

# **Physically-Based Animation and PDEs**

---

**Computer Graphics  
CMU 15-462/15-662**

# Last time: Optimization

- **Graphics as optimization**
- **Many complex criteria/constraints**
- **Technique: numerical optimization**
  - **pick initial guess**
  - **ski downhill**
  - **keep fingers crossed!**
- **Today: return to differential equations**
  - **saw ODEs—derivatives in time**
  - **now PDEs—also have derivatives in space**
  - **describe many natural phenomena (water, smoke, cloth, ...)**
  - **recent revolution in CG/visual effects**



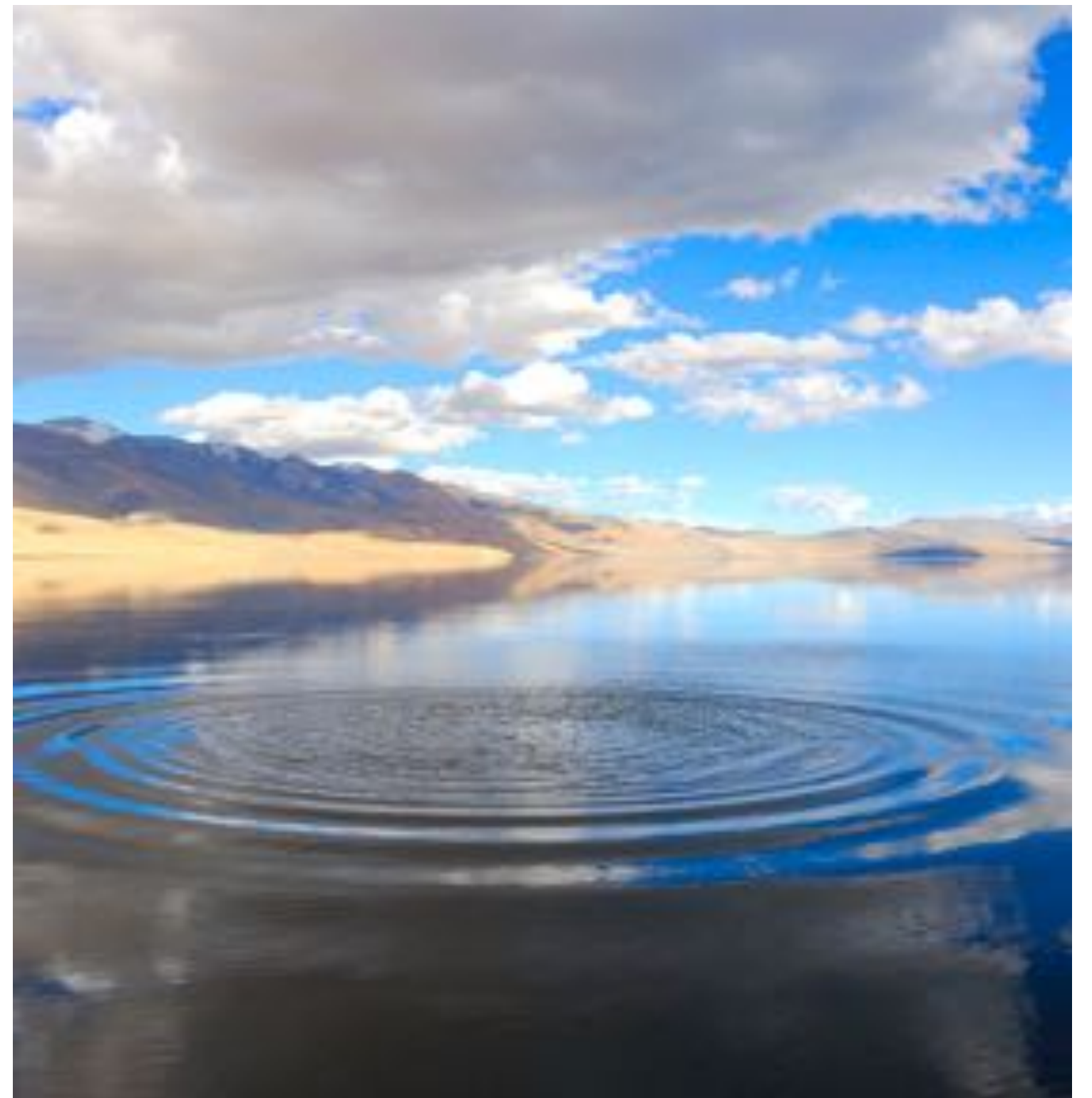
# Partial Differential Equations (PDEs)

- **ODE: Implicitly describe function in terms of its time derivatives**
- **Like any implicit description, have to solve for actual function**
- **PDE: Also include space derivatives in description**

**ODE—rock flies through air**



**PDE—rock lands in pond**



# To make a long story short...

- Solving ODE looks like *“add a little velocity each time”*

$$q_{k+1} = q_k + \tau f(q)$$

- Solving a PDE looks like *“take weighted combination of neighbors to get velocity (...and add a little velocity each time)”*

