Navier-Stokes Cow

William Olson[a)]
University of Arizona, Department of Physics

(Dated: 4 May 2010)

We set out here to describe the underlying physics and computational methods developed to solve the Navier-Stokes equation for incompressible fluids. This paper and the referenced code *Purple Cow* hopes to provide the insight and tools for others to develop their own applications of these techniques.

I. INTRODUCTION AND OVERVIEW

The goal of this final project is to create a two-dimensional hydrodynamic simulation and graphical representation derived from the Navier-Stokes equations for incompressible flow, including a graphical user interface (GUI), with a specific application to the diffusion of dye in a fluid. The hydrodynamic simulation and the Navier-Stokes equations apply directly to many of the discussions and subjects covered in Theoretical Mechanics of Particles and Continuum. This paper describes the theory of numerically solving the Navier-Stokes equations for incompressible flow, the methods of implementation into code, and results, featuring graphical representations of various scenarios, including diffusion of dye in a fluid. The code was written in C++, using Qt[1] for the construction of the GUI. The source code for the project, named *nscow* and also known as *Purple Cow*, was written by William Olson and released online[2] under the GNU General Public License (GPL)[3].

These smoothed particle simulations have a wide range of applications. These include simulation of the core of our galaxy, such as by Mapelli et al.[4] which allows us insight into otherwise situations that could not be practically simulated with traditional particles. Other applications in particular to manufacturing help us to create everything from better cars to better rockets.

The method behind the simulation generally follows the "stable fluids" method of Stam 1999[5]. The fluid simulation is represented on a grid of cells or pixels. The model requires the following basic constraints. First the volume of fluid is continuous on a two-dimensional rectangular or square domain. No models of free surface boundaries, such as the interface between water and air, are included. The simulation assumes constant density and no-slip boundary conditions. To simulate the behavior of a fluid, the state of the fluid must be mathematically represented at any given time, as a vector field. The Navier-Stokes equations for incompressible flow will be employed for this purpose.

FIG. 1. Output from the completed simulation. A single symmetric, outward force from the middle of the fluid field creates ripples in pressure. Blue represents high pressure and red represents low pressure. The ripple propagates outward as time progresses.

II. THE NAVIER-STOKES EQUATIONS FOR INCOMPRESSIBLE FLOW

The hydrodynamic model requires simplifying assumptions. We assume an incompressible, homogeneous fluid. Incompressibility is valid when the volume of any subregion of the fluid is constant over time. Homogeneity is valid when the density, ρ, is constant in space. Essentially, density is constant over time and space. These assumptions are applicable to a wide range of situations and form a good approximation for most day to day physics[6].

We use a regular Cartesian grid with spatial coordinates $\mathbf{x} = (x, y)$ and time t. The fluid is represented by the velocity field $\mathbf{u}(\mathbf{x}, t)$, and a scalar pressure field $p(\mathbf{x}, t)$. These fields then vary in space and time. If the velocity and pressure are known for the initial time $t = 0$, the state of the fluid over time is described by the Navier-Stokes and continuity equation for incompressible flow:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F} \tag{1}$$

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

where ρ is the (constant) fluid density, ν is the kinematic viscosity, and $\mathbf{F} = (f_x, f_y)$ represents any external forces that act on the fluid. Equation (1) is actually two equations, because $\mathbf{u}$ is a vector quantity: $\mathbf{u} = (u, v)$.

[a)] wtolson@gmail.com; http://code.google.com/p/nscow/

The continuity equation (2) enforces the assumption of incompressibility by declaring that the fluid always has zero divergence.

A. Terms in the Navier-Stokes Equation

We can decompose each 'linear' piece of Navier-Stokes Equation, breaking them down into steps that can later converted to code.

1. Self-Advection

The first term we find and one of the simpler to implement is $-(\mathbf{u} \cdot \nabla)\mathbf{u}$ or the advection of velocity on itself. The velocity of a fluid causes the fluid to transport objects, densities, and other quantities along with the flow. Imagine squirting dye into a moving fluid. The dye is transported, or advected, along the fluids velocity field. In fact, the velocity of a fluid carries itself along just as it carries the dye. The first term on the right-hand side of Equation 1 represents this self-advection of the velocity field and is called the advection term.

