Eurographics Workshop on Natural Phenomena (2007)
D. Ebert, S. Mérillou (Editors)

Fast Fluid Simulation Using Residual Distribution Schemes

Jason Sewall[†], Paul Mecklenburg, Sorin Mitran, and Ming Lin

University of North Carolina at Chapel Hill

Abstract

We present a fast method for physically-based animation of fluids on adaptive, unstructured meshes. Our algorithm is capable of correctly handling large-scale fluid forces, as well as their interaction with elastic objects. Our adaptive mesh representation can resolve boundary conditions accurately while maintaining a high level of efficiency.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Computer Graphics]: Types of Simulation—Animation;

1. Introduction

Fluid phenomena play important roles in everyday life — as blowing winds, jet streams, chemical dispersion, granular flows, ocean waves and currents, and so on. Although these phenomena are commonplace, they are fascinating, visually and physically, for the effects they produce. Mathematical models that describe them properly are nonlinear and lead to computational simulation processes that are very complex and challenging to perform efficiently; the intricate interplay of essential processes such as convection, diffusion, turbulence, surface tension, and their interaction with rigid and deformable solids demands careful attention to stability, temporal and spatial scales, and domain representations.

In this paper, we present an efficient method for physically-based animation of fluids that is also suitable for capturing fluid interaction with elastic solids at large scales of space. The two-way interaction of fluids and elastic bodies is unpredictable and visually interesting. We have developed a simple and efficient method for fluid simulation that also can capture these large-scale fluid phenomena and interaction in complex scenes.

Main Results: We investigate the applicability of *Residual Distribution Schemes* (RDS) for physically-based animation of fluids, that may also interact with solids. These schemes were initially introduced in computational fluid mechanics by Roe [Roe87] as multidimensional methods that do not require a Riemann solver. Residual distribution schemes have not been considered for use in computer graphics, but they exhibit a number of attractive properties:

- They are inherently parallel. The scheme is organized as a loop over computational cells. Each computational cell sends updates to nodal values. The process allows massive parallelization.
- Users can balance accuracy versus cost. In a basic form the schemes are first order in space and time. Higher-order accuracy can be imposed locally or overall by either carrying out multiple iterations over the cells or by higher-order (and more costly) interpolation of physical variables in a single cell.

Furthermore, RDS is capable of describing multi-physics applications. Different physical laws can be defined in each cell; although we have not exploited this property in this paper, it is an attractive property in our consideration when selecting appropriate mathematical formulations.

In this work, we also use an unstructured, adaptive tetrahedral mesh to represent our computational domain and effectively capture boundary conditions. We demonstrate our system on large-scale environments under high-energy forces — a strong wind rocking a bridge, skyscrapers bow-

[†] e-mail: sewall@cs.unc.edu

ing and twisting on a windy day, and a space station deforming in a flow of solar particles.

Organization: The rest of the paper is organized as follows: Section 2 presents a brief review of related work, section 3 presents the theory behind RDS and our model in detail, section 4 describes how our fluid simulation method is coupled with deformable body dynamics, followed by a presentation of results. We discuss some limitations of our approach and conclude with a discussion of potential future research directions.

2. Previous Work

In this section, we briefly discuss related work in computational fluid dynamics, residual distribution schemes, and modeling of deformable solids.

2.1. Computational Fluid Dynamics in Graphics

Realistic animation of fluids has been a topic of considerable interest in computer graphics. Among the first work on visual simulation of fluid dynamics was that of Foster and Metaxis, in which finite-difference methods were used to simulate free-surface flows modeled by the full 3D Navier-Stokes system of equations. Stam addressed the standard timestep restrictions due to CFL conditions in his pioneering work "Stable Fluids" [Sta99], introducing semi-Lagrangian advection to the graphics community. This work has made robust simulation of realistic fluid phenomena possible and popular. Numerous enhancements, such as particle level set methods for modeling free surfaces, followed in [FF01]. We refer the readers to the detailed surveys in [SY05, SSK05, BFMF06].

More recently, there has been an increasing desire to model fluid-structure interaction to achieve still more complex visual effects. Genevaux et. al. [GHD03] presented impressive results with a method for fluid interacting with objects represented as particles and Carlson, et. al. [CMT04] described an efficient and elegant method for modeling two-way coupling of fluid and rigid bodies using a finite-difference framework with the Distributed Lagrange Multiplier method. Their approach achieves impressive and beautiful results, but is mainly designed to handle the interaction of fluid with rigid bodies.

Fluid dynamics on irregular grids with finite-volume discretization was introduced around the same time by several authors [FOK05, ETK*07, WBOL07]. Subsequent work by Klingner et al. [KFCO06] models fluid interaction with moving boundary conditions by re-meshing the domain at each time step and projecting the field variables from the old mesh to the new. Recent work [CGFO06] extends the approach of Klingner et al. to handle the coupled simulation of fluids and elastic bodies with an implicit reformulation of the associated equations, and Batty et. al. [BBB07] augment the hybrid particle-grid approach of Zhu et. al. [ZB05] with a variational coupling of rigid body kinetics to the pressure correction step frequently used to simulate incompressible fluids.

2.2. Residual Distribution Schemes

Residual distribution schemes (RDS) were first presented in [Roe82] and subsequently developed in [Roe87, SDR91] for the Euler equations. Their applicability to hyperbolic systems of partial differential equations has led to their adoption in the aeronautics community, but RDS have also been adapted to solve other classes of equations, including incompressible Navier-Stokes [vdWDID99]. A thorough review of this area was recently presented by Abgrall [Abg06]. Despite their popularity in the aeronautics community, RDS have hitherto not been investigated for computer graphics.

2.3. Simulating Elasticity in Graphics

The modeling of deformable bodies have been heavily studied in computer graphics for more than three decades. We refer the readers to the recent survey for more detail [NMK*05, GM97]. In this paper, we model deformable objects with linear elasticity using a Galerkin finite-element formulation (see [Hug00] for an excellent introduction to the topic).

3. Residual Distribution Schemes in Flow Simulations

We begin this section with an overview of our method, then describe how the equations of fluid dynamics may be solved with residual distribution schemes.