	1	
1	-4	1
	1	

$f(q)$

$$q_{k+1} = q_k + \tau f(q)$$

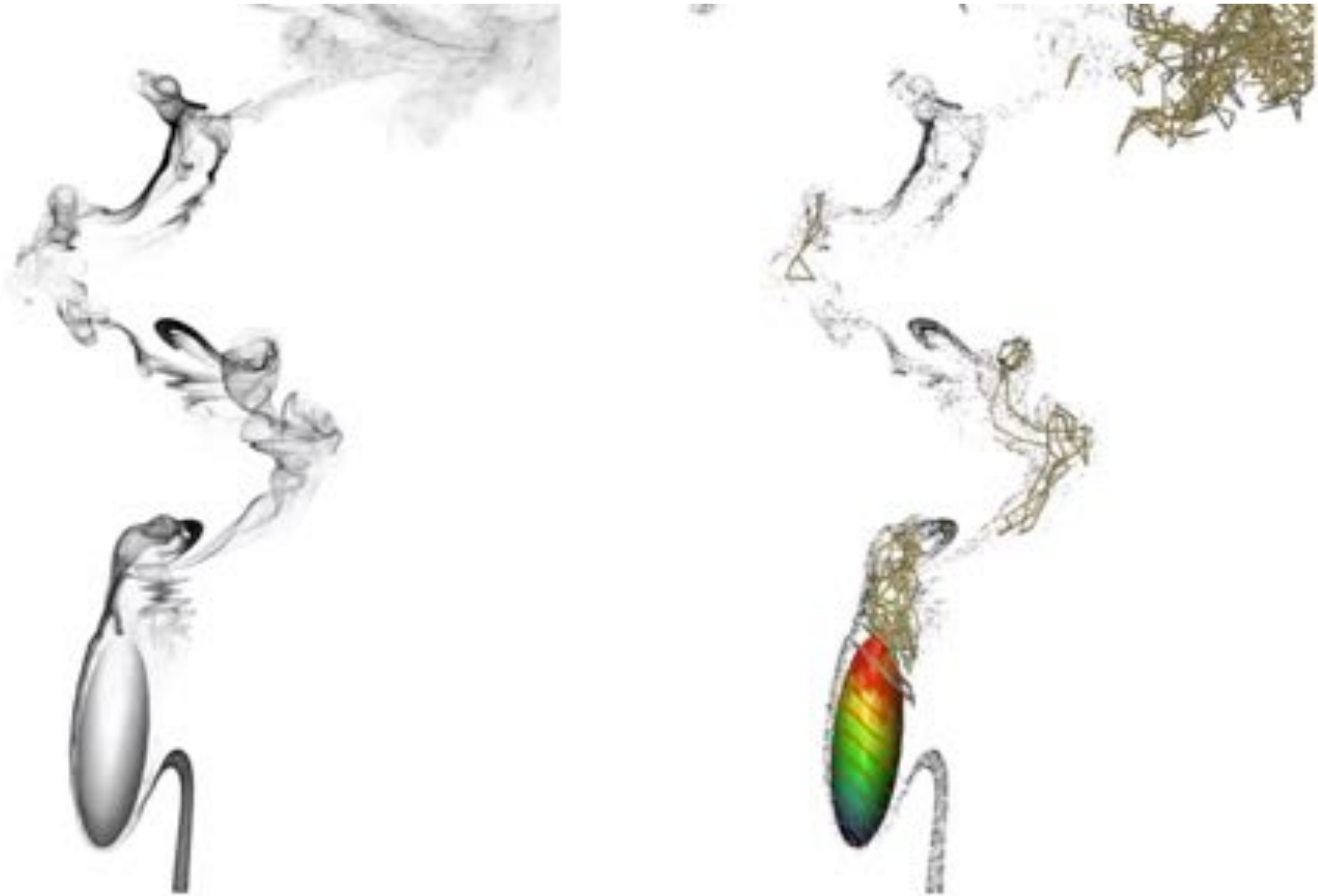
**...obviously there is a *lot* more to say here!**

# Liquid Simulation in Graphics



Losasso, F., Shinar, T. Selle, A. and Fedkiw, R., *"Multiple Interacting Liquids"*