2. Pressure

When force is applied to a fluid, it does not instantly propagate through the entire volume. Instead, the molecules close to the force push on those farther away, causing a build up of pressure. Because pressure is force per unit area, any pressure in the fluid leads to acceleration, according to Newtons second law, $\mathbf{F} = m\mathbf{a}$. The second term $-\frac{1}{\rho}\nabla p$, called the pressure term, represents this acceleration.

3. Viscous Diffusion

Viscosity is a measure of how resistive a fluid is to the flow adjacent to itself. This resistance results in an internal tension and diffusion of the momentum (and therefore velocity), so we call the third term $\nu\nabla^2\mathbf{u}$ the diffusion term where ν is the Kinematic viscosity.

4. External Force

The fourth term encapsulates acceleration due to external forces applied to the fluid. These forces may be either local forces or body forces. Local forces are applied to a specific region of the fluidfor example, the force of a fan blowing air or draging a stick through water. Body forces, such as the force of gravity, apply evenly to the entire fluid.

III. SOLVING THE NAVIER-STOKES EQUATIONS NUMERICALLY

The first step in adapting the equations into a form useful in a simulation is to transform the equations into a form better for a straightforward algorithm. With the Navier-Stokes equations the goal is to simultaneously solve for $\mathbf{u}$ and p. The Helmholtz-Hodge Decomposition Theorem is useful multiple times in this process.

A. The Helmholtz-Hodge Decomposition Theorem

Linear algebra states that any vector $\mathbf{v}$ can be described as a set of basis vector components whose sum is $\mathbf{v}$. In the same way, a vector field can be described as a sum of vector fields. Let D be the plane on which our fluid is defined, with a differentiable boundary ∂D with a normal direction $\mathbf{n}$. A vector field $\mathbf{w}$ on D can be uniquely decomposed in the form:

$$\mathbf{w} = \mathbf{u} + \nabla p \tag{3}$$

where $\mathbf{u}$ has zero divergence and is parallel to ∂D. The vector field $\mathbf{w}$ is now the sum of a divergence-free vector field and the gradient of a scalar field, and $\mathbf{u} \cdot \mathbf{n} = 0$ on ∂D. For a proof of this theorem see section 1.3 of Chorin and Marsden, 1993[7].

B. Using the Helmholtz-Hodge Decomposition

In solving Navier-Stokes, there are three computations to calculate the velocity at each time step: advection, diffusion, and force application. Each time this is done it results in a new velocity field, $\mathbf{w}$, with nonzero divergence. But the continuity equation (Equation 2) requires that we end each time step with a divergence-free velocity. To correct this, use the Helmholtz-Hodge Decomposition Theorem to subtract the gradient of the pressure field from the velocity field $\mathbf{w}$.

$$\mathbf{u} = \mathbf{w} - \nabla p \tag{4}$$

This gives an equation for the velocity vector $\mathbf{u}$. However, this equation requires the scalar field of pressure, p. Using Helmholtz-Hodge again, we apply the divergence operator to both sides of equation. This gives:

$$\nabla \cdot \mathbf{w} = \nabla \cdot (\mathbf{u} + \nabla p) = \nabla \cdot (\mathbf{u} + \nabla^2 p \tag{5}$$

From equation (2) we know that $\nabla \cdot \mathbf{u} = 0$, so the above equation simplifies to:

$$\nabla^2 p = \nabla \cdot \mathbf{w} \tag{6}$$

This is a Poisson equation for the pressure of the fluid.

