

Stable Fluids

(Notes From The Paper)

BY ASHWIN NANJAPPA

Web: <http://www.comp.nus.edu.sg/~ashwinna/>

October 27, 2006

1. Introduction

In computer graphics, the shape and the behaviour of the fluid are of primary interest, physical accuracy is not.

Most fluid solvers become *unstable* when solving for large time steps. This *instability* leads to numerical simulations that *blow-up* and need to be restarted with smaller time steps.

The method in this paper is stable and doesn't blow up.

This model is not accurate for engineering applications. It suffers from *numerical dissipation* i.e., the flow dampens too rapidly compared to real experiments.

To fully understand the fluid dynamics in this paper, read the following:

- A. J. Chorin and J. E. Marsden. *A Mathematical Introduction To Fluid Mechanics*. This is for the mathematically inclined.
- M. B. Abbott. *Computational Fluid Dynamics: An Introduction For Engineers*. For those with an engineering bent of mind.
- N. Foster and D. Metaxas. *Modeling The Motion Of A Hot, Turbulent Gas*.

2. Stable Navier-Stokes

2.1 Basic Equations

A fluid whose density and temperature are nearly constant is described by a *velocity field* \mathbf{u} and a *pressure field* p .

The *spatial coordinate* is denoted by \mathbf{x} , which for 2-dimensional fluids is $\mathbf{x} = (x, y)$ and for 3-dimensional fluids is $\mathbf{x} = (x, y, z)$.

If the velocity and pressure are known at initial time $t = 0$, the evolution of these 2 quantities over time is given by the Navier-Stokes equations:

$$(1): \nabla \cdot \mathbf{u} = 0$$

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

where,

ν : kinematic viscosity of the fluid

ρ : density of fluid

f : external force

Eq 2 is a vector equation for the 2 or 3 components of the velocity field.

\cdot denotes the dot product of 2 vectors.

∇ is a vector of spatial partial derivatives. $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ in 2 dimensions and $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$ in 3 dimensions.

Also, $\nabla^2 = \nabla \cdot \nabla$

The above 2 equations have to be supplemented by *boundary conditions*.

1. *Periodic boundary conditions*: The fluid is defined on an n-dimensional torus. There are no walls, the fluid wraps around.
2. *Fixed boundary conditions*: The fluid lies in a *bounded domain D*. The boundary conditions are given by a function \mathbf{u}_D which is defined on the *domain boundary* ∂D .

In both of the above cases, the normal component of the velocity field at the boundary should be zero i.e., no matter traverses walls.

Combining Velocity And Pressure Fields

According to *Helmholtz-Hodge Decomposition*, any vector field \mathbf{w} can be decomposed into the form:

$$(3): \mathbf{w} = \mathbf{u} + \nabla q$$

where,

\mathbf{u} has zero divergence i.e., Eq 1

q is a scalar field

When ∇ is applied on a scalar field, it results in a *gradient field*.

Hence, *any vector field is the sum of a mass conserving field and a gradient field*.

With this, we define an operator P which projects any vector field \mathbf{w} onto its divergence free part i.e., $\mathbf{u} = P\mathbf{w}$.

We define this operator by multiplying Eq 3 by ∇ on both sides:

$$\nabla \cdot \mathbf{w} = \nabla^2 q$$

This is a *Poisson equation* for the scalar field q with the *Neumann boundary condition* $\frac{\partial q}{\partial n} = 0$ at ∂D .

A solution to Eq 4 is used to compute \mathbf{u} :

$$\mathbf{u} = P\mathbf{w} = \mathbf{w} - \nabla q$$

When this projection operator P is applied to both sides of Eq 2, we get:

$$(5): \frac{\partial \mathbf{u}}{\partial t} = P(-(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + f)$$

where, $P\mathbf{u} = \mathbf{u}$ and $P\nabla p = 0$.

Eq 5 is the fundamental equation from the stable fluid solver will be developed.

2.2 Method Of Solution

Starting from an initial state of $\mathbf{u} = (x, 0)$ the Eq 5 is solved by marching through time with time steps of Δt . This is done in 4 steps:

$$\mathbf{u}(x, t) \rightarrow \mathbf{u}(x, t + \Delta t)$$

$$w_0(x) \xrightarrow{\text{add force}} w_1(x) \xrightarrow{\text{advect}} w_2(x) \xrightarrow{\text{diffuse}} w_3(x) \xrightarrow{\text{project}} w_4(x)$$

2.2.1 External Force

Adding the external force f is easy. The force is assumed to be applied at the beginning of each time step.

$$w_1(x) = w_0(x) + \Delta t f(x, t)$$

2.2.2 Advection

Advection (or convection) of the fluid over itself is given by $-(\mathbf{u} \cdot \nabla)\mathbf{u}$. This term makes the Navier-Stokes equation *non-linear*.