# Smoke Simulation in Graphics



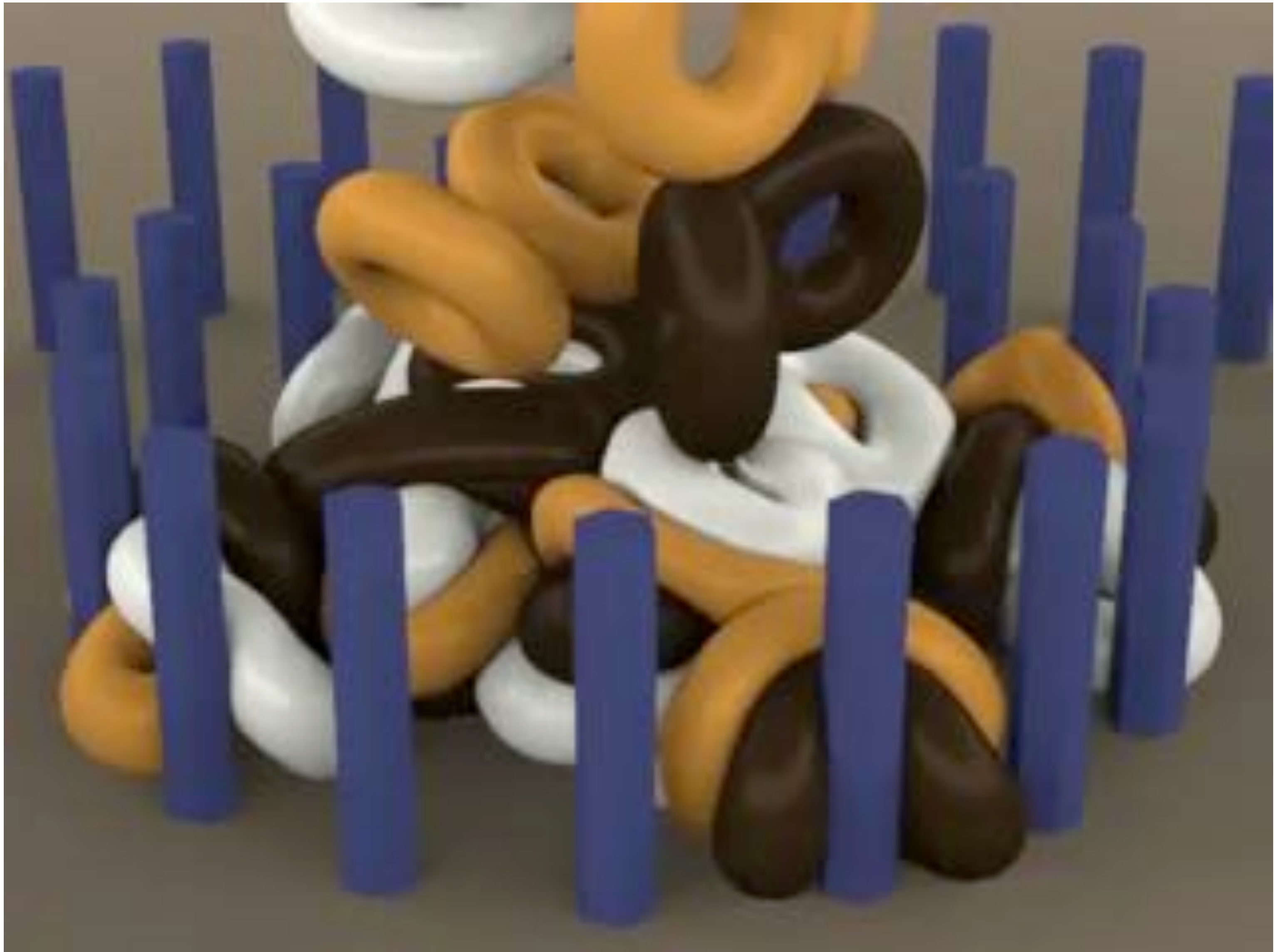
S. Weißmann, U. Pinkall. *"Filament-based smoke with vortex shedding and variational reconnection"*

# Cloth Simulation in Graphics



Zhili Chen, Renguo Feng and Huamin Wang, *"Modeling friction and air effects between cloth and deformable bodies"*

# Elasticity in Graphics



Irving, G., Schroeder, C. and Fedkiw, R., "Volume Conserving Finite Element Simulation of Deformable Models"

