2.3 Vorticity discretization into elements

Though an infinitely-thin vortex sheet is only accurately modeled by a mesh of at least C^1-continuous[2] surface elements, the decision to regularize the vorticity allows discretization to take on a myriad of forms. Without regularization, infinitely many elements would be required to properly discretize a vortex sheet, but with regularization, elements can smear their vorticity over a non-singular volume, covering any gaps in their distribution. Each of the different forms of computational element brings with it advantages and disadvantages, sometimes dependent on other components of the vortex method such as diffusion, remeshing, and the velocity calculation. This section will introduce the common element types categorized by dimension, then summarize their common behaviors, and finally discuss each of them in turn.

A vortex method in D dimensions ($D = \{2, 3\}$) can represent vorticity on any combination of elements with *geometric* dimension $\{0 \ldots D\}$. Axisymmetric vortex methods count as $D = 2$ for the purposes of classifying the element type. Because a regularized vortex sheet has non-zero thickness, the vorticity *support* of any elements chosen to represent it will have non-zero volume, though. Visual examples of each of the elements described below appear in figure 2.3.

The only zero dimensional element is a point, which in regularized vortex methods represents the center of a particle of non-zero radius, often called a "vortex blob" or "vorton." In planar two dimensional vortex methods, these represent parallel and infinite vortex filaments, and in axisymmetric methods they are co-axial vortex rings. This is the most common discretization type for vortex methods, due to its simplicity and generality.

The space curve is the general one dimensional geometric entity, and represents the centerline of a vortex filament in a three dimensional vortex method. It is discretized by material marker points connected with straight or higher-order segments. In two dimensional vortex methods, this entity is instead a plane curve, and represents a vortex sheet. Note that in this case the direction of the vorticity is perpendicular to the curve, whereas a filament has its vorticity parallel to the curve.

A similar distinction between two and three dimensional vortex methods exists for

[2] A C^1-continuous surface is one in which the first derivative exists and is continuous over all elements.

the two dimensional element, which is commonly discretized by a connected grid of triangles or quadrilaterals. In two dimensional vortex methods, each element represents a volume, and as such, does not rely on regularization to have a non-zero volume. In three dimensional methods, however, these elements define the location of vortex sheets whose vorticity is always tangent to the sheet elements.

A three dimensional element can only exist in a three dimensional vortex method, and in the uncommon case of its occurrence it is represented by a connected mesh of tetrahedra and triangular or quadrilateral prisms. Again, like triangles in two dimensional methods, each of these elements has a volume already and does not require regularization. These "vortex volume methods" are the least common of the above discretization types, despite many advantages.

Several important differences between the above discretizations are: treatment of the vortex stretching term, ability to remesh and the form remeshing takes, diffusion of vorticity, and element connectivity. These differences are highly interrelated, but it seems best to organize the rest of this section first by element type and then by characteristic.

2.3.1 Particle elements

In the simplest case, vorticity can be accommodated on a collection of discrete particles, each with a location and strength. The strength of each computational point can be assigned in one of three ways: Hald [1979] assigns the value of the vorticity contained in the surrounding blob, Beale and Majda [1982b] assign the value of the vorticity at the point times the volume of the blob, and other schemes assign a new value at every time step [Beale, 1988; Strain, 1996; Marshall and Grant, 1996]. Taking the view of Beale and Majda, particle strength can be stated

$$\boldsymbol{\omega}(\boldsymbol{x}, t) = \sum_p \boldsymbol{\alpha}_p \, \delta(\boldsymbol{x} - \boldsymbol{x}_p) \tag{2.8}$$

or

$$\boldsymbol{\alpha}_p = v_p \boldsymbol{\omega}_p, \tag{2.9}$$

where v_p is the volume and $\boldsymbol{\alpha}_p$ the strength of particle p.

The vorticity update equation (1.6)

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{\omega} = \boldsymbol{\omega} \cdot \nabla \boldsymbol{u} + \frac{1}{\rho^2} \nabla \rho \times \nabla p + \nu \nabla^2 \boldsymbol{\omega} \tag{2.10}$$

must be modified to represent the evolution of the strength of the particles

$$\frac{\partial \boldsymbol{\alpha}_p}{\partial t} = \boldsymbol{\alpha}_p \cdot \nabla \boldsymbol{u} + \frac{v_p}{\rho^2} (\nabla \rho)_p \times \nabla p + \nu \nabla^2 \boldsymbol{\alpha}_p. \tag{2.11}$$

The particle trajectories $\boldsymbol{x}_p$ follow the velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$, which is defined by solutions to the Biot-Savart law modified to accommodate the discretization. As such, the integral over the vorticity support in the original Biot-Savart law (1.14) becomes a summation over all of the particles, or

$$\boldsymbol{u}(\boldsymbol{x}, t) = \frac{1}{4\pi} \sum \frac{\boldsymbol{\alpha}_p(\boldsymbol{x}', t) \times (\boldsymbol{x} - \boldsymbol{x}')}{v_p \left| \boldsymbol{x} - \boldsymbol{x}' \right|^3} \, d\boldsymbol{x}' + \nabla \phi. \tag{2.12}$$

This problem can be solved in a variety of manners, see §2.2 for details.

History

The first dynamic vortex methods used particles, and were all two dimensional simulations of a sinusoidally-perturbed vortex sheet [Rosenhead, 1931; Birkhoff and Fisher, 1959; Abernathy and Kronauer, 1962; Chorin and Bernard, 1973; Kuwahara and Takami, 1973; Christiansen, 1973]. Beale and Majda [1982a] first proposed using spherical particles for three dimensional computations, using Lagrangian update to compute stretch, and forgoing all element connectivity information. All three dimensional vortex methods until then had maintained connectivity were thus technically filament methods. It took several more years for the first implementations of the disconnected particle vortex method [Beale, 1988; Winckelmans and Leonard, 1989] in three dimensions to appear. Since then, it has become the predominant form of modern computational vortex methods.

Connectivity

Particle methods are intrinsically different from filament or sheet methods in that there is no logical connectivity between particles—a particle knows nothing of its neighbors unless explicitly determined (usually by costly searches). Schemes such as diffusion require these local search operations in order to diffuse vorticity properly. The lack of connectivity information storage in particle methods can make remeshing more difficult, but it also simplifies the computer implementation.

Remeshing

The need for element remeshing in vortex methods comes as a result of the significant strain experienced in vortex flows and because convergence proofs require element overlap. The first convergence proofs of vortex particle methods were by Hald and Prete [1978] and Hald [1979]. In later work, proof was provided for the existence of a solution for short times, as long as the overlap between the vortex blobs remained [Beale and Majda, 1982a,b]. Beale [1986] gives a convergence proof of the disconnected, discrete vortex method by requiring that the vortex core radius be larger than the interparticle spacing.

Remeshing in a general particle method can be accomplished in one of two ways. In the pure Lagrangian sense, gaps in the particle distribution can be detected and filled as they appear (these gaps are easier to detect when elements store connectivity information, such as occurs in filament and sheet methods). This is a "local remeshing" method and is used in the vorticity redistribution method (VRM) [Shankar and van Dommelen, 1996]. More often, vortex particle methods undergo "global remeshing" or "regridding," whereby all particles are replaced with a new regular distribution of particles. This has been observed to provide hyperviscosity dissipation [Cottet and Koumoutsakos, 1999], and as such, imposes a limit on the frequency of the operation. Regridding, like VIC (see §2.2.3), is another instance of combining the benefits of Eulerian and Lagrangian methods.

Vortex stretching

The first particle vortex methods were strictly two dimensional, and thus did not have to account for vortex stretching. The first three dimensional vortex methods [Leonard, 1980a; Chorin, 1980] all used filament elements, so they also did not have to explicitly account for vortex stretching (see §2.3.2).

In the basic three-dimensional vortex particle method, the vortex stretching term is accounted for by calculating the velocity gradient tensor $\nabla \boldsymbol{u}$ at the location of the particle. This has been shown to create vorticity fields that are not solenoidal. Alternative methods have been proposed to rectify this problem. A summary of these methods is provided in [Winckelmans and Leonard, 1989] and [Cottet and Koumoutsakos, 1999]. An alternative solution (in the vein of hybrid methods) proposed by Marshall and Grant [1996] is to use an overlaid Eulerian grid to compute stretch. In comparison, vortex filament (§2.3.2) and vortex sheet (§2.3.3) methods automatically account for the effects of vortex stretching and will always generate solenoidal vorticity fields.

Diffusion

Vortex particle methods are advantageous for the more general case of viscous flow, as the viscous term in their evolution equation can be discretized and solved, provided sufficient particles are nearby with which to exchange strength. The advantage lies in the ability of vortex particle methods to easily create new nearby particles, whether it be via local or global remeshing.

Most viscous vortex methods account for the change in vorticity due to viscosity in a separate step from convection; this is called "viscous splitting" [Chorin, 1973] and is only first-order accurate overall. Higher orders can be achieved by adding more substeps [Beale and Majda, 1981]. This technique is common throughout vortex methods, not only for vortex particle methods. All diffusion methods listed below are typically executed in the viscous substep of a regular time step.

Of the methods devised for accounting for diffusion in vortex methods, core-spreading is likely the earliest. Core-spreading [Kuwahara and Takami, 1973; Leonard, 1980a] is a method by which the radius of a vortex blob (or vortex filament) changes in time ow-

ing to the effects of viscosity. The core-spreading technique derives from the solution to the two-dimensional Navier-Stokes equations in vorticity coordinates for the initial conditions

$$\omega(r, 0) = \Gamma_0 \, \delta(x) \, \delta(y). \tag{2.13}$$

In this case, the Navier-Stokes equations are simply

$$\frac{\partial \omega}{\partial t} = \nu \, \nabla^2 \omega. \tag{2.14}$$

The exact solution for the evolution of the total circulation is

$$\Gamma = \Gamma_0 \, (1 - e^{-r^2/4\nu t}) \tag{2.15}$$

Thus, one can approximate the decay of a two dimensional vortex particle or a three dimensional vortex filament by reducing the effective circulation or increasing the effective radius of the element. Core-spreading has been shown to approximate the wrong equations [Greengard, 1985], but can be corrected by instantaneous reconfiguration of large vortex blobs to thinner ones [Rossi, 1996].

The random vortex method (RVM) was introduced by Chorin [1973] to study slightly viscous flows. The RVM uses a Wiener process to perturb the motion—but not the strength—of each vortex particle, which simulates diffusion of vorticity. Leonard [1980a] references work that shows that to achieve accurate results of viscous diffusion, the RVM requires a large number of particles compared to the Reynolds number. The method was proven to converge to the heat equation by Hald [1986] and by Puckett [1989]. The RVM suffers from low-order non-uniform convergence due to its stochastic character, and is not well-suited for three dimensional methods because it is not pointwise accurate.

The two most popular viscous algorithms for vortex particle methods are based on modifying the particles' strengths instead of their locations. Both of these methods originate by discretizing the diffusion operator by an integral operator over nearby particles, and both work in two or three dimensions. The particle strength exchange (PSE) method [Mas-Gallic, 1987; Degond and Mas-Gallic, 1989a] writes a correction term for the vorticity on each particle, and can handle variable-sized particles [Cottet *et al.*, 1999].

PSE has been extended recently to be able to find general derivatives in particle methods [Eldredge *et al*., 2002], including one-sided derivatives for use in hyperbolic problems. Shankar and van Dommelen [1996] introduced the vorticity redistribution method (VRM) which can conserve moments to arbitrarily-high orders of accuracy and can fill holes in the particle distribution. These are the most common viscosity methods used for recent highly-resolved flow simulations [Ploumhans *et al*., 1999; Cottet and Poncet, 2002; Lakkis and Ghoniem, 1999; Gharakhani, 2003].

A combined Eulerian-Lagrangian method for viscous diffusion is given in [Najm, 1993] in which diffusion is computed in a deterministic manner on a temporary overlaid Eulerian grid, though Cottet and Koumoutsakos [1999] claim this scheme is not conservative.

Summary

In summary, particles are the most commonly-used elements in vortex methods primarily because of the overall implementation simplicity, but also because of their ability to represent arbitrary vortex structures and because of the wealth of existing research into particle methods.

2.3.2 Filament elements

Kelvin's result for the constancy of circulation along a filament allowed vortex filament methods to be the first tractable three-dimensional vortex methods. They enjoyed popularity throughout the 1980s, but a lack of general applicability and the publication of new convergence results around 1990 [Beale, 1986; Hou and Lowengrub, 1990; Cottet *et al*., 1991; Hou *et al*., 1991] shifted the emphasis toward vortex particle methods.

Filaments are usually discretized into straight or smoothed segments connected by Lagrangian nodes, and have a scalar-valued circulation Γ and sometimes a physical or regularization radius. Circulation can be defined by a closed line integral or by an integral over the surface of which that line is a boundary:

$$\Gamma = \int_L \boldsymbol{u} \cdot d\boldsymbol{r} = \int_S \boldsymbol{\omega} \cdot \boldsymbol{n}\, dS. \tag{2.16}$$

A vortex line in the Euler limit is a singular distribution of vorticity that is the result of shrinking the above surface S to a point. The circulation can thus be related to vorticity by

$$\boldsymbol{\omega}(\boldsymbol{x},t) = \Gamma\, \boldsymbol{s}(\boldsymbol{x},t)\, \delta(r), \tag{2.17}$$

where $\boldsymbol{s}$ is the unit tangent vector along the vortex line and $\delta(r)$ is the delta function of the distance in the perpendicular plane from that line. This allows the Biot-Savart law to be rewritten as

$$\boldsymbol{u}(\boldsymbol{x},t) = \frac{\Gamma}{4\pi} \int \frac{\boldsymbol{s} \times (\boldsymbol{x} - \boldsymbol{x}(s))}{|\boldsymbol{x} - \boldsymbol{x}(s)|^3}\, ds + \nabla\phi. \tag{2.18}$$

Leonard [1985] contains a good summary of the variations of the Biot-Savart law for vortex filaments with finite-sized cores and alternate core structures.

Vortex lines can be approximated by a series of nodes connected by segments. For singular distributions of vorticity, these lines must be C^2-smooth for the principal value of the Biot-Savart integral to have a solution. In a regularized method, though, the lines can be of varying degrees of smoothness, provided that the segment lengths are on the order of the regularization scale.

Following the derivation in [Cottet and Koumoutsakos, 1999, §1.3], the evolution equation for circulation is formed by combining equation (2.16) with the Lagrangian form for the acceleration of material particles, or

$$\begin{aligned} \frac{D\Gamma}{Dt} &= \frac{D}{Dt} \int_L \boldsymbol{u} \cdot d\boldsymbol{r} \\ &= \int_L \frac{D\boldsymbol{u}}{Dt} \cdot d\boldsymbol{r} + \int_L \frac{Dd\boldsymbol{r}}{Dt} \cdot d\boldsymbol{u}. \end{aligned} \tag{2.19}$$

Because the vortex lines are material lines (see §1.4.1), the second term simplifies to

$$\int_L \frac{Dd\boldsymbol{r}}{Dt} \cdot d\boldsymbol{u} = \int_L \boldsymbol{u} \cdot d\boldsymbol{u} = 0. \tag{2.20}$$

The acceleration term in (2.19) can be replaced with the Lagrangian form of the momentum equation (1.32) to make

$$\frac{D\Gamma}{Dt} = -\int_L \frac{\nabla p}{\rho} \cdot d\boldsymbol{r} + \nu \int_L \nabla^2 \boldsymbol{u} \cdot d\boldsymbol{r}. \tag{2.21}$$

Bjerken's theorem allows the baroclinic generation of vorticity from a density interface, and appears as a modification of the above equation

$$\frac{D\Gamma}{Dt} = \nu \int_L \nabla^2 \boldsymbol{u} \cdot d\boldsymbol{r}. + \int_S \frac{1}{\rho^2} \nabla\rho \times \nabla p \cdot \boldsymbol{n}\, dS, \tag{2.22}$$

where $\boldsymbol{n}$ is the unit vector in the direction normal to the density gradient and S is the surface of density discontinuity. Finally, in the absence of density effects and in the Euler limit of vanishing viscosity, Kelvin's theorem for the evolution of circulation on vortex filaments is

$$\frac{D\Gamma}{Dt} = 0. \tag{2.23}$$

Of special note is the absence of any vortex stretching term. This means that constancy of circulation along a filament is enough to automatically satisfy vortex stretching, obviating the need for any extra calculations like those required by vortex particle methods.

History

Leonard [1975] was likely the first to use connected filaments in a three dimensional vortex method. Soon afterward, Rehbach [1978] and Chorin [1980] presented vortex particle methods that used filament-like connectivity information to evaluate stretch, and are among the earliest vortex filament methods. Other early methods [Leonard, 1980a,b; Couët *et al*., 1981; Ashurst, 1981; Leonard, 1985] expanded the range of problems that could be addressed with vortex filaments. Later research [Ashurst, 1983; Ashurst and Meiburg, 1988; Knio and Ghoniem, 1991; Martin and Meiburg, 1996; Lindsay and Krasny, 2001] uses vortex filaments to approximate vortex sheets, a topic that will also be covered in the next section (§2.3.3).

The convergence of the vortex filament method is presented in Greengard [1986] using a centered-difference operator along the filament elements for vortex stretching.

Connectivity

In this section, a vortex filament method shall be defined as one in which neighboring elements maintain one-dimensional logical connectivity. Having and maintaining this

connectivity information greatly affects the capability of the method to account for vortex stretching and to diffuse or remesh vorticity. It also definitively separates vortex filament methods from vortex particle methods, as will be seen in the following paragraphs.

Remeshing

Remeshing involves the remaking or modification of the computational elements into a form that more accurately discretizes the vorticity. In a vortex filament method, remeshing can occur along the filament as a response to longitudinal stretching, transverse to the filament to respond to diffusion of vorticity away from the filament axis, and between filaments as a result of vortex mergers and recoalescence. The latter two methods will be described in the "diffusion" section below.

The need for longitudinal remeshing stems from the observation that vortex filaments encounter ever-increasing growth rates and arc lengths in three dimensional inviscid simulations [Siggia, 1985] (and presumably large but finite growth in slightly-viscous flow). Thus, more elements are required to properly discretize the flow and capture all necessary geometric details. Local remeshing along a moving discretized space curve is a common operation in modern curve-tracking methods, and possibly originated with the method of Berk and Roberts [1967] for increasing the resolution of plane curves. In that method, curve segments that exceeded a predetermined threshold were split into two new segments, with the new node being placed at the geometric center of the old segment. Early vortex three dimensional vortex filament methods used this very same procedure [Chorin, 1981; Ghoniem *et al.*, 1988], with Chorin [1981] claiming that a filament remeshing scheme more elaborate than this is unnecessary. Nevertheless, higher-order remeshing schemes were proposed and implemented in [Couët *et al.*, 1981; Ashurst and Meiburg, 1988; Martin and Meiburg, 1991].

Vortex stretching

One of the critical differences between regularized vortex particle methods and regularized vortex filament methods is the scheme used to evaluate vortex stretching. The extra work required to maintain element connectivity in a filament method pays off by allowing calculation of stretch without the velocity gradient calculation required by vor-

tex particle methods. To see this, imagine two adjacent marker points at locations $\boldsymbol{X}_n$ and $\boldsymbol{X}_{n+1}$ along a C^0-continuous curve (straight segments, the method introduced by Rehbach [1978]). The total vorticity for this segment is the scalar circulation Γ multiplied by the segment vector $d\boldsymbol{l} = \boldsymbol{X}_{n+1} - \boldsymbol{X}_n$. The change in total vorticity, to its leading order, can be given as

$$\begin{aligned}
\frac{d}{dt}(\Gamma\, d\boldsymbol{l}) &= \Gamma\, \frac{d}{dt}(\boldsymbol{X}_{n+1} - \boldsymbol{X}_n) \\
&= \Gamma\, \boldsymbol{u}(\boldsymbol{X}_{n+1}) - \boldsymbol{u}(\boldsymbol{X}_n) \\
&\simeq \Gamma\, \big[(\boldsymbol{X}_{n+1} - \boldsymbol{X}_n) \cdot \nabla\big] \boldsymbol{u}(\boldsymbol{X}_n) \\
&\simeq \Gamma\, (d\boldsymbol{l} \cdot \nabla) \boldsymbol{u}(\boldsymbol{X}_n),
\end{aligned} \tag{2.24}$$

which is the same form as the stretching term in the Helmholtz equation (1.33) and the vortex particle evolution equation (2.11). Thus, vortex filament methods automatically account for the change in vorticity due to stretching.

Diffusion

As mentioned above, schemes for diffusion in vortex filament methods are limited by the form and connectivity of the elements. This has resulted in few applications of vortex filaments to viscous flow simulation.

Because of the similarity between two dimensional vortex particle methods and three dimensional vortex filament methods, the technique of creating nearby overlapping elements to account for diffusion should remain valid (and necessary). In the case of filaments, an appropriate scheme should identify when and where new filaments must be formed, and how much vorticity should be exchanged. Only one known implementation of this style of vortex filament remeshing is known [Lindsay and Krasny, 2001], though it is activated to correct for cross-filament strain and not diffusion. Some methods initialize the vorticity support with a number of overlapping vortex filaments [Knio and Ghoniem, 1990], but the corresponding simulations do not diffuse vorticity among them.

Another form of remeshing exists which illustrates the interdependency between connectivity, remeshing, and diffusion in vortex filament methods; it is commonly called "hairpin removal" or "filament surgery," and represents a renormalization process by

which small-scale detail is removed by a local element redistribution process. Feynman [1957] first proposed topology change for closely-spaced and oppositely-signed vortex filaments in very close proximity in the context of flow in the Euler limit within superfluids. Leonard [1975] suggests that the bookkeeping for such an operation within a vortex method would be troublesome. It is further mentioned in [Leonard, 1980a, 1985], and developed in the work of Chorin [1990, 1993, 1996].

Summary

The natural way that vortex filament methods mimic the real physics of fluid flow assisted their adoption as the first three dimensional vortex methods. They remain useful for certain classes of problems, but lack generality due to the difficulty of efficient remeshing.

2.3.3 Surface mesh elements

Sheet elements represent a relatively uncommon discretization technique, primarily because of the difficulty involved in remeshing the surface elements to account for extensive stretch and folding of the shear layer. Vortex sheets are described by space curves in two dimensional vortex methods, but the emphasis in this section will be on three dimensional methods, for which a vortex sheet is described by a surface. Surface mesh elements can take the form of triangles, quadrilaterals, or any N-sided ($N \geq D$) shape, though practical applications have only used triangles and quadrilaterals. Additionally, surface elements may be locally flat or of higher-order continuity.

The definition of a vortex sheet is

$$\boldsymbol{\omega}(\boldsymbol{x}, t) = \boldsymbol{\gamma}(\boldsymbol{x}, t)\, \delta(n), \tag{2.25}$$

which allows writing the Biot-Savart equation as an integral over the sheet instead of a space integration of a summation of particles. Using this definition, the modified Biot-Savart law becomes

$$\boldsymbol{u}(\boldsymbol{x}, t) = \frac{1}{4\pi} \int \frac{\boldsymbol{\gamma}(\boldsymbol{x}', t) \times (\boldsymbol{x} - \boldsymbol{x}')}{|\boldsymbol{x} - \boldsymbol{x}'|^3}\, dS' + \nabla\phi. \tag{2.26}$$

Following the derivation in [Pozrikidis, 2000], the vorticity evolution equation can be obtained by the following steps. The fluid velocities on each side of the sheet are written in terms of the sheet velocity and the velocity jump

$$\boldsymbol{u}_1 = \boldsymbol{u}_{\text{sheet}} + \frac{1}{2}\boldsymbol{\Delta u} \qquad \text{and} \qquad \boldsymbol{u}_2 = \boldsymbol{u}_{\text{sheet}} - \frac{1}{2}\boldsymbol{\Delta u}. \tag{2.27}$$

Using these definitions, the Euler equations (1.34) without the baroclinic or surface tension terms can be written for each side of the vortex sheet (showing only side 1)

$$\frac{\partial \boldsymbol{u}_1}{\partial t} + \boldsymbol{u}_1 \cdot \nabla \boldsymbol{u}_1 = 0, \tag{2.28}$$

which expands to

$$\frac{\partial\left(\boldsymbol{u}_{\text{sheet}} + \frac{1}{2}\boldsymbol{\Delta u}\right)}{\partial t} + \left(\boldsymbol{u}_{\text{sheet}} + \frac{1}{2}\boldsymbol{\Delta u}\right) \cdot \nabla\left(\boldsymbol{u}_{\text{sheet}} + \frac{1}{2}\boldsymbol{\Delta u}\right) = 0. \tag{2.29}$$

The distributive property of the gradient and dot product allow this to be written as

$$\begin{aligned}\frac{\partial \boldsymbol{u}_{\text{sheet}}}{\partial t} + \frac{1}{2}\frac{\partial \boldsymbol{\Delta u}}{\partial t} + \boldsymbol{u}_{\text{sheet}} \cdot \nabla \boldsymbol{u}_{\text{sheet}} + \frac{1}{2}\boldsymbol{u}_{\text{sheet}} \cdot \nabla \boldsymbol{\Delta u} \\ + \frac{1}{2}\boldsymbol{\Delta u} \cdot \nabla \boldsymbol{u}_{\text{sheet}} + \frac{1}{4}\boldsymbol{\Delta u} \cdot \nabla \boldsymbol{\Delta u} = 0\end{aligned} \tag{2.30}$$

from which terms can be grouped to form

$$\frac{D\boldsymbol{u}_{\text{sheet}}}{Dt} + \frac{1}{2}\frac{D\boldsymbol{\Delta u}}{Dt} + \frac{1}{2}\boldsymbol{\Delta u} \cdot \nabla \boldsymbol{u}_{\text{sheet}} + \frac{1}{4}\boldsymbol{\Delta u} \cdot \nabla \boldsymbol{\Delta u} = 0. \tag{2.31}$$

Subtracting the resulting equation for side 2 from the above equation for side 1 gives

$$\frac{D\boldsymbol{\Delta u}}{Dt} = -\boldsymbol{\Delta u} \cdot \nabla \boldsymbol{u}_{\text{sheet}}, \tag{2.32}$$

which is the evolution equation for the velocity jump across a vortex sheet in the absence of density gradients and surface tension.