We solve Euler's equations of compressible, inviscid fluid dynamics with Roe's residual distribution schemes (RDS) [Roe87]. These are simple, narrow stencils applied to an unstructured grid for the solution of hyperbolic systems of partial differential equations. The residual of the governing equation is calculated over each simplicial element and distributed to each adjacent vertex (which we also refer to as *nodes*) every time step, then the accumulated residual contributed to each vertex is integrated in time and the procedure is repeated. The solution for each simplex in the computational grid is essentially independent for a given time step, affording a highly parallel solution; this proves to be very efficient at solving the expensive Euler equations of gas dynamics.

3.1. Overview of Fluid Solver

The system described by the Euler equations is a simplification of the general Navier-Stokes equations of fluid dynamics where the fluid is assumed to be compressible and inviscid. In terms of conservative variables, the system is as follows:

$$\begin{aligned}\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\mathbf{u})=0\\ \frac{\partial\rho\mathbf{u}}{\partial t}+\nabla((\rho\mathbf{u})\,\mathbf{u})+\nabla p=0\\ \frac{\partial\varepsilon}{\partial t}+\nabla\cdot(\mathbf{u}(\varepsilon+p))=0\end{aligned}\tag{1}$$

Here ρ is density, $\mathbf{u}$ the velocity vector, p pressure, and ε the total energy. For simplicity and efficiency, typical visual simulations of gaseous phenomena, such as air and the transport of smoke, further simplify (1) to be incompressible, i.e. to assume ρ is constant in space and time and introduce the zero-divergence condition $\nabla\cdot\mathbf{u}=0$ often used for models of water. While the resulting inviscid, incompressible system does not correctly model many physical phenomena, for low levels of energy, noticeable compression is unlikely to occur and the results are physically plausible.

The introduction of the divergence-free condition has far-reaching ramifications on the character of the system and how it can be efficiently solved. The Euler equations (1) form a strictly *hyperbolic* system of PDEs, while the imposition of the divergence-free condition adds an *elliptical* character to the system.

Put simply, the unmodified Euler equations model the propagation of perturbations at finite speeds (the speed of sound, to be specific) — we are guaranteed that perturbations will be local in a given time interval. Perturbations in the system with the zero-divergence condition travel with infinite speed; a change in one part of the spatial domain has instantaneously affect all other parts of the domain.

The Euler equations (1) are naturally amenable to local solution stencils, while the zero-divergence condition mandates the global pressure projection step used in many contemporary fluid simulation methods. Residual Distribution Schemes (RDS) were developed by Roe [Roe87] to take advantage of the hyperbolic character of the Euler equations; independent stencils are applied at each simplex at each time step. The spatial dependency of the method is minimal and it is well-suited to parallel computation.

Consider a slightly idealized example of driving a car in traffic, where what you do depends entirely on the car ahead of you. If the car ahead slows, you too must slow and if the car ahead of you accelerates, you too must accelerate. There is always a delay involved — you must first observe the change in acceleration of the car in front of you before you adjust your own. The effect of this latency can be great with distance; consider a 'column' of cars stopped at an intersection. When the light turns green, the first car accelerates, then the second car, and so forth. Cars approaching the back of this 'column' must slow down even while the cars at the front are accelerating. This is roughly analogous to a compressible system, where the space between cars is variable. If were to impose a fixed length between all cars (an 'incompressible system'), there can be no delay between changes in velocity (i.e. acceleration) of adjacent cars; their velocity must be uniform and *change uniformly*. Thus, information about changes in velocity must be passed through the system instantaneously.

The implication is that cars in 'compressible' traffic have local behavior — their behavior is governed entirely by the car ahead of them — while the cars in 'incompressible' traffic accelerate based on the state of the entire system. It should be clear that the former type of system is more amenable to parallelization than the latter.

3.2. Residual Distribution Schemes

RDS apply to *conservation laws* of the form

$$\mathbf{q}_t+\nabla\cdot\mathbf{F}(\mathbf{q})=0\tag{2}$$

with unknown vector $\mathbf{q}$ and vector flux function $\mathbf{F}$.

The discretization of (2) used in RDS takes the form

$$\mathbf{q}_i^{n+1}=\mathbf{q}_i^n+\frac{\Delta t}{V_i}\sum_T\beta_T^i\Phi_T\tag{3}$$

for a simplex T, where $\mathbf{q}_i^n$ is the solution vector at node i at time n, Δt the timestep used, V_i the dual volume associated with node i, Φ_T the vector of fluctuation (or residual) values of the equations over T, and β_T^i weights specifying how Φ_T is distributed to the nodes of T.

The fluctuation Φ_T is given by

$$\Phi_T=-\int_{V_T}\nabla\cdot\mathbf{F}(\mathbf{q})\,dV=-\oint_{S_T}\mathbf{F}(\mathbf{q})\cdot\mathbf{n}\,dS\tag{4}$$

over simplex T's volume V_T and surface S_T. Since $\mathbf{q}$ varies linearly over each simplex T, (4) simplifies to

$$\Phi_T=-V_T\frac{\partial\mathbf{F}}{\partial\mathbf{q}}\cdot\nabla\mathbf{q}=-\sum_{i\in T}\mathbf{K}_i\mathbf{q}_i\tag{5}$$

with

$$\mathbf{K}_i=\frac{1}{2D}\frac{\partial\mathbf{F}}{\partial\mathbf{q}}\cdot\mathbf{n}_i=\frac{1}{2D}\mathbf{A}\cdot\mathbf{n}_i\tag{6}$$

where D is the spatial dimension of the system, and $\mathbf{n}_i$ is the inward scaled normal of the face opposite node i in the simplex. The matrix $\mathbf{A}$ represents a problem-specific Jacobian of the *quasilinear* form of the equations; this concept is explained in greater detail in [Abg06].

