Methods

2.1 Introduction

In a paper published in 1989, Bell, Colella, and Glaz (BCG89) [7] developed a second-order finite volume method for the constant density incompressible Navier-Stokes equations. Using Colella's unsplit second order MUSCL scheme and a projection method based upon the Helmholtz Decomposition Theorem, their method was more robust than many other algorithms at the time. While this method would be improved upon in various ways in subsequent publications, the fundamental concepts in the algorithm remain the same; hence their original 1989 paper has had a lasting impact on the field.

We will now examine the development and extension of the BCG89 algorithm through to the version used by Martin, Colella, Graves [32], which uses adaptive mesh refinement in three dimensions. First we examine Colella's unsplit Godunov method for the advection terms. This will be followed by a brief treatment of the viscous terms. Next we will develop the theory of projection methods based upon the Helmholtz Decomposition Theorem. After using these pieces to solve the Navier Stokes equations on a single grid, we will consider the Berger-Oliger time-stepping algorithm for hyperbolic problems using adaptive mesh refinement. Finally, we will put all of these parts together as done in Martin, Collela, Graves [32].

2.2 Finite Volume Formulation

Finite volume methods are a broad class of algorithms for finding numerical solutions to conservation laws of the form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{q}. \tag{2.1}$$

They have enjoyed widespread use because of their robustness and ability to deal with difficult problems arising from conservation laws, such as those involving shocks. The theory behind finite volume methods respects the fundamental structure the equations being solved and is intimately tied to the mathematical theory of hyperbolic equations. Without loss of generality, we will develop the finite volume formulation of the incompressible Navier-Stokes equations in two dimensions as the formulation in three dimensions extends readily from this case.

Suppose we wish to find the solution to (2.1) over a rectangular domain, $R : [0, L] \times [0, M]$, that we divide into a grid of N by M computational grid cells. Each one of these cells will have dimensions Δx by Δy where

$$\Delta x = \frac{L}{N},\ \Delta y = \frac{H}{M}.$$

For convenience we will further label the *center* of each grid cell as (i, j), where $1 \leq i \leq$

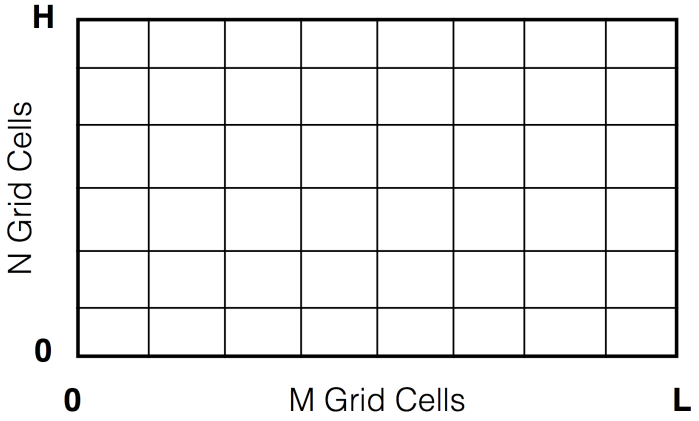

Figure 2.1: The computational domain (in two dimensions).

N, $1 \leq j \leq M$. The right *face* of the i, jth grid cell will be labeled $(i + \frac{1}{2}, j)$, with a similar convention for the other three faces.

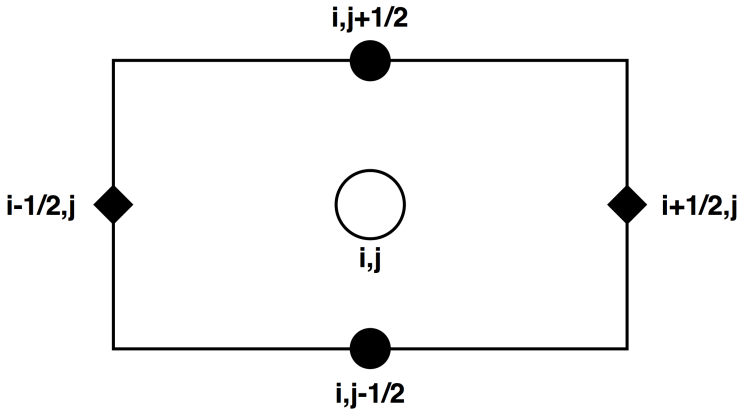

Figure 2.2: The i, jth computational grid cell (in two dimensions). The open circle indicates the cell centre. The diamonds indicate left and right cell faces, while the filled circles indicate the top and bottom cell faces.

Let $R_{i,j}$ denote the i, jth grid cell of the computational grid. Integrating (2.1) over this domain, using the divergence theorem, and dividing by $\Delta x \Delta y$ gives

$$\frac{1}{\Delta x \Delta y} \frac{\partial}{\partial t} \int_{R_{i,j}} \mathbf{u} \, dxdy + \frac{1}{\Delta x \Delta y} \int_{\partial R_{i+\frac{1}{2},j}} \mathbf{f} \cdot \hat{\mathbf{n}} \, dy - \frac{1}{\Delta x \Delta y} \int_{\partial R_{i-\frac{1}{2},j}} \mathbf{f} \cdot \hat{\mathbf{n}} \, dy + \frac{1}{\Delta x \Delta y} \int_{\partial R_{i,j+\frac{1}{2}}} \mathbf{f} \cdot \hat{\mathbf{n}} \, dx - \frac{1}{\Delta x \Delta y} \int_{\partial R_{i,j-\frac{1}{2}}} \mathbf{f} \cdot \hat{\mathbf{n}} \, dx = \frac{1}{\Delta x \Delta y} \int_{R_{i,j}} \mathbf{q} \, dxdy \quad (2.2)$$

where $\partial R_{i+\frac{1}{2},j}$ is the right cell face of $R_{i,j}$, and the other three faces are labeled in the same fashion. If we let

$$\mathbf{U_{i,j}}(t) = \frac{1}{\Delta x \Delta y} \int_{R_{i,j}} \mathbf{u}(x, y, t) \, dxdy \quad (2.3)$$

denote the average value of $\mathbf{u}$ over the i, jth grid cell at time t, using $\tilde{\mathbf{Q}}$ similarly for $\mathbf{q}$, and

$$\tilde{\mathbf{F}}_{i+\frac{1}{2},j}(t) = \frac{1}{\Delta y} \int_{\partial R_{i+\frac{1}{2},j}} \mathbf{f}\left(\mathbf{u}(x_{i+\frac{1}{2},j}, y, t)\right) \cdot \hat{\mathbf{n}} \, dy \quad (2.4)$$

be the (spatial) average flux through the right cell face, $\partial R_{i+\frac{1}{2},j}$, at time t, then equation (2.2) becomes

$$\frac{\partial \mathbf{U}_{i,j}}{\partial t} + \frac{\tilde{\mathbf{F}}_{i+\frac{1}{2},j} - \tilde{\mathbf{F}}_{i-\frac{1}{2},j}}{\Delta x} + \frac{\tilde{\mathbf{F}}_{i,j+\frac{1}{2}} - \tilde{\mathbf{F}}_{i,j-\frac{1}{2}}}{\Delta y} = \tilde{\mathbf{Q}}_{i,j}. \tag{2.5}$$

Suppose now that we wish to find our solution on the time interval $t \in [0, T]$. We achieve this by taking N discrete time-steps of (varying) size Δt^n. To determine a formula for advancing our solution from time t^n to time t^{n+1} we can integrate equation (2.5) over this interval and divide by Δt^n. This gives

$$\frac{\mathbf{U}_{i,j}^{n+1} - \mathbf{U}_{i,j}^{n}}{\Delta t^n} + \frac{\mathbf{F}_{i+\frac{1}{2},j}^{n+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2},j}^{n+\frac{1}{2}}}{\Delta x} + \frac{\mathbf{F}_{i,j+\frac{1}{2}}^{n+\frac{1}{2}} - \mathbf{F}_{i,j-\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta y} = \mathbf{Q}_{i,j}^{n+\frac{1}{2}}, \tag{2.6}$$

where

$$\mathbf{Q}_{i,j}^{n+\frac{1}{2}} = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \tilde{\mathbf{Q}}_{i,j}(t) dt \tag{2.7}$$

is the average value of $\mathbf{q}$ over the grid cell $R_{i,j}$ from time t^n to time t^{n+1}, and similarly,

$$\mathbf{F}_{i+\frac{1}{2},j}^{n+\frac{1}{2}} = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \tilde{\mathbf{F}}_{i+\frac{1}{2},j}(t) dt \tag{2.8}$$

is the average flux through the right cell face, $\partial R_{i+\frac{1}{2},j}$, from time t^n to time t^{n+1}. [1]

Notice that discrete equation (2.6) has the same conservative form as equation (2.1). This is satisfying from a physical point of view as our algorithm reflects the underlying nature of our system and is part of the reason that finite volume methods have enjoyed such success.

A second important point is that we have made no simplifying assumptions to obtain equation (2.6) (other than the regularity of our computational grid). This means that the order of accuracy of our algorithm will be completely determined by how we choose to evaluate the average fluxes and average forcing. We will examine one particular way of approximating the average flux in the next section.

[1] These are *not* the spatial averages $\tilde{\mathbf{Q}}_{i,j}(t)$ and $\tilde{\mathbf{F}}_{i+\frac{1}{2},j}(t)$ evaluated at the half time $t = t^n + \frac{\Delta t}{2}$.

2.2.1 Advection Terms: MUSCL Scheme

MUSCL (Monotone Upstream-centred Scheme for Conservation Laws) or Variable Extrapolation methods are a class of higher-order extensions to the first-order Godunov scheme. This class of methods can be broken down into three steps:

- **Reconstruct**: First, a piecewise polynomial reconstruction of $\mathbf{u}$ is buillt within each grid cell from the cell averaged values.
- **Evolve**: Next, the piecewise polynomial reconstruction is then evolved forward in time by solving the *Riemann problem* that results at each cell face from its neighbouring cells. The Riemann problem may be solved exactly, but often an approximate Riemann solver is used to spare computational cost.
- **Average**: Lastly, the solution to the Riemann problem is used to evaluate the average flux through the cell face. This may be done exactly, however the integral (2.8) is often approximated.

The advection terms in [7] are computed using Colella's second-order unsplit MUSCL type scheme [19], which uses a **primitive variable Riemann solver**. The conservation law, (2.1), is written in the form

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{c}(\mathbf{w}) \cdot \nabla \mathbf{w} = \mathbf{q} \tag{2.9}$$

were $\mathbf{w}$ are the primitive variables and $\mathbf{c} = (c_1, c_2)$. This could be the momentum equation, (1.1), or a the equation for a scalar which is advected with the fluid flow, for example. It is assumed that over one time-step the value of $\mathbf{w}$ is slowly varying and that the advection velocity can be approximated by a background state

$$\bar{\mathbf{c}}(x, y) = \mathbf{c}(\bar{\mathbf{w}}(x, y)) \tag{2.10}$$

and hence the primitive variable equation (2.9) becomes

$$\frac{\partial \mathbf{w}}{\partial t} + \bar{\mathbf{c}} \cdot \nabla \mathbf{w} = \mathbf{q}. \tag{2.11}$$

The Riemann problem for this equation can be solved *exactly* using the method of characteristics.

The average flux (2.8) is approximated using the midpoint rule

$$\begin{aligned}
\mathbf{F}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \tilde{\mathbf{F}}_{i+\frac{1}{2},j}(t)dt \\
&= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \frac{1}{\Delta y}\int_{\partial R_{i+\frac{1}{2},j}} \mathbf{f}\left(\mathbf{u}(x_{i+\frac{1}{2},j},y,t)\right)\cdot\hat{\mathbf{n}}\,dydt \\
&\approx \mathbf{f}\left(\mathbf{u}(x_{i+\frac{1}{2},j},y_{i+\frac{1}{2},j},t^n+\tfrac{\Delta t}{2})\right)\cdot\hat{\mathbf{n}} \qquad (2.12)
\end{aligned}$$

Because of the approximation (2.10), it is possible to write the flux in terms of the primitive variables $\mathbf{f}(\mathbf{w})\cdot\hat{\mathbf{n}} = \mathbf{c}(\bar{\mathbf{w}})\mathbf{w}$. Therefore (2.12) can be evaluated by finding an approximation to $\mathbf{w}$ at the right cell face at the half-time, $\mathbf{w}^{L,n+\frac{1}{2}}_{i+\frac{1}{2},j}$. This is accomplished using a Taylor series expansion about the cell-centre

$$\mathbf{w}^{L,n+\frac{1}{2}}_{i+\frac{1}{2},j} = \mathbf{w}(x_{i+\frac{1}{2},j},y_{i+\frac{1}{2},j},t^n+\tfrac{\Delta t}{2}) \approx \mathbf{W}^n_{i,j} + \frac{\Delta x}{2}\left(\frac{\partial \mathbf{w}}{\partial x}\right)^n_{i,j} + \frac{\Delta t}{2}\left(\frac{\partial \mathbf{w}}{\partial t}\right)^n_{i,j} \qquad (2.13)$$

and constitutes the reconstruction step discussed above. Since the time derivative is not easy to approximate apriori, equation (2.11) is solved for the time derivative and this is substituted into (2.13)

$$\begin{aligned}
\mathbf{w}^{L,n+\frac{1}{2}}_{i+\frac{1}{2},j} &\approx \mathbf{W}^n_{i,j} + \frac{\Delta x}{2}\left(\frac{\partial \mathbf{w}}{\partial x}\right)^n_{i,j} + \frac{\Delta t}{2}\left(-\mathbf{c}(\bar{\mathbf{w}})\cdot\nabla\mathbf{w}^n_{i,j} - \mathbf{Q}^n_{i,j}\right) \\
&= \mathbf{W}^n_{i,j} + \frac{\Delta x}{2}\left(1 - c_1(\bar{\mathbf{w}})\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial \mathbf{w}}{\partial x}\right)^n_{i,j} - c_2(\bar{\mathbf{w}})\frac{\Delta t}{2}\left(\frac{\partial \mathbf{w}}{\partial y}\right)^n_{i,j} - \frac{\Delta t}{2}\mathbf{Q}^n_{i,j}. \qquad (2.14)
\end{aligned}$$

The normal derivative can be evaluated using the second-order centred difference formula

$$\left(\frac{\partial \mathbf{w}}{\partial x}\right)^n_{i,j} = \frac{\mathbf{W}^n_{i+1,j} - \mathbf{W}^n_{i-1,j}}{\Delta x} \qquad (2.15)$$

and the transverse derivative can be evaluated using the upwinded difference formula

$$\left(\frac{\partial \mathbf{w}}{\partial y}\right)^n_{i,j} = \begin{cases} \frac{\mathbf{W}^n_{i,j} - \mathbf{W}^n_{i,j-1}}{\Delta y} & \text{if} \quad c_2 > 0 \\ \frac{\mathbf{W}^n_{i,j+1} - \mathbf{W}^n_{i,j}}{\Delta y} & \text{if} \quad c_2 < 0 \end{cases}. \qquad (2.16)$$

The normal and transverse derivatives may be slope limited, however, in practice this is often unnecessary at low Mach numbers. We can derive a similar approximation to $\mathbf{w}$ at the left cell face at the half time

$$\mathbf{w}_{i-\frac{1}{2},j}^{R,n+\frac{1}{2}} = \mathbf{W}_{i,j}^{n} - \frac{\Delta x}{2}\left(1 + c_1(\bar{\mathbf{w}})\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial \mathbf{w}}{\partial x}\right)_{i,j}^{n} - c_2(\bar{\mathbf{w}})\frac{\Delta t}{2}\left(\frac{\partial \mathbf{w}}{\partial y}\right)_{i,j}^{n} - \frac{\Delta t}{2}\mathbf{Q}_{i,j}^{n}. \tag{2.17}$$

After the left and right states are determined, the Riemann problem at each cell interface can be solved and used to calculate the numerical flux

$$\mathbf{F}_{i+\frac{1}{2},j}^{n+\frac{1}{2}} = \mathbf{c}(\bar{\mathbf{w}})\,\mathbf{w}_{i+\frac{1}{2},j}^{\text{Riemann}}. \tag{2.18}$$

The only outstanding decision to be made is how to choose the background state $\bar{\mathbf{w}}(x, y)$ used to evaluate the advection velocities. This can be chosen in a number of ways and we will examine how to do this in a predictor-corrector fashion in the next section.

An important point that has been overlooked so far is that of stability. It can be shown that the MUSCL scheme given above is stable provided the **Courant-Friedrichs-Lewy (CFL) Condition**

$$\Delta t < \frac{\Delta x}{\bar{\mathbf{c}}(x, y).} \tag{2.19}$$

is satisfied. This says that information will not propagate through more than one grid cell during one time-step. In practice the CFL condition is used to choose a stable time step

$$\Delta t^{n+1} = \frac{\sigma \Delta x}{\max\limits_{i,j}\ \bar{\mathbf{c}}_{i,j}^{n}} \tag{2.20}$$

where σ is our CFL factor[2]. Adaptive time-stepping of this form means that larger time-steps can be taken when the flow is slow and smaller time-steps will be used when necessary.

For convenience in what follows we will work in terms of our original primitive variables:

$$\frac{\mathbf{u}_{i,j}^{n+1} - \mathbf{u}_{i,j}^{n}}{\Delta t} + \left[(\mathbf{u}\cdot\nabla)\mathbf{u}\right]_{i,j}^{n+\frac{1}{2}} = -\left(\nabla p\right)_{i,j}^{n+\frac{1}{2}} + \nu\left(\nabla^2\mathbf{u}\right)_{i,j}^{n+\frac{1}{2}} \tag{2.21}$$

$$\frac{s_{i,j}^{n+1} - s_{i,j}^{n}}{\Delta t} + \left[\nabla\cdot(s\mathbf{u})\right]_{i,j}^{n+\frac{1}{2}} = \kappa(\nabla^2 s)_{i,j}^{n+\frac{1}{2}} \tag{2.22}$$

where it us understood that $\mathbf{u}_{i,j}^{n}$ denotes the *average* value of $\mathbf{u}$ over the i, jth grid cell even though we have dropped the capitalized notation.

[2] In most of our simulations we will use $\sigma = 0.5$

Advection Velocity

The face-centred advection velocity is first approximated by averaging the cell-centred velocity to the cell-faces:

$$\mathbf{u}^{\text{edge}}_{i+\frac{1}{2},j} = \frac{\mathbf{u}^n_{i,j} + \mathbf{u}^n_{i+1,j}}{2}, \tag{2.23}$$

with appropriate formulas for the other faces. Next, we correct this by extrapolating the *normal* component of the velocity from the cell-centre to the cell-faces via Taylor series expansion

$$u^{L,n+\frac{1}{2}}_{i+\frac{1}{2},j} = u^n_{i,j} + \frac{\Delta x}{2}\left(1 - u^{\text{norm}}_{i,j}\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial u}{\partial x}\right)^n_{i,j} - v^{\tan}_{i,j}\frac{\Delta t}{2}\left(\frac{\partial u}{\partial y}\right)^n_{i,j} + \frac{\Delta t}{2}\nu(\nabla^2 u)^n_{i,j} \tag{2.24}$$

$$u^{R,n+\frac{1}{2}}_{i-\frac{1}{2},j} = u^n_{i,j} - \frac{\Delta x}{2}\left(1 + u^{\text{norm}}_{i,j}\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial u}{\partial x}\right)^n_{i,j} - v^{\tan}_{i,j}\frac{\Delta t}{2}\left(\frac{\partial u}{\partial y}\right)^n_{i,j} + \frac{\Delta t}{2}\nu(\nabla^2 u)^n_{i,j} \tag{2.25}$$

where

$$u^{\text{norm}}_{i,j} = \frac{u^{\text{edge}}_{i-\frac{1}{2},j} + u^{\text{edge}}_{i+\frac{1}{2},j}}{2},\ v^{\tan}_{i,j} = \frac{v^{\text{edge}}_{i,j-\frac{1}{2}} + v^{\text{edge}}_{i,j+\frac{1}{2}}}{2} \tag{2.26}$$

and the partial derivatives are evaluated using the differencing schemes (2.15) and (2.16) described above[3].

Next, the Riemann problem is solved at each cell-face. For our equations this amounts to choosing the upwind state:

$$u^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = \begin{cases} u^{L,n+\frac{1}{2}}_{i+\frac{1}{2},j} & \text{if} \quad u^{\text{edge}}_{i+\frac{1}{2},j} > 0 \\ u^{R,n+\frac{1}{2}}_{i+\frac{1}{2},j} & \text{if} \quad u^{\text{edge}}_{i+\frac{1}{2},j} < 0 \\ \frac{1}{2}(u^{L,n+\frac{1}{2}}_{i+\frac{1}{2},j} + u^{R,n+\frac{1}{2}}_{i+\frac{1}{2},j}) & \text{if} \quad u^{\text{edge}}_{i+\frac{1}{2},j} = 0 \end{cases}. \tag{2.27}$$

Finally, the edge-centred advection velocity is MAC projected[4] to ensure it is incompressible

$$\mathbf{u}^{\text{half}} = \mathbb{P}^{MAC}\left(\mathbf{u}^{n+\frac{1}{2}}\right). \tag{2.28}$$

The is the velocity that is now used to advect any scalar quantities and the velocity itself.

[3] The pressure gradient is *not* included in the extrapolation step because the advection velocity will be projected with an edge-centred MAC projection which will in turn remove the pressure gradient.

[4] See Section 2.2.2 for an explanation of the MAC projection.

Scalar Advection

Scalar quantities are updated in a similar fashion using the extrapolation formulas

$$s_{i+\frac{1}{2},j}^{L,n+\frac{1}{2}} = s_{i,j}^{n} + \frac{\Delta x}{2}\left(1 - u_{i,j}^{\text{norm}}\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial s}{\partial x}\right)_{i,j}^{n} - v_{i,j}^{\tan}\frac{\Delta t}{2}\left(\frac{\partial s}{\partial y}\right)_{i,j}^{n} + \frac{\Delta t}{2}\kappa(\nabla^2 s)_{i,j}^{n} \quad (2.29)$$

$$s_{i-\frac{1}{2},j}^{R,n+\frac{1}{2}} = s_{i,j}^{n} - \frac{\Delta x}{2}\left(1 + u_{i,j}^{\text{norm}}\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial s}{\partial x}\right)_{i,j}^{n} - v_{i,j}^{\tan}\frac{\Delta t}{2}\left(\frac{\partial s}{\partial y}\right)_{i,j}^{n} + \frac{\Delta t}{2}\kappa(\nabla^2 s)_{i,j}^{n} \quad (2.30)$$

where

$$u_{i,j}^{\text{norm}} = \frac{u_{i-\frac{1}{2},j}^{\text{half}} + u_{i+\frac{1}{2},j}^{\text{half}}}{2},\ v_{i,j}^{\tan} = \frac{v_{i,j-\frac{1}{2}}^{\text{half}} + v_{i,j+\frac{1}{2}}^{\text{half}}}{2} \quad (2.31)$$

and the partial derivatives are evaluated using the differencing schemes (2.15) and (2.16). Next, the Riemann problem is solved at each cell-face. For our equations this amounts to choosing the upwind state:

$$s_{i+\frac{1}{2},j}^{n+\frac{1}{2}} = \begin{cases} s_{i+\frac{1}{2},j}^{L,n+\frac{1}{2}} & \text{if} \quad u_{i+\frac{1}{2},j}^{\text{edge}} > 0 \\ s_{i+\frac{1}{2},j}^{R,n+\frac{1}{2}} & \text{if} \quad u_{i+\frac{1}{2},j}^{\text{edge}} < 0 \\ \frac{1}{2}(s_{i+\frac{1}{2},j}^{L,n+\frac{1}{2}} + s_{i+\frac{1}{2},j}^{R,n+\frac{1}{2}}) & \text{if} \quad u_{i+\frac{1}{2},j}^{\text{edge}} = 0 \end{cases}. \quad (2.32)$$

The flux can now be calculated at each cell face as

$$F_{i+\frac{1}{2},j}^{n+\frac{1}{2}} = u_{i+\frac{1}{2},j}^{\text{half}} s_{i+\frac{1}{2},j}^{n+\frac{1}{2}} \quad (2.33)$$

and the scalar s can be updated using the discrete conservation law (2.6). The integral for the source term of the source term on the right side of (2.6) is approximated using the trapezoidal rule. This leads to an implicit Crank-Nicholson type scheme for the diffusive terms which can be solved using standard multigrid techniques. (See the section on the discretization of the viscous terms of the momentum equation for details).

Velocity Advection

The velocity field is updated using advective differencing. Reusing the *normal* velocities predicted in the advection velocity calculation, only the *tangential* component of the

velocity at each cell face is calculated now. The extrapolation formulas are

$$v_{i+\frac{1}{2},j}^{L,n+\frac{1}{2}} = v_{i,j}^{n} + \frac{\Delta x}{2}\left(1 - u_{i,j}^{\text{norm}}\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial v}{\partial x}\right)_{i,j}^{n} - v_{i,j}^{\tan}\frac{\Delta t}{2}\left(\frac{\partial v}{\partial y}\right)_{i,j}^{n} + \frac{\Delta t}{2}\nu(\nabla^2 v)_{i,j}^{n} \quad (2.34)$$

$$v_{i-\frac{1}{2},j}^{R,n+\frac{1}{2}} = v_{i,j}^{n} - \frac{\Delta x}{2}\left(1 + u_{i,j}^{\text{norm}}\frac{\Delta t}{\Delta x}\right)\left(\frac{\partial v}{\partial x}\right)_{i,j}^{n} - v_{i,j}^{\tan}\frac{\Delta t}{2}\left(\frac{\partial v}{\partial y}\right)_{i,j}^{n} + \frac{\Delta t}{2}\nu(\nabla^2 v)_{i,j}^{n} \quad (2.35)$$

where $u_{i,j}^{\text{norm}}$ and $v_{i,j}^{\tan}$ are given by (2.31), and the partial derivatives are evaluated using the differencing schemes (2.15) and (2.16). Next, the Riemann problem is solved at each cell-face:

$$v_{i+\frac{1}{2},j}^{n+\frac{1}{2}} = \begin{cases} v_{i+\frac{1}{2},j}^{L,n+\frac{1}{2}} & \text{if} \quad u_{i+\frac{1}{2},j}^{\text{half}} > 0 \\ v_{i+\frac{1}{2},j}^{R,n+\frac{1}{2}} & \text{if} \quad u_{i+\frac{1}{2},j}^{\text{half}} < 0 \\ \frac{1}{2}(v_{i+\frac{1}{2},j}^{L,n+\frac{1}{2}} + v_{i+\frac{1}{2},j}^{R,n+\frac{1}{2}}) & \text{if} \quad u_{i+\frac{1}{2},j}^{\text{half}} = 0 \end{cases} . \quad (2.36)$$

The MAC correction[5] is then added to the tangential velocity ...

$$\mathbf{v}^{\text{half}} = \mathbf{v}^{n+\frac{1}{2}} - (\nabla\phi)^{\text{MAC}} \quad (2.37)$$

and a cell-centred advection velocity $\mathbf{u}^{ADV}$ is formed by averaging the edge-centred MAC projected velocities to the cell centres. Lastly, the velocities are updated using equation (2.21) where the advection terms are calculated using the formulas

$$\left[(\mathbf{u}\cdot\nabla)\,u\right]_{i,j}^{n+\frac{1}{2}} = u_{i,j}^{\text{ADV}}\left(\frac{u_{i+\frac{1}{2},j}^{\text{half}} - u_{i-\frac{1}{2},j}^{\text{half}}}{\Delta x}\right) + v_{i,j}^{ADV}\left(\frac{u_{i,j+\frac{1}{2}}^{\text{half}} - u_{i,j-\frac{1}{2}}^{\text{half}}}{\Delta y}\right) \quad (2.38)$$

$$\left[(\mathbf{u}\cdot\nabla)\,v\right]_{i,j}^{n+\frac{1}{2}} = u_{i,j}^{\text{ADV}}\left(\frac{v_{i+\frac{1}{2},j}^{\text{half}} - v_{i-\frac{1}{2},j}^{\text{half}}}{\Delta x}\right) + v_{i,j}^{ADV}\left(\frac{v_{i,j+\frac{1}{2}}^{\text{half}} - v_{i,j-\frac{1}{2}}^{\text{half}}}{\Delta y}\right) \quad (2.39)$$

Viscous Terms

The integral average of the viscous terms in equation (2.21) is approximated using the trapezoidal rule. This results in a semi-implicit Crank-Nicholson scheme. The Laplacian operator is discretized using a standard five point stencil

$$(\nabla^2\mathbf{u})_{i,j} = \frac{\mathbf{u}_{i+1,j} - 2\mathbf{u}_{i,j} + \mathbf{u}_{i-1,j}}{\Delta x^2} + \frac{\mathbf{u}_{i,j+1} - 2\mathbf{u}_{i,j} + \mathbf{u}_{i,j-1}}{\Delta y^2} \quad (2.40)$$

[5] See Section 2.2.2 for details on how this is computed.

to achieve second order accuracy in space. The viscous term $(\nabla^2 \mathbf{u})_{i,j}^{n+\frac{1}{2}}$ is then evaluated as the average of the values at times t^n and t^{n+1},

$$(\nabla^2 \mathbf{u})_{i,j}^{n+\frac{1}{2}} = \frac{(\nabla^2 \mathbf{u})_{i,j}^{n+1} + (\nabla^2 \mathbf{u})_{i,j}^{n}}{2}. \tag{2.41}$$

This Helmholtz problem for $\mathbf{u}^{n+1}$ is solved using standard multigrid techniques. (See [14] or [49] for example).

2.2.2 Projection Methods

One of the difficulties in solving the incompressible Navier-Stokes equations is that there is no evolution equation for the pressure. Additionally one must decide how to enforce the incompressibility condition (1.2). Fortunately both of these difficulties can be addressed simultaneously using the Helmholtz Decomposition Theorem.

Recall that the Helmholtz Decomposition Theorem states if $\mathbf{V}^*$ is a twice differentiable vector field then it can be decomposed into an *incompressible* field $\mathbf{V}$ and an *irrotational* field $\nabla\phi$,

$$\mathbf{V}^* = \mathbf{V} + \nabla\phi. \tag{2.42}$$

Taking the divergence of this gives

$$\nabla^2\phi = \nabla \cdot \mathbf{V}^* \tag{2.43}$$

which we can solve formally for ϕ. Once we know ϕ we can solve equation (2.42) to obtain an expression for $\mathbf{V}$,

$$\mathbf{V} = \mathbf{V}^* - \nabla\phi. \tag{2.44}$$

We can now formally define the projection operator $\mathbb{P}$ to be

$$\mathbb{P}[\mathbf{V}^*] = (\mathbb{I} - \nabla(\nabla^2)^{-1}\nabla\cdot)\mathbf{V}^* = \mathbf{V}. \tag{2.45}$$

This operator returns the incompressible part of the vector field $\mathbf{V}*$. We can similarly define the complement to this operator

$$(\mathbb{I} - \mathbb{P})\left[\mathbf{V}^*\right] = \nabla\phi \tag{2.46}$$

which returns the gradient part of $\mathbf{V}^*$. To find boundary conditions for (2.43) we solve equation (2.42) for $\nabla\phi$ and then evaluate the normal component of this on the boundary:

$$\frac{\partial\phi}{\partial\hat{\mathbf{n}}} = \hat{\mathbf{n}} \cdot \nabla\phi = \hat{\mathbf{n}} \cdot (\mathbf{V}^* - \mathbf{V}). \tag{2.47}$$

Chorin [17] was the first to formally demonstrate that this process could be used enforce the incompressibility condition for a discretized form of the Navier-Stokes equation. By introducing discrete versions $\mathbf{D}$ and $\mathbf{G}$ of the divergence and gradient operators, respectively, a discrete version of the projection operator can be defined by

$$\mathbb{P}[\mathbf{V}^*] = (\mathbf{I} - \mathbf{G}(\mathbf{DG})^{-1}\mathbf{D})\mathbf{V}^*. \tag{2.48}$$

One can rewrite equation (2.21) in the form

$$\begin{aligned} \mathbf{u}^{n+1} + \Delta t \left(\nabla p\right)^{n+\frac{1}{2}} &= \mathbf{u}^n + \Delta t \left[(\mathbf{u}\cdot\nabla)\mathbf{u}\right]^{n+\frac{1}{2}} + \Delta t\nu \left(\nabla^2\mathbf{u}\right)^{n+\frac{1}{2}} & (2.49)\\ &= \mathbf{u}^{n+1,*} + \Delta t(\nabla p)^{n-\frac{1}{2}} & (2.50) \end{aligned}$$

where

$$\mathbf{u}^{n+1,*} = \mathbf{u}^n + \Delta t \left[(\mathbf{u}\cdot\nabla)\mathbf{u}\right]^{n+\frac{1}{2}} + \Delta t\nu \left(\nabla^2\mathbf{u}\right)^{n+\frac{1}{2}} - \Delta t(\nabla p)^{n-\frac{1}{2}} \tag{2.51}$$

and $(\nabla p)^{n-\frac{1}{2}}$ is the lagged pressure gradient, i.e. the average pressure gradient over the previous time-step. Applying the discrete projection operator to equation (2.50) gives

$$\mathbf{u}^{n+1} = \mathbb{P}\left[\mathbf{u}^{n+1,*} + \Delta t(\nabla p)^{n-\frac{1}{2}}\right] \tag{2.52}$$

and

$$(\nabla p)^{n+\frac{1}{2}} = \frac{1}{\Delta t}\left(\mathbb{I} - \mathbb{P}\right)\left[\mathbf{u}^{n+1,*} + \Delta t(\nabla p)^{n-\frac{1}{2}}\right]. \tag{2.53}$$

It can now be seen that in this formulation we are solving (2.43) for the pressure *directly* and then correcting using equation (2.44) to find the incompressible velocity field at the new time [6].

Unfortunately the discretization used in [17] for the operators lead to a local decoupling of grid cells and non trivial kernel for $\mathbf{G}$. While Bell, Colella, and Glaz [7] managed to use a similar method based on a finite element discretization and a more compact stencil for $\mathbf{DG}$, their method ultimately suffered from the same drawbacks. Additionally, the node-centred stencils used in both of these algorithms meant that two different sets of linear algebra solvers had to be carried around in their code: one for the projection and one for the viscous solve.

In pursuit of a cell-centred discretization of the projection which avoided local grid decoupling and fit into the framework of fast iterative solvers (i.e. multigrid), many authors

[6] This is in contrast to other formulations which use some combination of projecting the velocity update $\frac{\partial \mathbf{u}}{\partial t}$ and solve for the pressure increment $\nabla\phi = (\nabla p)^{n+\frac{1}{2}} - (\nabla p)^{n-\frac{1}{2}}$. Almgren et al. [3] showed that these formulations have less desirable properties than the current one for cell-centred stencils.

relaxed the requirement that the discretized projection be *exact*. This led to a class of algorithms known as **approximate projection methods**[7]. In this class of methods a convenient discretization is chosen for the Laplacian operator, $\mathbf{L}$, in equation (2.43), independent of the discretizations chosen for the divergence, $\mathbf{D}$, and the gradient, $\mathbf{G}$. Hence it is often the case that $\mathbf{L} \neq \mathbf{DG}$. This means that even with repeated application of the projection operator $\mathbb{P}$, the velocity field will have a divergence on the same order of magnitude error of the discretization used.

MAC Projection

In [31] the advection velocity is projected prior to being used to ensure it is (approximately) divergence free and to allow for a less severe CFL constraint [4]. This face-centred projection is known as a **MAC projection**. Equation (2.43) is solved for ϕ using standard multigrid techniques, where the divergence of the face-centred velocity is calculated as

$$(\nabla \cdot \mathbf{V}^*)_{i,j} = \frac{{U^*}_{i+\frac{1}{2},j} - {U^*}_{i-\frac{1}{2},j}}{\Delta x} + \frac{{V^*}_{i,j+\frac{1}{2}} - {V^*}_{i,j-\frac{1}{2}}}{\Delta y}. \tag{2.54}$$

Next, $\nabla\phi$ is calculated at the cell faces using the differencing formulas

$$(\phi_x)_{i+\frac{1}{2},j} = \frac{(\phi)_{i+1,j} - (\phi)_{i,j}}{\Delta x} \tag{2.55}$$

$$(\phi_y)_{i,j+\frac{1}{2}} = \frac{(\phi)_{i,j+1} - (\phi)_{i,j}}{\Delta y}. \tag{2.56}$$

and then used to correct the advection velocity using (2.44).

Cell-Centred Projection

Martin and Colella [31] also use a cell-centred approximate projection method to enforce the incompressibility condition at the end of each time-step. First an approximation to the velocity at the new time-step $\mathbf{u}^{n+1,*}$ is found at the cell centres using the procedure outlined in the previous sections. Next this cell-centred velocity is averaged to the cell faces:

$$u^{n+1,*}_{i+\frac{1}{2},j} = \frac{u^{n+1,*}_{i+1,j} + u^{n+1,*}_{i,j}}{2} \tag{2.57}$$

$$v^{n+1,*}_{i,j+\frac{1}{2}} = \frac{v^{n+1,*}_{i,j+1} + v^{n+1,*}_{i,j}}{2} \tag{2.58}$$

[7] See [3] for a thorough review of approximate projection methods.

The lagged pressure gradient at the cell faces is then added to this velocity

$$\mathbf{V}^* = \mathbf{u}^{n+1,*} + \Delta t^n (\nabla p)^{n-\frac{1}{2}}. \tag{2.59}$$

and this is the vector field that is then projected. The Laplacian operator in equation (2.43) is discretized using a standard five-point finite difference stencil (2.40) to achieve second order accuracy. Equation (2.43) is solved for ϕ using standard multigrid techniques, where the divergence of the face-centred velocity is calculated using (2.54). Next, $\nabla\phi$ is calculated at the cell faces using (2.57) and (2.58), and then averaged to the cell centres via

$$(\phi_x)^{n+1}_{i,j} = \frac{(\phi_x)^{n+1}_{i+\frac{1}{2},j} + (\phi_x)^{n+1}_{i-\frac{1}{2},j}}{2} \tag{2.60}$$

$$(\phi_y)^{n+1}_{i,j} = \frac{(\phi_y)^{n+1}_{i,j+\frac{1}{2}} + (\phi_y)^{n+1}_{i,j-\frac{1}{2}}}{2}. \tag{2.61}$$

Lastly this cell-centred gradient is used to correct the original cell-centred velocity field with equation (2.44).

2.2.3 Initial and Boundary Conditions

Given initial and boundary conditions we now have a recursive procedure to determine $\mathbf{u}$ and s at all points in our domain and for all times $t \in [0, T]$. We will now examine how to calculate the boundary terms and initialize all of our fields.

Boundary Conditions

Suppose that we have the boundary conditions

$$G\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}\right) = 0, \; H\left(s, \frac{\partial s}{\partial x}\right) = 0 \tag{2.62}$$

on ∂R, for some functions G and H. To implement these numerically a ring of **ghost cells** is introduced around the edge of the computational grid as seen in Figure 2.3. We then fill these cells by extrapolating in the direction normal to the boundary using the boundary condition at the edge of our domain and an appropriate number of interior grid cells. The order of our interpolating polynomial must match the order of our finite volume scheme to ensure we do not loose accuracy at the boundary. Additionally the depth of the ring of ghost cells will depend on the order of our scheme.

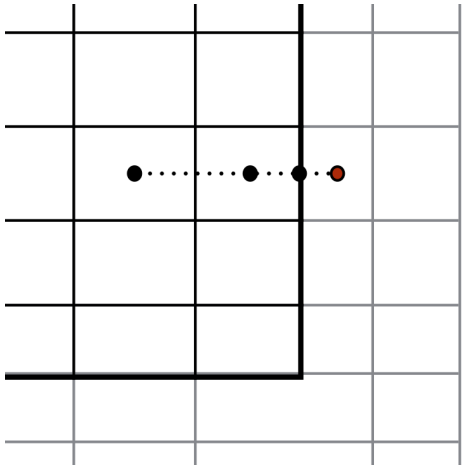

Figure 2.3: A ring of ghost cells two cells deep around the outside of a computational domain. The black circles show the points that are used for the extrapolation to fill in the value at the centre of the ghost cell, indicated with the red circle.

Initial Conditions

Given a set of initial conditions

$$\mathbf{u}(x, y, 0) = \mathbf{u}_0(x, y) \tag{2.63}$$

$$s(x, y, 0) = s_0(x, y) \tag{2.64}$$

we are not guaranteed that $\mathbf{u}_0$ is divergence free. Additionally, we need to find our initial pressure gradient as the mathematical theory of the Navier-Stokes equations tells us that we cannot specify both the velocity *and* the pressure. To accomplish these two goals we first perform a specified number of projections on the initial velocity field to drive its divergence towards zero. Typically only one projection is performed, but one could use more. Since we do not know the initial pressure gradient yet it is not included in the projection step.

To determine the initial pressure gradient we first initialize it to be zero. Next we take a time-step of size $\Delta t_{\text{init}} = \frac{\Delta t}{2}$ using the time-stepping algorithm described above. The cell-centred projection at the end of the time-step solves for the pressure. The velocities are then reset to the (divergence free) initial conditions. The process may be repeated any number of time, although it is typically only done a single time. Once the pressure gradient is initialized the simulation can begin at time $t = 0$ using a full time-step.

2.3 Adaptive Mesh Refinement

When using numerical analysis to find an approximate solution to any differential equation it is important to ensure that one uses a sufficiently large number of grid cells to guarantee that the solution has converged, i.e., using a larger number of grid cells won't improve the accuracy of the solution any further. In [44], it was demonstrated that having sufficient resolution is even more important that the order of accuracy of an algorithm in the sense that even high order schemes give poor results for under-resolved flows. It is often the case, especially for variable density flows, however, that the phenomena of interest is confined to a small region of the fluid domain. Hence if a uniform resolution is used for the entire domain much of the computational resources will be wasted in regions where the fluid flow is relatively uniform. **Adaptive Mesh Refinement (AMR)** methods seek to resolve the conflict between the need for sufficient resolution and finite computational resources by *only* refining the computational grid in regions where it is actually needed and adapting grids in time to follow flow features as they evolve.

While a number of different kinds of mesh refinement strategies are available, we would like to reuse as much of our previous algorithm for solving the Navier-Stokes equations as possible. We will therefore restrict ourselves to the use of **Block Structured Adaptive Mesh Refinement**. Block Structured AMR seeks to build a hierarchy of logically rectangular grids which satisfy a proper nesting criterion, as shown in Figure 2.4. This criterion ensures that each successive level of grid refinement is completely contained within the previous level, and that grids must remain orthogonal to coordinate lines. All of this results in a greatly simplified time-stepping algorithm and a significant increase in ease of implementation. Before we examine how to perform calculations on an AMR hierarchy we will introduce some precise definitions to clarify the constraints on our grids.

2.3.1 Preliminary Notions

To define our AMR hierarchy we will index our levels of grids with integers, $0 \leq l \leq N$, with 0 corresponding to the coarsest level and N corresponding to the finest. Let $\Omega^{(l)}$ denote the lth level and $\partial\Omega^{(l)}$ denote its boundary. Our levels will satisfy the criteria

$$\Omega^{(l+1)} \subset \Omega^{(l)} \tag{2.65}$$

and additionally will satisfy the proper nesting criterion

$$\partial\Omega^{(l-1)} \cap \partial\Omega^{(l+1)} = \emptyset. \tag{2.66}$$

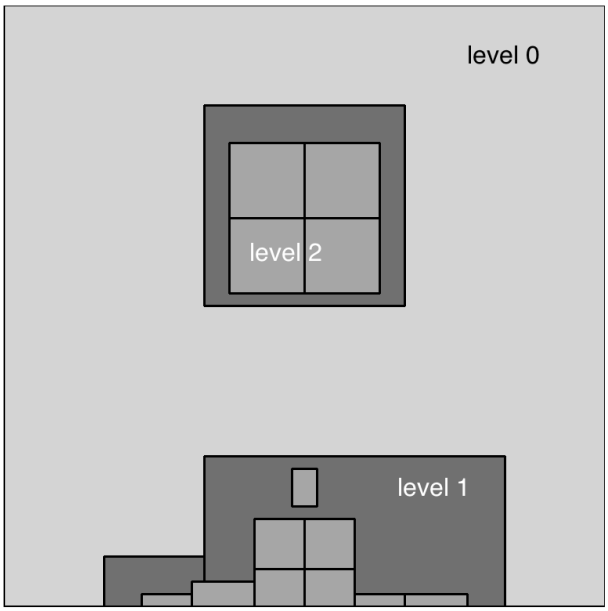

Figure 2.4: An example of an a properly nested block structured AMR hierarchy. The base level 0 (light gray) is the coarsest grid and fills the entire problem domain. Level 1 (dark gray) is the next finest level of grids and is completely contained within level 0. Finally level 2 (gray) contains the finest grids and is completely contained within level 1. The black lines show the boundaries between the various grids on each level. In this example level 2 consists of the union of 13 grids of varying size.

This ensures that there will be a ring at least one grid cell wide separating successive levels, which is sufficient for a second order method.

Let $\Gamma^{(0)}$ be the grid with cells of size $\Delta x^{(0)}$ which covers our entire base domain. Then $\Omega^{(0)} = \Gamma^{(0)}$. Let $\Gamma_k^{(l)}$ be a grid of cell size $\Delta x^{(l)}$ on level $\Omega^{(l)}$. We will require that our grids do not overlap,

$$\Gamma_j^{(l)} \cap \Gamma_k^{(l)} = \emptyset, \ j \neq k, \tag{2.67}$$

and completely cover our level,

$$\Omega^{(l)} = \cup \, \Gamma_k^{(l)}. \tag{2.68}$$

We can define the **refinement ratio** to be the integer amount, $r^{(l)}$, that we have decreased our cell size by:

$$r^{(l)} = \frac{\Delta x^{(l)}}{\Delta x^{(l+1)}}. \tag{2.69}$$

Since cells on levels lower in the hierarchy will be covered by cells on higher levels, it is useful to define the coarsening operator

$$\mathcal{C}_r(i,j) = \left(\left\lfloor \frac{i}{r} \right\rfloor, \left\lfloor \frac{j}{r} \right\rfloor \right). \tag{2.70}$$

This operator takes in index of a cell on a finer level and returns the index of the grid cell on the coarser level which lies beneath it. From the criteria (2.65) and (2.66) it follows that

$$\mathcal{C}_r\left(\Omega^{(l+1)}\right) \subset \Omega^{(l)}. \tag{2.71}$$

We are now in a position to define the **valid region**, $\Omega^{(l)}_{\text{valid}}$, of level l to be the region which is not covered by any finer levels:

$$\Omega^{(l)}_{\text{valid}} = \Omega^{(l)} \setminus \mathcal{C}_r\left(\Omega^{(l+1)}\right). \tag{2.72}$$

If $\mathbf{u}$ is one of our variables then we can define a **level variable**, $\mathbf{u}^{(l)}$, to be the value of $\mathbf{u}$ on level l, including the invalid region. Similarly we define a **composite variable**, $\mathbf{u}^{\text{comp}}$, values of $\mathbf{u}$ on the union of the valid regions of each level over the entire AMR hierarchy. When it is necessary to transfer data from a finer level to a coarser level we define the averaging operator $\left\langle \mathbf{u}^{(l)} \right\rangle$ to be the arithmetic average of $\mathbf{u}^{(l)}$ to $\mathcal{C}_r\left(\Omega^{(l+1)}\right)$. Finally, it will be understood that when we solve a problem on an AMR hierarchy that we will take our solution to be the composite version of our variables.

2.3.2 Grid Generation

Grid generation is handled using the algorithm outlined in [9]. A user specified criterion it used to determine whether a particular grid cells needs further refinement. The algorithm in [9] is then used to generate a series of grids from the point clusterings. In the present work the criterion

$$\|\boldsymbol{\omega}\| > \omega_{\text{threshold}}$$

is used to tag cells for refinement, where $\boldsymbol{\omega}$ is the vorticity field, given by 1.9, and $\omega_{\text{threshold}}$ is chosen on a problem-by-problem basis.

There are several important grid parameters which control the gridding process. The first is the **maximum box size**. For example, if a patch of tagged cells could be covered by a grid box of 1024×1024 grid cells, but the maximum box size is restricted to 256×256, then the grid generation algorithm will cover this patch of tagged cells with 4 boxes instead of 1. While this may seem like an unnecessary increase in computational cost, it a HPC

environment it can actually lead to better load balancing over a larger number of processors. On the base level the maximum grid size will determine how many boxes the base level should be decomposed into. Hence for a single grid run this amounts to simply performing a domain decomposition.

A second important grid parameter is the so-called **block-factor**. In the simplest sense this determines the minimum grid size that may be used on a given level. Again, in a HPC environment this parameter ensures that there is not an unnecessarily large number of grids on a given level, leading to a large communication cost between processors. In a deeper sense, due to the nature of the multigrid algorithm presented in Section 2.3.4, the block-factor also specifies the minimum amount that a given AMR level is coarsenable by. It will be required that the block-factor, bf, satisfies the property

$$\frac{bf}{r^l} = 2^N$$

for some integer N to ensure that the multigrid algorithm can be performed.

The last grid parameter that must be specified is the **fill ratio** or *efficiency parameter*, $0 < \epsilon < 1$. Given a cluster of tagged grid cell, ϵ specifies the minimum allowable ratio of tagged grid cells to total coarse grid points covered by a new grid. Hence, setting ϵ closer to one will generate a hierarchy of grids which more aggressively follow flow features. The trade-off however is that more small grids may have to be used.