The vortex sheet strength, as defined above, is related to the velocity jump by

$$\boldsymbol{\gamma} = \hat{\boldsymbol{n}} \times \boldsymbol{\Delta u}, \tag{2.33}$$

where $\hat{\boldsymbol{n}}$ is the surface normal vector. As a result, the evolution equation for vortex sheet strength arrives by taking the cross product of the normal vector and equation (2.32). This requires the use of the following identity

$$\hat{\boldsymbol{n}} \times \frac{D\boldsymbol{\Delta u}}{Dt} = \frac{D\boldsymbol{\gamma}}{Dt} + \hat{\boldsymbol{n}}\left(\boldsymbol{\gamma} \cdot \frac{D\hat{\boldsymbol{n}}}{Dt}\right) \tag{2.34}$$

and the formula for the rate of change of the normal vector

$$\frac{D\hat{\boldsymbol{n}}}{Dt} = -\left(\boldsymbol{P} \cdot \nabla \boldsymbol{u}_{\text{sheet}}\right) \cdot \hat{\boldsymbol{n}}, \tag{2.35}$$

where $\boldsymbol{P}$ is the tangential projection operator

$$\boldsymbol{P} = \boldsymbol{I} - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}. \tag{2.36}$$

The resulting evolution equation for the vortex sheet strength in the absence of baroclinic (which will be derived in §2.4) and surface tension effects is thus

$$\frac{D\boldsymbol{\gamma}}{Dt} = \hat{\boldsymbol{n}}[\boldsymbol{\gamma} \cdot (\nabla \boldsymbol{u}_{\text{sheet}}) \cdot \hat{\boldsymbol{n}}] - \hat{\boldsymbol{n}} \times (\boldsymbol{\Delta u} \cdot \nabla \boldsymbol{u}_{\text{sheet}}). \tag{2.37}$$

If the vortex sheet is the only vorticity present in the flow, a further simplification achieves the following form

$$\frac{D\boldsymbol{\gamma}}{Dt} = \boldsymbol{\gamma} \cdot (\nabla \boldsymbol{u}_{\text{sheet}}) - \boldsymbol{\gamma}\left(\boldsymbol{P} \cdot \nabla \cdot \boldsymbol{u}_{\text{sheet}}\right), \tag{2.38}$$

where the first term on the right-hand side is the familiar vortex stretching term and the second term accounts for changes in sheet strength due to elongation in the direction of $\boldsymbol{\gamma}$. More detailed derivations appear in [Wu, 1995; Pozrikidis, 2000].

History

Vortex sheets have been used extensively in two dimensional vortex methods modeled as either independent particles [Rosenhead, 1931; Chorin and Bernard, 1973; Baker, 1979], connected but unremeshed segments [Zaroodny and Greenberg, 1973; Zalosh, 1976; Longuet-Higgins and Cokelet, 1978; Meng and Thomson, 1978; Baker *et al*., 1980,

1982; Anderson, 1985], connected and remeshed segments [Meng, 1978; Tryggvason and Aref, 1983; Krasny, 1986b, 1987a; Tryggvason, 1988b], or as higher-order elements [Fink and Soh, 1978; van de Vooren, 1980; Baker, 1980; Pullin, 1982].

In three dimensions, vortex sheets were first represented by collections of overlapping vortex filaments [Couët, 1979; Leonard, 1980a; Ashurst, 1983; Leonard, 1985; Ashurst and Meiburg, 1988; Martin and Meiburg, 1991], or particles with filament-like connectivity [Rehbach, 1978; Chorin, 1980]. Agishtein and Migdal [1989] appears to be the first demonstration of a three dimensional vortex method based on vorticity discretization onto two-dimensional sheet elements. Their work used flat connected triangles. Soon afterward, work by Knio and Ghoniem [1991, 1992a,b] tracked a scalar layer with triangular and quadrilateral elements, but still discretized the vorticity on filaments. The movements of the scalar elements, though, helped determine stretch rates for the vortex filaments. Newer results exist for regularized [Brady *et al*., 1998; Lozano *et al*., 1998; Pozrikidis, 2000] and singular [Haroldsen and Meiron, 1998; Beale, 2002] kernels and for level set methods [Harabetian *et al*., 1996].

Connectivity

A sheet element method must either maintain persistent node *and* connectivity information or have persistent nodes but recreate the connectivity information at regular time intervals (as is done in vortex volume methods in §2.3.4). Additionally, to be considered a vortex sheet method, connectivity must be maintained or deduced for both in-sheet directions, such as exists between data-connected filaments, or collections of connected triangles or quadrilaterals.

Connectivity among elements strongly influences remeshing, vortex stretching, and diffusion. More connectivity aids the former two operations, but hinders the latter. The balance often depends on the modeling assumptions. Thus, vortex methods with sheet elements usually simulate density discontinuities and ignore diffusion while vortex particle methods focus on full Navier-Stokes solutions. Filament methods are unfortunately caught in the middle, not being able to capture either regime as well. This is unfortunate, as most three-dimensional vortex sheet models use filament elements.

Remeshing

Like other discretizations, vortex sheets must retain adequate resolution or element overlap in the presence of rapid and extensive elongation in order to remain accurate for long simulations. Vortex sheets must account for this by remeshing or regridding in both in-sheet directions, unlike filament methods that only remesh in one direction and particle and volume methods that must remesh in all three.

Remeshing of the sheet elements can take both local and global forms. In the global sense, all of the sheet elements can be removed and replaced with a hopefully simpler set of elements. The advancing front method [Lo, 1985] has been used to generate triangular meshes for finite element problems, and has been extended to three dimensions [Hinton *et al*., 1991], incorporated schemes to interpolate values other than the surface location [Löhner, 1996], used in a dynamic simulation of a droplet in Stokes flow [Kwak and Pozrikidis, 1998], and finally used every 6-12 steps in a vortex method by Brady *et al*. [1998].

Another possible method is to place the vortex sheet strength onto a temporary Eulerian grid, extract isosurfaces using the marching cubes or a similar algorithm, and then set the new elements' strengths based on the grid values. This would enforce finite regularization and would simplify the surface, but at the cost of introducing grid-level irregularities inherent in the marching cubes algorithm. It is possible that Level Set methods [Chang *et al*., 1996; Harabetian *et al*., 1996; Osher and Fedkiw, 2001], too, could aid in global remeshing of a triangular mesh.

The only demonstrated global remeshing in vortex methods was in Brady *et al*. [1998], which remeshes according to curvature in a topologically two-dimensional parameter space. Additionally, this method does not allow long-time runs, as not only does frontal area increase exponentially, but vortex sheets tend to create areas of high curvature.

Local remeshing for vortex sheet methods dispenses with the global calculations of the above methods and instead remeshes only in the areas where the element distribution would lead to inaccuracies. The scheme for local remeshing is highly dependent upon the shape of the elements and the vorticity discretization onto those elements.

Because many vortex sheet methods are topologically filament methods, they can leverage the straightforward schemes described in §2.3.2 for filament-length remeshing.

These methods incur inefficiencies when remeshing in the cross-filament direction because typically a full-length filament must be added even if only a small length is needed. Methods which do not remesh in the filament-wise direction do not converge to the Euler equations [Greengard, 1986]. Despite that, relatively few filament-element vortex sheet methods actually remesh between filaments [Lindsay and Krasny, 2001].

The origins of local remeshing of a triangular mesh originate from the work of Fritts and Boris [1979] in two dimensions. This was extended to triangular meshes in three dimensions by Unverdi and Tryggvason [1992], but neither of these represent vortex methods. Being necessary for even intermediate-time vortex sheet simulations, it is no surprise that most true vortex sheet methods perform some form of local remeshing to maintain resolution in the presence of significant strain.

Knio and Ghoniem [1991] present a method to remesh transport elements in both in-sheet directions, even though the vorticity is still discretized as filaments on the edge of the elements. This earlier work does not maintain a perfectly-closed mesh, and also only allows remeshing in a topologically rectangular parameter space. Thus, numerous thin element are created in areas where the major axis of strain is diagonal to the elements, wasting computational resources. Their later work [Knio and Ghoniem, 1992a,b] allows a continuous mesh, but still relies on the rectangular remeshing. The simulations only run to intermediate times and the pile-up of thin, diagonally-strained elements has not yet become a problem.

A vortex sheet method can exist without remeshing [Tryggvason *et al*., 1991a; Lozano *et al*., 1998], but would suffer from either limited applicability to problems with low in-sheet strain or from lack of intermediate- to long-time simulation accuracy.

Vortex stretching

Obviously, vortex sheet methods based on filaments have little difficulty with the vortex stretching term, as the circulation on a filament is unchanged for most flows studied (2.23). Likewise, the connectivity inherent in true vortex sheet methods also eases evaluation of the stretching component, despite the more complex expression in equation (2.38).

Knio and Ghoniem [1991, 1992a] evaluate stretch on triangular elements directly from the movement of their corner nodes, and on rectangular elements using Gaussian quadratures over nine points within the element. It is noted therein that this method, which relies on the Helmholtz vorticity theorem, is only applicable to constant-density flows. In variable-density flows, these methods evaluate the stretch by differentiating the Biot-Savart law and performing summations per the velocity.

Brady *et al*. [1998] tracks circulation gradients on a separate two-dimensional parameter space, and thus accounts for stretching in the same manner as vortex filament methods.

Lozano *et al*. [1998] calculates the stretching term's influence with an expression that includes derivatives of velocity and relations among the triangular elements' edge vectors. It is likely that the mathematics of this approach are identical to that of previous works [Knio and Ghoniem, 1991, 1992a] which also use local geometry to compute stretch.

Diffusion

As many vortex sheet methods mean to simulate flow in the Euler limit, diffusion is not ordinarily explicitly calculated. To extend the applicability of vortex sheet methods beyond this regime requires a method to increase the vorticity support perpendicular to the sheet geometry. This has occurred in two dimensional methods, but is rare or limited in three dimensions.

A sheet equivalent to the earlier core-spreading methodology is defined by recognizing that surfaces are locally one-dimensional, and within certain limits, the diffusion equation can be solved analytically in locally one-dimensional regions. This is called the Local Integral Moment (LIM) method, and it is described in [Tryggvason and Dahm, 1991] and used for combustion [Chang *et al*., 1991] and vorticity diffusion [Tryggvason *et al*., 1991a]. The local diffusion thickens sheets of vorticity or scalar gradient, but like core-spreading, this method loses accuracy in areas where the thickness approaches the scale of sheet discretization. This method has not been demonstrated in three dimensions.

To correct this problem, it is possible to diffuse vorticity to new sheets created on both sides of an existing sheet. This suffers the same problems as remeshing filaments by adding whole new filaments: an entire new sheet must be created even in situations where

the elements are not needed. As such, it has not been attempted. Knio and Ghoniem [1991, 1992a], though, begin their three dimensional simulations with several closely-spaced sheets, though vorticity is not diffused between them, nor are extra sheets created when the existing sheets separate in the normal direction. Initially, the sheets are closely-spaced, but during their regular motion, the sheets become separated in their normal directions at places such as vortex cores. This loss of regularization may result in poor accuracy, if the situation is equivalent to that of filament separation or particle separation in related regularized methods.

Instead of creating numerous closely-spaced layers of sheet elements, it is reasonable for a vortex sheet that represents an immiscible interface to shed vortex particles into the regions of homogeneous fluid on either side of the interface. Many studies have demonstrated three-dimensional methods for diffusing vorticity from solid-body elements onto freely-convecting vortex particles [Koumoutsakos *et al*., 1994; Gharakhani and Ghoniem, 1996b; Cottet and Koumoutsakos, 1999] or body-fitted non-convecting vortex sheets [Bernard, 1996, 1999; Buron and Pérault, 1999]. Additionally, Cortez [2000] sheds vorticity from a flexible impenetrable surface, but only in two dimensions. No references to a three-dimensional hybrid sheet-particle method have been found.

Summary

Despite their programming complexity, it seems that vortex sheet methods garner certain advantages in the high-Re flow regime, not limited to: capable remeshing, easy computation of stretch, and (as shall be presented in §2.4) straightforward computation of baroclinic vorticity generation. Efforts to extend the applicability of vortex sheet methods into intermediate Reynolds numbers are scarce, and may rely on hybrid element methods.

2.3.4 Volume elements

A final, and less common, discretization is that of volume elements, whereby each element encloses a volume within which is a defined function of vorticity. Because there have been no implementations of vortex volume methods with persistent elements (similar to filament and sheet methods), this category will include methods in which

portions of the calculation are performed on temporary volume elements. The formula governing the evolution of the elements' strength depends entirely on the form of the elements employed, though all are based on the integral formulation of Helmholtz's equation (1.6). The velocity influence of a volume element is commonly calculated with Gaussian quadratures or integrals over the volumes of the elements. See the references for these equations.

History

A Lagrangian triangular mesh was first used in Chacon-Rebollo and Hou [1990], but for the Euler equations in velocity variables. Two dimensional vortex volume methods first appeared in Russo and Strain [1994] and Strain [1996], though vortex volumes were used in their earlier work [Russo, 1991] to compute the diffusion term. Modifications of the velocity-finding and stretch algorithms allowed extension to axisymmetric simulations in Carley [2002]. Fully three-dimensional dynamic vortex volume calculations using tetrahedra were suggested in Russo and Strain [1994] but not performed until Grant and Marshall [1999] and Marshall *et al*. [2000].

Connectivity

As stated above, no vortex volume methods use persistent elements—elements whose connectivity information is retained between time steps. All demonstrated vortex volume methods recreate their volume elements at each step from a collection of persistent nodes. For velocity calculations, these nodes are joined into triangles in two dimensional methods or tetrahedra in three dimensional methods via a process called Delaunay triangulation. This is similar to strength processing techniques in vortex particle methods, but instead of merely updating the strength of each particle, the Delaunay triangulation defines a specific region inside of which the vorticity is usually taken as some function of the nodes' vorticity.

Remeshing

The reason that the element boundaries are recalculated from a Delaunay triangulation at every time step (see above) is to alleviate the programmatically challenging task of locally remeshing tetrahedra. Based on the difficulty involved with simply remeshing triangle meshes in three dimensional space, this seems a wise choice. This operation constitutes a global remeshing. Local remeshing in the existing Delaunay methods consists simply of adding new nodes, usually at the center of an element, when the velocity or vorticity gradient across the element exceeds a threshold. These new nodes participate in the next global remeshing operation, creating smaller elements and higher resolution in areas with large gradients.

A two-dimensional vortex volume method (of the non-Delaunay type) may use any of the remeshing procedures used in three dimensional (manifold) vortex sheet methods, as they are no more topologically complex. A scheme for conservative remeshing of two dimensional meshes is given in Ramshaw [1985], though it remains unused in the context of vortex methods. The method of Chacon-Rebollo and Hou [1990] uses no remeshing whatsoever, resulting in a rapid loss of accuracy when the elements become distorted. Strain [1996] claims that it is difficult to make a two dimensional triangulated vortex method that is more than second-order in space, though a later generalization of triangulated methods [Strain, 1997] remedies that.

Vortex stretching

The only three dimensional vortex volume methods [Grant and Marshall, 1999; Marshall *et al*., 2000] both use a method of moving least squares to calculate spatial derivatives of flow quantities for the vortex stretching term. It is self-described as nonconservative.

Diffusion

The first diffusion method using Lagrangian volumes (to rule out finite element model calculations) was called the Free-Lagrange method [Fritts *et al*., 1985] which constructed discrete approximations to differential operators on a temporary mesh of Voronoi poly-

topes. Voronoi polytopes are regions whose boundaries are defined by the set of points equidistant from the central node and all nodes connected to that node in the Delaunay triangulation. Russo [1991] used the discretization scheme from the Free-Lagrange method for the viscous diffusion term in a two dimensional vortex method. The scheme is straightforward and conservative, and the method for constructing the Voronoi diagram is $\mathcal{O}\left(N \log N\right)$ or better in time. A different and non-conservative method was employed in the diffusion calculations in the three dimensional methods [Grant and Marshall, 1999; Marshall *et al*., 2000].

Summary

Vortex volume methods are somewhat similar to vortex particle methods in the respect that, through the Delaunay triangulation, the strengths of the resulting elements are modified in a manner resembling strength-processing schemes in particle methods. Leveraging the temporary volumetric discretization, volume methods allow more proper treatment of vorticity generation at solid surfaces, which is a problem common to vortex particle methods [Cottet and Koumoutsakos, 1999, §4.5]. In addition, the discretization of the Laplacian operator on a Voronoi tessellation allows conservative diffusion [Russo, 1991]. Nevertheless, vortex volume methods have few adherents, possibly owing to the complexity of the triangulation calculation, or the inability of three dimensional implementations to conservatively calculate spatial derivatives.

2.4 Baroclinic generation

The presence of density gradients in a flow causes creation and destruction of vorticity due to the influence of pressure gradients, most notably those caused by gravity. The form of the baroclinic generation term, though, will depend strongly on whether the density gradients are infinitely-thin sheets across which there is a density jump or a finite-thickness layers within which exist non-zero density gradients.

In the general sense, the baroclinic generation term in the nondimensional vorticity

update equation—now appearing as equations (1.23), (1.33), and (1.36)—is

$$\left.\frac{D\boldsymbol{\omega}}{Dt}\right|_{\text{baroclinic}} = \frac{1}{\rho}\left(\nabla\rho \times \nabla p\right). \tag{2.39}$$

A vortex particle method could use this equation directly by assigning a value for $\nabla\rho$ to each particle and composing the appropriate evolution equation.

Tightening the assumptions to discount viscosity allows some simplification. Rearranging the inviscid momentum equation (1.34) for ∇p gives

$$\nabla p = \frac{1}{\mathrm{Fr}}\frac{\boldsymbol{g}}{g} - \frac{D\boldsymbol{u}}{Dt}, \tag{2.40}$$

recalling that

$$\mathrm{Fr} \equiv \frac{U^2}{gL}, \tag{2.41}$$

and all other terms are dimensionless quantities. This equation (2.40) is not immediately valid on an inviscid vortex sheet because no fluid (and thus no definitive velocity $\boldsymbol{u}$) exists exactly on the vortex sheet. Combining (2.39) and (2.40) gives

$$\left.\frac{D\boldsymbol{\omega}}{Dt}\right|_{\text{baroclinic}} = \frac{1}{\mathrm{Fr}}\frac{1}{\rho}\nabla\rho \times \frac{\boldsymbol{g}}{g} - \frac{1}{\rho}\nabla\rho \times \frac{D\boldsymbol{u}}{Dt}, \tag{2.42}$$

which shows the origin of the gravity term in the aforementioned nondimensional vorticity evolution equations (1.23), (1.33), and (1.36).

In a method that tracks only sheets of discontinuity, such as the inviscid vortex sheet method presented herein, it makes sense to model the velocity as well as the density as discontinuous across the tracked surface. Thus, neither the vorticity $\boldsymbol{\omega}$ nor the density gradient $\nabla\rho$ in (2.42) are appropriate (both being infinite on the sheet itself) and a new form of the baroclinic generation term must be created. The solution, described earlier in §2.3.3, is to model the vorticity as a vortex sheet strength (a jump in the sheet-tangential velocity) and the density as a non-dimensional density jump.

The same arguments that support modeling high-Re flows with regularized inviscid vortex sheets (in §2.1) also support modeling high Péclet number flows with the same methods. Recall that the relative magnitudes of the diffusion term in both the nondimensionalized vorticity evolution (1.23) and scalar advection-diffusion (1.17) equations

are given by the Reynolds (1.21) and Péclet (1.18) numbers, respectively. The Schmidt number is the ratio of these two numbers, or

$$\mathrm{Sc} \equiv \frac{\nu}{D} = \frac{\mathrm{Pc}}{\mathrm{Re}}. \tag{2.43}$$

For Schmidt numbers near unity, which is frequently the case, the fluid species diffuse just as much as vorticity, and the conclusions made concerning regularization remain valid. Thus, the approximation of a slightly-dissipative two-fluid boundary by an identically non-dissipative interface should be reasonably accurate if $\mathrm{Sc} = \mathcal{O}(1)$.

Note that two fluids separated by a non-dissipative interface are said to be immiscible, and immiscible fluids are also frequently characterized by nontrivial amount of surface tension. An important dimensionless coefficient for surface tension is the Weber number, defined as

$$\mathrm{We} = \frac{\rho v^2 l}{\sigma}, \tag{2.44}$$

where σ is the coefficient of surface tension. The Weber number relates the influence of a fluid's inertia to its surface tension and can easily be much smaller than the Reynolds number. In this sort of situation, surface tension would play a more important role than viscous dissipation and a surface tension term would need to be included in the governing equations. Surface tension is most often encountered in flows with large density differences between fluid phases or when surfactants are present. Though surface tension will not be included in the present model, the use of sheet elements to discretize the interface eases the major burden of implementation—defining the curvature of the surface.

For simulations of inviscid vortex sheets, the baroclinic generation terms of the vorticity evolution equation can be derived from the Euler equation in the same manner as was done in §2.3.3. Writing the Euler momentum equation (1.34) for the points just above and below the sheet and replacing the velocities with functions of the sheet velocity and jump velocity

$$\boldsymbol{u}_1 = \boldsymbol{u}_{\mathrm{sheet}} + \frac{1}{2}\boldsymbol{\Delta u} \qquad \text{and} \qquad \boldsymbol{u}_2 = \boldsymbol{u}_{\mathrm{sheet}} - \frac{1}{2}\boldsymbol{\Delta u} \tag{2.45}$$

gives an expression for the baroclinic generation term for $\boldsymbol{\Delta u}$ variables

$$\left.\frac{D\boldsymbol{\Delta u}}{Dt}\right|_{\text{baroclinic}} = 2A\left(\bar{\boldsymbol{a}} - \frac{1}{\text{Fr}}\frac{\boldsymbol{g}}{g}\right), \tag{2.46}$$

where A is the Atwood ratio

$$A \equiv \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2} = \frac{\Delta\rho}{2\rho} \in [-\infty, \infty], \tag{2.47}$$

and $\bar{\boldsymbol{a}}$ is the average of the fluid acceleration on both sides of the vortex sheet, or

$$\bar{\boldsymbol{a}} = \frac{1}{2}\left(\frac{D\boldsymbol{u}_1}{Dt} + \frac{D\boldsymbol{u}_2}{Dt}\right) = \frac{D\boldsymbol{u}_{\text{sheet}}}{Dt} + \frac{1}{4}\boldsymbol{\Delta u}\cdot\nabla\boldsymbol{\Delta u}. \tag{2.48}$$

The baroclinic generation term in the vortex sheet evolution equation is obtained by taking the cross-product of the sheet normal vector $\hat{\boldsymbol{n}}$ with the above equation (2.46), making

$$\left.\frac{D\boldsymbol{\gamma}}{Dt}\right|_{\text{baroclinic}} = 2A\,\hat{\boldsymbol{n}}\times\left(\bar{\boldsymbol{a}} - \frac{1}{\text{Fr}}\frac{\boldsymbol{g}}{g}\right). \tag{2.49}$$

See [Baker and Moore, 1989; Wu, 1995; Pozrikidis, 2000] for more thorough derivations.

2.4.1 Boussinesq limit

In the simplest case, it shall be assumed that the density jump is small ($\Delta\rho/\rho \ll 1$, thus $A \to 0$) and that the Froude number is small (or $g \to \infty$), such that the coefficient of the baroclinic term is constant and finite:

$$A\,g \equiv \theta. \tag{2.50}$$

Under these assumptions, the coefficient on the term representing baroclinic generation due to hydrodynamic pressure is zero, and thus only hydrostatic pressure effects are treated. This is the Boussinesq approximation. The Boussinesq coefficient θ will be assumed to have been nondimensionalized in a manner similar to the Froude number,

$$\theta^* = \frac{A}{\text{Fr}} = A\,\frac{g\,L}{U^2} = \theta\,\frac{L}{U^2}. \tag{2.51}$$

For simulations with more than two fluids, the relative magnitudes of the density jumps can be related with a unique θ for each interface. The final form of the baroclinic generation of vortex sheet strength is

$$\left.\frac{D\boldsymbol{\gamma}}{Dt}\right|_{\text{baroclinic}} = -2\,\theta\,\hat{\boldsymbol{n}} \times \frac{\boldsymbol{g}}{g}. \tag{2.52}$$

The Boussinesq approximation can also be derived by assuming that the pressure can be treated to only first order ($\nabla p = \rho_0\,\boldsymbol{g} + \mathcal{O}(\frac{\nabla\rho}{\rho_0})$ [Meng and Thomson, 1978; Meng, 1978], or ($\nabla p = \rho_0\,\boldsymbol{g} + \mathcal{O}(\frac{\nabla\rho}{p_0})$ [Brecht and Ferrante, 1990], which also makes the hydrodynamic term insignificant.

