Volume Tracking on Adaptively Refined Grids with Curvature Based Refinement

Mayank Malik[1], Markus Bussmann[2]

Department of Mechanical & Industrial Engineering, University of Toronto

[1]mayank.malik@utoronto.ca, [2]bussmann@mie.utoronto.ca

We present a piecewise-linear interface calculation (PLIC) volume tracking method implemented on an adaptively-refined structured mesh, that one could incorporate into an adaptive interfacial flow solver to accurately track interfaces characterized by a wide range of length scales and curvatures. The technique uses adaptive hierarchical grids that are recursively subdivided on a Cartesian mesh, with a refinement criterion related to the local curvature of an interface. Results are presented of the scheme, that requires much less computational time than a uniform mesh to reach a given error norm. A new test case is also presented, with features that are typical of multiscale interfacial flows.

1. INTRODUCTION

An important and challenging application of CFD is interfacial and multiphase flows, that involve multiple immiscible fluids separated by one or more interfaces. These flows occur in many natural and industrial processes, including combustion, petroleum refining, many materials processing techniques, and nuclear technologies [1]. Numerical simulation is an important tool for the investigation of such flows. What sets such simulations apart from those of single phase flows is the additional requirement of tracking interfaces in space and time. Various tracking methodologies have been developed, including markers [2], level sets [3], and volume tracking [4].

We focus in this paper on volume tracking methods, that are popular and have been successfully applied to the simulation of a variety of interfacial flows. Rider and Kothe [4] and Rudman [5] have written excellent reviews of such methods, and so we present only a brief overview here. Volume tracking methods have their origin in algorithms exemplified by the Volume of Fluid (VOF) scheme of Hirt and Nichols [6], that represented interfaces in a piecewise-constant manner, that manifested itself as a stair-stepped representation of a curved interface. Many well-known codes, including SOLA-VOF [7] and until recently, FLOW-3D [8], utilized such algorithms. Today, such methods are largely considered obsolete, and have been replaced by algorithms that approximate an interface with a line (in 2D) or a plane (in 3D) at any orientation to a mesh cell. Such methods are referred to as piecewise-linear interface calculation (PLIC) methods; examples include the work of Youngs [9,10], Rider and Kothe [4], and Scardovelli and Zaleski [11].

PLIC schemes have usually been designed for, and implemented into, codes that utilize fixed structured meshes. This can lead to difficulties when simulating interfacial flows characterized by features at different scales: for a given computational expense, the results are usually characterized by regions of the domain that are under-resolved, and other regions that are unnecessarily refined. As well, interfacial flows often involve dramatically different fluids, the properties of which can vary by orders of magnitude. It is thus often appropriate to resolve an interface more accurately than the domain far from it.

Interfacial flow modelling is thus an obvious candidate for the application of adaptive refinement techniques. In this paper, we focus on the volume tracking algorithm, and present the implementation of a well-known PLIC scheme on an adaptively-refined mesh. Rather than resolve the interface to some uniform level, as was presented recently [12], we utilize local

curvature (that is a relative measure of interface complexity), as our refinement criterion.

2. METHODOLOGY

We begin here by presenting a brief overview of volume tracking on a uniform cartesian mesh: the mathematical basis, and the numerical methodology of Youngs [9]. We then present our implementation of a similar algorithm on an adaptive mesh, and discuss (i) why and how we use a quadtree implementation for adaptive refinement, (ii) the aspects of the implementation that differ from that on a uniform mesh, and (iii) the use of interface curvature as a refinement criterion.

2.1. Mathematical Model

In volume tracking methods, an interface is not explicitly tracked, but rather the fluid volume is utilized as an indicator function for the interface. Consider two fluids denoted as "dark" and "light". We define an existence function f that has a value of one in the dark fluid, and a value of zero in the light.

A standard advection equation governs the evolution of f ; for a given flow field $\vec{u}$:

$$\frac{Df}{Dt} = 0 \quad \Rightarrow \quad \frac{\partial f}{\partial t} + \nabla \cdot (\vec{u} f) = 0 \tag{1}$$

As we assume the flow to be incompressible:

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

Equation (2) can be interpreted as a statement of conservation of fluid volume, which then implies conservation of the function f .

2.2. Uniform Mesh Volume Tracking

If we now consider a uniform two-dimensional (2D) grid of dimension $h = \Delta x = \Delta y$, we can define a quantity f_{ij} as follows:

$$f_{ij} h^2 = \int_h \int_h f(x, y)\, dx dy \tag{3}$$

f_{ij} is the "volume fraction" of dark fluid in cell *(i,j)*. If $f_{ij} = 1$, the cell contains only dark fluid; if $f_{ij} = 0$, only light fluid. In general, the volume fraction of the dark fluid is:

$$0 \le f_{ij} \le 1 \tag{4}$$

The corresponding fraction of light fluid is then $1 - f_{ij}$, and we identify an "interface cell" when:

$$0 < f_{ij} < 1 \tag{5}$$

It is these volume fractions that implicitly identify an interface; the evolution of the volume fractions is governed by equation (1). A PLIC volume tracking algorithm is simply a geometric (rather than algebraic) approach to solving equation (1), and it generally consists of a two-part procedure: a piecewise-linear reconstruction of the interface from known volume fractions, and a time marching, or advection, step.

