

PDE Solvers for Fluid Flow

issues and algorithms for the Streaming Supercomputer

Eran Guendelman

February 5, 2002

Topics

- Equations for incompressible fluid flow
- 3 model PDEs: Hyperbolic, Elliptic, Parabolic
 - ★ Examples
 - ★ Solution methods
 - ★ Issues for Streaming Supercomputer
- Solving incompressible fluid flow on a uniform Cartesian grid

Notation

- $\frac{\partial \rho}{\partial t} = \rho_t$ Partial derivative
- $\frac{D\rho}{Dt}$ Material (a.k.a. total) derivative
 - ★ If $\rho = \rho(t, x, y, z)$, then by the chain rule

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x} \frac{dx}{dt} + \frac{\partial \rho}{\partial y} \frac{dy}{dt} + \frac{\partial \rho}{\partial z} \frac{dz}{dt} = \rho_t + \mathbf{u} \cdot \nabla \rho$$

- ∇ Spatial gradient operator
 - ★ e.g. In 3-d: $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$

Notation (cont'd)

- $\nabla \cdot$ Divergence operator
 - ★ e.g. In 3-d: $\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$
- $\Delta = \nabla \cdot \nabla = \nabla^2$ Laplacian operator
 - ★ e.g. In 3-d: $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$
 - ★ It is a generalization of the 1-d curvature operator
- Bold variables represent vector quantities

Equations for Incompressible Fluid Flow

- The equations come from two fundamental physical principles
 - ★ Conservation of mass
 - ★ Conservation of momentum
- Conservation of energy does not factor in unless one is simulating *compressible* fluids

Conservation of Mass

- Conservation of mass leads to the **continuity equation**:

$$\frac{D\rho}{Dt} + \rho (\nabla \cdot \mathbf{u}) = 0$$

- For incompressible fluids:

- ★ Incompressibility \Rightarrow Mass density independent of time $\Rightarrow \frac{D\rho}{Dt} = 0$
- ★ Substituting into the continuity equation gives

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

- ★ We say \mathbf{u} is **divergence free**

Conservation of Momentum

- Expressed in the **Navier-Stokes equations**:

$$\frac{D\mathbf{u}}{Dt} = \mu\Delta\mathbf{u} - \frac{\nabla p}{\rho} + \mathbf{f}$$

- Meaning of each term:

- ★ $\frac{D\mathbf{u}}{Dt}$: We track a particle through the fluid and see how its velocity changes in time
- ★ $\mu\Delta\mathbf{u}$: Acceleration due to fluid viscosity (μ is the viscosity coefficient)
- ★ $-\frac{\nabla p}{\rho}$: Acceleration due to the local pressure gradient
- ★ \mathbf{f} : Acceleration due to forces such as gravity, buoyancy, etc.

- Left hand side is often written in expanded form: $\mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u}$

- Called **Euler equations** if $\mu = 0$ (inviscid)

Equations for Incompressible Fluid Flow

- Four equations in four unknowns (p and the three-component \mathbf{u}):

$$\begin{aligned}\nabla \cdot \mathbf{u} &= 0 \\ \mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u} + \frac{\nabla p}{\rho} &= \mu \Delta \mathbf{u} + \mathbf{f}\end{aligned}$$

- Solving these equations requires being able to solve the three most fundamental types of partial differential equations (PDEs)

The Three Model PDEs

- Partial differential equations are normally classified using 3 model PDEs:
 - ★ Hyperbolic
 - ★ Elliptic
 - ★ Parabolic
- Examples and solution methods for each type will now be discussed

Hyperbolic PDEs

- Time dependent
- Model transient movement of signals along velocity fields
 - ★ *i.e.* At any point, flow is in one direction

Hyperbolic PDEs: Examples

- One-way wave equation for incompressible flow

$$\rho_t + \mathbf{u} \cdot \nabla \rho = 0$$

which represents the advection of density through the fluid

- The equation

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u} = 0$$

which will come up later when we solve the Navier Stokes equations by splitting

- The main level set equation

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0$$

which, although similar in structure to the above equations, requires different (more accurate) solution methods

General Setup for Solving PDEs

- Space is discretized into a uniform Cartesian grid (we will assume $\Delta x = \Delta y = \Delta z$ for simplicity)
- Δt will denote the current time step of the simulation
- In our 1-d examples, $\{\star\}_i^n$ will denote the quantity \star at position $x = x_0 + i\Delta x$, and at time $t = \sum_{k=0}^{n-1} \Delta t_k$ (since the time steps may not be uniform)
- Sometimes I may just use the time index: e.g. \mathbf{u}^n is the velocity field at time step n