The Boussinesq approximation was first used in vortex flows in the VIC method of Meng and Thomson [1978] for the simulation of a buoyant cylinder and gravity current in two dimensions, and also by Meng [1978] for the study of vortex ring evolution in stratified or shearing flows. This same formulation can be used for the Taylor-Saffman instability of flow through a porous medium, as was pointed out by de Josselin de Jong [1960] and demonstrated by Meng and Thomson [1978]. Other examples of vortex methods with weak density discontinuities include [Anderson, 1985], but still only in two dimensions.

2.4.2 Stronger density discontinuities

In the presence of stronger density discontinuities ($|A| \geq 0.1$) the full form of the baroclinic term (2.49) must be used. The obvious problem with this form is that the fluid acceleration cannot be accurately established until the kinematic computation of velocity is complete (or by using backward differences [Knio and Ghoniem, 1992a]). Methods first used to solve this problem include direct summation [Zaroodny and Greenberg, 1973; Zalosh, 1976], matrix inversion [Longuet-Higgins and Cokelet, 1976], and iterative methods [Baker *et al.*, 1980, 1982; Tryggvason and Aref, 1983]. Brecht and Ferrante [1989, 1990] demonstrated the first vortex method to use strong stratification in three dimensions, and were later followed by [Knio and Ghoniem, 1992a; Lozano *et al.*, 1998; Haroldsen and Meiron, 1998]

An interesting alternative derivation of the baroclinic term for strong stratification

is given by [Brecht and Ferrante, 1989, 1990], which integrates (2.42) over a volume to obtain an expression for baroclinic generation of circulation as a function of the logarithm of the density ratio, or

$$\left.\frac{D\boldsymbol{\Gamma}}{Dt}\right|_{\text{baroclinic}} = |\boldsymbol{n}|\ \ln\left(\frac{\rho_2}{\rho_1}\right)\hat{\boldsymbol{n}} \times \left(\boldsymbol{g} - \frac{D\boldsymbol{u}}{Dt}\right), \tag{2.53}$$

where $|\boldsymbol{n}|$ is the normalized area represented by a given particle. Because this method presumes that the vorticity and density gradient exist within a finite volume (or finite-thickness layer), there now *does* exist a value for the material acceleration of a point midway across the sheet. It would seem, though, that the integration of the fluid acceleration would include alternate terms—such as appear in (2.48)—if the velocity profile were not linear; this is not accounted for in the reference. Following the same procedure to convert this expression to be applicable to a finite-thickness vortex sheet gives

$$\left.\frac{D\boldsymbol{\gamma}}{Dt}\right|_{\text{baroclinic}} = \ln\left(\frac{\rho_2}{\rho_1}\right)\hat{\boldsymbol{n}} \times \left(\boldsymbol{g} - \frac{D\boldsymbol{u}}{Dt}\right). \tag{2.54}$$

Conveniently, the logarithmic term in both of these forms simplifies in the case of $\rho_1 \approx \rho_2$. To show this, define

$$\Delta\rho = \rho_2 - \rho_1 \qquad \text{and} \qquad \rho_0 = \frac{1}{2}(\rho_2 + \rho_1), \tag{2.55}$$

which then makes

$$\frac{\rho_2}{\rho_0} = 1 + \frac{\Delta\rho}{2\rho_0} \qquad \text{and} \qquad \frac{\rho_1}{\rho_0} = 1 - \frac{\Delta\rho}{2\rho_0}. \tag{2.56}$$

The power series expansion of the natural logarithm

$$\ln\left(\frac{1+x}{1-x}\right) = 2\left[x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots\right] \qquad (|x| < 1) \tag{2.57}$$

allows the following simplification

$$\ln\left(\frac{\rho_2}{\rho_1}\right) = \ln\left(\frac{1 + \frac{\Delta\rho}{2\rho_0}}{1 - \frac{\Delta\rho}{2\rho_0}}\right) \approx 2\frac{\Delta\rho}{2\rho_0} = 2A. \tag{2.58}$$

This brings the baroclinic generation term for finite-thickness sheets (2.54) nearly in line with the infinitely-thin version (2.49), with the exception of acceleration (which in the Boussinesq approximation drops out, owing to $|\boldsymbol{g}| \gg |\boldsymbol{a}|$).

2.5 Sub-filter scale dissipation

The engineering demands of human size and time scales is at odds with the important physical properties of the fluids that humans operate within. The viscosities of common fluids combined with the velocity scales encountered in engineering applications dictate that the smallest unique scales of motion are on the order of several microns. In order to gain predictive capability of human-scale flows, the motion of very small scales must be accounted for. Direct numerical simulation (DNS), wherein all of the scales of motion are fully resolved, requires computational effort equivalent to the cube of the Reynolds number [in Pope, 2000, Chapter 9].

A promising explicit method to reduce these computational requirements is called Large Eddy Simulation (LES). LES arose from the realization that the statistics of the small and intermediate scales of turbulent motion are mostly self-similar and isotropic but most of the energy and anisotropy are contained in the large scales. Thus, an LES method defines a spatial filter, above the length scale of this filter are the "resolved scales" and below the "unresolved scales." The filter length is usually set such that $> 80\%$ of the energy is resolved. Other steps include rewriting the governing equations, achieving closure by modeling the subgrid-scale (SGS) residual stress tensor, and numerically integrating the resolved equations. Nevertheless, most explicit LES models have only achieved simulation stability, and cannot reproduce local residual stress and energy production fields [review in Burton and Dahm, 2005].

It has been noted that some vortex methods—implicit in their construction—exercise some form of subgrid-scale dissipation, making them LES methods. Couët *et al*. [1981] claims that the grid dissipation in a VIC method is equivalent to LES-like subgrid-scale dissipation and uses a cutoff filter in Fourier space to smooth the aliasing created by rectangular interpolation kernels. Chorin [1990, 1993] proposes purely Lagrangian LES schemes called "hairpin-removal" whereby local mesh redistribution of vortex filaments

aims to absorb the effects of the smallest length scales. Finally, any scheme which, in three dimensions, serves to reduce the resolved velocity gradient also provides implicit dissipation [Gharakhani, 2004]. Results have not yet shown that these implicit models remove the proper amount of energy from the proper places, but many do exhibit the characteristics of LES-computed flows.

An important step in an explicit LES method is to write the filtered equations of motion. Nearly all LES models filter the Navier-Stokes momentum equation, but it would be more natural for a vortex method to begin by filtering the vorticity evolution equation. With an overbar representing a filtered or grid-resolved quantity, the filtered vorticity equation is

$$\frac{D\,\overline{\omega}_i}{Dt} = \overline{\boldsymbol{\omega}} \cdot \nabla \overline{u}_i + \nu \nabla^2 \overline{\omega}_i + \frac{\partial}{\partial x_j}(\Phi_{ij} - \Phi_{ji}), \tag{2.59}$$

with

$$\Phi_{ij} = \overline{\omega_i u_j} - \overline{\omega}_i \overline{u}_j \tag{2.60}$$

representing the residual Helmholtz, or vorticity, stresses. These are the only terms in the filtered vorticity equation that do not appear in the original equation (1.6), and they model the influence of the unresolved, or subgrid, scales of motion on the resolved scales. Note that few vorticity-LES schemes exist, and fewer still model the equation in this manner.

As a basic example, Mansfield *et al*. [1996] proposes an explicit Smagorinsky-equivalent [Smagorinsky, 1963] subgrid-scale model for the filtered vorticity equations. In a Smagorinsky model, the subgrid stresses—in this case expressed by (2.60)—are modeled with a linear eddy viscosity model. For the Helmholtz stresses above, this takes the form of

$$\Phi_{ij} = \nu_T \frac{\partial \overline{\omega}_i}{\partial x_j}, \tag{2.61}$$

where the eddy diffusivity ν_T is

$$\nu_T = (c_T \delta)^2 \sqrt{2 \overline{S}_{ij} \overline{S}_{ij}}, \tag{2.62}$$

the filtered rate-of-strain tensor $\overline{S}_{ij}$ is

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right), \tag{2.63}$$

and the constant in the eddy diffusivity equation is taken as $c_T = 0.15$ [Mansfield *et al*., 1996], similar in magnitude to the constant in the Smagorinsky model in velocity variables. The evolution equation for vorticity can thus be written

$$\frac{D\,\overline{\omega}_i}{Dt} = \overline{\boldsymbol{\omega}} \cdot \nabla \overline{u}_i + \nu \nabla^2 \overline{\omega}_i + \nu_T \left(\frac{\partial \overline{\omega}_i}{\partial x_j \partial x_j} - \frac{\partial \overline{\omega}_j}{\partial x_i \partial x_j}\right). \tag{2.64}$$

It is worth noting that (2.64) depends only on resolved values and their derivatives. Mansfield *et al*. [1996] mentions that Φ_{ij} represents subgrid-scale vortex stretching and tilting due to unresolved motion and Φ_{ji} reflects vortex transport by subgrid-scale velocity fluctuations.

Many LES vortex methods apply subgrid-scale dissipation using methods originally developed for isotropic viscous dissipation. This is easy to see by observing the similarities between the second and third terms of equation (2.64): both contain second derivatives in space of the resolved vorticity field. The two primary Lagrangian methods for accounting for viscous dissipation in vortex methods are the particle strength exchange (PSE) scheme [Degond and Mas-Gallic, 1989a,b] and the vorticity redistribution method (VRM) [Shankar and van Dommelen, 1996]. The PSE method works by approximating the diffusion operator with an integral one, and discretizing it over a collection of local particles. VRM conserves moments to arbitrarily-high orders of accuracy by solving a system of equations involving all local particles and requiring insertion of new particles if no acceptable answer is found. Alternatively, the diffusion operator could be discretized on a temporary, overlaid grid, as done in [Graham, 1988; Najm, 1993; Liu and Doorly, 2000]. Other methods for accounting for viscous effects exist [Chorin, 1973; Börgers and Peskin, 1987; Rossi, 1996] but are tangent to the discussion.

Recent years have seen the beginning of LES particle vortex methods. Cottet [1996] and Cottet *et al*. [2003] present an anisotropic version of the PSE scheme to account for errors introduced by unremeshed vortex particle methods. Mansfield *et al*. [1996] also

CHAPTER 3

Proposed Numerical Method

This chapter will present the numerical and computational methods which can be used to solve for the motion of regularized vortex sheets.

This description will begin in §3.1 with the scheme for vorticity discretization, followed in §3.2 by the interpolation methods required to convey information between the regular computational grid and the Lagrangian elements. Then, the technique used to calculate the velocity field from the vortex elements will be detailed in §3.3. Following that will be an introduction to the triangle-edge-splitting and adjacent-node-merging schemes in §3.4 and §3.5 with detailed descriptions and procedures. Section 3.6 will describe the method for creating new vortex sheets as they are shed from splitter plates and other solid edges. Then, §3.7 will present the treatment of the vorticity transport equation with respect to weak density discontinuities. Section 3.8 will introduce an experimental scheme for element-based subfilter-scale dissipation. Finally, a short discussion of the numerical integration techniques will appear in §3.9.

A review of the key modeling equations is warranted before jumping into the current chapter. The equation of primary importance is the one governing the dynamic evolution of the vortex sheet strength, developed in §2.3.3 and §2.4, and repeated here.

$$\frac{D\boldsymbol{\gamma}}{Dt} = \underbrace{\boldsymbol{\gamma} \cdot \nabla \boldsymbol{u}}_{stretch} \underbrace{- \boldsymbol{\gamma}(\boldsymbol{P} \cdot \nabla \cdot \boldsymbol{u})}_{dilatation} \underbrace{+ 2\, A\, \hat{\boldsymbol{n}} \times (\bar{\boldsymbol{a}} - \boldsymbol{g})}_{baroclinic}, \tag{3.1}$$

In the above equation $\boldsymbol{P} = \boldsymbol{I} - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T$ is the tangential projection operator. This equation differs from the pure vorticity evolution equation by the inclusion of the in-sheet

dilatation term. The equivalent equation for circulation is

$$\frac{D\Gamma}{Dt} = \underbrace{2\,A\,\left(\frac{D\phi}{Dt} - \frac{1}{2}|\boldsymbol{u}|^2 + \frac{1}{8}|\Delta\boldsymbol{u}|^2 - \boldsymbol{g}\cdot\boldsymbol{x}\right)}_{baroclinic}, \tag{3.2}$$

which, notably, depends only on the existence of density discontinuities. This feature will be exploited in the coming sections.

The other important equation is the kinematic velocity equation, which relates the vorticity field to the velocity field.

$$\nabla^2\boldsymbol{u} = -\nabla\times\boldsymbol{\omega} \tag{3.3}$$

Its origins are described in §2.2, and has been used in three-dimensional vortex-in-cell methods for over two decades.

Upon completion of this chapter, the complete computational method for the calculation of the motion of regularized vortex sheets just described will be ready to be exercised by the examples in Chapter 4.

3.1 Element discretization

As is common in three-dimensional front-tracking schemes, the front, in our case a vortex sheet, is discretized into flat triangles, each defined by its connectivity to three Lagrangian nodes. The edges of each triangular element p store scalar-valued circulations $\Gamma_{p,1\to3}$. As mentioned in §2.3, a triangle with defined edge circulations can be represented as both a triangle with an area-distribution of vortex sheet strength, and as a particle with a point-value of total vorticity.

The interchangeability of these three different discretization schemes allows the method to take advantage of the different formulations for their evolution equations. A primary example pertains to the accounting of three-dimensional vortex stretching. The evolution equations for circulation (Γ) contain no vortex stretching term at all, thus circulation is a natural choice for vorticity discretization in a vortex method. Likewise, as will be seen in this and §3.4, accounting for extensional strain transverse to the direction of vortic-

ity is handled well by representing the vorticity as areas of constant-valued vortex sheet strength (γ). Representing the element vorticity as a blob of total vorticity (α), as is common for most particle vortex methods, is less useful in a vortex sheet method, and shall only be mentioned here for the sake of completeness. As presented in §2.3, the conversions between these three representations are as follows.

$$\boldsymbol{\alpha}_p = v_p\,\boldsymbol{\omega}_p = a_p\boldsymbol{\gamma}_p = \sum_{i=1}^{3} \Gamma_{p,i}\,\Delta l_{p,i} = \frac{L^3}{T}, \tag{3.4}$$

where v_p and a_p are the element volume and area, and $\Delta l_{p,1\to3}$ are the edge vectors.

3.1.1 Implementation

Equation (3.4) can be used to easily convert an element's total vorticity from one representation to another, except for the operation to determine edge circulations. When a triangle's edge circulations must be set from a given change in total vorticity, a routine is called to solve the following overconstrained set of equations.

$$\begin{bmatrix} \Delta l_{p,1,x} & \Delta l_{p,2,x} & \Delta l_{p,3,x} \\ \Delta l_{p,1,y} & \Delta l_{p,2,y} & \Delta l_{p,3,y} \\ \Delta l_{p,1,z} & \Delta l_{p,2,z} & \Delta l_{p,3,z} \\ 1 & 1 & 1 \end{bmatrix} \begin{pmatrix} \Gamma_{p,1} \\ \Gamma_{p,2} \\ \Gamma_{p,3} \end{pmatrix} = \begin{pmatrix} \alpha_{p,x} \\ \alpha_{p,y} \\ \alpha_{p,z} \\ 0 \end{pmatrix} \tag{3.5}$$

When the total vorticity is planar to the triangular element, the same total vorticity can be recovered with equation (3.4). If the total vorticity in (3.5) is not planar to the element the matrix will still solve but the total vorticity will be reoriented and strengthened. This behavior is actually desirable and will be described in context in §3.4.3.

3.1.2 Validation

To test the capability of this unique discretization method, two simple tests were performed. These two tests will show that, even without remeshing, the method can maintain accurate circulation and vorticity under stretch along *both* in-plane directions (parallel and transverse to the vorticity).

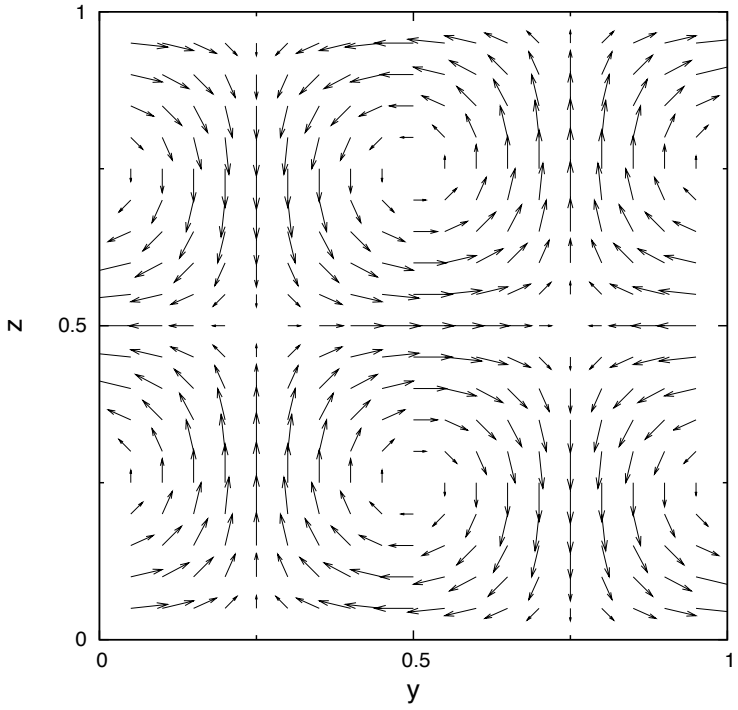

Figure 3.1: Velocity field for discretization tests; note velocity on $z = 0.5$ plane.

For each test, an artificial, divergence-free velocity field of the form

$$\begin{aligned} u_x &= u = 0 \\ u_y &= v = \cos(2\pi y)\cos(2\pi z) \\ u_z &= w = \sin(2\pi y)\sin(2\pi z) \end{aligned} \tag{3.6}$$

is artificially imposed in lieu of the kinematic velocity equation. This velocity field is illustrated in figure 3.1. The computational domain covers $[0:1][0:1][0:1]$ and exhibits periodicity in all three coordinate directions. In each test, a flat vortex sheet Ψ, completely spanning the domain and connecting to itself across the periodic boundaries, is created at $z = 0.5$. This sheet is composed of a 19 by 19 grid of triangle pairs and is illustrated at $t = 0$ and $t = 0.24$ in figure 3.2. The capability of the method will be determined by assembling the vorticity field from the computational elements using the methods put forth in §3.3.1 on a regular grid using the area-weighting method and with $\Delta x = 0.05$. Thus, a unit-magnitude vortex sheet strength will correspond to a vorticity

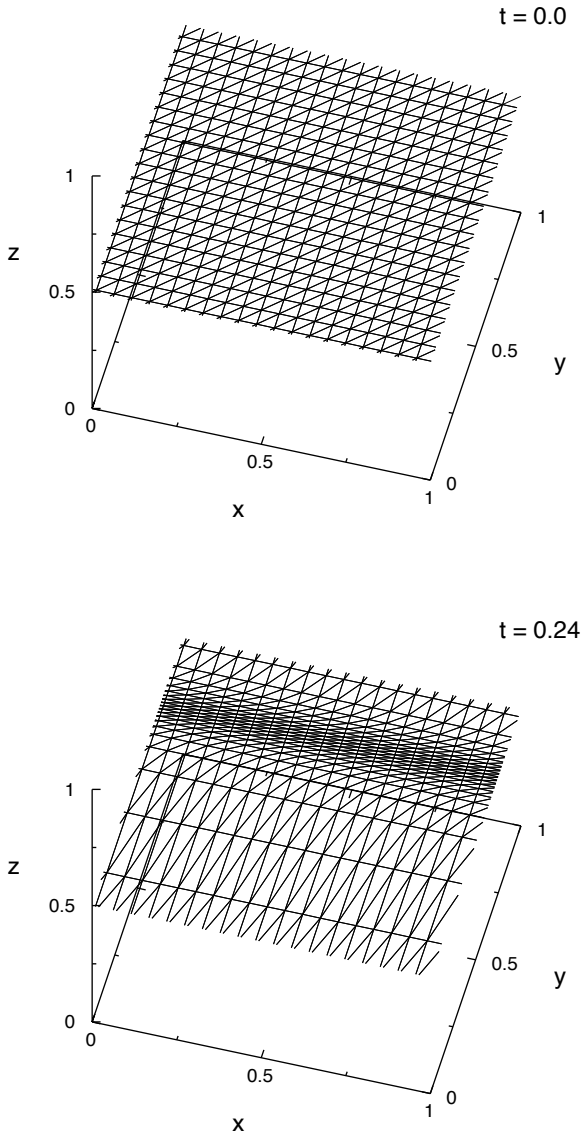

Figure 3.2: Computational elements for discretization tests, start and end times.

magnitude of $1/\Delta x = 20$.

Note that there is no z-component of velocity in the plane of the sheet, so the computational elements will remain at $z = 0.5$ for the duration of the simulation. The rates of strain at $z = 0.5$ are as follows:

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = 0 \\ \frac{\partial v}{\partial y} &= 2\pi \sin(2\pi y). \end{aligned} \tag{3.7}$$

Any vorticity in the y-direction is thus expected to experience positive and negative stretch, and any x-vorticity to be unaffected by stretch but undergo convection.

Expanding the material derivative in equation (3.1) and removing the unnecessary baroclinicity and external forcing terms produces the relevant equation governing the change in vortex sheet strength for this exercise.

$$\frac{\partial \boldsymbol{\gamma}}{\partial t} = \underbrace{-(\boldsymbol{u} \cdot \nabla)\,\boldsymbol{\gamma}}_{convection} \underbrace{+(\boldsymbol{\gamma} \cdot \nabla)\,\boldsymbol{u}}_{stretch} \underbrace{-\boldsymbol{\gamma}(\boldsymbol{P} \cdot \nabla \cdot \boldsymbol{u})}_{dilatation} \tag{3.8}$$

The tangential projection operator for a sheet with normal $\hat{k}$ is $\boldsymbol{P} = \boldsymbol{I} - \hat{k}\hat{k}^T$. Applying this operator, and removing all u and $\partial/\partial x$ terms from the equation above reveals the evolution equations for each component of vortex sheet strength.

$$\frac{\partial \gamma_x}{\partial t} = -v\frac{\partial \gamma_x}{\partial y} - w\frac{\partial \gamma_x}{\partial z} - \gamma_x\frac{\partial v}{\partial y} \tag{3.9}$$

$$\frac{\partial \gamma_y}{\partial t} = -v\frac{\partial \gamma_y}{\partial y} - w\frac{\partial \gamma_y}{\partial z} + \gamma_y\frac{\partial v}{\partial y} + \gamma_z\frac{\partial v}{\partial z} - \gamma_y\frac{\partial v}{\partial y} \tag{3.10}$$

$$\frac{\partial \gamma_z}{\partial t} = -v\frac{\partial \gamma_z}{\partial y} - w\frac{\partial \gamma_z}{\partial z} + \gamma_y\frac{\partial w}{\partial y} + \gamma_z\frac{\partial w}{\partial z} - \gamma_z\frac{\partial v}{\partial y} \tag{3.11}$$

For the first test, the vortex sheet strength is

$$\boldsymbol{\gamma}(\Psi, t = 0) = 1.0\hat{j} \tag{3.12}$$

meaning that the only strain present in the plane of the vortex sheet acts in the direction of the vorticity. Application of the boundary conditions to equations (3.9-3.11) yields the

solution

$$\frac{\partial \boldsymbol{\gamma}}{\partial t} = 0. \tag{3.13}$$

As such, the vortex sheet strength over the sheet Ψ is expected to be constant. Note that this is *not* the same solution that would come from the vorticity evolution equation. The difference stems from the presence of the dilatation term in the vortex sheet strength evolution equation (3.8), absent in the vorticity equation.

The performance of the method shall be judged by examining the vorticity field created from the computational elements. Results of the simulation appear in figure 3.3. As the figure indicates, the vorticity field loses some smoothness as the elements stretch, but remains solidly near the expected value, $\omega_y = \gamma_y/\Delta x = 20$. This same test shall be repeated in §3.4.4 with the element remeshing routines enabled.

Vortex sheet methods with our proposed discretization method are able to account for stretch along the direction of vorticity to machine precision, despite the extreme element deformation. Vortex filament methods are similarly capable of maintaining resolution in the case where the only strain is parallel to vorticity. In this case, the endpoints of the segments that constitute the filaments are pulled apart along the direction of the filament, resulting in no actual change to the morphology of the filaments—merely a thinning and clustering of endpoints along the filaments. Elements need not be split to maintain the discretization accuracy unless the vortex lines become curved. Vortex particle methods, on the other hand, require remeshing or element splitting to maintain discretization accuracy for even this simple case.