Consider a 2D square domain of uniform cells. In any interface cell, we reconstruct the interface by a line:

$$\vec{n} \cdot \vec{x} + \rho = 0 \tag{6}$$

or:

$$n_x x + n_y y + \rho = 0 \tag{7}$$

For a given slope (n_x, n_y) , there exists only one value of ρ that corresponds exactly to the known volume fractions. Referring to Figure 1, we calculate a cell-centered normal by averaging normals calculated at the cell vertices, that are each obtained from a simple four point stencil. For example, the normal at the bottom left corner of cell (i, j) is given by:

$$\begin{aligned} n_{x_{i-1/2, j-1/2}} &= \frac{1}{2h}\left[f_{i,j} + f_{i,j-1} - f_{i-1,j} - f_{i-1,j-1}\right] \\ n_{y_{i-1/2, j-1/2}} &= \frac{1}{2h}\left[f_{i,j} + f_{i-1,j} - f_{i,j-1} - f_{i-1,j-1}\right] \end{aligned} \tag{8}$$

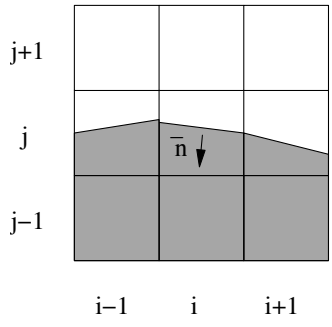

Figure 1: *The 3*3 stencil used to calculate the normal for the cell (i,j).*

For a given $\vec{n}$, the line constant ρ can then be calculated analytically [5] or iteratively [4].

With the interface reconstructed, the second step is to advect the volume fractions. Following the algorithm of Youngs [9], we employ an operator-split time integration scheme, and integrate separately in each of the dimensions *(x,y)*. To ameliorate anisotropic effects, we reverse the order of advection each timestep, from *(x,y)* to *(y,x)* and back.

The advection calculation from a time n to $n+1$ proceeds as follows. After advecting in one dimension (in this case x), an interim volume fraction field is calculated as:

$$f_{ij}^{*} = \frac{f_{ij}^{n} V_{ij} - (u_{i+1/2,j}\langle f\rangle_{i+1/2,j} - u_{i-1/2,j}\langle f\rangle_{i-1/2,j})h\Delta t}{V_{ij}^{*}} \qquad (9)$$

where the interim cell volume is:

$$V_{ij}^{*} = V_{ij} - (u_{i+1/2,j} - u_{i-1/2,j})h\Delta t \qquad (10)$$

Scaling the interim volume fractions by V_{ij}^{*} ensures that they remain bounded: $0 \le f_{ij}^{*} \le 1$. The interim field is then reconstructed again, and advected in the second direction:

$$f_{ij}^{n+1} = \frac{f_{ij}^{*} V_{ij}^{*} - (v_{i,j+1/2}\langle f\rangle_{i,j+1/2} - v_{i,j-1/2}\langle f\rangle_{i,j-1/2})h\Delta t}{V_{ij}} \qquad (11)$$

In equations (9) and (11), the $\langle f \rangle$ represent the volume fractions fluxed across faces; these are calculated geometrically, as depicted in Figure 2.

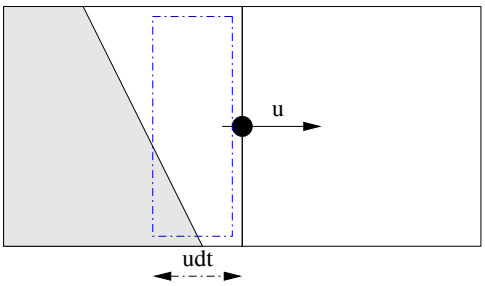

Figure 2: *Advection between two cells at the same refinement level.*

2.3. Quadtree Grid Generation

The term *quadtree* [13] is used to describe a class of hierarchical data structures that are based on the principle of recursive decomposition of space. Thus, quadtrees are ideally suited for an adaptive refinement framework, and in the context of volume tracking, allow us to repeatedly refine parts of a domain based on the value of some criterion. As an interface moves, and the value of the refinement criterion changes, so the quadtree can evolve in a dynamic manner.