C. Solution for Poisson Equations

The Poisson equation is a matrix equation of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ is the vector of values for which we are solving, in this case, p. The vector $\mathbf{b}$ is a vector of constants, and $\mathbf{A}$ is a matrix. Here, $\mathbf{A}$ is represented in the Laplacian operator ∇^2. The iterative solution technique starts with an initial "guess" for the solution, denoted as $\mathbf{x}^{(0)}$, and each step k outputs an improved solution, $\mathbf{x}^{(k)}$. The iteration technique used here is called Jacobi iteration. The derivation of Jacobi iteration for general matrix equations is found in Golub and Van Loan, 1996[8]. The discrete solution to the Jacobi iteration to solve for pressure at each grid cell used here is:

$$x_{i,j}^{(k+1)} = \frac{x_{i-1,j}^{k} + x_{i+1,j}^{k} + x_{i,j-1}^{k} + x_{i,j+1}^{k} + \alpha b_{i,j}}{\beta} \quad (7)$$

where α and β are constants. For the Poisson-pressure equation, x represents p, b represents $\nabla \cdot \mathbf{w}$, $\alpha = -(\delta x)^2$, and $\beta = 4$. Essentially, "the past" pressure at any given location is used as the initial guess for the Jacobi equation. Solving this differential equation requires initial and boundary conditions. In the simulation, for each time step, the boundary must be updated, taking into account the present pressure and velocity, but also conforming to the boundary conditions. For the fluid simulation, assume the fluid initially has zero velocity and zero pressure everywhere. Because the fluid is simulated on a rectangular grid, assume that it is a fluid in a box and cannot flow through the sides of the box.When solving the Poisson-pressure equation, the boundary conditions require pure Neumann boundary conditions: $\frac{\partial p}{\partial n} = 0$. This means that at a boundary, the rate of change of pressure in the direction normal to the boundary is zero. Later, while solving for velocity, the "no-slip" condition applies, which dictates that velocity goes to zero at the boundaries.

D. Solving for the Velocity Field w

So far the Helmholtz-Hodge Decomposition has led to a method for computing the three unknowns in the Navier-Stokes equations u,v ($\mathbf{u}$), and p. However, a method to solve for the velocity field $\mathbf{w}$ is needed. When finding a numerical expression for $\mathbf{w}$, Helmholtz-Hodge comes in handy again. Linear algebra states that the projection of a vector $\mathbf{r}$ onto a unit vector $\hat{\mathbf{s}}$ is the dot product of $\mathbf{r}$ and $\hat{\mathbf{s}}$. The Helmholtz-Hodge Decomposition allows for the definition of a projection operator, $\mathbb{P}$, that projects a vector field $\mathbf{w}$ onto its divergence-free component, $\mathbf{u}$. Applying $\mathbb{P}$ to equation 3 gives:

$$\mathbb{P}\mathbf{w} = \mathbb{P}\mathbf{u} + \mathbb{P}(\nabla p) \quad (8)$$

By the definition of $\mathbb{P}$, $\mathbb{P}\mathbf{w} = \mathbb{P}\mathbf{u} = \mathbf{u}$, so $\mathbb{P}(\nabla p) = 0$. Applying $\mathbb{P}$ to equation (1) of Navier-Stokes gives:

$$\mathbb{P}\frac{\partial \mathbf{u}}{\partial t} = \mathbb{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{F}) \quad (9)$$

Because $\mathbf{u}$ is divergence-free, the projection operator returns $\mathbf{u}$, and the derivative $\frac{\partial \mathbf{u}}{\partial t}$ is also divergence-free so $\mathbb{P}\frac{\partial \mathbf{u}}{\partial t} = \frac{\partial \mathbf{u}}{\partial t}$. $\mathbb{P}(\nabla p) = 0$, so the pressure term drops out and the follow remains:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbb{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu\nabla^2\mathbf{u} + \mathbf{F}) \quad (10)$$

In the algorithm for simulating fluid flow, first the terms inside the parentheses on the right-hand side are computed: advection, diffusion, and force. This results in a divergent velocity field $\mathbf{w}$ to which the projection operator is applied to get a new divergence-free field, $\mathbf{u}$. This means solving equation for p and then subtracting ∇p from $\mathbf{w}$. In actual implementation, the various components are not computed and then summed as in the above equation. Instead A more detailed discussion of implementation will be covered in the next section.