3.3. The system N-scheme

The vectors β_i of coefficients describe how the fluctuation Φ_T is distributed per simplex; specific variants of RDS have different procedures for computing these coefficients. We use the N-scheme (N for 'narrow'), which is the simplest such scheme that enforces upwinding, preserves linearity, and is monotonic. For a system, it is as follows:

$$\beta_T^i = \frac{\gamma_T^j}{\Phi_T}, \quad \sum_{j \in T} \gamma_T^j = \Phi_T, \tag{7}$$

$$\left(\sum_{j \in T} \mathbf{K}_j^+ \right) \gamma_T^j = \mathbf{K}_i^+ \sum_{j \in T} \left[\mathbf{K}_j^- \left(\mathbf{q}_i^n - \mathbf{q}_j^n \right) \right]. \tag{8}$$

Here $\mathbf{K}^+$ and $\mathbf{K}^-$ are products of the diagonalization of $\mathbf{K}$

$$\mathbf{K} = \mathbf{R}\Lambda\mathbf{R}^{-1} = \mathbf{R}(\Lambda^+ + \Lambda^-)\mathbf{R}^{-1} = \mathbf{K}^+ + \mathbf{K}^- \tag{9}$$

with Λ^+ containing the positive eigenvalues of $\mathbf{K}$ and Λ^- the negative.

3.4. Discretization of the Euler equations in 3D

We reform the Euler equations (1) as a conservation law (2).

$$\mathbf{q} = \begin{bmatrix} \rho & l & m & n & \varepsilon \end{bmatrix}^T \tag{10}$$

$$l = \rho u, \; m = \rho v, \; n = \rho w \tag{11}$$

(u, v, w) are the components of velocity. In 3D, Eqn. (2) has the form $\mathbf{q}_t + \mathbf{f}_x(\mathbf{q}) + \mathbf{g}_y(\mathbf{q}_z) + \mathbf{h}_z(\mathbf{q}) = 0$. The flux functions for the Euler equations $\mathbf{f}(\mathbf{q})$, $\mathbf{g}(\mathbf{q})$, $\mathbf{h}(\mathbf{q})$ are defined as:

$$f = \begin{bmatrix} l \\ p + l^2/\rho \\ lm/\rho \\ ln/\rho \\ \frac{l(p+\varepsilon)}{\rho} \end{bmatrix} \; g = \begin{bmatrix} m \\ ml/\rho \\ p + m^2/\rho \\ mn/\rho \\ \frac{m(p+\varepsilon)}{\rho} \end{bmatrix} \; h = \begin{bmatrix} n \\ nl/\rho \\ nm/\rho \\ p + n^2/\rho \\ \frac{n(p+\varepsilon)}{\rho} \end{bmatrix} \tag{12}$$

3.4.1. Similarity Transformations

We would like analytical expressions of the fluid eigenmodes, but these difficult to obtain from the Euler equations in conservation form (1). We can transform to the primitive variables $\mathbf{Q} = (\rho, u, v, w, p)^T$ of the Euler equations through the differential relationship:

$$\frac{\partial \mathbf{q}}{\partial \mathbf{Q}} = \mathbf{M}, \quad \frac{\partial \mathbf{Q}}{\partial \mathbf{q}} = \mathbf{M}^{-1} \tag{13}$$

($\mathbf{M}$ is given in Eqn. (28) in Appendix A) The primitive variable equations are of the form

$$\mathbf{Q}_t + \mathbf{F_Q}\nabla\mathbf{Q} = 0 \tag{14}$$

so we obtain the relationship between Jacobians

$$\mathbf{F_Q} = \mathbf{M}^{-1}\mathbf{F_q}\mathbf{M}, \quad \mathbf{F_q} = \mathbf{M}\mathbf{F_Q}\mathbf{M}^{-1} \tag{15}$$

We assume wave solutions of the form

$$\mathbf{Q} = \mathbf{R} \exp i(\mathbf{k} \cdot \mathbf{x} - \lambda t) \tag{16}$$

with eigenvectors $\mathbf{k}$, eigenvalues λ, and $\mathbf{R}$ the right eigenvector matrix from Eqn. (9). This leads to the eigenproblem

$$\left(\mathbf{F_Q}\mathbf{k}\right)\mathbf{R} = \left[\mathbf{M}^{-1}(\mathbf{F_q}\mathbf{k})\mathbf{M}\right]\mathbf{R} = \lambda\mathbf{R} \tag{17}$$

3.4.2. Primitive Variable Eigensystem

We now seek the eigensystem for the matrix $\mathbf{K} = \mathbf{F_Q}\mathbf{k}$,

$$\mathbf{K} = \begin{bmatrix} \mathbf{V}\cdot\mathbf{k} & k_x\rho & k_y\rho & k_z\rho & 0 \\ 0 & \mathbf{V}\cdot\mathbf{k} & 0 & 0 & k_x/\rho \\ 0 & 0 & \mathbf{V}\cdot\mathbf{k} & 0 & k_y/\rho \\ 0 & 0 & 0 & \mathbf{V}\cdot\mathbf{k} & k_z/\rho \\ 0 & k_x a & k_y a & k_z a & \mathbf{V}\cdot\mathbf{k} \end{bmatrix} \tag{18}$$

with $\mathbf{V} = \langle u, v, w \rangle$.

The matrix varies over a computational simplex as a result of the linear variation of the flow variables; to establish a single set of eigenmodes upon which to base an upwinding procedure (as in section 3.3), we must choose an appropriate reference state for the flow variables. It can be shown [Roe87] that the Roe average is the proper choice to ensure discrete conservation, a property crucial to shock capturing. In the following, assume that all flow variables are evaluated at the Roe average (given in Eqn (34) of Appendix A). The eigenvalues are

$$\lambda_+ = (\mathbf{v}\cdot\mathbf{k} + c) > \lambda = \mathbf{v}\cdot\mathbf{k} > (\mathbf{v}\cdot\mathbf{k} - c) = \lambda_- \tag{19}$$

with λ having an algebraic multiplicity of 3, and the right eigenvector matrix is

$$\mathbf{R} = \begin{bmatrix} \rho & 1 & 0 & 0 & \rho \\ k_x c & 0 & -k_y & 0 & -k_x c \\ k_y c & 0 & k_x & -k_z & -k_y c \\ k_z c & 0 & 0 & k_y & -k_z c \\ c^2\rho & 0 & 0 & 0 & c^2\rho \end{bmatrix} \tag{20}$$

3.4.3. K-Matrix Decomposition

Let $\mathbf{n}_i$ be the unit normal vector to face i pointing into the simplex. We seek to identify which of the eigenmodes computed above are inflowing and which are outflowing. This step is needed for proper upwinding of the scheme. The four potential cases are enumerated in Appendix A

3.5. Solution Procedure

Each simplex T has four nodes, and a vector of unknowns for the conservative Euler equations $\mathbf{q}$ is stored at each of these nodes. The fluctuation over a simplex T at a given time step is calculated and split according to the pseudocode given in Figure 1.