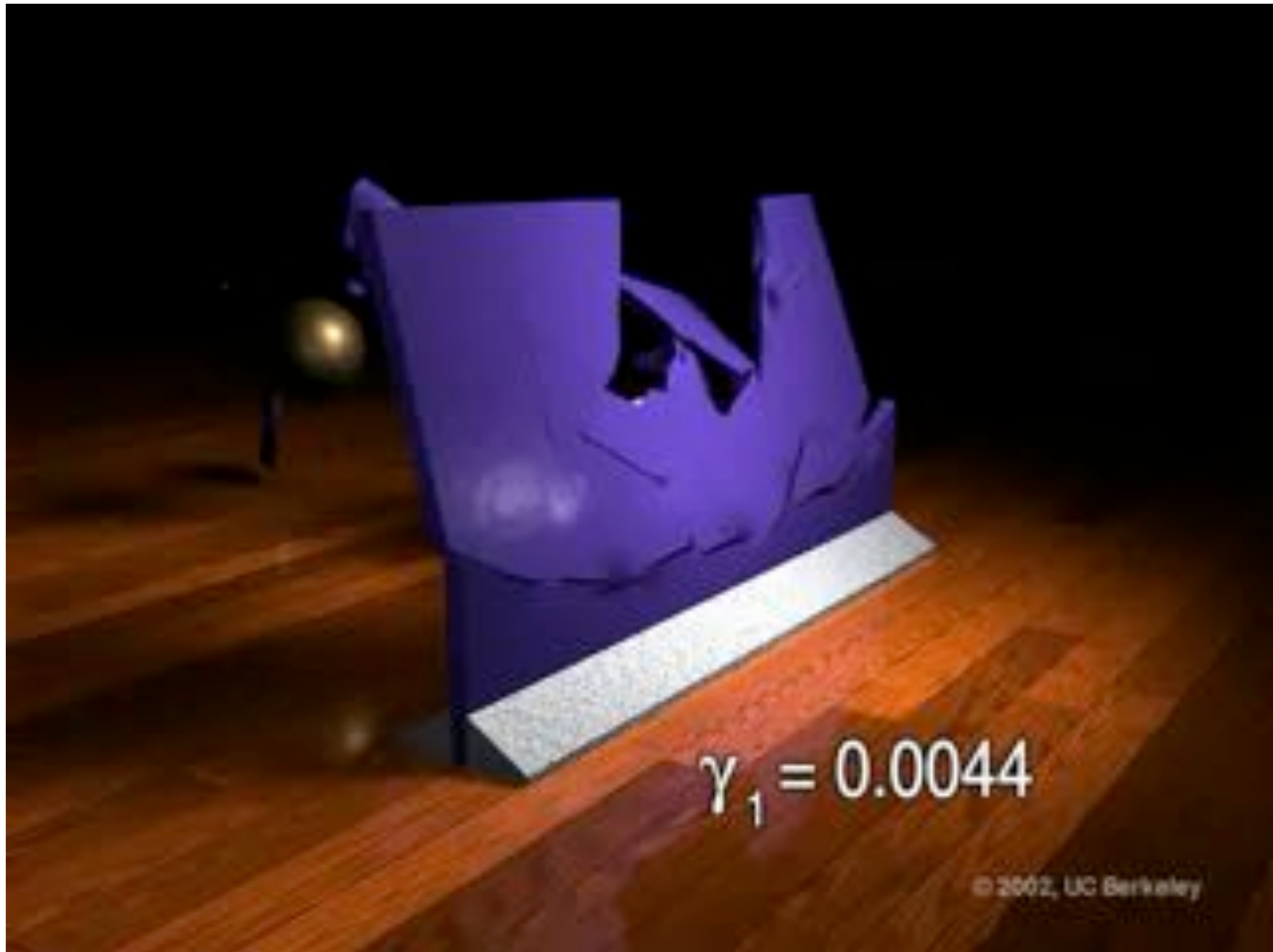


# Hair Simulation in Graphics



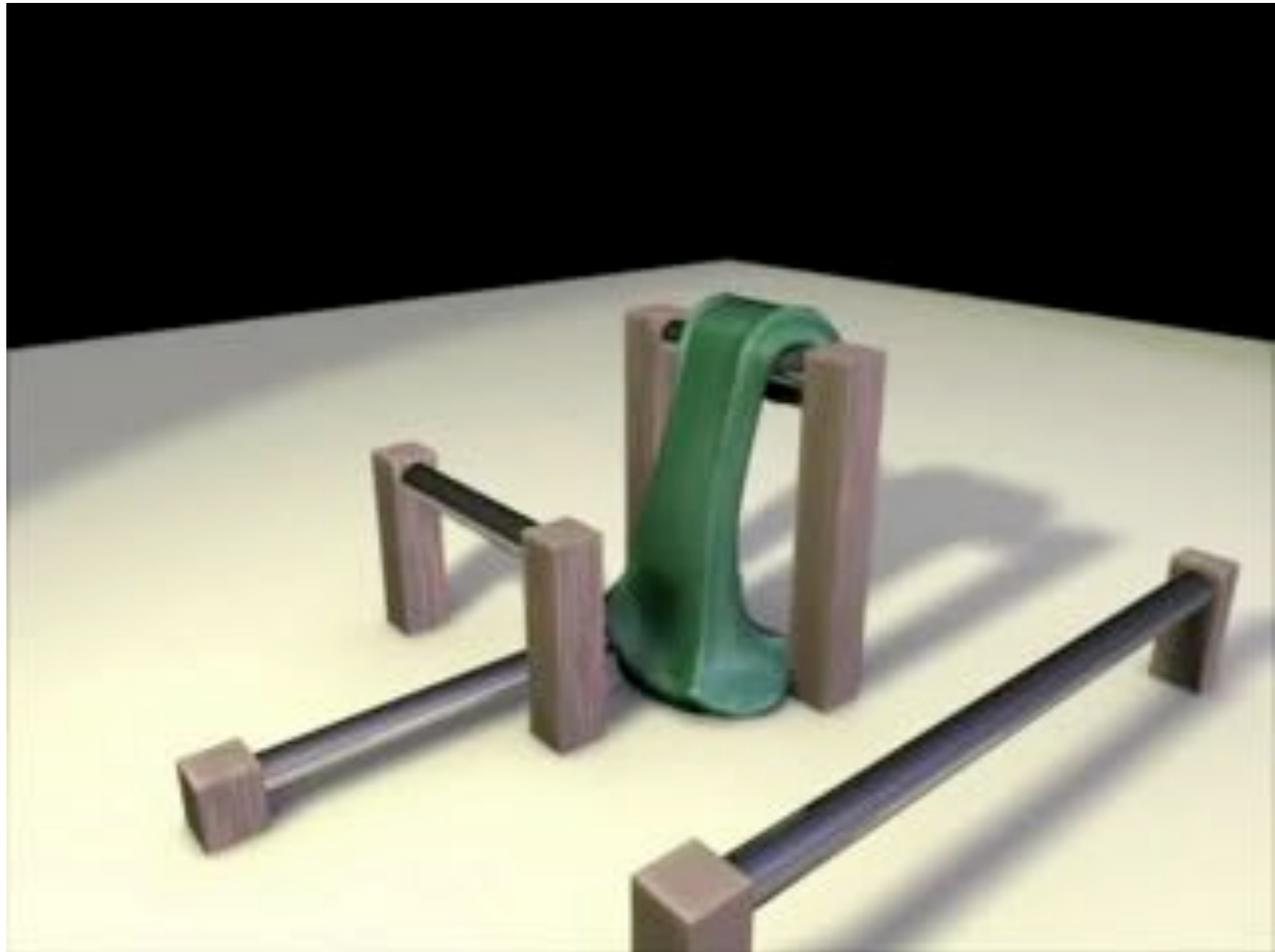
Danny M. Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, Eitan Grinspun,  
*"Adaptive Nonlinearity for Collisions in Complex Rod Assemblies"*