E. Solving for Self-Advection

To compute the advection of the fluid, how the fluid moves along the velocity field is calculated at each grid cell. A initial idea might be to move the position of the fluid along the velocity field the distance it would travel in the time step of the program. This approach has two issues. First, explicit methods for advection are unstable for large time steps (especially if the magnitude of the velocity vector is greater than the size of the grid cell). The answer to these computational conundrums is rather than advecting the fluid by computing where the particle moves over the time step, instead trace the trajectory of the particle from each grid cell "back in time" to its former position, and then copy the state of the fluid at that "past" position to the starting grid cell. Mathematically, a quantity q is updated per the following equation:

$$q(\mathbf{x}, t + \delta t) = q(\mathbf{x} - \mathbf{u}(\mathbf{x}, t)\delta t, t) \quad (11)$$

This method is stable for arbitrary time steps and velocities[5].

F. Solving for Viscous Diffusion

Viscous fluids have a resistance to flow, which causes in a dissipation of velocity. The partial differential equation for viscous diffusion is given by:

$$\frac{\partial \mathbf{u}}{\partial t} = \nu\nabla^2\mathbf{u} \quad (12)$$

The explicit Euler method for computing viscous diffusion is unstable (similar to the explicit Euler solution to computing advection) for large values of ∂t and ν. An implicit solution is found in Stam, 1999:

$$(\mathbf{I} - \nu\delta t\nabla^2)\mathbf{u}(\mathbf{x}, t+\delta t) = \mathbf{u}(\mathbf{x}, t) \tag{13}$$

Where $\mathbf{I}$ is the identity matrix. This equation happens to be in the form of the Poisson equation for velocity. The same technique for solving the Poisson-pressure equation can be used to solve for viscous diffusion. In the discrete form of the Jacobi iteration solution to the Poisson equation (equation 7), viscous diffusion is calculated by letting both x and b represent $\mathbf{u}$, $\alpha = \frac{-(\delta x)^2}{\nu\delta t}$, and $\beta = 4 + \alpha$.

IV. IMPLEMENTATION

The physics and math of the problem have been described, so now a method of implementation can be formed. The algorithm is summarized by the following pseudocode:

```
// Apply the first 3 operators in Equation 12
advect(u);
diffuse(u);
addForces(u);
```

These three steps results in the divergent velocity field, $\mathbf{w}$. Solve equation 10 for p, and then subtract ∇p from $\mathbf{w}$ (equation 6). These last two calculations can be achieved with the projection operator.

```
// Now apply the projection operator to the
    result.
computePressure(u);
subtractPressureGradient(u, p);
```

The variables u and p contain the velocity and pressure field information. This algorithm represents only one time step. Each step takes a field as its input, and modifies that field as the output. It is important to note that boundary conditions are enforced at each time step. The "fluid in a box" simulation needs no-slip (zero) velocity boundary conditions and pure Neumann pressure boundary conditions. The boundary conditions are implemented by using a virtual extended perimeter of our grid with boundary values calculated when field index is "out of bounds". Details of the implemented algorithms can be found the source file 'NavierStokes.cpp'.

A. Results

The final hydrodynamics simulation is able to produce an arbitrary pixel grid visual representation of a two dimensional fluid. Parameters for viscosity, time step, grid scale, and dye dispersion are all adjustable in real time.

The GUI displays the fluid field in terms of the velocity, pressure, vorticity, and density (of dye). Figure 2 shows the pressure differences in a fluid where forces are applied randomly, while figure 3 shows the velocity field under similar conditions. To examine the behavior of the simulation in a more intuitive way, a common physical situation, a ripple in a pool of water, was modeled. Figure 4 displays the velocity field of a ripple placed near the grid boundary in order to show the reflection of the wave off the wall.

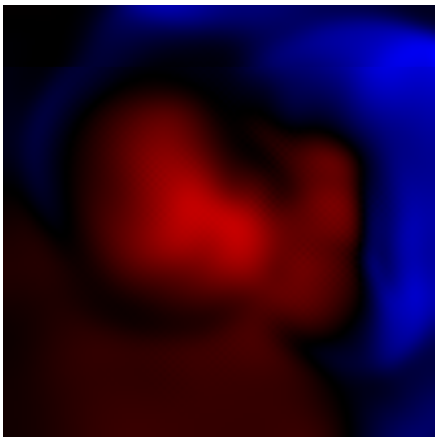

FIG. 2. Shows a 100x100 grid pressure field, where blue is high pressure relative to areas that are low pressure (red). Forces applied to the fluid grid created the pressure difference seen here.