2.3.3 Stencils for Composite and Level Variables

Now that it is understood what we mean when we talk about an AMR hierarchy, we are in a position to describe the stencils used for computations on both composite and level variables. These will look very similar to the single grid case on the interior of each level, however, special attention will need to be paid to computations near the coarse-fine interface between two levels.

Divergence and Flux Registers

Since we would like to solve (2.1) using AMR it is necessary to determine how to evaluate the divergence of level and composite variables. Let $\mathbf{u} = (u, v)$ be an edge-centred vector field on level l. The **level divergence** of $\mathbf{u}$ is given by the differencing formula

$$\left(D^{(l)}\mathbf{u}\right)_{i,j} = \frac{u^{(l)}_{i+\frac{1}{2},j} - u^{(l)}_{i-\frac{1}{2},j}}{\Delta x^l} + \frac{v^{(l)}_{i,j+\frac{1}{2}} - v^{(l)}_{i,j-\frac{1}{2}}}{\Delta x^l}, \tag{2.73}$$

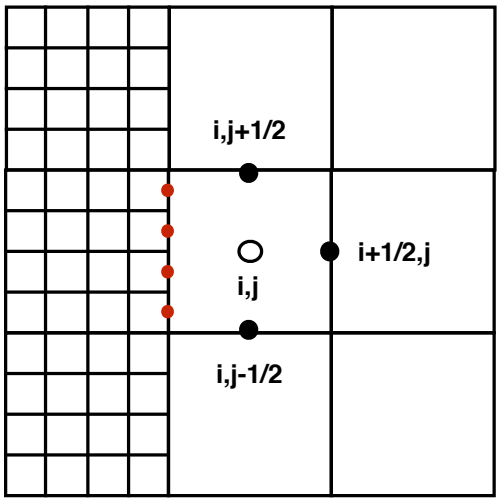

Figure 2.5: An example of the composite divergence stencil at a coarse-fine interface. The value of $\mathbf{u}^{(l)}$ at the black dots, as well as the *average* value of $\mathbf{u}^{(l+1)}$ at the red dots is used to calculate the composite divergence of $\mathbf{u}$ at the cell centre of the i,jth grid cell on level l.

ignoring any coarse-fine boundaries. If none of the edges of the i,jth grid cell are on the interface of the next finest level then the **composite divergence** of $\mathbf{u}$, $D^{\text{comp},l}\mathbf{u}$, is given by the same formula. If, however, any of the edges of a grid cell are on the interface of the next finest level then the *average* value $\left\langle \mathbf{u}^{(l)} \right\rangle$ is used. For example, for the grid cell in Figure 2.5. the stencil is given by

$$\left(D^{\text{comp},l}\mathbf{u}\right)_{i,j} = \frac{u^{(l)}_{i+\frac{1}{2},j} - \left\langle u^{(l+1)} \right\rangle_{i-\frac{1}{2},j}}{\Delta x^l} + \frac{v^{(l)}_{i,j+\frac{1}{2}} - v^{(l)}_{i,j-\frac{1}{2}}}{\Delta x^l}. \tag{2.74}$$

If any of the edges of a grid cell are on the interface of the next coarser level then nothing needs to be done since the value of $\mathbf{u}^{(l)}$ is valid there. It is inconvenient to keep track of all of the different cell edges where a different stencil will need to be applied. In the interest of computational efficiency, it would be convenient if we could use the standard level divergence stencil and then compute the difference between $\mathbf{u}^l$ and $\mathbf{u}^{(l+1)}$ only when necessary. To this end we will define the **flux register**,

$$\delta\mathbf{u}^{(l+1)} = \left\langle \mathbf{u}^{(l+1)} \right\rangle - \mathbf{u}^{(l)} \tag{2.75}$$

which is defined on $\mathcal{C}_r\left(\partial\Omega^{(l+1)}\right)$, i.e.,the coarse-fine interface. $\delta\mathbf{u}^{(l+1)}$ belongs to level $l+1$ but has the grid spacing and indexing of level l. We can now express the composite

divergence of $\mathbf{u}$ as

$$\left(D^{\text{comp},l}\mathbf{u}\right)_{i,j} = \left(D^{(l)}\mathbf{u}^{(l)}\right)_{i,j} + \left(D_R^{(l)}\,\delta\mathbf{u}^{(l+1)}\right)_{i,j} \tag{2.76}$$

where

$$\left(D_R^{(l)}\,\delta\mathbf{u}^{(l+1)}\right)_{i,j} = \frac{1}{\Delta x^l}\sum \pm\delta\mathbf{u}^{(l+1)} \tag{2.77}$$

is the **reflux divergence**, where the sum is over all of the cell edges on the coarse-fine interface and the $\pm$ corresponds to the right/top and left/bottom sizes of the cell, respectively.

The Gradient and Coarse-Fine Interpolation

In addition the divergence of a composite variable we can also define how to take the gradient. The **composite gradient**, $G^{\text{comp}}\phi$, of a cell centred scalar field ϕ is a face centred vector field which is defined on all valid edges in the AMR hierarchy. At cell which are not adjacent to the coarse-fine interface between two levels, the stencil for the composite gradient reduces to the usual centred differencing formulas

$$\left(G_x^{\text{comp},l}\phi\right)_{i+\frac{1}{2},j} = \frac{\phi^{(l)}_{i+1,j} - \phi^{(l)}_{i,j}}{\Delta x^l} \tag{2.78}$$

$$\left(G_y^{\text{comp},l}\phi\right)_{i,j+\frac{1}{2}} = \frac{\phi^{(l)}_{i,j+1} - \phi^{(l)}_{i,j}}{\Delta x^l}. \tag{2.79}$$

For cells on a finer level which are adjacent to a coarse-fine boundary, ghost cells are filled in by interpolating data from the coarser level. Figure 2.6 shows typical stencils used for this case. First, a quadratic interpolation is performed on the coarser level in a direction *parallel* to the coarse-fine interface to find an intermediate value. Next, a second quadratic interpolation is performed on the finer level in a direction *perpendicular* to the coarse-fine interface using data from the finer level as well as the intermediate value from the first interpolation. In the event that there are not enough valid cells on the coarse level to use the regular centred interpolation stencil, the stencil is shift by one coarse grid cell, as shown by the lower stencil in Figure 2.6. In the rare case that there are not enough valid coarse grid cells to use the shifted stencil then the order of the interpolation is dropped. We will denote the above **coarse-fine interpolation** procedure for a variable ϕ to fill in ghost cells on level l using data from level $l-1$ as

$$\phi^l = I(\phi^{(l-1)}, \phi^{(l)}) \text{ on } \partial\Omega^{(l)}. \tag{2.80}$$

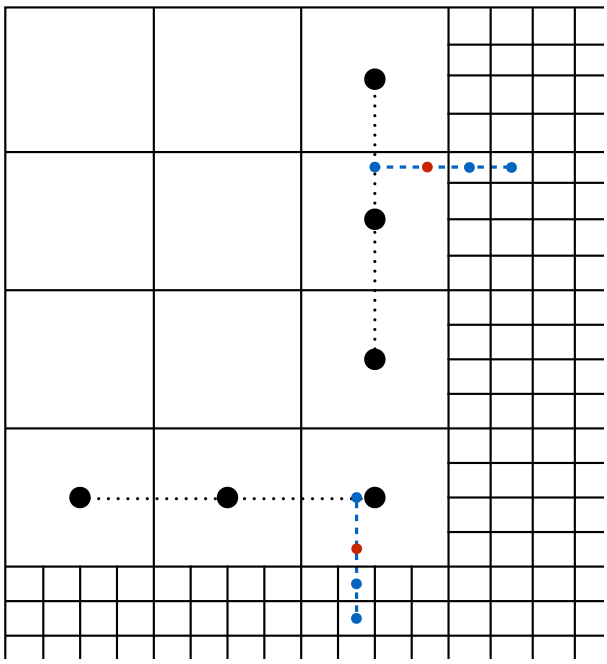

Figure 2.6: An example of the quadratic interpolation stencils that are used to fill in ghost cells on the finer level at the coarse-fine interface. The upper stencil shows the most typical case. First data on the coarser level (black dots) is used to interpolate parallel to the coarse-fine interface to obtain an intermediate value (the blue dot on the dotted line). Next a second quadratic interpolation is performed using fine level data and the intermediate value from the first interpolation to fill in the ghost cell (red dot) of the finer level. The lower grouping of grid cells shows and example of a case when the first quadratic interpolation on the coarser level would be shifted as there are not enough valid coarse grid cells to use a centred stencil.

To compute the **level gradient**, $G^{(l)}\mathbf{u}^{(l)}$, of a variable $\mathbf{u}$ on level l, the same procedure described for the composite gradient is used, with the exception that $G^{(l)}$ is computed on *all* edges of level l, not just the valid ones.

The Laplacian

The Laplacian is straightforward to define for both level and composite variables as the divergence of the gradient:

$$L^{\text{comp}}\mathbf{u} = D^{\text{comp}}\left(G^{\text{comp}}\mathbf{u}\right) \quad \text{on} \bigcup_{l\geq 0}\Omega^{(l)}_{\text{valid}} \tag{2.81}$$

$$L^{(l)}\mathbf{u}^{(l)} = D^{(l)}(G^{(l)}\mathbf{u}^{(l)}) \quad \text{on } \Omega^{(l)}. \tag{2.82}$$

Hence, on the interior of a level, the Laplacian reduces to the usual five-point stencil. At the coarse-fine interface however data from the coarser level will be used to interpolate to ghost cells on the finer level.

2.3.4 Martin-Cartwright Multigrid Algorithm

The hierarchical nature of the grids used in adaptive mesh refinement lends itself naturally to the use of geometric multigrid. There are two kinds of elliptic solves that we must concern ourselves with. The first involves solving for a level variable on a single level, while the second involves solving for a composite variable on the AMR hierarchy for levels $l \geq l_{base}$, where the base level l_{base} isn't necessarily the coarsest level.

Level Solve

To solve an elliptic problem on one level, the usual multigrid algorithm is used, with a few minor modifications. First, physical boundary conditions are used on the problem domain boundary, however at the coarse-fine interface data is interpolated from the coarser level. Next, the level solve ignores all other grids in the AMR hierarchy: at each stage in the V-cycle grids are coarsened by a factor of two regardless of the refinement ratio used to build the AMR hierarchy. Once the grids cannot be coarsened any further, the stabilized biconjugate gradient method is used to solve the residual equation. After the solution is

found on the coarsest level, this is used to correct the solution on the next finer level. A few passes of Gauss-Siedel are performed and then the correction continues up the multigrid hierarcy until the initial level is reached. This process is shown pictorially in Figure 2.7.

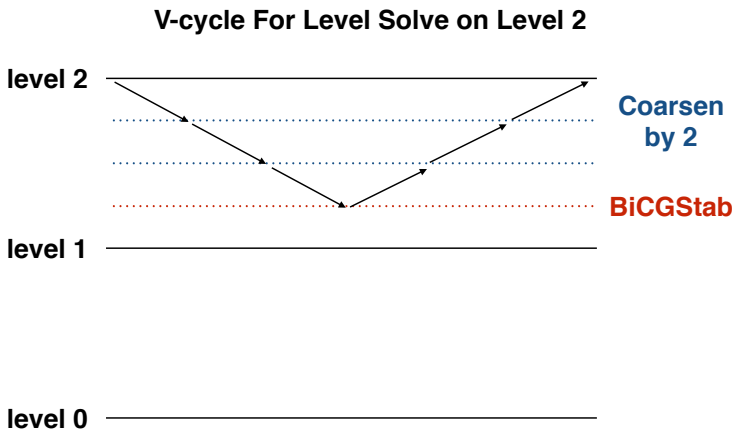

Figure 2.7: An example of the V-cycle used when solving for a level variable. In this case a variable is being solved for on the second level in the AMR hierarchy and the block factor is $2^3 = 8$. As this is a level solve, all coarser grids in the hierarchy are ignored when coarsening and are only used to provide the boundary data at a coarse-fine interface. At each stage in the multigrid solve grids are coarsened by a factor of 2, regardless of the refinement ratio used. Once the coarsest multigrid level is reach, the stabilized biconjugate gradient method is used.

Composite Solve

Martin and Cartwright [33] developed a cell-centred geometric multigrid algorithm for elliptic problems on an AMR hierarchy. The elliptic problem is solved on levels $l \geq l_{base}$, where l_{base} not necessarily the coarsest level. Just as in the usual multigrid algorithm, starting on the finest AMR level the initial residual for the problem is calculated. Next, a level V-cycle is performed, coarsening by a factor of two $\log_2 r^{(l)} - 1$ times before coarsening by a factor of $r^{(l)}$ down to the next AMR level. This process continues down the AMR hierarchy until level l_{base} is reached. At this point, the same algorithm described in the

previous section is used to solve the problem on level l_{base}. The solution from l_{base} is used to correct the solution on level $l_{\text{base}}+1$. Next, another level V-cycle is performed to smooth the solution before the correction process continues recursively up the AMR hierarchy until the finest level is reached. This situation is shown pictorially in Figure 2.8. AMR multigrid has been demonstrated to be fast and robust over a wide range of problems.

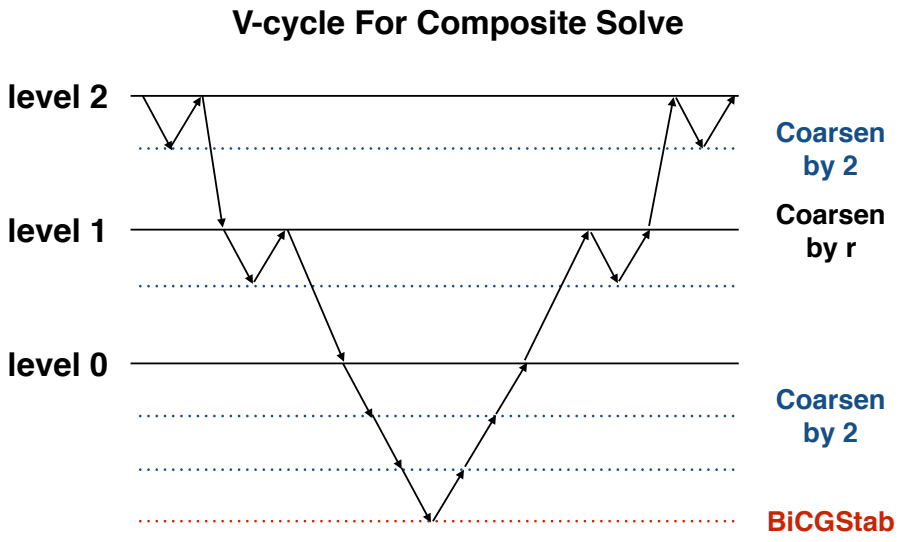

Figure 2.8: An example of the V-cycle used when solving for a composite variable. In this case, the base level is $l_{\text{base}} = 0$, the refinement ratio is 4, and the block factor is $2^3 = 8$. A level V-cycle is performed on level 2 before coarsening to level 1. This process continues on level 1 until level 0 is reached. On level 0, the same procedure described for the level solved is used, with the stabilized biconjugate gradient method being used on the coarsest multigrid level. At this point the solution on level 0 is used to correct level 1. After another level V-cycle is performed, level 2 is corrected, and a final V-cycle is performed.

2.3.5 Composite and Level Projection

The incompressibility condition can be enforced on each level in the AMR hierarchy using the same cell-centred approximate projection method described previously. The biggest difference now is that data must be interpolated at the coarse-fine interface to evaluate

the Laplacian. Additionally, we want to ensure that our velocity field satisfies the incompressibility condition in the *composite* sense as it is the composite version of our primitive variables that we have defined to be the solution.

Level Projection

If $\mathbf{V}^{(l),*}$ is a cell-centred level velocity, then we can define our **level projection** on level l by the following procedure:

1. Average $\mathbf{V}^{(l),*}$ from the cell centres to the cell faces.
2. Solve
$$L^{(l)}\phi^{(l)} = D^{(l)}\mathbf{V}^{(l),*} \text{ on } \Omega^{(l)} \tag{2.83}$$
for $\phi^{(l)}$, applying physical boundary conditions on the domain boundary and interpolating at the coarse fine interface.
3. Compute $G^{(l)}\phi$ at the cell-faces and average this to the cell centres.
4. Correct the original cell-centred level velocity
$$\mathbf{V}^{(l)} = \mathbf{V}^{(l),*} - G^{(l)}\phi^{(l)} \text{ on } \Omega^{(l)} \tag{2.84}$$

We will denote this procedure by $\mathbb{P}^{(l)}$. The solve in the second step is performed using multigrid on one AMR level. A **level MAC projection** for a face-centred velocity can be defined by omitting the first and last averaging steps and will be denoted by $\mathbb{P}^{(l)}_{\text{MAC}}$.

Composite Projection

We can similarly define a **composite projection** for a cell-centred composite velocity $\mathbf{V}^{\text{comp},*}$ by the following procedure:

1. Average $\mathbf{V}^{\text{comp},*}$ from the cell centres to the cell faces.
2. Solve
$$L^{\text{comp}}\phi^{\text{comp}} = D^{\text{comp}}\mathbf{V}^{\text{comp},*} \text{ on } \bigcup_{l \geq l_{\text{base}}} \Omega^{(l)}_{\text{valid}} \tag{2.85}$$
for ϕ^{comp}, applying physical boundary conditions on the domain boundary and interpolating at the coarse fine interface.

3. Compute $G^{\text{comp}}\phi$ at the cell faces and average this to the cell centres.

4. Correct the original cell-centred composite velocity

$$\mathbf{V}^{\text{comp}} = \mathbf{V}^{\text{comp},*} - G^{\text{comp}}\phi^{\text{comp}} \quad \text{on} \bigcup_{l \geq l_{\text{base}}} \Omega^{(l)}_{\text{valid}} \tag{2.86}$$

We will denote this procedure by $\mathbb{P}^{\text{pcomp}}$. The solve in the second step is performed using the Martin-Cartwright multigrid algorithm on all levels $l \geq l_{\text{base}}$.

2.3.6 Berger-Oliger Time-Stepping

As the resolution increases in the AMR hierarchy, so does the time-step restriction give by the CFL criteria (2.19) on each level. If all of the levels in the AMR hierarchy are stepped forward in time simultaneously using the time-step of the finest level, then a significant amount of extra work will be done on the coarser levels where a larger time-step could be used. It would be more efficient if instead the coarsest level was advanced first using a larger time-step, and then the finer levels were recursively advanced using successively finer time-steps. This kind of recursive time-stepping procedure for an AMR hierarchy is known as **subcycling**. A recursive procedure for solving hyperbolic problems on an AMR hierarchy was first introduced in [11] and then specialized for solving conservations laws of the form (2.1) using finite volume techniques in [10]. Without loss of generality, we will assume a fixed time-step here, however, in Section 2.4 we will allow a variable time-step.

Starting at level $l = 0$:

1. Update the level variable $\mathbf{U}^{(l)}$ using the conservative update (2.6):

$$\mathbf{U}^{(l),n+1} = \mathbf{U}^{(l),n} - \Delta t^{(l)}(D^{(l)}\,\mathbf{F}^{(l),n+\frac{1}{2}}) \quad \text{on } \Omega^{(l)} \tag{2.87}$$

2. If we are not on the finest level, then initialize the flux register for that level:

$$\delta\mathbf{F}^{(l+1)} = -\mathbf{F}^{(l),n+\frac{1}{2}} \cdot \hat{\mathbf{n}} \quad \text{on } \mathcal{C}_{r^{(l)}}\left(\partial\Omega^{(l+1)}\right) \tag{2.88}$$

3. If we are not on the coarsest level, then update the flux register on this level using the average flux:

$$\delta\mathbf{F}^{(l)} \to \delta\mathbf{F}^{(l)} + \tfrac{1}{r^{(l-1)}}\left\langle \mathbf{F}^{(l),n+\frac{1}{2}} \cdot \hat{\mathbf{n}} \right\rangle \quad \text{on } \mathcal{C}_{r^{(l-1)}}\left(\partial\Omega^{(l)}\right) \tag{2.89}$$

4. If we are not on the finest level, take $r^{(l)}$ time-steps of length $\Delta t^{(l+1)} = \frac{\Delta t^{(l)}}{r^{(l)}}$ using steps 1 to 4 on level $l+1$, else continue time-stepping using step 1 until we have reached the same time as the next coarsest level.

5. If we are not on the finest level, average the solution from the next finer level down to this level:
$$\mathbf{U}^{(l),n+1} = \left\langle \mathbf{U}^{(l+1),n+1} \right\rangle \quad \text{on } \mathcal{C}_{r^{(l)}} \left(\partial\Omega^{(l+1)} \right) \tag{2.90}$$

6. If we are not on the finest level, correct the solution on this level with the reflux divergence:
$$\mathbf{U}^{(l),n+1} \rightarrow \mathbf{U}^{(l),n+1} - \Delta t^{(l)} (D_R^{(l)} \, \delta\mathbf{F}^{(l)}) \quad \text{on } \Omega^{(l)} \tag{2.91}$$

We will refer to steps 5 and 6 above as **synchronization** as they force the update for the composite solution to be in discrete conservation form. A pictorial representation of this subcycling process can been seen in Figure 2.9. It is important to note here that at a

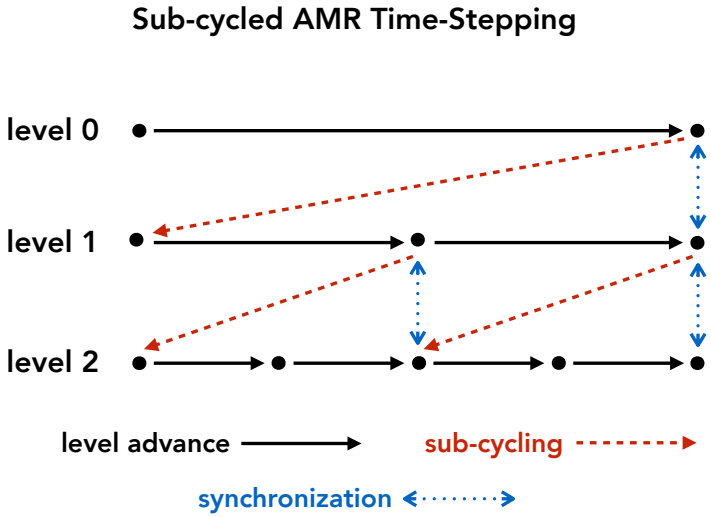

Figure 2.9: An example of sub-cycled AMR time-stepping for a hierarchy of two AMR levels and a refinement ratio of 2. First the coarsest level is advanced one time-step. Next, the first AMR level is advanced using a time step half that of the coarsest level. This recursive process continues to the finest level. The finest level, level 2, is then advanced until it reaches the same time as level 1. At this point levels 1 and 2 are synchronized. Level 1 is then advanced a second time-step, followed by advances on level 2. Finally once all three levels have reach the same time as level 0 they are collectively synchronized.

coarse-fine interface, data from the coarser level is interpolated in both space *and time* to fill in ghost cells on the finer level. This is in contrast to a purely elliptic problem where there is no time-dependence at the coarse-fine interface.

2.3.7 Freestream Preservation

In [31], it was found that $O(\Delta x^{(l)})$ errors that occur at the coarse-fine interface can contaminate the solution of a nearly constant scalar field on the interior of a level after successive regridding operations. To remedy this, an auxillary variable, Λ, was introduced to correct the advection velocity. Λ is a passive, conserved scalar

$$\frac{\partial \Lambda}{\partial t} + \nabla \cdot (\mathbf{u}\Lambda) = 0. \tag{2.92}$$

and is initialized to be unity everywhere. In theory, Λ should remain unity for all times, therefore $\Lambda - 1$ is a measure of how much the free stream has deviated. The elliptic problem

$$\nabla^2 e_\Lambda = \eta(\Lambda - 1) \tag{2.93}$$

is solved to find a correction to the advection velocity

$$\mathbf{u}_\Lambda^{\text{ADV}} = \mathbf{u}^{\text{ADV}} + \nabla e_\Lambda, \tag{2.94}$$

where $\eta > 0$ is a parameter which determines the strength of the correction. The boundary conditions for e_Λ are

$$\frac{\partial e_\Lambda}{\partial n} = 0. \tag{2.95}$$

The effect of this correction is to drive Λ back to unity. The elliptic problem (2.93) is solved in the composite sense for levels $l \geq l_{base}$ during the synchronization step. Additionally, the correction is recomputed after each regridding operation.

2.3.8 Viscous Terms

The TGA Scheme

In [32] it was determined that using the Crank-Nicholson scheme (2.41) as was done in [7] led to weak instabilities at the coarse-fine interface between levels which persisted throughout a simulation. In order to remedy this, a modified version of the second order

L_0-stable scheme presented by Twizell, Gumel, and Arigu (TGA) [51] was used. If we write the momentum equations in the form

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla^2 \mathbf{u} + \mathcal{N}(\mathbf{u}) \tag{2.96}$$

where $\mathcal{N}$ is the nonlinear and pressure gradient terms, then the modified TGA scheme presented in [32] is

$$\mathbf{u}^{n+1} = \left(I - c_1 \nu \nabla^2\right)^{-1} \left(I - c_1 \nu \nabla^2\right)^{-1} \left[\left(I + c_3 \nu \nabla^2\right) \mathbf{u}^n + \Delta t \left(I + c_4 \nu \nabla^2\right) \mathcal{N}^{n+\frac{1}{2}}\right] \tag{2.97}$$

where $\mathcal{N}^{n+\frac{1}{2}}$ is $\mathcal{N}$ evaluated at the half-time and

$$\begin{aligned} c_1 &= \tfrac{2a-1}{a+\sigma}\Delta t \quad c_3 = (1-a)\Delta t \\ c_2 &= \tfrac{2a-1}{a-\sigma}\Delta t \quad c_4 = \left(\tfrac{1}{2} - a\right)\Delta t \end{aligned} \tag{2.98}$$

where

$$a = 2 - \sqrt{2} + \epsilon \tag{2.99}$$

$$\sigma = \sqrt{a^2 - 4a + 2} \tag{2.100}$$

and ϵ is a small parameter chosen to be 10^{-8}. When used on an AMR hierarchy, data is interpolated in both space and time from a coarser level when necessary, and all inversions are solved using the multigrid algorithm described previously. We will denote the above process by

$$L^{\mathrm{TGA}}(\mathbf{u}) = \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} - \mathcal{N}^{n+\frac{1}{2}}. \tag{2.101}$$

In the AMR case, this solve will be a level solve and the Laplacian ∇^2 will be replaced by the level Laplacian $L^{(l)}$.

Implicit Refluxing

In the case where there is viscosity, the refluxing for the velocity is handled implicitly. The viscous flux generated by the TGA scheme is

$$\mathbf{F}^{\mathrm{TGA},(l)} = \frac{\nu}{\Delta t}\left(\mu_1 \mathbf{u}^{(l),n+1} + \mu_2 \mathbf{u}^{\dagger} + \mu_3 \mathbf{u}^{(l),n} + \Delta t^{(l)} \mu_4 \mathcal{N}^{(l),n+\frac{1}{2}}\right) \tag{2.102}$$

where $\mathbf{u}^\dagger$ is the solution the intermediate problem

$$\left(I - c_1 \nu L^{(l)}\right) \mathbf{u}^\dagger = \left(I + c_3 \nu L^{(l)}\right) \mathbf{u}^{(l),n} + \Delta t \left(I + c_4 \nu L^{(l)}\right) \mathcal{N}^{(l),n+\frac{1}{2}}. \tag{2.103}$$

During the time-step on one level the flux registers are then updated as

$$\delta \mathbf{V}^{(l+1)} = -\left(\mathbf{u}^{\mathrm{AD},(l)} \cdot \hat{\mathbf{n}}\right) \mathbf{u}^{\mathrm{half},(l)} - \mathbf{F}^{\mathrm{TGA},(l)} \cdot \hat{\mathbf{n}} \tag{2.104}$$

on $\mathcal{C}_{r^{(l)}}\left(\partial\Omega^{(l+1)}\right)$ if we aren't on the coarsest level, and

$$\delta \mathbf{V}^{(l)} \to \delta \mathbf{u}^{(l)} + \frac{1}{r^{(l-1)}} \left\langle \left(\mathbf{u}^{\mathrm{AD},(l)} \cdot \hat{\mathbf{n}}\right) \mathbf{u}^{\mathrm{half},(l)} + \mathbf{F}^{\mathrm{TGA},(l)} \cdot \hat{\mathbf{n}} \right\rangle \tag{2.105}$$

on $\mathcal{C}_{r^{(l-1)}}\left(\partial\Omega^{(l)}\right)$ if we aren't on the finest level. Finally, during the synchronization step, an implicit problem is solved in the composite sense

$$(I - \Delta t^{l_{\mathrm{base}}} L^{\mathrm{comp}}) \delta \mathbf{u}^{\mathrm{comp}} = \Delta t^{(l)} D_R^{(l)} \delta V^{(l+1)} \tag{2.106}$$

for $l \geq l_{\mathrm{base}}$, and then the velocity is refluxed using this correction

$$\mathbf{u}^{(l)} \to \mathbf{u}^{(l)} + \delta \mathbf{u}^{(l)} \tag{2.107}$$

for $l \geq l_{\mathrm{base}}$.

2.4 Putting it All Together

Now we have all of the pieces necessary to layout the full AMR algorithm for the incompressible Navier-Stokes equations as in [32]. For the sake of clarity and ease of presentation we give the algorithm in the form of pseudocode. Note: the Initialization step is performed both at the beginning of a new simulation, and also after the regridding step.

Algorithm 1 Main

function MAIN
  Build Initial Grids
  Compute $\Delta t^{(0)}$, set $t^{(0)} = 0$
  Initialize(0,$t^{(0)}$)
  while $t^{(0)} < T_{\max}$ **do**
    $\Delta t^{(0)} = \text{TimeStep}(0, t^{(0)}, \Delta t^{(0)})$
    $t^{(0)} \to t^{(0)} + \Delta t^{(0)}$
  end while
end function

Algorithm 2 Berger-Oliger time stepping

function TIMESTEP(l,$t^{(l)}$,$\Delta t^{(l)}$)
  if $n == n_{\text{regrid}}$ **then**
    Regrid(l,$t^{(l)}$)
    $n = 0$
  end if
  LevelAdvance(l,$t^{(l)}$,$\Delta t^{(l)}$)
  Subcycle:
  if $l < l_{\max}$ **then**
    $\Delta t^{(l+1)} = \frac{\Delta t^{(l)}}{r^{(l)}}$
    while $t^{(l+1)} < t^{(l)}$ **do**
      $\Delta t^{(l+1)} = \text{TimeStep}(l+1, t^{(l+1)}, \Delta t^{(l+1)})$
      $t^{(l+1)} \to t^{(l+1)} + \Delta t^{(l+1)}$
    end while
  end if
  Synchronize:
  if $(t^{(l)} + \Delta t^{(l)}) < (t^{(l-1)} + \Delta t^{(l-1)})$ **then**
    Synchronize(l, $t^{(l)} + \Delta t^{(l)}$, $\Delta t^{(l)}$)
  end if
  Set n = n+1
  Compute $\Delta t^{(l)}$
  return $\Delta t^{(l)}$
end function

Algorithm 3 Level Advance

function LEVELADVANCE(l,$t^{(l)}$,$\Delta t^{(l)}$)

Compute the normal components of the velocity, and MAC project, $\mathbf{u}^{\mathrm{norm},(l)}$

Compute the scalar fluxes using the MUSCL scheme, $\mathbf{F}_s^{(l)}$, $\mathbf{F}_\Lambda^{(l)}$ and advect

$$\begin{aligned} s^{(l),n+1} &= s^{(l),n} - \Delta t^{(l)} D^{(l)} \mathbf{F}_s^{(l)} \\ \Lambda^{(l),n+1} &= \Lambda^{(l),n} - \Delta t^{(l)} D^{(l)} \mathbf{F}_\Lambda^{(l)} \end{aligned}$$

Compute the tangential components of the velocity, and add MAC correction, $\mathbf{u}^{\mathrm{tan},(l)}$

Compute advection velocity by averaging normal and tangential velocities to cell centres. $\mathbf{u}^{\mathrm{AD},(l)}$

Correct advection velocity with the free-stream potential,

$$\mathbf{u}_\Lambda^{\mathrm{ADV},(l)} = \mathbf{u}^{\mathrm{ADV},(l)} + \nabla e_\Lambda^{(l)}$$

Advect the velocity field

$$\mathbf{u}^{(l),*} = \mathbf{u}^{(l),n} - \Delta t^{(l)} \left[(\mathbf{u}\cdot\nabla)\mathbf{u}\right]^{(l),n+\frac{1}{2}} - \Delta t^{(l)} (\nabla p)^{(l),n-\frac{1}{2}} + \Delta t^{(l)} L^{\mathrm{TGA}} \mathbf{u}^{(l),*}$$

Update flux registers :

if $l < l_{\max}$ **then**

$$\left.\begin{aligned} \delta s^{(l+1)} &= -\mathbf{F}_s^{(l),n+\frac{1}{2}} \cdot \hat{\mathbf{n}} \\ \delta \Lambda^{(l+1)} &= -\mathbf{F}_\Lambda^{(l),n+\frac{1}{2}} \cdot \hat{\mathbf{n}} \\ \delta \mathbf{V}^{(l+1)} &= -\left(\mathbf{u}^{\mathrm{AD},(l)} \cdot \hat{\mathbf{n}}\right) \mathbf{u}^{\mathrm{half},(l)} - \mathbf{F}^{\mathrm{TGA},(l)} \cdot \hat{\mathbf{n}} \end{aligned}\right\} \text{ on } \mathcal{C}_{r^{(l)}}\left(\partial\Omega^{(l+1)}\right)$$

end if

if $l > 0$ **then**

$$\left.\begin{aligned} \delta s^{(l)} &\to \delta s^{(l)} + \tfrac{1}{r^{(l-1)}} \left\langle \mathbf{F}_s^{(l),n+\frac{1}{2}} \cdot \hat{\mathbf{n}} \right\rangle \\ \delta \Lambda^{(l)} &\to \delta \Lambda^{(l)} + \tfrac{1}{r^{(l-1)}} \left\langle \mathbf{F}_\Lambda^{(l),n+\frac{1}{2}} \cdot \hat{\mathbf{n}} \right\rangle \\ \delta \mathbf{V}^{(l)} &\to \delta \mathbf{u}^{(l)} + \tfrac{1}{r^{(l-1)}} \left\langle \left(\mathbf{u}^{\mathrm{AD},(l)} \cdot \hat{\mathbf{n}}\right) \mathbf{u}^{\mathrm{half},(l)} + \mathbf{F}^{\mathrm{TGA},(l)} \cdot \hat{\mathbf{n}} \right\rangle \end{aligned}\right\} \text{ on } \mathcal{C}_{r^{(l-1)}}\left(\partial\Omega^{(l)}\right)$$

end if

Level Project to find the divergence free velocity and new pressure gradient

$$\begin{aligned} \mathbf{u}^{(l),n+1} &= \mathbb{P}^{(l)} \left[\mathbf{u}^{(l),*} + \Delta t^{(l)} (\nabla p)^{n-\frac{1}{2}}\right] \\ (\nabla p)^{(l),n+\frac{1}{2}} &= \tfrac{1}{\Delta t^{(l)}} \left(\mathbb{I} - \mathbb{P}^{(l)}\right) \left[\mathbf{u}^{(l),*} + \Delta t^{(l)} (\nabla p)^{n-\frac{1}{2}}\right] \end{aligned}$$

end function

Algorithm 4 Synchronize

function SYNCHRONIZE(l_{base},t^{sync},Δt^{sync})

Refluxing:

for $l = l_{\max} - 1$ to $l_{\text{base}} - 1$ **do**

$$\begin{aligned} s^{(l)} &\rightarrow s^{(l)} - \Delta t^{(l)} D_R^{(l)} \delta s^{(l+1)} \\ \Lambda^{(l)} &\rightarrow \Lambda^{(l)} - \Delta t^{(l)} D_R^{(l)} \delta \Lambda^{(l+1)} \end{aligned}$$

end for

Implicit Refluxing:

Solve:

$$(I - \Delta t^{\text{base}} \nu L^{\text{comp}}) \delta \mathbf{u}^{\text{comp}} = \Delta t^{(l)} D_R^{(l)} \delta V^{(l+1)} \quad \text{for } l \geq l_{\text{base}}$$

then correct

for $l = l_{\max} - 1$ to $l_{\text{base}} - 1$ **do**

$\mathbf{u}^{(l)} \rightarrow \mathbf{u}^{(l)} + \delta \mathbf{u}^{(l)}$

end for

Sync Projection:

Solve:

$$L^{\text{comp}} e_{\text{sync}} = D^{\text{comp}} \mathbf{u}^{\text{comp},*} \quad \text{for } l \geq l_{\text{base}}$$

then correct

$$\mathbf{u}^{\text{comp}} = \mathbf{u}^{\text{comp},*} - \Delta t^{\text{sync}} D^{\text{comp}} e_{\text{sync}}$$

Freestream presevation

Solve:

$$L^{\text{comp}} e_\Lambda = \frac{\eta}{\Delta t^{\text{sync}}} (\Lambda^{\text{comp}} - 1) \quad \text{for } l \geq l_{\text{base}}$$

then compute the freestream correction:

$$\mathbf{u}_p = G^{\text{comp}} e_\Lambda$$

end function

Average solution down from finer levels

for $l = l_{\max} - 1$ to $l_{\text{base}} - 1$ **do**

$$\left.\begin{aligned} \mathbf{u}^{(l)} &= \left\langle \mathbf{u}^{(l+1)} \right\rangle \\ s^{(l)} &= \left\langle s^{(l+1)} \right\rangle \\ \Lambda^{(l)} &= \left\langle \Lambda^{(l+1)} \right\rangle \end{aligned}\right\} \quad \text{on } \mathcal{C}_{r^{(l)}}(\Omega^{(l+1)})$$

end for

Algorithm 5 Regrid

function REGRID(l_{base},t_{base})
Tag cells
Build new grids using Berger-Rigtous algorithm
Average/Interpolate primitive variables from old grids to new grids
Initialize(l_{base},t_{base})
end function

Algorithm 6 Initialize/PostRegrid

function INITIALIZE(l_{base},t^{init})
Project the velocity field
Solve:

$$L^{\text{comp}} e_{init} = D^{\text{comp}} \mathbf{u}^{comp,*} \quad \text{for } l > l_{\text{base}}$$

then correct

$$\mathbf{u}^{\text{comp}} = \mathbf{u}^{\text{comp},*} - G^{\text{comp}} e_{\text{init}}$$

Initialize freestream correction:
Solve:

$$L^{\text{comp}} e_{\Lambda} = \frac{\eta}{\Delta t^{\text{base}}} (\Lambda^{\text{comp}} - 1) \quad \text{for } l \geq l_{\text{base}}$$

then compute the free stream correction

$$\mathbf{u}_p = G^{\text{comp}} e_{\Lambda}$$

Initialize the pressure gradient:
Set $p^{(l)} = 0$ for $l > l_{\text{base}}$
Set $\Delta t^{\text{init}} = \frac{1}{2} \Delta t^{(l_{\max})}$
for $l = l_{\text{base}}$ to $l_{\max}$ **do**
Compute $\mathbf{u}^{(l),*}$ as in a normal time-step
Set $\mathbf{V}^* = \mathbf{u}^{(l),*} + \Delta t^{\text{init}} G^{(l)} p^{(l)}$
Solve:

$$L^{(l)} p^{(l)} = D^{(l)} V^*$$

and then correct

$$\mathbf{u}^{(l)} = \mathbf{V}^* - \Delta t^{\text{init}} G^{(l)} p^{(l)}$$

end for
end function

Chapter 3

Results

3.1 The AMRINS Code

The AMRINS code used for all of the simulations is included with the standard Chombo repository. Upon first use the performance of AMRINS was less than satisfactory. The first issue that needed to be addressed was severe memory leakage. The initial version of the code crashed on the computer system Orca after running out of memory. Intel Inspector was used to profile the code's memory usage and find a number of memory leaks. Most of these leaks were cause by hanging pointers whose allocated memory was not correctly deallocated during regridding operations.

The second issue that needed to be addressed was the poor performance of the multigrid solvers. It would often take 15-30 iterations for the solvers to converge, and both the MAC and level projection solvers would hang frequently. Initially the solver used at the bottom of the multigrid V-cycle simply used a few iterations of Gauss-Siedel relaxation to try and solve the residual equation. If a small block factor was used, then the level solvers would often not converge. An example of this performance is shown in Figure 3.1 for a typical level solve using a block factor of 8. Using a larger block factor (64 for example) remedies this because a large block factor ensured the multigrid solver could coarsen a sufficient number of times to render the solver on the bottom level trivial. A deeper inspection of the residuals at each level during a V-cycle confirmed this. An example of this performance is shown in Figure 3.2 for a typical level projection solve using a block factor of 64.

Unfortunately it is not always optimal, or even possible in three dimensions, to use a large block factor. To remedy the lack of convergence for smaller block factors, the Gauss-Siedel solver was replaced with a stabilized biconjugate gradient solver. After this, the

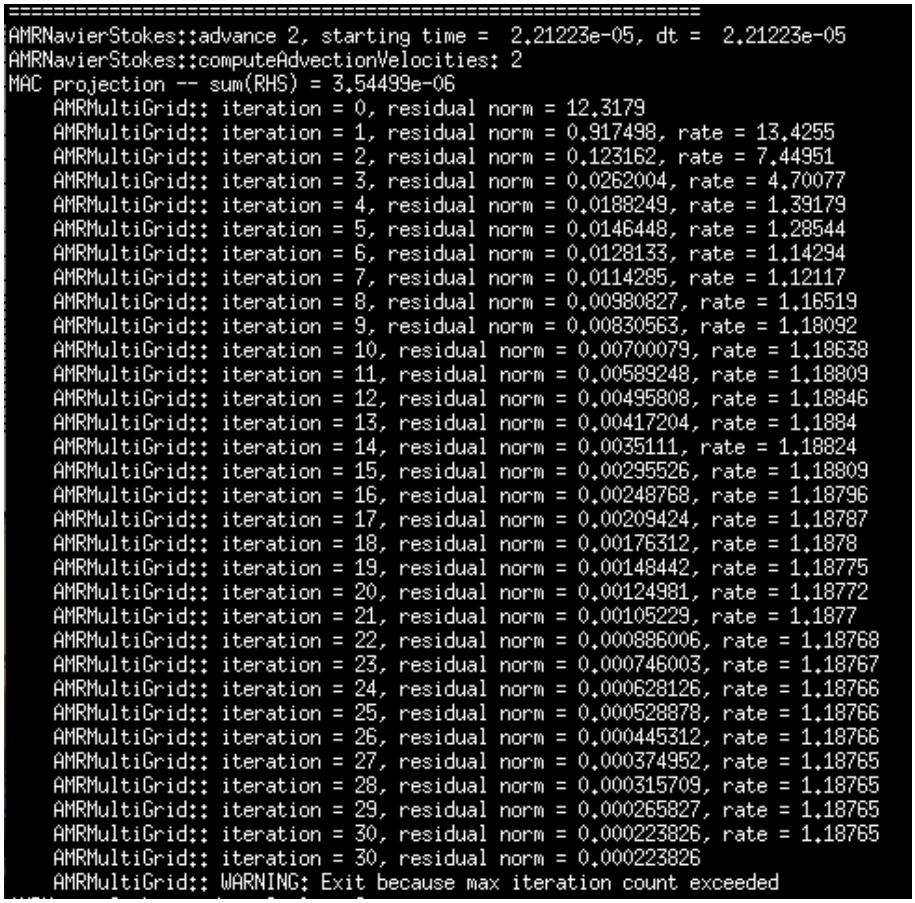

Figure 3.1: A typical level projection solve using a block factor of 8 and the Gauss-Seidel relaxation bottom solver.

various multigrid parameters, such as the number of smoothings between V-cycles, and number of preconditioner iterations was thoroughly tuned on a solver-by-solver basis for each of the multigrid solves in the code. All of this effort ensures that we can be confident that the code is converging as expected. An example of this performance is shown in Figure 3.3 for a typical level projection solve using a block factor of 8 after the performance tuning.

Lastly, Intel VTune Amplifier was used to performance profile the code for a few of the simulations conducted for scale testing. From this profiling it was determined that the AMRINS code is well optimized and that no further performance gains could be obtained without significant changes to the code. The vast majority of the computational time

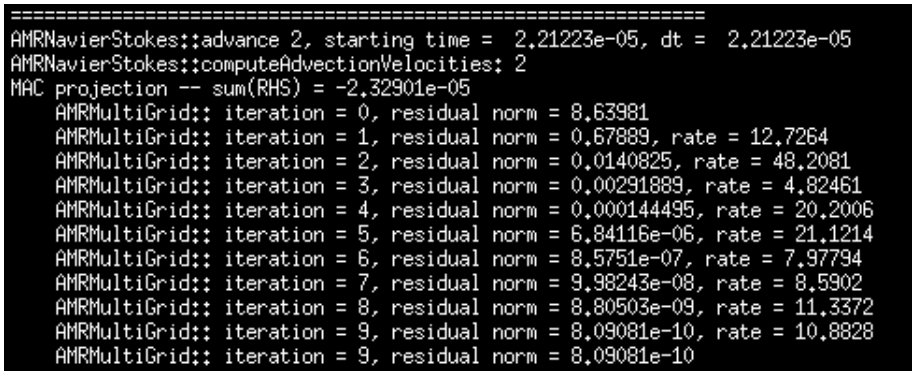

Figure 3.2: A typical level projection solve using a block factor of 64 and the Gauss-Seidel relaxation bottom solver.

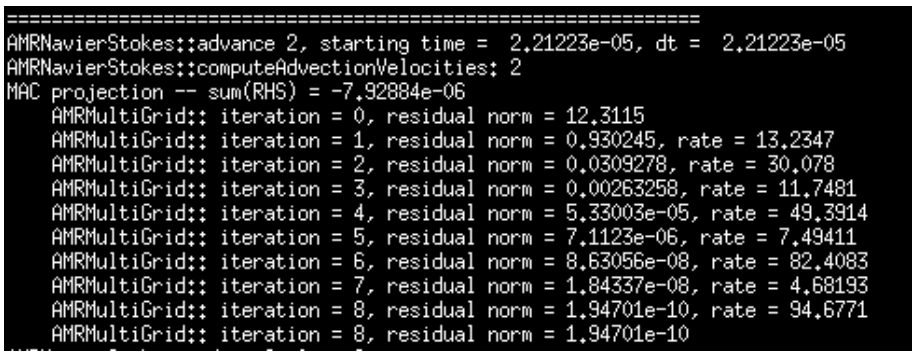

Figure 3.3: A typical level projection solve using a block factor of 8 and the BiCGStab bottom solver.

during each time-step is spent on the multigrid solves and, to a lesser extent, the advection calculations. This is very typical for computational fluid dynamics software.

3.2 Scale and Performance Testing

As discussed in the opening chapter, it is important to understand the strengths and limitations of any software in a HPC environment. Knowing the potential scalability of a code, and the optimal runtime parameters, will allow one to be a good steward of available computational resources and not consume CPU time unnecessarily. Given the plethora of grid parameters available in the AMRINS code, such as the maximum grid size, number

of AMR levels, and the grid efficiency, it was not immediately clear how to choose the parameters to obtain optimum performance. To address this, a test problem involving a translating dipole was chosen and the code was run with a small number of fixed time steps to explore the space of possible grid parameters. It should be emphasized that the purpose of these numerical experiments is to determine the optimal grid generation parameters, and *not* to rigorously demonstrate the scalability of the Chombo library. In [52] the Chombo library is shown to have good weak scalability for up to 131,000 processors for both a hyperbolic gas dynamics code, and a matrix-free geometric multigrid solver for an elliptic problem.

3.2.1 Test Problem

Clercx and Bruneau [18] suggest a problem involving the interaction of a two-dimensional vortex dipole with a solid wall as a difficult benchmark for incompressible Navier-Stokes solvers. The computational domain is taken to be a 2×2 square and the initial conditions are given by a pair of shielded monopoles of opposite circulation in close proximity. The monopoles are initialized as

$$\omega(x, y, 0) = \omega_0 \left(1 - \frac{r^2}{r_0^2}\right) e^{-r^2/r_0^2} \tag{3.1}$$

where the initial strength of each monopole is chosen to be $\omega_0 = \pm 300$, r is the radial distance from the centre of the monopole, and the initial radius of the monopole is chosen to be $r_0 = 0.1$. The monopole centres are placed a distance of $b = 2r_0$ and at a height of $h = 10r_0$ from the bottom wall of the domain. These initial conditions can be seen in the top panel of Figure 3.4. After the start of the simulation the monopole pair sheds the shielding and translates downwards, impacting the bottom wall of the domain. This situation can be seen in the bottom panel of Figure 3.4.