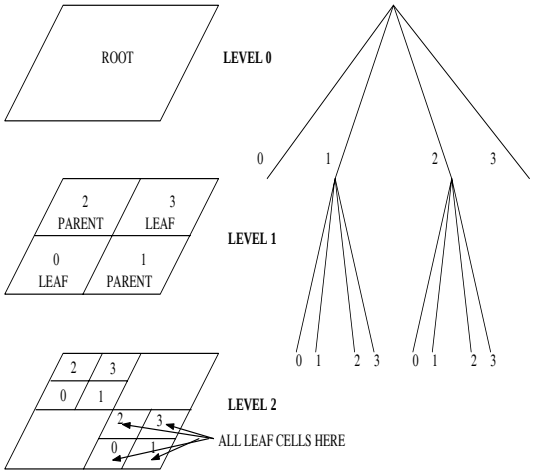

Figure 3: *An example of a three-level quadtree.*

Figure 3 illustrates a quadtree structure. The following is nomenclature associated with the use of quadtrees:

Root: corresponds to the entire grid.
Cell: a square region in the grid.
Parent: a cell that has been subdivided into four child cells.
Leaf Cell: a cell that is no further subdivided.
Level: the position of a cell in the grid hierarchy; the root corresponds to level 0; each subsequent level of refinement corresponds to a halving of one or more cells; the dimension of a cell at level L is $1/2^{L}$ the dimension of the root cell.
Neighbours: the collection of all cells that adjoin a given cell, sharing either a face or a vertex; we use Samet's [13] recursive algorithm to compute neighbours.

2.4. Adaptive Mesh Volume Tracking

The introduction of adaptivity complicates two aspects of a PLIC algorithm: the calculation of normals, and the advection routine, because neighbouring cells may be at different levels of refinement. The explanations that follow, and the results that are presented in this paper, reflect a constraint that we have chosen to apply, that neighbouring cells cannot differ by more than one level of refinement.

We begin by considering the calculation of normals, and note that when a cell and all of its neighbours are at the same level of refinement, a normal is calculated via equation (8). But when a cell *(i,j)* has one or more neighbours at a different refinement level (e.g. as illustrated in Figure 4, where one vertex neighbour is more refined), then we calculate the cell-centred normal by constructing an effective 3*3 stencil.

In particular, for the case of Figure 4, we calculate the volume fraction of the *(i-1,j-1)* parent cell, simply by averaging the volume fractions of the four children. Referring to Figure 5, this corresponds to calculating the single volume fraction on the left from the four on the right.

The flipside is that of a cell *(i,j)* with one or more neighbours at a lower level of refinement. In this case, we utilize the reconstruction of the coarse neighbour to calculate the effective volume fractions of four children, if they existed. Referring again to Figure 5, this corresponds to calculating the four volume fractions depicted in the right schematic from the single fraction to the left.

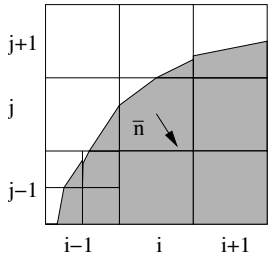

Figure 4: *Calculating a normal for cell (i,j), when neighbouring cells are further refined.*

.

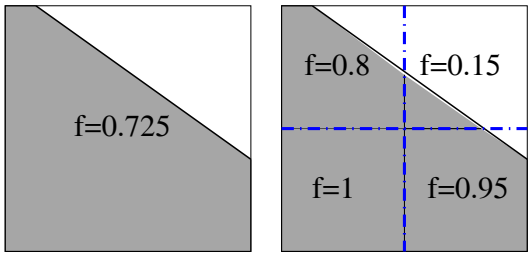

Figure 5: *Corresponding volume fractions between a parent cell and its four child cells.*

Note that to implement this approach to calculating normals, we sweep the quadtree level by level, computing normals in leaf cells at successively finer levels.

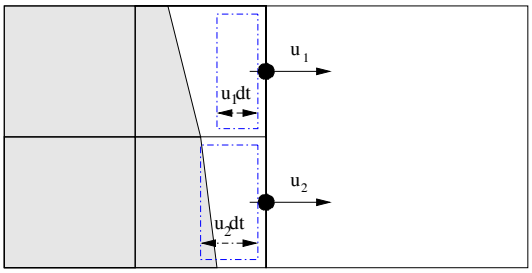

Figure 6: *Advection between cells at different levels of refinement.*

Turning now to advection between cells at different levels of refinement, we refer you to Figure 6, that illustrates the situation. Velocities are presumed to be known at all faces: u_1 and u_2 need not be the same, although the average of the two will equal the left face velocity of the larger cell to the right. Advection into a less refined cell, as pictured in Figure 6, then consists of two flux calculations. For the reverse situation, two fluxed volume fractions are calculated, one for each of the two more refined cells that neighbour a coarser one.

2.5. Criterion for Refinement and Coarsening

The criterion for refining or coarsening a cell is related to the local curvature:

$$\kappa_{ij} = -\nabla \cdot \hat{n} \qquad (12)$$

that we can calculate simply from the vertex normals of the cell *(i,j)*, that are already known from the calculation of the cell-centred normal. The rationale for this choice of criterion relates to the realization that to adequately resolve a curved interface requires some minimum level of resolution. At a simple level, a mesh must be locally fine enough that the piecewise-linear reconstructions are a reasonable estimate of the actual interface topology in a cell.