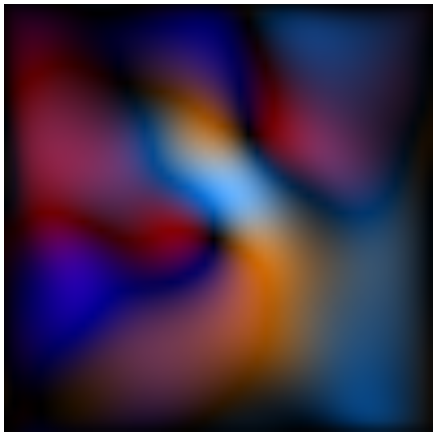

FIG. 3. An example of an output velocity field for a 100x100 fluid grid that has had forces applied to it. In this image dark blue represents motion in $+\hat{x}$, light blue represents motion in $-\hat{x}$, orange represents motion in $+\hat{y}$, and red represents motion in $-\hat{y}$.

Tools for adding dye as well as application of linear or radial forces are provided and can be applied interactively in real time. This provides a flexibility and immersive experience for the user.

FIG. 4. In the velocity field shown here, a force in the lower right-hand corner created a ripple, and then the waves reflect off of the boundaries, where the no-slip (zero) velocity boundary condition is enforced.

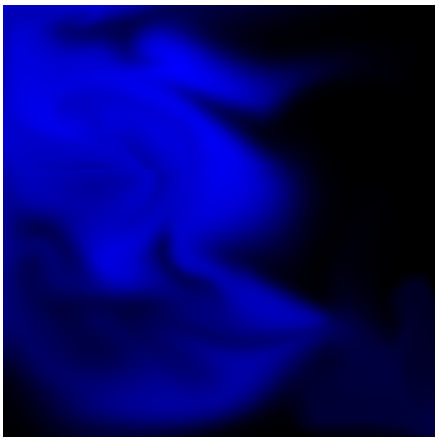

FIG. 5. Shows the density field on a 100x100 pixel grid. The intensity of blue represents $\phi(r,t)$, the concentration of dye "dropped" into the grid. Subsequently forces were applied to the grid to create the swirls in density.

V. APPLICATION

A. The Physics of Dye Diffusion in a Fluid

In addition to the basic hydrodynamic simulation, a specific application of the hydrodynamic code pursued for the purpose of this paper is the simulation of diffusion of particles (dye). There are two parts to the diffusion of dye. First, the dye advects along with the rest of the fluid. Essentially, this means "tagging" certain parts of the fluid and visualizing how they advect within the fluid. Second, diffusion of the particles into the rest of the fluid must be calculated. The diffusion equation describes density fluctuation in a material. This equation for particle diffusion was originally derived by Adolf Fick in 1855[9].

$$\frac{\partial\phi(\mathbf{r},t)}{\partial t} = \nabla\cdot[D\nabla\phi(\mathbf{r},t)] \tag{14}$$

where $\phi(\mathbf{r},t)$ is the density of the diffusing material at location $\mathbf{r}$ and time t and $D(\phi,\mathbf{r})$ is the collective diffusion coefficient for density at any location. In this simulation, D is a constant allowing us to rewrite the equation.

$$\frac{\partial\phi(\mathbf{r},t)}{\partial t} = D\nabla^2\phi(\mathbf{r},t) \tag{15}$$

This equation can yet again be solved by our application of equation (7) with $x = b = \phi$, $\alpha = \frac{(\delta x)^2}{D\delta t}$, and $\beta = 4 + \alpha$. After implementing this equation, the dye should appear to move smoothly and systematically from high-concentration areas to low-concentration areas.

B. Results

The simulation successfully depicts the diffusion of dye in a stable fluid grid, from areas of high-concentration to low concentration. Figure 6 shows the change in the density distribution over time. The diffusion is actually gaussian, as intended. When forces are applied to the fluid with dye dispersed, as in figure 5, advection causes differences in dye concentration. The resullting swirls are reminiscent of creamer in coffee (although at a lower resolution).