Foster and Metaxas solve this term using *finite differencing*. For large time steps, their method blows up.

This paper uses a technique of solving partial differential equations known as the *method of characteristics*. This method is guaranteed to not blow up for time steps of any size.

In simple terms, this method calculates the velocity at a point x at time $t + \Delta t$ by backtracing from that point through velocity field w_1 over Δt . This defines a path $p(x, s)$ corresponding to a *partial streamline* of the velocity field. The new velocity at point x is set to the value at the backtraced location Δt ago:

$$w_2(x) = w_1(p(x, -\Delta t))$$

The above step needs a *particle tracer* (to backtrace) and a *linear interpolator* (to find velocity value at backtraced location).

2.2.3 Diffusion

The third step solves the effect of viscosity:

$$\frac{\partial w_2}{\partial t} = \nu \nabla^2 w_2$$

This paper uses an implicit method for this:

$$(I - \nu \Delta t \nabla^2) w_3(x) = w_2(x)$$

where,

I is the *identity operator*

∇^2 is the *diffusion operator*

When the diffusion operator is discretized, this leads to a *sparse linear system* for the unknown w_3 .

2.2.4 Projection

The above 3 steps might have taken the field out of the space of divergence free fields. The projection step ensures that the field is divergence free.

This step involves the resolution of the Poisson problem defined in Eq 4:

$$\nabla^2 q = \nabla \cdot w_3$$

$$w_4 = w_3 - \nabla q$$

This Poisson equation when spatially discretized, becomes a *sparse linear system*. This is similar to the diffusion step.

2.3 Periodic Boundaries And The FFT

If we consider a domain with periodic boundary conditions, we can transform the velocity into the Fourier domain:

$$\mathbf{u}(x, t) \rightarrow \hat{\mathbf{u}}(\mathbf{k}, t)$$

In the Fourier domain, the gradient operator “ ∇ ” is equivalent to multiplication by $i\mathbf{k}$, where $i = \sqrt{-1}$.

Hence, both the diffusion and projection steps (step 3 and 4) are simpler to solve.

In the Fourier domain, the diffusion operator is:

$$I - \nu\Delta t\nabla^2 \rightarrow 1 + \nu\Delta tk^2$$

The diffusion can be interpreted as a low pass filter whose decay is proportional to both the time step and the viscosity.

In the Fourier domain, the projection operator is:

$$Pw \rightarrow \hat{P}\hat{w}(\mathbf{k}) = \hat{w}(\mathbf{k}) - \frac{1}{k^2}(\mathbf{k}\cdot\hat{w}(\mathbf{k}))\mathbf{k}$$

where,

$$k = |\mathbf{k}|$$

Operator \hat{P} projects the vector $\hat{w}(\mathbf{k})$ onto the plane which is normal to the *wave number* \mathbf{k} .

The Fourier transform of the velocity of a divergence free field is hence always perpendicular to its wavenumbers.

So, all that is needed now is a particle tracer and a *fast Fourier transform* (FFT).

The complete solver is:

FourierStep($w_0, w_4, \Delta t$):

(1) Add force: $w_1 = w_0 + \Delta t f$

(2) Advect: $w_2(x) = w_1(p(x, -\Delta t))$

Transform to Fourier: $\hat{w}_2 = \text{FFT}\{w_2\}$

(3) Diffuse: $\hat{w}_3(\mathbf{k}) = \frac{\hat{w}_2(\mathbf{k})}{(1 + \nu\Delta tk^2)}$

(4) Project: $\hat{w}_4 = \hat{P}\hat{w}_3$

Transform back from Fourier: $w_4 = \text{FFT}^{-1}\{\hat{w}_4\}$

2.4 Moving Substances Through The Fluid

A non-reactive substance added to the fluid will be both advected and diffused at the same time.