3.6. Parallel Application of RDS

Residual distribution schemes are straightforward to parallelize; the pseudocode in Figure 2 describes the procedure

EULERRDS(T)

for each node $i \in$ INCIDENTNODES(T)
    do ▷ As in section (3.4.1)
        $\mathbf{Q}_i \leftarrow$ PRIMITIVEVARIABLES($\mathbf{q}_i$)
        ▷ Roe parametrization
        $\mathbf{Z}_i \leftarrow \sqrt{\rho_i}\langle 1, u_i, v_i, w_i, H_i \rangle$
▷ Compute Roe average (See Appendix A)
$\bar{\mathbf{Z}} \leftarrow \frac{\Sigma_i \mathbf{Z}_i}{4}$
for each node $i \in$ INCIDENTNODES(T)
    do ▷ As in eq. (19)
        $\Lambda_i \leftarrow$ EIGENVALUES($\bar{z}$, $\mathbf{n}_i$)
        ▷ As in section 3.4.3
        $(\mathbf{K}_i^+, \mathbf{K}_i^-) \leftarrow$ KMATRIXDECOMP($\Lambda_i \bar{z}$, $\mathbf{n}_i$)
        ▷ Compute reference inflow state $\bar{\mathbf{Q}}$
        $\bar{\mathbf{Q}} \leftarrow (\Sigma_i \mathbf{K}_i^-)^{-1}(\Sigma_i \mathbf{K}_i^- \mathbf{Q}_i)$
        $\Phi_i \leftarrow \mathbf{K}_i^+(\mathbf{Q}_i - \bar{\mathbf{Q}})$
        ▷ **M** from Appendix Eq. (28)
        return $\mathbf{M}\Phi_i$

Figure 1: *RDS for the Euler equations*

for a full solve of the Euler equations in parallel (each **for** statement can be applied in parallel over its iterates).

PARALELLFLUIDSOLVE()

for each node $n \in$ *LeafFluidNodes*
    do CLEARACCUMULATOR(n)
▷ implicit barrier
for each simplex $T \in$ *LeafFluidCells*
    do *NodeUpdates* $\leftarrow$ EULERRDS(T)
        for each node $n \in$ INCIDENTNODES(T)
            do ATOMICINC(n, *NodeUpdates*[n])
▷ implicit barrier
for each node $n \in$ *LeafFluidNodes*
    do TIMEINTEGRATE(n)

Figure 2: *Parallel procedure for solving the Euler equations*

The subroutines in Figure 2 are defined as follows.

CLEARACCUMULATOR(n) Clears the **q** accumulator at node n
EULERRDS(T) Computes the fluctuation over simplex T according to Figure 1; returns **q** updates for T adjacent nodes.
INCIDENTNODES(T) as in sec. 3.5
ATOMICINC(n, *NodeUpdate*) Atomically adds *NodeUpdate* to the accumulator at node n
TIMEINTEGRATE(n) Multiplies the quantity in the accumulator of node n by $\Delta t / V_n$ and adds it to n's solution vector

4. Fluid-Solid Interaction

One of the complex and interesting fluid phenomena is interaction with deformable solids. To test the suitability of our simulation method in modeling such visual effects, we will next describe a preliminary system that uses our fluid simulation method based on residual distribution schemes (RDS) and couples it together with the commonly used finite element methods (FEM) for modeling deformable bodies.

4.1. Overview

Our system is composed of several modules working in unison: a fluid solver based on RDS, an elastic solver using a standard FEM formulation, mesh management utilities, and other numerical methods and geometric operations binding these components together. Figure 3 provides an overview of our system and the interaction between these components.

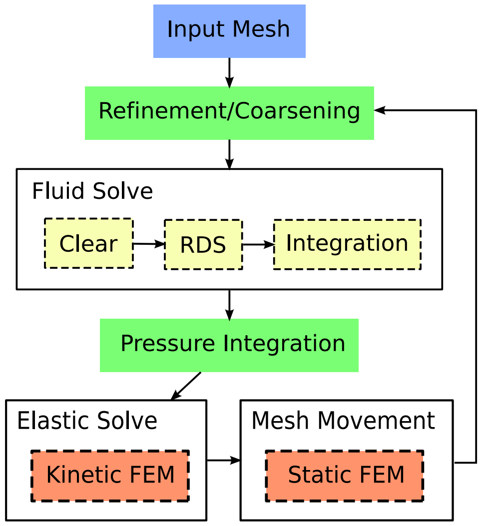

Figure 3: *The structure of our system*

We solve Euler's equations of compressible, inviscid fluid dynamics with Roe's residual distribution schemes [Roe87] as described above. Our elastic solver is a Galerkin formulation of the equations of linear elasticity commonly used in computer graphics; the solution is implicitly integrated in time for stability and the method yields a sparse, symmetric, positive-definite system of equations efficiently handled by iterative solvers.

We use adaptive mesh refinement to focus computational resources on the areas of the simulation most interesting from a standpoint of both visual and dynamic effects. These modules are combined together with a facility for the coupling of the two dynamical systems and an effective method for the updating movement of the computational domain.