With this choice of initial conditions, the initial root-mean-square velocity is $U_{\text{rms}} \approx 1$, and the integral-scale Reynolds number can increased by lowering the kinematic viscosity. For the performance testing, an integral Reynolds number of $Re_{\mathcal{I}} = 2500$ was choosen. In comparison, the initial circulation around the inner part of one of the shielded monopoles can be calculated to be approximately $\Gamma_0 \approx 105$. This gives a vortex Reynolds number of

$$Re_\Gamma = \frac{\Gamma_0}{\nu} \approx 105 Re_{\mathcal{I}} \approx 2.626 \times 10^5. \tag{3.2}$$

As reported in [18], it is difficult to estimate the initial translation speed of the dipole pair as it does not completely form until around $t = 0.1$; however, the dipole collides with the

wall for times in the range $0.32 < t < 0.37$, depending on the Reynolds number, so we can estimate the translation speed to be on the order of $U_s \approx \frac{1}{1/3} = 3$. This gives a Reynolds number of

$$Re = \frac{2U_s r_0}{\nu} \approx \frac{6}{10\nu} = \frac{3}{5} Re_{\mathcal{I}} \approx 1500. \tag{3.3}$$

Four sets of simulations were carried out, each with increasing resolution using the same initial setup. In each set of simulations, timings for a single grid simulation are compared to the simulations using one, two, or three levels of AMR: where the resolution on the finest level is the same as the resolution of the single grid resolution. The same fixed time-step is used on the finest level of each simulation to ensure that every simulation ends at the same final time. For a given number of AMR levels, multiple simulations are carried out using different maximum AMR grid sizes and numbers of processors. The refinement ratio is always taken to be 4, as it was shown in [31] that, for a given resolution, a refinement ratio of 2 is usually less efficient due to the needed extra level of refinement and costliness of the synchronization operations. Additionally, the multigrid solvers are generally less efficient for a refinement ratio greater than 4.

3.2.2 Results

All of the simulations for performance testing were run on the system Orca described in Section 1.4.

For the first set of simulations, the resolution of the finest level was chosen to be $\Delta x = \frac{2}{2048}$. A stable fixed time-step of $\Delta t = 2.0 \times 10^{-5}$ was used on the finest level and 160 time-steps were taken on the finest level of each simulation. The maximum AMR grid size was varied between 128^2 grid cells and 1024^2 grid cells, however, maximum grid sizes in the range 256^2 to 512^2 proved to be the most effect. Results are shown in Figures 3.5 and 3.6 for maximum grid sizes in the latter range.

Figure 3.5 shows the wall-clock timings against the number of cores for simulations using a single grid, one, and two levels of AMR. The different colour curves correspond to different maximum grid sizes. The AMR runs finish in roughly the same amount of time as the single grid run using far fewer cores. The runs using a maximum grid size of 256^2 or 384^2 perform better than those using a maximum grid size of 512^2.

Figure 3.6 shows the *relative speedup* of the code againt the number of cores. The speedup is calculated relative to the lowest number of cores for each set of simulations. The relative speedup was used as it was not a practical use of resources to run simulations on a single core. The black dashed line shows the slope of the *perfect speedup* curve. It can

be seen that the single grid runs scale very well, while the AMR runs do not. This is due to the fact that the AMR runs are already achieving a very low wall-clock time on just 4 or 8 processors.

The minimum wall-clock time for each maximum grid size and number of AMR levels is compiled in Table 3.1 along with the number of cores that were used. Additionally, the amount of CPU time used has been calculated. For this first series of simulation, the single grid case using a maximum grid size of 256^2 finished in the least amount of time, 90 seconds, using 64 cores. The next lowest wall-clock timing was achieved by the simulation using 2 levels of AMR, and a maximum grid size of 384^2 on 16 cores in 148 seconds. While this is significantly slower, the amount of CPU time used is substantially less: corresponding to a savings of 59%. Similarly, the run using 2 levels of AMR and a maximum grid size of 256^2 on 8 cores yields a savings of 79%.

AMR Levels	Max. Box Size	Cores	Wall-Clock Time (s)	CPU Time (h)
0	**256**	**64**	**90**	1.6
	384	64	178	3.16
	512	16	246	1.09
1	256	8	172	0.382
	384	16	165	0.733
	512	16	220	0.977
2	256	8	151	0.335
	384	16	148	0.657
	512	16	176	0.782

Table 3.1: Minimum wall-clock times for $\Delta x = \frac{2}{2048}$. The

For the next set of simulations, the resolution on the finest level was increased to $\Delta x = \frac{2}{4096}$. A stable fixed time-step of $\Delta t = 1.0 \times 10^{-5}$ was used on the finest level and 160 time-steps were taken on the finest level of each simulation. As before the maximum grid size, number of AMR levels, and number of cores was allowed to vary.

Figure 3.7 shows the wall-clock timings against the number of cores for simulations using a single grid, and one and two AMR levels. The runs using AMR are able to finish in roughly the same amount of wall-clock time as the fastest single grid run using far fewer cores. On this larger problem, using a maximum grid size of 512^2 appears to be more advantageous. Runs using a maximum grid size of 768^2 and 1024^2 were conducted, however they used significantly more wall-clock time than any of the simulations presented here.

Figure 3.8 shows the relative speedup against the number of cores for the same set of simulations. The relative speedup was again calculated relative to the minimum number of cores used for each series of simulations at a given maximum grid size and number of AMR levels. As before, the single grid simulations show very good scaling properties. The dashed line shows the slope of the perfect speedup curve. Those runs using AMR show very good speedup for lower numbers of cores, but then quickly cap out again. This is likely due to the fact that they are able to achieve a very low wall-clock time on far few cores than the singe grid case.

The minimum wall-clock time for each maximum grid size and number of AMR levels is compiled in Table 3.2 along with the number of cores that were used. Additionally the total CPU time used has been calculated. For this series of simulations the minimum wall-clock time was achieved using two AMR levels and a maximum grid size of 512^2 on 16 cores. This run also used the lowest amount of CPU time at just 1.41 hours. When compared to the fastest single grid, which used 6.72 CPU hours, this represents a savings of 79%.

AMR Levels	Max. Box Size	Cores	Wall-Clock Time (s)	CPU Time (h)
0	256	128	376	13.37
	384	64	496	8.82
	512	64	378	6.72
1	256	32	356	3.16
	384	32	370	3.29
	512	16	374	1.66
2	256	32	332	2.95
	384	16	330	1.47
	512	**16**	**317**	**1.41**

Table 3.2: Minimum wall-clock times for $\Delta x = \frac{2}{4096}$

For the next set of simulations, the resolution on the finest level was increased to $\Delta x = \frac{2}{8192}$. A stable fixed time-step of $\Delta t = 2.0 \times 10^{-6}$ was used on the finest level and 320 time-steps were taken on the finest level of each simulation. The maximum grid size, number of AMR levels, and number of cores was allowed to vary. The single grid runs in this set of simulations took over an hour and so were terminated as to not waste computer resources. Additionally, a set of simulations with 3 levels of AMR was added.

Figure 3.9 shows the wall-clock timings against the number of cores for simulations using up to 3 levels of AMR. For a given number of cores, runs using 2 or 3 AMR levels

performed better than those using just one. Using 3 AMR levels does not appear to have any advantage over using 2. It is likely because the features for this test case are highly localized and all of the AMR levels cover roughly the same geometric area in the domain. Hence there is not a significant reduction in the number of grid points that need to be updated. Additionally, the operations required for synchronization are expensive due to the elliptic solves. Overall it appears that using a maximum grid size of 512^2 achieves lower wall-clock times.

Figure 3.10 shows the relative speedup against the number of cores for the same set of simulations. The dashed lines once again show the slope of the perfect speedup curve. Unlike the previous sets of simulations, the AMR runs are continuing to achieve speedup for larger numbers of cores. For 2 AMR levels it appears that the relative speedup is at least doubled when the number of cores is quadrupled.

The minimum wall-clock time for each maximum grid size and number of AMR levels is compiled in Table 3.3 along with the number of cores that were used. Additionally the total CPU time used has been calculated. For this series of simulations the minimum wall-clock time was achieved using 2 AMR levels and a maximum grid size of 384^2 on 64 cores. This run used 23.86 CPU hours, which represents a savings of *at least* 81% over the equivalent single grid run.

AMR Levels	Max. Box Size	Cores	Wall-Clock Time (s)	CPU Time (h)
0	256	128	$>$ 1h	$>$ 128
	384	128	$>$ 1h	$>$ 128
	512	128	$>$ 1h	$>$ 128
1	256	128	1559	55.43
	384	128	1700	60.44
	512	32	1742	15.48
2	256	128	1407	50.02
	384	**64**	**1342**	**23.86**
	512	32	1451	12.9
3	256	128	1527	54.29
	384	64	1399	24.87
	512	64	1423	25.29

Table 3.3: Minimum wall-clock times for $\Delta x = \frac{2}{8192}$

For the last set of simulations, the resolution on the finest level was increased to $\Delta x =$

$\frac{2}{16384}$. A stable fixed time-step of $\Delta t = 2.5 \times 10^{-6}$ was used on the finest level and 256 time-steps were taken on the finest level of each simulation. The maximum grid size, number of AMR levels, and number of cores was allowed to vary. Again, no single grid runs were carried out due to the significant amount of time they required.

Figure 3.11 shows the wall-clock timings again the number of cores for simulations using up to 3 levels of AMR. It appears that this problem is large enough that the amount of wall-clock time used continues to decrease for large numbers of cores. Additionally, a maximum grid size of 512^2 appears to the optimal grid size for a given number of cores. Figure 3.12 shows the relative speedup for the same set of simulations, although it was less merit than for the previous simulations as there are fewer data points. The dashed lines again show the slope of the perfect speedup curve.

The minimum wall-clock time for each maximum grid size and number of AMR levels is compiled in Table 3.4 along with the number of cores that were used. Additionally the total CPU time has been calculated. For this series of simulations the minimum wall-clock time was achieved using 2 AMR levels and a maximum grid size of 512^2 on 128 cores. If the number of cores were doubled up to 256 it is very likely that lower wall-clock times could be achieved.

AMR Levels	Max. Box Size	Cores	Wall-Clock Time (s)	CPU Time (h)
1	256	128	3545	126.0
	384	128	2869	102.0
	512	128	1.8h	230.4
2	256	128	2929	104.14
	384	128	2437	86.65
	512	**128**	**1896**	**67.41**
3	256	128	2995	106.48
	384	128	2640	93.87
	512	128	2009	71.43

Table 3.4: Minimum wall-clock times for $\Delta x = \frac{2}{16384}$

3.2.3 Discussion

The results of the previous section have demonstrated a number of important aspects of the AMRINS code. The first is that, in all but the most trivial grid resolutions, adaptive

mesh refinement offers a significant cost savings over a single grid in terms of both wall-clock time and number of cores used. More succinctly, AMR requires significantly less CPU time. In most cases, using a maximum box size of 512^2 was the most advantageous. This became most evident for the very high resolution cases. Lastly, the code appears to scale well and a factor of at least 2 speedup can be achieved by quadrupling the number of cores used.

A corollary to the above is that for a given computer system, an AMR code can be used to perform simulations that would not be accessible with a single grid. This has important implications for scientists with access to limited computer resources. In a later chapter three dimensional simulations with a resolution equivalent to a single grid with 768x768x384 grid points will be performed using only 24 cores in 4 days on a single node system. A simulation of this size might typically require over 500 cores using a single grid on an HPC cluster.

The number of different computer architectures which are available is staggering. While traditional Cray-type HPC clusters have a large number of nodes with 1 or two processors per node, many commodity HPC clusters today are built using large chips with as many as 6 to 8 cores per processor, with large shared L2 and L3 caches. It is not realistic to expect that a software application will run optimally with the same parameters on both of these very different types of systems. It is not clear why a maximum grid size of 512 was optimal in our particular study. This could be due to any number of factors including various cache sizes and interconnect speeds. In light of this, it would be prudent for any application with as many parameters affecting performance as AMRINS to include a simple benchmarking test such as the one presented here which users could use to tune the code for a particular computer system.

While all of testing was done in two dimensions, the results will be extrapolated and a maximum grid size of $64^3 = 512^2$ will be used in three dimensions. In the future, similar three dimensional scale testing should be performed to determine optimal grid parameters for three dimensional simulations. Such a study could help differentiate between the effects of node interconnect and cache sizes. A 64^3 grid will occupy the same space in memory as a 512^2 grid, however it requires significantly more communication overhead. Thus if it were found that 64^3 is the optimal grid size for three dimensional simulations, then it could be concluded that the cache size is the limiting factor. However, if a larger grid size were more optimal in three dimensions, then it could be concluded that communication overhead is the limiting factor.

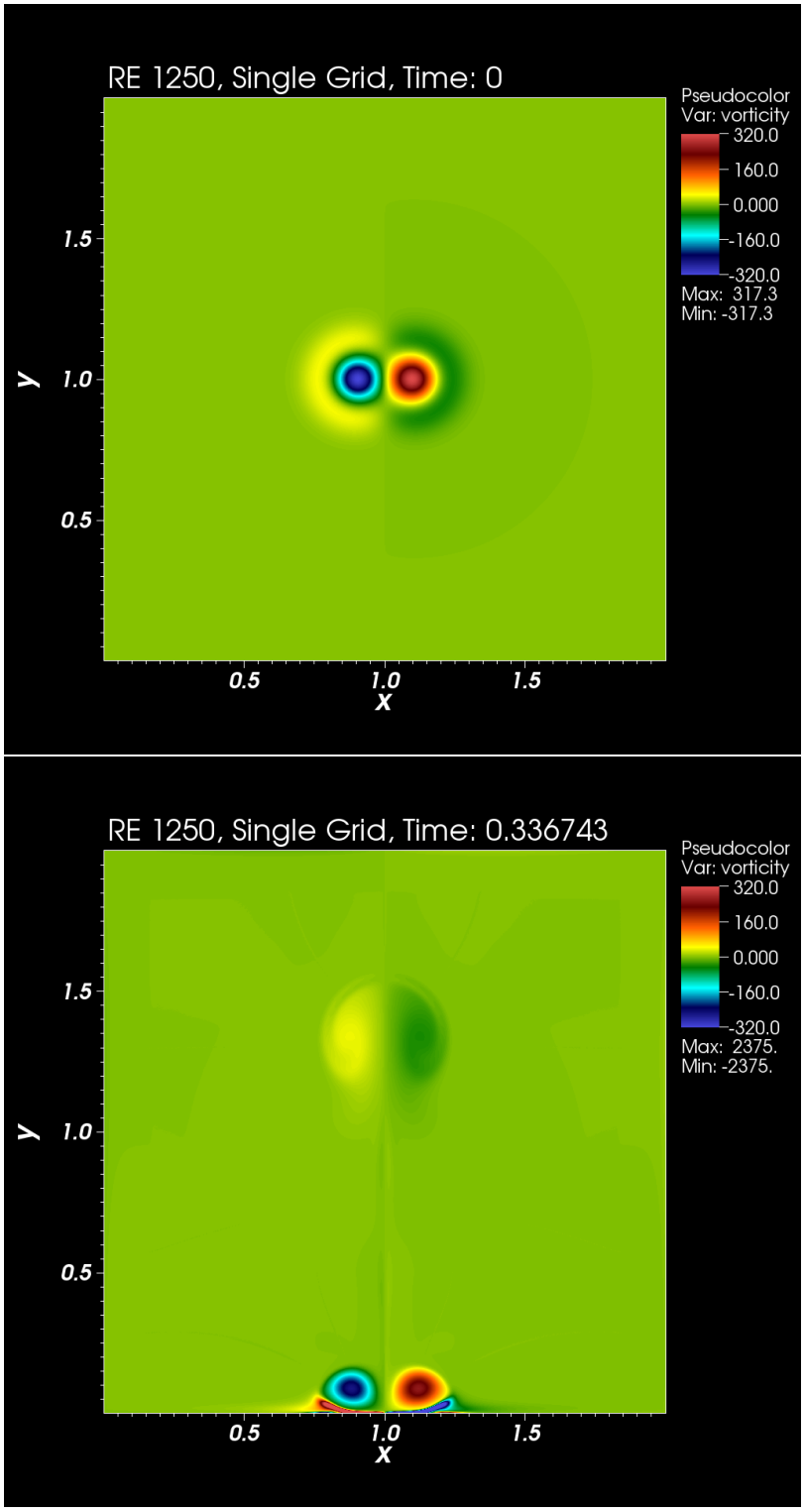

Figure 3.4: The top panel shows the initial vorticity field for the benchmark problem taken from Clercx and Bruneau [18]. After the start of the simulation, the vortex dipole begins propagating downwards, eventually interacting wit the bottom boundary as can be seen in the bottom panel.

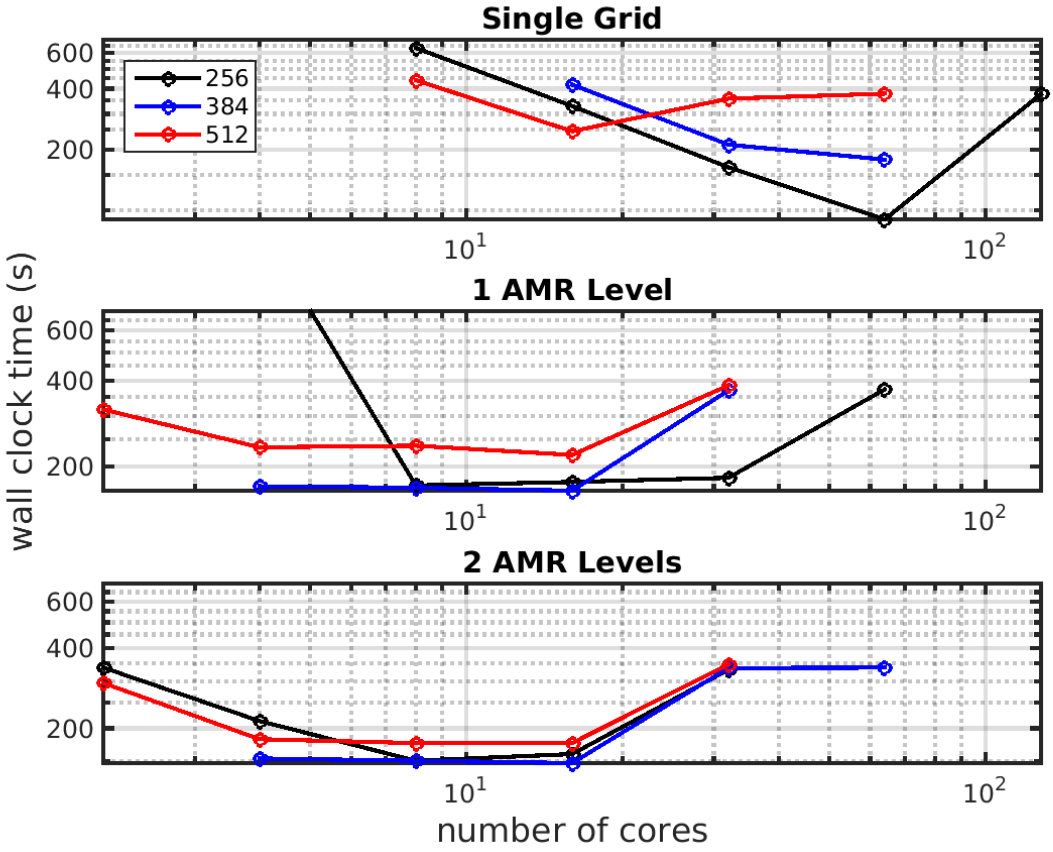

Figure 3.5: Wall-clock time vs. number of cores for $\Delta x = \frac{2}{2048}$. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. The runs with 1 and 2 levels of AMR achieve roughly the same wall-clock times as the single grid run using far fewer cores.

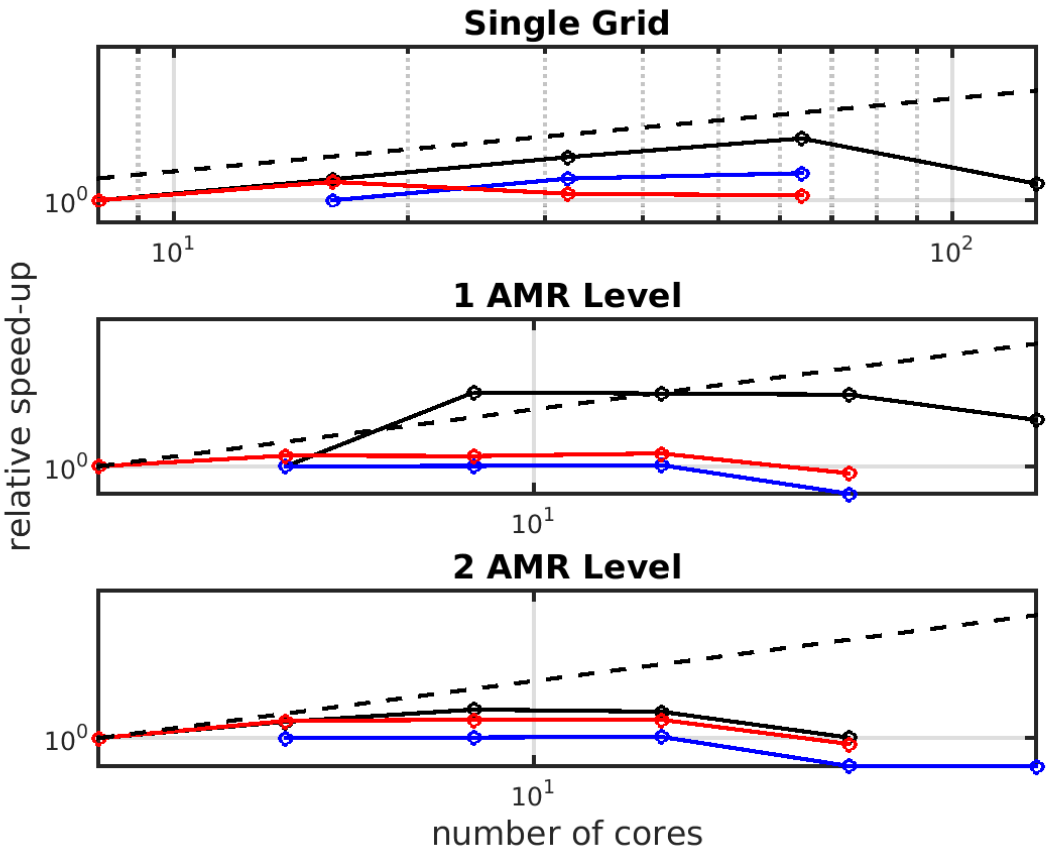

Figure 3.6: Relative speedup vs number of cores for $\Delta x = \frac{2}{2048}$. The speedup is calculated relative to the smallest number of cores used in each series of simulation as it was impractical to run these simulations on one core. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. The single grid runs achieve good scalability for for a maximum grid size of 256^2. The AMR runs however demonstrate poor speedup. This is due to the fact that they already achieve a low wall-clock time using a small number of cores.

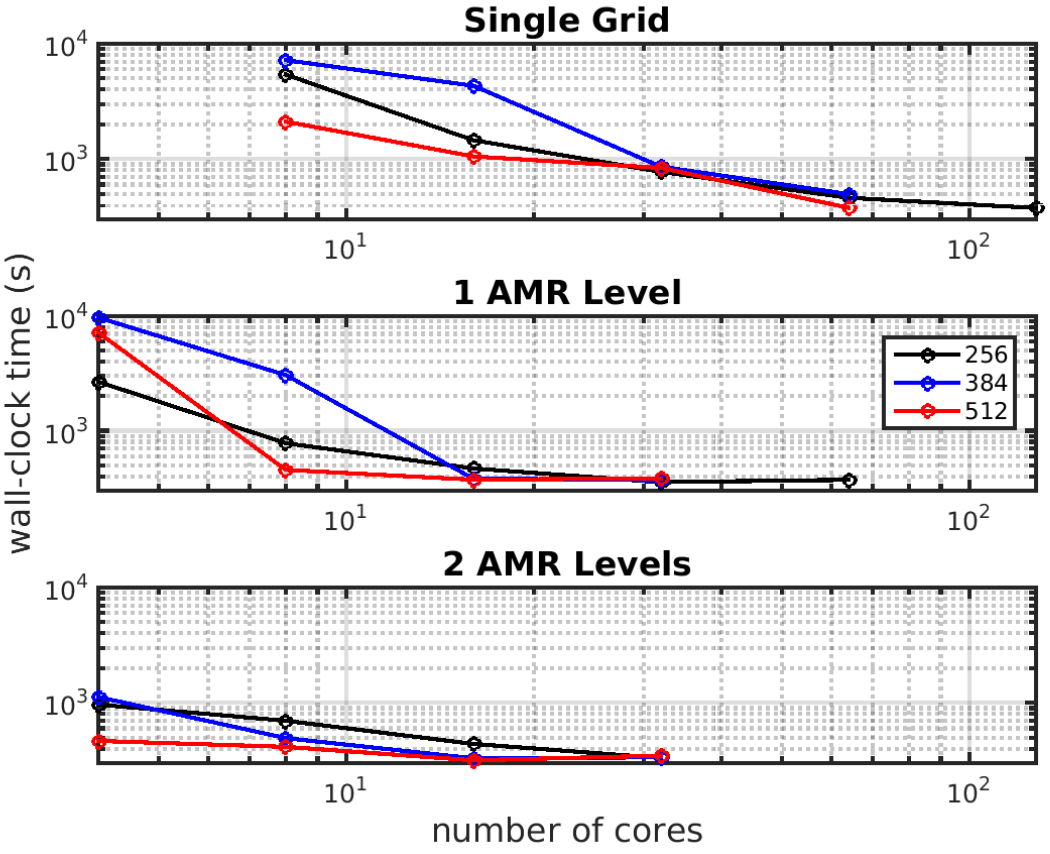

Figure 3.7: Wall-clock time vs. number of cores for $\Delta x = \frac{2}{4096}$. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. Again, the AMR runs achieve roughly the same wall-clock time as the single grid runs using far few cores.

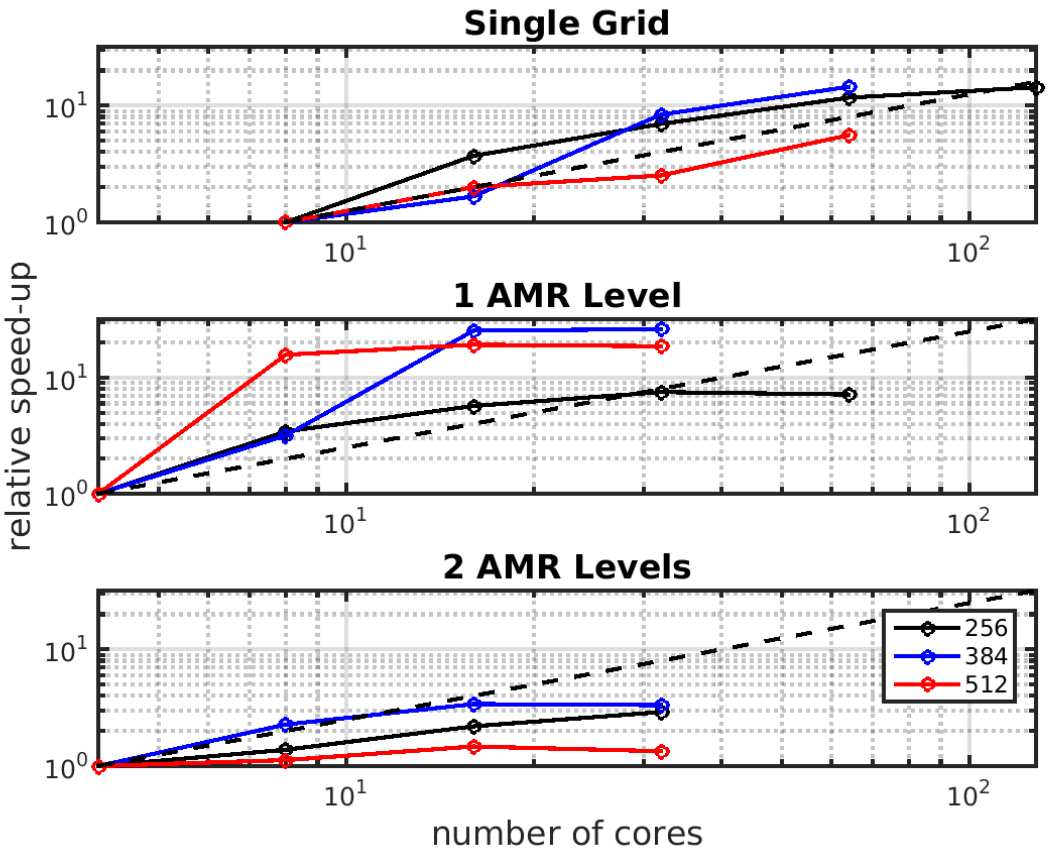

Figure 3.8: Relative speedup vs. number of cores for $\Delta x = \frac{2}{4096}$. The speedup is calculated relative to the smallest number of cores used in each series of simulations as it was impractical to run these simulations on one core. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. The single grid runs achieve good scalability for all numbers of cores, while the AMR runs only achieve good scalability for the small numbers of cores. This is again due to the fact that the AMR simulations quickly obtain a low wall-clock time on a small number of cores.

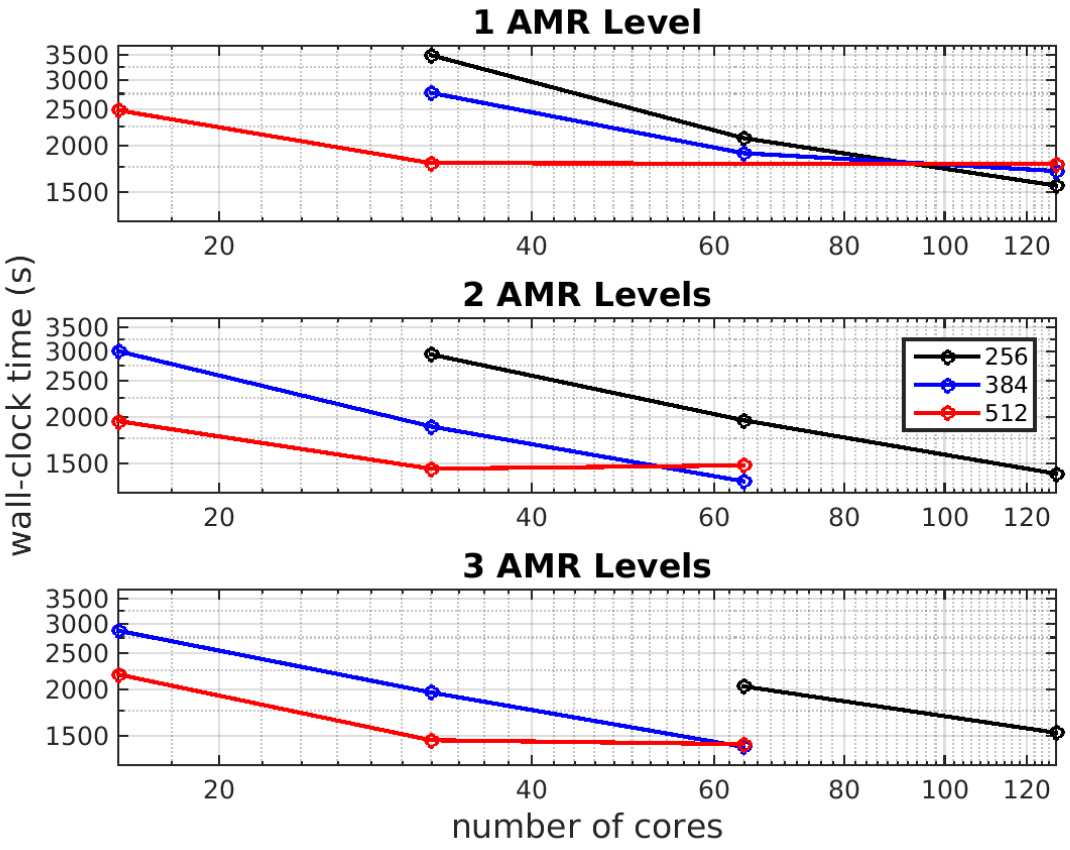

Figure 3.9: Wall-clock time vs. number of cores for $\Delta x = \frac{2}{8192}$. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. All single grid runs took over an hour of wall-clock time so they were terminated. Additionally, a series of runs with 3 levels of AMR were conducted. It appears that there is no gain from using 3 levels of AMR over 2 levels.

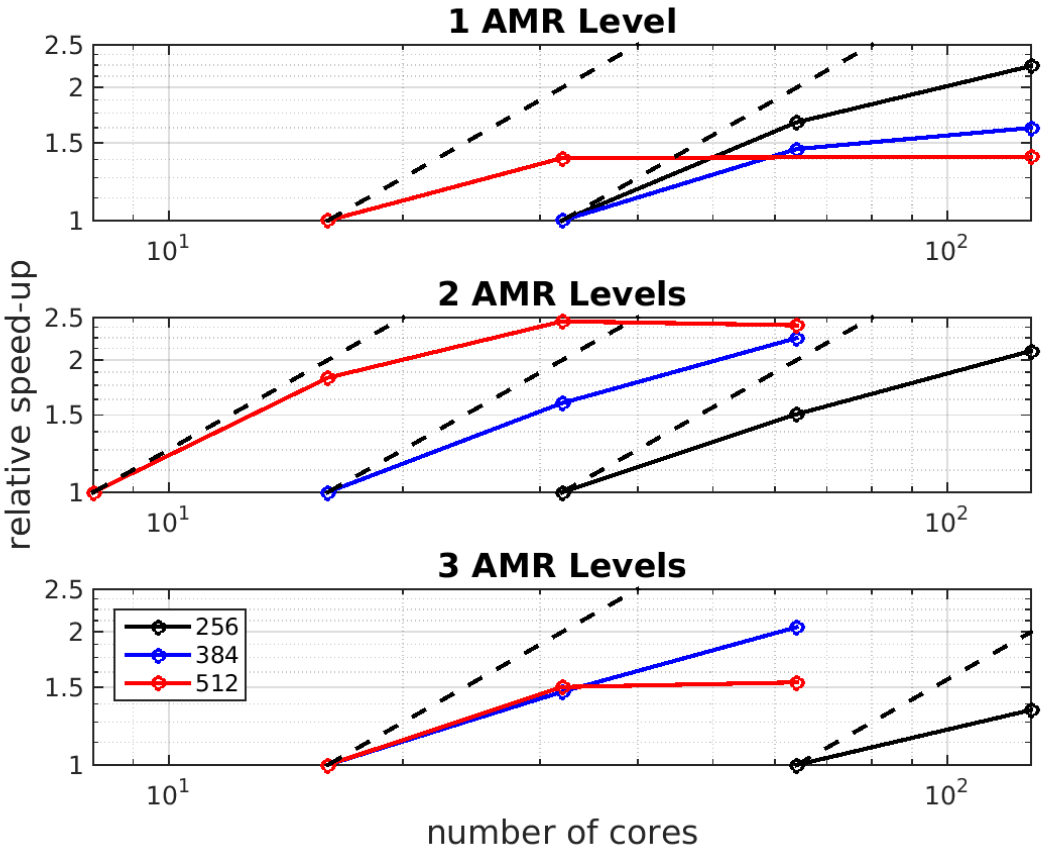

Figure 3.10: Relative speedup vs. number of cores for $\Delta x = \frac{2}{8192}$. The speedup is calculated relative to the smallest number of cores used in each series of simulation as it was impractical to run these simulations on one core. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. Most of the AMR simulations achieve a speedup of at least two by quadrupling the number of cores used.

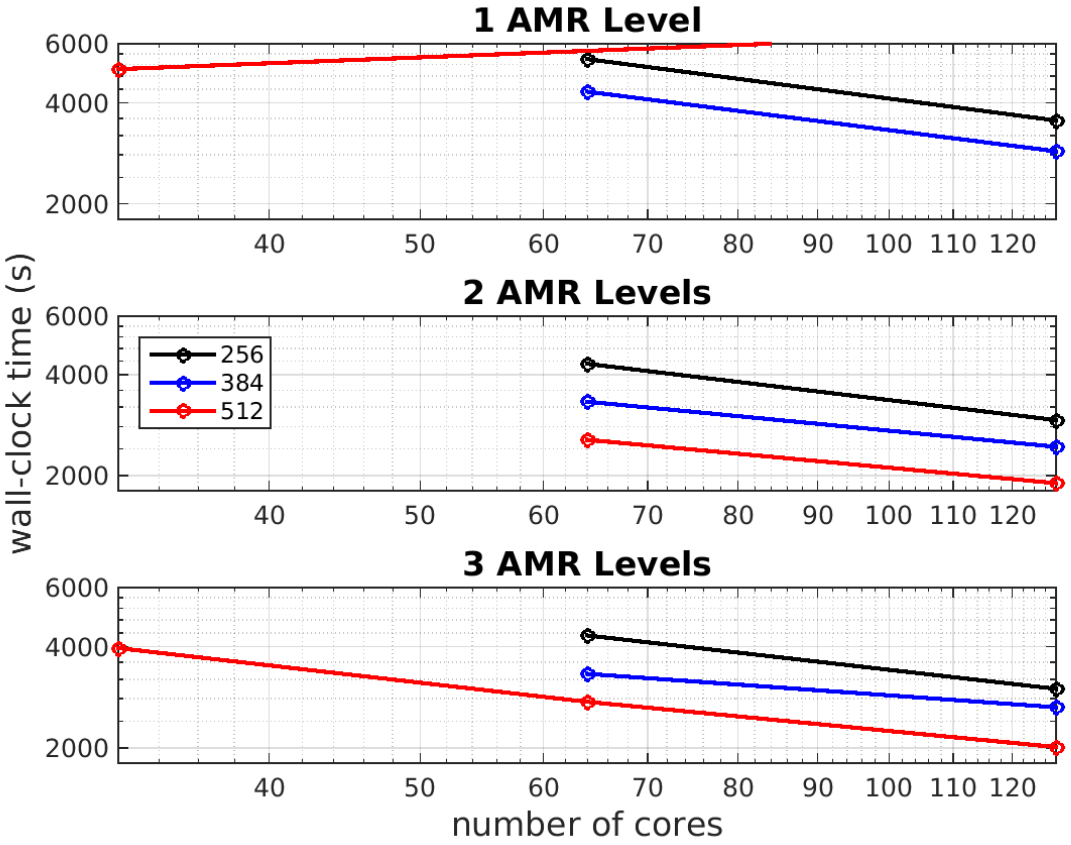

Figure 3.11: Wall-clock time vs. number of cores for $\Delta x = \frac{2}{16384}$. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. For all runs it using a maximum grid size of 512^2 appears to be the most optimal. Again, there is no advantage to using 3 levels of AMR over 2.

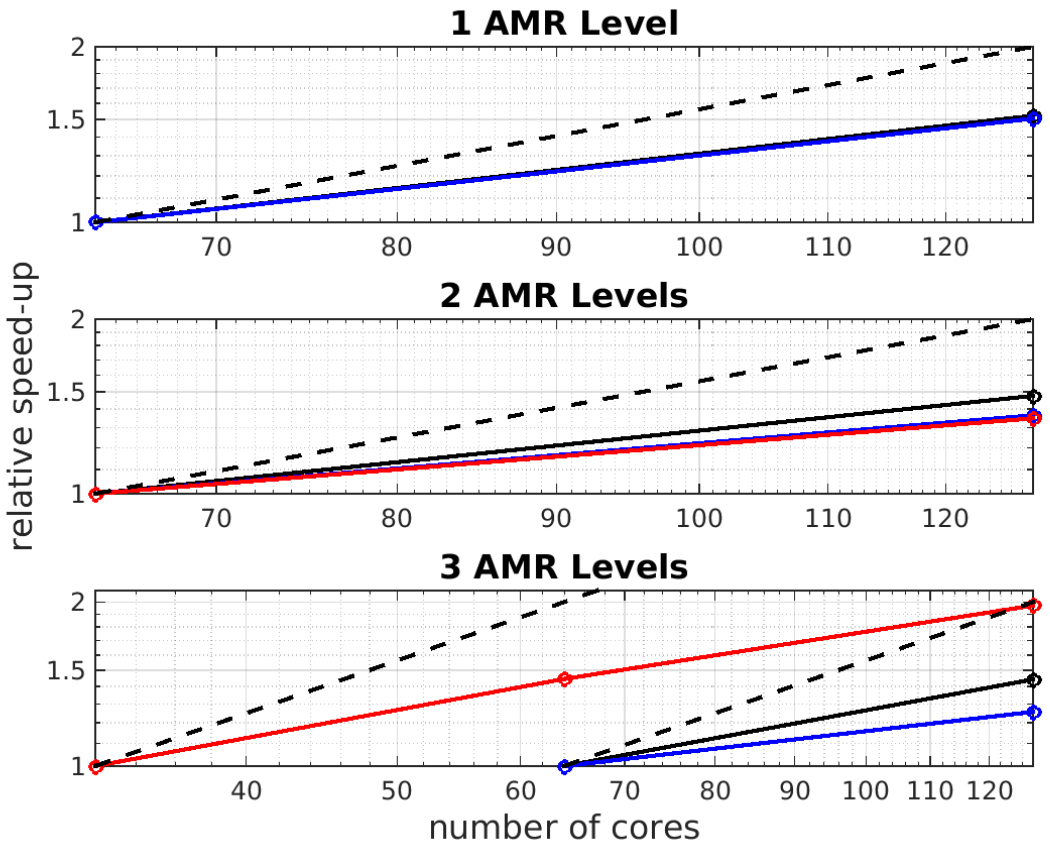

Figure 3.12: Relative speedup vs. number of cores for $\Delta x = \frac{2}{16384}$. The speedup is calculated relative to the smallest number of cores used in each series of simulation as it was impractical to run these simulations on one core. The black curve corresponds to a maximum grid size of 256^2, the blue curve to 384^2, and the red curve to 512^2. While the speedup curves are less meaningful as there are only 2 data points for many simulations, the 512^2 case using 3 AMR levels again shows a speedup of 2 when the number of processors is quadrupled.

3.3 Vortex-Dipole Wall Interactions

As mentioned in the previous section, Clercx and Bruneau [18] proposed the interaction of a two-dimensional vortex dipole with a solid boundary as a challenging computational benchmark for incompressible Navier-Stokes solvers. Initial conditions are given by 3.1 and the computational domain is taken to be a 2x2 square with no-slip boundary conditions. After initialization, the dipole propagates downward, shedding its shielding in the process, and finally interacting with the bottom boundary. With the given initial conditions the integral-scale Reynolds number is approximately $Re_{\mathcal{I}} = \frac{1}{\nu}$.

In [18] a finite difference solver based on the primitive variable form of the Navier-Stokes equations, (1.1) - (1.2), as well as a spectral collocation solver based on the vorticity equation (1.19) were used to carry out the simulations. Experiments were conducted over a broad range of Reynolds numbers, and at each Reynolds number the resolution was gradually increased until convergence was achieved. The total kinetic energy and total enstrophy time-series were analyzed, and the maximum values of enstrophy and the time at which they occurred were recorded. All of the metrics are intended for benchmarking purposes.

In a follow up study by Kramer and Clercx [28], two separate regimes were identified based on the integral-scale Reynolds numbers. For $Re_{\mathcal{I}} \lessapprox O(10^4)$, the dipole rebounds from the wall, creating vorticity of opposite sign in the boundary layer which rolls up into another weaker vortex. The newly formed vortices pair with their parent vortices to form new dipoles of unequal strength and recirculate to interact with the bottom boundary a second time. This process continues until dissipation occurs. For $Re_{\mathcal{I}} \gtrapprox O(10^4)$, a recirculation region forms, leading to an instability in the boundary layer and an eruption of smaller vortices.

While AMRINS has been tested with a number of simple benchmark problems in the past, this is the most rigorous benchmark it has been tested with, to date. Additionally, while the vortex-dipole benchmark has been used to test a number of different codes such as [45], [26] and [23], this is the first time we are aware of that it has been applied to a code using block-structured adaptive mesh refinement. While the wavelet based code in [23] also used resolution only where it is needed, the results had a wide margin of error in the maxima of the total enstrophy time-series. In [26] a spectral code which uses a penalization method on the boundaries appears to perform well against this benchmark.

3.3.1 Setup

To test the abilities of AMR, the vortex-dipole bench mark was conducted on a single-grid for integral-scale Reynolds numbers 1250, 2500, 5000, and $10,000$. Resolution was gradually increased at each Reynolds number to ensure grid convergence. The bench mark was then conducted for the same Reynolds numbers at the same resolutions using AMR. Two levels of AMR were used in each simulation with an refinement ratio of 4, and regridding was performed on each level every 2 time-steps. A maximum grid size of 512^2 was also used, as this was determined to be optimal in the previous section. Cells were tagged for refinement whenever $|\omega| > 7.5$. All of the simulations were conducted on the computer system "Orca" described in Section 1.4[1].

3.3.2 Results

Re 1250

For an integral-scale Reynolds number of 1250, the benchmark was run three times on a single grid using 2048×2048, 3072×3072, and 4096×4096 grid cells respectively. As computed in the previous section, the initial circulation Reynolds number is $Re_\Gamma \approx 105 Re_\mathcal{I} \approx 1.312 \times 10^5$. The top panel of Figure 3.4 shows the initial vorticity field. Each shielded monopole has a core which is surrounded by a shielding of opposite vorticity. The bottom panel shows the dipole which forms and propagates downwards just after it makes contact with the bottom boundary. Since we are not interested in the second weaker dipole which forms from the shielding, as its dynamics are similar to the stronger dipole, we will focus on the area directly surrounding the stronger dipole in the remaining figures.

Figures 3.13 and 3.14 show a time-series of the dipole interacting with the bottom boundary for the simulation using 4096×4096. Each half of the initial dipole induces vorticity of opposite circulation in the boundary layer adjacent to the solid wall due to the no-slip condition. The boundary layer rolls up to form a new dipole of unequal strength compared to its parent vortex. Because the parent vortex in each of the newly formed dipoles is unevenly matched with its child vortex the dipoles have nonzero angular momentum and recirculate to interact with the wall a second time. This cyclical process occurs twice more until viscous dissipation finally diffuses the vortices.

[1]Unfortunately, there was an issue with the Sharcnet filesystem during the time which all of these simulations were conducted which caused performance issues due to slowed I/O capabilities, so there is no reliable timing data available to compare the performance of the single grid and AMR runs

The same series of simulations was repeated using AMR. The AMR runs had a base grid resolution of 128×128, 192×192, and 256×256, which with 2 levels of AMR and a refinement ratio of 4 yields a resolution on the finest level equivalent to that of the single grid runs. Figures 3.15 and 3.16 show a comparison of the vorticity field for the single grid run of resolution 4096×4096 with the AMR run of base grid resolution 256×256 at a similar output time. There is a very good agreement between the two simulations. Additionally, the vorticity field of the AMR run is overlaid with the outline of the grids used. Notice that the grids evolve in time to follow the flow features.

To further demonstrate the equivalence of the results for the highest resolution single grid and the AMR runs, we plot the time-series of the total kinetic energy and total enstrophy for both cases in Figure 3.17. It can be immediately seen that the kinetic energy and enstrophy curves are identical for the single grid and AMR runs. The maximum values of the enstrophy and the times at which they occur are tabulated in Table 3.5. The values from [18] are included for comparison purposes. Recall that two numerical methods were used in [18]: a finite difference method and a spectral method. The times at which the maximum enstrophy values occur agree to two decimal places for all resolutions. For the finest resolution, both the single grid and AMR runs agree with the benchmark values for the maximum enstrophy to less than 1%.