Solving Hyperbolic PDEs: Explicit Method

- We will solve the 1-d version of the one-way wave equation

$$\rho_t + u\rho_x = 0$$

- The explicit method solves this PDE by approximating the partial derivatives using finite differences
- We approximate ρ_t using the **forward Euler approximation**:

$$\{\rho_t\}_i^n \approx \frac{\{\rho\}_i^{n+1} - \{\rho\}_i^n}{\Delta t}$$

Solving Hyperbolic PDEs: Explicit Method (cont'd)

- To approximate ρ_x , we first define the **upwind approximations**:

$$D^+ \{\rho_x\}_i^n = \frac{\{\rho\}_{i+1}^n - \{\rho\}_i^n}{\Delta x}$$

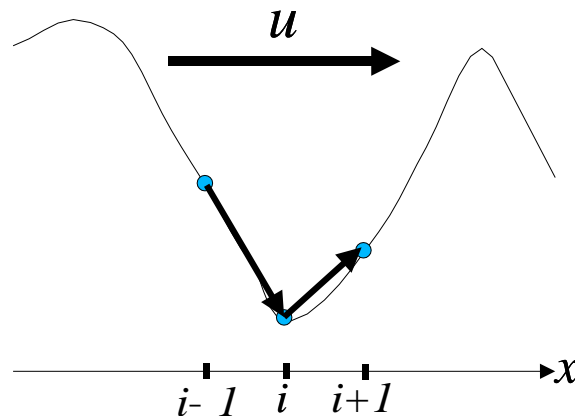
$$D^- \{\rho_x\}_i^n = \frac{\{\rho\}_i^n - \{\rho\}_{i-1}^n}{\Delta x}$$

- Then to approximate ρ_x we use either D^+ or D^- depending on the velocity at that point:

$$\{\rho_x\}_i^n \approx \begin{cases} D^+ \{\rho_x\}_i^n & \text{if } \{u\}_i^n < 0 \\ D^- \{\rho_x\}_i^n & \text{otherwise} \end{cases}$$

Solving Hyperbolic PDEs: Explicit Method (cont'd)

- This is needed in order to account for the direction in which information is flowing



- So for each i we set the updated density according to

$$\{\rho\}_i^{n+1} = \{\rho\}_i^n - \Delta t \{u\}_i^n D^\pm \{\rho_x\}_i^n$$

Boundary Conditions

- We need to specify boundary conditions so that we can evaluate e.g.

$$D^- \{\rho_x\}_0^n = \frac{\{\rho\}_0^n - \{\rho\}_{-1}^n}{\Delta x}$$

- Two types of boundary conditions:

- ★ **Dirichlet:** Specify the value of an out-of-bound term such as $\{\rho\}_{-1}^n$
- ★ **Neumann:** Specify the derivative at the boundary, e.g. $D^- \{\rho_x\}_0^n = 0$

Explicit Method: Issues

- For stability, Δt must obey the Courant-Friedrichs-Lewy (CFL) condition which for this particular discretization is

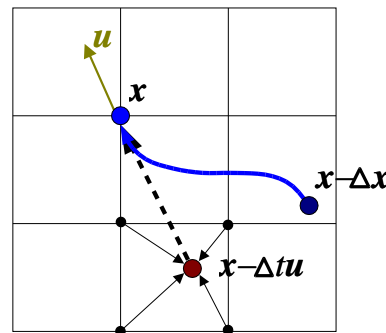
$$\Delta t < \frac{\Delta x}{\max |\mathbf{u}|}$$

- ★ Ensures that no information further than one cell away can arrive at the current cell in time interval Δt (since $\max |\mathbf{u}| < \frac{\Delta x}{\Delta t}$)

Solving Hyperbolic PDEs: Semi-Lagrangian Method

- Also known as the **method of characteristics**
- Suppose a particle at position x_0 at time t_0 moves to position $x_0 + \Delta x$ at time $t_0 + \Delta t$
- The particle's mass density ρ does not change because that's precisely what the PDE is dictating: $\frac{D\rho}{Dt} = 0$
- So to determine the new density at point x , we should look up the old density at point $x - \Delta x$ (the old position of the particle now at x):

$$\{\rho\}_x^{n+1} = \{\rho\}_{x-\Delta x}^n$$



Solving Hyperbolic PDEs: Semi-Lagrangian Method (cont'd)