4.2. Elasticity Simulation

The elastic bodies in our system are modeled with the equations of linear elasticity; we use a Galerkin finite element method (FEM) formulation as described in [Hug00, NMK*05] to build the stiffness matrix $\mathbf{K}$ for elastic bodies at rest state; this is used to construct the following system for the displacement $\Delta\mathbf{x}$ of each node in the body:

$$\mathbf{M}\Delta\ddot{\mathbf{x}} + \mathbf{K}\Delta\mathbf{x} = \mathbf{F} \tag{21}$$

where $\mathbf{M}$ is a diagonal matrix of the mass associated with the dual volume around each node, $\mathbf{K}$ the linear elastic stiffness matrix as above, and $\mathbf{F}$ the forces acting on each node. We employ a backward Euler temporal discretization as per [BW98] to obtain:

$$\left(\mathbf{M} + \Delta t^2\mathbf{K}\right)\Delta\mathbf{v} = \Delta t\mathbf{F}_{\text{ext}} - \Delta t\mathbf{K}\Delta\mathbf{x} - \Delta t^2\mathbf{K}\Delta\mathbf{v}^n \tag{22}$$

$$\Delta\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta\mathbf{v} \tag{23}$$

$$\Delta\mathbf{x}^{n+1} = \Delta\mathbf{x}^n + \Delta t\mathbf{v}^{n+1} \tag{24}$$

The left-hand side of (22) is a symmetric, positive-definite matrix; its sparsity and block structure is such that we are able to store each compressed row with the associated elastic domain node. We use the conjugate gradient method [She94] to solve for the velocity $\mathbf{v}^{n+1}$ and apply (23, 24) for the resulting displacement.

4.3. Fluid-Structure Coupling

As mentioned previously, we split the solution of our system in time, advancing the fluid in time, then the elasticity. Between these separate solution stages, we propagate the necessary information across domains in the form of boundary conditions. This method is considerably simpler to formulate and solve than an implicitly coupled system as in [CGFO06] and allows the individual solvers to be changed independently.

The force due to pressure on a given surface S is simply:

$$\mathbf{F}_S = \int_S p\mathbf{n}\,dS \tag{25}$$

where p is the pressure along the surface, and $\mathbf{n}$ the surface normal oriented toward the interior of the body under pressure. In our system, Eqn. 25 is integrated over the dual area on the surface of the fluid-solid interface surrounding each node. For simplicity's sake, we assume that the pressure p is constant over this area; the formula for the force $\mathbf{f}_i$ due to the pressure on node i with dual surface area A_i and incident faces enumerated by j is then:

$$\mathbf{F}_i = A_i\mathbf{n}_i\frac{1}{|j|}\sum_j p_j \tag{26}$$

The effect of a solid body's motion on the surrounding fluid is obtained by simply setting the fluid velocity of each node on the fluid-structure boundary to be the velocity of the body at the point.

4.4. Adaptive, Semi-Regular Simplicial Meshes

We use unstructured simplicial meshes to represent the fluid and elastic domains and have developed a robust and efficient method for managing the geometric and simulation data used in the numerical methods.

We provide a coarse initial mesh with minimal boundary information and subdivide the mesh as needed to efficiently and accurately represent the solution; the refinement criteria for this process can range from geometric predicates to more sophisticated, domain-specific approaches based on the solution state during simulation. Currently, we only refine cells to enforce volume constraints, but we are investigating schemes based on solution configurations.

4.4.1. Splitting Scheme

For a given simplex, our subdivision scheme uses the midpoints of each edge along with the original vertices as the vertices of the child simplices. For triangles, we produce 4 similar triangles of $\frac{1}{4}^{\text{th}}$ the area of the original, and for tetrahedra, we produce 4 similar tetrahedra incident on each of the original vertices, leaving an octahedron with the new edge-midpoint vertices as its vertices. We further divide this octahedron into 4 tetrahedrons; each of these children are $\frac{1}{8}^{\text{th}}$ the volume of their parent tetrahedron.

4.4.2. Representation

Our adaptive mesh cannot guarantee on the order in which the simplices and vertices (which we call *cells* and *nodes* respectively, to emphasize their role in the solution of a physical system) are created. Thus, for efficiency's sake, we explicitly track each face and edge in addition to cells and nodes to facilitate the quick location of child nodes and incident faces, edges, and cells. An additional advantage of this practice is that we are able to efficiently and directly operate on each of these computational elements without having to locate them by searching incidence information and deal with the inevitable multiplicity of reference.

4.5. Mesh movement

One difficulty in coupling fluid and solid dynamics lies with the way they are typically formulated; the most popular equations describing the behavior of fluids are Eulerian formulations, while elasticity is most naturally described with a Lagrangian formulation. If not carefully handled, difficulty will arise in solving the fluid equations as cells are inverted and become co-located by the changing boundary conditions.

To reconcile these opposing models, we adapt our fluid mesh to capture the moving boundary conditions due to the motion of solid bodies. The motion of the actual fluid-solid interface is completely described by the displacements Δx of the solid body; we are left to determine how to best move the internal fluid nodes to maintain correct meshes. This step is achieved by treating the mesh itself as an elastic body and solving for the node displacements of the internal fluid nodes with a steady formulation of elasticity. This can be obtained by solving the simple system:

$$\mathbf{K}\Delta\mathbf{x} = 0 \tag{27}$$

The displacements of the solid body (see 4.2) and the domain boundaries provide Dirichlet-type boundary conditions for (27), yielding a system that can be efficiently solved via the method of conjugate gradients. Figure 4 demonstrates our mesh movement scheme for a simple 2D configuration. There are situations where this approach does not work well. In particular, topological changes in the fluid domain would require more sophisticated mesh management; one very viable fix for this problem would be to re-mesh the domain whenever deformations become severe enough to invaldidate tetrahedra, as in [KFCO06].

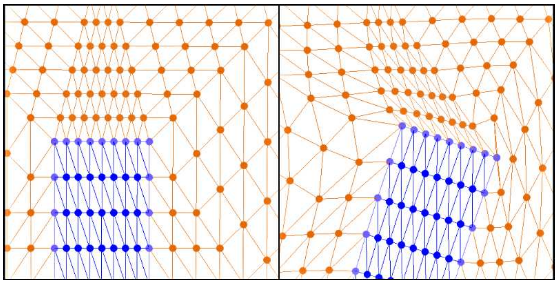

Figure 4: *An elastic body and enveloping mesh at rest (on left) and after deformation/mesh movement (right)*

5. Implementation and Results

Our mesh management and simulation code was developed in C++ and the multi-threaded components parallellized with OpenMP (see [OPE05]). The tetrahedral meshes were generated using the Tetgen package (see [Si04]). We performed our modeling and animation with Blender [ble], and rendering was performed with a combination of Blender and the open-source RenderMan-like renderer, Pixie [Pix].