The second test sees the vortex sheet strength set to

$$\boldsymbol{\gamma}(\boldsymbol{x}, t = 0) = 1.0i. \tag{3.14}$$

In this case, the component of strain is in-plane, but transverse to the vorticity. This is the type of strain that neither standard vortex filament nor vortex particle methods can easily account for. Simplifying the evolution equations (3.9-3.11), and recognizing that $\partial\gamma_x/\partial z = 0$ returns

$$\frac{\partial \gamma_x}{\partial t} = -v\frac{\partial \gamma_x}{\partial y} - \gamma_x\frac{\partial v}{\partial y}, \tag{3.15}$$

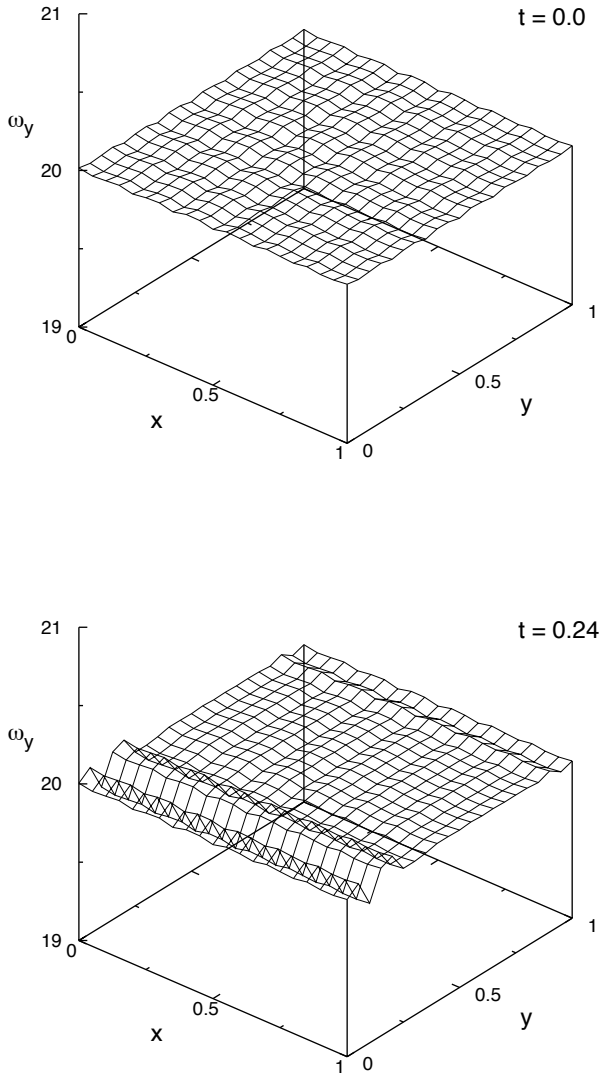

Figure 3.3: ω_y magnitude, $z = 0.5$ plane, start and end times.

and substituting the velocity and its derivatives (with $\cos(2\pi z) = -1$ on Ψ) gives

$$\frac{\partial \gamma_x}{\partial t} = \cos(2\pi y)\frac{\partial \gamma_x}{\partial y} - 2\pi \sin(2\pi y)\gamma_x. \tag{3.16}$$

The full analytical solution asymptotes to infinity at $y = 0.75$, a state that will not be reached in the simulation due to regularization. The solutions along the lines of peak extensional strain ($y = 0.25$) and peak compressional strain ($y = 0.75$) are the exponentials

$$\gamma_x(y = 0.25, z = 0.5, t \geq 0) = e^{-2\pi t} \tag{3.17}$$

$$\gamma_x(y = 0.75, z = 0.5, t \geq 0) = e^{2\pi t}, \tag{3.18}$$

which correspond to grid vorticity values of

$$\omega_x(y = 0.25, z = 0.5, t \geq 0) = 20\, e^{-2\pi t} \tag{3.19}$$

$$\omega_x(y = 0.75, z = 0.5, t \geq 0) = 20\, e^{2\pi t}, \tag{3.20}$$

The simulation, though, can only accurately track this solution in the areas that are not significantly smoothed out by regularization. For example, the solution at $y = 0.75, z = 0.5$ (3.20) is a peak that in the analytic solution is narrower than this simulation's regularization length scale ($\delta = \Delta x = 0.05$). Figure 3.4 illustrates ω_x over the whole sheet at the maximum solved time. In figure 3.5 are plots of the vorticity vs. time at along the lines of peak extensional and compressional strain. The effect of regularization is not very pronounced in the latter figure, as the simulation could only run to $t = 0.26$. Simulation results to $t = 1.0$ will appear in §3.4.4 and §3.5.4 and will show that regularization significantly reduces peak vorticity.

Figure 3.6 compares the results of the proposed method to one that only uses vortex filaments to represent the vortex sheet. During the sheet evolution, extensional strain pulls the individual vortex lines apart in the transverse direction, eventually separating them beyond the ability of the grid to resolve them smoothly. Thus, a vortex filament method without the capacity to remesh is clearly incapable of resolving a smooth vortex sheet, even in a simulation as simple as this.

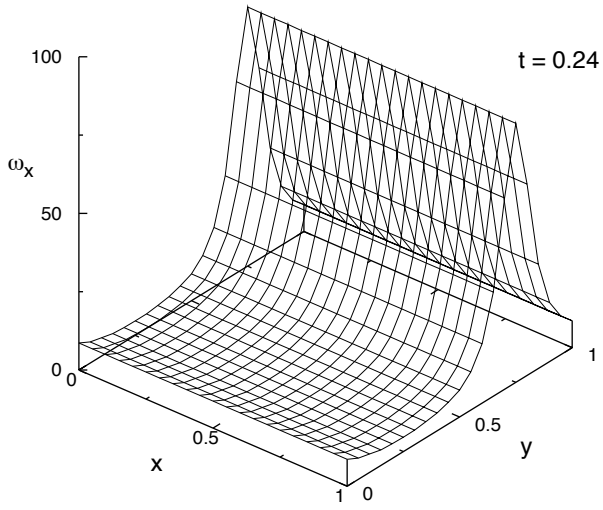

Figure 3.4: ω_x magnitude, $z = 0.5$ plane, end time.

Further tests demonstrating accuracy under prolonged strain and folding require periodic remeshing to maintain the accuracy of the sheet and will be presented in §3.4.4 and §3.4.5.

3.1.3 Caveat

One deficiency of the proposed method is its inherent tendency to convert second derivatives of velocity into rotations. This is a problem with all methods that use triangles with bound circulations to represent vortex sheets.

The following examples will illustrate this effect. A flat, infinite, triangulated vortex sheet lies in the x-y plane. One triangular within that sheet, shown in figure 3.7 has edge circulations $\Gamma_{1..3}$.

Two elementary cases will be studied, each with the same geometry but with the

following different edge circulations

$$\Gamma_{a,1..3} = \Big[+\frac{2}{3}, -\frac{1}{3}, -\frac{1}{3} \Big] \tag{3.21}$$

$$\Gamma_{b,1..3} = \Big[0, +\frac{1}{2}, -\frac{1}{2} \Big]. \tag{3.22}$$

Equation (3.4) gives the initial vortex sheet strength for each case:

$$\begin{aligned}
a\boldsymbol{\gamma} &= \sum_{i=1}^{3} \Gamma_i \, \Delta l_i \\
\frac{1}{2}\boldsymbol{\gamma}_a &= \frac{2}{3}\Big(\hat{i}\Big) - \frac{1}{3}\Big(-\frac{1}{2}\hat{i} + \hat{j}\Big) - \frac{1}{3}\Big(-\frac{1}{2}\hat{i} - \hat{j}\Big) \\
\boldsymbol{\gamma}_a &= 2\hat{i} \qquad (3.23) \\
\frac{1}{2}\boldsymbol{\gamma}_b &= +\frac{1}{2}\Big(-\frac{1}{2}\hat{i} + \hat{j}\Big) - \frac{1}{2}\Big(-\frac{1}{2}\hat{i} - \hat{j}\Big) \\
\boldsymbol{\gamma}_b &= 2\hat{j}. \qquad (3.24)
\end{aligned}$$

It will be assumed that the entire vortex sheet for each case begins with constant $\boldsymbol{\gamma}$. Case a corresponds a sheet with vorticity parallel to the velocity, and case b with vorticity transverse to the velocity.

The velocity field is

$$\boldsymbol{u}(\boldsymbol{x}, t) = x^2\hat{i}, \tag{3.25}$$

which allows simplification of the evolution equations for vortex sheet strength (3.8) to

$$\frac{\partial \gamma_x}{\partial t} = -x^2 \frac{\partial \gamma_x}{\partial x} \tag{3.26}$$

$$\frac{\partial \gamma_y}{\partial t} = -x^2 \frac{\partial \gamma_y}{\partial x} - 2x\gamma_y. \tag{3.27}$$

The solution to these equations for case a is simply

$$\boldsymbol{\gamma}_a(\boldsymbol{x}, t) = 2\hat{i} \tag{3.28}$$

because $\partial \gamma_{a,x}/\partial x = 0$ at $t = 0$. The solution for case b is complicated by the linear term,

but can be solved by separation of variables, resulting in

$$\boldsymbol{\gamma}_b(\boldsymbol{x}, t) = 2^{1-xt} x^{2xt} \hat{i}. \tag{3.29}$$

The theoretical and actual vortex sheet strengths of these two cases will be compared after undergoing a simple forward integration step of $\Delta t = 1$.

After one Euler step the nodes have relocated to their primed positions, the new triangle area is $a' = 1$, and the total vorticity for each case can be computed using the new edge vectors.

$$\begin{aligned} \boldsymbol{\gamma}_a &= \frac{2}{3}\Big(2\hat{i}\Big) - \frac{1}{3}\Big(-\frac{5}{4}\hat{i} + \hat{j}\Big) - \frac{1}{3}\Big(-\frac{3}{4}\hat{i} - \hat{j}\Big) \\ \boldsymbol{\gamma}'_a &= 2\hat{i} \end{aligned} \tag{3.30}$$

$$\begin{aligned} \boldsymbol{\gamma}_b &= +\frac{1}{2}\Big(-\frac{5}{4}\hat{i} + \hat{j}\Big) - \frac{1}{2}\Big(-\frac{3}{4}\hat{i} - \hat{j}\Big) \\ \boldsymbol{\gamma}'_b &= -\frac{1}{4}\hat{i} + \hat{j}. \end{aligned} \tag{3.31}$$

Case a obeys the evolution equation (3.28) exactly, but case b has created an unphysical vortex sheet strength component. This new component appeared because a finite-sized triangle is being used to approximate a continuous sheet. The tilted vorticity is mitigated somewhat by neighboring triangles that experience equal and opposite artificial rotations. This dampening effect sustains itself only as long as the affected triangles remain in close proximity, something that is not guaranteed in typical vortex sheet evolution.

This effect occurs no matter which parts of the triangle the circulation is attached to: corner-to-corner, midpoint-to-corner, or midpoint-to-midpoint.

Obviously, minimizing this effect means maintaining triangular elements that are small compared to the second derivative of velocity. As will be seen in §3.2.4, the smoothness properties of the particle-grid operator have a strong influence on this source of error.

3.2 Particle-grid operations

The velocity field is calculated from the vorticity field, both of which exist only on temporary regular grids. The vorticity field is calculated by interpolating the strengths of

the participating triangular vortex sheet elements onto the regular grid. The qualities of the interpolation scheme can have a significant effect on the resulting dynamics, to the point of exacerbating existing instabilities and creating unphysical instabilities.

In this section, several numerical interpolation schemes will be presented and their behavior analyzed. It will be shown that many popular interpolation schemes introduce unacceptable behavior in these three- dimensional Lagrangian simulations, and are not recommended for use in further research.

The vorticity field is the result of a summation over all N triangular elements according to:

$$\boldsymbol{\omega}(\boldsymbol{x}) = \frac{1}{\Delta x \Delta y \Delta z} \sum_{p=1}^{N} a_p \boldsymbol{\gamma}_p \tilde{\delta}\Big(\frac{\boldsymbol{x} - \boldsymbol{x}_p}{\Delta \boldsymbol{x}}\Big), \tag{3.32}$$

where the function $\tilde{\delta}$ is a particle-grid operator—a grid-wise approximation to a delta function. The method for velocity field creation is detailed in §3.3.1.

It is computationally efficient for a three-dimensional VIC method to use an interpolation scheme that is rectangular (no dependence on the 2-norm of the distance) and compact (requires evaluation over a finite number of grid points). The area-weighting (or Cloud-In-Cell), M4′, and Peskin functions all fit these requirements. A non-rectangular version of the Peskin function was also tested in order to demonstrate the undesirable asymmetry of the rectangular kernels. These forms are described below.

Most three-dimensional rectangular interpolation methods are simply tensor products of three one-dimensional functions according to

$$\tilde{\delta}\Big(\frac{\boldsymbol{x} - \boldsymbol{x}(s)}{\Delta \boldsymbol{x}}\Big) = \delta\Big(\frac{x - x(s)}{\Delta x}\Big)\, \delta\Big(\frac{y - y(s)}{\Delta y}\Big)\, \delta\Big(\frac{z - z(s)}{\Delta z}\Big). \tag{3.33}$$

Plots of the one-dimensional functions used in the present work appear in figure 3.8.

The performance of these interpolation methods, including conservation properties, accuracy, and smoothness, shall be elaborated upon in §3.2.4 and Chapter 4.

3.2.1 Area-weighting

The area-weighting (also Cloud-In-Cell or volume-weighting) method originates from the convolution of two top-hat functions, was used in the first vortex-in-cell method, and

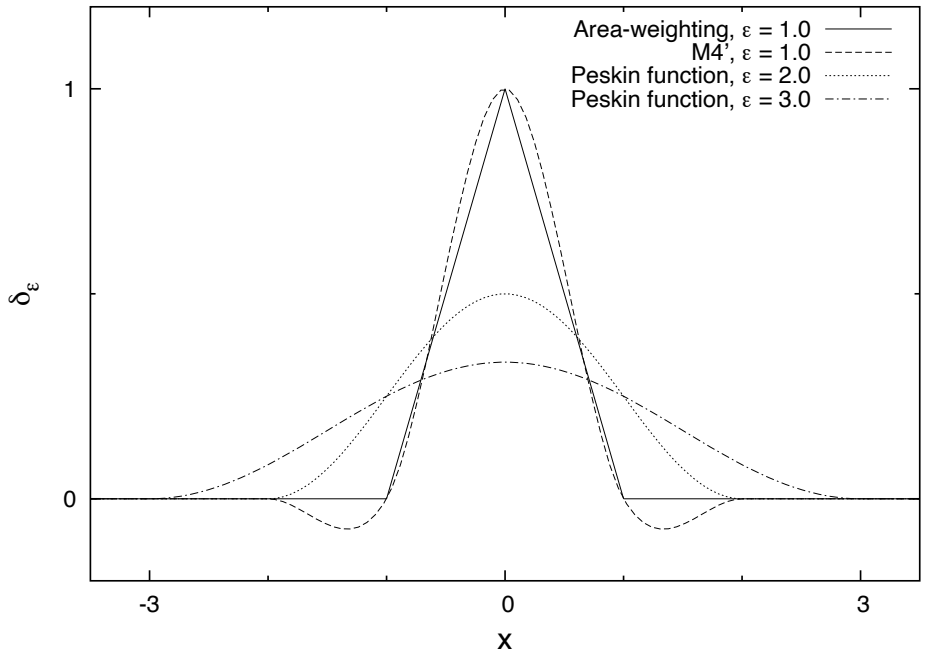

Figure 3.8: 1-Dimensional interpolation functions.

allows the fastest particle-grid operations.

$$\delta(x) = \begin{cases} 0 : |x| > 1 \\ 1 - |x| : |x| \leq 1 \end{cases} \tag{3.34}$$

Evaluation of this kernel requires knowledge of two grid points in each dimension, making 8 evaluations for each instance. While fast, this method is only first-order accurate.

3.2.2 M4′

The M4′ method [Monaghan, 1985] is used in smoothed particle hydrodynamics as well as in some previous vortex methods studies.

$$\delta(x) = \begin{cases} 0 : & |x| > 2 \\ \frac{1}{2}(2 - |x|)^2(1 - |x|) : & 1 \leq |x| \leq 2 \\ 1 - \frac{5x^2}{2} + \frac{3|x|^3}{2} : & |x| \leq 1 \end{cases} \tag{3.35}$$

The method is not strictly positive, as are the other methods. Evaluation of this kernel requires knowledge of four grid points in each dimension, making 64 evaluations for each instance. For that effort, M4′ rewards the user with third order accuracy.

3.2.3 Peskin function

The one-dimensional Peskin function [Peskin, 1977] is a smoothing kernel, has positive support, and is first-order accurate. Its main difference lies in its variable radius, which can be tuned to a desired smoothness.

$$\delta(x) = \begin{cases} \frac{1}{2\varepsilon}\left[1 + cos\left(\frac{\pi x}{\varepsilon}\right)\right] : |x| \leq \varepsilon \\ 0 : |x| > \varepsilon \end{cases} \tag{3.36}$$

In the present work, $\varepsilon = 2$ or 3, though any multiple of 0.5 at or over 1.0 will preserve the conservation properties. The kernel, thus, requires 4 or 6 data points in each dimension, totaling 64 or 216 evaluations for each object. This shall be referred to as the "rectangular Peskin" function.

A radially-symmetric version of this kernel is proposed, as it should have improved smoothness properties. The following will be referred to as the "radially-symmetric Peskin" function

$$\delta(r) = \begin{cases} \frac{1}{\varepsilon^3}\left(\frac{4\pi}{3} - \frac{8}{\pi}\right)^{-1}\left[1 + cos\left(\frac{\pi r}{\varepsilon}\right)\right] : |r| \leq \varepsilon \\ 0 : |r| > \varepsilon \end{cases} \tag{3.37}$$

$$\tilde{\delta}(\boldsymbol{x} - \boldsymbol{x}(s)) = \delta(||\boldsymbol{x} - \boldsymbol{x}(s)||) \tag{3.38}$$

A plot of this curve appears in figure 3.9 for two values of ε. Unlike the rectangular Peskin function, this kernel does not guarantee scalar conservation (making it technically zero-order accurate), though it can be used with any ε. As with the rectangular Peskin function, this kernel is typically used with $\varepsilon > 1$. See figure 3.10 for the performance of this kernel with respect to ε. The kernel radius ε is typically set to the same values as the rectangular version, or $\varepsilon = 1.5, 2.0,$ or 3.0.

3.2.4 Kernel comparisons

There exist in the literature several comparisons of these and other kernels for two-dimensional vortex methods. Most kernels exhibit good smoothness and accuracy for two-dimensional vortex methods, but the absence of any stretching term in two-dimensional flow means that rectangular kernels' asymmetry is never exacerbated. In this section, then, a fully three-dimensional problem will be solved using the interpolation kernels described above and the resulting behavior quantitatively analyzed for accuracy and smoothness. The eight variants of the four interpolation kernels are: area weighting, M4′, rectangular Peskin function with $\varepsilon = [1.5, 2, 3]$, and radially-symmetric Peskin function with $\varepsilon = [1.5, 2, 3]$.

To study the influence of the interpolation kernel, full dynamic simulations using methods from coming sections of the current chapter were run. In an effort to be complete, the simulation details will nonetheless appear.

3.2.5 Accuracy

The definition of the order of accuracy of an interpolation method is based on the highest order polynomial that the method is able to exactly interpolate. A first-order accurate interpolation method, such as the area-weighting or Peskin function, is able to accurately interpolate an arbitrary linear polynomial, but not an arbitrary quadratic.

Particle-grid operators are most frequently used in vortex methods to interpolate an irregular collection of particles onto a regular grid in the form a vorticity field, such that the old particles can be replaced with a regularly-spaced collection of new particles. This same operation occurs in the present vortex method, except that the interim vorticity field is merely used to create the velocity field, and is not used to regrid the computational elements. As such, the accuracy of the interpolation method will be determined by comparing functions of the interpolated vorticity field.

The problem set-up for this test is that of a unit-radius cylinder centered at the origin and aligned parallel to the z-axis. It is composed of a 47 by 256 matrix of triangle pairs. These elements are initiated with a unit-magnitude vortex sheet strength oriented parallel to the axis of the cylinder. The computational volume covers $[-2:2], [-2:2], [0:1]$, is

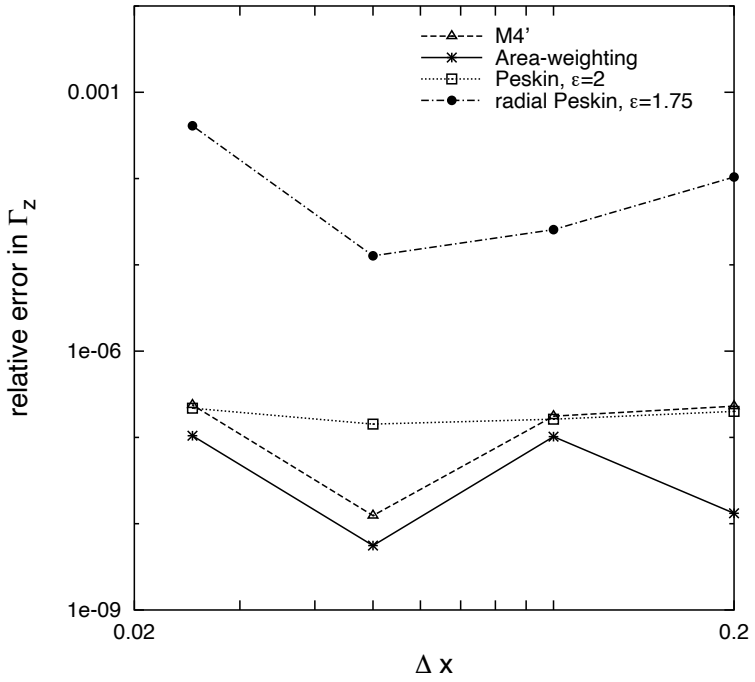

Figure 3.11: Relative error of z-component of circulation for a periodic cylindrical vortex sheet; M4′, area-weighting, Peskin, and radially-symmetric Peskin kernels.

periodic in the z-axis and has wall boundaries on the x- and y-axes. The grid resolution takes on values of $\Delta x = \frac{1}{5}, \frac{1}{10}, \frac{1}{20}$, and $\frac{1}{40}$. A separate simulation with the M4′ kernel at $\Delta x = \frac{1}{80}$ is the datum by which the errors below are computed. Only one velocity evaluation is completed.

As detailed in §4 many invariants of inviscid, incompressible flowfields can be calculated from the vorticity field alone, among these are circulation, linear impulse, and angular impulse. For the problem posed, many of these values are theoretically zero, and the simulations return values for these invariants that are within 10^{-15} of zero. The z-circulation (Γ_z), x- and y-impulses ($I_x = -I_y$), and angular impulses ($A_x = A_y \neq A_z$) are non-zero, so those values are calculated from the simulation results and compared in the following figures. Figure 3.11 shows Γ_z vs. Δx for four different interpolants, and figures 3.12 and 3.13 show the same for I_x and A_z, respectively. Figure 3.11 shows that the zeroth-order accurate radially-symmetric Peskin function follows theory and does not conserve circulation as well as the other higher-order interpolants. Surprisingly, all interpolants except for the radially-symmetric Peskin function conserve linear impulse

(figure 3.12), a quantity that should only be conserved by 2nd or higher order interpolating functions. As predicted, though, the angular impulse (figure 3.13) is conserved only the third-order M4′ interpolator. No explanation can currently be offered for the increase in error for the $\Delta x = \frac{1}{40}$ case involving the radially-symmetric Peskin function.

Another comparison of kernel accuracy appears in §3.7.2.

3.2.6 Smoothness

The smoothness of an interpolating kernel is related to the number of continuous derivatives of its function. By this measure, both the Peskin function and the M4′ function are equally smooth, having discontinuous second derivatives, and they are more smooth than the area-weighting function, which has discontinuous first derivatives. While this is an appropriate measure of smoothness for one-dimensional functions, the following results will show that grid-based interpolation with these functions does not lead to representatively "smooth" results for three-dimensional applications. No other work has been found that compares particle-grid operators for three-dimensional Vortex-In-Cell calculations.

In a VIC method, the smoothness of an interpolant manifests itself in the velocity field because the velocity field is the result of an operation on the derivatives of the vorticity field; see equation (3.3) and §3.3 for details. A non-smooth velocity field causes unrealistic distortions of the mesh, which summarily feed back into the vorticity field for the next time step. Thus, the rate of growth of these distortions can be a measure of the smoothness of the interpolant.

The case that will be used to study the smoothness of the interpolation kernel will be that of a periodic, cylindrical vortex sheet supporting only an axial velocity jump. This is equivalent to an infinitely-long cylinder of fluid moving axially in stagnant fluid. A variant of this test allows sinusoidal radial perturbations [Brady *et al*., 1998] the evolution of which is a Kelvin-Helmholtz roll-up in the circumferential plane. When the sheet is not initially perturbed, the asymmetries in each particle-grid operator will cause unique distortions in the evolving vortex sheet. These distortions would not appear if the velocity field were to be calculated with a direct summation method, but may appear with a treecode approach [see Winckelmans *et al*., 1996, figure 4].