The results presented in this paper were obtained by relating each refinement level to a range of curvatures. Procedures for refinement and coarsening correspond to the techniques already presented, illustrated in Figure 5.

Finally, note that obtaining accurate curvatures from volume fractions is difficult. Average values of curvature calculated over an interface are usually correct, but single calculations are

often approximate at best. We observed many instances of a calculated curvature that indicated the need for mesh coarsening; once coarsened, the new value of curvature indicated that the cell should be refined. As a result, we implemented a loose version of the refinement criterion, and only coarsened a cell when the change in curvature was deemed appreciable.

3. RESULTS

We present results to four test problems, and compare adaptive grid results with corresponding results from uniform mesh calculations. All test cases were calculated in a unit square domain, with a timestep corresponding to a Courant number of 0.4. Velocity fields are specified as stream functions ψ; velocity components u and v may be calculated as:

$$u = \frac{\partial \psi}{\partial y}; \quad v = -\frac{\partial \psi}{\partial x} \tag{13}$$

Two errors were calculated for each simulation:

$$L1_{ic} = \frac{\Sigma_{ic}(\left|f_{ij}^{T} - f_{ij}^{o}\right| A_{ij})}{\Sigma_{ic} A_{ij}} \tag{14}$$

$$L1_{all} = \frac{\Sigma_{all}(\left|f_{ij}^{T} - f_{ij}^{o}\right| A_{ij})}{\Sigma_{all} A_{ij}} \tag{15}$$

The subscript *ic* refers to interface cells, and *all* to all cells; the superscripts *o* and *T* to initial and final values, respectively; A_{ij} represents the area of the cell *(i,j)*. In all cases, the tests were constructed to return the fluid to its initial position, to facilitate the error evaluation. Finally, note that the two errors are fundamentally different: $L1_{all}$ includes the inherent decrease in total interface cell area with mesh refinement, where $L1_{ic}$ sums strictly over interface cells, and so represents the average change in an interfacial volume fraction between the initial and final state.

3.1. Solid Body Rotation

The first test is one full rotation of an ellipse (major axis diameter of 0.3, minor diameter of 0.1) centred in the domain, and illustrated in Figure 7. Uniform mesh calculations were run at grid resolutions of 32*32, 64*64 and 128*128; the adaptive calculation refined the interface according to the following criteria: $\kappa < 6$, level 4; $6 < \kappa < 15$, level 5; $15 < \kappa < 30$, level 6; $30 < \kappa$, level 7. Note that level 4 corresponds to a uniform 16*16 mesh, and level 7 to a 128*128. Table 1 details the errors. The average number of interface cells is an indication of relative mesh resolution. The errors associated with the adaptive simulation are in line with the uniform mesh errors.

MESH	L1 (INTERFACE ERROR)	L1 (ALL CELLS ERROR)	AVERAGE # INTERFACE CELLS
32*32	0.0766	0.00434	54
64*64	0.04285	0.001192	108
128*128	0.02152	0.000271	217
ADAPTIVE	0.03716	0.00104	155

Table 1: *Results for the solid body rotation test.*

3.2. Elongational Field

Next, we consider an ellipse centered at (0.5,0.5) but with a major axis diameter of 0.4, such that the aspect ratio is 4:1. The stream function is:

$$\psi = xy \tag{16}$$

This field collapses the ellipse into a circle, then stretches it into an ellipse along the other coordinate axis. We then reverse the velocities to return the ellipse to its initial position. Table 2 presents a summary of the error norms. Note that in this and for the final two test cases, the $L1_{ic}$ for the adaptive simulation is actually less than the same error associated with the finest uniform mesh.

GRID TYPE	L1 (INTERFACE ERROR)	L1 (ALL CELLS ERROR)	AVERAGE # INTERFACE CELLS
32*32	0.016329	0.00108	55
64*64	0.01068	0.000354	110
128*128	0.008912	0.000143	220
ADAPTIVE	0.00522	0.000229	144

Table 2: *Results for the elongational field.*

3.3. Slotted Disk

This is a variation of a test case that is often referenced [5]. Consider a circle of radius 0.25, centered at (0.5,0.5), with a slot of a width 0.125. We rotate the slotted disk one full revolution, then reverse the velocity field for another. Results are illustrated in Figure 9; error norms are tabulated in Table 3.

GRID TYPE	L1 (INTERFACE ERROR)	L1 (ALL CELLS ERROR)	AVERAGE INTERFACE CELLS
32*32	0.10371	0.00861	81
64*64	0.09624	0.004238	164
128*128	0.07612	0.001608	330
ADAPTIVE	0.07535	0.002193	271

Table 3: *Results for the slotted disk.*

The mesh was refined according to the following criteria: $\kappa < 5$, level 4; $5 < \kappa < 10$, level 5; $10 < \kappa < 15$, level 6; $15 < \kappa$, level 7.