5.1. Benchmarks and Performance

We have tested and applied our method to a number of challenging problems with applications in computer animation: (a) air current speeding past an iconic bridge, rocking it back and forth, (b) wind buffeting a skyscraper, causing it to bend and twist, and (c) a flow of solar particles passing over a space station suspended high above the Earth. The numbers of tetrahedra listed are for the input meshes given to the solver.

- **Bridge**: The first benchmark scene is shown on the cover page in Fig. A. the bridge and the fluid domain are composed of 31,478 tetrhedral elements.
- **Skyscrapers**: In the skyscrapers benchmark as shown in Fig. A, the buildings and the fluid domain are composed of 9,088 tetrahedral elements.
- **Space Station**: For the space station benchmark in Fig. 8, the space station and the fluid domain are composed of 25,129 tetrahedral elements.

Scene	**# cells**	**secs/frame**		
		Fluid	Solid	Total
Bridge	31k	0.6	5.73	6.36
Skyscrapers	9k	0.15	4.77	4.92
Space station	25k	0.46	14.53	14.99

Table 1: *Runtime performance for each benchmark.*

Table 1 shows the runtime performance achieved by our prototype implementation on the three benchmarks. The timings were collected on a Pentium D 3.4GHz processor with 2 GB of RAM. Our fluid simulation using RDS runs in real time. The dominating computational cost in our simulator is due to FEM simulation of deformable soids.

5.2. Scaling

To demonstrate the scalability of RDS, we have implemented our algorithm (as described in Figure 2) with the parallization facilities provided by OpenMP. This model of parallel computing is well-suited to the multi-core, shared-memory architectures commonly available on desktop workstations and laptops. It will also be directly applicable to many-core architectures. We achieve near-linear scaling for up to 8 processors on the skyscraper model (see Figure 5).

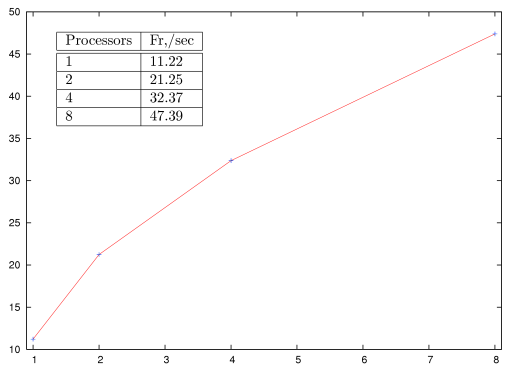

Figure 5: *Linear performance scaling of residual distribution schemes for the Euler equations over the Skyscraper scene.on an SGI Altix cluster.*

6. Summary and Conclusion

We have introduced residual distribution schemes (RDS) to computer graphics as method for efficiently simulating high-energy fluids on modern architectures. We demonstrate that

RDS are computationally attractive in several regards; they can effectively model multi-physics phenomena, such as two-way coupling between fluids and solids. It offers a natural balance between efficiency and accuracy. Our method also takes advantage of adaptive mesh refinement to focus computational efforts on areas of visual and physical significance. Therefore, it is able to deform the computational domain and avoid inaccuracies due to inverted computational cells.

6.1. Limitations

Our method also has a few limitations. Our mesh adaptation scheme assumes limited solid movement and would require re-meshing to handle arbitrary object motion (in particular, topological changes to the computational volume). The N-type residual distribution scheme we use is linear and thus suffers from diffusion; this could be addressed with a non-linear scheme such as Low-Diffusion Advection (LDA) and Positive-Streamwise Invariant (PSI) schemes (see [Abg06]).

6.2. Future Work

There are a number of directions for our future work. Our mesh deformation and adaptation methods could benefit from some refinement. We would like to investigate multiple splitting schemes and more advanced criteria for performing the splits. We would also like to investigate alternatives to remeshing gross deformations in the computational mesh. There are many options for future work with residual distribution schemes. We would like to combine them with a Poisson solver to allow for the simulation of incompressible fluid. We would like to investigate higher-order and less diffusive distribution schemes, such as the LDA and PSI methods described in the aeronautics community. We are currently investigating the potential application of RDS to the equations of elasticity for a more integrated approach to fluid-solid interaction, and we would like to investigate some different time integration schemes for RDS as well.

Due to the inherent parallelism of RDS, we also plan to implement parts of our algorithm on GPUs and future many-core architectures, as well as other new commodity parallel architectures, to exploit RDS' computational properties and further improve its overall performance. We hope to achieve at one to two orders of performance gain, making this method interactive on desktop workstations or mobile platforms.

References

[Abg06] ABGRALL R.: Resigual distribution schemes: Current status and future trends. *Computers & Fluids*, 35 (2006), 641–669.

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. In *To appear in the proceedings of SIGGRAPH 2007* (2007).

[BFMF06] BRIDSON R., FEDKIW R., MULLER-FISCHER M.: Fluid simulation: Siggraph 2006 course notes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM Press, pp. 1–87.

[ble] Blender 2.43 — http://www.blender.org/ (april 2007).

[Bra77] BRANDT A.: Multi-level adaptive solutions to boundary-value problems. *Mathemetics of Computation 31*, 138 (April 1977), 333–390.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 43–54.

[CGFO06] CHENTANEZ N., GOKTEKIN T. G., FELDMAN B. E., O'BRIEN J. F.: Simultaneous coupling of fluids and deformable bodies. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2006), ACM Press/Addison-Wesley Publishing Co.

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. 23*, 3 (2004), 377–384.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 736–744.

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (2007), 4.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 23–30.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[FOK05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM Press, pp. 904–909.

[GHD03] GENEVAUX O., HABIBI A., DISCHLER J.-M.: Simulating fluid-solid interaction. In *Proc. Graphics Interface '03* (2003).

[GM97] GIBSON S. F. F., MIRTICH B.: *A Survey of Deformable Modeling in Computer Graphics*. Tech. Rep. TR1997-019, Mitsubishi Electronic Research Labratories, 1997.