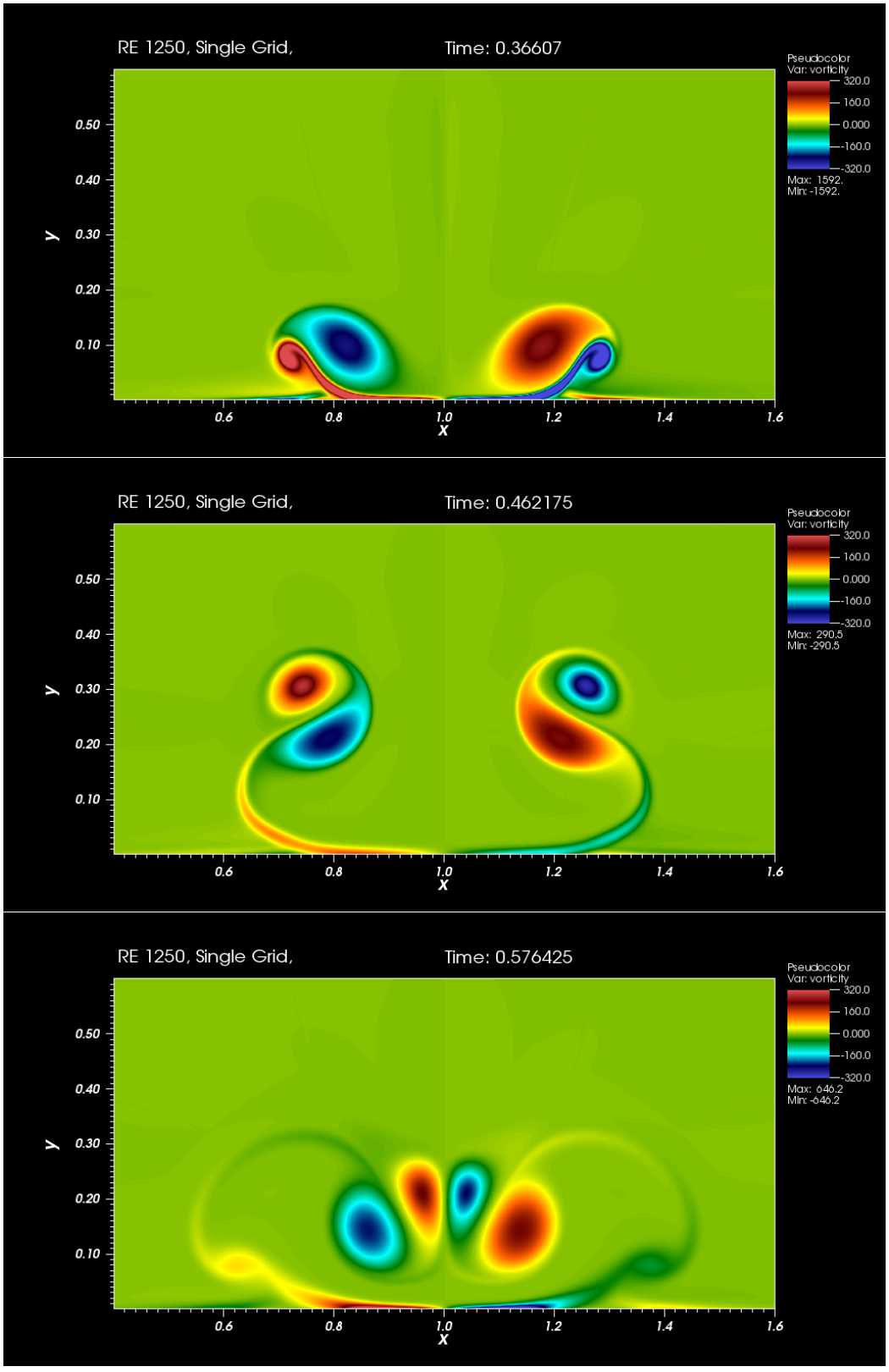

Figure 3.13: A time-series of the vorticity field after the initial dipole collides with the bottom boundary for $Re = 1250$. The initial dipole halves induce vorticity of opposite circulation in the boundary layer, which eventually rolls up and the parent vortices to form two new dipoles of unequal strength. These dipoles then recirculated and collide with the wall a second time. (Cont'd in Figure 3.14)

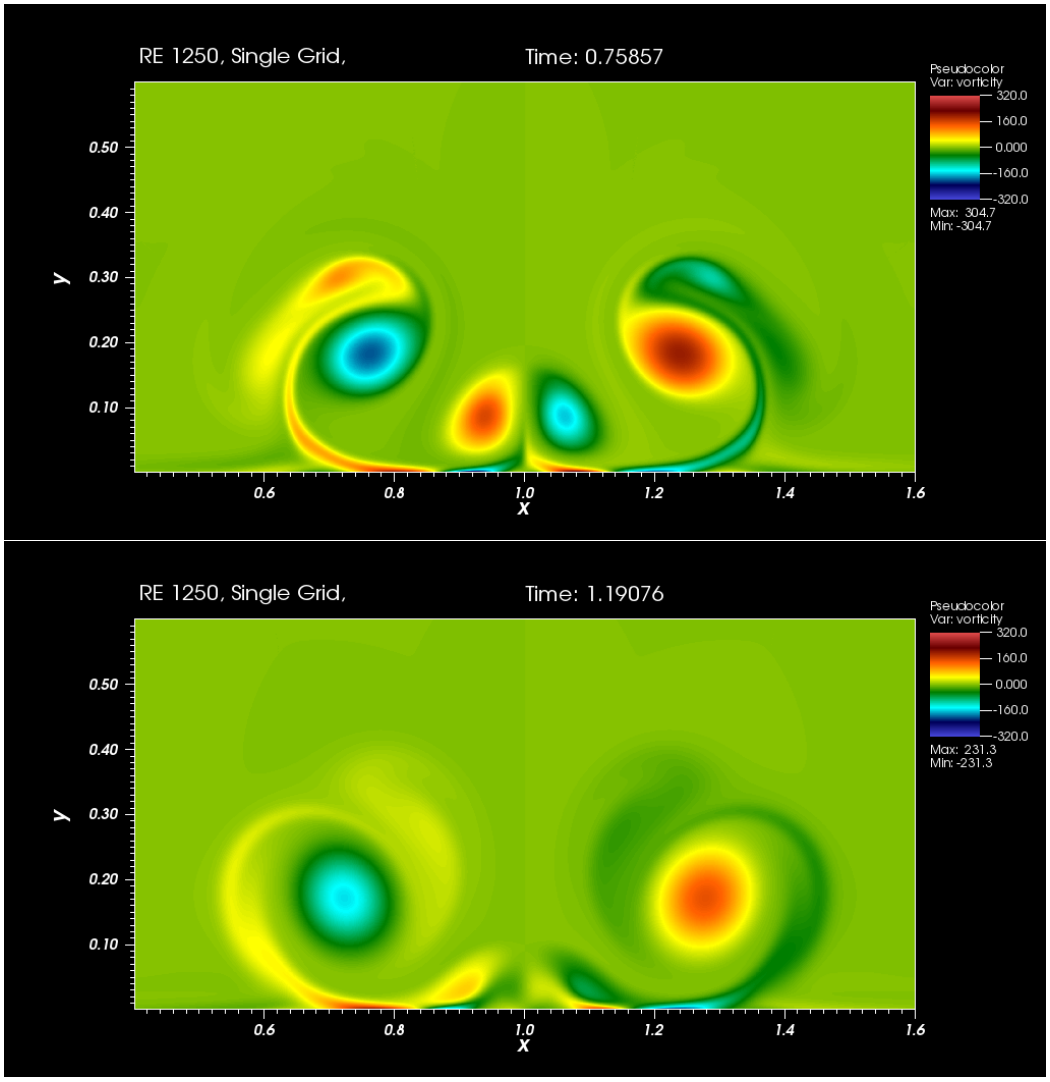

Figure 3.14: (Cont'd from Figure 3.13) After the new dipoles interact with the wall, the recirculation process continues until viscous dissipation eventually diffuses the vortices.

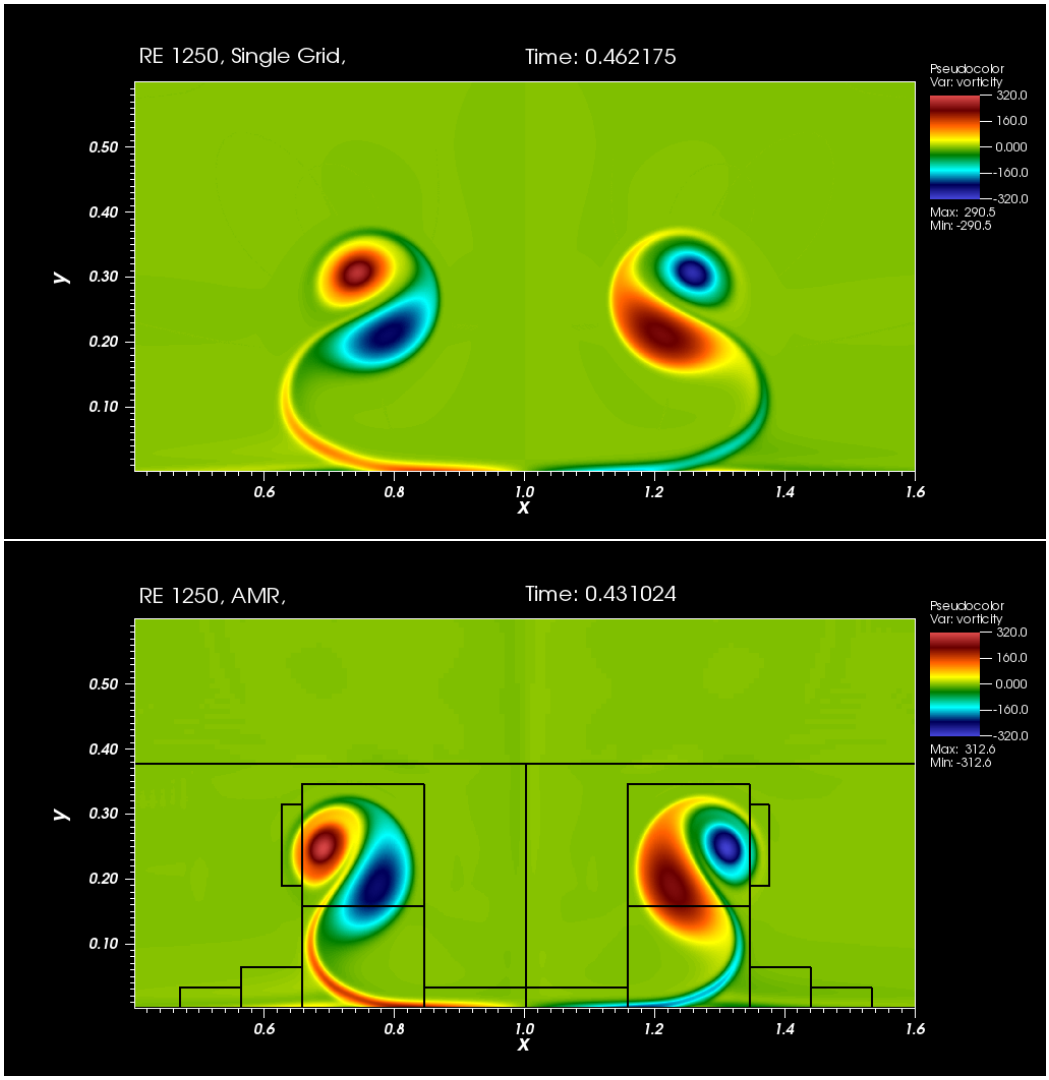

Figure 3.15: A comparison of the single grid and AMR runs with the highest resolution for $Re = 1250$ at similar times. There is excellent agreement between the two simulations. The vorticity field in the lower panel is overlaid with the outline of the grids used for the AMR simulation.

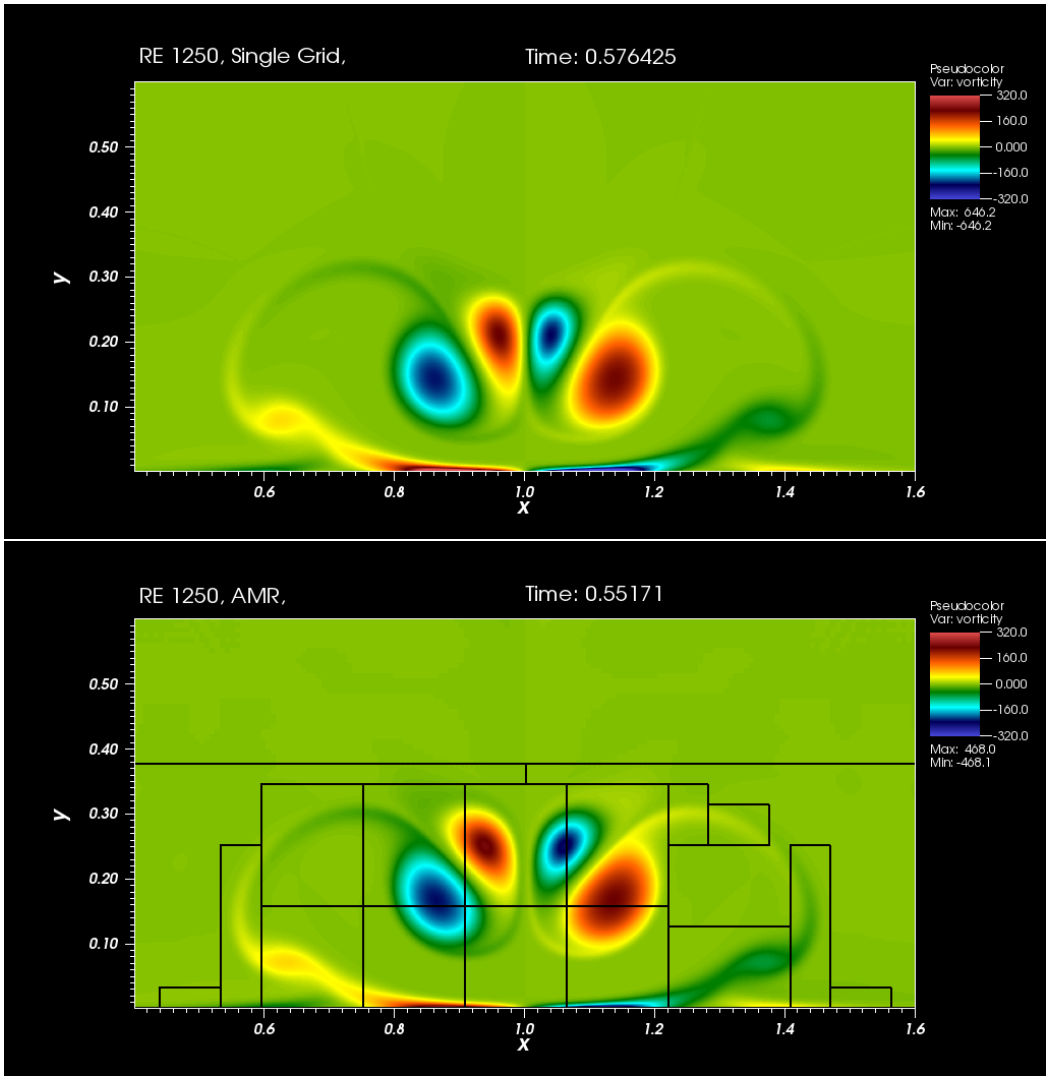

Figure 3.16: A comparison of the single grid and AMR runs with the highest resolution for $Re = 1250$ at similar times. There is excellent agreement between the two simulations. The vorticity field in the lower panel is overlaid with the outline of the grids used for the AMR simulation.

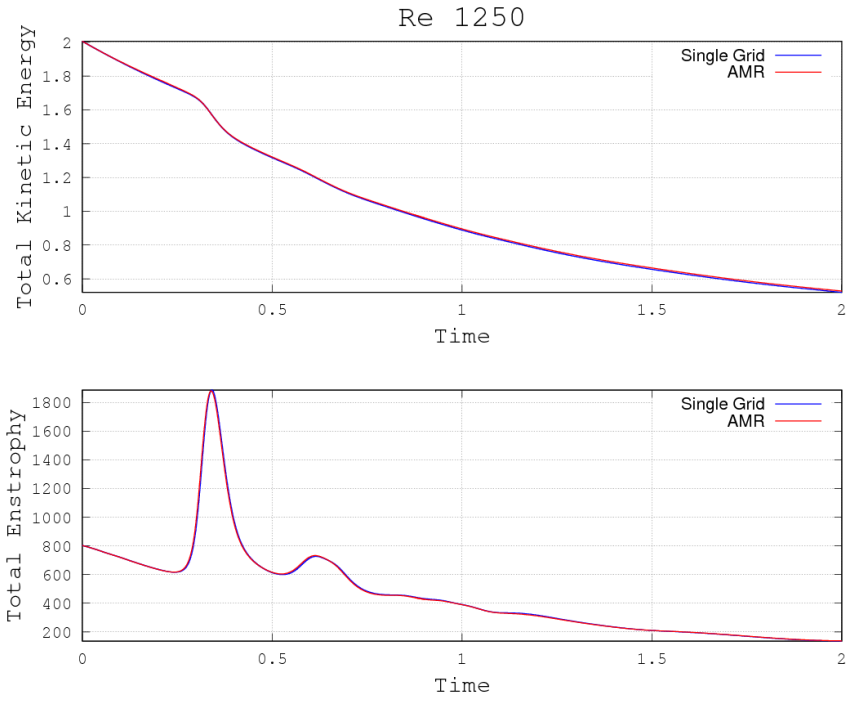

Figure 3.17: A comparison of the total kinetic energy and total enstrophy curves for the highest resolution single grid and AMR simulations for $Re = 1250$. There is a very good agreement between both sets of simulations for all times.

	Base Grid Resolution	t_1	Ω_1	t_2	Ω_2
Single Grid	2048×2048	0.34197	1829.2	0.61616	720.6
	3072×3072	0.34134	1871.0	0.61553	725.3
	4096×4096	0.34105	1886.7	0.61530	726.9
AMR	128×128	0.33968	1819.4	0.61393	726.8
	192×192	0.33946	1869.0	0.61372	728.6
	256×256	0.33925	1886.4	0.61292	732.4
Clercx. FD	1536×1536	0.341	1891	0.616	724.9
Clercx. SM	384×384	0.3414	1899	0.6162	725.3

Table 3.5: The first two maximum enstrophy values, Ω_1 and Ω_2, and the times at which they occur, t_1 and t_2, respectively, for $Re = 1250$. Results from [18] are included for comparison purposes.

Re 2500

For an integral-scale Reynolds number of 2500, the benchmark was run twice on a single grid using 3072×3072, and 4096×4096 grid cells, respectively. As computed in the previous section, the initial circulation Reynolds number is $Re_\Gamma \approx 105 Re_\mathcal{I} \approx 2.625 \times 10^5$. The same series of simulations was repeated using AMR. The AMR runs had a base grid resolution of 192×192, 256×256, and 384×384, which with 2 levels of AMR and a refinement ratio of 4 yields a resolution on the finest level equivalent to that of a single grid with 3072×3072, 4096×4096, and 6144×6144 grid cells respectively.

Figures 3.18 and 3.19 show a time series of the simulation carried out using a single grid and a resolution of 4096×4096. The dynamics at this Reynolds number are similar to those seen in the previous case with $Re = 1250$ with the exception that the vortices persist longer. This can be seen by comparing the bottom panel of Figure 3.14 with the bottom panel of Figure 3.19, which occur at roughly the same physical time.

Figures 3.20 and 3.21 show a comparison of the single grid run of resolution 4096×4096 with the AMR run with base grid resolution 384×384 at similar output times. There is again a good qualitative agreement between the two simulations.

In Figure 3.22 the total kinetic energy and total enstrophy curves of the highest resolution single grid and AMR runs are plotted against each other. Good agreement is shown between the two methods for all times. The first two maximum values of the enstrophy and the times at which they occur are compiled in Table 3.6 for all of the single grid and AMR runs. Additionally, the values from [18] are included for comparison purposes. The times at which the maximum values occur agree to two decimal places for all simulations.The maximum enstrophy values of the highest resolution single grid run were within 1% of the finite difference code, and 2% of the spectral code results from [18], while the highest resolution AMR run was within $< 1\%$ and 1%, respectively, of the same benchmark results.

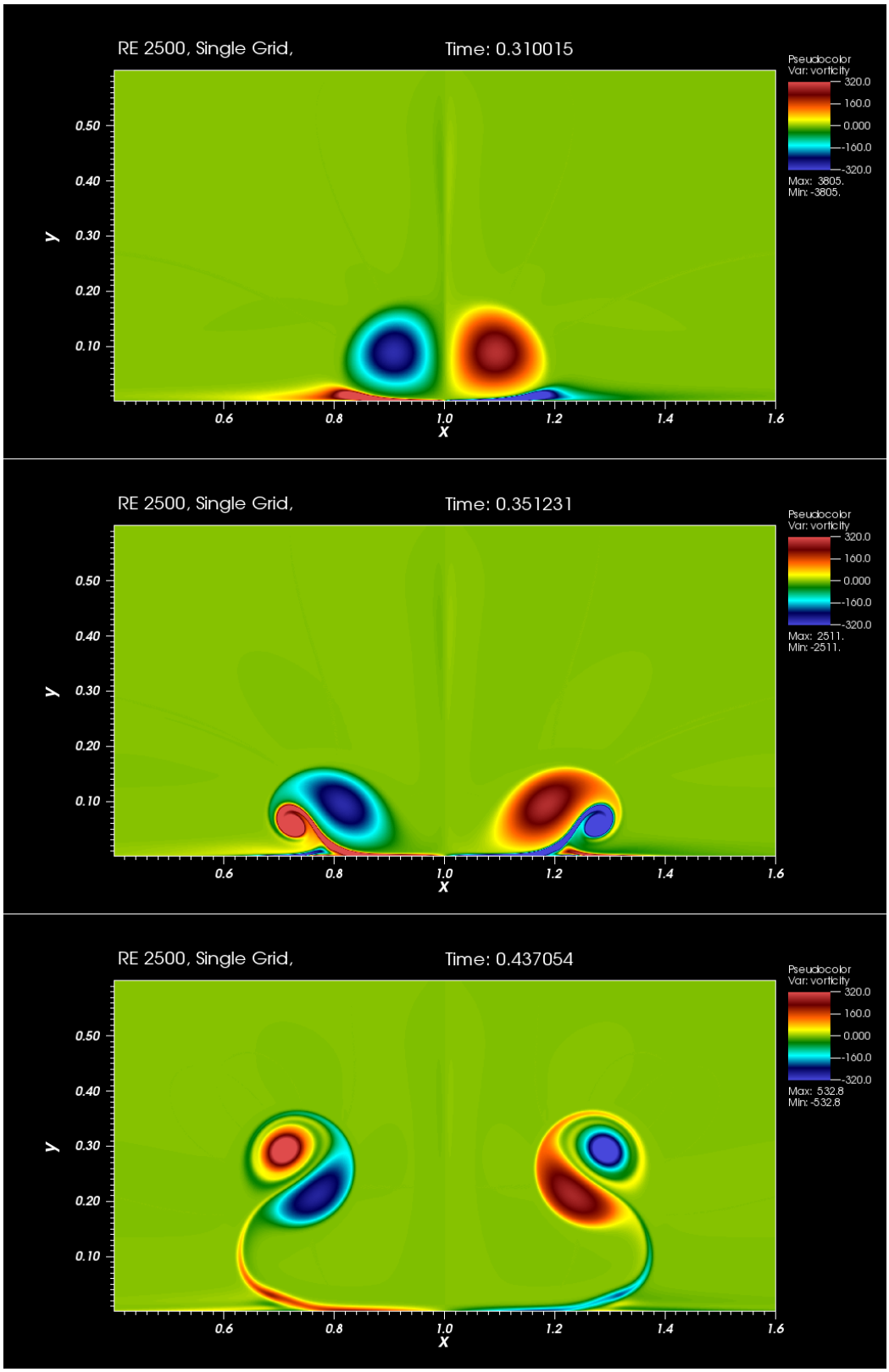

Figure 3.18: A time-series of the vorticity field after the initial dipole collides with the bottom boundary for $Re = 2500$. The dynamics for the first part of the evolution are similar to the $Re = 1250$ case. (Cont'd in Figure 3.19)

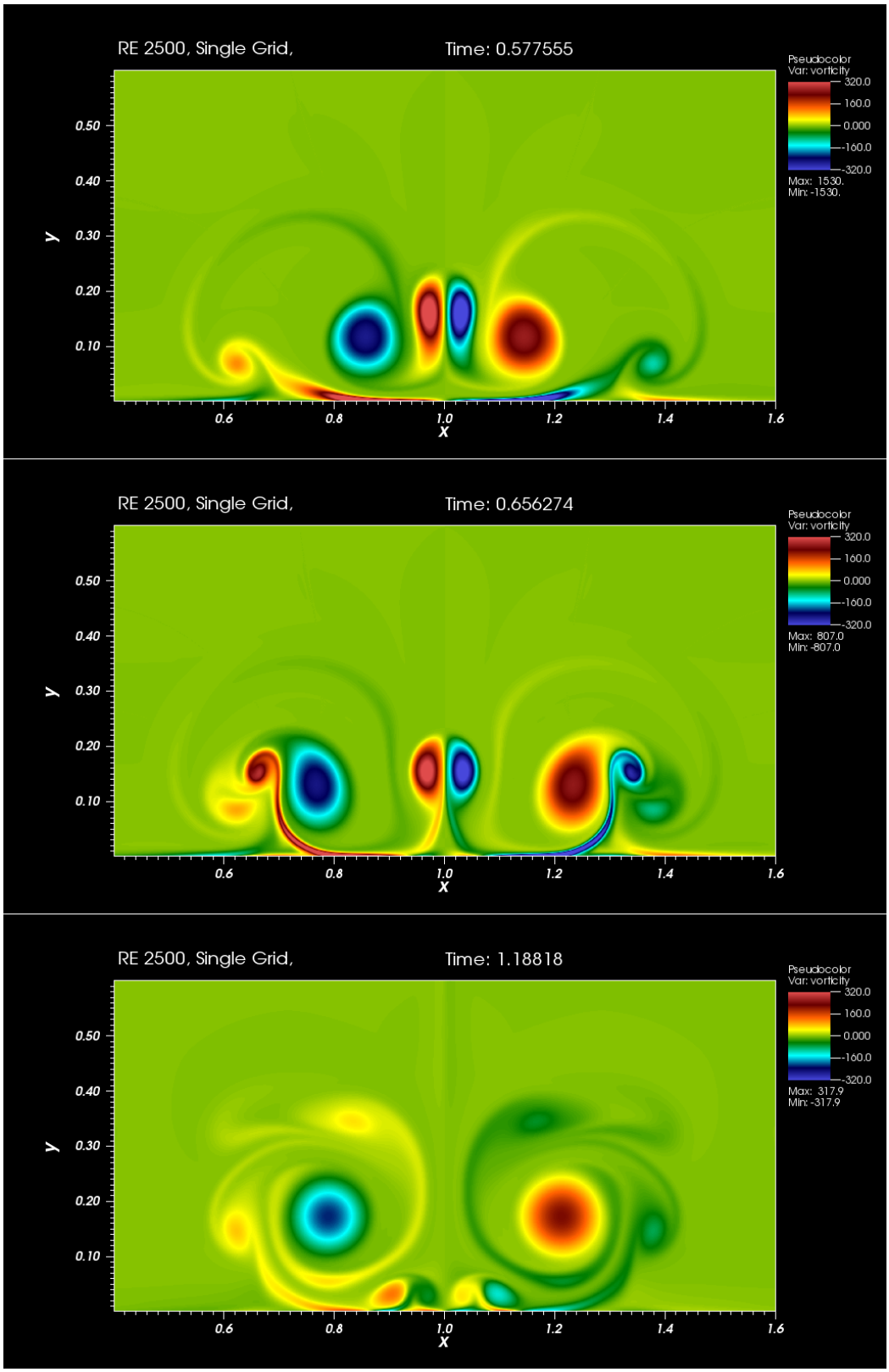

Figure 3.19: (Cont'd from Figure 3.19) The viscous dissipation is weaker than in the previous case so vorticity persists for longer times in this simulation.

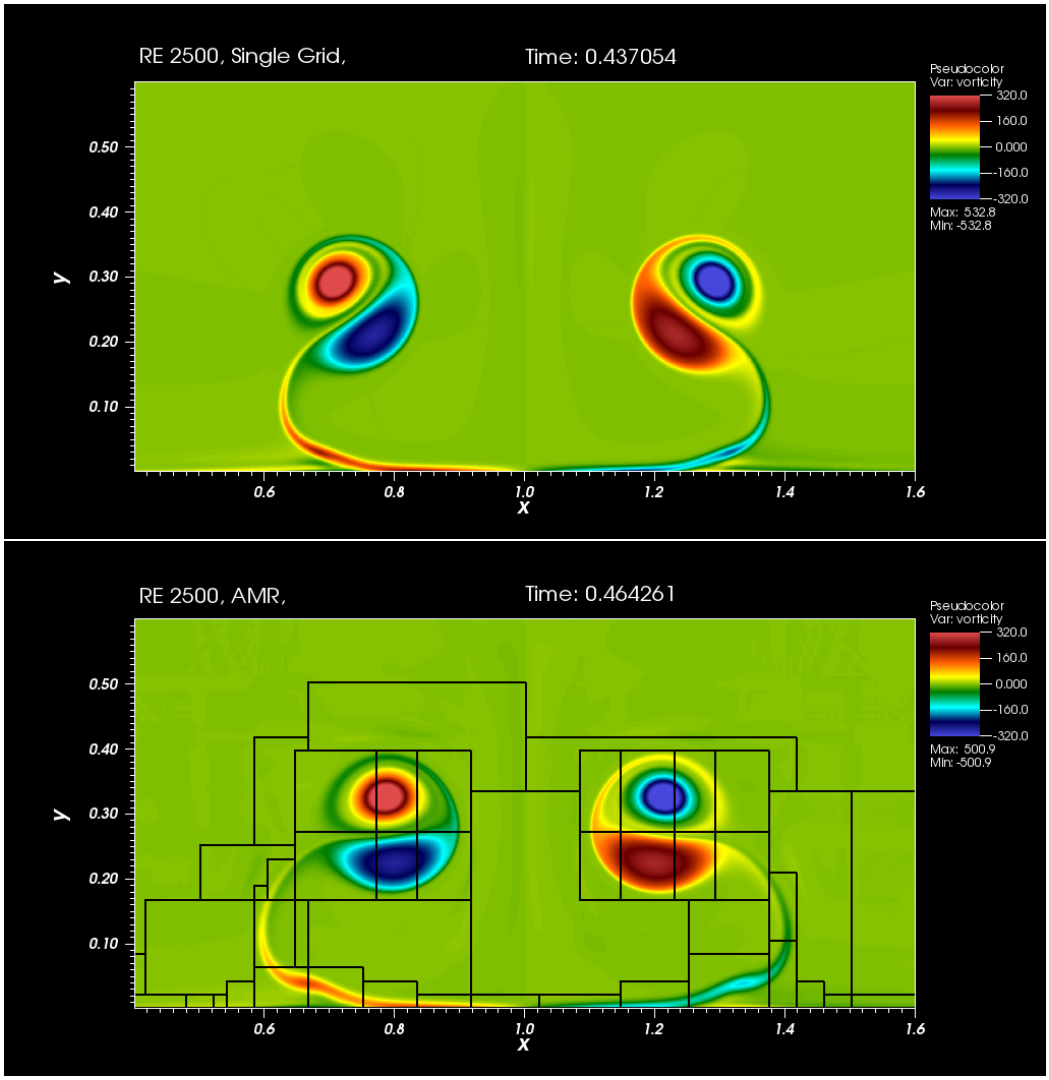

Figure 3.20: A comparison of the single grid and AMR runs with the highest resolution for $Re = 2500$ at similar times. There is excellent agreement between the two simulations. The vorticity field in the lower panel is overlaid with the outline of the grids used for the AMR simulation.

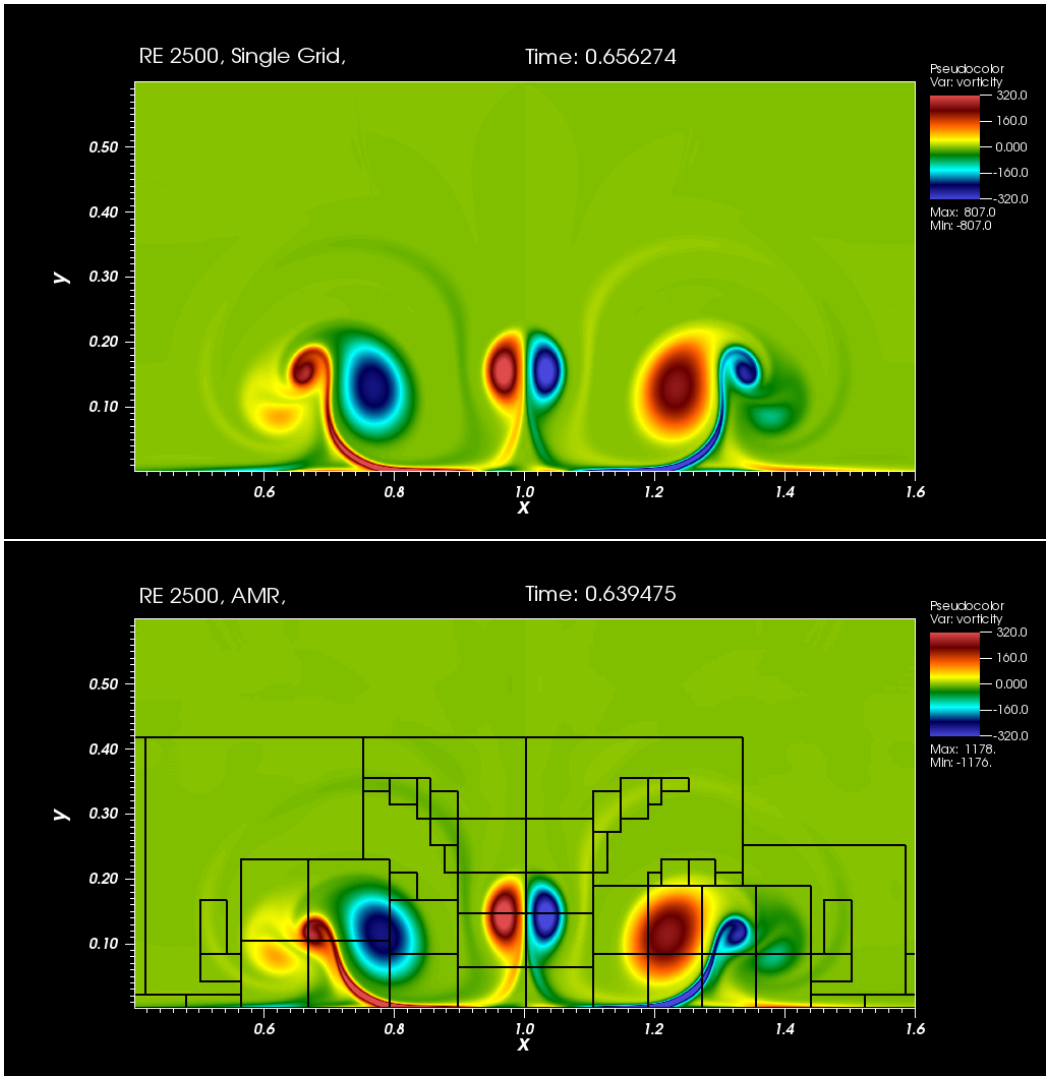

Figure 3.21: A comparison of the single grid and AMR runs with the highest resolution for $Re = 2500$ at similar times. There is excellent agreement between the two simulations. The vorticity field in the lower panel is overlaid with the outline of the grids used for the AMR simulation.

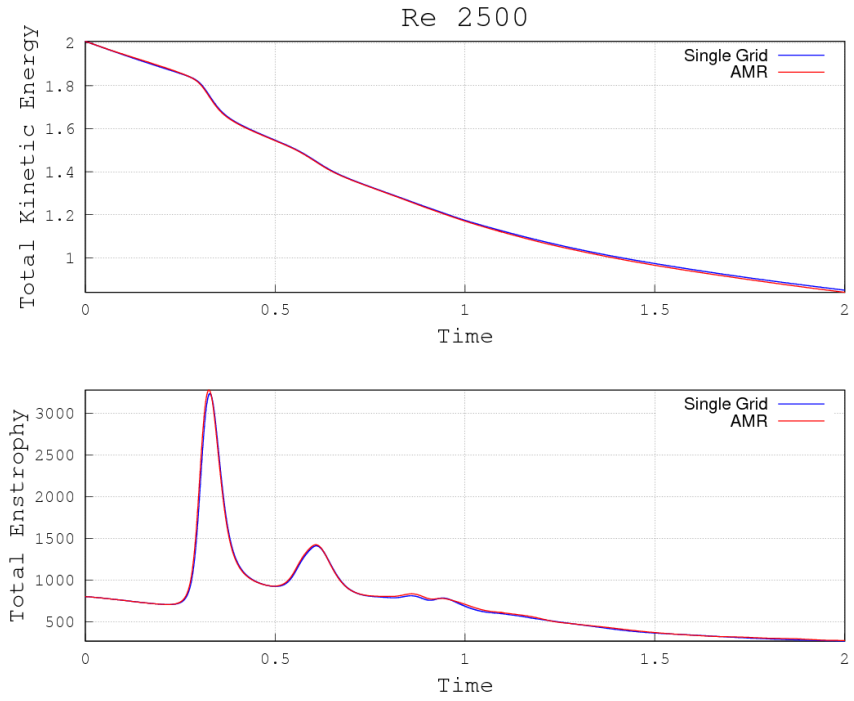

Figure 3.22: A comparison of the total kinetic energy and total enstrophy curves for the highest resolution single grid and AMR simulations for $Re = 2500$. There is a very good agreement between both sets of simulations for all times.

	Base Grid Resolution	t_1	Ω_1	t_2	Ω_2
Single Grid	3072×3072	0.32841	3170.2	0.60888	1399.7
	4096×4096	0.32793	3234.6	0.60847	1410.2
AMR	192×192	0.32655	3165.1	0.60672	1404.3
	256×256	0.32615	3217.4	0.60539	1417.8
	384×384	0.32554	3277.9	0.60732	1423.4
Clercx. FD	2048×2048	0.328	3270	0.608	1408
Clercx. SM	512×512	0.3279	3313	0.6089	1418

Table 3.6: The first two maximum enstrophy values, Ω_1 and Ω_2, and the times at which they occur, t_1 and t_2, respectively, for $Re = 2500$. Results from [18] are included for comparison purposes.

Re 5000

For an integral-scale Reynolds number of 5000, the benchmark was run on a single grid using 4096×4096 grid cells. As computed in the previous section, the initial circulation Reynolds number is $Re_\Gamma \approx 105 Re_\mathcal{I} \approx 5.25 \times 10^5$. The same series of simulations was repeated using AMR. The AMR runs had a base grid resolution of 256×256, 384×384, and 512×512. With 2 levels of AMR and a refinement ratio of 4 this yields a resolution on the finest level equivalent to that of a single grid run with 4096×4096, 6144×4096, and 8192×8192 grid cells, respectively.

Figures 3.23 and 3.24 show a time series of the simulation carried out using a single grid and a resolution of 4096×4096. The dynamical processes observed at this Reynolds number are quite different than the previous two. The bottom panel of Figure 3.23 reveals that smaller vortices continue to be generated in the boundary layer in the wake of the two newly formed dipoles. After the child dipoles recirculate and undergo a second collision with the boundary, several dipoles are generated travelling in different directions, as can be seen in the middle panel of Figure 3.24. The entire cyclical motion persists for a long time after the initial collision producing a complex pattern of vortices. A comparison with the AMR simulations is omitted here for the sake of brevity, however the results match the time series shown for the single grid simulation.

To compare the AMR and single grid simulations of finest resolution, the total kinetic energy and total enstrophy are shown in Figure 3.25. The kinetic energy curves agree for both simulations almost exactly. For the enstrophy curves, the AMR case slightly overshoots the maximum values predicted by the single grid case, however the AMR run has a higher resolution on its finest level. The first two maximum values of the enstrophy and the time at which they occur are compiled in Table 3.7, along with the values from [18] for comparison purposes. The times at which the maxima occur agree to two decimal places for all simulations. Additionally, the highest resolution AMR run agrees with the maximum values of the benchmark to within 2%.

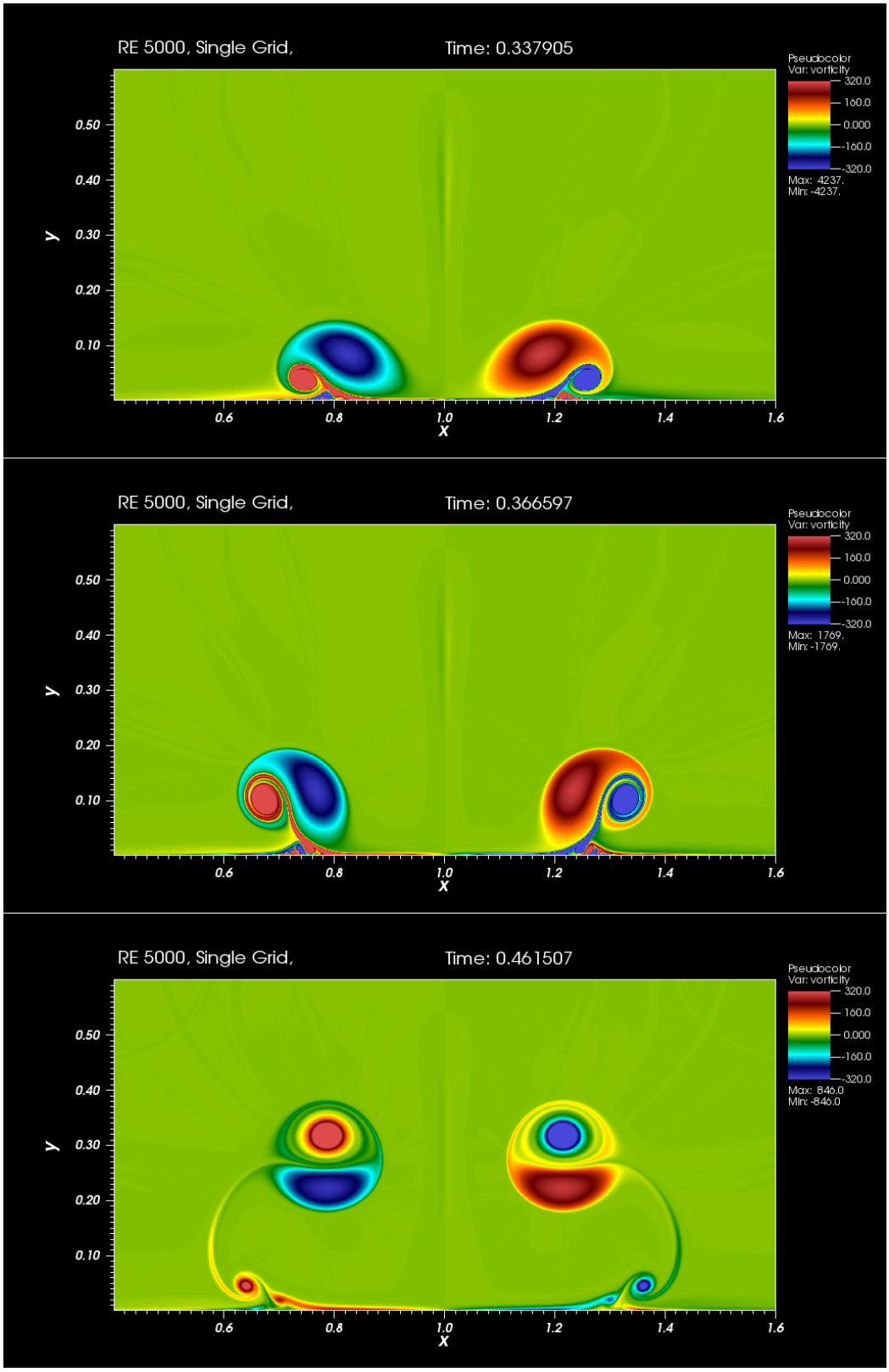

Figure 3.23: A time-series of the vorticity field after the initial dipole collides with the bottom boundary for $Re = 5000$. While the overall dynamics are similar here to the previous two cases, the bottom panel reveals a trail of smaller vortices being shed from the boundary layer in the wake of the two child dipoles. (Continued in Figure 3.24)

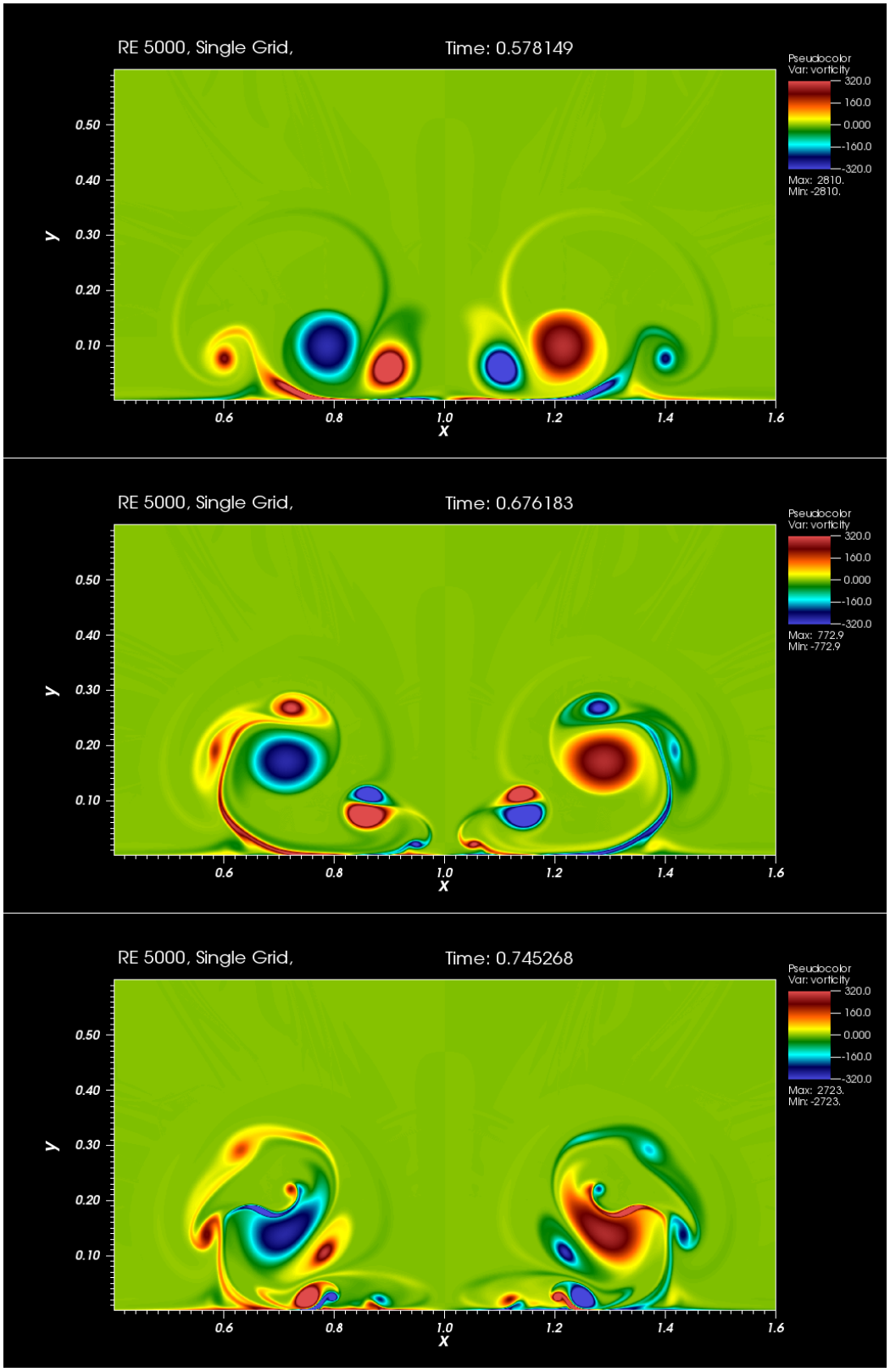

Figure 3.24: (Continued from Figure 3.23) After the first pair of child dipoles recirculate and collide with the wall a second time, several dipole pairs are generated, travelling in various directions. After the recirculation process continues again, the bottom panel reveals a a complex grouping of vortices which persist late into the simulation before finally being dissipated.

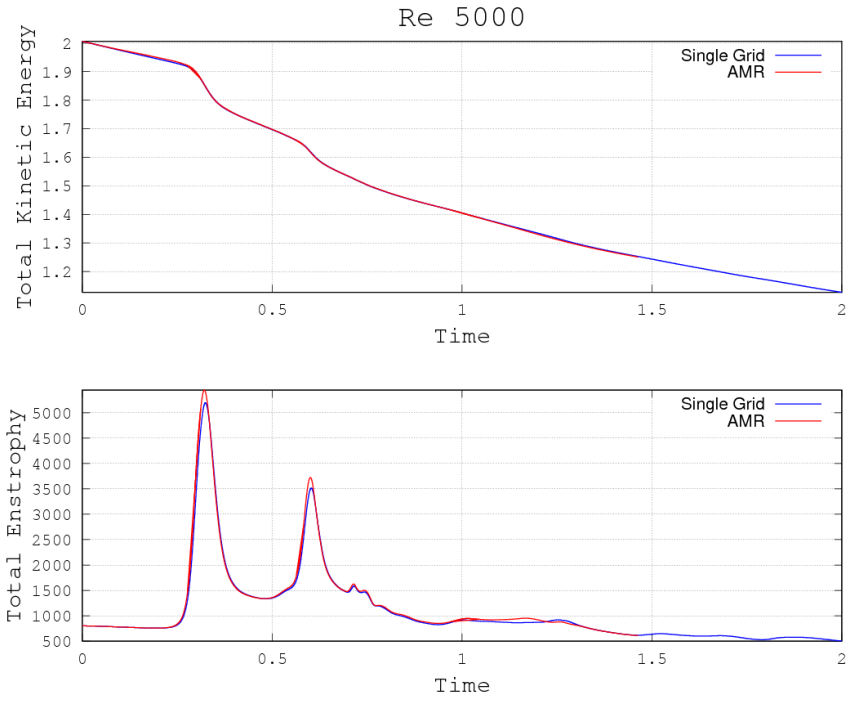

Figure 3.25: A comparison of the total kinetic energy and total enstrophy curves for the highest resolution single grid and AMR simulations for $Re = 5000$. There is good agreement between both simulations for all times. The AMR run slightly overshoots the maximum enstrophy values of the single grid run, however since the AMR run has finer resolution this could simply mean that the single grid simulation is under resolved.

	Base Grid Resolution	t_1	Ω_1	t_2	Ω_2
Single Grid	4096×4096	0.32387	5195.1	0.60279	3514.9
AMR	256×256	0.32240	5167.9	0.60067	3522.0
	384×384	0.32139	5367.9	0.60060	3671.5
	512×512	0.32114	5446.84	0.60063	3723.4
Clercx. FD	1536×1536	0.323	5435	0.605	3667
Clercx. SM	384×384	0.3234	5536	0.6035	3733

Table 3.7: The first two maximum enstrophy values, Ω_1 and Ω_2, and the times at which they occur, t_1 and t_2, respectively, for $Re = 5000$. Results from [18] are included for comparison purposes.