A unit-radius cylinder composed of 8704 triangular elements is aligned with the z-axis. This cylindrical sheet supports a unit-magnitude vortex sheet strength that is oriented purely in the circumferential direction. The exact solution to this flow has the velocity within the cylinder as $\boldsymbol{u} = \hat{k}$ and outside the cylinder $\boldsymbol{u} = 0$. Thus, the velocity of the mesh elements should be $\boldsymbol{u} = \frac{1}{2}\hat{k}$. The computational domain covers [-2:2][-2:2][0:0.4], is periodic in the z-axis, and has slip-wall boundary conditions on the x- and y-boundaries. In order to perform a fair comparison, the regularization length scale δ for all tests is held constant at 0.1. This requires a higher-resolution temporary grid for the Peskin kernels and commensurately longer run times. The extra computational costs will be accounted for in the analysis.

The lack of smoothness in the tested interpolation kernels manifests as a lack of smoothness in the assembled vorticity field, and thus in the z-velocity field, also. The z-velocity of a ring of nodes is illustrated in figure 3.14 for the area-weighting interpolator. As a result of uneven vertical velocities around the circumference of the cylinder, the nodes of an element will move at different rates. This rotates the element, which realigns its total vorticity vector out of the x-y plane. Finally, the ω_z that is created by the rotation causes the perimeter to deform in the x-y plane. Slices of the cylinder for the eight tested interpolation kernels appear in figure 3.15. It is not surprising that the kernels with the greatest order of accuracy exhibit the most deformation, and vice versa. It is also notable that the amount of deformation is unrelated to the true "smoothness" of the interpolating function.

To quantify this distortion, the relative error of the perimeter of an x-y cross-section is calculated. This is easily obtained from the total interface area and the cylinder length. The relative error of the perimeter for the eight simulations appears in figure 3.16. The error for the Peskin kernels increases more slowly than for the area-weighting or M4′ kernels, even though the support for the smaller-radius Peskin kernels ($\varepsilon \leq 2$) is equal to or smaller than that of the M4′ kernel. Figure 3.17 contains a graph of the computational cost per element for each of the eight kernels. As expected, the area-weighting kernel requires the least computational resources, and the large-radius Peskin functions require the most. The most useful measure of the balance of smoothness and execution time is the product of the perimeter error and the computational cost per step per element. This will

boundary conditions to be solvable.

The Poisson equation solver used for the present simulations is called HW3CRT and is from the numerical package called Fishpack [Swarztrauber and Sweet, 1975]. HW3CRT uses a seven-point kernel to solve the finite difference approximation to the Poisson equation in Cartesian coordinates. The solver is called once for each component of velocity, each time with a unique set of right-hand-side vector and boundary conditions, described below.

3.3.1 Vorticity field creation

The right-hand-side of the matrix equation for velocity is the negative curl of vorticity. This must be determined for every interior and boundary node in the grid.

The vorticity field is created by first splitting each triangular element into a number of equal-area subelements and then adding the total vorticity of each subelement to a regular grid using a consistent particle-grid operator. The reason that elements are split into subelements is that elements may span more than one grid cell; thus to insure smoothness in the resulting vorticity field those elements' total vorticity must not be lumped all in one place. M Subsections are created by partitioning each edge of the triangular element into $\sqrt{M}$ segments and connecting the segments' nodes across the triangle, as seen in figure 3.19. The vorticity field is created by a summation over the M subsections of N triangular elements according to

$$\boldsymbol{\omega}(\boldsymbol{x}) = \frac{1}{\Delta x^3} \sum_{i=1}^{N} \frac{a_i \boldsymbol{\gamma}_i}{M} \sum_{j=1}^{M} \tilde{\delta}\Big(\frac{\boldsymbol{x} - \boldsymbol{x}_{i,j}}{\Delta \boldsymbol{x}}\Big), \tag{3.40}$$

where Δx is the edge length of a grid cell, a_i is the element area, $\tilde{\delta}$ is a particle-grid operator (see §3.2), and $\boldsymbol{x}_{i,j}$ is the center of the given sub-triangle partition. For computational efficiency, the above equation for $\boldsymbol{\omega}$ is only expanded for grid nodes within the regularization distance ε of the respective particle-grid operator (see §3.2). Note that the sub-triangle center used in the above equation is not perturbed normal to the triangle, as would be done for higher-order (curved or spline) elements.

From this discretized vector field, the curl is taken using second-order, two-point,

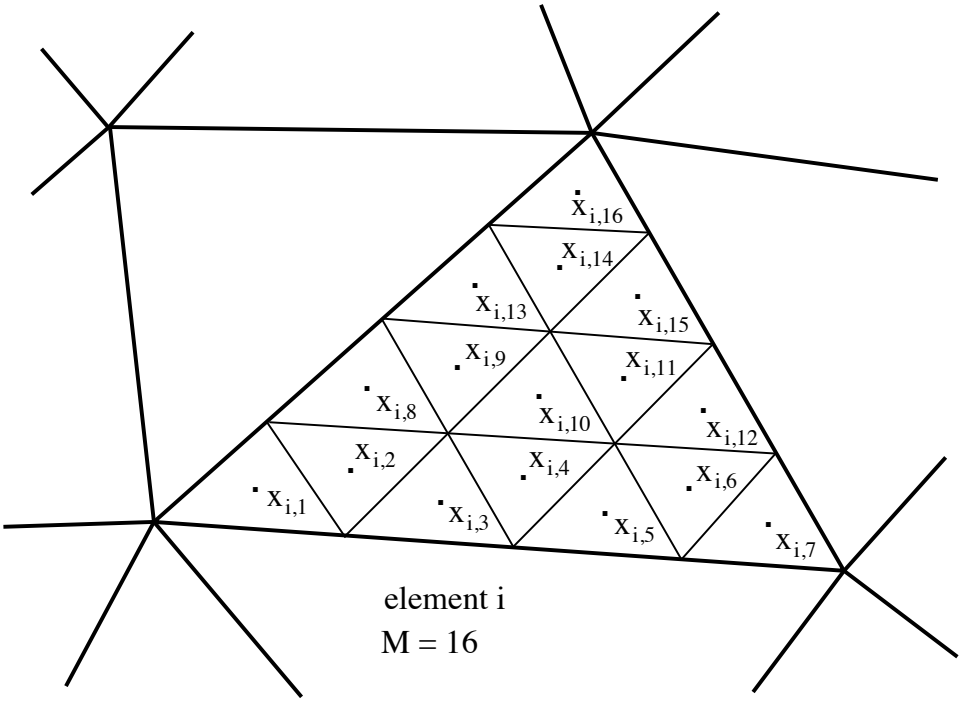

Figure 3.19: Partitioning a triangular element into 16 subelements.

first derivative operators. Derivatives normal to non-periodic boundaries use a first-order, one-sided, two-point derivative.

3.3.2 Boundary conditions

As stated previously, flow simulations fall under two categories according to their boundaries: internal and external. Internal flows require wall boundary conditions, while external flows require either free-space or periodic boundaries.

The HW3CRT solver supports Dirichlet, Neumann, or periodic boundary conditions or combinations of each. This allows easy implementation of uniform inlet, wall, periodic, or open (free) boundaries.

Several aspects of the proposed numerical method are affected by the choice of boundary condition type: discretization of triangles that approach and cross the boundary, treatment of grid vorticity for elements within the regularization distance from the boundary, and grid values on the boundary and arguments sent to the HW3CRTsolver.

3.3.3 Periodic boundaries

Many VIC methods use periodic boundary conditions to mimic free-space boundaries because they are very easy to implement in most grid Poisson solvers. True "open" boundary conditions (see §3.3.4) provide more accurate results, as they eliminate possible directional biasing created by vorticity in adjacent identical volumes.

It is easy to implement periodic boundary conditions for Lagrangian methods which use elements with no connectivity, as particles that exit one boundary appear unchanged on the opposite side of the computational domain. Because the current method uses triangles, some extra care is required. The "natural" location of the nodes of any triangular element is within the bounds of the computational domain. Thus, a triangular element that "straddles" a periodic boundary has one node on the opposite side of the computational domain as the other two. Clearly this is not a state that will allow accurate element properties to be calculated: the area, center, and normal will all be incorrect. To account for this, a special test is made for each triangular element before any operation that requires accurate values for its area, normal, or center is begun. This test loops over the coordinate directions that are defined to have periodic bounds and checks for the maximum distance between the element's corner nodes in that direction. If the maximum is greater than $2.5\Delta x$ in any of these directions, the element is considered to be straddling that direction's boundary. If an element straddles any of the three possible periodic boundaries, all nodes that are near the upper end of those directions are moved down one computational domain along that axis. See figure 3.20 for an illustration of elements that straddle one and two boundaries and their node locations before and after this test. Once the node or nodes are moved, the element is operated upon normally. After the operation is complete, the nodes are returned to their natural state.

Periodic boundaries require relatively little change to the method that creates the vorticity field from the element locations and strengths. The only difference is in accounting for the periodicity of the vorticity field. If the grid nodes in any coordinate axis are numbered $[0..N]$, with the N index representing the 0-index of the next domain, then any vorticity that is to be added to a grid node with an index less than zero ($i_{node} < 0$) is instead added without modification to the grid node with index $i_{node} \leftarrow i_{node} + N$. Likewise, any vorticity that is to be added to a grid node with an index greater than $N - 1$

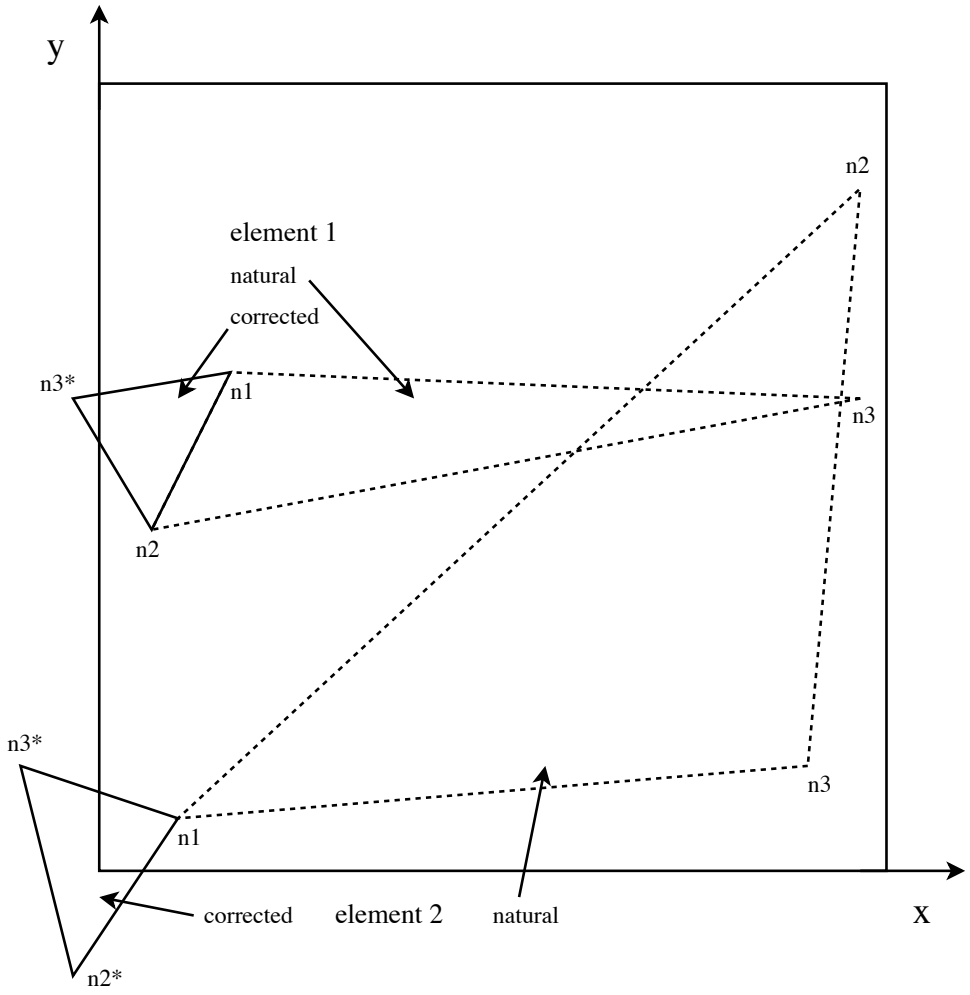

Figure 3.20: Treatment of elements that straddle periodic boundaries, depicting natural (as stored in data structure) and corrected (as used in calculations) node positions and element outlines.

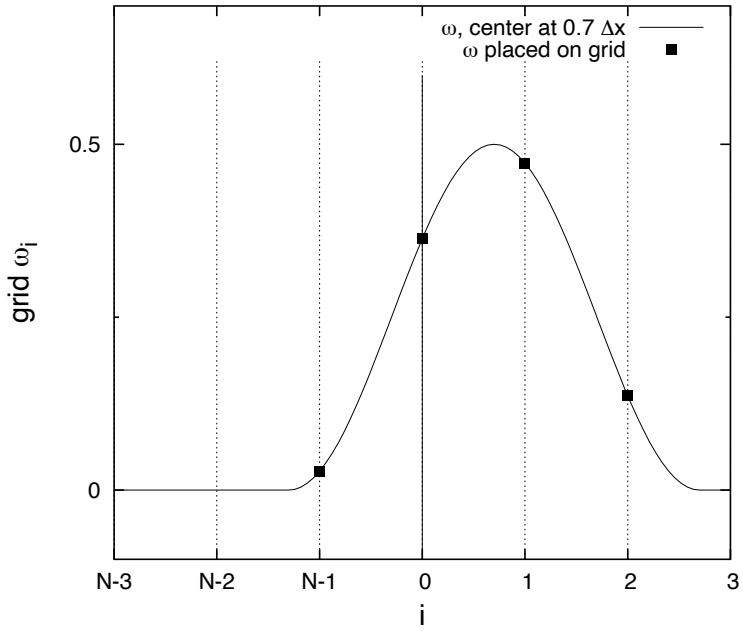

Figure 3.21: Interpolating components of vorticity onto a periodic mesh using the Peskin function with $\varepsilon = 2$.

($i_{node} > N - 1$) is instead added to the grid node with index $i_{node} \leftarrow i_{node} - N$. See figure 3.21 for a graphic demonstration. After all elements have been added to the vorticity field, the vorticity at 0-indexed grid nodes is copied to the nodes with index N for consistency.

The HW3CRT solver requires no other data preparation save the passed-in flag indicating the existence of periodic boundary conditions.

3.3.4 Open boundaries

Many vortex methods test cases require the use of free-space, or "open," boundary conditions. Applying this sort of boundary condition is somewhat difficult in a PM (particle-mesh) method. This is because a grid solution of the Poisson equation requires foreknowledge of either the value of the solution at the boundary (Dirichlet boundary condition) or its derivative (Neumann boundary condition).

A simple solution exists if no vorticity is near any of the open boundaries. In this case, Dirichlet conditions for the velocity on the boundaries are used. The unknown velocities are determined at each boundary node by directly summing the Biot-Savart integral over

the interior vorticity field. Obviously, the effort for this method scales as $\mathcal{O}(M^{5/3})$, where M is the number of grid cells. To speed the calculation, the integration is performed over every fourth boundary grid node and over only the internal cells that contain non-zero vorticity. The velocity on the remaining boundary grid nodes is interpolated using second-order finite difference formulas. Further speedup could be obtained using FMM or a treecode method.

Calculating the kinetic energy for a flow with open boundaries is non-trivial in the proposed method because the velocity is only solved for cells within the computational domain. All other flow invariants can be calculated directly from the vorticity field, see Chapter 4 for formulae. To determine the kinetic energy for the entire free-space problem, the divergence theorem is invoked. Assuming that only potential flow exists outside the domain, that the velocity approaches zero at infinity, and that a scalar velocity potential can be calculated on the boundary, the divergence theorem can be rewritten to determine the value of the kinetic energy due to all flow outside of the computational boundary.

$$E = \frac{1}{2}\Big[\int_{\mathcal{V}} |\boldsymbol{u}|^2 \, d\boldsymbol{x} + \int_{\mathcal{S}} \phi \, (\boldsymbol{u} \cdot \boldsymbol{n}) \, dA\Big] \tag{3.41}$$

To evaluate this, a scalar streamfunction ϕ must be computed on the domain boundary, and a surface integration carried out. Most simulations in the present work have open boundaries and use this method to calculate the total kinetic energy.

Another complication created by the use of open boundaries is the treatment of computational elements that enter or exit across the boundary. The simulations presented in this work are designed to avoid this situation, but it will be addressed in future work.

3.3.5 Wall boundaries

As the proposed method has been designed to simulate high-Reynolds number flow, any vorticity created at a wall boundary is assumed to stay within a small distance of the boundary. Thus, wall boundaries in the proposed method can be treated as slip-wall boundaries.

Vortex sheets can interact with a wall in one of two ways: they can begin with an edge attached to a wall, or they can approach a wall via convection. Only when a vortex

sheet begins with an edge on a wall does it need special treatment. In those cases, the edge must not leave the wall, so a special test is performed during the convection step that insures that any node within $\epsilon = 10^{-5}$ of a wall boundary stays within ϵ of the boundary.

The application of equation (3.40) to create the vorticity field requires special care in the presence of wall boundaries because of the presence of image vorticity. When the center of an element (or subelement) is within the regularization distance of a wall boundary, the strength of the element (or subelement) is modified if the grid node lands directly on or beyond a wall. If the grid nodes in any coordinate axis are numbered $[0..N]$, with the 0 and N indexes representing nodes on opposite walls, then any vorticity that is to be added to a grid node on a wall ($i_{node} = 0$ or $i_{node} = N$) will have its wall-normal component doubled and its wall-tangential components set to zero before being added. Similarly, any vorticity that is to be added to a grid node that is outside of the computational domain ($i_{node} < 0$ or $i_{node} > N$) is instead added to the reflected node ($i_{node} \leftarrow -i_{node}$ or $i_{node} \leftarrow 2N - i_{node}$, respectively) with its normal component set to zero and its tangential component multiplied by -1. Only in this way are near-wall vortex lines smoothed as if they and their reflections existed. See figure 3.22 for a graphic demonstration.

Slip-wall boundary conditions are applied by assigning the normal velocity and the wall-normal derivative of the tangential velocity at the boundary to zero. Additionally, the HW3CRT solver is sent flags indicating these conditions.

Simulations such as circular jets and mixing layers require special boundary conditions. For these situations, the normal velocity component is modified accordingly, and shedding edges are manually inserted. See §3.6 for details.

3.4 Splitting elements to accommodate extensional strain

It has been shown that the proposed vorticity discretization technique has ideal conservation properties under planar strain, even if the nodes of the triangles become separated beyond the grid or regularization scale (see §3.1). In real flows, however, a vortex sheet will be bent and contorted, and without a means to increase the resolution of the mesh, accuracy will be lost. In a two-dimensional front-tracking simulation, inserting

points while maintaining sheet connectivity is a simple matter of identifying long segments and splitting them in two. Increasing the resolution of a three dimensional mesh requires a similar method, though the individual steps require more effort.

The approach employed herein to maintain the resolution of connected triangle meshes under Lagrangian motion is to insert nodes in the middle of selected element edges and then to reset the strength and connectivity of the participating triangular elements. At every time step, the following sequence of actions is performed as many times as are necessary for all flagged edges to be split:

1. Reset all node and element flags, recompute all surface normal vectors.
2. All elements whose longest edge length is greater than a fixed threshold, (Δ_{split}), usually close to the regularization length ($\delta \simeq \Delta x$), are placed into a list.
3. The list is sorted by edge length, with the longest edges first.
4. For each edge in the list:
   (a) All elements that share that edge are placed into a second list.
   (b) A new node is created at the geometric midpoint of the edge.
   (c) For each element in the second list:
      i. Two new elements are created from the original element.
      ii. The nodes of the new elements are set.
      iii. The vortex sheet strengths of the new elements are set to be equal to the vortex sheet strength of the original element according to the algorithm in §3.1.
      iv. The original element is removed.
   (d) If a more advanced midpoint-finding method is used, the new node is only now moved to its final location (see §3.4.3 for details).

This is illustrated in figure 3.23. The output of this algorithm is a valid connected set of elements that accurately replaces the original set, but with no element-poor regions. Note that this algorithm is valid in instances where more than two elements share one edge (a triple junction), as would result from the full node merging procedure described in §3.5.

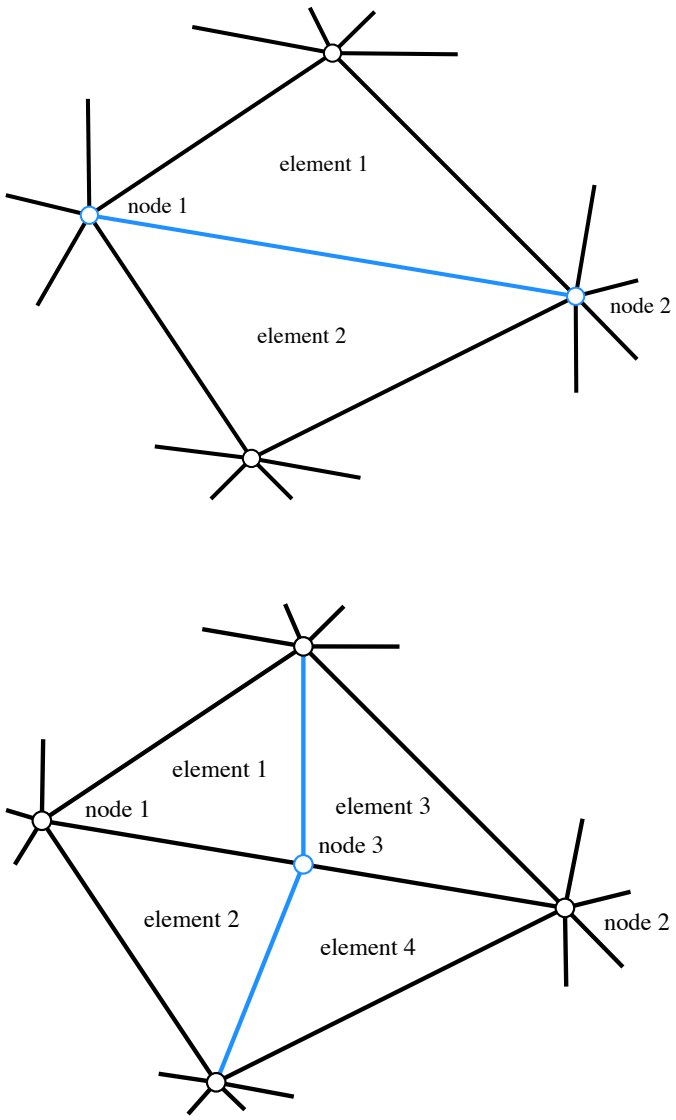

Figure 3.23: Graphical example of element remeshing accomplished by long edge splitting.

3.4.1 Midpoint selection

Once an edge has been flagged for splitting, and any other splitting criteria are satisfied, the location of the new node must be calculated. The most simple method is to use the average of the locations of the edge's two end nodes. This is the geometric midpoint method. The advantage of the geometric midpoint method is that it exactly conserves the volume enclosed by the mesh during the edge splitting operation. This may seem like a desirable feature, but in reality it is less able to account for volume loss due to curvature. More advanced methods are capable of accounting for the volume loss by better approximating the real, curved surface. Two improved methods will be described herein, one which fits a cubic spline to the edge, and another which fits a cylinder. Other methods typically work by introducing a free variable (such as the distance along an approximate surface normal) and constructing an optimization problem to determine its value.

One method that can be used to better approximate a curved surface involves constructing a cubic spline between the endpoints of the edge. The function values are the node locations, and the first derivatives are constructed from the tangential projection operator. The method is as follows.