[Hug00] HUGHES T. J. R.: *The Finite Element Method—Linear Static and Dynamic Finite Element Analysis*. Dover Publishers, New York, NY, 2000.

[HW65] HARLOW F., WELCH J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids 8* (1965), 21–82.

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 820–825.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM Press, pp. 457–462.

[NMK*05] NEALEN A., M ULLER M., KEISER R., BOXERMANN E., CARLSON M.: Physically based deformable models in computer graphcics (state of the art report). Eurographics Association. Eurographics 05 STAR.

[OPE05] Openmp version 2.5 specification, May 2005.

[Pix] Pixie — http://www.cs.utexas.edu/ okan/pixie/pixie.htm (april 2007).

[Roe82] ROE P. L.: Fluctuation and signals: A framework for numerical evolution problems. In *Proceedings of IMA Conference on Numerical Methods for Fluid Dynamics* (London, 1982), Academic Press, pp. 219–257.

[Roe87] ROE P. L.: *Linear advection schemes on triangular meshes*. Tech. Rep. 8720, Cranfield Institute of Technology, 1987.

[SDR91] STRUIJS R., DECONINCK H., ROE P. L.: Flucuations splitting schemes for the 2d euler equations. *VKI LS 1991-01* (1991).

[She94] SHEWCHUK J.: An introduction to the conjugate gradient method without the agonizing pain. 1994.

[Si04] SI H.: *TetGen, A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator, v1.3 User's Manual*. Tech. Rep. 9, Weierstrauss Institute for Applied Analysis and Stochastics, 2004.

[SSK05] SONG O.-Y., SHIN H., KO H.-S.: Stable but nondissipative water. *ACM Trans. Graph. 24*, 1 (2005), 81–97.

[Sta99] STAM J.: Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 121–128.

[SY05] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Trans. Graph. 24*, 1 (2005), 140–164.

[TW88] TERZOPOULOS D., WITKIN A.: Physically-based models with rigid and deformable components. In *Proc. Graphics Interface '88* (1988), pp. 146–154.

[vdWDID99] VAN DER WEIDE E., DECONNICK H., ISSMAN E., DEGREZ G.: A parallel, implicit, multi-dimensional upwind, residual distribution method for the navier-stokes equations on unstructured grids. *Computational Mechanics 23* (1999), 199–208.

[WBOL07] WENDT J., BAXTER W., OGUZ I., LIN M.: Finite-volume flow simulations in arbitrary domains. *Graphical Models 69*, 1 (2007), 19–32.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM Press, pp. 965–972.

Appendix A: Residual distribution details

Conservative → primitive variable transformation matrix

$$\mathbf{M} = \frac{\partial \mathbf{q}}{\partial \mathbf{Q}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 \\ w & 0 & 0 & \rho & 0 \\ \mathbf{V}^2/2 & \rho u & \rho v & \rho w & 1/\gamma_{\text{-}1} \end{bmatrix} \quad (28)$$

with $\mathbf{V}^2 = u^2 + v^2 + w^2$, $\gamma_{\text{-}1} = \gamma - 1$ ($\mathbf{M}^{-1}$ is easily computed analytically).

The primitive variable Euler equations are

$$\mathbf{Q}_t + \mathbf{F_Q} \cdot \nabla \mathbf{Q} = f \quad (29)$$

with the Jacobian components $\mathbf{F_Q} = A\mathbf{e}_x + B\mathbf{e}_y + C\mathbf{e}_z$

$$\mathbf{A} = \begin{bmatrix} u & \rho & 0 & 0 & 0 \\ 0 & u & 0 & 0 & \frac{1}{\rho} \\ 0 & 0 & u & 0 & 0 \\ 0 & 0 & 0 & u & 0 \\ 0 & a & 0 & 0 & u \end{bmatrix} \quad (30)$$

$$\mathbf{B} = \begin{bmatrix} v & 0 & \rho & 0 & 0 \\ 0 & v & 0 & 0 & 0 \\ 0 & 0 & v & 0 & \frac{1}{\rho} \\ 0 & 0 & 0 & v & 0 \\ 0 & 0 & a & 0 & v \end{bmatrix} \quad (31)$$

$$\mathbf{C} = \begin{bmatrix} w & 0 & 0 & \rho & 0 \\ 0 & w & 0 & 0 & 0 \\ 0 & 0 & w & 0 & 0 \\ 0 & 0 & 0 & w & \frac{1}{\rho} \\ 0 & 0 & 0 & a & w \end{bmatrix} \quad (32)$$

with

$$a = \left[\frac{p}{\rho} - \rho \left(\frac{\partial e}{\partial \rho} \right)_p \right] \left(\frac{\partial e}{\partial p} \right)_\rho^{-1} = \gamma p \quad (33)$$

the last equality being valid for a perfect gas.

Roe average. The Roe average for the Euler equations is a weighted average of the vertex velocities (u_R, v_R, w_R) and enthalpy h_R, the quantities that appear in the eigenmodes. The weights are defined by the square root of the vertex density.

$$u_R = \frac{\sum_{i=1}^5 \sqrt{\rho_i} u_i}{\sum_{i=1}^5 \sqrt{\rho_i}}, \quad (34)$$

with similar expressions for the other variables.

K Matrix Decomposition Cases

1. $(\mathbf{v} \cdot \mathbf{n}_i + c) > \mathbf{v} \cdot \mathbf{n}_i > (\mathbf{v} \cdot \mathbf{n}_i - c) > 0$. All eigenvalues are positive indicating inflow of all wave modes. The splitting is trivial: $\mathbf{K}_i^+ = \mathbf{K}_i, \quad \mathbf{K}_i^- = 0$.
2. $(\mathbf{v} \cdot \mathbf{n}_i + c) > \mathbf{v} \cdot \mathbf{n}_i > 0 \geq (\mathbf{v} \cdot \mathbf{n}_i - c)$. We have $\mathbf{K}_i^+ = \mathbf{K}_{2i}^+$ with $\mathbf{K}_{2i}^+$ given in Appendix Eq. (35) and $\mathbf{K}_i^- = \mathbf{K}_i - \mathbf{K}_i^+$.
3. $(\mathbf{v} \cdot \mathbf{n}_i + c) > 0 \geq \mathbf{v} \cdot \mathbf{n}_i > (\mathbf{v} \cdot \mathbf{n}_i - c)$. We have $\mathbf{K}_i^+ = \mathbf{K}_{3i}^+$ with $\mathbf{K}_{3i}^+$ given in Appendix Eq. (40) and $\mathbf{K}_i^- = \mathbf{K}_i - \mathbf{K}_i^+$.
4. $0 \geq (\mathbf{v} \cdot \mathbf{n}_i + c) > \mathbf{v} \cdot \mathbf{n}_i > (\mathbf{v} \cdot \mathbf{n}_i - c)$. All eigenvalues are negative indicating outflow of all wave modes: $\mathbf{K}_i^+ = 0, \quad \mathbf{K}_i^- = \mathbf{K}_i$.