Re 10,000

For an integral-scale Reynolds number of $10,000$, the benchmark was run twice on a single grid using 4096×4096 and 6144×6144 grid cells respectively. As computed in the previous section, the initial circulation Reynolds number is $Re_\Gamma \approx 105 Re_\mathcal{I} \approx 1.05 \times 10^6$. The same series of simulations was repeated using AMR. The AMR runs had a base grid resolution of 384×384 and 512×512, which with 2 levels of AMR and a refinement ratio of 4 yields a resolution on the finest level equivalent to that of a single grid run with 6144×6144 and 8192×8192 grid cells respectively.

Figures 3.26 through 3.28 show a timer series of the simulation carried out using AMR with a base grid resolution of 512×512. The dynamics of this case are similar to the previous case, with the addition of a boundary layer instability as discussed in [28]. The top panel in Figure 3.29 shows a close up of the right dipole shortly after it collides with the bottom boundary. The boundary layer rolls up due to a shear-instability before it even detaches from the wall. The bottom panel of Figure 3.29 shows the distribution of vorticity in the boundary layer at the same time as the top panel, confirming that there are regions of alternating vorticity.

An important observation from the time series is that the symmetry of the simulation is broken. As observed in laboratory experiments, the symmetry of the system is unstable to small perturbations. This perturbation could come from any number of steps in the solution procedure, including the regridding or synchronization steps. Further study needs to be done, turning off various features of the model one at a time, to determine the exact cause.

The first two maximum values of the enstrophy and the times at which they occur are compiled in Table 3.8. The value and time for the first maximum as computed in [28] is included, however the authors did not give a time of occurrence and value for the second maximum. The time at which the first maximum occurs agrees to two decimal places for all simulations. There is a difference of 5% in the maximum enstrophy value of the AMR run with the highest resolution and that of the benchmark. The time of occurrence for the second enstrophy maximum agree to one decimal place amounts of the the simulations. The value of the second enstrophy maximum of the single grid run with the finest resolution and the AMR run with the finest resolution agrees to within 1%. It is likely that the resolution of both the AMR and single grid runs needs to be increased further to better agree with the benchmark results. The divergence between the enstrophy curves for the single grid and AMR runs after the second maximum occurs is likely due to the symmetry breaking of the AMR simulation.

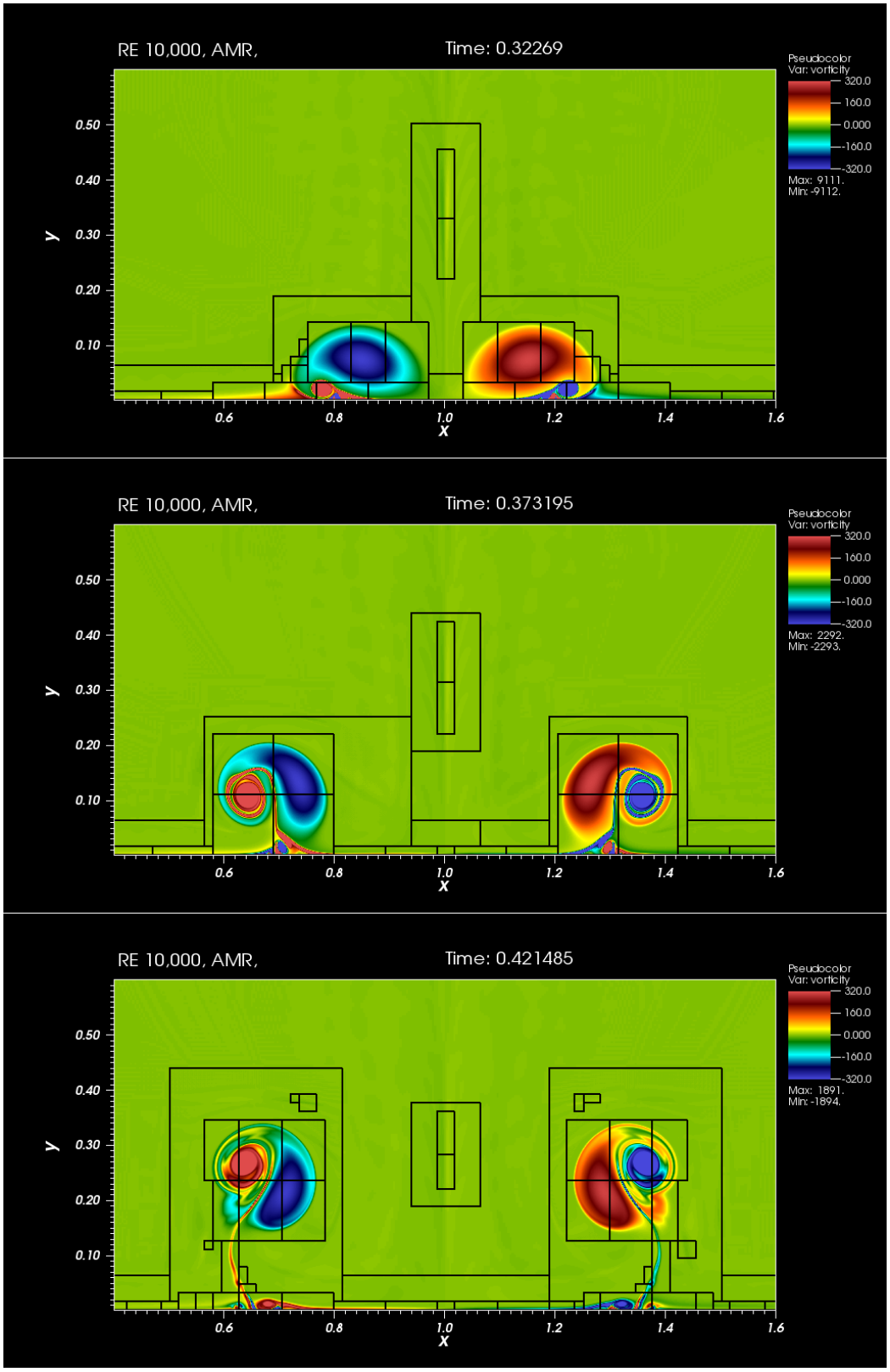

Figure 3.26: A time-series of the vorticity field after the initial dipole collides with the bottom boundary for $Re = 10,000$. The vorticity field from the highest resolution AMR run is shown, overlaid with the outline of the grids used. At this Reynolds number there is an instability in the boundary layer which causes vortices to roll up before the boundary layer detaches. (Cont'd in Figure 3.27)

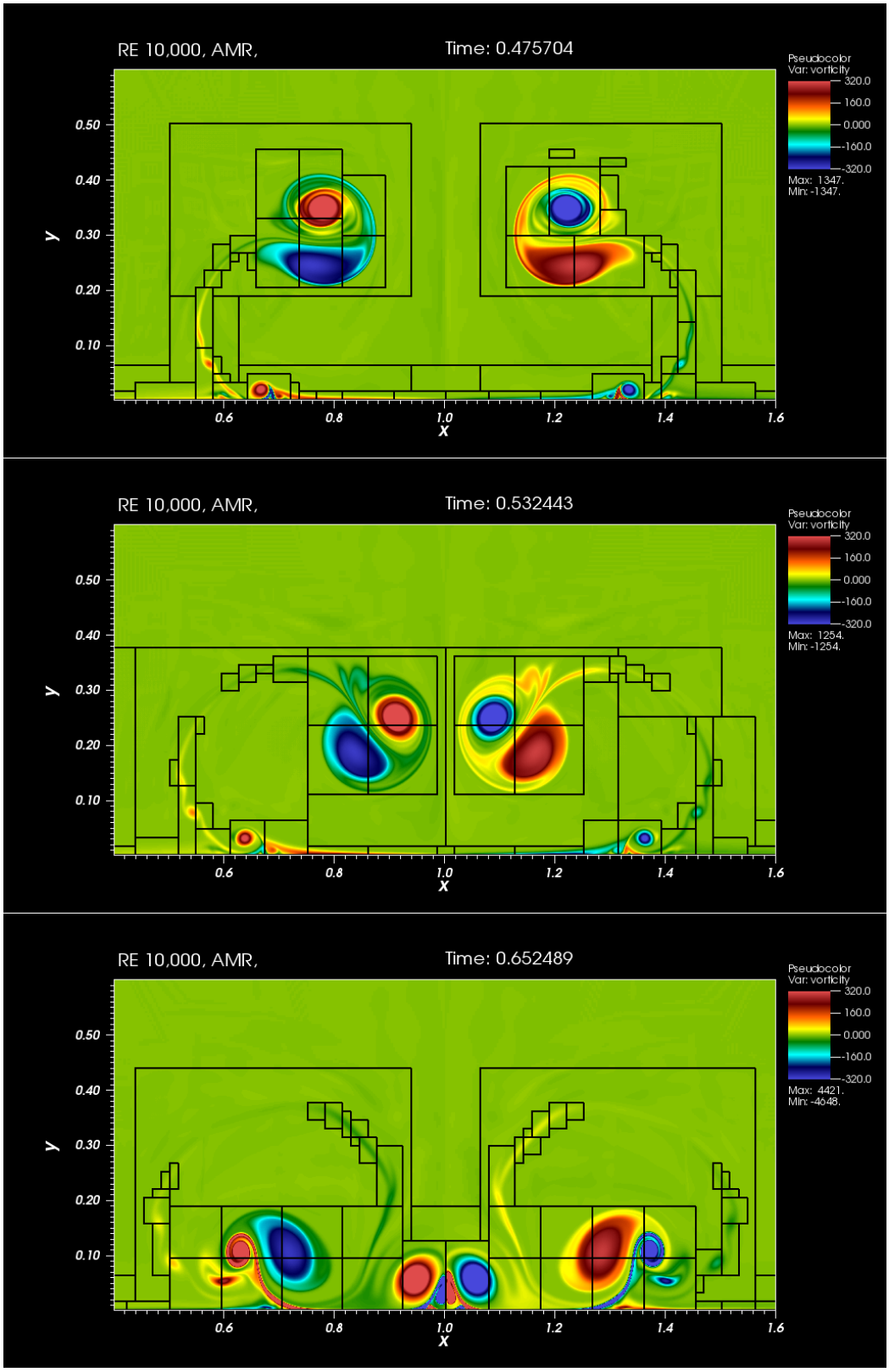

Figure 3.27: (Cont'd from Figure 3.27) As the child dipoles continue to recirculated and interact with the wall, the structure of the swirls in the vorticity field is much finer at this Reynolds number. Additionally, a number of strong compact vortices are ejected from the boundary layer in the wake of the child dipoles. (Cont'd in Figure 3.28)

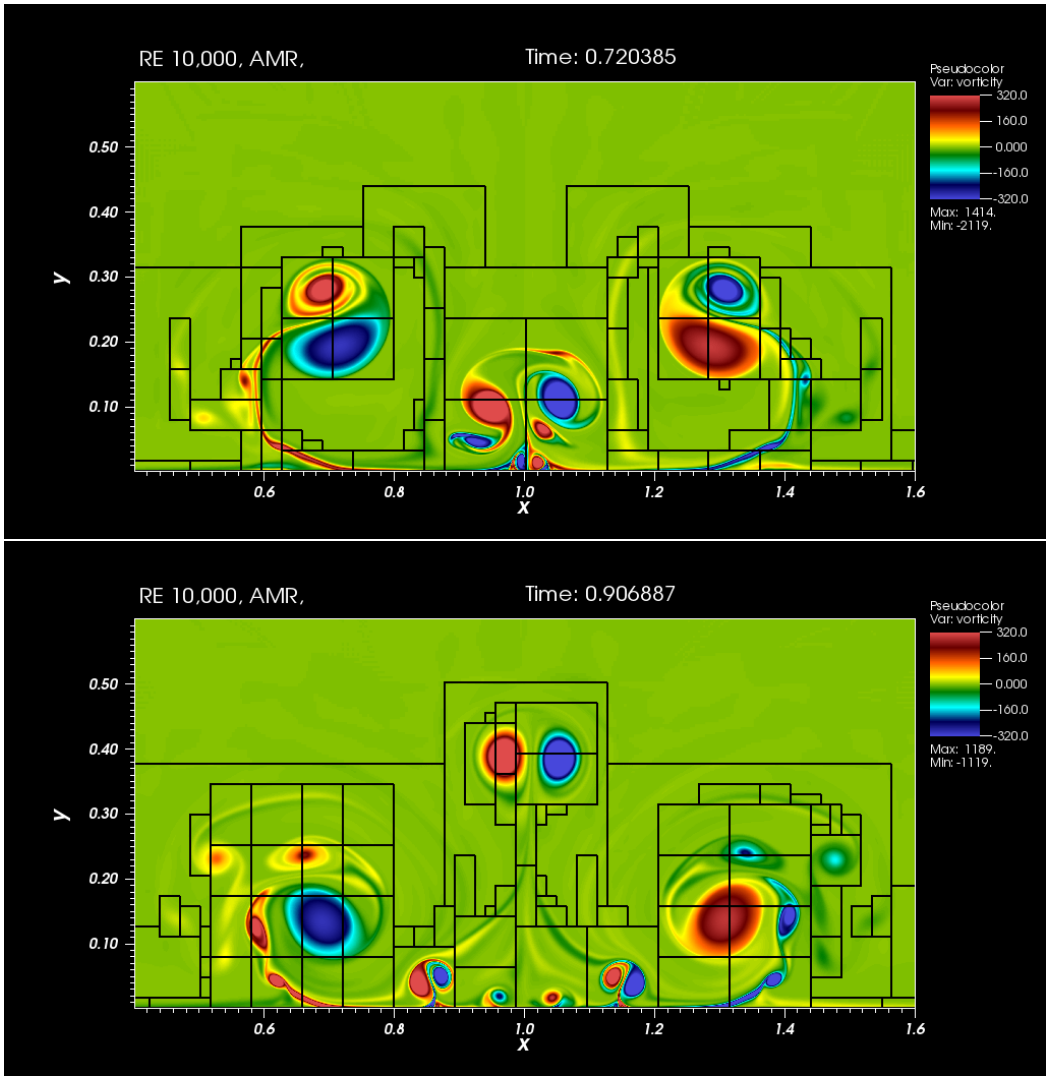

Figure 3.28: (Cont'd from Figure 3.27) Towards the end of the simulation the symmetry of the vorticity field is broke, as seen in the top pannel. Notice that the grids used in the bottom panel are not symmetric either. The cause of the symmetry breaking still needs to be determined.

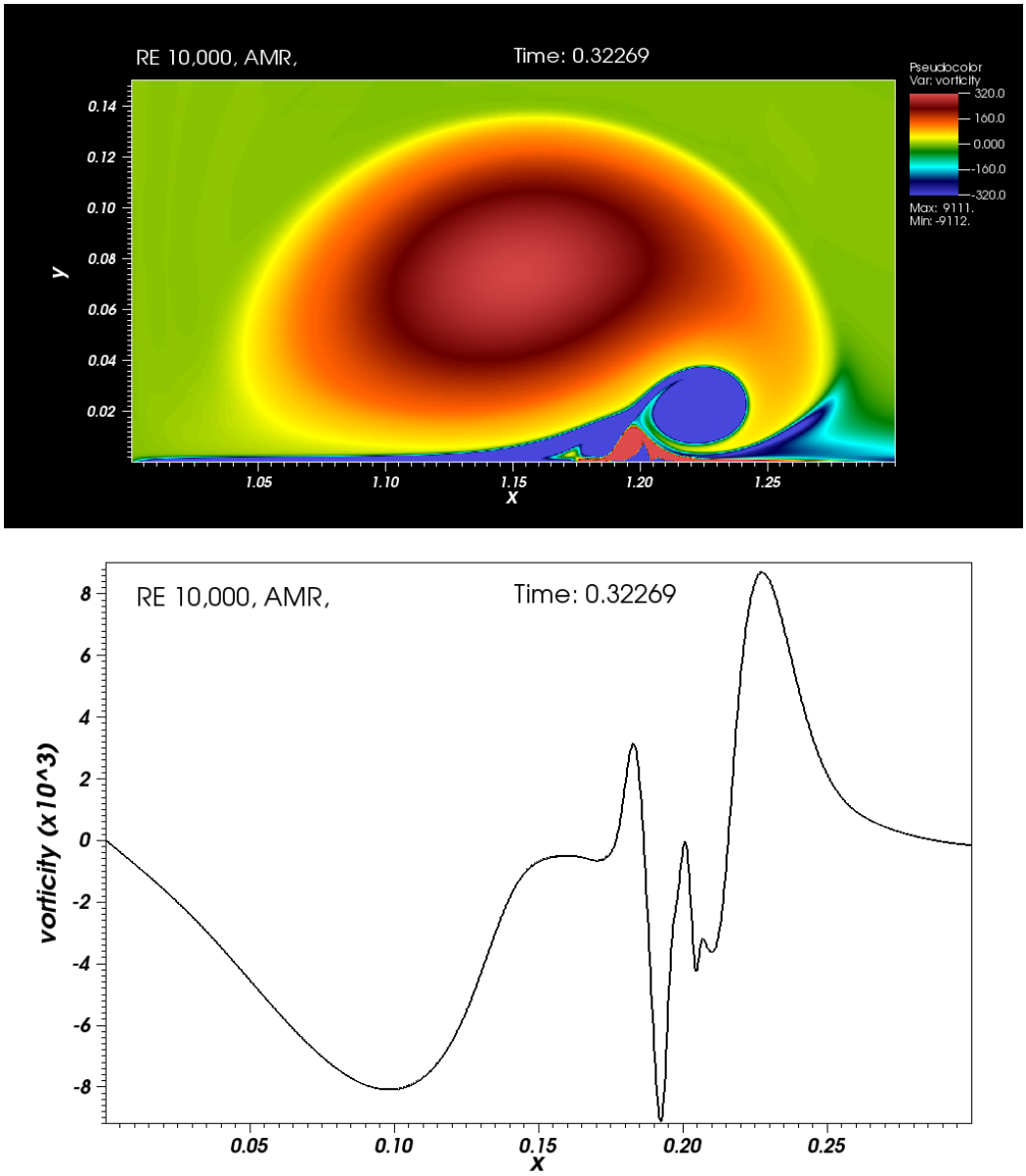

Figure 3.29: The top panel shows the right half of the dipole a very short time after the initial dipole collides with the wall. The shear instability in the boundary layer can be seen as small vortices roll up even before the boundary layer detaches from the wall. The lower panel shows the vorticity profile in the boundary layer at the same instant.

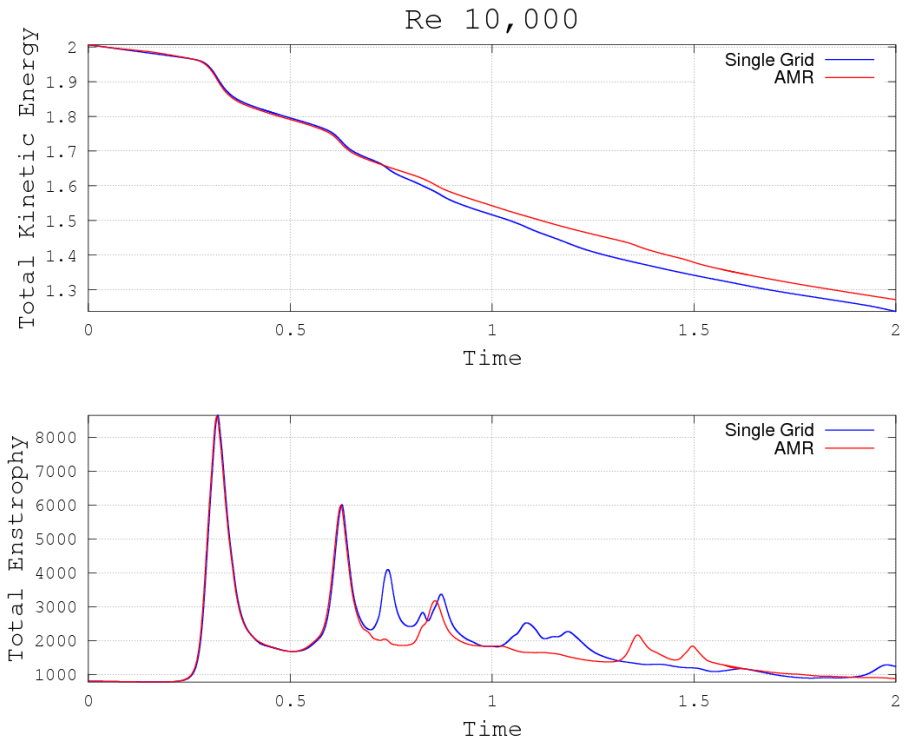

Figure 3.30: A comparison of the total kinetic energy and total enstrophy curves for the highest resolution single grid and AMR simulations for $Re = 10,000$. There is good agreement between both simulations for early times ($t \lesssim 0.6$), however after this they diverge. It is likely that this is caused by the symmetry breaking of the AMR simulation.

	Base Grid Resolution	t_1	Ω_1	t_2	Ω_2
Single Grid	4096×4096	0.32068	7941.5	0.62464	5348.9
	6144×6144	0.32029	8650.8	0.62782	6011.7
AMR	384×384	0.31828	8621.4	0.62528	4989.3
	512×512	0.31837	8942.2	0.63330	5989.8
Clercx. SM	1536×1024	0.3201	9426	-	-

Table 3.8: The first two maximum enstrophy values, Ω_1 and Ω_2, and the times at which they occur, t_1 and t_2, respectively, for $Re = 10,000$. Results from [28] are included for comparison purposes.

Re 100,000

While the highest Reynolds number considered for the vortex dipole wall interactions simulated in [28] is $Re = 20,000$, others have modelled dynamics in regimes higher than this using a variety of methods. Spalart, Strelet, and Shur [43] model the interaction of a Lamb-Oseen type dipole pair with a solid boundary, with and without a constant background flow, using a Reynolds-Averaged Navier-Stokes (RANS) type model. The translational Reynolds number of their simulations is $Re_{\text{tr}} = 5 \times 10^6$, however their results are likely severely under resolved or suffer from excessive numerical dissipation as the vorticity field is as smooth or smoother than the low Reynolds number cases we have shown above. Turk, Coors, and Jacob [50] also model the interaction of a Lamb-Oseen type dipole pair with a solid boundary using a vorticity equation based numerical model. They find that for a circulation Reynolds number of $Re = 3.33 \times 10^6$ that very strong compact vortices form and erupt from the boundary layer. Lastly, in order to isolate the stability of the boundary layer at Reynolds number up to 10^5, Gargano et al. [22] model flow in the boundary layer using a vorticity equation based model which is forced by a large rectilinear vortex. They find a complex pattern of vortices form in the boundary layer on a scale much smaller than that seen in the $Re = 10^4$ regime presented above.

To test the limits of the AMR code, the Clercx and Bruneau dipole problem was simulated, this time using an integral-scale Reynolds number of $Re_{\mathcal{I}} = 10^5$. The circulation based Reynolds number for this case is therefore $Re_\Gamma = 105 Re_{\mathcal{I}} = 1.05 \times 10^6$, and the translational Reynolds number is $Re_{\text{tr}} \approx \frac{3}{5} Re_{\mathcal{I}} = 6.0 \times 10^4$. The simulation was performed using two levels of AMR with a refinement ratio of 4. Regridding was performed every two time-steps on each level and a base grid resolution of 1024×1024 grid cells was used. This gives a resolution of $\Delta x = \frac{2}{16384}$ on the finest level. The simulation was performed on the system "Orca" using 128 cores. The simulation was run to a time of $t = 0.6$.

Since we are interested in the boundary layer physics, a time series of the right half of the dipole is shown around the moment at which the first collides with the wall is show in Figures 3.31 to 3.34. This time series compares the dynamics of the integral-scale Reynolds number 2500, 5000, 10,000, and 100,000 cases. As can be seen in the bottom right panel of Figure 3.31 the boundary layer dynamics at Reynolds number 100,000 is markedly different from the previous cases. The boundary layer has rolled up into a number of very strong, compact vortices with circulation opposite to that of the parent vortex. Figures 3.32 and 3.33 show that a cloud of child vortices recirculate with the initial parent vortex, however the structure is much less coherent that the other three cases. Additionally, the boundary layer continues to shed a number of strong, compact vortices. Lastly, it can be seen in Figure 3.34 that the parent vortex and cloud of child vortices reach the boundary for a

second time much faster than the do in the lower Reynolds number cases.

Figure 3.35 compares a close up of the boundary layer an infinitesimally small time after the dipole collides with the wall for the integral-scale Reynolds number of $10,000$, and $100,000$. The maximum vorticity in the $Re = 100,000$ simulation is roughly three times larger than the $Re = 10,0000$ case. Additionally a large number of small vortices, with both positive and negative circulation, can be seen rolling up the in the boundary layer and undergoing a series of interactions before they exit the boundary layer. In comparison, only one large vortex with negative circulation is generated in the left panel. Finally, Figure 3.37 shows the vorticity in the boundary layer at the same time and location as Figure 3.35 for the $Re = 100,000$ case.

3.3.3 Discussion

We have seen in the previous simulations that AMR gives results that are equivalent to a single grid simulation using the same underlying algorithm. The total kinetic energy and enstrophy time series showed excellent agreement between the AMR and single grid simulations. Additionally, the maximum entrophy values and the times at which the occurred agreed well with the results from Clercx and Bruneau [18]. For the AMR simulation with integral-scale Reynolds number $10,000$, the symmetry of the problem was broken. The cause of this symmetry breaking is as of yet unclear, and needs to be determined by systematic further study.

A higher integral-scale Reynolds number regime of $Re = 100,000$ was simulated for the vortex dipole wall interaction presented in [18]. To the best of our knowledge, this is the first time this regime has been simulated for this particular problem. In this case the initial dipole induces the formation of a large number of strong, compact vortices which erupt from the boundary layer. A close-up of the boundary layer just after the dipole collides with the wall initially reveals a complex series of interactions among small vortices which is not observed at lower Reynolds numbers.

While the dynamics of the last simulation are interesting, it is not clear whether they would still be physically realized in a fully three-dimensional flow. As noted in the literature, [30], a three-dimensional pair of counter-rotating vortex tubes can undergo both a short wave instability, known as the **elliptic instability**, and a long wave instability, known as the **Crow instability**, which will eventually leading to their breakdown. This phenomenon has been observed in wind tunnel and wave-tank experiments [13, 24]. The three-dimensional vortex pair problem, along with other possible extensions of the simulations in this section will be discussed in the following chapter.

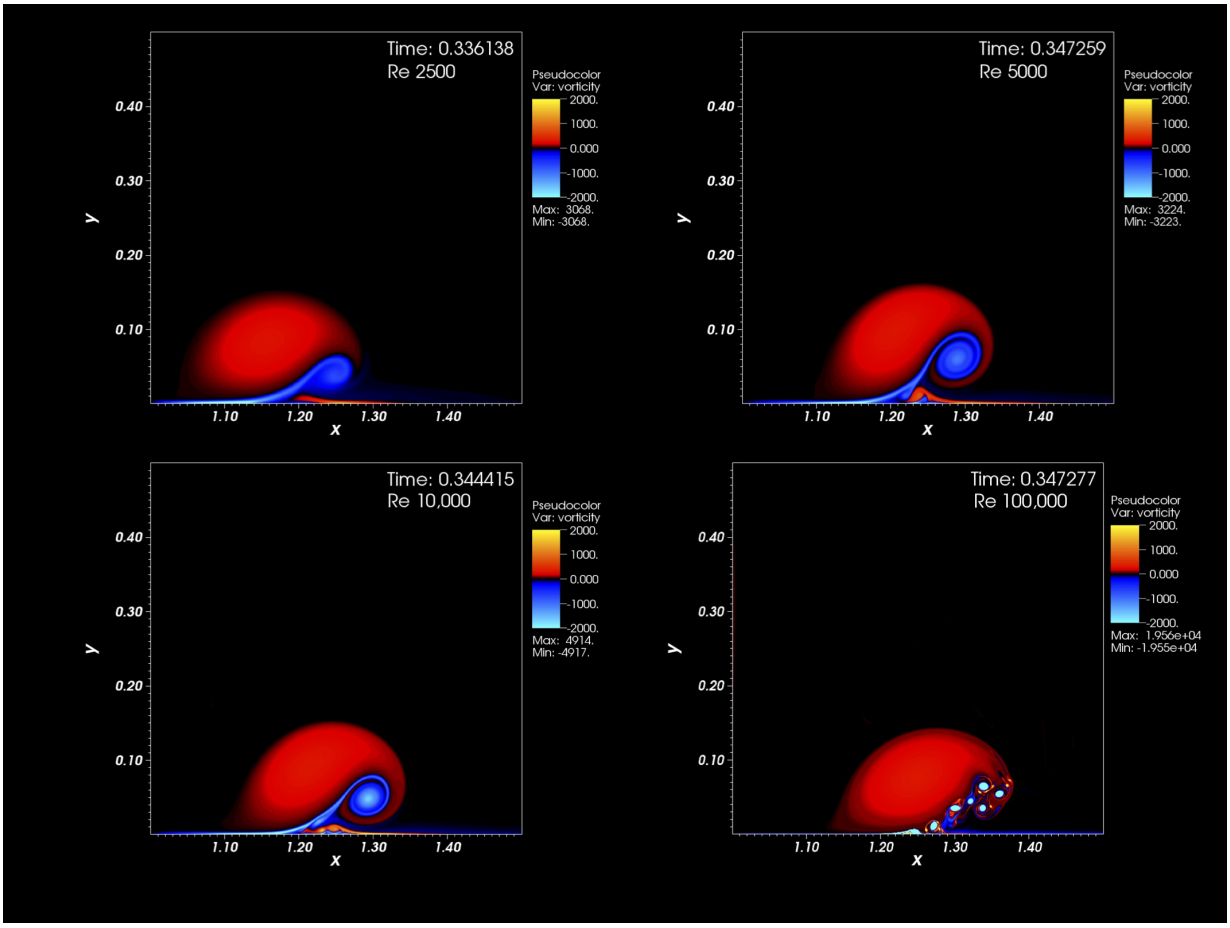

Figure 3.31: A time-series comparing the boundary layer dynamics shortly after the initial dipole collides with the wall for Reynolds numbers 2500, 5000, 10,000, and 10,000. The highest Reynolds number case is remarkably different from the other three as there is the appearance of a number of strong, compact vortices which form and are ejected from the boundary layer. (Cont'd in 3.31)

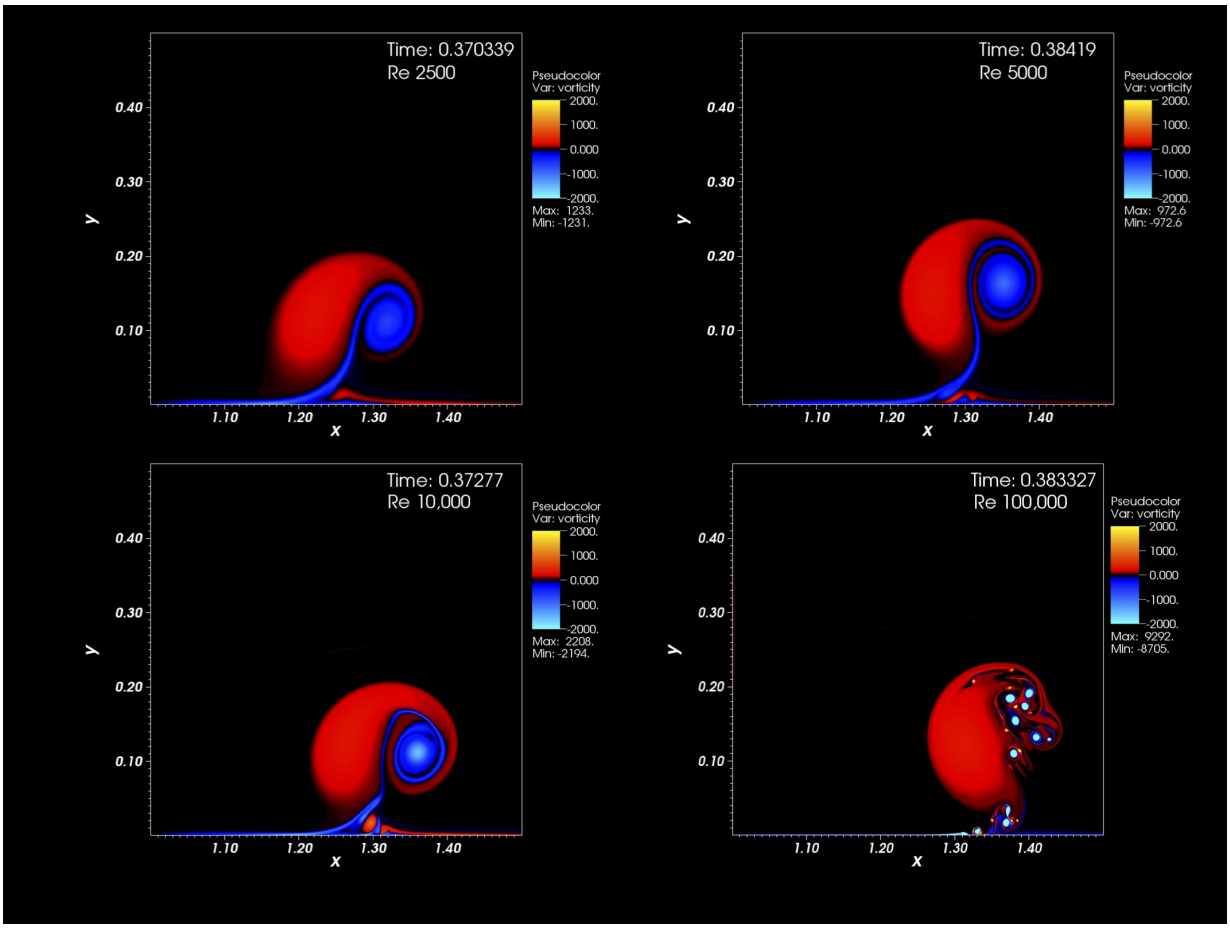

Figure 3.32: (Cont'd from 3.31) As the initial vortex recirculates, the smaller vortices are entrained and carried with it, as shown in the bottom right panel. This is in contrast to the other three cases were vorticity of opposite circulation rolls up from the boundary layer and pairs with the initial vortex to form a new dipole. (Cont'd in 3.33)

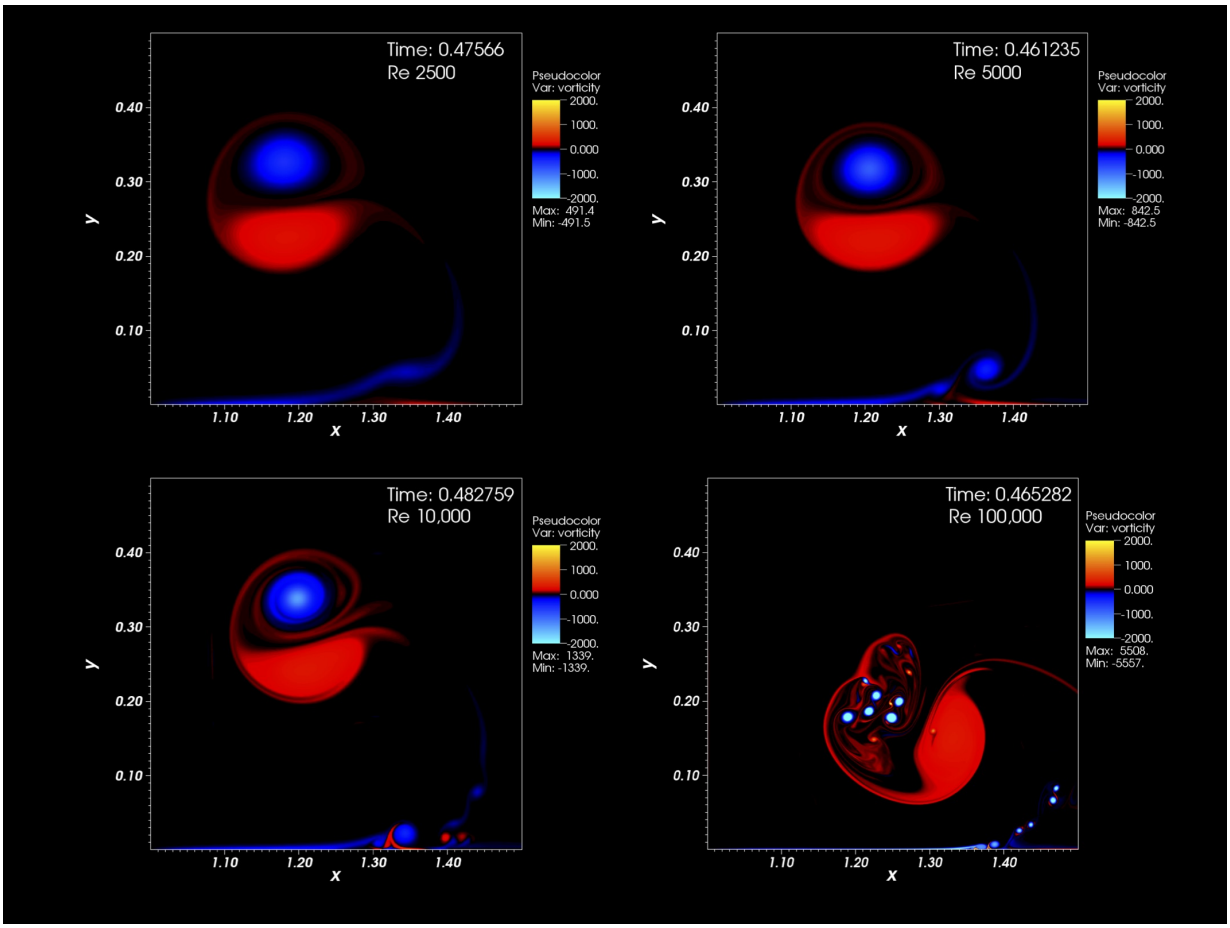

Figure 3.33: (Cont'd from 3.32) For the cases with Reynolds number 5000, 10,000 and 100,000 a trail of vortices continues to be shed in the wake of the newly formed dipole. The bottom right panel shows the primary vortex in the highest Reynolds number case does not regain as much height as in the other three cases. (Cont'd in 3.34)

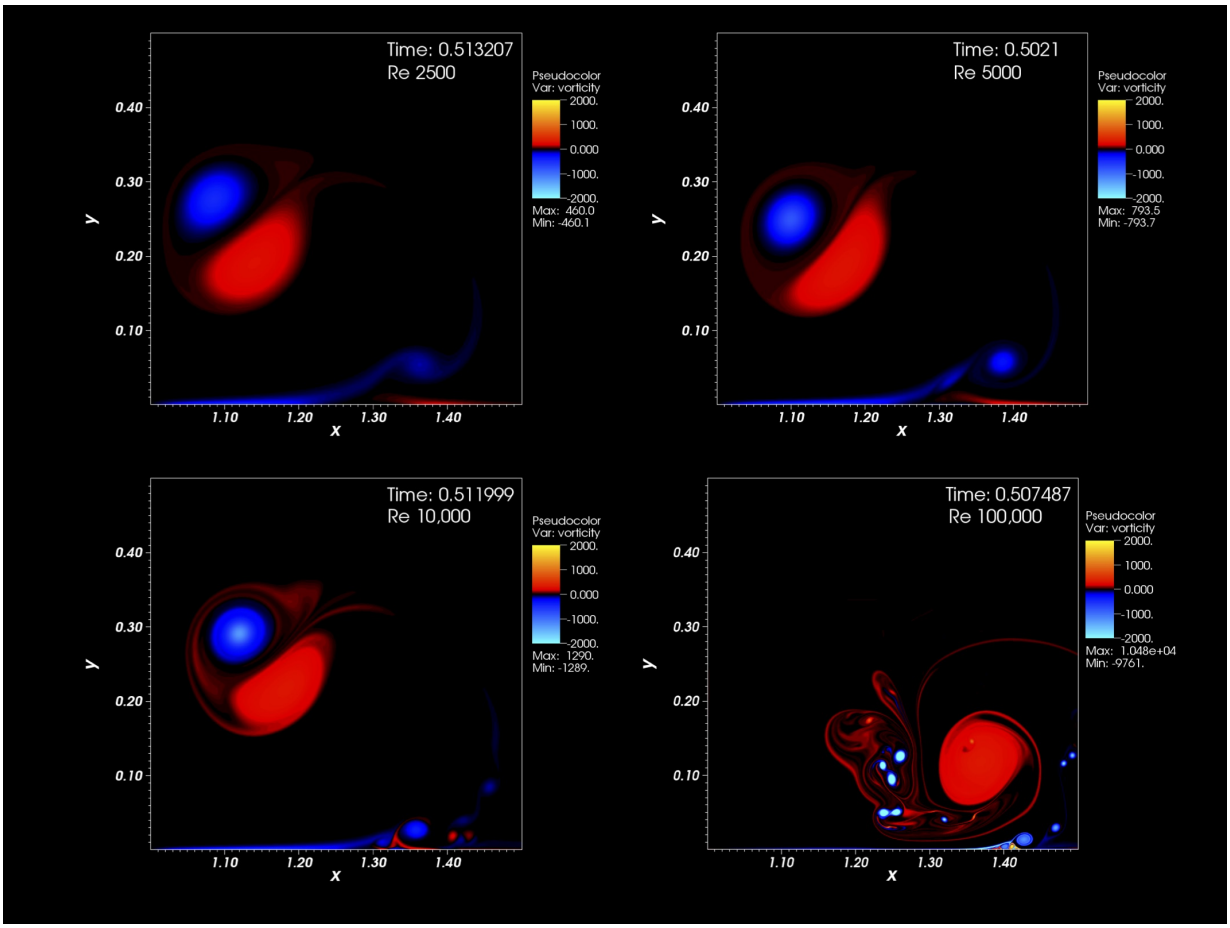

Figure 3.34: (Cont'd from 3.33) The bottom right panel shows the primary vortex quickly reaches the wall again, leaving behind it a thing wispy tail of positive vorticity in the highest Reynolds number case. In the other three cases the newly formed dipole is still recirculating and will not make contact with the wall again until further into the simulation.

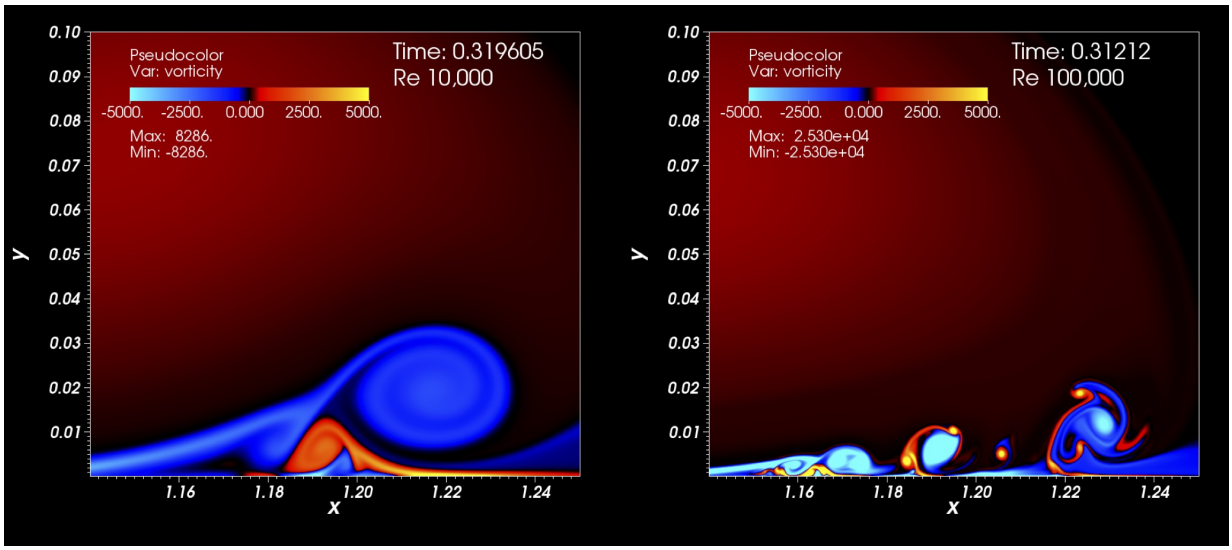

Figure 3.35: A close-up comparison of the boundary layer structures for the $Re = 10,000$ (left) and $Re = 100,000$ (right) cases. In the $Re = 10,000$ case only one large vortex of opposite circulation to the primary vortex has rolled up and an formed in the boundary layer. In the $Re = 100,000$ case a the boundary layer has formed a a number of strong, compact vortices which continue to rollup and interact as they exit the boundary layer.

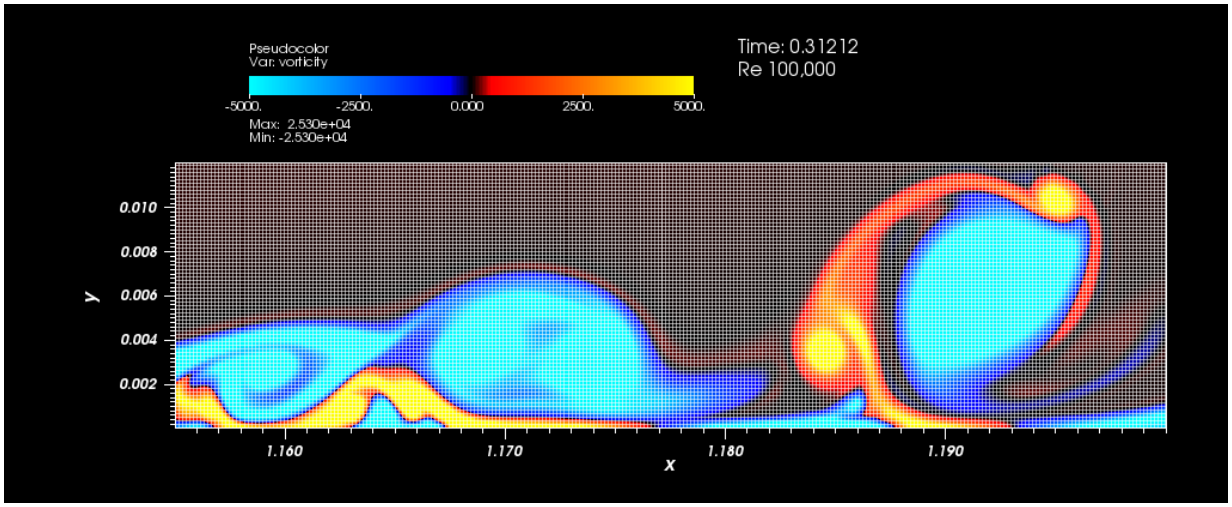

Figure 3.36: An close-up of the small scale vortices formed at Reynolds number 100,000 overlaid with the grid reveals that all of the flow features are adequately resolved. At the resolution used for this simulation there are approximately 8 grid points per 1.0×10^{-3} units.

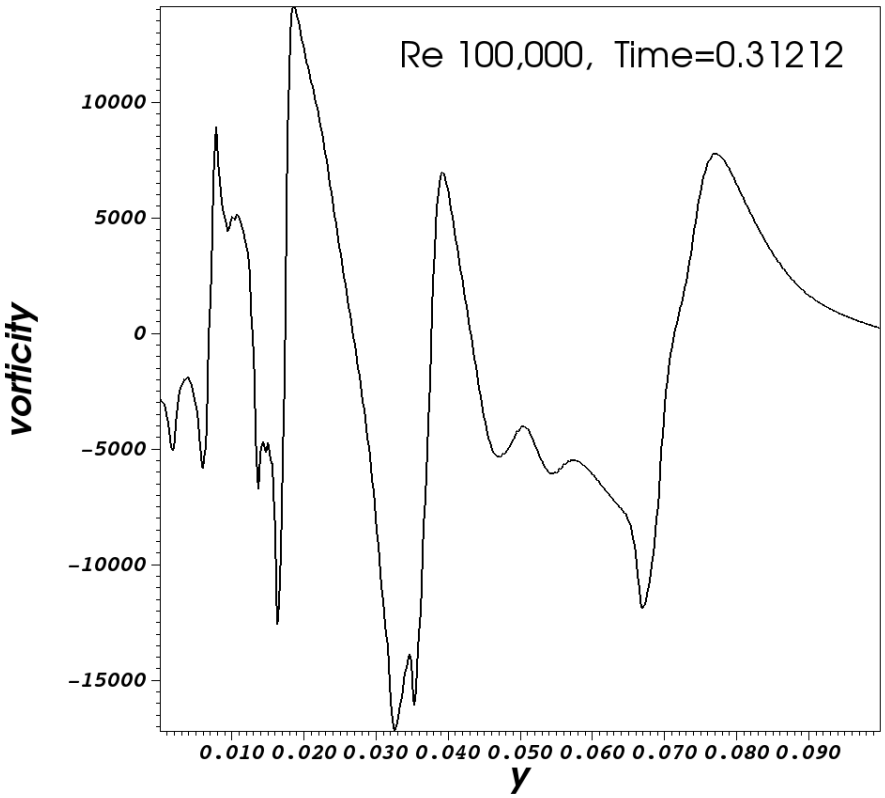

Figure 3.37: The vorticity profile the boundary layer at the same instant as shown in the right panel Figure 3.35 for $Re = 100,000$.

3.4 Vortex Ring Wall Interactions

There have been a number of laboratory and computational studies on the interaction of vortex rings with a solid wall, and a thorough review of the topic is given in [55]. We will briefly discuss some of the previous investigations here.