# Fracture in Graphics



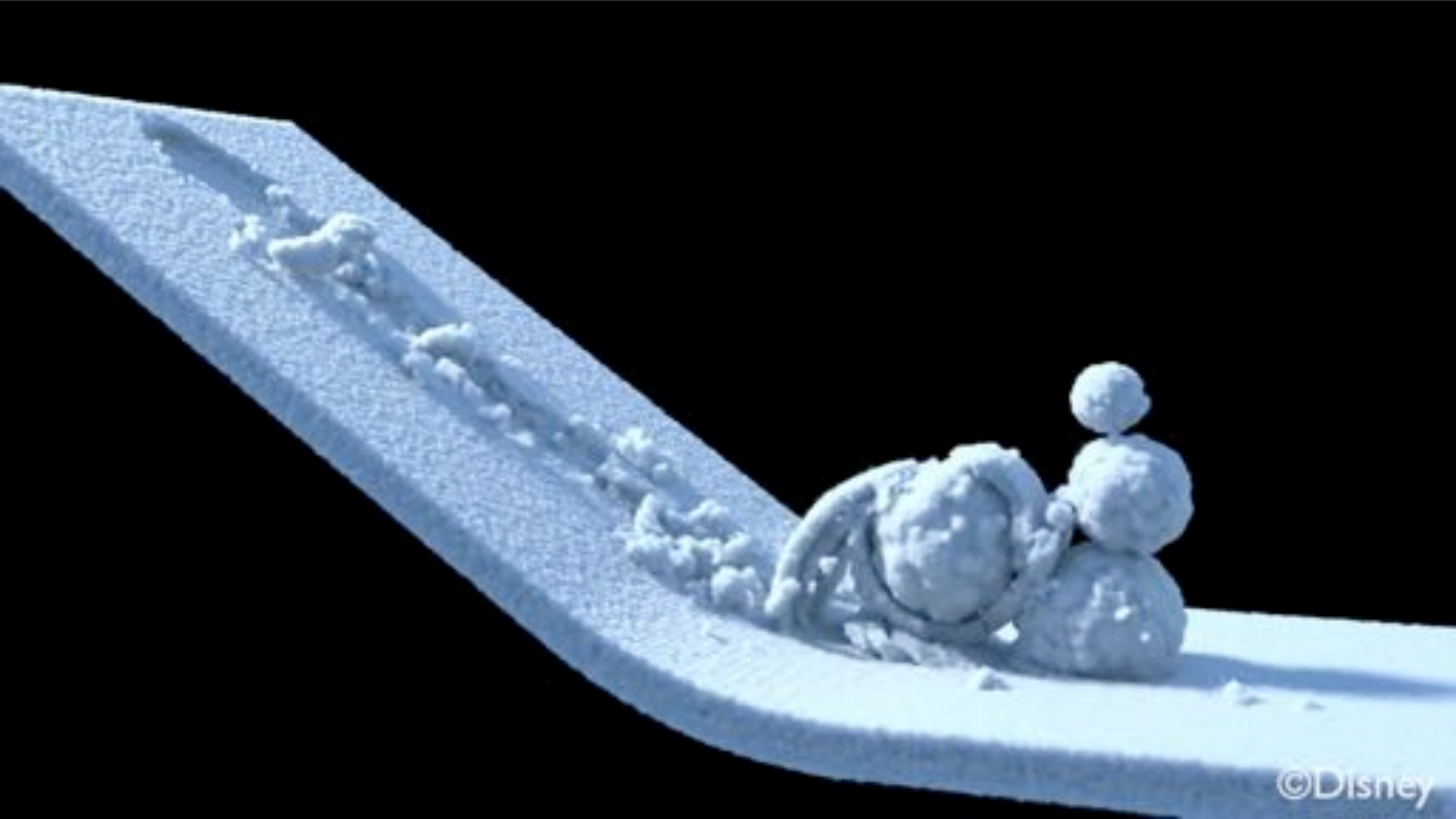
James F. O'Brien, Adam Bargteil, Jessica Hodgins, *"Graphical Modeling and Animation of Ductile Fracture"*

# Viscoelasticity in Graphics



Chris Wojtan, Greg Turk, *"Fast Viscoelastic Behavior with Thin Features"*

# Snow Simulation in Graphics



Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, Andrew Selle, *"A Material Point Method For Snow Simulation"*