1. Calculate the surface normal at each end node ($\hat{\boldsymbol{n}}_j, j = 1, 2$) as the mean of the normals of the N_j elements adjacent to that node weighted by the angle (ϕ_e) subtended by the two element legs containing that node.

$$\hat{\boldsymbol{n}}_j = \text{norm}\left(\sum_{e=1}^{N_j} \hat{\boldsymbol{n}}_e \phi_{j,e} \right) \tag{3.42}$$

2. Compute the tangential projection matrix $\boldsymbol{P}$ for each node.

$$\boldsymbol{P}_j = \boldsymbol{I} - \hat{\boldsymbol{n}}_j \hat{\boldsymbol{n}}_j^T \tag{3.43}$$

3. For each node, find the product of $\boldsymbol{P}$ and the vector between the nodes.

$$\boldsymbol{f}_j = \boldsymbol{x}_j \tag{3.44}$$

$$\boldsymbol{f}'_j = \boldsymbol{P}_j(\boldsymbol{f}_2 - \boldsymbol{f}_1) \tag{3.45}$$

4. For each coordinate direction ($i = 1, 2, 3$), compute the spline weights.

$$\begin{aligned} a_{0,i} &= f_{1,i} \\ a_{1,i} &= f'_{1,i} \\ a_{2,i} &= 3\left(f_{2,i} - f_{1,i}\right) - \left(f'_{2,i} + 2f'_{1,i}\right) \\ a_{3,i} &= 2\left(f_{1,i} - f_{2,i}\right) + \left(f'_{2,i} + f'_{1,i}\right) \end{aligned} \tag{3.46}$$

5. Compute the location of the midpoint of the spline.

$$x_{3,i} = a_{0,i} + \frac{a_{1,i}}{2} + \frac{a_{2,i}}{2^2} + \frac{a_{3,i}}{2^3} \tag{3.47}$$

Another method fits a cylinder to the endpoints of the long edge using surface normal information at the edge's end nodes. The procedure for determining the midpoint using this new method is as follows:

1. Calculate the surface normal at each end node ($\hat{\boldsymbol{n}}_j, j = 1, 2$) using equation (3.42).

2. Define the axis $\bar{\boldsymbol{a}}$ of the cylinder to be parallel to the cross-product of the endpoints' normals.

$$\boldsymbol{a} = \frac{\hat{\boldsymbol{n}}_1 \times \hat{\boldsymbol{n}}_2}{||\hat{\boldsymbol{n}}_1 \times \hat{\boldsymbol{n}}_2||} \tag{3.48}$$

3. Set θ equal to the angle between the two normals.

$$\theta = \frac{1}{2}\cos^{-1}\left(\hat{\boldsymbol{n}}_1 \cdot \hat{\boldsymbol{n}}_2\right) \tag{3.49}$$

4. Find the distance between the two end nodes, projected into the plane normal to $\boldsymbol{a}$.

$$\boldsymbol{dl} = \boldsymbol{x}_2 - \boldsymbol{x}_1 \tag{3.50}$$

$$l = \sqrt{||\boldsymbol{dl}||^2 - (\boldsymbol{dl} \cdot \boldsymbol{a})^2} \tag{3.51}$$

5. Find the ideal distance from that plane to the new point.

$$d = \frac{l}{2}\left(\frac{1}{\sin\theta} - \frac{1}{\tan\theta}\right) \tag{3.52}$$

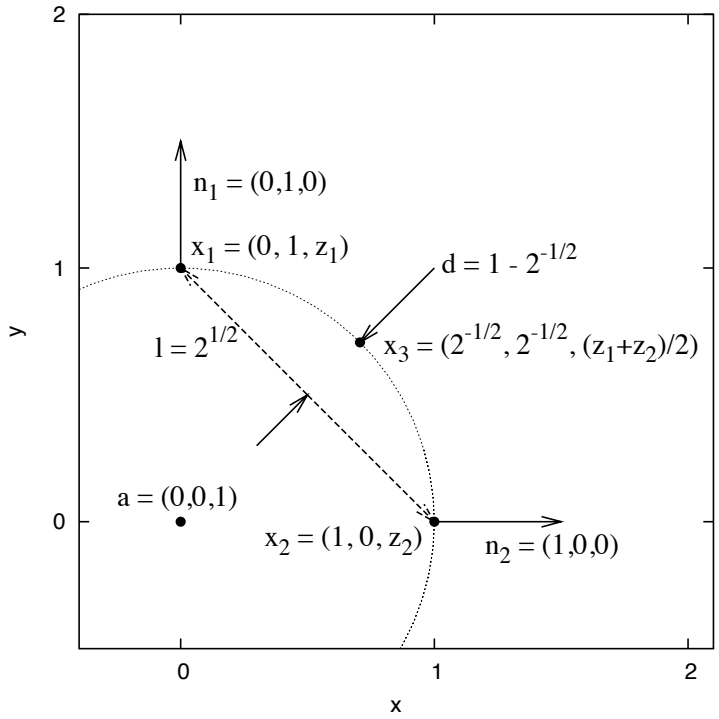

Figure 3.24: Cylindrical midpoint method example: split edge lies between nodes $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, new node is placed at $\boldsymbol{x}_3$, coordinates chosen for clarity.

6. If the two normals are concave (point to the interior of the cylinder) then flip the sign of d.

7. Place the new node d along the vector of the average normal from the true midpoint of the edge.

$$\boldsymbol{x}_3 = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2}{2} + d\,\frac{\hat{\boldsymbol{n}}_1 + \hat{\boldsymbol{n}}_2}{||\hat{\boldsymbol{n}}_1 + \hat{\boldsymbol{n}}_2||} \tag{3.53}$$

With the exception of the case of endpoints with parallel normals, this procedure uniquely defines a cylinder that contains the two end nodes and whose surface is perpendicular to the two points' normals. A didactic case is illustrated in figure 3.24.

3.4.2 Performance of midpoint selection methods

Due to the finite resolution of the elements composing the mesh, its enclosed volume will change during convection. Thus, maintaining the mesh resolution by splitting edges will alleviate some, but not all of the volume change. The following test will quantify

these losses for the various splitting and midpoint-selecting schemes presented above, but will not address the circulation distribution method.

An artificial velocity field is imposed on an initially spherical volume. The velocity field is defined analytically as the influence of three infinitely-long desingularized vortex filaments, each pointing in a primary coordinate direction, offset from the origin along the other two axes by $\Delta = 0.2$, with strength 4π, and desingularization parameter $\delta = 0.2$; or

$$\begin{aligned} u &= \frac{-(y-\Delta)}{\left((y-\Delta)^2+(x-\Delta)^2+\delta^2\right)^{\frac{3}{2}}} + \frac{z+\Delta}{\left((z+\Delta)^2+(x+\Delta)^2+\delta^2\right)^{\frac{3}{2}}} \\ v &= \frac{-(z-\Delta)}{\left((z-\Delta)^2+(y+\Delta)^2+\delta^2\right)^{\frac{3}{2}}} + \frac{x-\Delta}{\left((x-\Delta)^2+(y-\Delta)^2+\delta^2\right)^{\frac{3}{2}}} \\ w &= \frac{-(x+\Delta)}{\left((x+\Delta)^2+(z+\Delta)^2+\delta^2\right)^{\frac{3}{2}}} + \frac{y+\Delta}{\left((y+\Delta)^2+(z-\Delta)^2+\delta^2\right)^{\frac{3}{2}}} \end{aligned} \tag{3.54}$$

This velocity field is defined on a grid with $\Delta x = 0.05$, and time is advanced using a second-order Runge-Kutta integrator with uniform time step $\Delta t = 10^{-3}$ (corresponding to a Courant number of 0.273). The sphere has a radius of 0.4 and is located at the origin. Its geometry is initially composed of 5120 triangular elements with an average edge length of 0.0263. The threshold length for splitting an edge is $\Delta_{split} = 0.04$.

Figure 3.25 depicts the surface of the sphere at various simulation times. Note the extreme deformation experienced by the surface even at interim times. The normalized error in enclosed volume for the cases with no splitting, splitting with geometric midpoints, splitting with cylindrical-projection midpoints, and splitting with spline-fit midpoints appears in figure 3.26. As mentioned above, the cylindrical and spline midpoint methods provide a better approximation to the smooth curved surface, and for the test case, maintain the initial volume with an order of magnitude less error than even the geometric midpoint method (which itself is an order of magnitude better than not remeshing).

3.4.3 Assignment of circulation to new elements

The total vorticity assigned to each element (in the form of edge circulations) during the edge-splitting procedure is computed using the algorithm presented in §3.1. This procedure is straightforward for the geometric midpoint-finding method, in which the two child elements are co-planar with their parent. In that case, the total vorticity of the child elements identically equals the total vorticity of the parent.

$$\begin{aligned} \gamma_{parent} &= \gamma_{child1} = \gamma_{child2} \\ a_{parent} &= a_{child1} + a_{child2} \\ (a\gamma)_{parent} &= (a\gamma)_{child1} + (a\gamma)_{child2} \end{aligned} \tag{3.55}$$

Other midpoint-finding methods do not share this simplicity. Owing to the perturbed location of the new node, the new child elements are not coplanar, thus

$$a_{parent} \neq a_{child1} + a_{child2} \tag{3.56}$$

and

$$\hat{\boldsymbol{n}}_{parent} \neq \hat{\boldsymbol{n}}_{child1} \neq \hat{\boldsymbol{n}}_{child2}, \tag{3.57}$$

which disqualify immediate application of the circulation assignment scheme from §3.1. To rectify this, and to create a method that conserves circulation despite the splitting of elements into non-coplanar children, the new node is created at the *geometric* midpoint before applying the circulation assignment scheme to the child elements. Only after the proper circulation is assigned to the child elements is the new node moved to its perturbed position. This final movement merely stretches the vortex lines without changing their circulations. It does, however, change the total vorticity of the new elements to that which would be required by the newly-curved surface.

The simulations in the next two sections will test the procedure that assigns the vortex sheet strength to the elements formed in the edge-splitting procedure defined above.

3.4.4 Validation of circulation assignment - planar

To determine the ability of the remeshing method to accurately maintain the original vortex sheet's qualities, the validation tests used in §3.1.2 were repeated with edge splitting enabled. Pertinent results from that section will be repeated here, but the details of the problem setup will not.

A smooth vortex sheet with specified initial vortex sheet strength spans the xy-plane of a periodic computational volume. An artificial velocity field (3.1) is imposed for the duration of the simulation. The sheet experiences only in-plane convection, resulting in compressive and extensional strain in different areas of the sheet. Figure 3.27 shows the computational elements at the end of the simulation for the cases without and with edge splitting. The following two exercises test the response of the discretization and remeshing algorithms by examining the vorticity field at a finite time.

The first test has $\boldsymbol{\gamma}(t = 0) = 1.0\hat{j}$, so the imposed velocity will convect the vortex sheet parallel to the vorticity. The resulting vorticity field magnitude on the vortex sheet appears in figure 3.28 for the cases without and with remeshing by edge splitting. The solution of the vortex sheet strength equation (3.13) shows that the elements should convect without change in sheet strength, thus the resulting vorticity field is expected to equal $\omega_y = \gamma_y/\Delta x = 20$ on the sheet. Clearly, the edge splitting method does not reduce the accuracy inherent in the discretization method. In its effort to eliminate large triangles, it actually increases the accuracy for this particular test.

The second test begins with a sheet with strength $\boldsymbol{\gamma}(t = 0) = 1.0\hat{i}$, which represents strain acting in a direction transverse to the vorticity. Thus, the vorticity should decline exponentially in the areas of extensional strain, and increase exponentially in the areas of compressional strain. The current test was run to $t = 1.0$ and results for vorticity on the sheet appear in figures 3.29 and 3.30. The results show that the minimum ω_x follows the analytic solution very closely, even out to $t = 1.0$, when the total elongation experienced at $y = 0.25, z = 0.5$ is 25. The maximum ω_x, which occurs on the line of peak compressional strain, falls far below the analytic value. This is due to the inability of the finite regularization to resolve small details, and is a key characteristic of regularized computational methods.

It should be noted that the scheme for identifying "straddling" elements (see §3.3.3)

prevented the tests in §3.1.2 from progressing farther than $t = 0.26$, but enabling the edge splitting routine allows these simulations to continue indefinitely.

3.4.5 Validation of circulation assignment - dynamic

As mentioned previously, the circulation assignment scheme is simply applied when the child elements of a split operation are co-planar. This section will test the ability of the scheme to conserve circulation in a large-scale dynamic test.

A unit-radius sphere, centered at the origin, is initially composed of 5120 roughly equal-area triangular elements. The initial vortex sheet strength on the sphere is that required to satisfy the no-through-flow condition on a sphere in a freestream in potential flow, namely $\boldsymbol{\gamma} = -\frac{3}{2}\, r\, (\boldsymbol{n} \times \boldsymbol{U})$. In our case, $\boldsymbol{U} = -\hat{k}$, making $\Gamma_{ring} = 3$. This problem frequently appears in the vortex methods literature [Winckelmans *et al.*, 1996; Nitsche, 1996; Nie and Baker, 1998; Pozrikidis, 2000; Nitsche, 2001b,a].

The computational domain consists of a traveling window, centered on the sheet, with fixed size $[-2:2], [-2:2], [z:z+4]$ and open boundary conditions. The VIC grid size is $\Delta x = \frac{1}{10}$ and all particle-grid operations use the area-weighting scheme, making the regularization length $\delta = \frac{1}{10}$, also. The time step is fixed at 0.05, which for $v_{max} \simeq 2.2$ corresponds to a CFL number of $\simeq 1.1$.

Edge-splitting is enabled with a length threshold of $\Delta_{split} = 0.1$ (compared to the initial mean triangle edge length of 0.0657) and separate runs are made for each of the three midpoint-finding algorithms. Of primary importance is tracking the overall vortex ring circulation and comparing it with the initial value. Any difference between the initial value and the theoretical value ($\Gamma_{ring} = 3$) is due to error in the initial discretization, which is shown in figure 3.31 to be second order in relation to the average triangle edge length.

Images of the computational mesh at various simulations times appear in figure 3.32. The conservation of vortex ring circulation, measured as

$$\Gamma_{ring} = \sum_{p=1}^{N} \frac{1}{2\pi ||\boldsymbol{x}_p||^2} \sum_{i=1}^{3} \Gamma_{p,i} \left(\Delta l_{p,i} \times \boldsymbol{x}_p \right) \cdot \hat{k}, \tag{3.58}$$

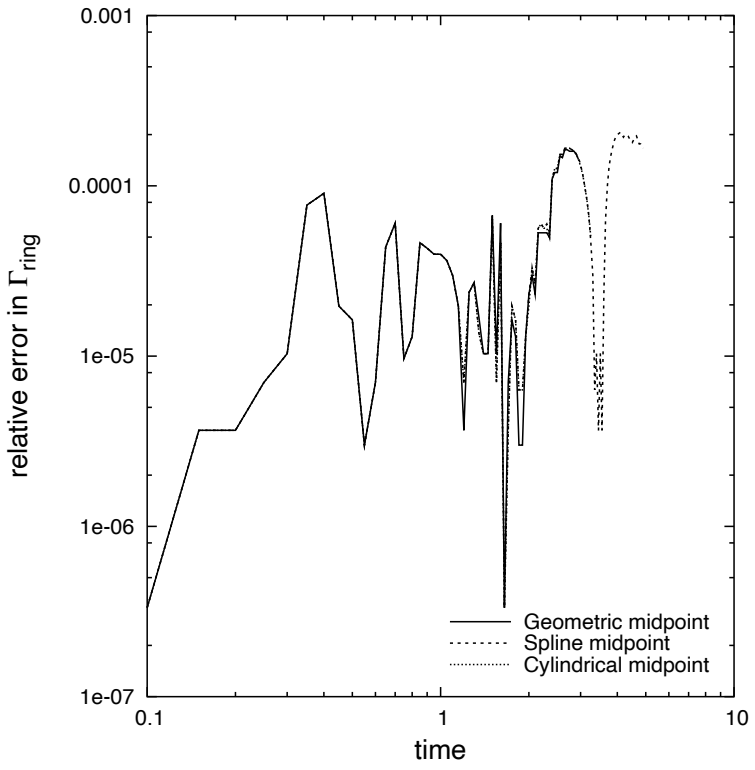

Figure 3.33: Error in total vortex ring circulation vs. initial value, geometric, spline, and cylindrical midpoint-finding methods.

where $\boldsymbol{x}_p$ is the vector from the z-axis to the center of element p, is quantified in figure 3.33 for the three midpoint-finding methods. As the figure illustrates, all three methods equivalently maintain the vortex ring circulation to within 10^{-4} of the original value. Additionally, figures 3.34 and 3.35 show that the higher-order midpoint methods conserve volume and kinetic energy better than the geometric midpoint method. Section 4 details the method for calculating the total kinetic energy for a free-space problem.

In conclusion, the design itself of the element discretization scheme allows easy application of a straightforward element splitting method so as to accurately maintain sheet connectivity and vortex sheet strength despite severe strain and remeshing in both in-sheet directions.

3.5 Merging nodes for mesh simplification

As previously noted, tracking a vortex sheet involves adding triangles in order to maintain the resolution of the sheet. Because triangles are split only when an edge is longer than a threshold, the mesh that results is usually one of great numbers of very low aspect-ratio triangles. Without a method to simplify the mesh by merging these triangles together, the duration of the simulations is severely limited.

The process of merging elements is similar to that of splitting in the sense that pairs of nodes are flagged and a new node is created between each pair. One of the two main differences between merging and splitting is that the merging algorithm *replaces* the original pair of nodes with the new node, whereas the splitting algorithm keeps all three. Thus, a split operation typically adds two elements, and a merge operation typically removes two.

The second difference between edge splitting and node merging is that the two nodes in the splitting operation are necessarily shared by one or more triangular elements. In the merging operation, they can be shared by a common element(s), or they can belong to two completely separate and disconnected sheets. These two possibilities are separated in the logic of the method: the procedure can either merge only nodes that share an element, or it can merge *any* close pair of nodes, regardless of connectivity. These operations shall be referred to as "in-sheet" and "full" merging, respectively. A sheet-merge operation is illustrated in figure 3.36.

3.5.1 Node-merging procedure

The node-merging operation is done in conjunction with the edge-splitting operation (described in §3.4), with each operation taking turns until no elements are subsequently modified. This is the complete "remesh triangles" step referred to in the flowcharts in figures A.1 and A.2. The edge-splitting operation is performed first, and with one modification to the original sequence:

1. Reset all node and element flags, recompute all surface normal vectors.

2. All elements whose longest edge length is greater than Δ_{split} are placed into a list.

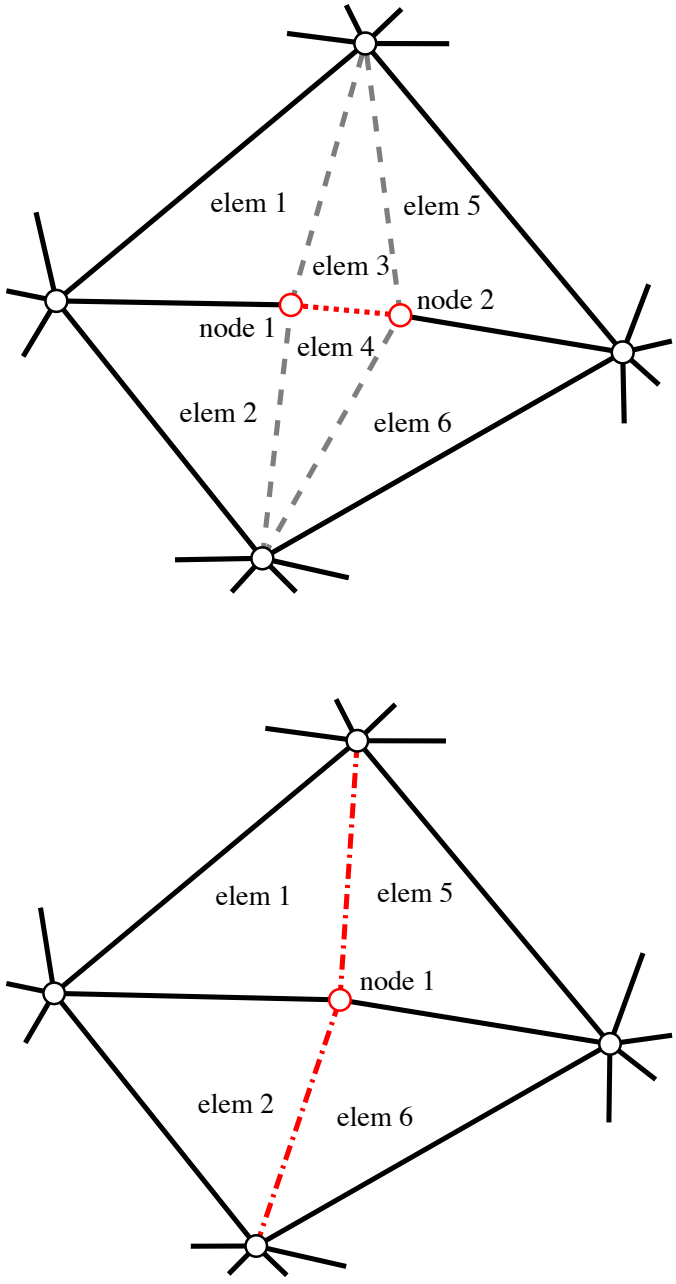

Figure 3.36: Element remeshing accomplished by merging two nodes within a connected sheet; top: original element locations, edge between nodes 1 and 2 will be collapsed to a point; bottom: final configuration, note that elements 3 and 4 have been collapsed to a line and removed.

New: Additionally, all elements whose longest edge length is $2\Delta_{merge} < l < \Delta_{split}$, and whose altitude is $a < 0.5\Delta_{merge}$ (signifying a poor aspect ratio) are placed in the same list.

3.-4. Splitting operation continues normally.

After completion of each iteration of the edge-splitting operation the following sequence of actions (called the node-merging operation) is performed:

1. Reset all node and element flags, recompute all surface normal vectors.
2. Make a list of node pairs with the potential to merge.

   Full merge: All node pairs whose Euclidean distance is less than a fixed threshold (usually a fraction of the regularization length) are placed into the list.

   In-sheet merge: All node pairs whose Euclidean distance is less than a fixed threshold *and* which share at least one common element are placed into the list.
3. Sort the list by distance, with the closest pairs first.
4. For each node pair (nodes referred to as "node 1" and "node 2") in the list:
   (a) Label the nodes "node 1" and "node 2," note that node 2 will eventually be removed and node 1 will be relocated.
   (b) Do not merge the pair if either of the nodes was moved or removed in a recent merge operation.
   (c) Compute the new location for node 1 using the geometric or spline-based midpoint-finding method, (see §3.5.2).
   (d) Create four lists:
      i. A list of elements that contain node 1 but not node 2.
      ii. A list of elements that contain node 2 but not node 1.
      iii. A list of elements that contain *both* nodes 1 and 2—these will be collapsed to a line and removed.

iv. A list of element pairs (one element from each of lists *i.* and *ii.*) that will become coincident when nodes 1 and 2 merge.

(e) Save the vortex sheet strength (γ) from each element in lists *i.* and *ii.*

(f) Save the total vorticity ($\sum a\gamma$) of all of the elements in list *iii.*

(g) Remove each element in list *iii.*

(h) Remove one element from every pair appearing list *iv.* from the simulation, including its appearance in list *i.* or *ii.*

(i) Reset the pointer in each element in list *ii.* to use node 1 instead of node 2.

(j) Delete node 2.

(k) Relocate the original node 1 to the location decided upon in step 4c.

(l) Re-apply each of the remaining elements' original vortex sheet strength back onto the element.

(m) Convert the total vorticity from step 4f into a vortex sheet strength by dividing by the total area of the remaining elements, then apply that vortex sheet strength uniformly onto the remaining elements.

By iterating the split and merge operations, any poor-quality elements created by one operation are subject to refinement and correction by the alternate operation. Experience has shown that every possible configuration of triangles will eventually arise, so methods that aim to maintain the quality of the resolution of the mesh must be robust enough to accommodate any element-group configuration.

The following sections will detail the performance of this merging algorithm.

3.5.2 Midpoint selection

Before the close node pair can be merged, a location for the new node must be determined. Because of the similarity between the splitting and merging operations, the same midpoint-finding routines may be applied. The only exception to this is that the cylindrical midpoint method cannot be used for the full-merge operation, as it assumes that the original two nodes are part of a continuous sheet. See §3.4.1 for algorithm details for the midpoint splitting methods.

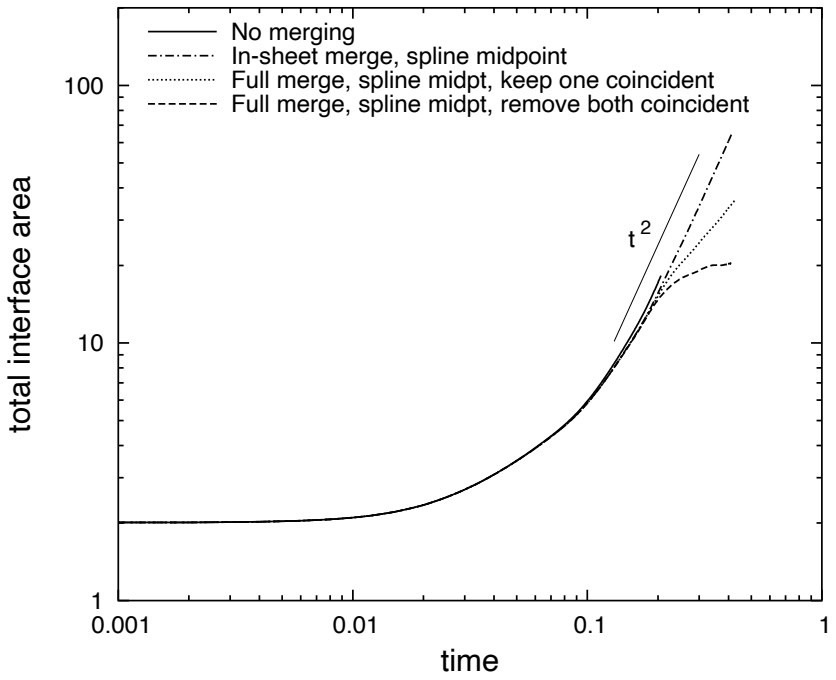

Figure 3.37: Total interfacial area vs. time for four merging types: none, in-sheet, full (remove one of each coincident pair), full (remove both of each coincident pair); all use spline midpoint method; initially spherical surface evolving under the influence of three fixed vortex lines.

3.5.3 Geometric performance

The performance of the merging procedure in regards to mesh quality and volume conservation is determined from tests identical to those in §3.4.2, in which the surface of a sphere is tracked as it deforms under the influence of three infinite desingularized vortex lines. The sphere is initially composed of 5120 triangles with a mean edge length of 0.0263, the splitting routine is enabled and maintains a maximum edge length of 0.04, and the merging routine, when used, aims to merge any node pairs within a distance of 0.01. A number of runs were conducted, varying the merge method (none, in-sheet, full) and the midpoint-finding method (geometric, spline). Each test used the spline midpoint method for the edge-splitting operation.