- Since we don't know the exact displacement $\Delta \mathbf{x}$ of the particle, we use a first order approximation: $\Delta \mathbf{x} = \Delta t \mathbf{u}$.
- Evaluating the density at position $\mathbf{x} - \Delta t \mathbf{u}$ requires interpolation from neighbouring grid cells. Linear interpolation is acceptable in this case since we're using a first order approximation for the particle path.
 - ★ We could have used a higher order accurate approximation for the particle path $\Delta \mathbf{x}$ and used a correspondingly higher order interpolation.
- This method is popular in the atmospheric science community (for clouds, large radioactive plumes, *etc.*).

Semi-Lagrangian Method: Issues

- Unconditionally stable
- But lose accuracy
 - ★ Keep Δt at around $5 \times \text{CFL}$ limit to maintain good accuracy

Hyperbolic PDEs: Issues for Streaming Supercomputer

- Explicit method
 - ★ The upwind approximations will (in general) require support for conditionals in kernels
 - ★ Only looks to a cell's immediate neighbours, so can make use of locality of reference
- Semi-Lagrangian method
 - ★ Less locality of reference: For a given cell, may have to look up value at a cell arbitrarily far away (although we can limit this to e.g. 5-30 cells by bounding Δt)

Elliptic PDEs

- No time dependence
- Model steady state phenomena

Elliptic PDEs: Examples

- Laplace equation

$$\Delta p = f$$

$$\text{(Recall } \Delta p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = p_{xx} + p_{yy} + p_{zz}\text{)}$$

- Poisson equation

$$\nabla \cdot (\beta \nabla \rho) = f$$

which reduces to the Laplace equation if β is constant.

Solving Elliptic PDEs

- e.g. Solving 1-d Laplace equation:

$$p_{xx} = f$$

- We use a **central difference** approximation for the second order derivative:

$$\{p_{xx}\}_i \approx \frac{\{p\}_{i+1} - 2\{p\}_i + \{p\}_{i-1}}{\Delta x^2}$$

- Get a linear system of equations of the form

$$\{p\}_{i+1} - 2\{p\}_i + \{p\}_{i-1} = \Delta x^2 \{f\}_i$$

Solving Elliptic PDEs (cont'd)

- Boundary conditions:

- ★ Dirichlet: e.g. Specify $\{p\}_{-1} = \alpha$, get

$$\{p\}_1 - 2\{p\}_0 = \Delta x^2 \{f\}_0 - \alpha$$

- ★ Neumann: e.g. Specify $\{p_x\}_0 = 0$, get $\{p\}_0 = \{p\}_{-1}$, so $\{p_{xx}\}_0$ becomes

$$\{p_{xx}\}_0 = \frac{\{p\}_1 - \{p\}_0}{\Delta x^2}$$

Solving Elliptic PDEs (cont'd)

- We can write this system of linear equations in matrix form $Ax = b$
- A is a square matrix with the following properties:
 - ★ Size $N \times N$ where N is the number of grid cells
 - ★ Banded: e.g. In 3-d it has 7 bands
 - * Only need to store the bands - saves a lot of space!
 - ★ Symmetric
 - ★ Negative definite
 - * Means: All eigenvalues are negative; equivalently, $x^T Ax < 0$ for all $x \neq 0$

Solving Elliptic PDEs (cont'd)

- Various efficient techniques for solving Elliptic PDEs
 - ★ **FFT methods**
 - * Only good for periodic domains
 - ★ **Multigrid solvers**
 - * Optimal, but only become efficient as N gets fairly big
 - ★ Solve $Ax = b$ using **Preconditioned Conjugate Gradient**
 - * Described next!

Conjugate Gradient (CG) Method

- Requires A to be symmetric and positive definite
 - ★ So we will apply this method to $-A$
- Based on the method of steepest descent:
 - ★ The quadratic form $\frac{1}{2}x^T Ax - b^T x$ has a global minimum at the point x satisfying $Ax = b$
 - ★ Steepest descent is an iterative procedure which gets you to this minimum point by starting at some initial guess x_0 , and then at each iteration picking a direction and moving from x_n to the minimum point x_{n+1} along that direction
- Conjugate gradient is steepest descent with a particular choice of directions

Preconditioned Conjugate Gradient (PCG)