# Definition of a PDE

- Want to solve for a function of time *and* space

$$u(t, x)$$

time      space

- Function given implicitly in terms of derivatives:

$$\dot{u}, \ddot{u}, \frac{d}{dt^3} u, \frac{d}{dt^4} u, \dots$$

any combination of time derivatives

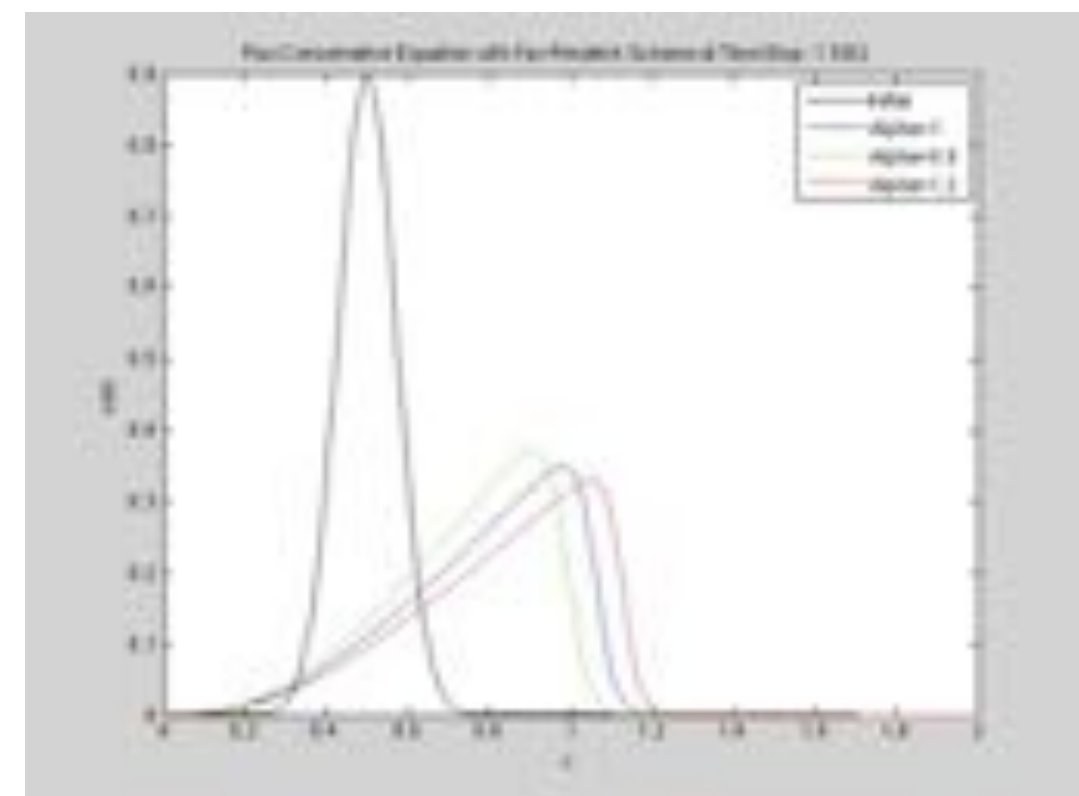
$$\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^m + n u}{\partial x_i^m \partial x_j^n}, \dots$$

plus any combination of space derivatives

- Example:

$$\dot{u} + uu' = au''$$

(Burgers' equation)



# Anatomy of a PDE

- Linear vs. nonlinear: how are derivatives combined?

**nonlinear!**

$$\dot{u} + uu' = au'' \quad \text{(Burgers' equation)}$$

$$\dot{u} = au'' \quad \text{(diffusion equation)}$$

- Order: how many derivatives in space & time?

1st order in time

$$\dot{u} + uu' = au'' \quad \text{(Burgers' equation)}$$

2nd order in space

2nd order in time

$$\ddot{u} = au'' \quad \text{(wave equation)}$$

2nd order in space

- Nonlinear / higher order  $\Rightarrow$  HARDER TO SOLVE!

# Model Equations

- Fundamental behavior of many important PDEs is well-captured by three model linear equations:

“Laplacian” (more later!)

LAPLACE EQUATION (“ELLIPTIC”)  $\Delta u = 0$

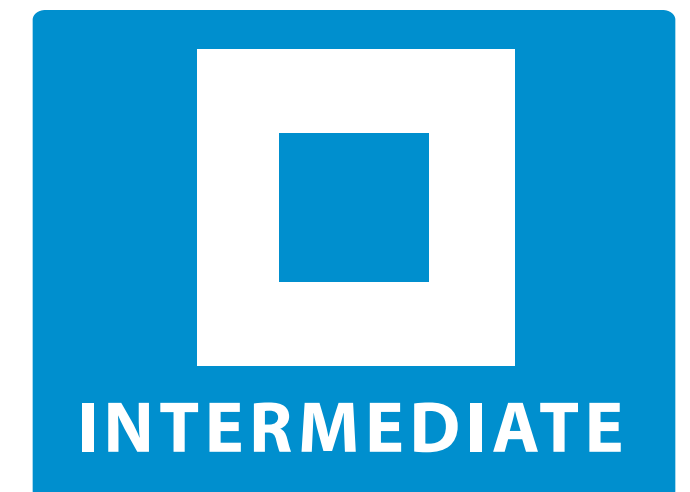
“what’s the smoothest function interpolating the given boundary data”

Solve numerically?



HEAT EQUATION (“PARABOLIC”)  $\dot{u} = \Delta u$

“how does an initial distribution of heat spread out over time?”