The capability of the merging operation to reduce the number of elements is illustrated in figures 3.37 and 3.38, which present data for the total sheet area and number of elements in the mesh. The total surface area for this particular simulation grows at a rate

proportional to the square of the simulation time ($A \sim t^2$) for the cases with no merging and with in-sheet merging. Two variants of the full-merging operation were tested: the first uses the algorithm as described in §3.5.1, the second modifies step 4h such that *both* coincident elements are removed. This can be done only for simulations without explicit vorticity-tracking, such as this one. The surface area for the cases with full-merging falls below the t^2 curve because the aim of the full-merging operation is to coalesce very thin sheets, or for the second variant, remove them. The latter case appears to have settled on a nearly-constant surface area, which is to be expected if only thick sections of the initial volume remain and the volume is to be constant throughout the simulation.

The plot in figure 3.38 shows the remarkable effect that merging has on the number of elements required to represent the sheet. The case with no merging appears to require $N_{nomerge} \sim t^{\simeq 3}$ elements, a case that clearly prevents the completion of any long-duration runs. This is because edge-splitting alone allows numerous thin triangles to be created by the combined action of tangential strain and edge splitting. The primary purpose of the merging routines is to replace these thin triangles with ones of larger area and smaller aspect ratio. The case with in-sheet merging supports an increase in element count that is roughly proportional to the increase in surface area ($N_{sheet} \sim A_{sheet} \sim t^2$). This means that the method is properly converting small triangles into triangles of a globally-ideal size. The cases with full merging similarly show the element count increasing along with the total surface area, indicating that, again, elements of consistent size are being created and maintained.

Figures 3.39 and 3.40 allow further elaboration on the ability of the merging routines to maintain element area and quality, as they present plots of the mean element area and triangle aspect ratio for the same case. As mentioned above, the in-sheet and full-merge cases exhibit nearly equal growth of surface area and element count, which results in approximately-constant element areas and aspect ratios, despite the 30-fold increase in total surface area by $t = 0.4$. Even the full-merge case, with its significant morphological differences, maintains the same performance with respect to these variables as the case with in-sheet merging. The no-merge case exhibits a power-law decrease in mean element area equivalent to $a_{nomerge} \sim t^{-1.4}$. Combined with the scaling relationship for surface area, it follows that the number of elements required by the no-merge case obeys

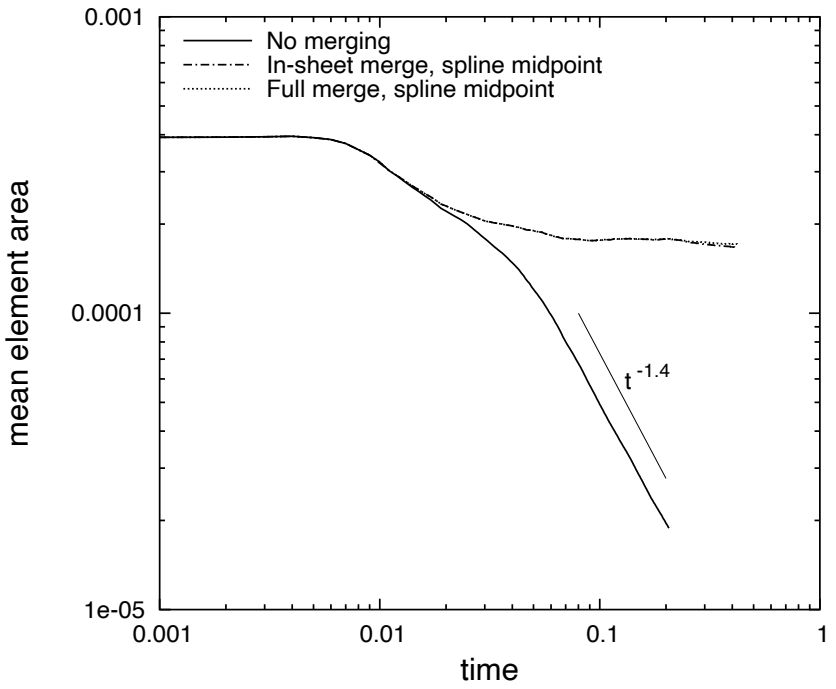

Figure 3.39: Average element area vs. time for three merging operations: none, in-sheet, full; using spline midpoint method; initially spherical surface evolving under the influence of three fixed vortex lines.

$N_{nomerge} \sim t^{3.4}$.

The enclosed volume errors are presented in figure 3.41. The case with no merging, obviously, results in the least volume error, holding at a steady value of 5×10^{-3}. For long runs, the volume error for all of the merging schemes increases as $\sim t^2$, despite relatively good performance for interim times. Though this volume error may appear poor, it is helpful to remember that the number of elements in the simulation also increases at that rate, indicating that the mean volume error per element remains constant.

Figure 3.42 illustrates the difference in the appearance of the computational mesh between three selected simulations. The no-merge and in-sheet merge cases look nearly identical, and have volume errors of 4.1×10^{-4} and 3.1×10^{-3}, respectively, though the in-sheet merge case requires one order of magnitude fewer elements. At $t = 0.2$, the full-merge case has already begun merging closely-spaced sheets, and as a result, appears less smooth. The volume error is much greater (1.7×10^{-2}), and the mesh requires only marginally fewer triangles than the case with in-sheet merging. Because thin sheets are merged together, it is expected that there will be more volume loss in the case with full

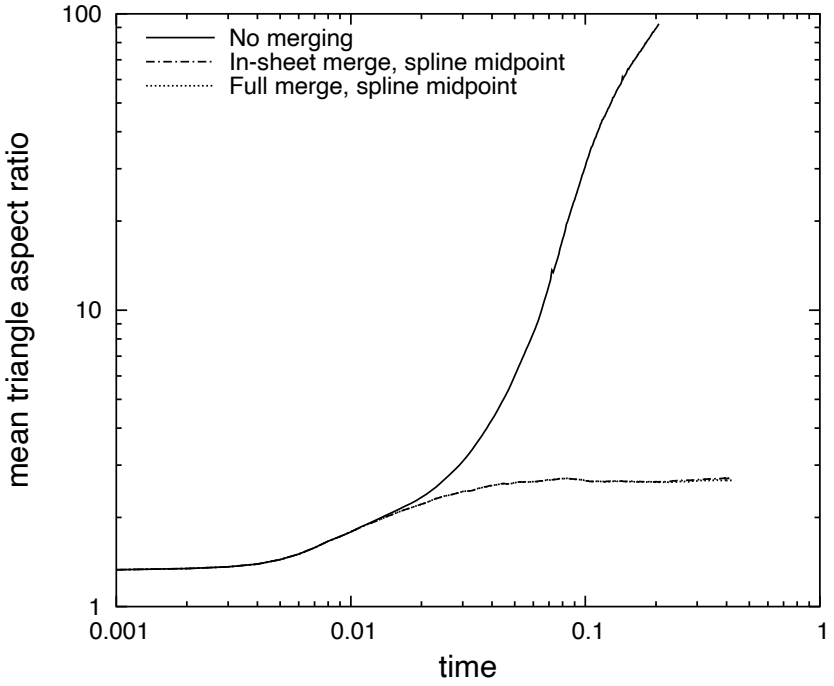

Figure 3.40: Average triangle aspect ratio vs. time for three merging operations: none, in-sheet, full; using spline midpoint method; initially spherical surface evolving under the influence of three fixed vortex lines.

merging than the case with only in-sheet merging. Figure 3.43 compares cross-sections of the mesh for in-sheet and full merging at a more advanced stage than figure 3.42 ($t = 0.4$ instead of $t = 0.2$). This figure clearly illustrates the ability of the full-merging operation to reduce the morphological complexity of the surface, at the cost of volume. The volume errors for both cases at $t = 0.4$ are 1.43% and 17.4%, respectively. When the merging routines are enabled, the essential shape of the surface is maintained, despite the drastic reduction in the number of triangular elements, and the quality of the remaining triangles is significantly improved.

3.5.4 Validation of circulation assignment - planar

Two simple tests of the merging method's ability to maintain circulation involve planar, uni-axial strain on a flat vortex sheet. These are the same two tests performed in §3.1.2 and §3.4.4 to determine the performance of the discretization and edge-splitting methods. The problem set-up is described in detail in §3.1.2, and will only be summa-

rized here.

A smooth vortex sheet with specified initial vortex sheet strength spans the xy-plane of a periodic computational volume. An artificial velocity field (3.1) is imposed for the duration of the simulation. The sheet experiences only in-plane convection, resulting in compressive and extensional strain in different areas of the sheet. Figure 3.44 shows the computational elements at the end of the simulation for the cases without and with node merging. The following two exercises test the response of the discretization and remeshing algorithms by examining the vorticity field at a finite time.

The first test has $\boldsymbol{\gamma}(t = 0) = 1.0\hat{j}$, so the imposed velocity will convect the vortex sheet parallel to the vorticity. The resulting vorticity field magnitude on the vortex sheet appears in figure 3.45 for the cases without and with remeshing by node merging. The solution of the vortex sheet strength equation (3.13) shows that the elements should convect without change in sheet strength, thus the resulting vorticity field should equal $\omega_y = \gamma_y/\Delta x = 20$ on the sheet.

The second test begins with a sheet with strength $\boldsymbol{\gamma}(t = 0) = 1.0\hat{i}$, which represents strain acting in a direction transverse to the vorticity. Thus, the vorticity is expected to decline exponentially in the areas of extensional strain, and increase exponentially in the areas of compressional strain. This test was run to $t = 1.0$ and results for vorticity on the sheet appear in figures 3.46 and 3.47. As observed in §3.4.4, the simulation follows the solution closely along the line of peak extensional strain (minimum ω_x), but lags behind the analytic solution along the line of peak compressional strain (and maximum ω_x). This error is a result of the inability of a regularized method to capture very localized phenomena–details smaller than the regularization length scale $\delta = \Delta x = 0.05$.

3.5.5 Validation of circulation assignment - dynamic

Because the merging scheme must change the local topology of the mesh, and triangular elements cannot contain normal components of total vorticity, the reassignment of total vorticity back onto the elements does not explicitly conserve total vorticity. Additionally, because this method contains no explicit viscous dissipation, there is no outlet for relieving localized errors in the reassignment. The performance of the scheme presented in §3.5.1 and §3.1 under these limitations shall be addressed in this section.

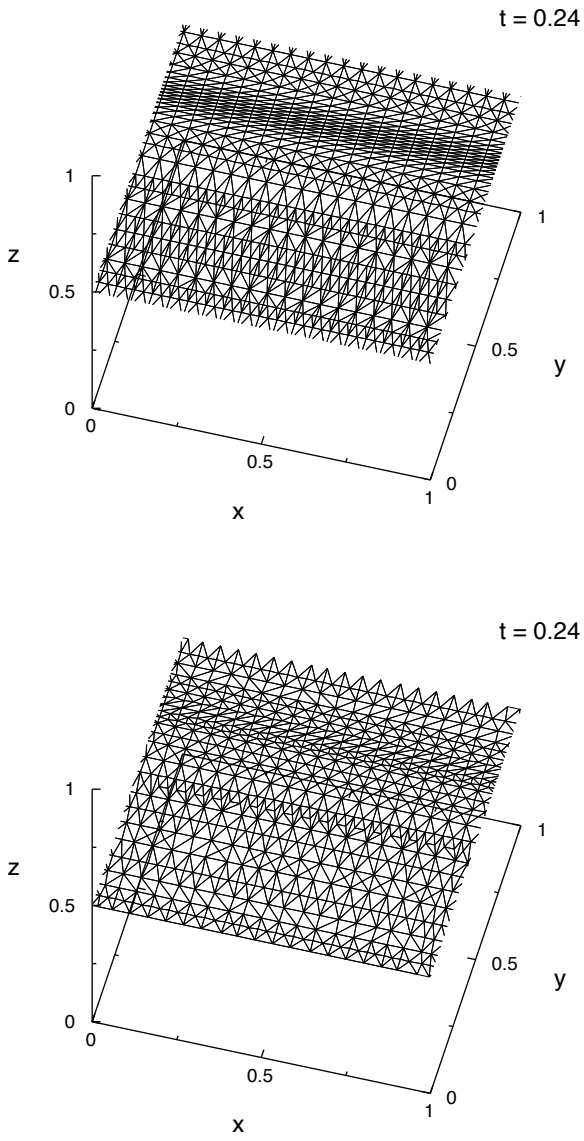

Figure 3.44: Computational elements for merging tests, t=0.24; top: without node merging, 1634 elements; bottom: with in-sheet node merging, 1186 elements.

The merging scheme shall be tested with the problem first presented in §3.4.5—that of a unit-radius sphere with fixed vortex sheet strength evolving under self-influential motion. All parameters of the original problem are maintained, and the merging threshold distance is set to $\Delta_{merge} = 0.025$. The edge-splitting threshold remains $\Delta_{split} = 0.1$.

Images of the computational mesh at various simulations times for the case with in-sheet merging appear in figure 3.48. These can be directly compared to the images from the case with no merging which appear in figure 3.32. Obvious in these images is the early formation of the primary vortex ring ($t < 3$), and its subsequent evolution into a fully-turbulent ring ($t \simeq 6$). The generation of non-axisymmetric perturbations on the underside of the structure is due to the rectangular footprint of the area-weighting particle-grid operator. It was found that all particle-grid operators with rectangular kernels destroy the axisymmetric behavior in this and other initially-axisymmetric simulations. Kernels with larger support, though, delay the transition to non-symmetric behavior. This will be elaborated upon in Chapter 4.

The error in circumferential circulation appears in figure 3.49. The case without merging clearly conserves circulation better than cases with merging, but an error of 10^{-1} for the full-merging case is still reasonable considering the extensive amount of remeshing that takes place. Figures 3.50 and 3.51 show the change in kinetic energy and enclosed volume for the three merging types. Again, full merging does not offer improvements for energy or volume conservation, though in-sheet merging performs nearly as well as the no merging case. It is worthwhile to note that the modification of nodes and elements in the mesh due to merging qualifies as a subfilter-scale dissipation process similar in nature to the "hairpin removal" schemes described in detail in §2.5. The gradual decrease in both circulation and energy is unsurprising in that case, as both node merging and hairpin removal selectively remove energy from the small scales as they simplify the geometry. The change in total kinetic energy due to the merging schemes will be further quantified in a later section (§3.8).

3.6 Shear layer shedding

The scheme to create vortex sheets via vortex shedding from simple edges shall be described in this section. A consequence of discretizing the vorticity field as vortex sheet strengths on a triangular mesh is that shedding edges and their behavior can be easily defined.

In the current work, shedding edges serve mainly as drivers for more interesting simulations. As such, our shedding edges always create constant-strength vortex sheets, and do not exhibit two-way coupling as a true shed edge or splitter plate would. This simplifies the algorithms and does not significantly affect the simulation results.

The geometric manipulations involved in the edge-shedding algorithm rely on effective classification of the nodes and elements according to their roles. The triangle mesh, containing nodes connected with triangles, represents every vortex sheet present in the simulation regardless of its ability to move under its own influence. All nodes in the mesh are either "static" or "free," with the only distinction being that free nodes will convect at a velocity interpolated from the velocity field. Triangular elements, though, are either "static," "free," or "shedding." Static elements are those whose three nodes are also static. Static elements maintain a *constant* total vorticity throughout the simulation. Note that no extra boundary conditions are imposed on these elements. Such a condition would elevate the simulation to a boundary integral method and require solution of a matrix equation for the unknown variable strengths. Free elements are those whose three nodes are also free. Shedding elements are those that span both free and static nodes and thus connect static elements and free elements. These elements will stretch because one set of nodes advects while the others remain fixed. The vorticity discretization scheme maintains constant vortex sheet strength throughout this stretching, but the edge-splitting scheme will eventually flag the elements for splitting.

When the average length of the elements' edges along a shed edge exceeds a predetermined threshold (d_{max}), all shedding element pairs in that shed edge undergo a special splitting operation. This operation shortens the shedding elements and places a pair of free elements in the resulting gap. The new node locations are always the geometric midpoints of their respective triangle edges. Figure 3.52 illustrates this operation. The

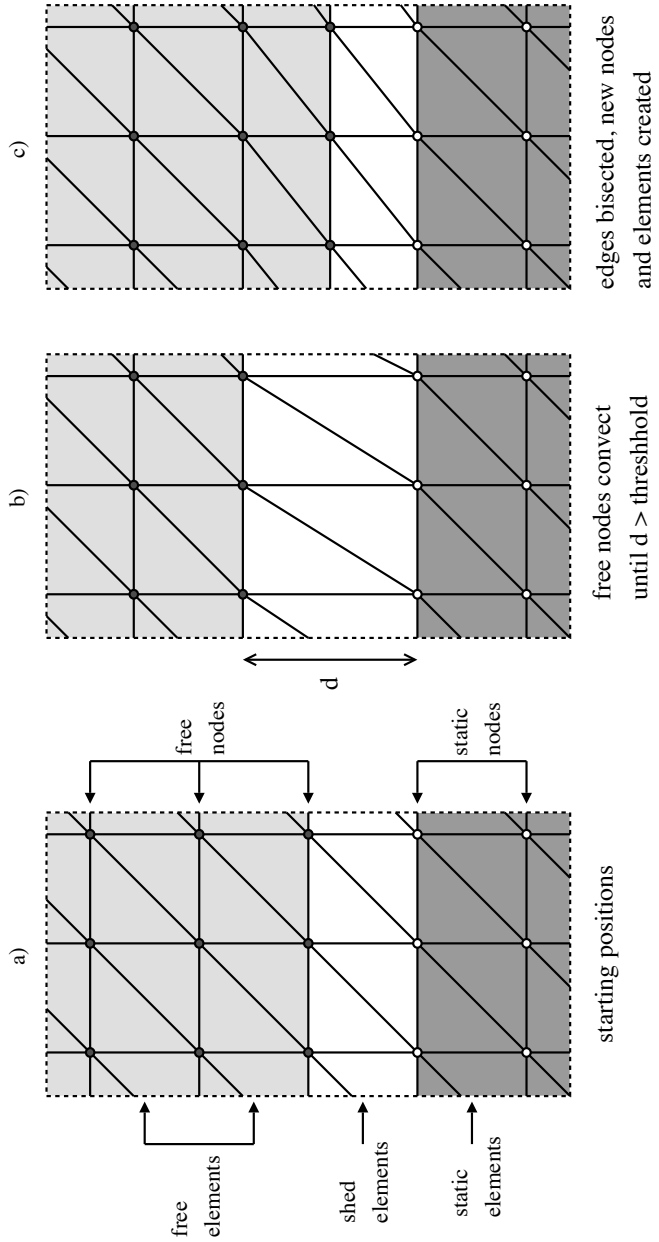

Figure 3.52: Shedding edge activity; a) initial state, b) after convection has moved the free nodes enough that $d > d_{max}$, c) after splitting the perpendicular edges and creating the new nodes and elements.

simulation proceeds normally until the threshold is exceeded once again.

One test of the accuracy of this method is to determine the rate of circulation creation for a simple shed edge. The test shall simulate the vortex roll-up downstream of a splitter plate that separates flows of velocities $\boldsymbol{u}_{upper} = 1.5\hat{i}$ and $\boldsymbol{u}_{lower} = 0.5\hat{i}$. A static vortex sheet exists at $(0 < x < 1, y, z = 0)$, and a row of shedding element pairs covers $(1 < x < 1.0556, y, z = 0)$, both with $\gamma = 1.0\hat{j}$. The domain is $[0 : 32][0 : 0.5][-8 : 8]$ and is periodic in the y-direction. The boundary velocity is a fixed $\boldsymbol{u} = 1.5\hat{i}$ on the $z = 8$ boundary and on the top halves of the $x = 0$ and $x = 8$ boundaries. Likewise, the $z = -8$ boundary and the lower halves of the $x = 0$ and $x = 32$ boundaries experience a fixed boundary condition of $\boldsymbol{u} = 0.5\hat{i}$. The cell size is $\Delta x = 0.1$ and the particle-grid operator is the area-weighting kernel ($\delta = \Delta x = 0.1$). The step size is $\Delta t = 0.05$, which corresponds to a CFL number of $\simeq 0.95$. Remeshing consists of edge splitting and node merging.

Images of the mesh in the x-z plane appear in figure 3.53. The shear layer created by the shed edge initially rolls up smoothly into a single vortex, but Kelvin-Helmholtz rolls appear rapidly and grow to cover major portions of the mesh. The K-H rolls are graphically determined to be spaced $\lambda_{roll} \simeq 1.1$ apart, which is close to the wavelength of the most unstable mode $\lambda = 13.2\delta = 1.32$.

The rate of creation of circulation per unit length (along the shed edge) is expected to be constant and equal to the product of the velocity jump (the vortex sheet strength) and the mean velocity (the rate of sheet creation), in this case $\partial\Gamma/\partial t = 1$. Figure 3.54 is a plot of the change in total system circulation from step to step normalized by the time step size and the length of the shed edge. It shows that the simulation approaches the ideal circulation creation rate only after the initial vortex has moved far enough away from the shedding edge. Initially, circulation is created faster than expected; this is because the free nodes of the shedding elements move downward due to the influence of the static elements, and, thus, the nodes travel farther than they would if they simply moved straight out from the shedding edge. Longer elements with constant vortex sheet strength means greater total circulation and vorticity. A shed edge with true two-way coupling should not exhibit this problem, but preliminary tests of such a system revealed unrecoverable instabilities, and the effort was abandoned. The rapid fluctuations in the circulation creation

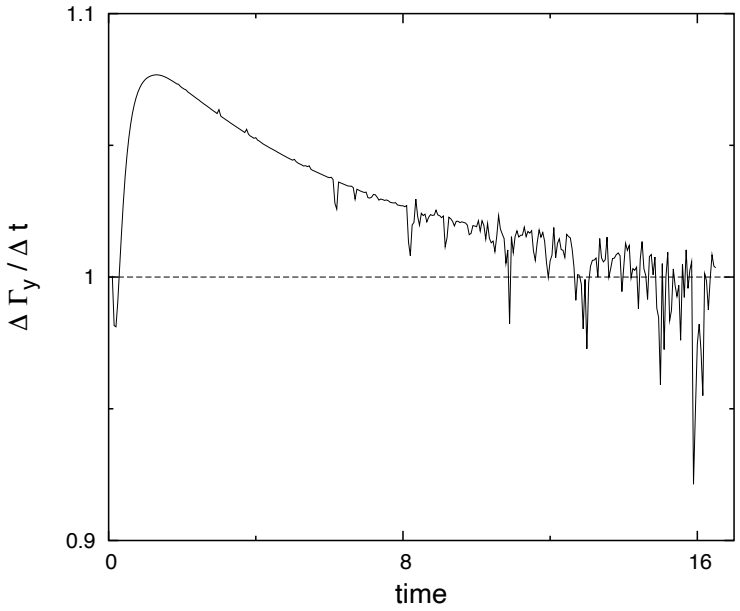

Figure 3.54: Rate of creation of circulation (Γ_y), simple shedding edge.

rate curve are caused by the node merging scheme, which must subtly adjust elements' vortex sheet strengths.

3.7 Weak density discontinuities

As was concluded in §2.4, the baroclinic generation term in the nondimensional vortex sheet evolution equation corresponding to the Boussinesq assumption is

$$\frac{D\gamma}{Dt} = 2\,\theta\,\hat{\boldsymbol{n}} \times \hat{\boldsymbol{g}}, \tag{3.59}$$

where $\theta = A/\mathrm{Fr}$, and A is the Atwood ratio, defined as

$$A = \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2}. \tag{3.60}$$

Also $\hat{\boldsymbol{n}}$ is the unit-length surface normal vector, and $\hat{\boldsymbol{g}}$ is the normalized gravity vector. The numerical method must be able to account for creation and destruction of vortex sheet strength according to this equation.

3.7.1 Implementation

Each element in the simulation contains in its data structure its own local Atwood number. Logically, all elements in an interconnected sheet share the same value, though there may be two or more separate sheets in a given simulation. An example of a simulation that contains two separate vortex sheets appears in §4.4.

For each element i in the interface which supports a density jump, the change in the element's total vorticity is first calculated as

$$\Delta\boldsymbol{\alpha}_i = \Delta t \, 2 \, a_i \, \theta_i \, \hat{\boldsymbol{n}}_i \times \hat{\boldsymbol{g}}, \tag{3.61}$$

with a_i representing the triangular element's area. This vector quantity is converted to edge circulations according to the procedure defined in §3.1, and those new circulations are added to the element's current edge circulations.

Updating the vortex sheet strength for the elements takes a negligible amount of time, so it is done for every substep in the forward integration.

3.7.2 Validation

To test the accuracy of the proposed method, comparisons were made with flows with known analytical solutions. In the limit of small perturbations, linear theory predicts specific behavior of an infinitely-thin sinusoidally-perturbed horizontal interface separating two fluids of different densities [Chandrasekhar, 1961]. The shape of the surface follows

$$f(x,t) = F_0 \; \exp\big(ikx + nt\big) \tag{3.62}$$

$$n^2 = -A \, g \, k = -\theta \, k \tag{3.63}$$

where k is the wavenumber ($k = 2\pi/\lambda$), A is the Atwood ratio, and g is the acceleration due to gravity.