One of the most complete set of laboratory experiments on this problem, conducted over a wide range of Reynolds numbers, is presented in [15]. The results are encyclopedic in nature and have proven to be an invaluable tool for comparison in a number of computational studies. A vortex ring is formed by forcing fluid through a restricted opening using a pump. Dye is placed both in the orifice from which the vortex ring is generated and along the bottom wall to visualize the interaction.

In [15] it was observed that as the initial or primary vortex ring (PVR) interacts with the wall it induces vorticity of opposite circulation in the boundary layer. This vorticity rolls up and separates from the boundary later forming a secondary vortex ring (SVR) which migrates up and over top of the primary vortex ring into its centre. For translational Reynolds numbers $Re_{\text{tr}} > 350$ the formation of a weaker tertiary vortex ring (TVR) was also observed. In the range of Reynolds numbers $470 < Re_{\text{tr}} < 1600$, the SVR undergoes a wavy instability and forms what was termed a *loop structured* vortex ring in [15][2]. In this case the SVR is stretched under and over the PVR.

For translational Reynolds numbers in the range $Re_{\text{tr}} > 2500$, the SVR and TVR are much stronger. The SVR again develops a wavy instability as it migrates to the centre of the PVR, however it is much less apparent and the SVR is not pulled under and over the PVR. This was dubbed a *kink strutured* vortex ring in [15]. The tertiary vortex ring, which also has stronger circulation in this regime, also migrates up and over the PVR. Once the TVR has migrated over the PVR, the velocity it induces on the SVR cause sthe SVR to be ejected away from the boundary and into the bulk fluid, dragging the TVR with it. In [15] this process is described as forming a mushroom cloud like structure. The PVR subsequently breaks down into a turbulent cloud which quickly dissipates. It is noted in [15] that the structure of the SVR transitions from a loop structured to a kink structured vortex in the range $1600 Re_{\text{tr}} < 2500$.

A number of different computer simulations studies of vortex ring wall interactions have been performed previously. In [36] a series of axis-symmetric simulations are performed for a translational Reynolds numbers $564 \leq Re_{\text{tr}} \leq 3000$ using a finite difference code. Additionally a three dimensional simulation is performed for $Re_{\text{tr}} = 1250$. Their results

[2] An excellent sketch of this situation is provided in [15]

are compared to the laboratory experiments in [56]. In [47] spectral method is used to simulate both perturbed and unperturbed vortex ring wall interactions at a translational Reynolds number of $Re_{\text{tr}} = 645$. In [16] a lattice-Boltzman type model is used to simulate both normal and oblique collisions of a vortex ring with a wall for translation Reynolds numbers up to $Re_{\text{tr}} = 1000$. Their oblique collisions are conducted over a wide range of angles of incidence. Both the vorticity field and vortex core trajectories are compared to laboratory experiments. In [53] the vortex ring wall interaction problem was used as a test case to validate an AMR code. Simulations were performed for a translational Reynolds number of $Re_{\text{tr}} = 570$ and 670. The code is able to correctly model the development of the secondary and tertiary vortex rings, as well as the trajectories of the cores of all three rings.

With this backdrop, we will look to push the envelope using AMR and try to perform full three dimensional simulations of the interaction of a vortex ring with a solid wall for higher Reynolds numbers that have been considered previously.

3.4.1 Setup

To simulate the collision of a vortex ring with a solid wall, a Gaussian vortex ring with with major radius r_0 and the initial velocity profile

$$\mathbf{u}_\phi = \frac{\Gamma_0}{2\pi\sigma}\left(1 - e^{-\sigma^2/\sigma_0^2}\right)\hat{\boldsymbol{\phi}} \tag{3.4}$$

was used, where $\hat{\boldsymbol{\phi}}$ is the unit vector which is tangential to circles of circulation about the ring's core and σ is the radial distance from the ring's core. The parameters σ_0 and Γ_0 are the minor radius and initial circulation, respectively. A sketch of this profile is shown in Figure 3.38. The major and minor axes, and initial circulation were chosen to be

$$r_0 = \frac{5}{12}, \quad \sigma_0 = 0.21 r_0, \quad \Gamma_0 = 1.8534. \tag{3.5}$$

The initial translational velocity of the vortex ring is [39]:

$$U_s = \frac{\Gamma_0}{4\pi r_0}\left(\ln\frac{8r_0}{\sigma_0} - \frac{1}{4}\right) \approx 1.2. \tag{3.6}$$

This gives a translational Reynolds number of

$$Re_{\text{tr}} = \frac{2U_s r_0}{\nu} \approx \frac{1}{\nu}. \tag{3.7}$$

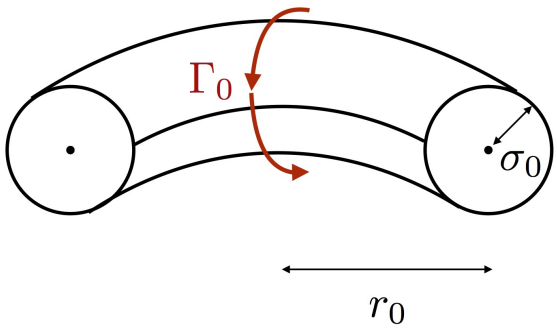

Figure 3.38: A sketch of the vortex ring defined by equaton (3.4).

Comparatively, the vortex Reynolds number is

$$Re_\Gamma = \frac{\Gamma_0}{\nu} \approx 1.8534 Re_{\text{tr}}. \tag{3.8}$$

For the first experiment with $Re_{\text{tr}} = 1000$, the computational domain was taken to be a rectangular box with dimensions $4.0 \times 4.0 \times 2.0$. This large domain ensured that the domain boundaries are sufficiently distant from the ring so as not to effect its evolution. The vortex ring is positioned over the centre of the bottom boundary at a height of $z0_{=}3r_0$. No-slip boundary conditions are used on the bottom vertical boundary, while no-shear boundary conditions are used on all others. Two levels of AMR were used on top of the base grid and the refinement ratio was taken to be $r = 4$. Cells on all levels were tagged for refinement whenever the norm of the velocity exceeded the threshold

$$\|\boldsymbol{\omega}\| > 7.5 \tag{3.9}$$

which is approximately 10% of the maximum initial vorticity. A grid efficiency factor of 90% was used and regridding was performed every two time-steps on all levels. The initial setup is shown in Figure 3.39.

For the next two experiments with $Re_{\text{tr}} = 1500$, and $Re_{\text{tr}} = 2000$, one quarter of the ring was simulated using symmetry boundary conditions on a quarter of the full domain, i.e. a $2 \times 2 \times 2$ box with the vortex ring centred over one corner at a height of $z_0 = 3r_0$. This allowed for a higher resolutions to be achieved without significant extra computational cost. The $Re_{\text{tr}} = 1000$ experiment was also simulated on a half domain and quarter domain using symmetry boundary conditions to determine what effect, if any, this technique might have on the results of the higher Reynolds number cases.

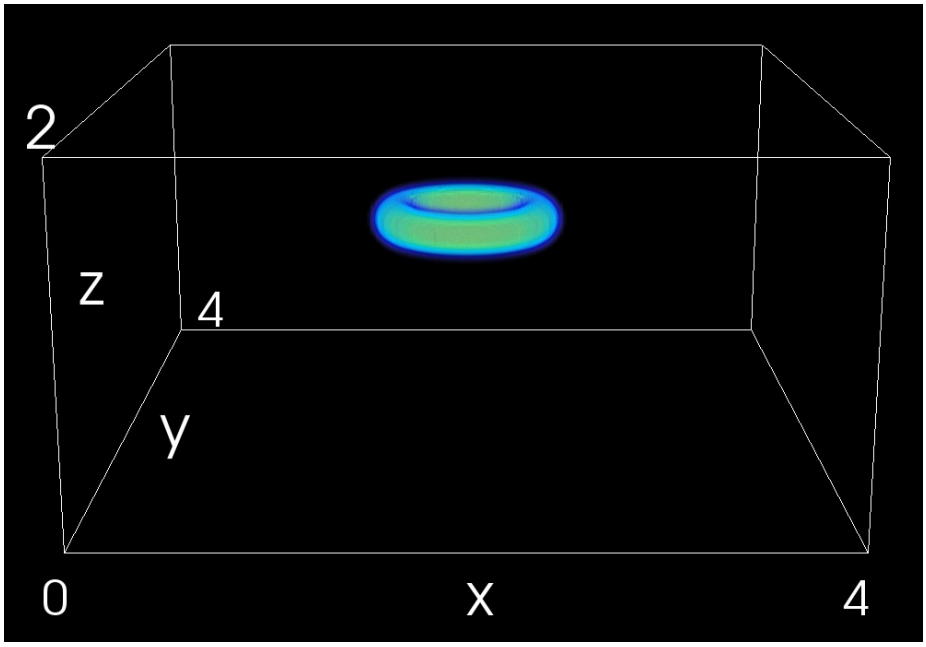

Figure 3.39: The initial setup for the vortex ring wall collision simulations.

The initial conditions were not seeded with a perturbation or any noise as it has been documented in the literature that using a rectangular grid for a vortex ring wall collision results in a perturbation of wavenumber four [54]. We will examine what effect AMR has on this result.

3.4.2 Results

Re 1000

For the first set of simulations, the Reynolds number was taken to be $Re_{\text{tr}} = 1000$. This simulation was performed on the full domain using a base grid resolution of $32 \times 32 \times 16$, $48 \times 48 \times 24$ and $64 \times 64 \times 32$ grid cells. Computations for the first two resolutions were carried out on the machine "Hood" described in Section 1.4. The lowest resolution run required 4 hours of wall-clock time on 24 cores. The second run required 37.5 hours of wall-clock time on 24 cores. The highest resolution run was conducted on the Sharcnet cluster described in Section 1.4 and required 81.2 hours of wall clock time on 64 cores.

Figures 3.40 through 3.48 show a time series of the vortex ring collision with the bottom

boundary. The magnitude of the vorticity field, $\|\boldsymbol{\omega}\|$, is plotted in three dimensions. Hotter colours correspond to higher vorticity. The top panel of each figure shows a side view of the domain, while the bottom panel shows the ring from above. Both the top and bottom panels occur at the same instant and have the same colour gradient.

Figure 3.40 shows the PVR shortly before it interacts with the wall. As the ring approaches the boundary, it induces vorticity of opposite circulation in the viscous boundary layer. This vorticity rolls up and, as the boundary layer separates, forms the SVR. This process can be seen in Figures 3.41 and 3.42. The SVR migrates upwards and over the top of the PVR and develops a wavy instability , as seen in Figure 3.43. It is important to note that PVR does not actually make contact with the bottom boundary; rather, it "rebounds" and expands parallel to the wall.

At this Reynolds number, as the SVR migrates to the middle of the primary vortex ring it forms a loop pattern as described in [15]. This can be seen clearly in Figure 3.45 along with the generation of a TVR which has the same circulation as the PVR. Next, the SVR is pulled under and over the stronger PVR. This interaction continues in an organized fashion (Figures 3.46 throught 3.48) until eventually all three rings are diffused by viscosity. Notice in Figure 3.48 that the structures created by the vortex rings interacting remain quite stable late into the simulation.

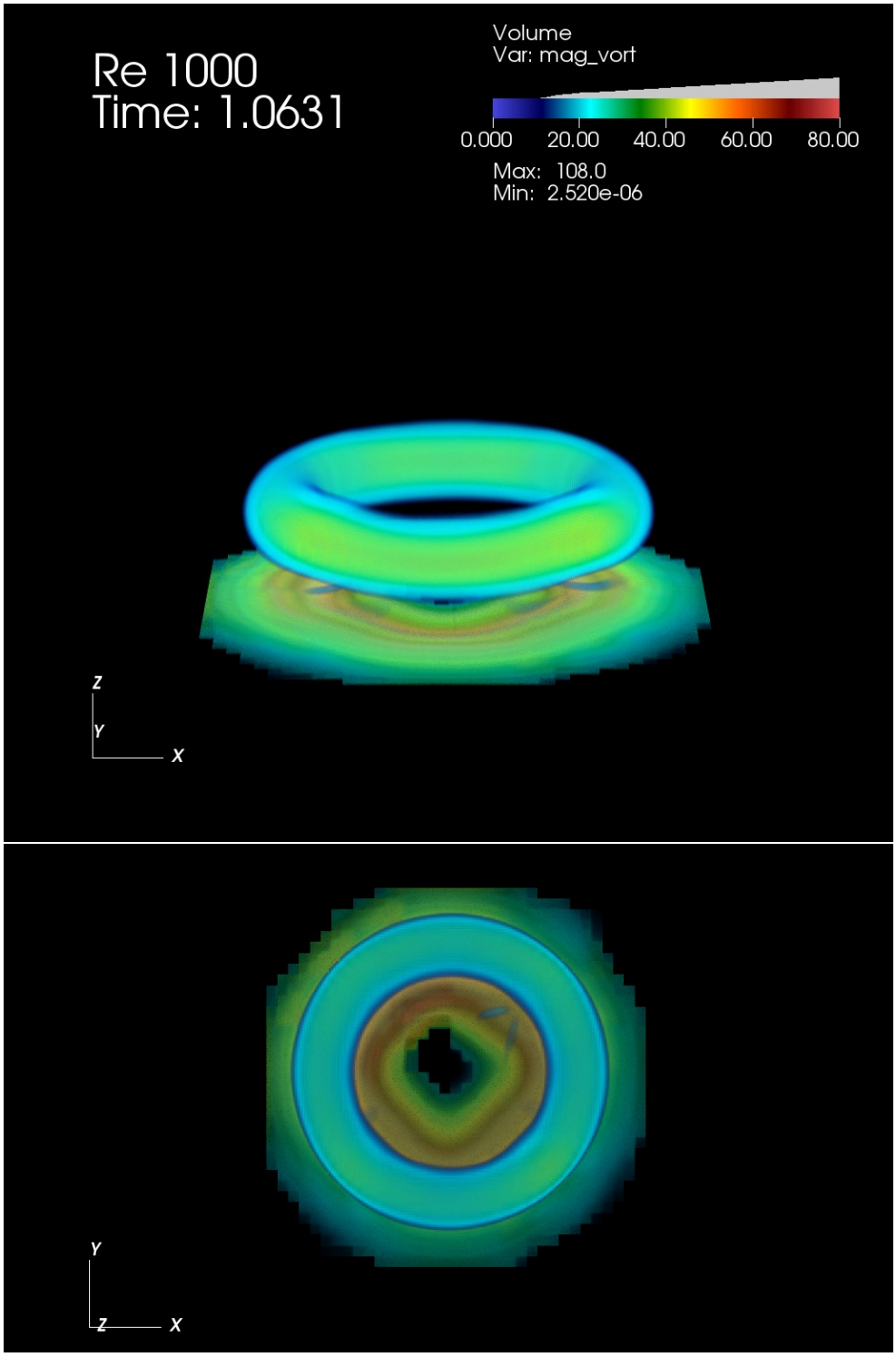

Figure 3.40: The primary vortex ring at it begins to interact with the boundary for $Re_{\text{tr}} = 1000$. The upper panel shows a side view of the ring, while the lower panel shows a top-down view.

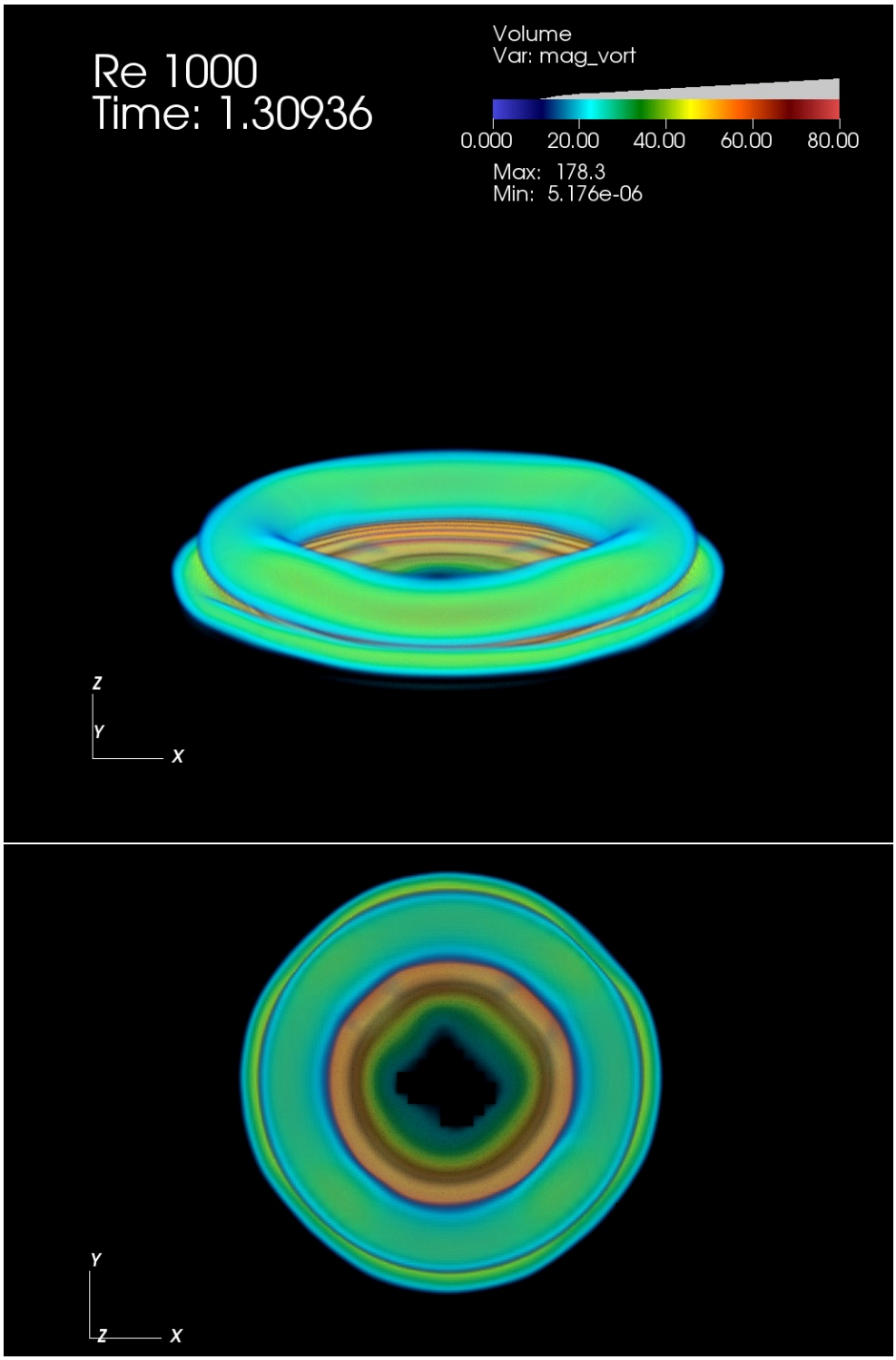

Figure 3.41: The primary vortex ring induced vorticity in the boundary layer with opposite circulation.

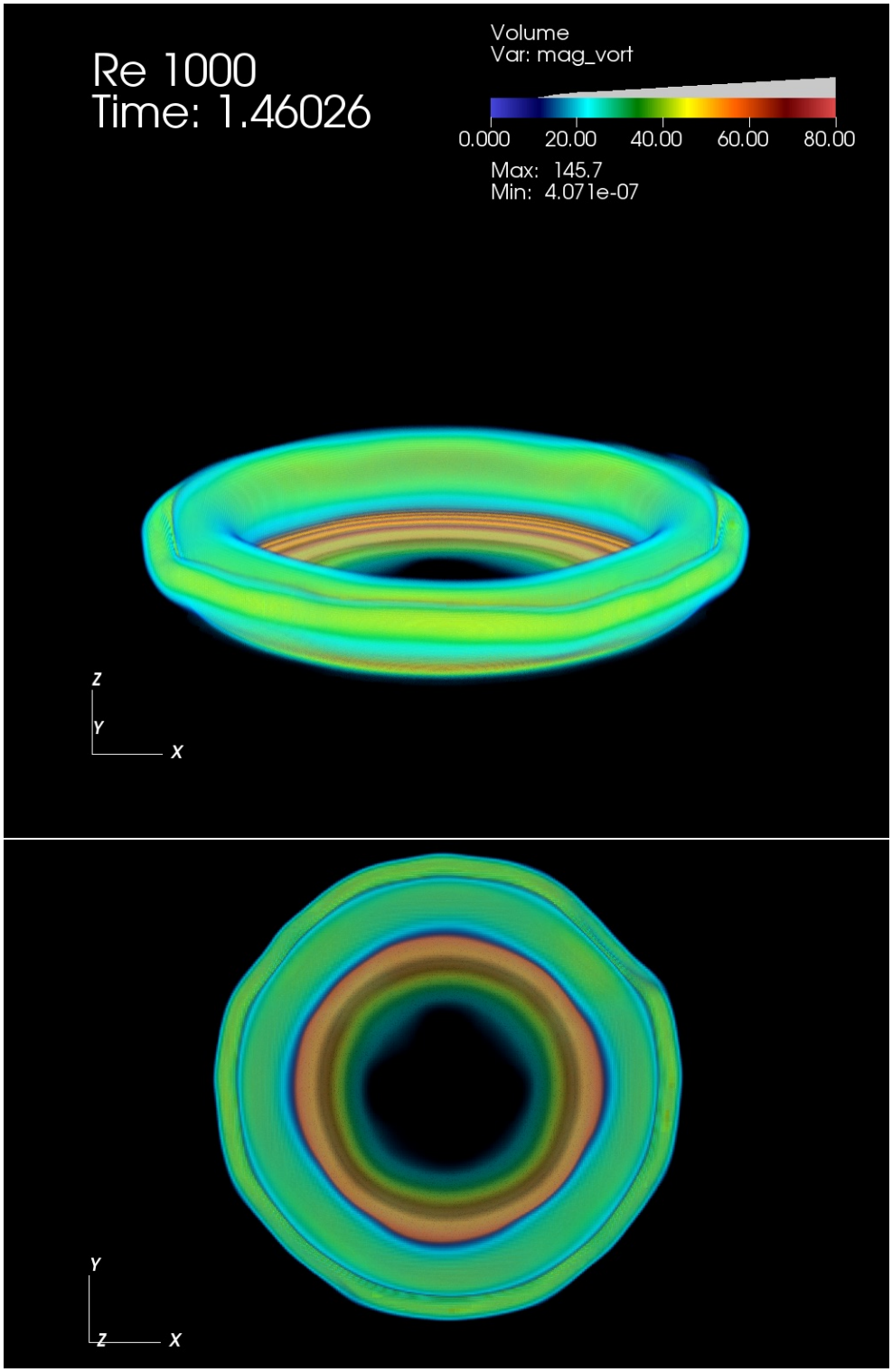

Figure 3.42: As the vorticity induced in the boundary layer separates, it forms a secondary vortex ring. This secondary ring migrates up and over top of the primary ring.

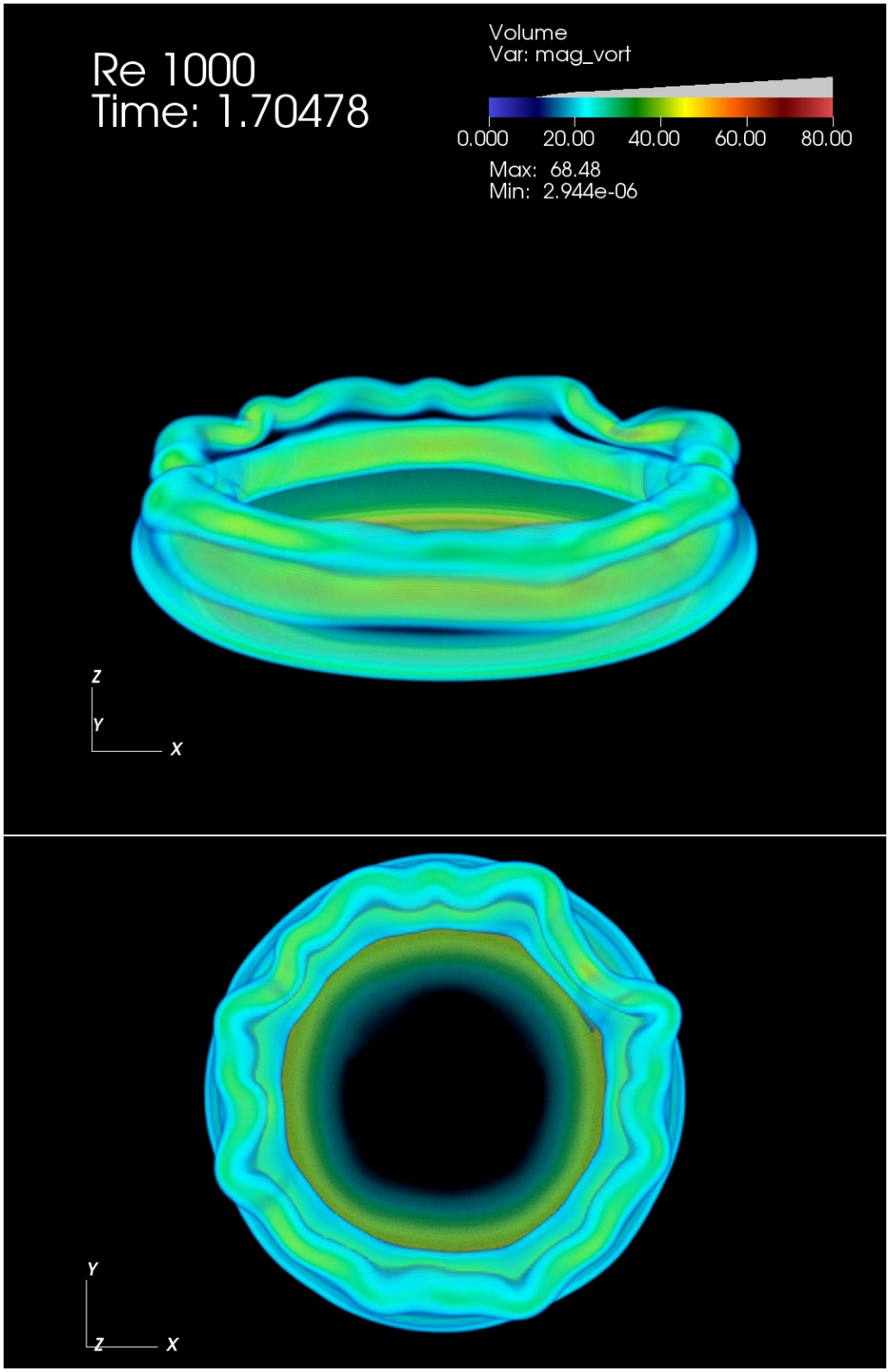

Figure 3.43: As the secondary vortex ring migrates over top of the primary ring it begins to develop a wavy instability.

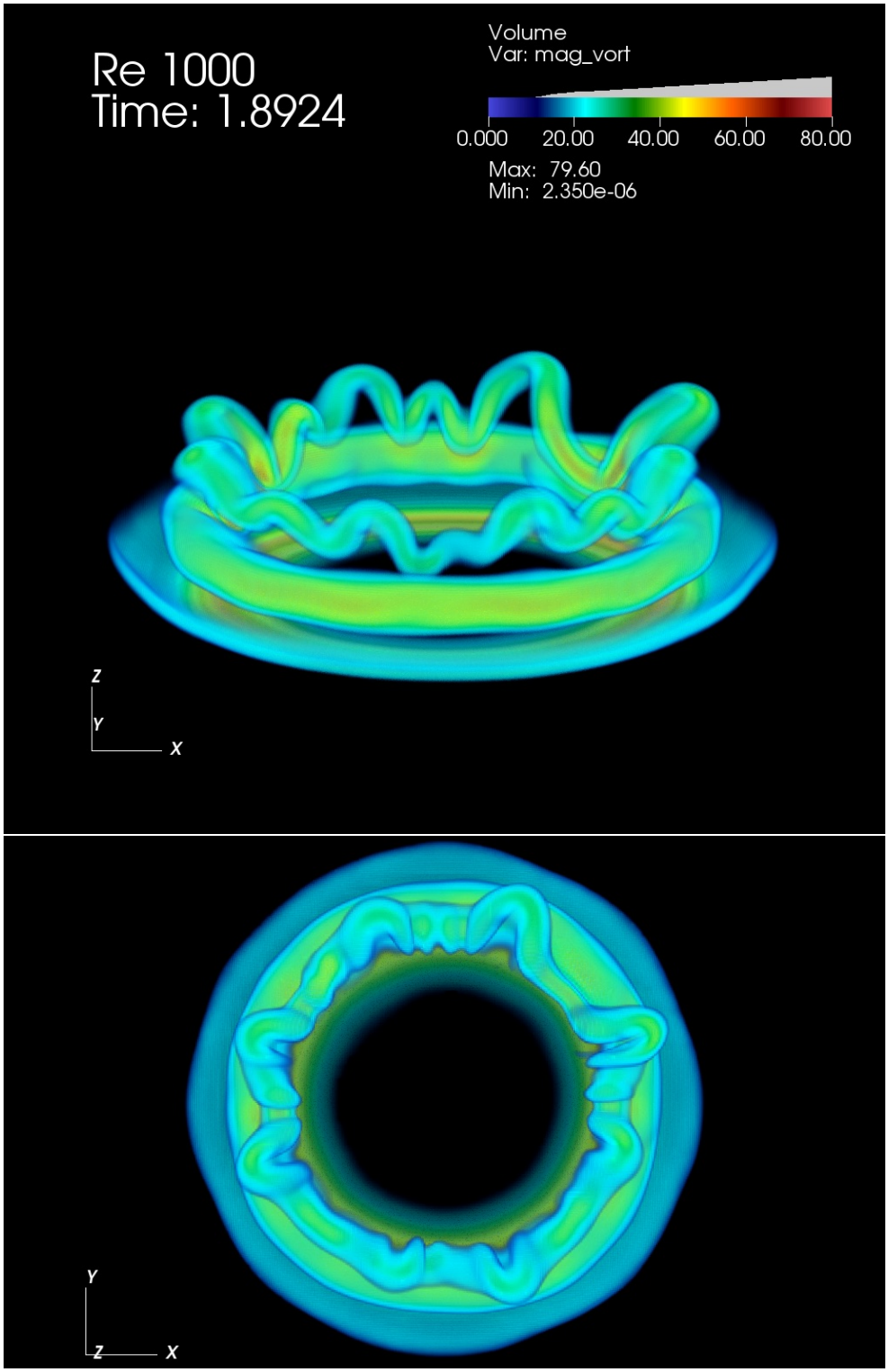

Figure 3.44: The wavy instability of the secondary vortex ring grows as it continues to migrate into the centre of the primary vortex ring.

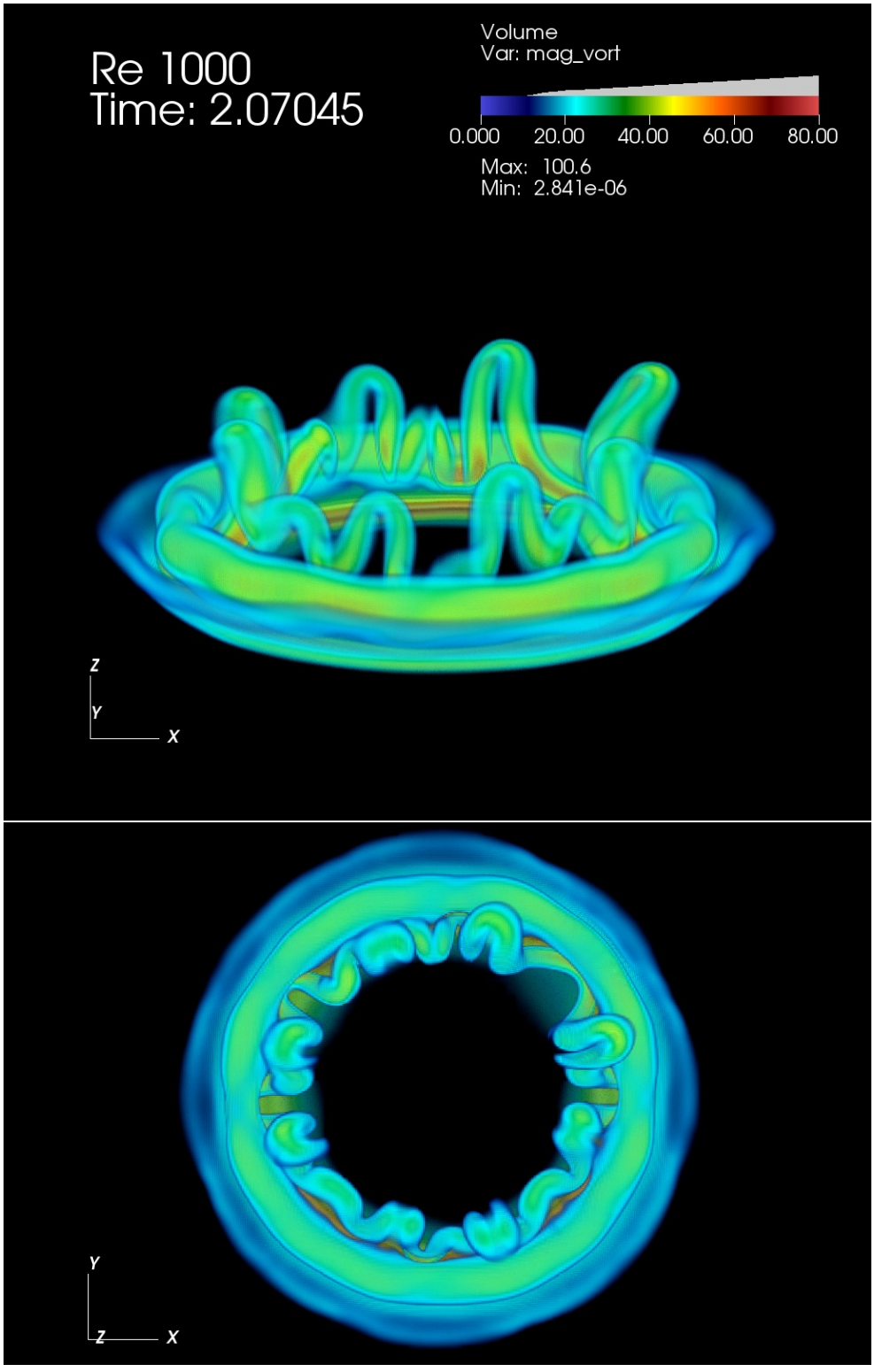

Figure 3.45: Once the secondary vortex ring reaches the centre of the primary vortex ring it forms the classic loop structures observed in experiments. Additionally, the development of a tertiary vortex ring can be observed on the outside of the primary ring.

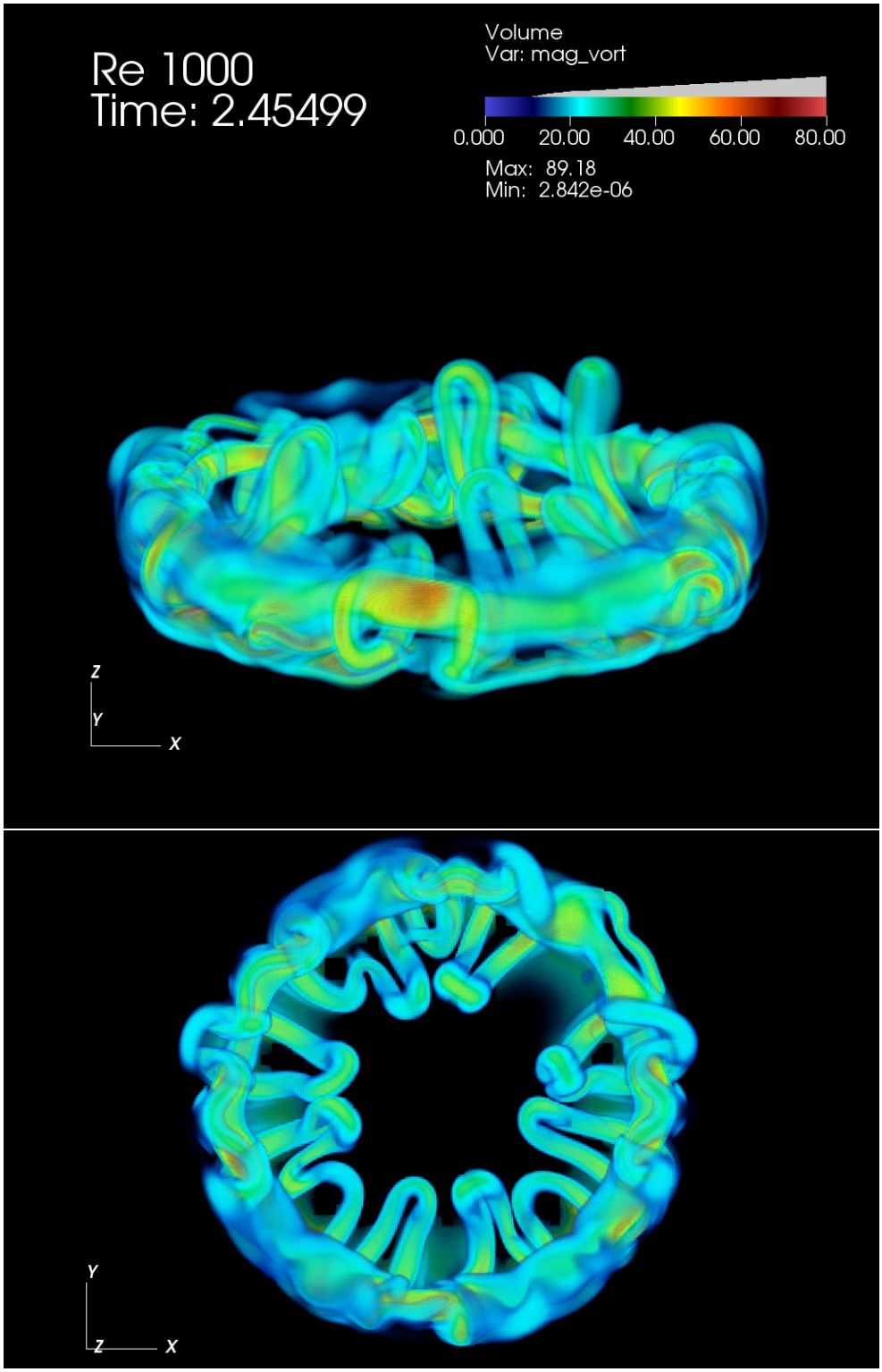

Figure 3.46: The secondary vortex ring is stretched under and over the primary ring causing local increases in vorticity.

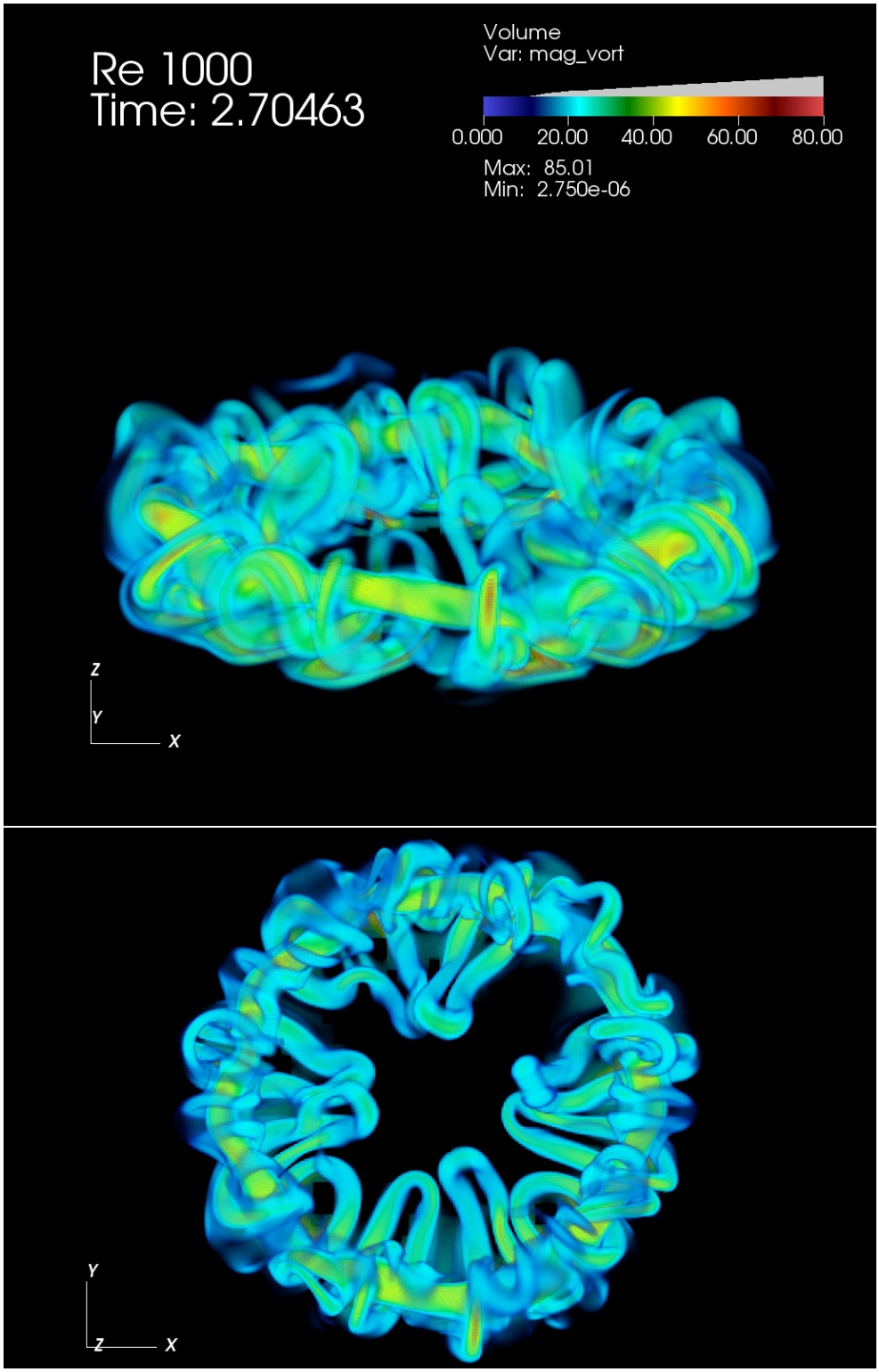

Figure 3.47: The primary and secondary vortex rings continue to stretch each other. The flow remains very organized.

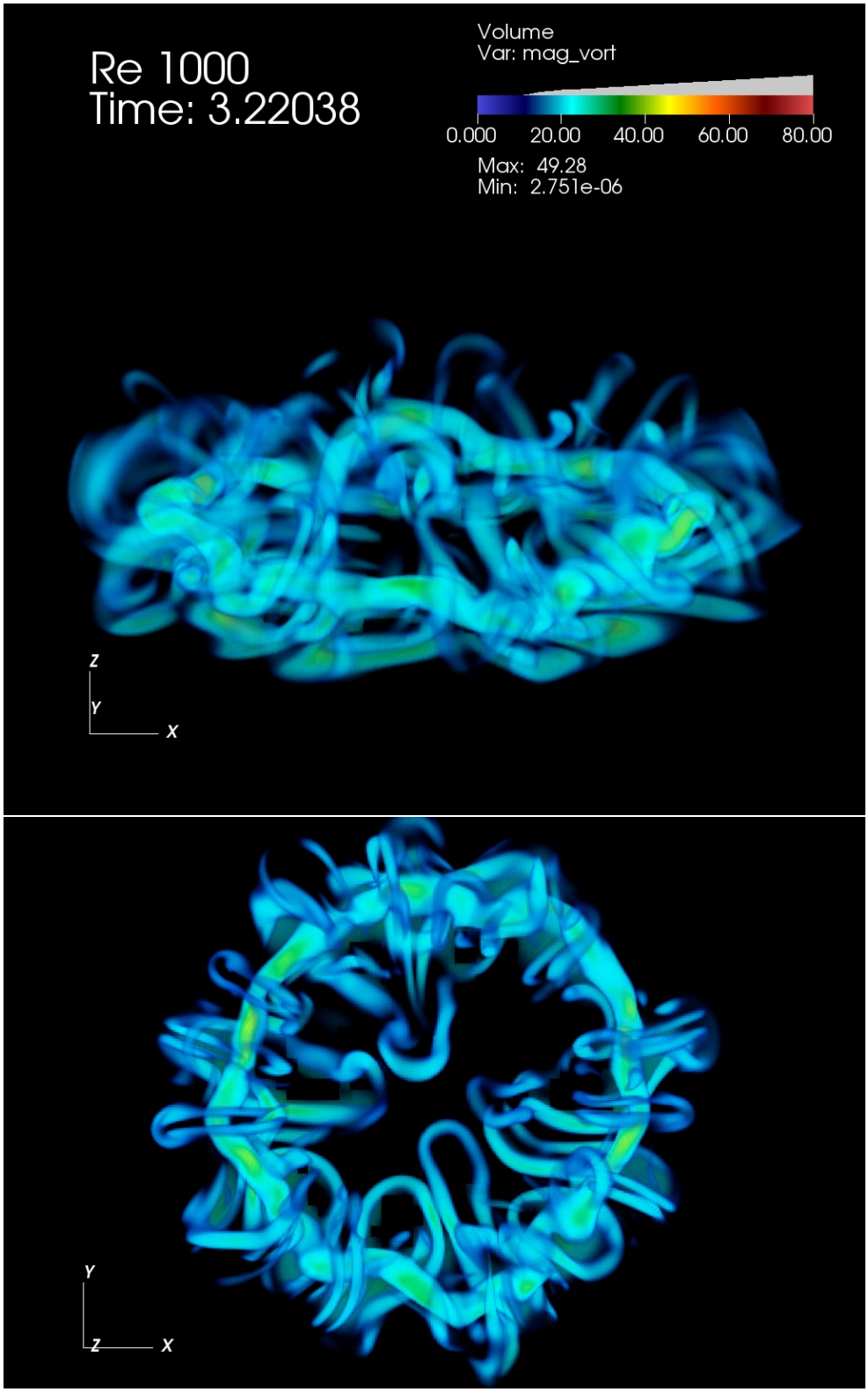

Figure 3.48: Late in the simulation the flow continues to remain organized as viscosity slowly dissipates the vorticity field. The primary vortex ring structure is still clearly visible.

Re 1500

The simulation was conducted on a quarter domain using $Re_{\text{tr}} = 1500$ and symmetry boundary conditions. The base grid resolution for this run was $32 \times 32 \times 32$ grid cells. Calculations were carried out on the machine "Boogaloo" described in Section 1.4. The simulation required approximate 12 hours of wall-clock time to complete using just 16 cores.

Figures 3.49 through 3.55 show a time series of the magnitude of the vorticity field. Note: the colour gradient was adjusted slightly in this sequence of plots to include more of the cool region as some of the important features are relatively weak. Figure 3.49 shows the generation of the SVR, which has a stronger circulation at this Reynolds number than at $Re_textuptr = 1000$. The SVR again migrates over top of the PVR, as seen in Figure 3.50, and then down into its centre, as seen in Figure 3.51. The waviness of the SVR ring is not as pronounced in this simulation. Figure 3.51 also reveals that a much stronger TVR is generated in this simulation. Figure 3.52 shows the TVR being stretched over the top of the PVR, while simultaneously the SVR is being pulled underneath. The result of this process is a slowing and thickening of all three structures as can be seen in Figures 3.53 and 3.54. By the end of the simulation, seen in Figure 3.55, the structure of the SVR and TVR has broken down into wispy filaments, and the PVR is weak and hardly coherent. Vorticity has not completely diffused at this point however.

Re 2000

In the last experiment, the vortex ring wall interaction is simulated at $Re_{\text{tr}} = 2000$ on a quarter domain using symmetry boundary conditions. The simulation was run using a base grid resolution of $48 \times 48 \times 48$ grid cells. Computations were carried out on the machine "Boogaloo" described in Section 1.4 and required 45.5 hours to complete using 24 cores.

Figure 3.56 shows the generation of the SVR, which has an even strong circulation at this Reynolds number. The SVR again migrates over top of the PVR, developing a wavy instability, as seen in Figure 3.57. In this case however, the SVR is strong enough to induce waviness in the PVR. As the SVR migrates to the centre of the PVR it again forms a loop pattern. The SVR is again pulled under the PVR, however the mutual compression of the PVR and SVR causes an increase in vorticity in the vicinity of the subduction, as seen in the bottom panel of Figure 3.58. A stronger TVR is again generated in this simulation. As the stretching of the SVR continues in Figure 3.59, its circulation continues to intensify. The PVR continues to be stretched in this region as well. Figure 3.60 shows a network of strong vortex filaments are starting to form. This picture has a complex topology, and

while the SVR appears to still be visible in the bottom panel of Figure 3.60, it is not 100% clear where the PVR ends and the SVR begins. In Figure 3.61 we see that the entire structure is starting to break down into a cloud of small vortex filaments, until finally at the end of the simulation the cloud of vortex filaments appears to be random in structure, as seen in Figure 3.62. Notice the significant drop in the intensity of vorticity in Figure 3.62: the maximum in the vorticity field has decreased by over 50% in a very short amount of time.