WAVE EQUATION (“HYPERBOLIC”)  $\ddot{u} = \Delta u$

“if you throw a rock into a pond, how does the wavefront evolve over time?”

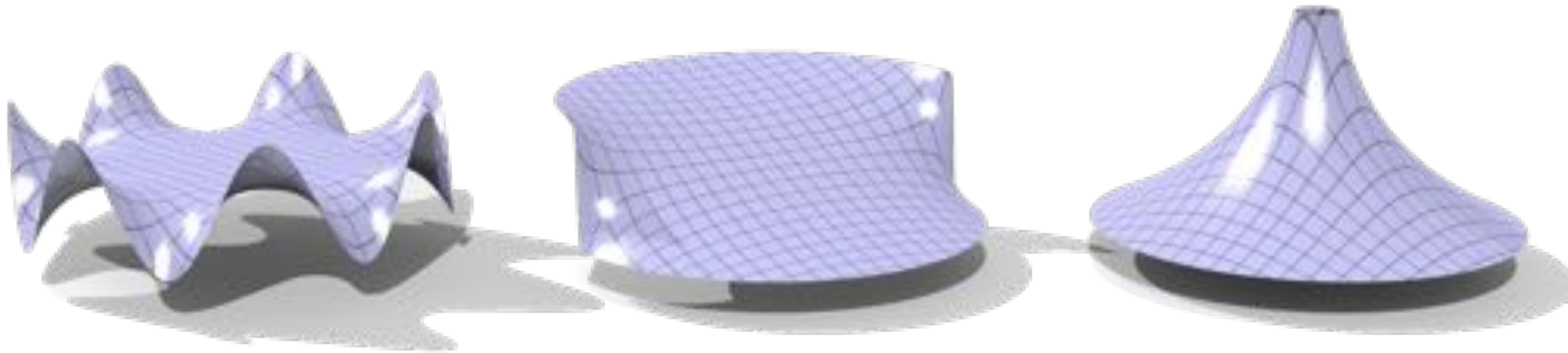


[ NONLINEAR + HYPERBOLIC + HIGH-ORDER ]