3.4. Star Stretching

The initial geometry for the final test problem is quite different from those of the first three. We consider a star-like shape which has a large range of curvature, and then advect the shape with the following stream function:

$$\psi = x^2 y + x y^2 + \frac{y^3}{3} \qquad (17)$$

GRID TYPE	L1 (INTERFACE ERROR)	L1 (ALL CELLS ERROR)	AVERAGE INTERFACE CELLS
32*32	0.10952	0.0077	57
64*64	0.09701	0.00341	123
128*128	0.08467	0.001478	271
ADAPTIVE	0.051582	0.001763	221

Table 4: *Results for the star stretching.*

Results are presented in Figure 10 and in Table 4. This is a more demanding test than the previous three, especially because of the fine tips at the edge of the star, that are difficult to resolve even on a 128*128 mesh. However, such features, and such a range of curvatures, are not uncommon in complex multiphase flows, and so a test such as this can serve to more fully exercise a tracking algorithm.

4. SUMMARY

We have implemented a PLIC volume tracking scheme onto an adaptively-refined Cartesian mesh. Complications that result from the adaptivity make the implementation only a little more complex than that on a uniform mesh. We then utilize estimates of local interface curvature as a refinement criterion, in order to optimize the use of available resources. Finally, we present the results of four test cases, and demonstrate that not only do the adaptive results compare well with uniform mesh results, but in some cases errors are actually lower on an adaptive mesh than on a finer uniform one.

5. REFERENCES

[1] M. Ishii and N. Zuber, Drag coefficient and relative velocity in bubbly, droplet or particulate flows, *AIChE J.*, vol. 25, pp. 843, 1979.

[2] S. Popinet and S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, *Int. J. Num. Methods Fluids*, vol 30, pp. 775-793, 1999.

[3] J.A. Sethian, *Level Set Methods*, Cambridge University Press, 1996.

[4] W.J. Rider and D.B. Kothe, Reconstructing volume tracking, *J. Comp. Phys.*, vol. 141, pp. 112-152, 1998.

[5] M. Rudman, Volume-tracking methods for interfacial flow calculations, *Int. J. Num. Methods Fluids,* vol. 24, pp. 671-691, 1997.

[6] C.W. Hirt and B.D. Nichols, Volume of Fluid (VOF) method for the dynamics of free boundaries, *J. Comp. Phys.*, vol. 39, pp. 201-225, 1981.

[7] B.D. Nichols, C.W. Hirt, and R.S. Hotchkiss, SOLA-VOF: A solution algorithm for transient fluid flow with multiple free boundaries, LASL, Report LA-8355, 1980.

[8] Flow Science Inc., Santa Fe, NM.

[9] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, in *Numerical Methods for Fluid Dynamics*, Academic Press, New York, 1982.

[10] D.L. Youngs, An interface tracking method for a 3D Eulerian hydrodynamics code, AWRE, Report 44/92/35, 1984.

[11] R. Scardovelli and S. Zaleski, Direct numerical simulation of free-surface and interfacial flows, *Annual Review of Fluid Mechanics*, vol. 31, pp. 567-603, 1999.

[12] D. Greaves, A quadtree adaptive method for simulating fluid flows with moving interfaces, *J. Comp. Phys.*, vol. 194, pp. 35-56, 2004.

[13] H. Samet. Applications of spatial data structures: computer graphics, image processing, and GIS. Addison-Wesley, 1990.