- Speed up CG by solving $M^{-1}Ax = M^{-1}b$ instead
 - ★ Use $M \approx A$
 - ★ Want M^{-1} easy to compute (or at least want to be able to compute $M^{-1}y$ efficiently)
- One particular method is known as **incomplete Cholesky preconditioning**
 - ★ Note: The Cholesky factorization of A is $A = LL^T$ (L lower triangular)
 - ★ Choose $M = \hat{L}\hat{L}^T$ where \hat{L} is the **incomplete Cholesky** factor of A
 - * \hat{L} is computed similarly to L , but is zero wherever A is zero
 - * So \hat{L} has the same sparsity pattern as A

Preconditioned Conjugate Gradient Algorithm

- x : iterated solution, r : residual $b - Ax$, d : direction, ϵ : tolerance
- Initialization: $i \leftarrow 0$, $r \leftarrow b - Ax$, $d \leftarrow M^{-1}r$, $\delta_{new} \leftarrow r^T d$, $\delta_0 \leftarrow \delta_{new}$

while $i < i_{max}$ and $\delta_{new} > \epsilon^2 \delta_0$ do

$\alpha \leftarrow \frac{\delta_{new}}{d^T A d}$ (Set α so that $x + \alpha d$ is the minimum along direction d)

$x \leftarrow x + \alpha d$

$r \leftarrow b - Ax$ (Update the residual - can compute $r \leftarrow r - \alpha A d$ instead)

$\delta_{old} \leftarrow \delta_{new}$

$\delta_{new} \leftarrow r^T M^{-1} r$

$\beta \leftarrow \frac{\delta_{new}}{\delta_{old}}$

$d \leftarrow M^{-1} r + \beta d$ (Choose the next A -orthogonal direction)

$i \leftarrow i + 1$

- For a 200×200 grid smoke simulation with $x_0 = 0$ and $\epsilon = 10^{-5}$, PCG takes around 250 iterations for the pressure calculation

Elliptic PDEs: Issues for Streaming Supercomputer

- Need to be able to solve $Ax = b$!

Solving Inviscid, Incompressible Fluid Flow

- Using our hyperbolic and elliptic PDE solvers is sufficient to solve the special case of inviscid ($\mu = 0$) incompressible fluid flow
- We'll assume the fluid is homogeneous ($\rho = 1$) for simplicity
- We need to solve the following two equations simultaneously

$$\begin{aligned}\nabla \cdot \mathbf{u} &= 0 \\ \mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p &= \mathbf{f}\end{aligned}$$

Splitting

- We use a forward Euler approximation for \mathbf{u}_t in the Navier Stokes equations
- We split the Navier Stokes equations into three equations

$$\begin{aligned}\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + (\mathbf{u} \cdot \nabla) \mathbf{u} &= 0 \\ \frac{\mathbf{u}^{**} - \mathbf{u}^*}{\Delta t} &= \mathbf{f} \\ \frac{\mathbf{u}^{n+1} - \mathbf{u}^{**}}{\Delta t} + \nabla p &= 0\end{aligned}$$

and proceed by solving each in turn

- \mathbf{u}^* and \mathbf{u}^{**} are intermediate velocity fields
- This splitting (*a.k.a* projection) method is due to Chorin (1965)

Solving in Steps

- **Step 1: Solve**

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = 0$$

- ★ A hyperbolic PDE
- ★ Use explicit or semi-Lagrangian method

- **Step 2: Solve**

$$\frac{\mathbf{u}^{**} - \mathbf{u}^*}{\Delta t} = \mathbf{f}$$

- ★ Here we account for gravity, buoyancy, and also add artificial forces such as vorticity confinement which enhances our simulation results

Solving in Steps (cont'd)

- **Step 3: Solve**

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^{**}}{\Delta t} + \nabla p = 0$$

- ★ We want the final velocity field \mathbf{u}^{n+1} to be divergence free (*i.e.* to satisfy $\nabla \cdot \mathbf{u}^{n+1} = 0$)
- ★ Need to find pressure field p to make this hold

Solving in Steps (cont'd)

- **Step 3 (cont'd):**

- ★ **Projection method:**

- * Apply divergence operator to both sides:

$$\frac{\nabla \cdot \mathbf{u}^{n+1} - \nabla \cdot \mathbf{u}^{**}}{\Delta t} + \nabla \cdot \nabla p = 0$$