The evolution of this *scalar field* can be described by an *advection diffusion type equation*:

$$\frac{\partial a}{\partial t} = -\mathbf{u}\cdot\nabla a + \kappa_a\nabla^2 a - \alpha_a a + S_a$$

where,

κ_a is a diffusion constant

α_a is a dissipation rate

S_a is a source term

This equation is very similar to the Navier-Stokes equation, it has an advection term, a diffusion term and a force term. So, all these terms are solved in a way similar to solving for the velocity field.

The dissipation term which is not present in the Navier-Stokes equation is solved over a time-step as follows:

$$(1 + \Delta t \alpha_a) a(x, t + \Delta t) = a(x, t)$$

3. Our Solver

3.1 Setup

The general structure of the simulator is:

```
while (simulating)
{
    /* Handle display and user interaction */
    /* Get forces 'F' and sources 'Ssource' from UI */
    swap(u1, u0);
    swap(s1, s0);
    Vstep(u1, u0, visc, F, dt);
    Sstep(s1, s0, kS, aS, u1, Ssource, dt);
}
```

The velocity solver has 4 steps:

```
Vstep(u1, u0, visc, F, dt)
{
    for (i = 0; i < NDIM; ++i)
        addForce(u0[i], F[i], dt);
    for (i = 0; i < NDIM; ++i)
        Transport(u1[i], u0[i], u0, dt);
    for (i = 0; i < NDIM; ++i)
        Diffuse(u0[i], u1[i], visc, dt);
    Project(u1, u0, dt);
}
```

The scalar field solver is also similar:

```
Sstep(s1, s0, k, a, u, source, dt)
{
    addForce(s0, source, dt);
    Transport(s1, s0, u, dt);
    Diffuse(s0, s1, k, dt);
    Dissipate(s1, s0, a, dt);
}
```

The `Transport` routine is an important step which accounts for the movement of the substance due to the velocity field. It resolves the non-linearity of the Navier-Stokes equations.

```

Transport(s1, s0, u, dt)
{
    for each cell (i, j, k) do
        X = 0 + (i + 0.5, j + 0.5, k + 0.5) * D;
        TraceParticle(X, U, -dt, X0);
        S1[i, j, k] = LinInterp(X0, S0);
    end
}

```

The routine `TraceParticle` traces a path through the field U from X over a time $-dt$. $X0$ is where it ends up. For this backtrace, a simple second order *Range-Kutta (RK2) method* is used for the particle trace and an *adaptive particle tracer* (which subsamples the time step only in regions of high velocity gradients) is used.

The routine `LinInterp` linearly interpolates the value of the scalar field S at the location $X0$.

To solve for the diffusion (`Diffuse`) and projection (`Project`), a sparse linear solver from the FISHPAK library is used.

4. Results

5. Conclusions

The authors intended to base their work on Foster and Metaxas' method, but that had instabilities. The solver in this paper is unconditionally stable.

Future work might include simulating fluids with *free boundaries* (like water).

A. Method Of Characteristics

This method can be used to solve advection equations of the type:

$$\frac{\partial a(x, t)}{\partial t} = -v(x) \cdot \nabla a(x, t)$$

where,

a is a scalar field

v is a steady vector field

a_0 is the field at time $t=0$ i.e., $a_0(x) = a(x, 0)$

Let $p(x_0, t)$ be the *characteristics* of the vector field v which flows through the point x_0 at $t=0$:

$$\begin{aligned} \frac{d}{dt} p(x_0, t) &= v(p(x_0, t)) \\ p(x_0, 0) &= x_0 \end{aligned}$$

Let $\bar{a}(x_0, t) = a(p(x_0, t), t)$ be the value of the field along the characteristic passing through the point x_0 at $t=0$.

The variation of \bar{a} over time can be computed using the *chain rule of differentiation*:

$$\frac{d\bar{a}}{dt} = \frac{\partial a}{\partial t} + v \cdot \nabla a = 0$$

This means that the value of the scalar does not vary along the streamlines:

$$\bar{a}(x_0, t) = \bar{a}(x_0, 0) = a_0(x_0)$$

Hence, the initial field and the characteristics together completely define the solution to the advection problem.

The field for any given location and time (x, t) can be computed by tracing back in time the location x along the characteristic to point x_0 , and then evaluating the initial field at that point:

$$a(p(x_0, t), t) = a_0(x_0)$$

This is the method used to solve the fluid advection problem in this paper over the interval $[t, t + \Delta t]$. Here, $v = \mathbf{u}$ and a_0 is the fluid velocity.

B. FISHPAK Routines

□