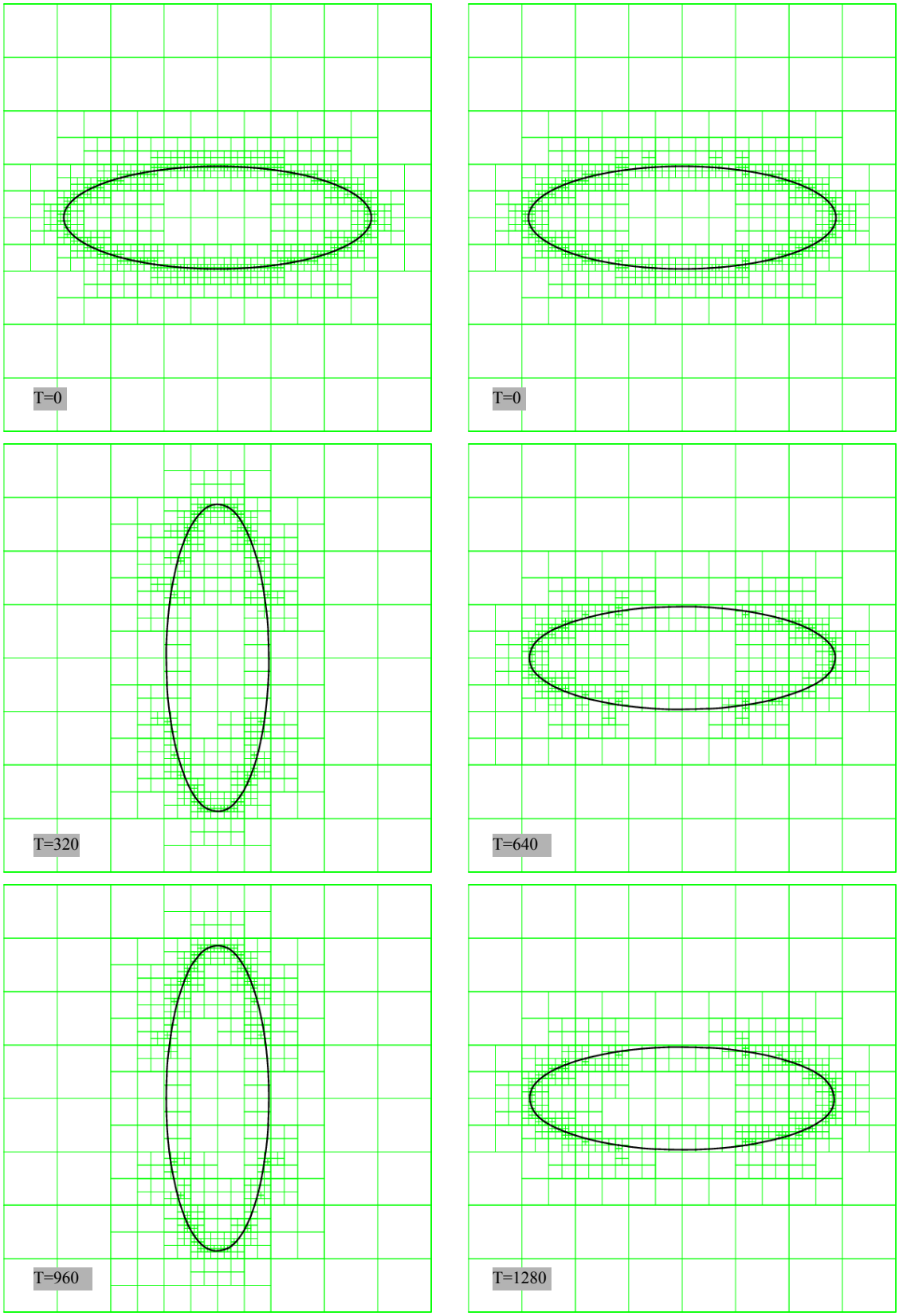

Figure 7: *Ellipse rotation. The upper left plot illustrates the ellipse initially refined to level 7 (to accurately initialize the volume fractions); the upper right plot illustrates the actual initial refinement.*