- * We assume \mathbf{u}^{n+1} is divergence free, so the $\nabla \cdot \mathbf{u}^{n+1}$ term disappears, and we get the elliptic PDE

$$\Delta p = \frac{\nabla \cdot \mathbf{u}^{**}}{\Delta t}$$

- * Solve $Ax = b$ using PCG
- ★ Now that we have p (and \mathbf{u}^{**}) we can get \mathbf{u}^{n+1}

Visualization

- The above three steps allow us to simulate the evolution of the velocity field in time
- To visualize the fluid flow, we can render the density (or any other passive scalar) in each cell, and move it along the velocity field using the hyperbolic PDE

$$\rho_t + \mathbf{u} \cdot \nabla \rho = 0$$

Sample Animations

- 2-d inviscid fluid flow written in C++

Parabolic PDEs

- Time dependence
- Information flows at infinite speed in every direction (infinite domain of dependence)

Parabolic PDEs: Examples

- Heat/Diffusion equation

$$u_t = \nabla \cdot (\mu \nabla u)$$

where u is the heat and μ the thermal coefficient

- The equation

$$\mathbf{u}_t = \mu \Delta \mathbf{u}$$

which will come up when we solve for *viscous* fluid flow. \mathbf{u} is the velocity, μ the viscosity coefficient. (This is similar to the heat equation with constant μ)

Solving Parabolic PDEs: Explicit Method

- e.g. Solving

$$u_t = \mu \Delta u$$

- Use forward Euler for u_t , and use u^n on the right hand side:

$$\frac{u^{n+1} - u^n}{\Delta t} = \mu \Delta u^n$$

- This method is “explicit” because u^{n+1} does not appear on the right hand side
- We’ve already discussed the finite difference form of Δu^n
- Requires small Δt : $\Delta t = O(\Delta x^2)$

Solving Parabolic PDEs: Implicit Method

- Use u^{n+1} on right hand side:

$$\frac{u^{n+1} - u^n}{\Delta t} = \mu \Delta u^{n+1}$$

- Rearranging gives

$$u^{n+1} - \Delta t \mu \Delta u^{n+1} = u^n$$

which can be written in matrix form as

$$(I - \Delta t \mu \Delta) u^{n+1} = u^n$$

with the matrix for the Laplacian operator (Δ) constructed similar to the solution of the elliptic Laplace equation

- Comes down to solving $Ax = b$ using preconditioned conjugate gradient as for elliptic PDEs

Parabolic PDEs: Issues for Streaming Supercomputer

- For the implicit method, we again need to be able to solve $Ax = b$

Adding Viscosity to our Fluid Flow Solver

- We need to add one step:

- ★ Step 1: Solve $\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = 0$

- ★ **Step 2:** Compute $\mathbf{u}^{*1/2}$ by solving the parabolic equation

$$\frac{\mathbf{u}^{*1/2} - \mathbf{u}^*}{\Delta t} = \mu \Delta \mathbf{u}^{*1/2}$$

using PCG

- ★ Step 3: Solve $\frac{\mathbf{u}^{**} - \mathbf{u}^{*1/2}}{\Delta t} = \mathbf{f}$

- ★ Step 4: Solve $\frac{\mathbf{u}^{n+1} - \mathbf{u}^{**}}{\Delta t} + \nabla p = 0$

Sample Animations 2

- Smoke simulation with added viscosity

Summary

- **Hyperbolic PDEs** (e.g. used for density and velocity advection)
 - ★ **Explicit method**: Compute next value using forward Euler approximation and upwind approximations (D^+ or D^- depending on velocity). Requires looking up neighbouring values
 - ★ **Semi-Lagrangian method**: Compute next value at \mathbf{x} by looking up old value at $\mathbf{x} - \Delta t \mathbf{u}$ (interpolate value)
- **Elliptic PDEs** (e.g. compute pressure to make velocity divergence free)
 - ★ **Solving**: Need to solve $Ax = b$ using PCG
- **Parabolic PDEs** (e.g. used for viscosity in fluid flow)
 - ★ **Explicit method**: Compute next value using forward Euler approximation and finite difference form of $\Delta \mathbf{u}$ (requires looking up neighbouring values)
 - ★ **Implicit method**: Need to solve $Ax = b$ using PCG

Discussion

- Semi-Lagrangian method
 - ★ Look up grid values arbitrarily far away = no locality?
- Preconditioned conjugate gradient
 - ★ Using a preconditioner speeds up CG, but also has overhead
 - ★ Investigate different preconditioners and CG with no preconditioning
- Where do we go next with smoke?
 - ★ Port 2-d code to Brook?