Inflow/outflow splitting, case 2, 3 inflow matrices.

$$\mathbf{K}_{2i}^+ = \begin{bmatrix} k_{2i}^{(1)} & k_{2i}^{(2)} & k_{2i}^{(3)} & k_{2i}^{(4)} & k_{2i}^{(5)} \end{bmatrix} \quad (35)$$

$$k_{2i}^{(1)} = \begin{bmatrix} \mathbf{v} \cdot \mathbf{n}_i & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad (36)$$

$$\begin{bmatrix} k_{2i}^{(2)} & k_{2i}^{(3)} & k_{2i}^{(4)} \end{bmatrix} = \quad (37)$$

$$\frac{1}{2} \begin{bmatrix} \rho n_{ix} \lambda_+ / c & \rho n_{iy} \lambda_+ / c & \rho n_{iz} \lambda_+ / c \\ 2V_{ni} - n_{ix}^2 \lambda_- & -n_{ix} n_{iy} \lambda_- & -n_{ix} n_{iz} \lambda_- \\ -n_{ix} n_{iy} \lambda_- & 2V_{ni} - n_{iy}^2 \lambda_- & -n_{iy} n_{iz} \lambda_- \\ -n_{ix} n_{iz} \lambda_- & -n_{iy} n_{iz} \lambda_- & 2V_{ni} - n_{iz}^2 \lambda_- \\ c\rho n_{ix} \lambda_+ & c\rho n_{iy} \lambda_+ & c\rho n_{iz} n_{iy} \lambda_+ \end{bmatrix} \quad (38)$$

$$k_{2i}^{(5)} = \begin{bmatrix} -\frac{\lambda_-}{2c^2} & \frac{n_{ix}\lambda_+}{2\rho c} & \frac{n_{iy}\lambda_+}{2\rho c} & \frac{n_{iz}\lambda_+}{2\rho c} & \frac{\lambda_+}{2} \end{bmatrix} \quad (39)$$

$$\mathbf{K}_{3i}^+ = \frac{\lambda_+}{2} \begin{bmatrix} 0 & \frac{\rho n_{ix}}{c} & \frac{\rho n_{iy}}{c} & \frac{\rho n_{iz}}{c} & \frac{1}{c^2} \\ 0 & n_{ix}^2 & n_{ix} n_{iy} & n_{ix} n_{iz} & \frac{n_{ix}}{c\rho} \\ 0 & n_{ix} n_{iy} & n_{iy}^2 & n_{iy} n_{iz} & \frac{n_{iy}}{c\rho} \\ 0 & n_{ix} n_{iz} & n_{iy} n_{iz} & n_{iz}^2 & \frac{n_{iz}}{c\rho} \\ 0 & c\rho n_{ix} & c\rho n_{iy} & c\rho n_{iz} & 1 \end{bmatrix} \quad (40)$$

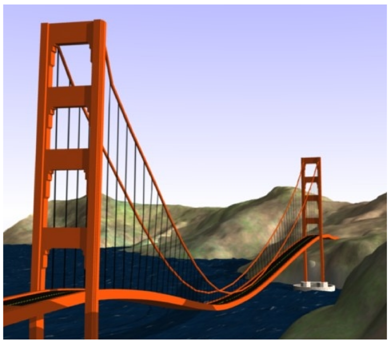

Figure 6: *Simulated flows rocking a suspension bridge.*

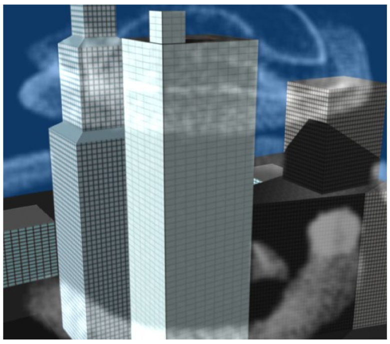

Figure 7: *Skyscrapers in a whirlwind (*$9,088$ *tetrahedra*

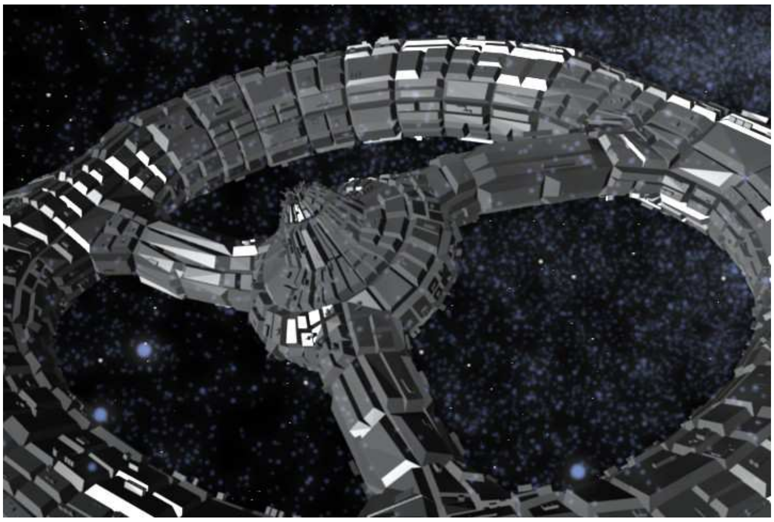

Figure 8: *Twisting Space Station (*$25,129$ *tetrahdra)*