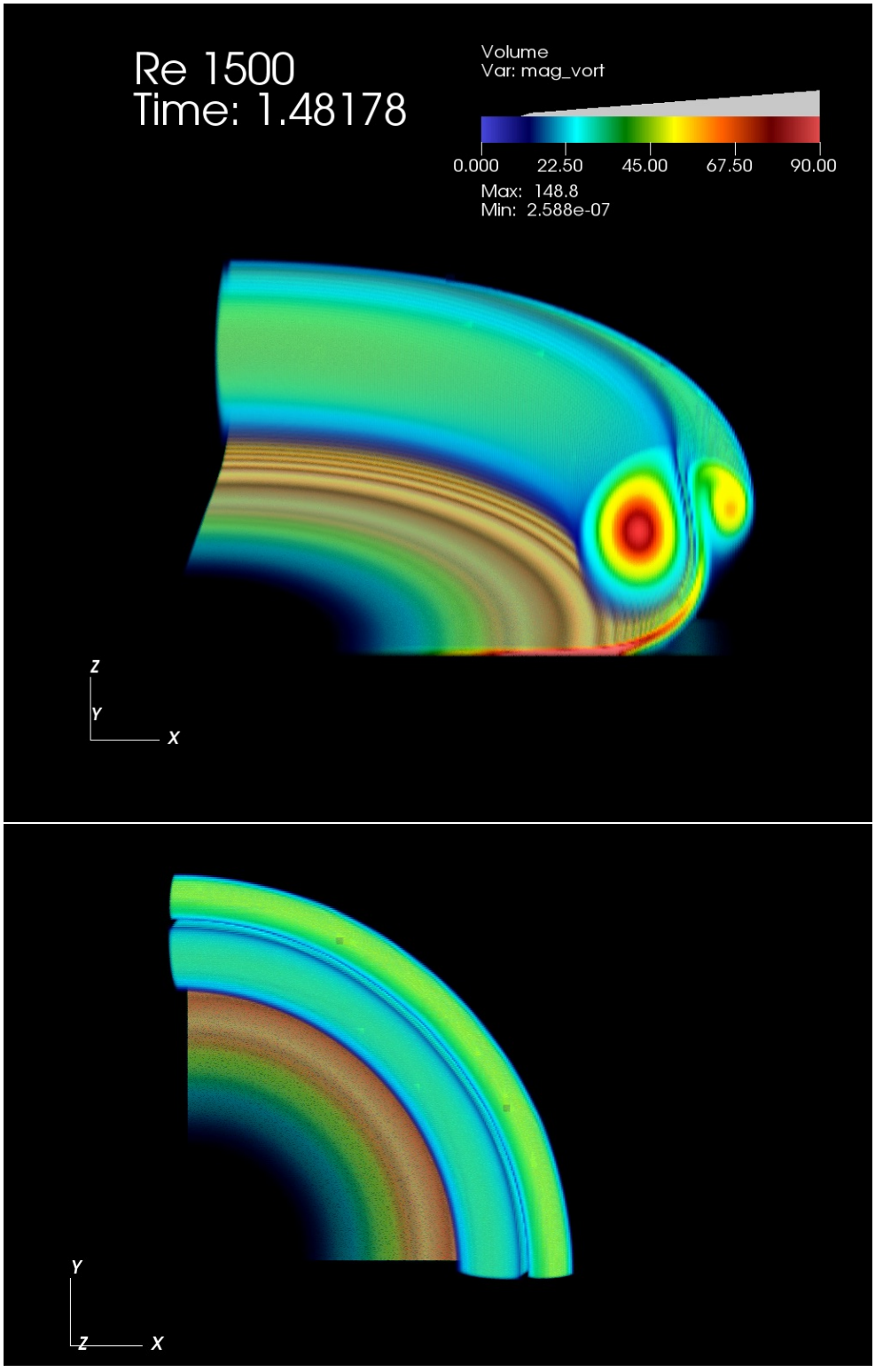

Figure 3.49: The generation of the secondary vortex ring for the $Re_{\text{tr}} = 1500$ case. The circulation of the secondary ring is strong that in the previous case.

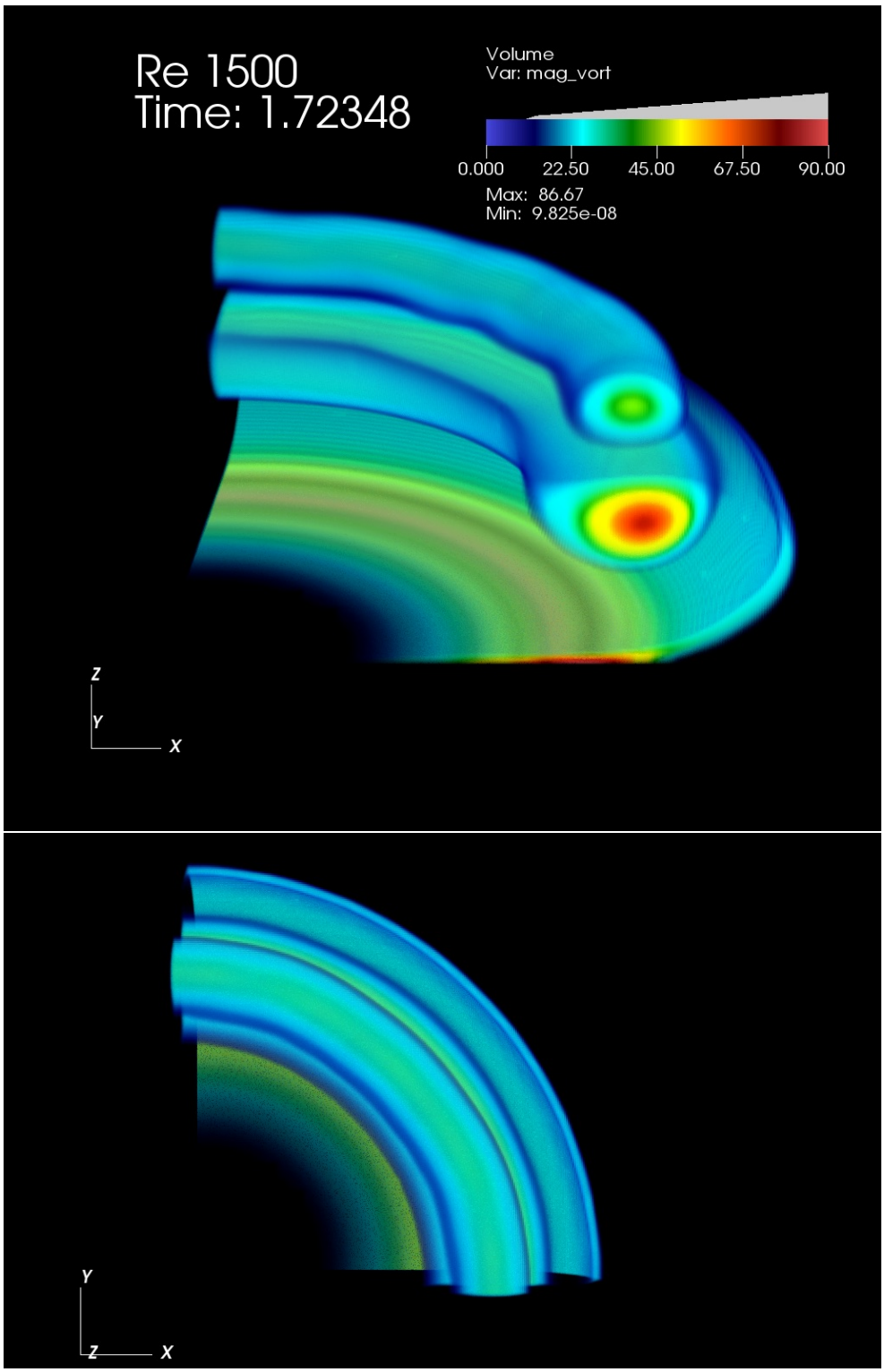

Figure 3.50: The secondary vortex ring migrates over top of the primary vortex ring.

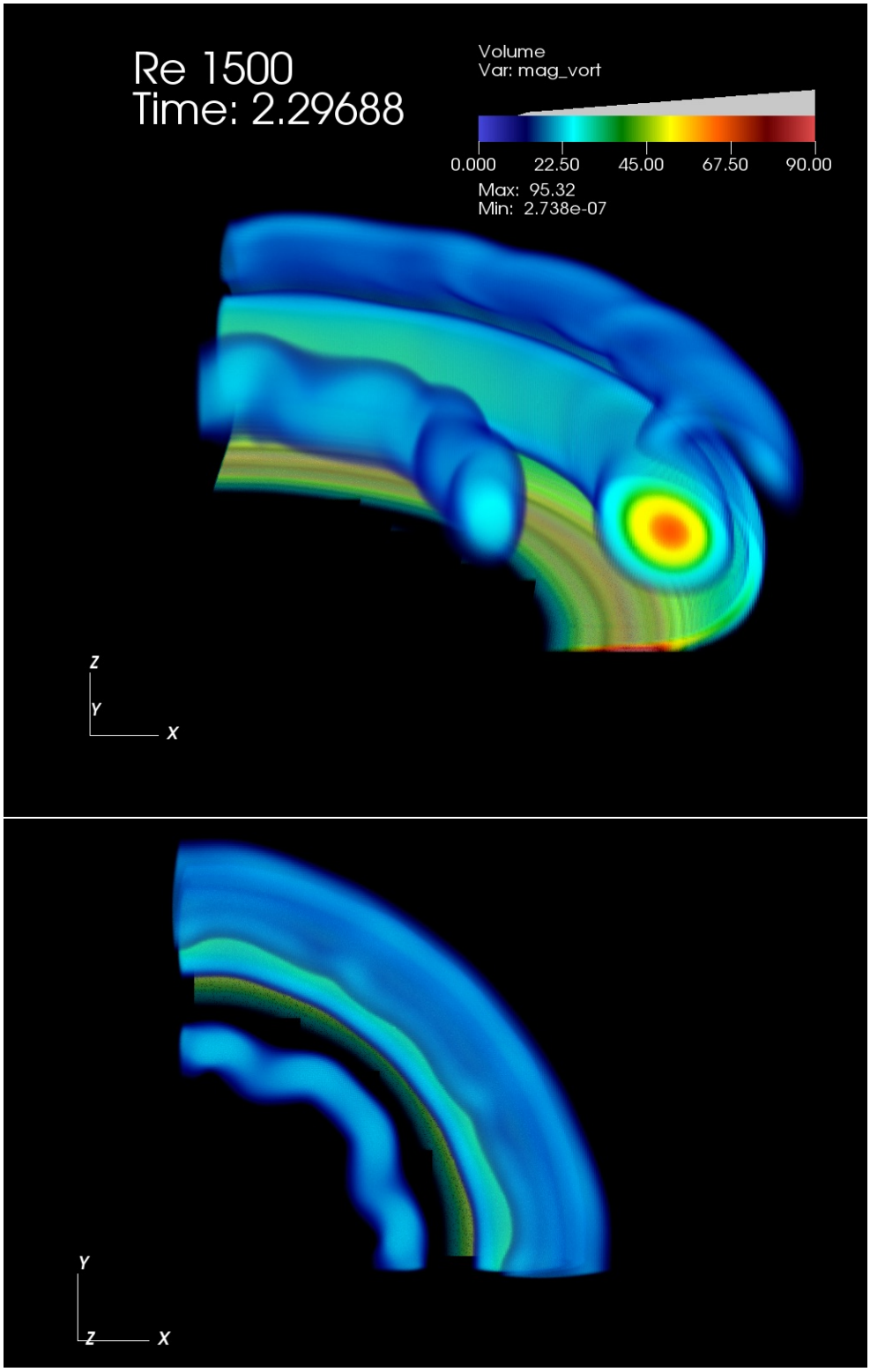

Figure 3.51: The secondary vortex ring has migrated into the centre of the primary ring. Here it is less wavy that in the $Re_{\mathrm{tr}} = 1000$ case. Additionally, the tertiary vortex ring generated at this Reynolds number is stronger and can be seen migrating over top of the primary vortex ring.

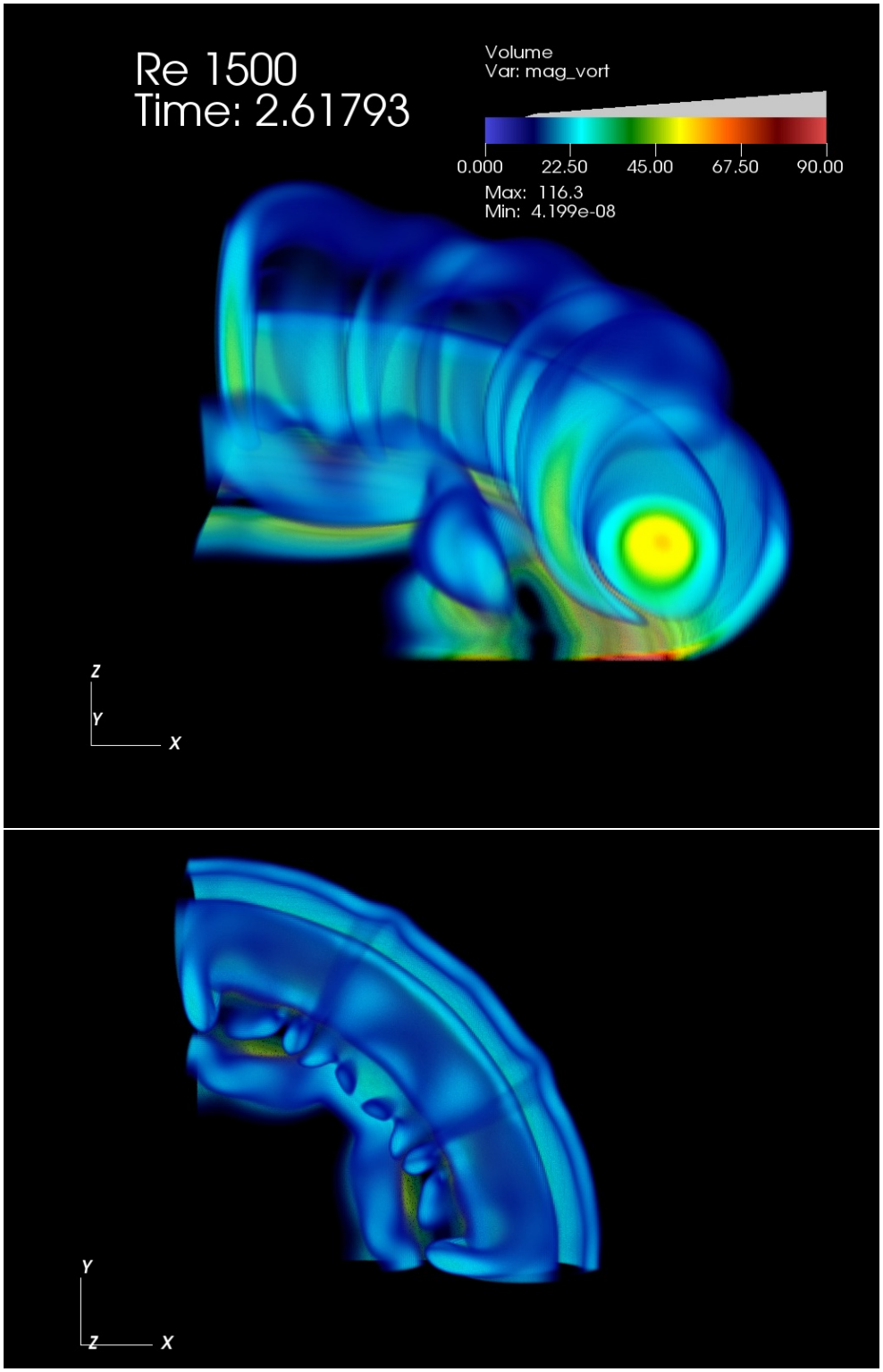

Figure 3.52: The tertiary vortex ring is stretched down over top of the primary vortex ring and is pulled beneath it along with the secondary vortex ring.

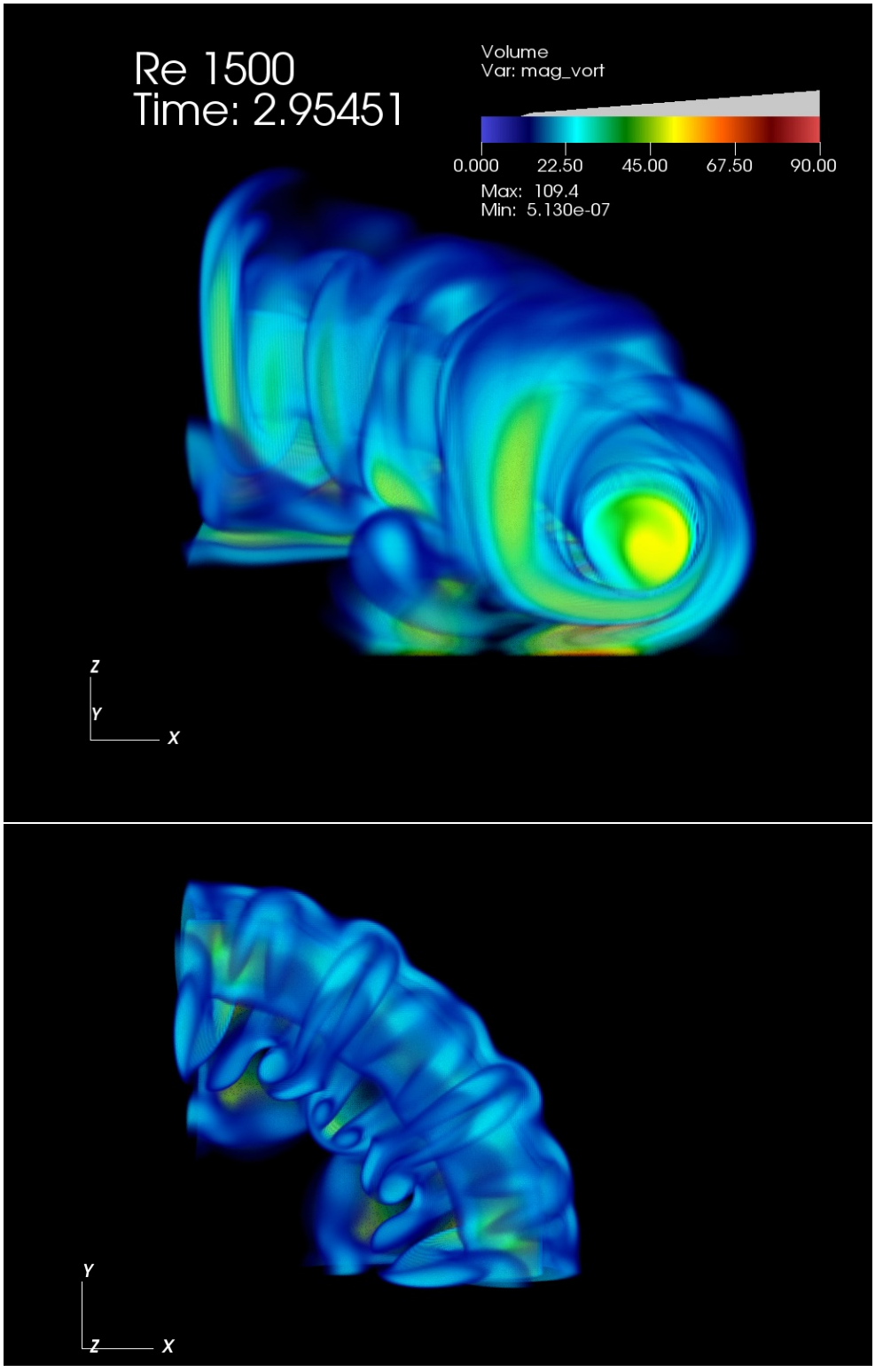

Figure 3.53: As the secondary and tertiary rings continue to be stretched and wrapped around the primary ring.

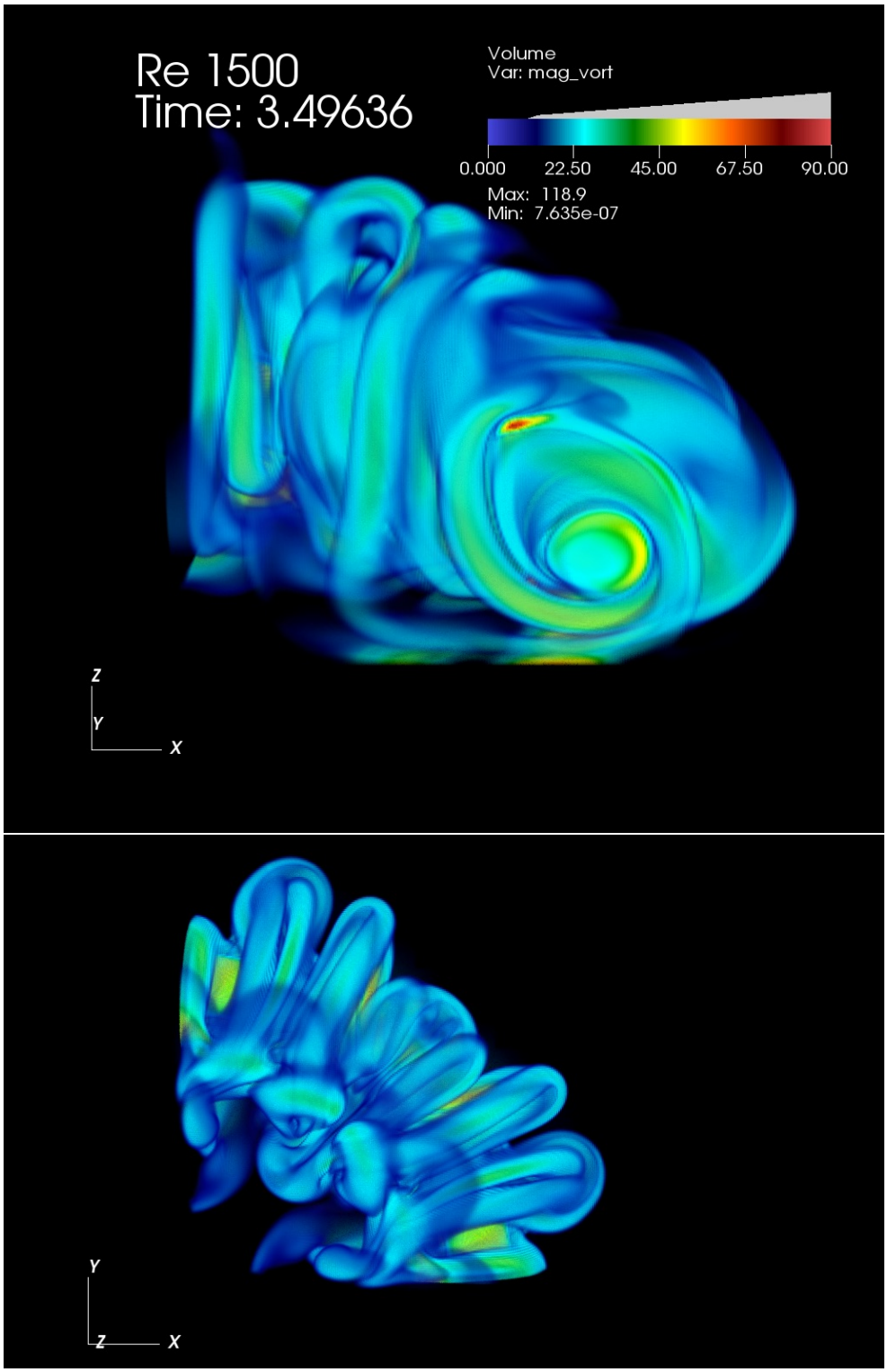

Figure 3.54: The structures wrapped around the primary vortex ring begin to thicken.

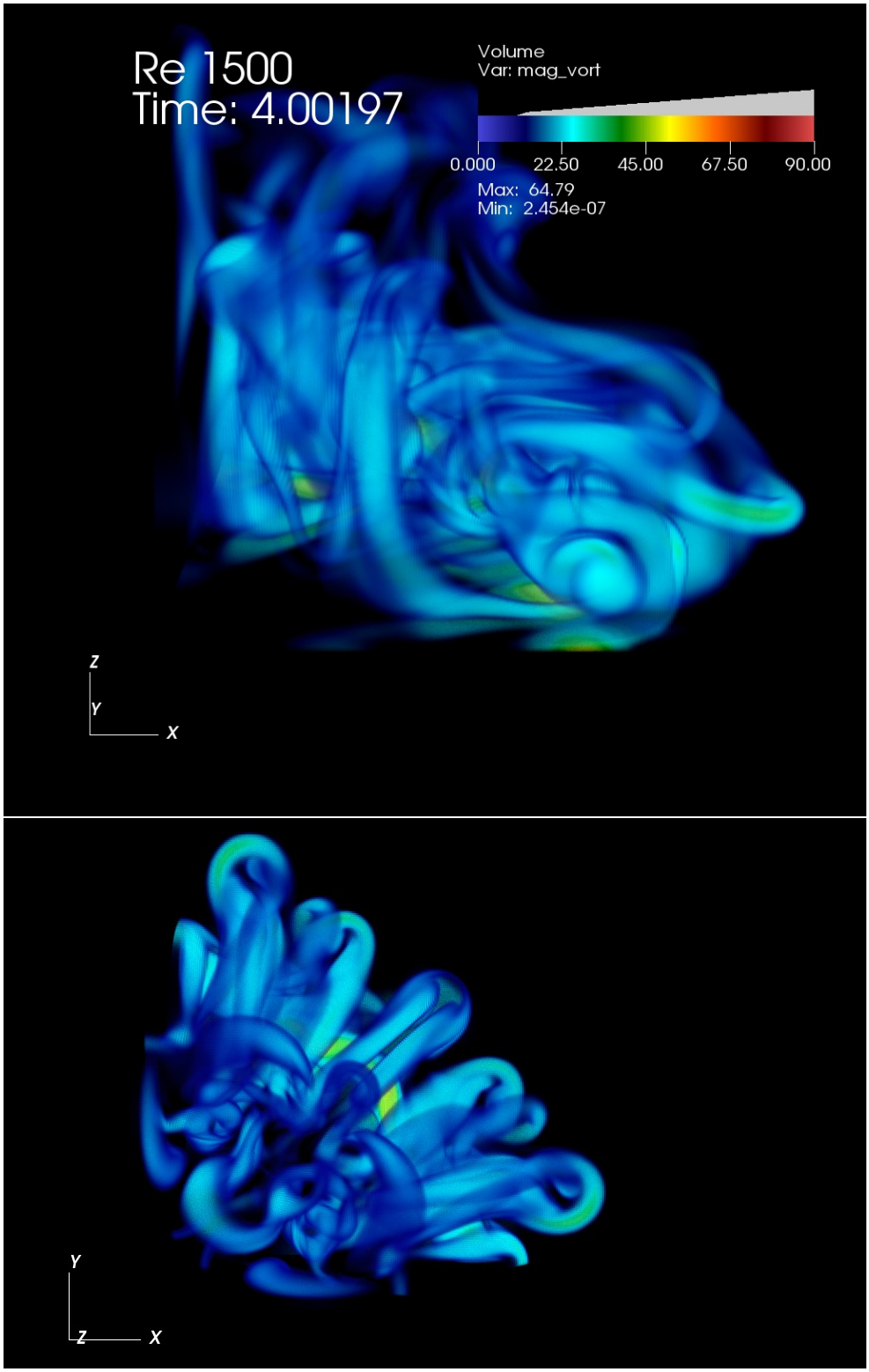

Figure 3.55: By the end of the simulation the structure of the SVR and TVR has broken down into wispy filaments, and the PVR is weak and hardly coherent. Vorticity has not completely diffused at this point however.

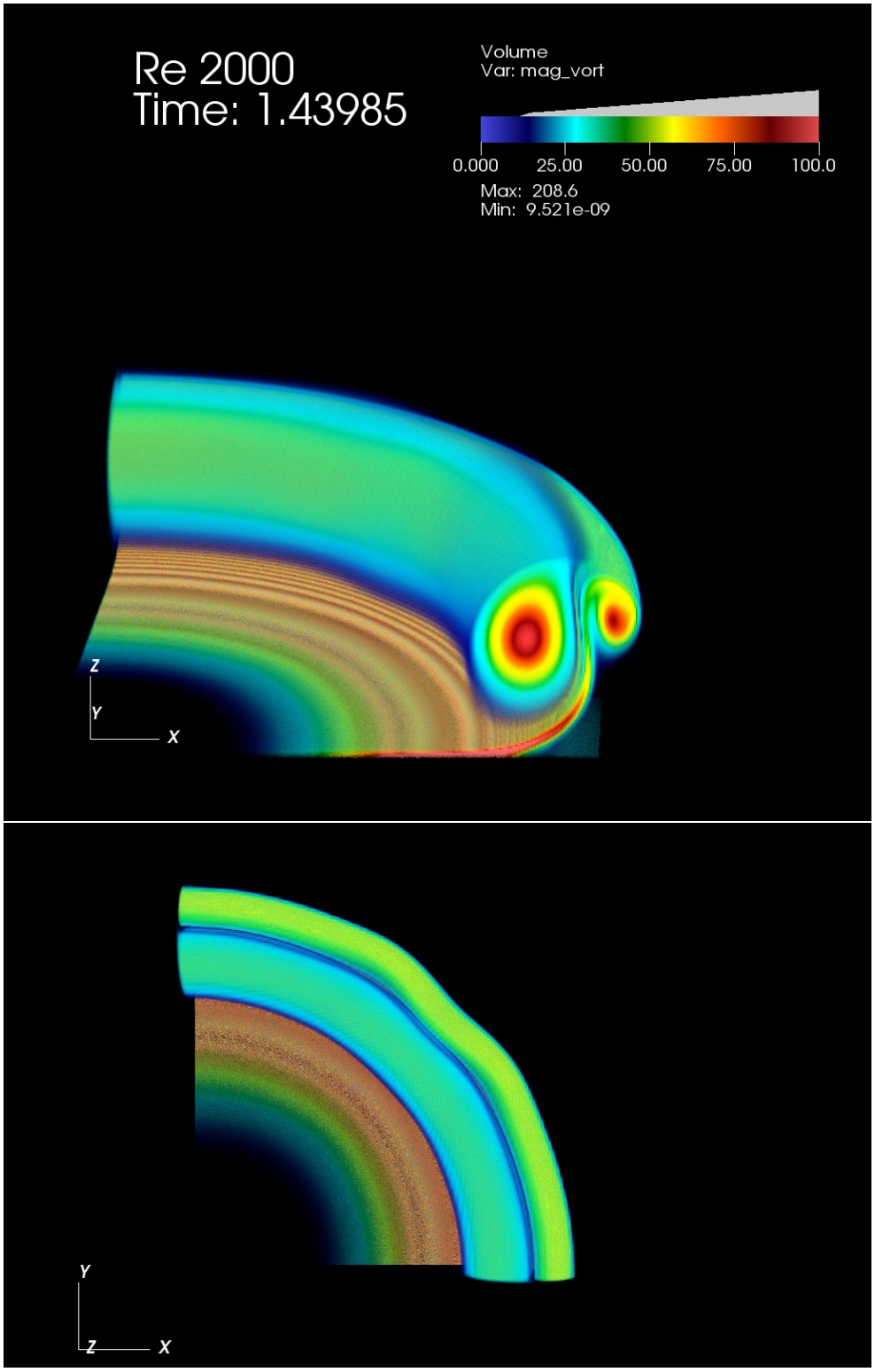

Figure 3.56: The generation of the secondary vortex ring at $Re_{\mathrm{tr}} = 2000$. The secondary ring has a core vorticity of the same magnitude as the primary ring in this case.

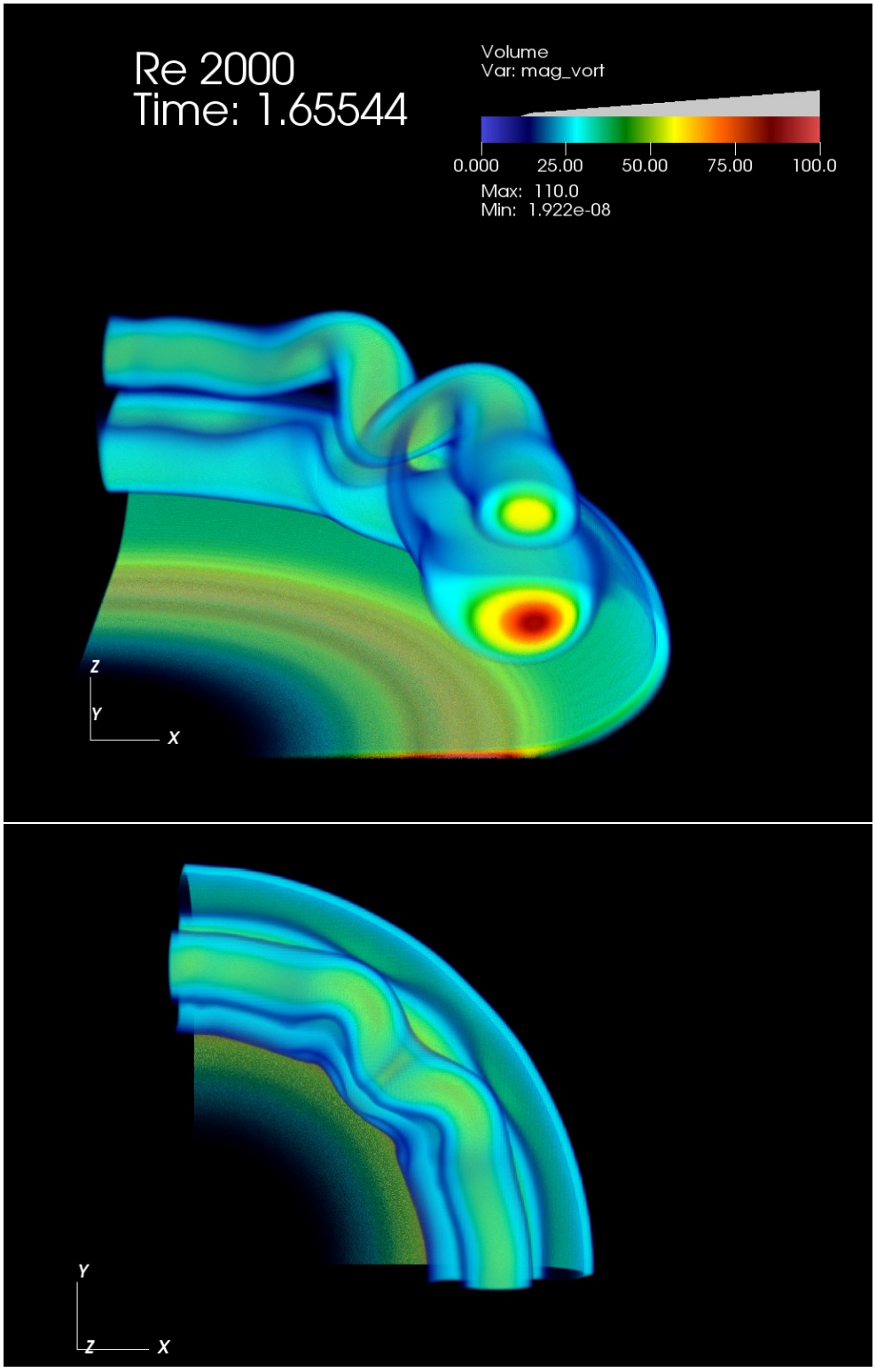

Figure 3.57: As the secondary vortex ring migrates over the top of the primary ring it undergoes a wave instability. The primary vortex ring is deformed by the secondary ring.

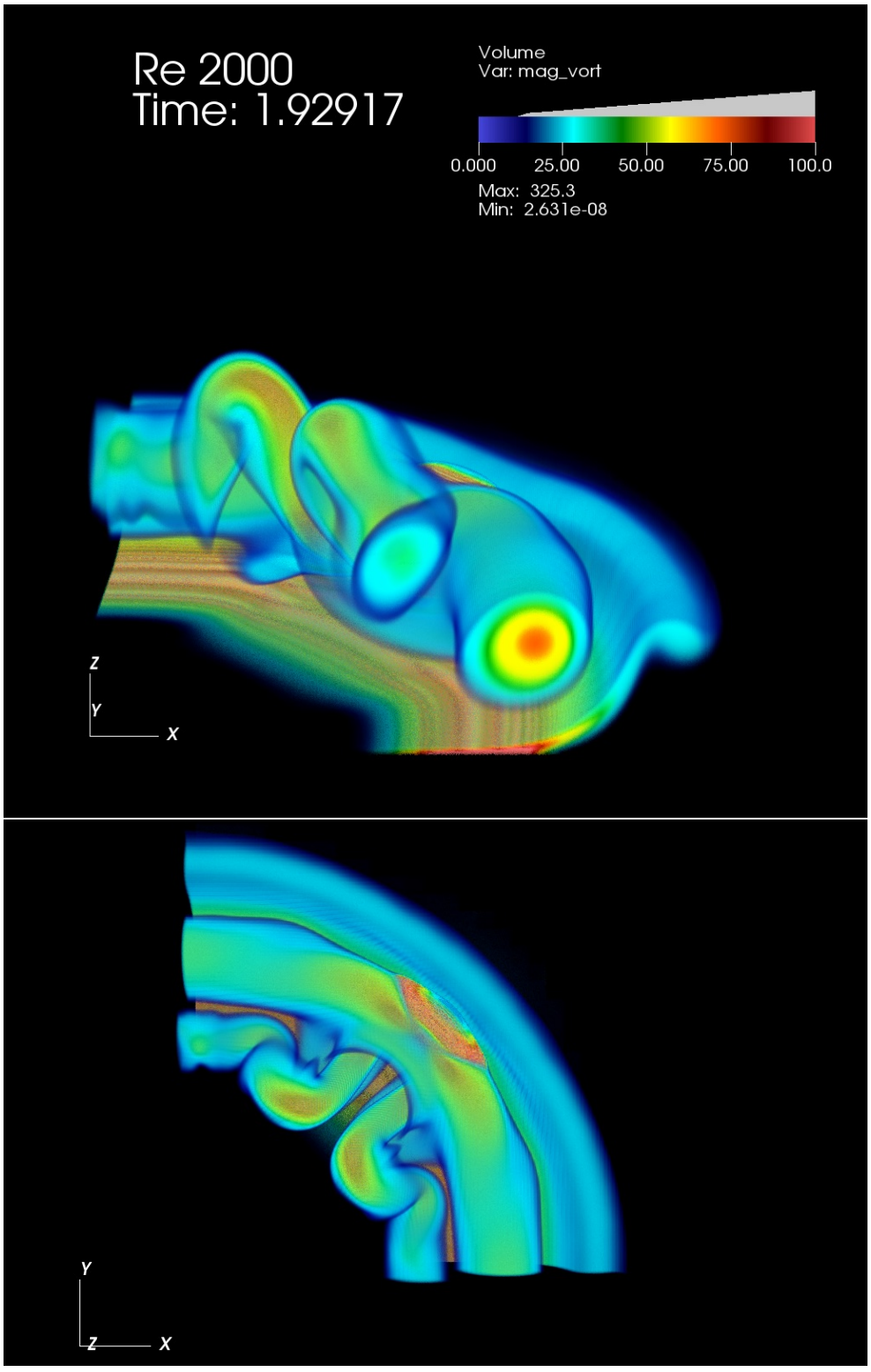

Figure 3.58: The secondary vortex ring is puled under the primary ring and its circulation is intensified due to the stretching process.

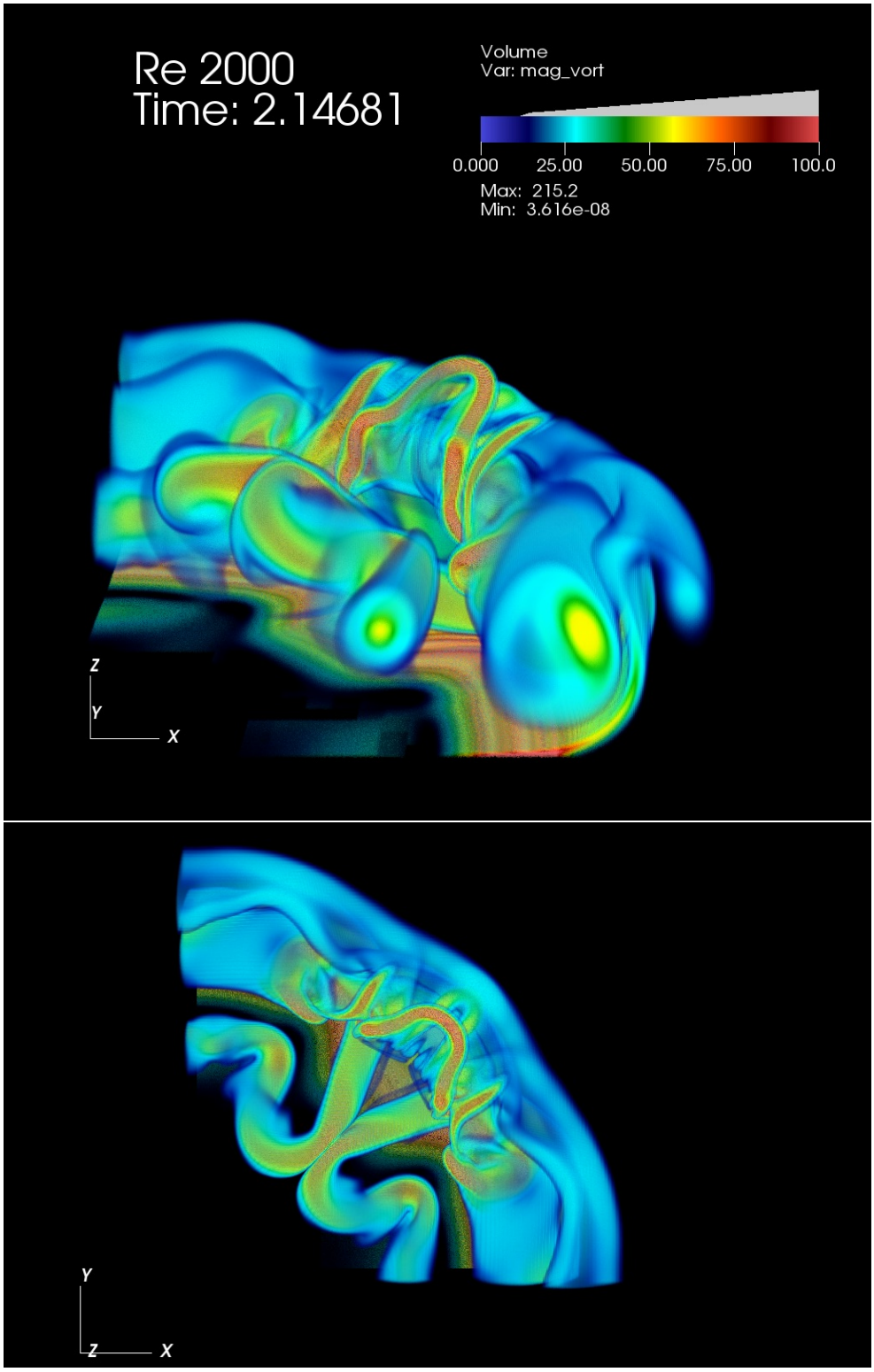

Figure 3.59: As the secondary vortex ring continues to be stretched thinner, it mutually stretches the primary ring causing it to deform. The weaker tertiary vortex ring has formed and begins to migrate upwards.

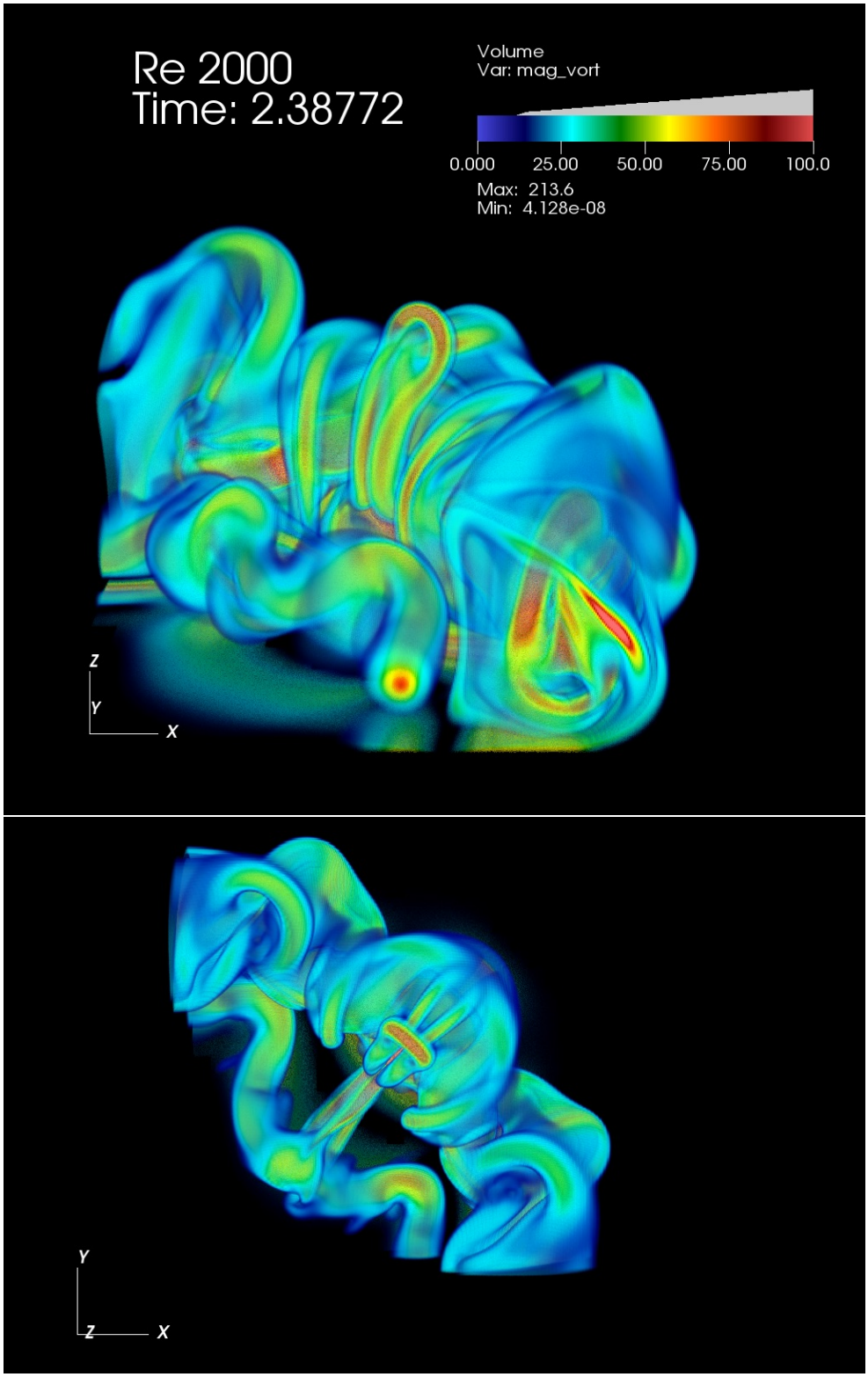

Figure 3.60: A complex network of strong thin vortex filaments has formed. While the structure of the secondary vortex ring is still somewhat visible in the lower panel, the primary vortex ring is no longer discernable.

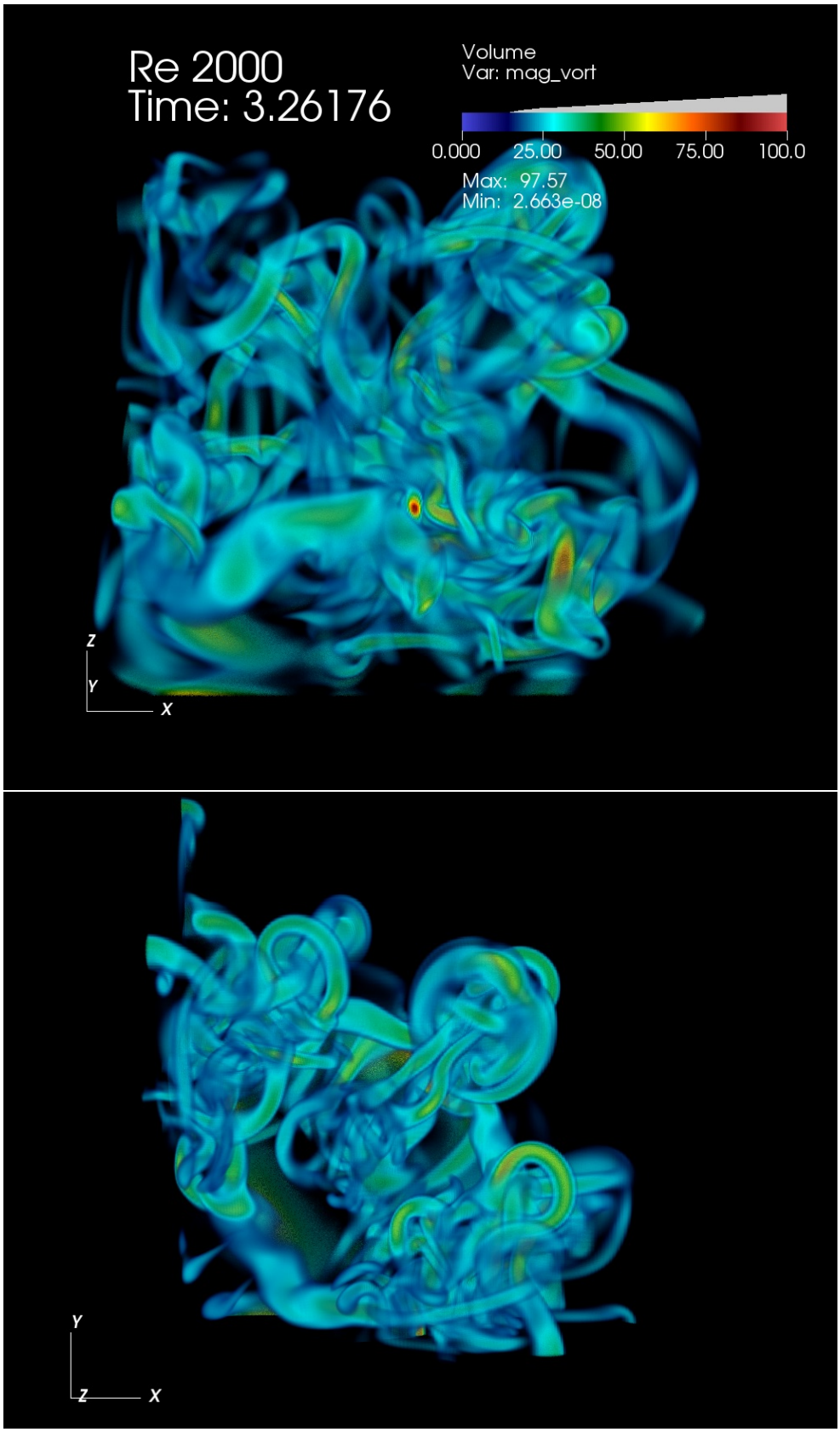

Figure 3.61: The flow as begun to unravel into a cloud of small vortex filaments.