VI. CONCLUSION

Hydrodynamic modeling has become part of the larger field of computational fluid dynamics (CFD). Hydrodynamic modeling differs from other CFD specializations in its focus on the movement of water. Hydrodynamic simulations are used in both academia (physics, chemistry, oceanography, geophysics), and industry (gaming, meteorology, aerospace and automotive design, ventilation systems). With this first version of a hydrodynamic simulator there is much room for growth with this code into specialized versions and high performance computational versions of any hydrodynamic environment.

ACKNOWLEDGMENTS

I would like to thank Dr. Srinivas Manne for his patience and guidance as well as Master Sarah Braden for support. I would also like to thank the many people who have shared their work under an open license and have made this work possible[10].

[1] `http://qt.nokia.com/`.
[2] `http://code.google.com/p/nscow/`.

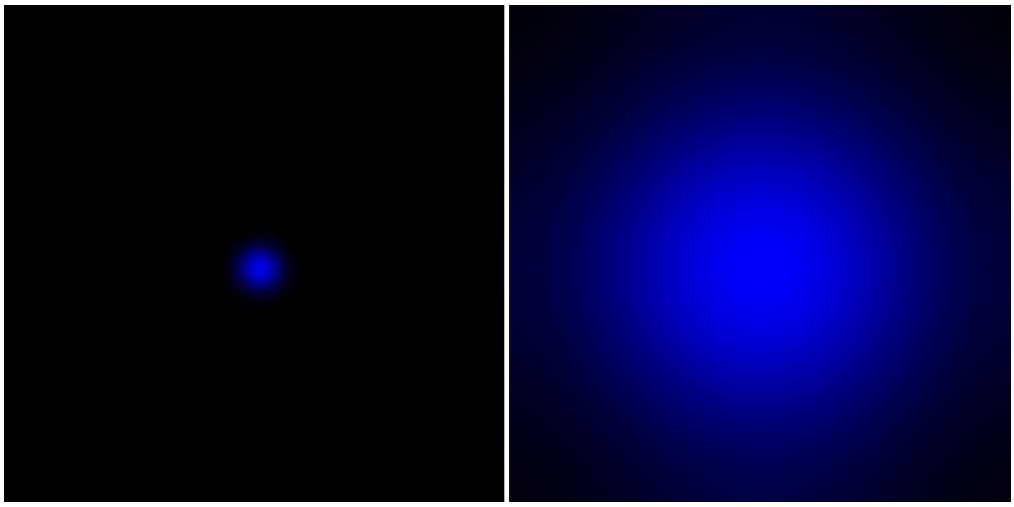

FIG. 6. Shows the density field on a 100x100 pixel grid. The image on the left shows one "group" of dye is inserted into the undisturbed fluid field at time $t = 0$. On the right, at $t = 30$, the dye has mostly dispersed through the fluid field. The intensity of blue represents $\phi(r, t)$, the concentration of dye "dropped" into the grid.

[3] `http://www.gnu.org/licenses/gpl.html`.

[4] M. Mapelli, T. Hayfield, L. Mayer, and J. Wadsley, "In situ formation of the massive stars around SgrA*," (2008), `http://arxiv.org/abs/0805.0185`.

[5] J Stam, "Stable Fluids," in *Proceedings of SIGGRAPH 1999* (1999).

[6] D J Acheson, *Elementary Fluid Dynamics* (Oxford University Press, 1990).

[7] A J Chorin and J E Marsden, *A Mathematical Introduction to Fluid Mechanics*, 3rd ed. (Springer, 1993) p. 172.

[8] G H Golub and C F Van Loan, *Matrix Computations*, 3rd ed. (The Johns Hopkins University Press, 1996).

[9] Adolph Fick, "On Liquid Diffusion," Philosophical Magazine and Journal of Science **9**, 30–39 (1855).

[10] `http://nscow.googlecode.com/svn/trunk/ACKNOWLEDGMENTS`.