If the upper fluid is lighter, $A > 0$, and the value in parentheses in equation (3.62) is purely imaginary. Thus, the interface will oscillate according to

$$f(x,t) = F_0 \exp\left(ikx + it\sqrt{|\theta|k}\right) \tag{3.64}$$

which has a period of

$$\tau = \frac{2\pi}{\sqrt{|\theta|k}}. \tag{3.65}$$

If the upper fluid is heavier, the interface will continuously deform, with the amplitude of the deformation following

$$f_{max} = F_0 \cosh\left(t\sqrt{|\theta|k}\right) \tag{3.66}$$

which clearly contains an exponentially-growing component.

The simulation set-up for comparison with linear theory is as follows. A three-dimensional computational domain of dimensions [0:1][0:1][-4:4] with periodic boundaries in the x- and y-directions and slip wall boundaries in the z-direction is set up. An interface is created in the xy-plane according to $z = 0.01\sin(2\pi x)$. This corresponds to $k = 2\pi$. The Boussinesq coefficient θ is set to unity. With these values, the period in the stable case should be

$$\tau = \frac{2\pi}{\sqrt{|\theta|k}} = \frac{2\pi}{\sqrt{2\pi}} = 2.50663 \tag{3.67}$$

and the magnification factor for the unstable case

$$m = \frac{f_{max}}{F_0} = \cosh\left(t\sqrt{|\theta|k}\right) = \cosh\left(t\sqrt{2\pi}\right). \tag{3.68}$$

The VIC grid resolutions tested are $\Delta x = \frac{1}{15}, \frac{1}{30}, \frac{1}{60}$. The computational mesh for $\Delta x = \frac{1}{15}$, corresponding to a grid of 29 by 29 pairs of triangles, is illustrated in figure 3.55. Interpolation is done using the rectangular Peskin function with $\varepsilon = 3$, the M4′ kernel, and the area-weighting kernel. A second-order Runge-Kutta method with uniform $\Delta t = 0.05$ (which corresponds to about 50 steps per period) is used for the forward integration.

The performance of the numerical method described above with respect to linear theory is illustrated in figures 3.56 and 3.57. Because the proposed method is regularized and the linear theory is based on infinitely-thin vortex sheets, the overall velocities should be smaller, causing the oscillation period to be longer and the unstable growth rate to be smaller than theory. The lowest errors are observed for the M4′ kernel and the highest errors for the Peskin kernel—due to that kernel's large support radius. All kernels exhibit approximately first-order convergence to the analytical limits presented above except for

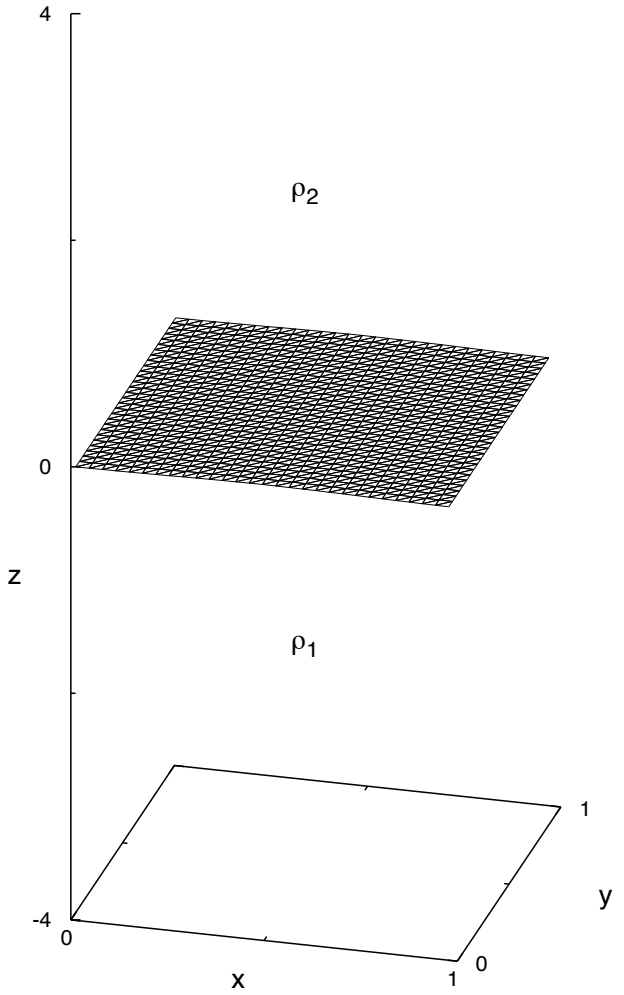

Figure 3.55: Computational mesh for validation of vorticity generation according to Boussinesq assumption, for $\Delta x = \frac{1}{15}$ case.

the case of the stable oscillation period using the M4′ interpolant, which demonstrates second-order convergence.

3.8 Sub-filter scale dissipation

The Smagorinsky-equivalent subgrid-scale model presented in Mansfield *et al*. [1996] and §2.5 is of the form

$$\frac{D\,\overline{\omega}_i}{Dt} = \overline{\boldsymbol{\omega}} \cdot \nabla \overline{u}_i + \nu \nabla^2 \overline{\omega}_i + \nu_T \left(\frac{\partial \overline{\omega}_i}{\partial x_j \partial x_j} - \frac{\partial \overline{\omega}_j}{\partial x_i \partial x_j} \right). \tag{3.69}$$

where the eddy diffusivity ν_T is

$$\nu_T = (c_T \delta)^2 \sqrt{2 \overline{S}_{ij} \overline{S}_{ij}} \tag{3.70}$$

the filtered rate-of-strain tensor $\overline{S}_{ij}$ is

$$\overline{S}_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \tag{3.71}$$

The constant in the eddy diffusivity equation is be taken to be $c_T = 0.15$ [Mansfield *et al*., 1996], similar in magnitude to the constant in the Smagorinsky model for subgrid-scale dissipation in velocity variables.

The present method will compute these derivatives on the Eulerian description of the resolved vorticity field, and the change in vortex sheet strength will be interpolated from the grid back to the individual elements. This follows the grid-based approach put forward by previous studies, but interpolates back to existing elements instead of remeshing the vorticity support with replacement particles.

3.8.1 Implementation

The numerical implementation of the proposed subfilter-scale dissipation model is as follows:

1. For each grid cell, $\overline{S}_{ij}$ is calculated from the velocity field using second-order centered differences, with non-periodic boundary cells using first-order one-sided two-point differences (3.71).

2. $\overline{S}_{ij}$ is then contracted and multiplied by the coefficient to create the ν_T field (3.70).

3. Then, the vorticity correction vector field—in parentheses in equation (3.69)—is created from the pre-corrected vorticity field using five-point second-order second derivatives, and four-point second-order mixed second derivatives.

4. The vorticity correction vector field is then multiplied elementwise by the scalar ν_T field.

5. Finally, the filtered vortex sheet strengths on each element are modified according to the interpolated vorticity correction vector field at the center of the element multiplied by the integration time step size.

The effects of this subfilter dissipation can be seen in the following section.

3.8.2 Observations

A proper study of the subfilter-scale energy transfer in the present method must account for both the implicit and explicit behaviors, arising from node merging routines and the aforementioned Eulerian subgrid-scale dissipation model, respectively. A primary, and preliminary, measure of this transfer is the kinetic energy within the resolved scales of motion. A minimum requirement of an LES model is to be able to move energy from the smallest resolved scales into the unresolved scales, thus stabilizing the flow and decreasing the total resolved kinetic energy. Unfortunately, this is the best that some of the most frequently-used subfilter-scale models are able to achieve [Burton and Dahm, 2005]. Because the eddy-viscosity model of the present method is very similar to those elementary LES models, it is not expected to improve upon their performance significantly.

The global energy behavior can be further quantified by relating energy to its spatial scale, thereby elucidating the scales from which energy is input and removed from

Figure 3.58: Rendered view of computational mesh for periodic box turbulence case.

the system, and quantifying its movement among the scales. A major indicator of the validity of a subfilter-scale model is whether the energy spectrum for the intermediate length scales exhibits the classic classic $k^{-5/3}$ power-law scaling of Kolmogorov [1991]. The longer run-times required to achieve this scaling are not easy to achieve given the present method's persistent front-tracking scheme. Thus, caution should be taken when interpreting the subsequent results.

The simulation from which this data shall be drawn consists of a periodic cube of side length $L = 1$, filled with 200 vortex cylinders, randomly located and oriented, each with radius $0.3L$, random length between $0.015L$ and $0.06L$, and random circumferential circulation within ± 1. The initial computational mesh appears in figure 3.58. Each cubic grid cell has $\Delta x = \frac{L}{32}$, and the area-weighting interpolation method is used for all particle-to-grid and grid-to-particle transfers, making $\delta = \Delta x = \frac{L}{32}$.

The parameters that characterize these simulations are the box size L, the regularization length δ, the initial circulation of the cylinders Γ, and time t. From these, length

scales are normalized by the box size ($\delta^* = \delta/L$), time scales as $t^* = t\,\Gamma/L^2$, and energy as $k^* = kL^2/\Gamma^2$.

The total kinetic energy was calculated conventionally. The energy spectrum was determined by a Fourier transform of the mean diagonal of the unscaled two-point spatial correlation tensor $R_{ij}(r) = \overline{u_i'(\boldsymbol{x})u_j'(\boldsymbol{x}+r)}$ evaluated at 100 points along a randomly-oriented vector with length $L/2$ originating at each of 100000 random points in the computational volume.

Runs were made with the subfilter scale model on and off, and with the node merging routines off and set to in-sheet merging and full merging. Typical flowfields for the late stage of the simulations appear in figure 3.59, which illustrates the initial and final solved states for the case with full merging and explicit subfilter-scale dissipation. Significant deformation of the computational surface has occurred, though the element count, and by definition the surface area, has increased 8-fold. Unfortunately, $t^* = 2$ represents the longest simulation possible with available computational resources and only corresponds weakly to isotropic turbulence. The mean spatial correlation tensor at the final solved time is

$$R_{ij}(r=0,t^*=2) = \begin{pmatrix} 0.00779549 & 0.000430999 & -0.000318269 \\ 0.000432364 & 0.00786703 & 0.000106385 \\ -0.000317849 & 0.000106869 & 0.00882724 \end{pmatrix} \tag{3.72}$$

The global influence of the merging and explicit subfilter dissipation on the kinetic energy is seen in figure 3.60. These data contains several noteworthy features. Most obviously, the explicit subfilter dissipation scheme removes far more energy than the merging scheme alone. This is because the merging scheme will only remove energy in the specific event of a merging operation, which is triggered by entirely ad-hoc criteria. On the other hand, the subfilter-scale dissipation acts uniformly over the entire mesh at every time step based on physical features of the regularized flowfield. While the merging operation is algorithmically different than previous filament element-wise remeshing schemes, its similar nature implies that relying on such an implicit, geometry-based LES model will ignore much of the "true" subfilter energy transfer. Secondly, the merging operation that is limited to in-sheet merging transfers little more energy to the

3.9 Numerical integration

Of the two key equations—the kinematic velocity-vorticity relationship and the dynamic vortex sheet strength evolution equation—only the dynamic equation needs to be integrated forward in time to continue the simulation. The kinematic relationship between vorticity and velocity is used only when a velocity field must be determined for the current configuration.

The proposed numerical method will work with any multistep integrator, and first-order Euler, second-order Runge-Kutta, and fourth-order Runge-Kutta schemes are all programmed. Figure A.1 illustrates in flowchart form the procedure used for one time step of a first-order Euler forward integrator. In summary, edge-splitting (§3.4) and node-merging (§3.5) procedures are conducted together as the remeshing operation once per time step, and the vorticity update terms (§3.7 and §3.8) are evaluated once per integrator substep.

Finally, the program can run with a fixed time step size, or a fixed maximum Courant number, defined as

$$CN = \frac{||\boldsymbol{u}_{max}||}{\Delta x/\Delta t}. \tag{3.73}$$

APPENDIX A

The *vort3d* program

A computer code called *vort3d* was written to test the proposed vortex method. Its design and usage will be described in this section.

A.1 Program flowchart

Figure A.1 demonstrates the flow of logic in the *vort3d* program for the 1st-order Euler integrator. The program flow for the standard 2nd-order Runge-Kutta forward integrator is illustrated in figure A.2.

A.2 Problem setup

The *vort3d* program allows a wide variety of simulations to be run without recompilation, thanks to an extensive input file format. A user should create the input file as generic text file, with no spaces in the filename, and making sure to use the extension `.v3d`.

Within that file, the user should include any parameters that are specific and relevant to the problem to be solved. The most commonly-used input file parameters are described in figures A.3 to A.6. All *italicized* components are to be replaced with actual values, and all *[parenthesized]* components are optional.

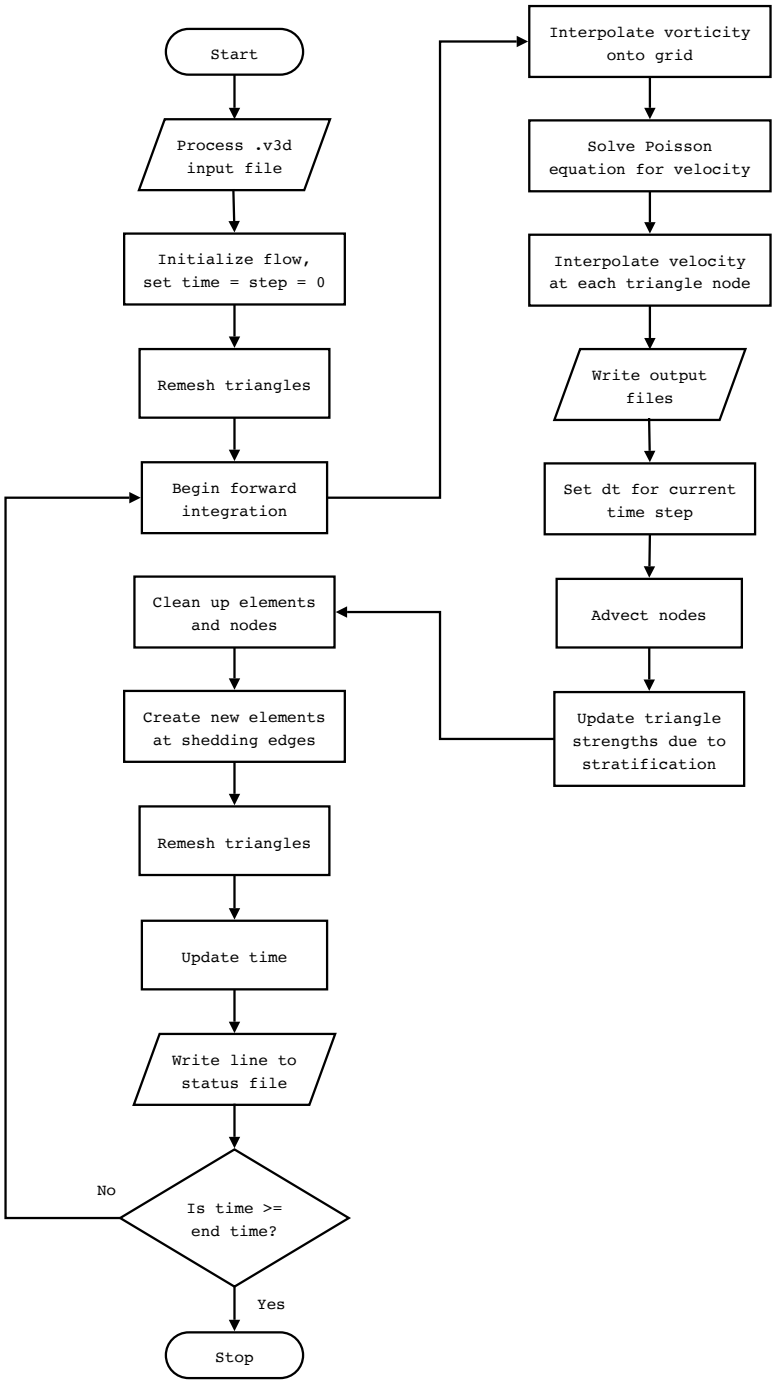

Figure A.1: *vort3d* program flowchart, 1st-order Euler time-stepping.

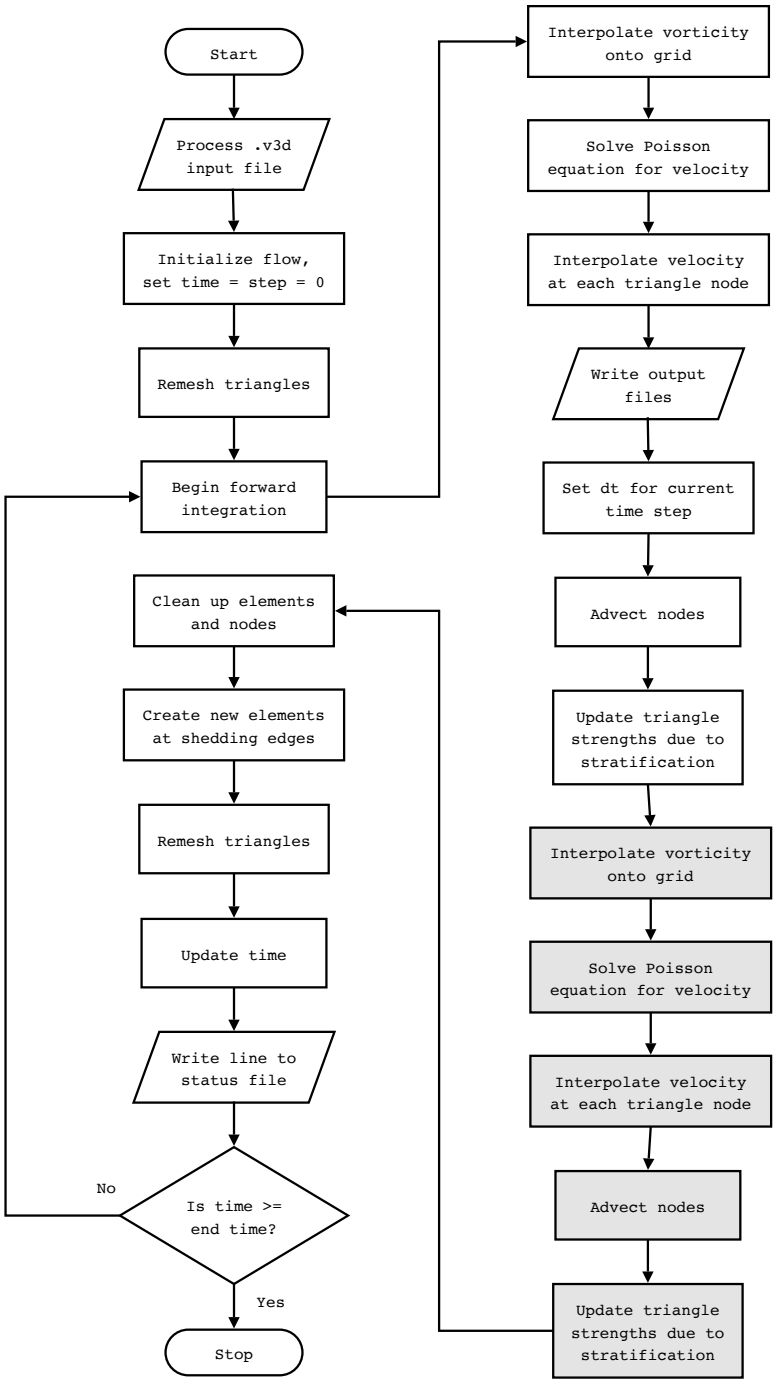

Figure A.2: *vort3d* program flowchart, 2nd-order Runge-Kutta time-stepping, shaded boxes represent steps not present in 1st-order Euler method.

`domain` *`xsize ysize zsize`* domain is the x,y,z size of the computational domain
`dstart` *`xstart ystart zstart`* sets the minimum-valued corner location, can be negative
`plotd` *`xstart xend ystart yend zstart zend`* sets the plot domain bounds
`boundary` *`c[:c] c[:c] c[:c]`* sets boundary conditions: *`c`* is `p` for periodic, `w` for wall, `o` for open, `w:o` for wall on the lower side, open on the upper
`traveling` *`c c c`* indicates that the domain position (`dstart`) should change in time to keep the vortex sheet centered along any given axis; *`c`* can take on values `y` or `n`
`gravity` *`x y z`* sets the gravity vector (vector will be used as given, not normalized)
`vic_density` *`density`* sets the density of vortex-in-cell computational cells per world unit
`mesh_quality` *`quality`* scales the average size of the triangular elements, keep this above 0.5 for Peskin function interpolation, and above 0.8 for M4′ or lower-order interpolants
`end_time` *`time`* sets a finite end time for the simulation
`out_dt` *`dt`* simulation time per output step
`courant` *`num`* sets the maximum internal time step as equal to this factor (the Courant number) times the maximum time step allowed by the CFL condition
`dt` *`dt`* sets the maximum internal time step if the Courant number is not used
`dissipation vortsmag` *`[coeff]`* activates a vorticity-based Smagorinsky model to transfer energy from the resolved to the unresolved scales; takes an optional coefficient

Figure A.3: *vort3d* input file parameters, part 1.

`split` *`[type] [threshold]`* activates an edge-splitting procedure; *`type`* can be `length` (the default), in which case edges that are longer than `threshold` times the cell size divided by `quality` are flagged for splitting (default = 0.8); *`type`* can be `angle`, whereupon edges whose end nodes' normals differ by more than `threshold` radians are flagged (default = 0.1); or *`type`* can be `curvature`, for which edges are split only if the average of the end nodes' mean curvature is greater than the `threshold` (default = 0.05)
`midpoint` *`[type]`* sets how the midpoint is selected in instances where edges are split; if *`type`* is omitted, the default geometric midpoint option is selected; if *`type`* is `cylinder` or `spline`, a more accurate and volume-preserving method is used
`merge` *`[type] [threshold]`* activates the node merging routines; pairs of nodes will be flagged for merging if they are closer than `threshold` times the cell size divided by `quality`; when *`type`* is `sheet`, only nodes that a common element are merged; when *`type`* is `full`, any close node pair can be merged; and when *`type`* is `first`, merging is only done once, before the simulation begins
`mergemidpoint` *`[type]`* sets how the merged node location is selected; if *`type`* is omitted, the default geometric midpoint option is selected; if *`type`* is `cylinder` or `spline`, a more accurate and volume-preserving method is used
`vortinterp` *`[type] [radius]`* controls the interpolation method used on the vorticity field; *`type`* can be `speskin` (radial, or spherical Peskin function), `peskin` (non-spherical, or rectangular Peskin function), `m4p` (Monaghan's $M4'$), or `area` (area- or volume-weighting); *`radius`* can only be defined for `speskin` and `peskin` interpolation functions
`velinterp` *`[type] [radius]`* controls the interpolation method used on the velocity field, all options are identical to `vortinterp`, above

Figure A.4: *vort3d* input file parameters, part 2.

`setup` *`key [arguments]`*
activates one of many built-in simulation initializations, some require additional arguments to realize; some important ones are listed separately, below:

`setup 36` *`filename.obj x y z scale`*
reads in a Wavefront-compatible triangle mesh file, translates it by `x y z` after scaling it by `scale`

`setup 28` *`height shear scalar amplitude`*
sets up a shear layer in the x-y plane at z=`height`, with an x-velocity jump of `shear`, a non-dimensional scalar jump of `scalar`, and random perturbations in the z-direction of scale `amplitude`

`setup 29` *`[on|off] [atwood]`*
turns the weak stratification calculation on or off, and sets the Atwood ratio as applicable

`setup 44` *`xpert ypert zpert`*
perturbs the nodes of the existing mesh randomly within a range of the cell size times the x-, y- and z-perturbations given

`setup 45` *`axis xpert xwave ypert ywave zpert zwave`*
perturbs the nodes of the existing mesh in the `axis` direction (`0`=x, `1`=y, `2`=z) according to $x' = x + \texttt{xpert}\sin(2\pi x/\texttt{xwave})$, etc

Figure A.5: *vort3d* input file parameters, part 3.

`output` *`format variable [axis] [plane] [black] [white]`*
each `output` line creates one specific output file for each output time step in the simulation;

the available values for *`format`* are `txt` and `png` (for 2D raster data), `eps` (for 2D vector data, or contours of raster data), `obj`, `ply`, and `bin` (for 3D triangle mesh data), `tec` and `b3d` (for 3D vector raster data);

the available values for *`variable`* are `vel` and `vort` for vector quantities, `u`, `v`, and `w` for scalar velocities, `wx`, `wy`, and `wz` for scalar vorticities, `um` and `wm` for velocity and vorticity magnitudes, and `ud` and `wd` for velocity and vorticity divergences;

`axis` can be `xy`, `xz`, or `yz`;

`plane` can be `i` for an integrated plane, `x` for the maximum value in the column, `n` for the minimum, a number $[0:1]$ identifying the relative location of the plane in the free direction, or a number preceded by `+` or `-` to identify the absolute coordinate of the plane in the free direction

some examples:

```
output png elem xz
output obj elem free
output eps elem xz 0.5
output eps vel xz 0.5
output tec vel
output png w xz 0.5 -1.0 1.0
output png ud xz i 0.0 0.001
output png wz xy x 0.0 10.0
output png wy xz -0.5 -10.0 10.0
```

`output_scale` *`pixels`*
sets the scale for all raster (image) output in pixels per world length unit

Figure A.6: *vort3d* input file parameters, part 4.