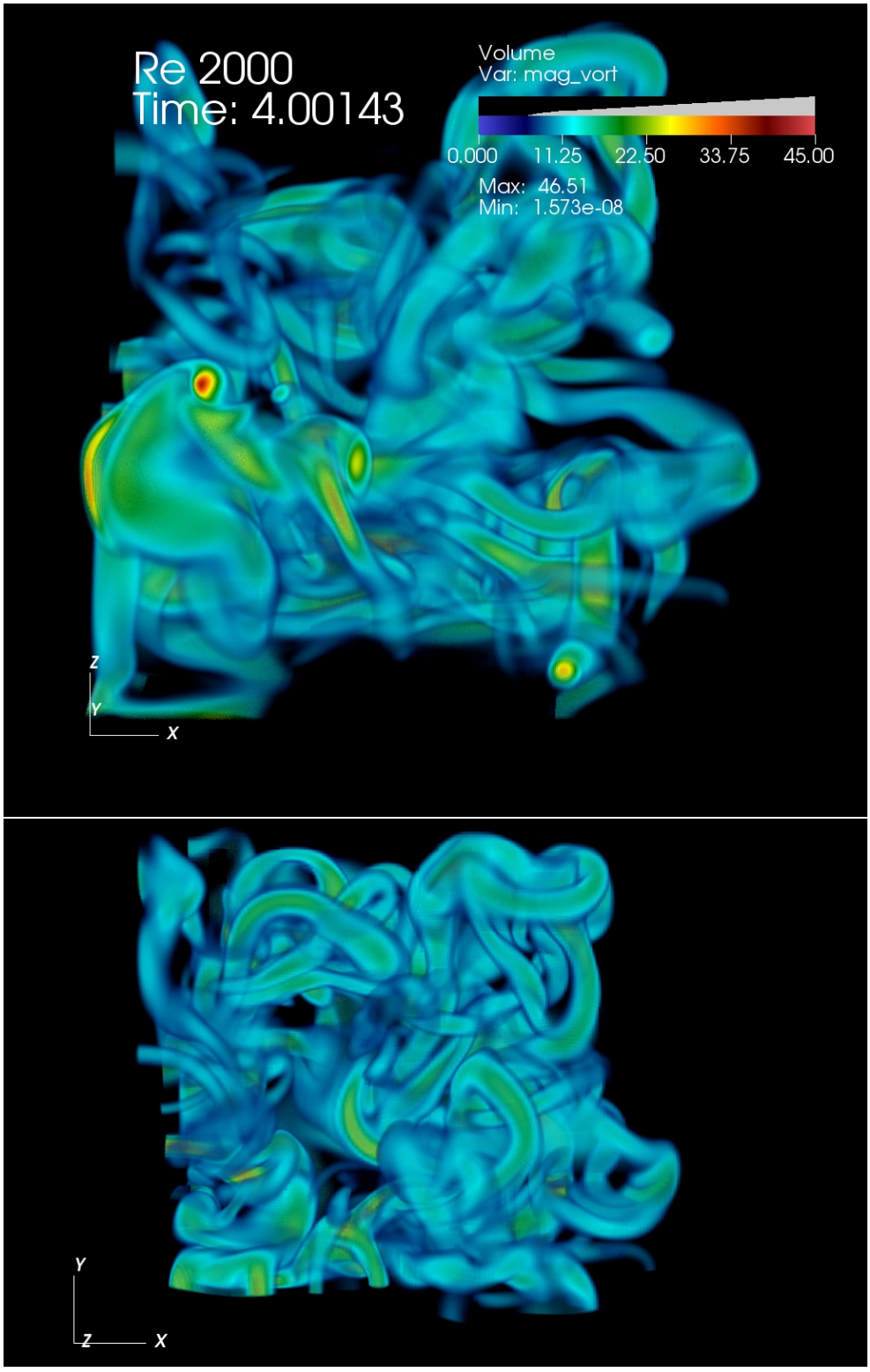

Figure 3.62: The cloud of small vortex filaments has continued to diffuse into the domain while the strength of their circulation has diminished significantly. Notice the maximum value on the colour axis has been lowered to 45 from 100.

Kinetic Energy and Enstrophy

To better compare the dynamics of the above simulations, the total kinetic energy and total enstrophy curves for all three experiments are plotted on the same axes in Figure 3.63. In order to give meaning to the comparison, the total kinetic energy and total enstrophy curves for the $Re_{\text{tr}} = 1000$ are taken from the simulation performed on the quarter domain. In the top panel we see the total kinetic energy curves. As expect, all three curves decay from the start to the finish of each simulation. While the $Re_{\text{tr}} = 1000$ and $Re_{\text{tr}} = 1500$ curves have roughly the same shape, the kinetic energy of the $Re_{\text{tr}} = 2000$ simulation quickly decreases shortly after time $t = 2$. The lower panel showing the total enstrophy curves offers a hint a what is happening. All three simulations begin with the same amount of total entrophy. At times between $1.35 < t < 1.4$ all three curves attain their first maximum. This roughly corresponds to the time when the vortex ring initially interacts with the wall. As was observed for the two dimensional vortex dipole simulations, the maximum enstrophy value is larger for higher Reynolds numbers. While the total enstrophy curves for the $Re = 1000$ and $Re = 1500$ decrease from their initial maximum and remain lower than this value for all times after, the $Re = 2000$ curve sees a second significant spike in enstrophy production around time $t = 2$. This occurs shortly after the time when the SVR is pulled under the PVR and the series of strong stretching events begins. The kinetic energy decreases most rapidly after this point as the strong stretching breaks the PVR and SVR down into a cloud of vortex filaments. Based on these facts, it would be safe to conjecture that the breakdown of the vortex ring into a turbulent cloud is what has been observed. This is in contrast to the $Re_{\text{tr}} = 1000$ case where Figure 3.48 which shows that the PVR still has a fundamentally ring-like structure.

Symmetry Conditions

To examine the effect that the use of symmetry boundary conditions to simulate part of the vortex ring has on its overall dynamics, the $Re_{\text{tr}} = 1000$ case was simulated two more times, once on a half domain and once on a quarter domain using the same resolution on the finest AMR level as in the full domain case. Figures 3.64 through 3.66 show a time series comparing each of these simulations at roughly the same output times. In Figure 3.64 we see the initial production of the SVR. The vorticity at the core of the SVR is marginally stronger in the full domain simulation that in the two simulations that used symmetry conditions. As the SVR migrates over top of the PVR, the wavy instability of the SVR is stronger in the full domain case than in either the half of quarter domain simulations. This is seen in Figure 3.65. Finally, Figure 3.66 shows that the SVR is pulled

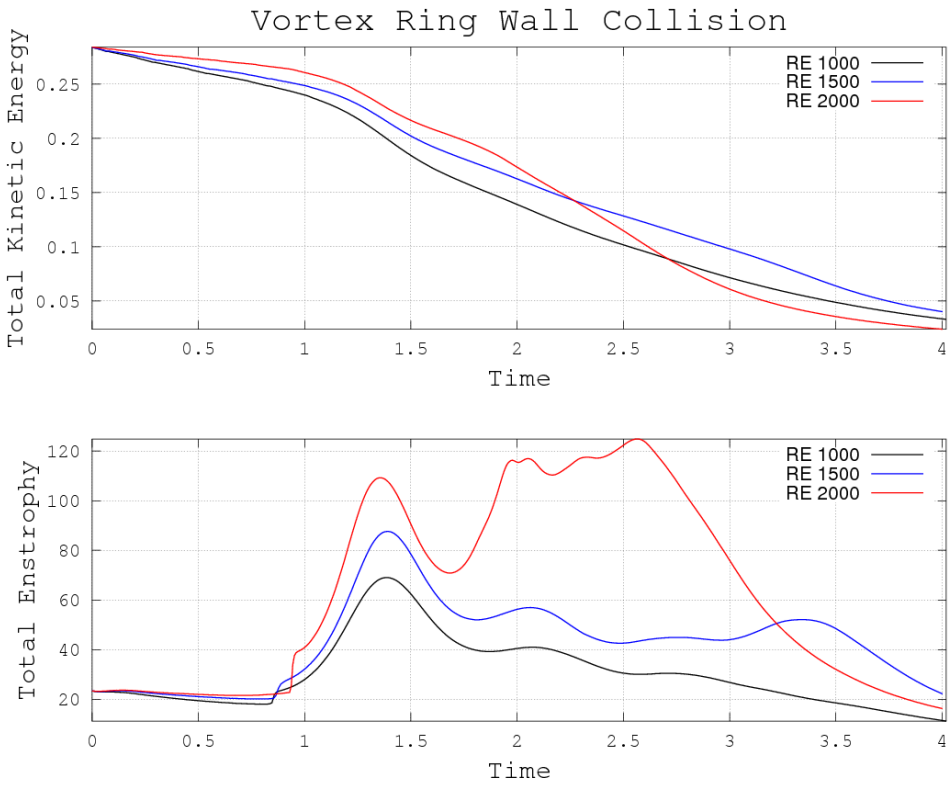

Figure 3.63: The total kinetic energy and total enstrophy curves for the vortex ring wall interactions at translational Reynolds numbers 1000, 1500, and 2000. While the total kinetic energy of the first two cases decays at roughly the same rate, the curve for the $Re = 2000$ case quickly decreases after time $t = 2$. This corresponds to a period of significant enstrophy production in the flow. Both these events correlate the the development of a network of strong small vortex filaments which eventually lead to the breakdown and diffusion of the vortex rings. It can be conjecture that this is the turbulent breakdown of the vortex ring.

under the PVR in the full domain case, compressing the PVR and causing an intensification of circulation. In the symmetry cases, however, the SVR remains in the centre of the PVR and is relatively weak. These results suggest that low radial or axial modes are responsible for some of the behaviours we have observed in the SVR as the lowest of these modes would be geometrically filtered in the two symmetry cases.

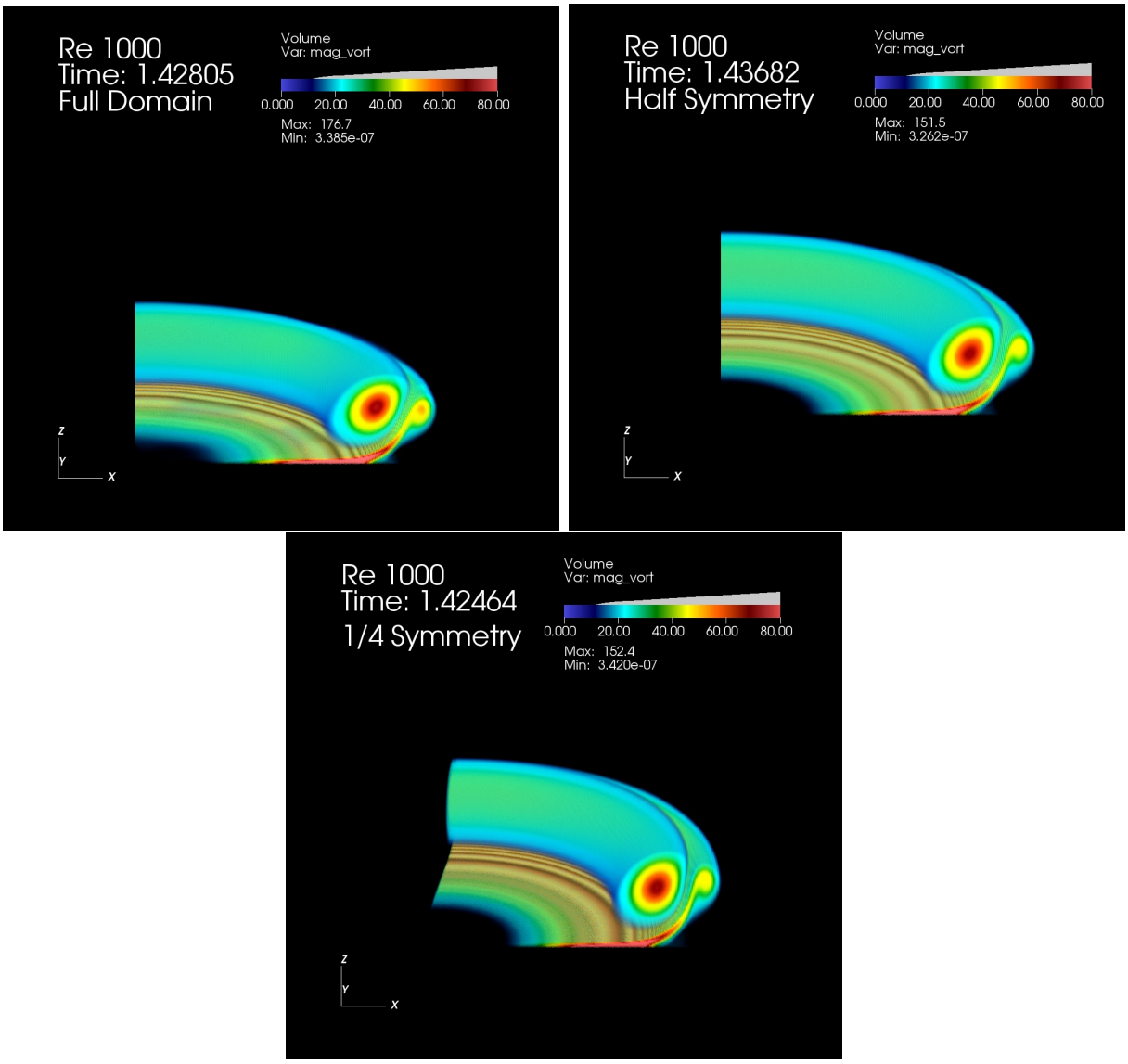

Figure 3.64: A comparison of the secondary vortex ring formation at $Re_{\mathrm{tr}} = 1000$ for simulations using half and quarter domain with symmetry boundary conditions to the full domain simulation. The vorticity at the core of the secondary vortex appears marginally stronger in the full domain case than for the half and quarter domain cases.

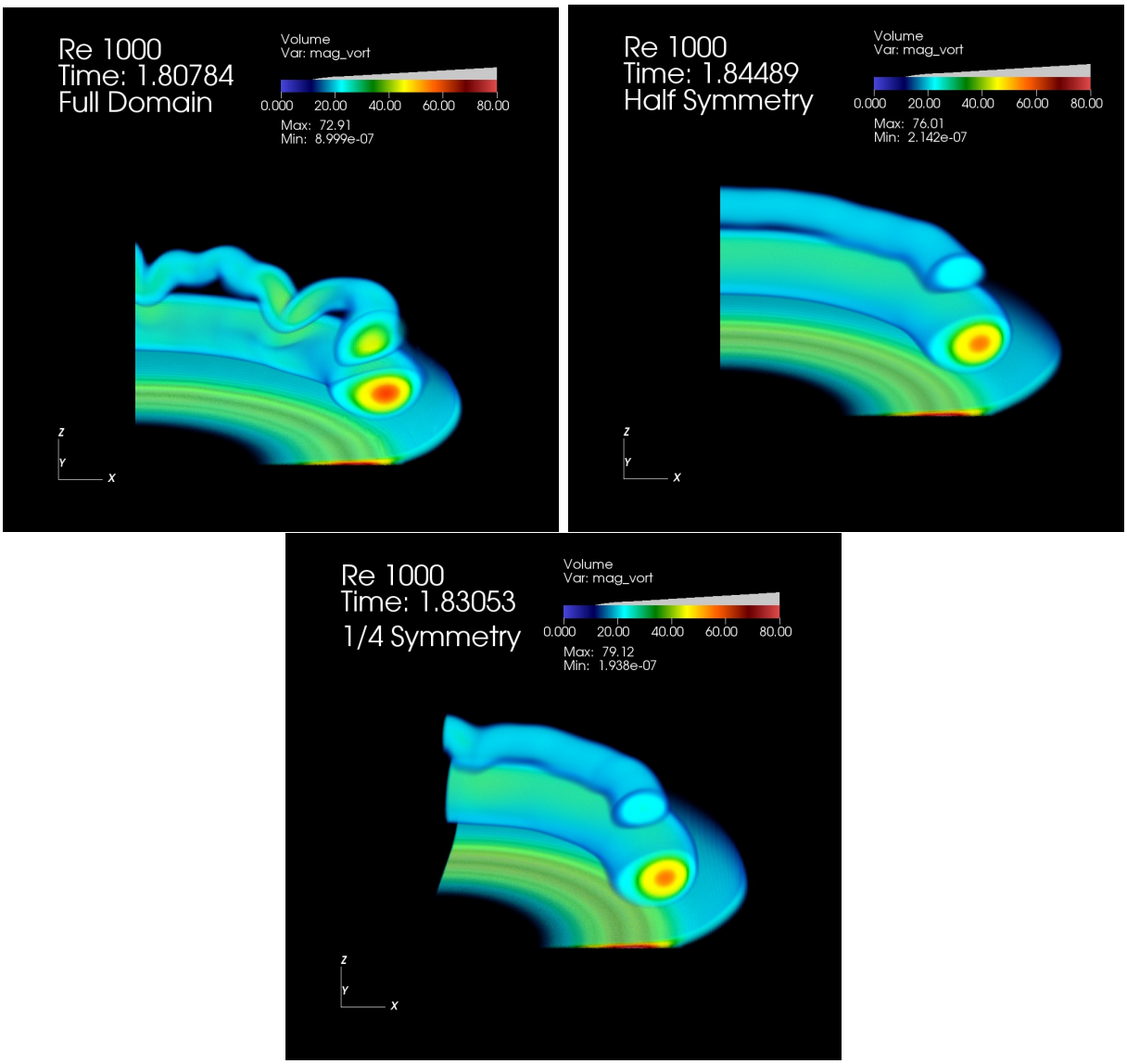

Figure 3.65: As the secondary vortex ring migrates over the top of the primary vortex ring, the wavy instability is much more pronounced in the full domain simulation than in either the quarter or half domain simulations.

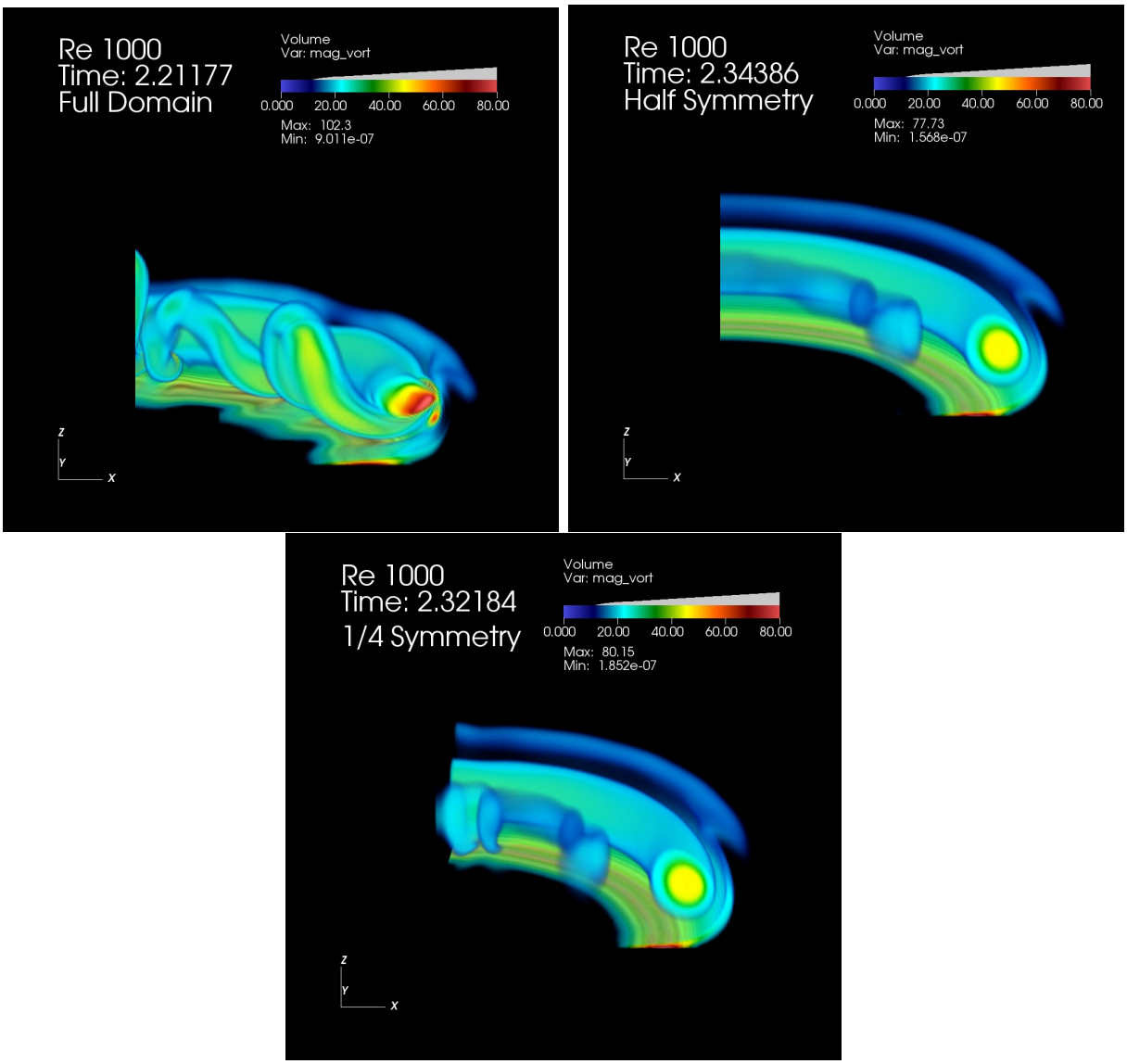

Figure 3.66: While the secondary vortex ring develops the expected loop shaped structure in the full domain simulations, it is weak and remains in the centre of the primary ring in the half and quarter domain simulations.

Bottom Shear Stress

In [35], the resuspension of a sediment bed by a vortex ring was investigated experimentally. A bed of particles was created at the bottom of a tank to simulate sediment, an a vortex ring was generated using a pumping method. The vortex ring travels downwards and interacts with the bottom boundary, undergoing a normal collision. The Reynolds number based on the translational speed of the ring was fixed between experiments, while the diameter of the particles was varied from $90\mu m$ to $1600\mu m$, and the relative density of the particles was varied from 1.2 to 7.

Since the resuspension of the particles is directly tied to the shear stress in the bottom boundary, in [35] the viscous stress in the bottom boundary is found indirectly by measuring the velocity field directly adjacent to the wall. It is found that the highest values of the shear stress occur directly below the core of the PVR, and that the maximum value of shear stress in the bottom boundary occurs when the PVR first interacts with the bottom boundary. Additionally, it is noted that the azimuthal velocities induced by the breakdown of the PVR and SVR are of secondary significance as they only lead to small-scale scarring features around the edge of the crater formed by the initial PVR collision.

While the AMRINS currently does not have the capabilities to model particles advected with a flow, it is possible to calculated the viscous shear stress

$$\tau_z|_{z=0} = \sqrt{\frac{\partial u}{\partial z}^2 + \frac{\partial v}{\partial z}^2} \tag{3.10}$$

in the bottom boundary for the vortex ring simulations carried out above and compare to the results of [35].

Figure 3.67 shows the bottom shear stress for the case when the Reynolds number is 1500. The upper panel corresponds to the instant when the first maximum value of bottom shear stress occurs, which is roughly when the PVR first interacts with the bottom boundary. Additionally, the shape of the bottom shear stress profile mimics that of the PVR, with the maximum value occurring directly beneath its core. The lower panel corresponds to the instant when the second maximum value of bottom shear stress occurs, which is roughly when the SVR begins to stretch and breakdown the PVR. The values of shear stress here are smaller than in the upper panel and isolated to small patches around the edge of the PVR. These small patches of shear stress would lead to the scarring behaviour observed in [35], but would not necessarily suspend a significant amount of sediment depending on the relative density and size of the particles.

Figure 3.68 shows the bottom shear stress for the case when the Reynolds number is 2000. The results here are similar to the previous case, with the first maximum value of the bottom shear stress occurring directly the below the PVR core when the PVR first interacts with the the bottom boundary. The secondary maximum value of bottom shear stress again occurs when the SVR begins to breakdown the PVR, however, the maximum values are more localized and smaller in magnitude when compared to the upper panel.

3.4.3 Discussion

We have seen the results of three sets of computer simulations of a vortex ring interacting with a solid wall at higher Reynolds numbers than have been simulated previously. The Retr $= 1000$ experiments agree very well with those observed in [15], such as the generation and loop structure of the SVR. For Reynolds numbers in the range $1600 < Re_{\text{tr}} < 2500$, it was observed in [15] that the SVR transitions gradually from a loop structure to a kink structure. For $Re_{\text{tr}} > 2500$ the kink structured SVR was ejected away from the wall into the bulk fluid after an interaction with the TVR. While one could conjecture that we have observed the turbulent breakdown of the PVR here based on the rapid stretching events which lead to a cloud of small vortex filaments and a rapid decay of total kinetic energy, the ejection of the SVR was not observed. It is not clear from [15] however that this should be observed at this Reynolds number, which is less than 2500.

The comparison of the simulations at $Re_{\text{tr}} = 1000$ for different partial domains using symmetry conditions suggest that some caution must be taken when using these kinds of approximations. Hence, the higher Reynolds number simulations should be recalculated on a full computational domain. Additionally, for a better comparison with experiments, the trajectory of the vortex core should be tracked for each Reynolds number as was done in [15] and other studies.

Lastly, the shear stress induced in the bottom boundary by the vortex ring was calculated for the simulations conducted herein. These profiles are qualitatively similar to those observed in [35] as the largest values of the bottom shear stress occur directly below the core of the PVR when it first collides with the bottom boundary. The shear stresses induced by the breakdown of the primary vortex ring by the secondary vortex ring are found to be smaller and magnitude and isolated to small patches around the edge of the primary ring. These small isolated patches would lead to the scarring patterns observed in [35]. Despite some open questions, AMR has proven to be a strong tool here again, allowing new regimes to be accessed in the simulation of a vortex ring interacting with a wall using minimal computational resources.

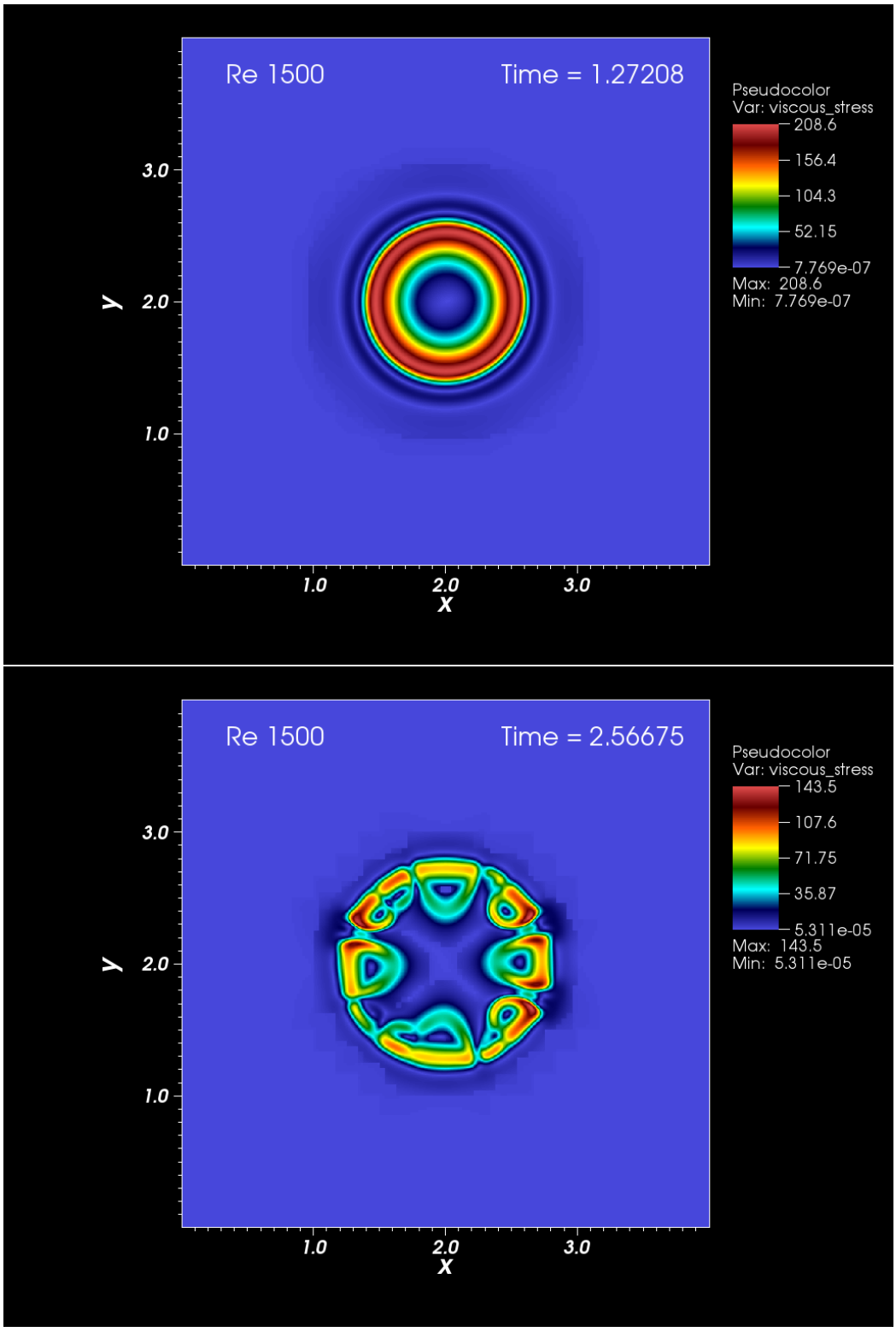

Figure 3.67: The shear stress induced by the vortex ring in the bottom boundary for Reynolds number 1500. The upper and lower panels show the instants at which the first and second maximum values of bottom shear stress occur respectively. Note: The colour axes are different between the upper and lower panels.

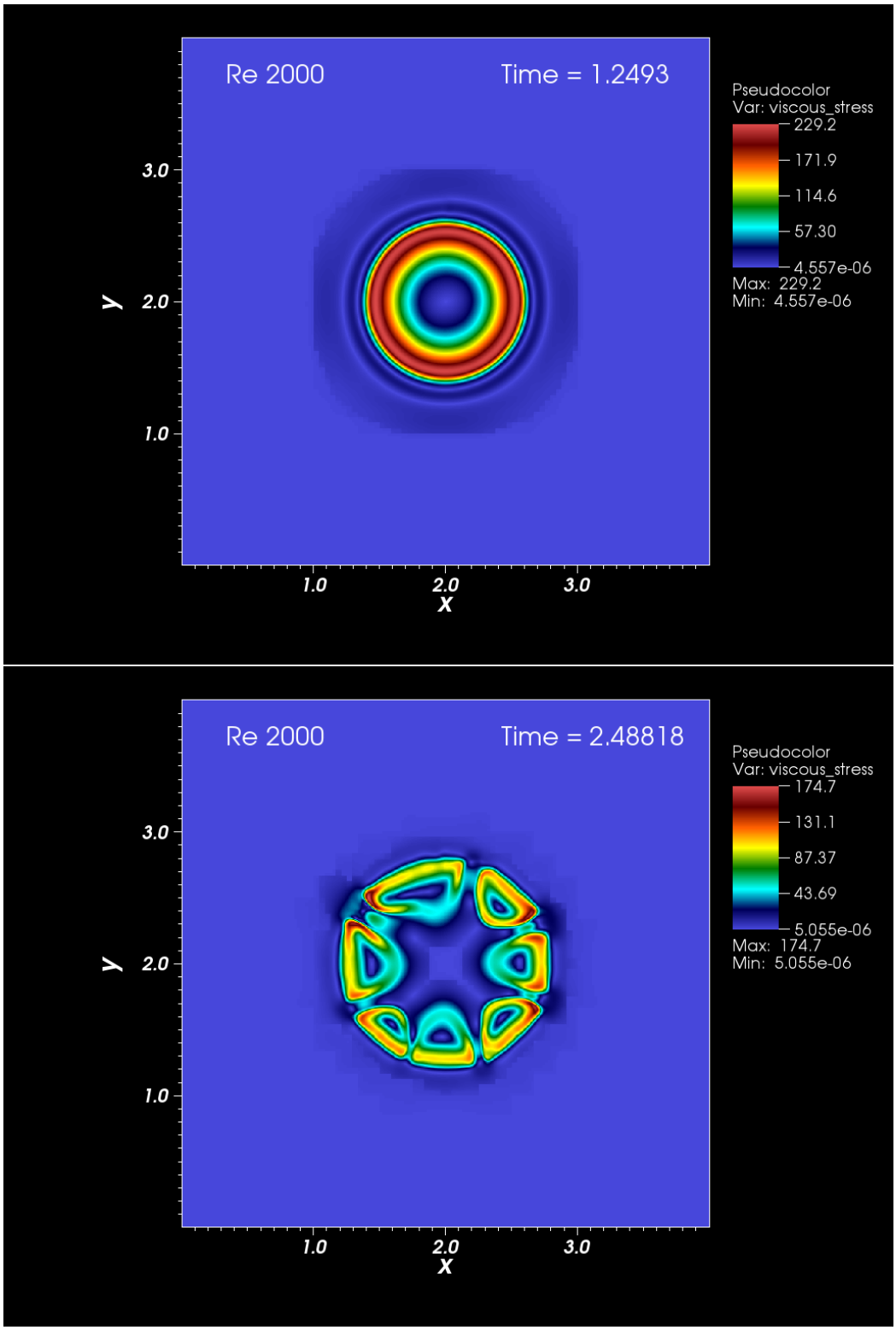

Figure 3.68: The shear stress induced by the vortex ring in the bottom boundary for Reynolds number 2000. The upper and lower panels show the instants at which the first and second maximum values of bottom shear stress occur respectively. Note: The colour axes are different between the upper and lower panels.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

A finite volume based model using adaptive mesh refinement has been used to simulate vortex-wall interactions in both two and three dimensions. Scale testing of this code revealed that it achieved good scalability for very high resolution simulations. These tests also revealed that AMR offered a significant savings in CPU time over an equivalent single grid simulation. This allows for simulations to be performed using a smaller number of cores and has important implications for research groups with limited computational resources.

The interaction of a two dimensional vortex dipole with a solid wall was also simulated. The results of the AMR model were compared to benchmark results taken from the literature and matched up very well. Additionally, the flow was simulated for a much higher Reynolds number than had been done previously, revealing the formation and ejection of a cloud of strong, compact vortices from the boundary layer.

Lastly the interaction of a three dimensional vortex ring with a solid wall was simulated. In laboratory experiments it is observed that a complex series of events takes place generating a secondary and tertiary vortex ring. Using the AMR code we were able to correctly obtain the loop structure of the secondary vortex ring at a moderate Reynolds number. In the final simulation, the Reynolds number was pushed further than has been simulated previously using a fully three dimensional model and it can be conjectured to be the turbulent breakdown of the vortex ring that was simulated.

4.2 Future Work

AMR techniques are becoming increasingly popular in a wide variety of applications. While the results of the simulations conducted have been promising, the diligent scientist is always looking ahead to future problems which may be studied. We now give a short list of possible extensions to the work that has been presented.

4.2.1 Vortex Wall Interactions

Vortex Ring

There are still a few outstanding questions regarding the vortex ring wall interactions presented in the last chapter. First, the higher Reynolds number simulations should be conducted using the full computational domain without symmetry boundary conditions. This would require a substantial amount of computing power and such simulation could possibly benefit from using a deeper AMR hierarch, i.e. 3 or 4 levels of refinement. Additionally the trajectories of the primary, secondary, and tertiary vortex ring cores should be calculated. If simulations could be performed for Reynolds numbers greater than 2500 then it should be possible to observe the ejection of the secondary vortex ring into the bulk fluid.

Three Dimensional Vortex Pairs

The interaction of a vortex pair with a solid boundary in three dimensions has been studied both computationally and in laboratory and wind tunnel experiments by a number of others. A review paper was published in January of this year related to the topic [30]. This problem is of particular interest to the aerospace industry due to the wingtip vortices generated by passenger and cargo planes. As seen in the chapter on vortex dipole wall interactions, it is very computationally intensive to simulated these types of flows at high Reynolds numbers. This fact is compounded further in three dimensions. So far computational studies have been done for low to moderate Reynolds numbers, however few if any DNS studies have been done for high Reynolds number flows. An AMR based model appears to be the perfect candidate in this situation as the vortex pair only occupy a small portion of the computational domain.

4.2.2 Variable Density Flows

AMR has been successfully used for variable density flows in [5, 6, 40] among others. Due to the highly templated nature of the Chombo library, it should be relatively straightforward to change the classes which handle the projection operations in the AMRINS code to perform a variable density projection. A number of interesting variable density flows such as those involving gravity currents or shear instabilities of internal solitary waves involve a number of different length and time scales. Such flows would benefit from the use of AMR as the phenomena of interest require significant resolution but only occupy a small subset of the entire domain.

4.2.3 Free Surface Flows

AMR has been used to simulate free surface flows in [46]. Level Set and Volume of Fluid techniques naturally fit within the finite volume frame work and allow for accurate interface tracking as well as the inclusion of surface tension. A significantly challenging problem involving free surface flows is that of modelling breaking surface waves. In recent years it has been discovered that microscale breaking wave occupy a large portion of the surface of many lakes a low to moderate wind speeds [25]. These waves break without the entrainment of air and contain features, such as parasitic capillary waves, which are highly dependent on surface tension. Such waves have been shown to greatly enhance the flux of heat and dissolved gases across the air-water interface of lake whenever they are present [57].

4.2.4 Complex Geometries

A number of AMR applications have used embedded boundary methods to simulate problems on arbitrary geometries in three dimensions [38, 5, 53]. This coupled with the ability to handle variable density flows would provided a very useful computational tool for studying a number of difficult problems of interest to the lake and ocean modelling communities, such as the shoaling and breaking of internal waves. Problems of this type involved a number of length scales and would require significant resolution to capture instabilities and mixing which occur. At the present moment the AMRINS code can only simulate rectangular domains. Another embedded boundary Navier-Stokes solver which comes with the standard Chombo library also assume constant density and is a non-subcycled algorithm. Since subcycling provides a significant cost savings for deep AMR hierarchies this is a disadvantage. Given all of this a code using embedded boundaries which could simulate

variable density flows would likely need to be designed from the ground up and would be a significant undertaking.

4.2.5 Higher Order Finite Volume Methods

Fourth order routines for the Chombo library have recently been developed [34, 58]. These new parts of the library and slated to be included in the next publicly available release of Chombo sometime later this year. This would allow for the extension of the AMRINS code to a completely fourth order scheme.

References

[1] M. Adams, P. Colella, D. T. Graves, J. N . Johnson, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo software package for amr applications - design document. Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory.

[2] Ronald J Adrian. Hairpin vortex organization in wall turbulence. *Physics of Fluids*, 19(4):041301, 2007.

[3] Ann S Almgren, John B Bell, and William Y Crutchfield. Approximate projection methods: Part i. inviscid analysis. *SIAM Journal on Scientific Computing*, 22(4):1139–1159, 2000.

[4] Ann S Almgren, John B Bell, and William G Szymczak. A numerical method for the incompressible navier-stokes equations based on an approximate projection. *SIAM Journal on Scientific Computing*, 17(2):358–369, 1996.

[5] Michael F Barad, Phillip Colella, and S Geoffrey Schladow. An adaptive cut-cell method for environmental fluid mechanics. *International journal for numerical methods in fluids*, 60(5):473–514, 2009.

[6] Michael F Barad and Oliver B Fringer. Simulations of shear instabilities in interfacial gravity waves. *Journal of Fluid Mechanics*, 644:61–95, 2010.

[7] John B Bell, Phillip Colella, and Harland M Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85(2):257–283, 1989.

[8] John B Bell and Daniel L Marcus. A second-order projection method for variable-density flows. *Journal of Computational Physics*, 101(2):334–348, 1992.

[9] Marsha Berger and Isidore Rigoutsos. An algorithm for point clustering and grid generation. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(5):1278–1286, 1991.

[10] Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[11] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

[12] George Bluman and Sukeyuki Kumei. *Symmetries and differential equations*, volume 154. Springer Science & Business Media, 2013.

[13] C Breitsamter. Wake vortex characteristics of transport aircraft. *Progress in Aerospace Sciences*, 47(2):89–134, 2011.

[14] William L Briggs, Steve F McCormick, et al. *A multigrid tutorial*. Siam, 2000.

[15] AW Cerra Jr and Charles R Smith. Experimental observations of vortex ring interaction with the fluid adjacent to a surface. Technical report, DTIC Document, 1983.

[16] Ming Cheng, Jing Lou, and Li-Shi Luo. Numerical study of a vortex ring impacting a flat wall. *Journal of Fluid Mechanics*, 660:430–455, 2010.

[17] Alexandre Joel Chorin. On the convergence of discrete approximations to the navier-stokes equations. *Mathematics of Computation*, 23(106):341–353, 1969.

[18] Herman JH Clercx and C-H Bruneau. The normal and oblique collision of a dipole with a no-slip boundary. *Computers & fluids*, 35(3):245–279, 2006.

[19] Phillip Colella. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics*, 87(1):171–200, 1990.

[20] Phillip Colella, Daniel T Graves, Benjamin J Keen, and David Modiano. A cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics*, 211(1):347–366, 2006.

[21] Matthieu Duponcheel. *Direct and large-eddy simulation of turbulent wall-bounded flows: further development of a parallel solver, improvement of multiscale subgrid models and investigation of vortex pairs in ground effect*. PhD thesis, Ecole Polytechnique de Louvain, 2009.

[22] Francesco Gargano, Marco Sammartino, and Vincenzo Sciacca. High reynolds number navier–stokes solutions and boundary layer separation induced by a rectilinear vortex. *Computers & fluids*, 52:73–91, 2011.

[23] Seyed Amin Ghaffari and Kai Schneider. Development of an adaptive multi-resolution method to study the near wall behavior of two-dimensional vortical flows. Technical report, 2011.

[24] DM Harris and CHK Williamson. Instability of secondary vortices generated by a vortex pair in ground effect. *Journal of Fluid Mechanics*, 700:148–186, 2012.

[25] AT Jessup, CJ Zappa, and Harry Yeh. Defining and quantifying microscale wave breaking with infrared imagery. *Journal of Geophysical Research: Oceans*, 102(C10):23145–23153, 1997.

[26] Geert H Keetels, Umberto D'Ortona, Werner Kramer, HJH Clercx, Kai Schneider, and GJF Van Heijst. Fourier spectral and wavelet solvers for the incompressible navier–stokes equations with volume-penalization: Convergence of a dipole–wall collision. *Journal of Computational Physics*, 227(2):919–945, 2007.

[27] S Kohn. Samrai: Structured adaptive mesh refinement applications infrastructure. Technical report, Technical report, Lawrence Livermore National Laboratory, 1999.

[28] Werner Kramer, HJH Clercx, and GJF van Heijst. Vorticity dynamics of a dipole colliding with a no-slip wall. *Physics of Fluids (1994-present)*, 19(12):126603, 2007.

[29] P Kundu, I Cohen, and D Dowling. *Fluid Mechanics, 5th edition*. Academic Press, 2012.

[30] Thomas Leweke, Stéphane Le Dizès, and Charles HK Williamson. Dynamics and instabilities of vortex pairs. *Annual Review of Fluid Mechanics*, 48:507–541, 2016.

[31] Daniel F Martin and Phillip Colella. A cell-centered adaptive projection method for the incompressible euler equations. *Journal of computational Physics*, 163(2):271–312, 2000.

[32] Daniel F Martin, Phillip Colella, and Daniel Graves. A cell-centered adaptive projection method for the incompressible navier–stokes equations in three dimensions. *Journal of Computational Physics*, 227(3):1863–1886, 2008.

[33] DF Martin and Kelley L Cartwright. *Solving Poisson's equation using adaptive mesh refinement*. Electronics Research Laboratory, College of Engineering, University of California, 1996.

[34] Peter McCorquodale and Phillip Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.

[35] RJ Munro, N Bethke, and SB Dalziel. Sediment resuspension and erosion by vortex rings. *Physics of Fluids (1994-present)*, 21(4):046601, 2009.

[36] Paolo Orlandi and Roberto Verzicco. Vortex rings impinging on walls: axisymmetric and three-dimensional simulations. *Journal of Fluid Mechanics*, 256:615–646, 1993.

[37] RT Pierrehumbert. A family of steady, translating vortex pairs with distributed vorticity. *Journal of Fluid Mechanics*, 99(01):129–144, 1980.

[38] Stéphane Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600, 2003.

[39] Philip G Saffman. *Vortex dynamics*. Cambridge university press, 1992.

[40] Edward Santilli and Alberto Scotti. The stratified ocean model with adaptive refinement (somar). *Journal of Computational Physics*, 291:60–81, 2015.

[41] L Ridgway Scott, Terry Clark, and Babak Bagheri. *Scientific parallel computing*, volume 146. Princeton University Press Princeton, 2005.

[42] Philippe R Spalart. Airplane trailing vortices. *Annual Review of Fluid Mechanics*, 30(1):107–138, 1998.

[43] PR Spalart, M Kh Strelets, AK Travin, and ML Shur. Modeling the interaction of a vortex pair with the ground. *Fluid Dynamics*, 36(6):899–908, 2001.

[44] JM Straka, Robert B Wilhelmson, Louis J Wicker, John R Anderson, and Kelvin K Droegemeier. Numerical solutions of a non-linear density current: A benchmark solution and comparisons. *International Journal for Numerical Methods in Fluids*, 17(1):1–22, 1993.

[45] Christopher J Subich, Kevin G Lamb, and Marek Stastna. Simulation of the navier–stokes equations in three dimensions with a spectral collocation method. *International Journal for Numerical Methods in Fluids*, 73(2):103–129, 2013.

[46] Mark Sussman, Ann S Almgren, John B Bell, Phillip Colella, Louis H Howell, and Michael L Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148(1):81–124, 1999.

[47] JD Swearingen, JD Crouch, and RA Handler. Dynamics and stability of a vortex ring impacting a solid boundary. *Journal of Fluid Mechanics*, 297:1–28, 1995.

[48] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2009.

[49] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Academic press, 2000.

[50] L Türk, D Coors, and D Jacob. Behavior of wake vortices near the ground over a large range of reynolds numbers. *Aerospace science and technology*, 3(2):71–81, 1999.

[51] Edward H Twizell, AB Gumel, and MA Arigu. Second-order, l 0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathematics*, 6(1):333–352, 1996.

[52] Brian Van Straalen, Phil Colella, Daniel Graves, and Noel Keen. Petascale block-structured amr applications without distributed meta-data. *Euro-Par 2011 Parallel Processing*, pages 377–386, 2011.

[53] Marcos Vanella, Patrick Rabenold, and Elias Balaras. A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid–structure interaction problems. *Journal of Computational Physics*, 229(18):6427–6449, 2010.

[54] R Verzicco and P Orlandi. Normal and oblique collisions of a vortex ring with a wall. *Meccanica*, 29(4):383–391, 1994.

[55] R Verzicco and P Orlandi. Wall/vortex-ring interactions. *Applied Mechanics Reviews*, 49(10):447–461, 1996.

[56] JDA Walker, CR Smith, AW Cerra, and TL Doligalski. The impact of a vortex ring on a wall. *Journal of Fluid Mechanics*, 181(1):99–140, 1987.

[57] CJ Zappa, WE Asher, and AT Jessup. Microscale wave breaking and air-water gas transfer. *Journal of Geophysical Research: Oceans*, 106(C5):9385–9391, 2001.

[58] Qinghai Zhang, Hans Johansen, and Phillip Colella. A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation. *SIAM Journal on Scientific Computing*, 34(2):B179–B201, 2012.