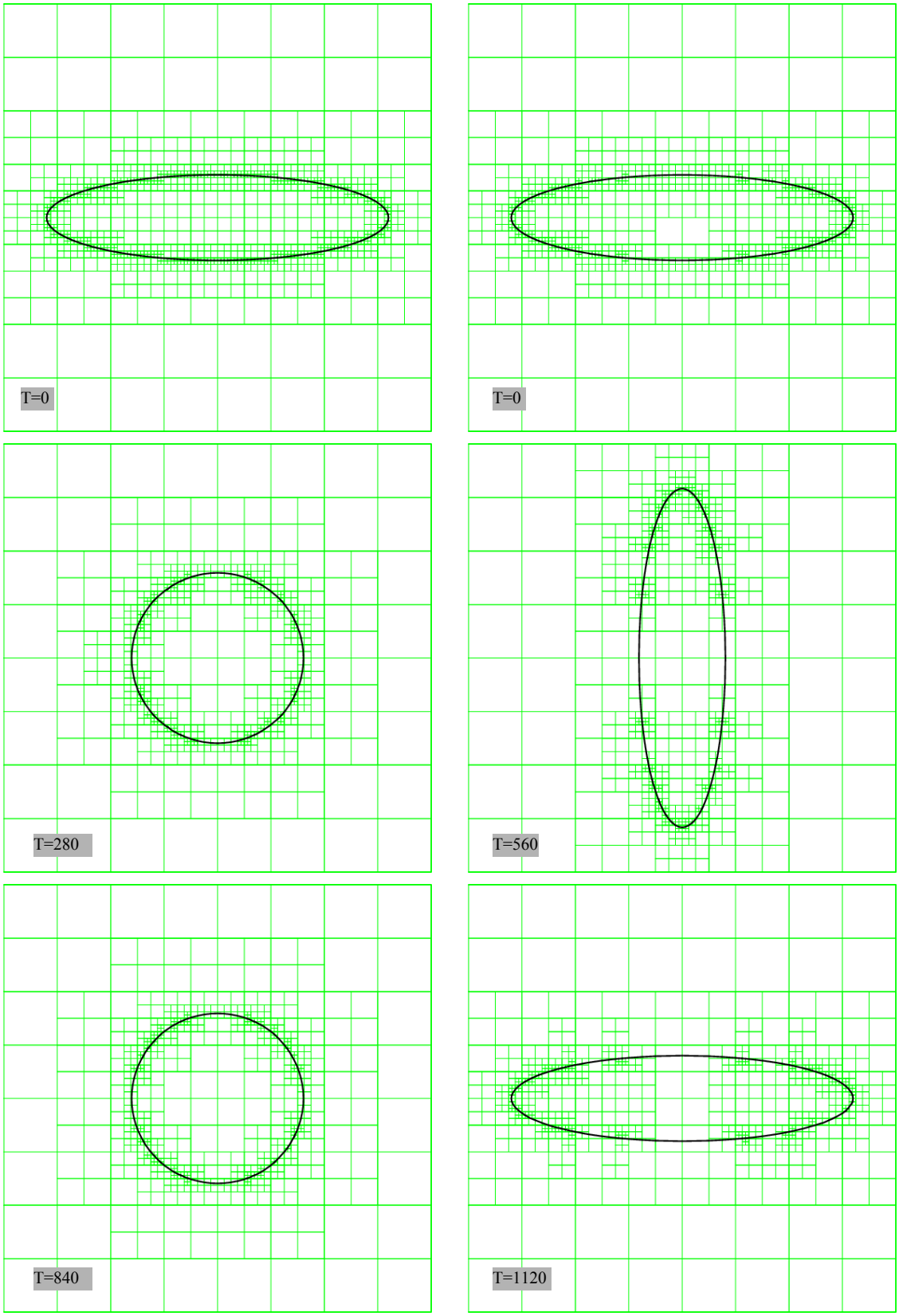

Figure 8: An elongational velocity field. The initial ellipse was stretched until t=560, and then the velocity field was reversed until t=1120.

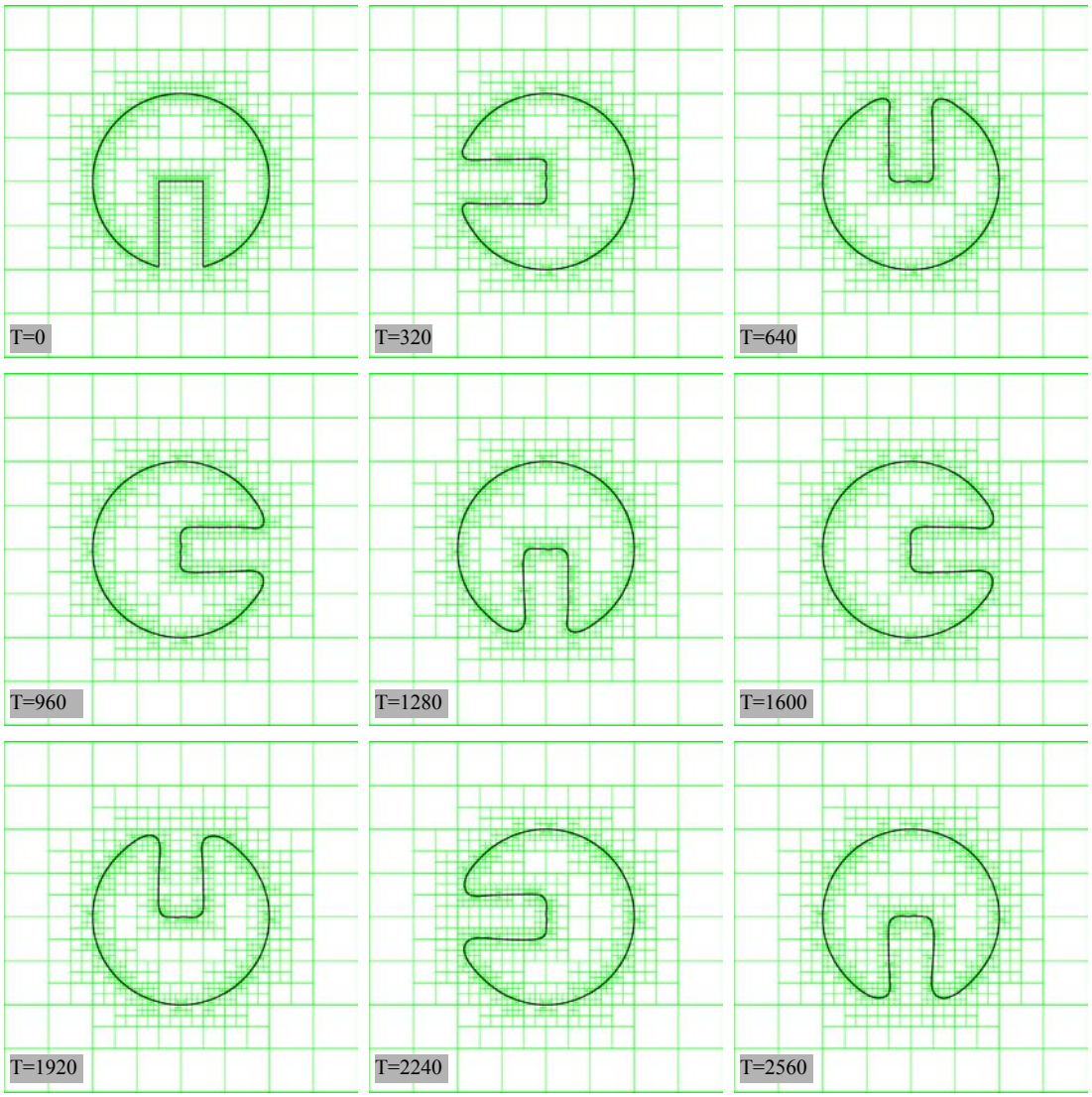

Figure 9: *A slotted disk rotated 360 degrees clockwise, then 360 degrees counter-clockwise.*

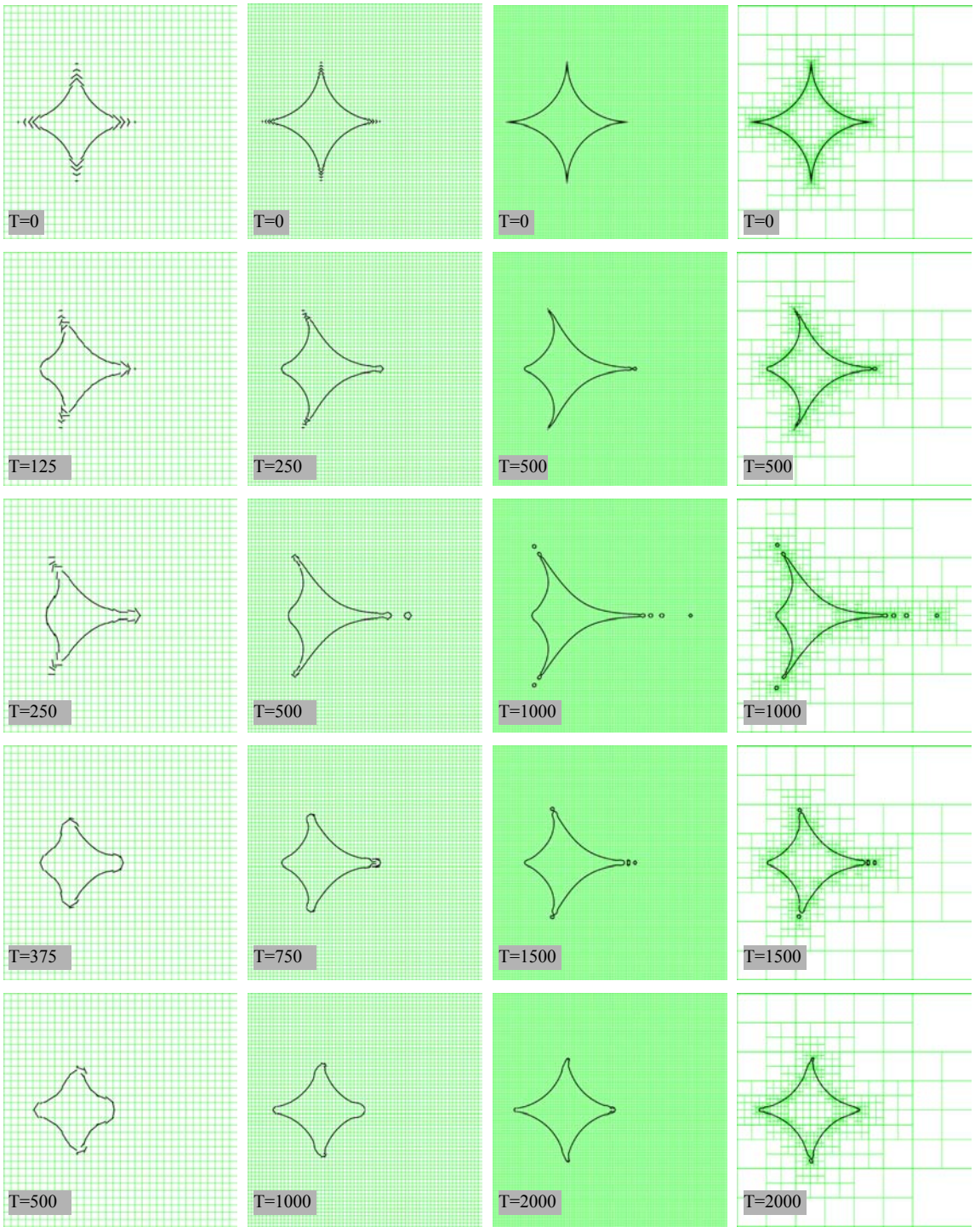

Figure 10: *Star stretching, at grid resolutions of 32*32, 64*64, 128*128, and on an adaptive grid.*