# Elliptic PDEs / Laplace Equation

- **“What’s the smoothest function interpolating the given boundary data?”**

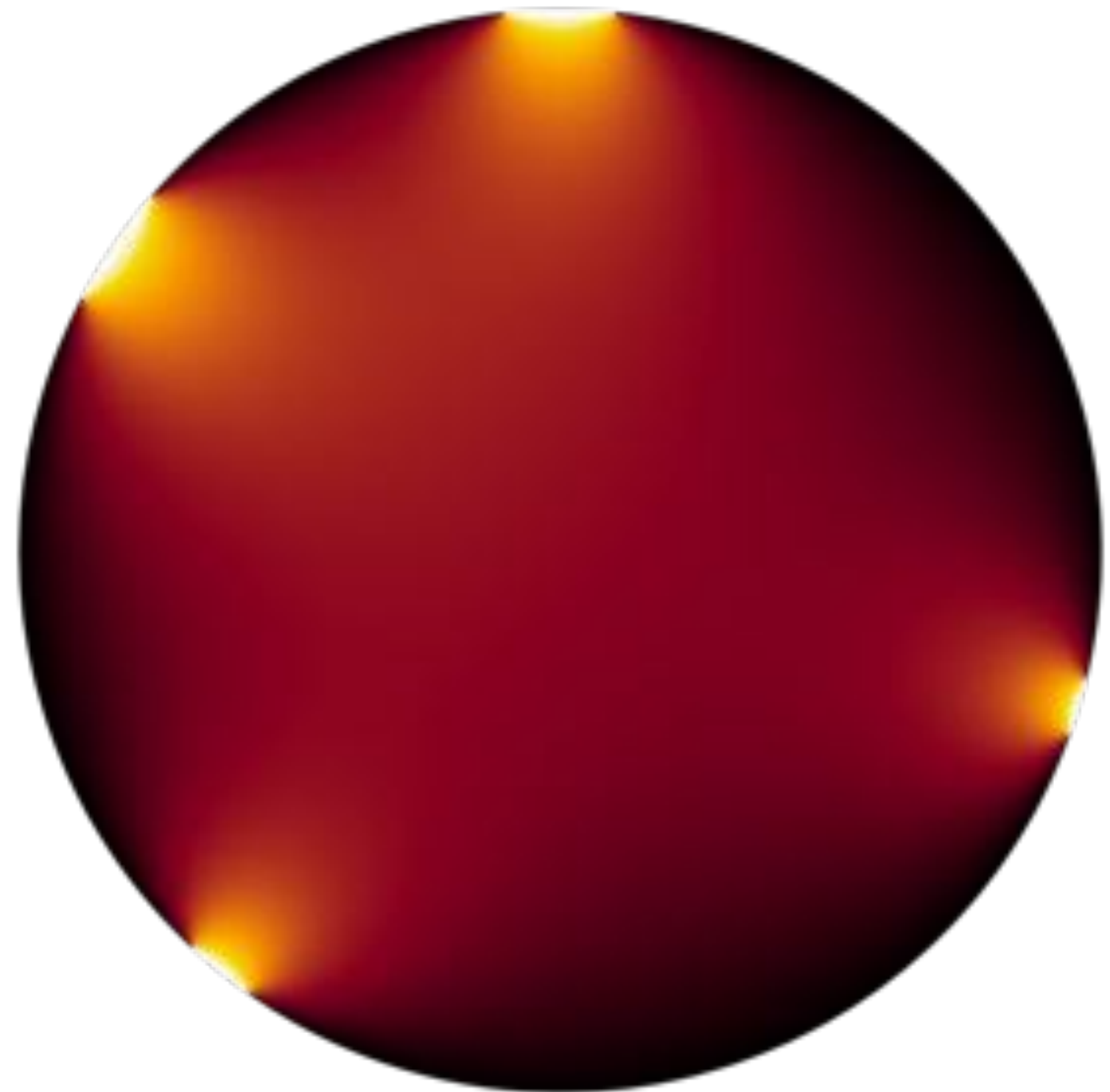
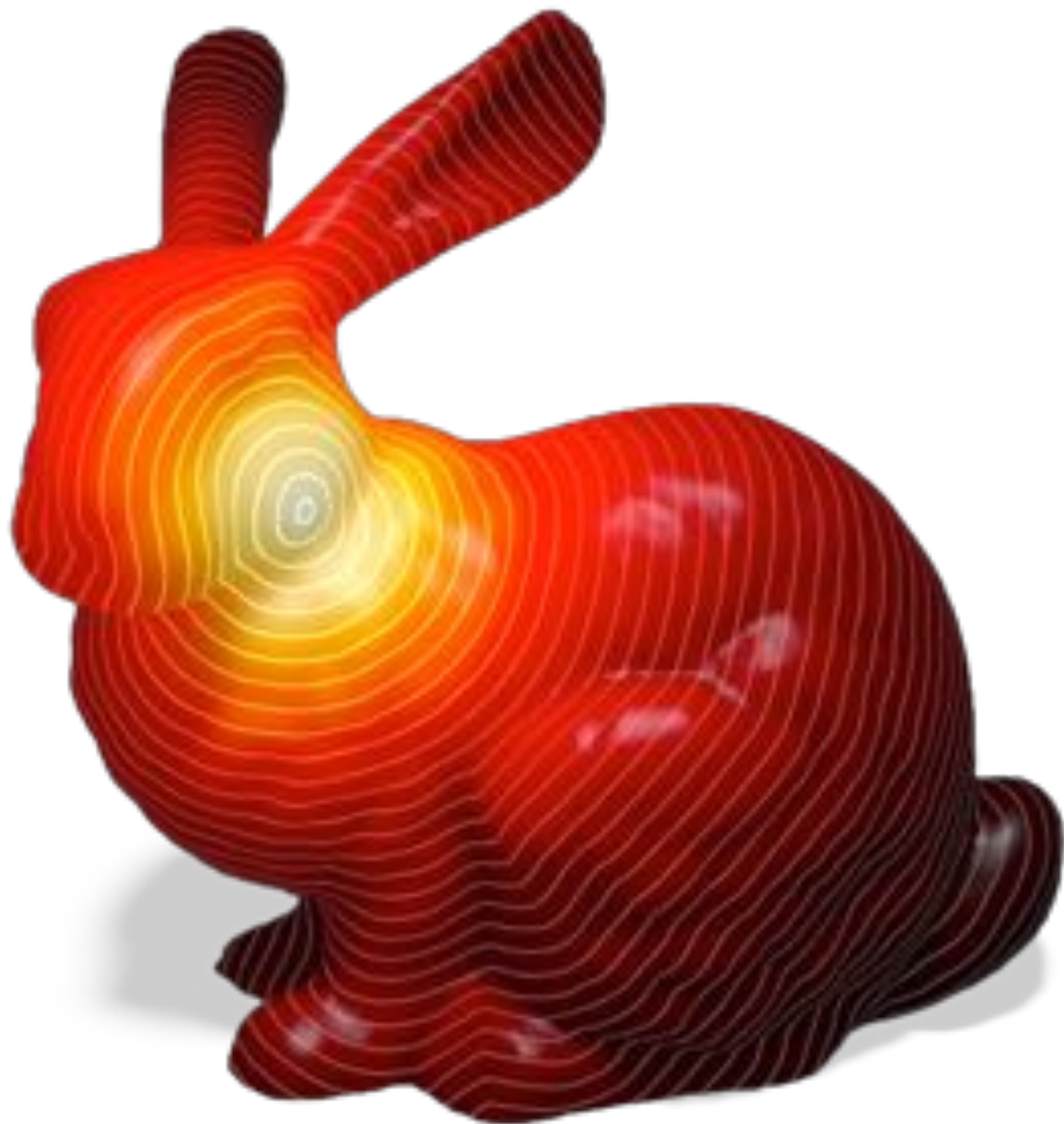


- **Conceptually: each value is at the average of its “neighbors”**
- **Roughly speaking, why is it easier to solve?**
- **Very robust to errors: just keep averaging with neighbors!**



# Parabolic PDEs / Heat Equation

- “How does an initial distribution of heat spread out over



- After a long time, solution is same as Laplace equation!
- Models damping / viscosity in many physical systems

# Hyperbolic PDEs / Wave Equation

- **“If you throw a rock into a pond, how does the wavefront evolve over time?”**



- **Errors made at the beginning will persist for a long time! (hard)**

**How did we do that?**

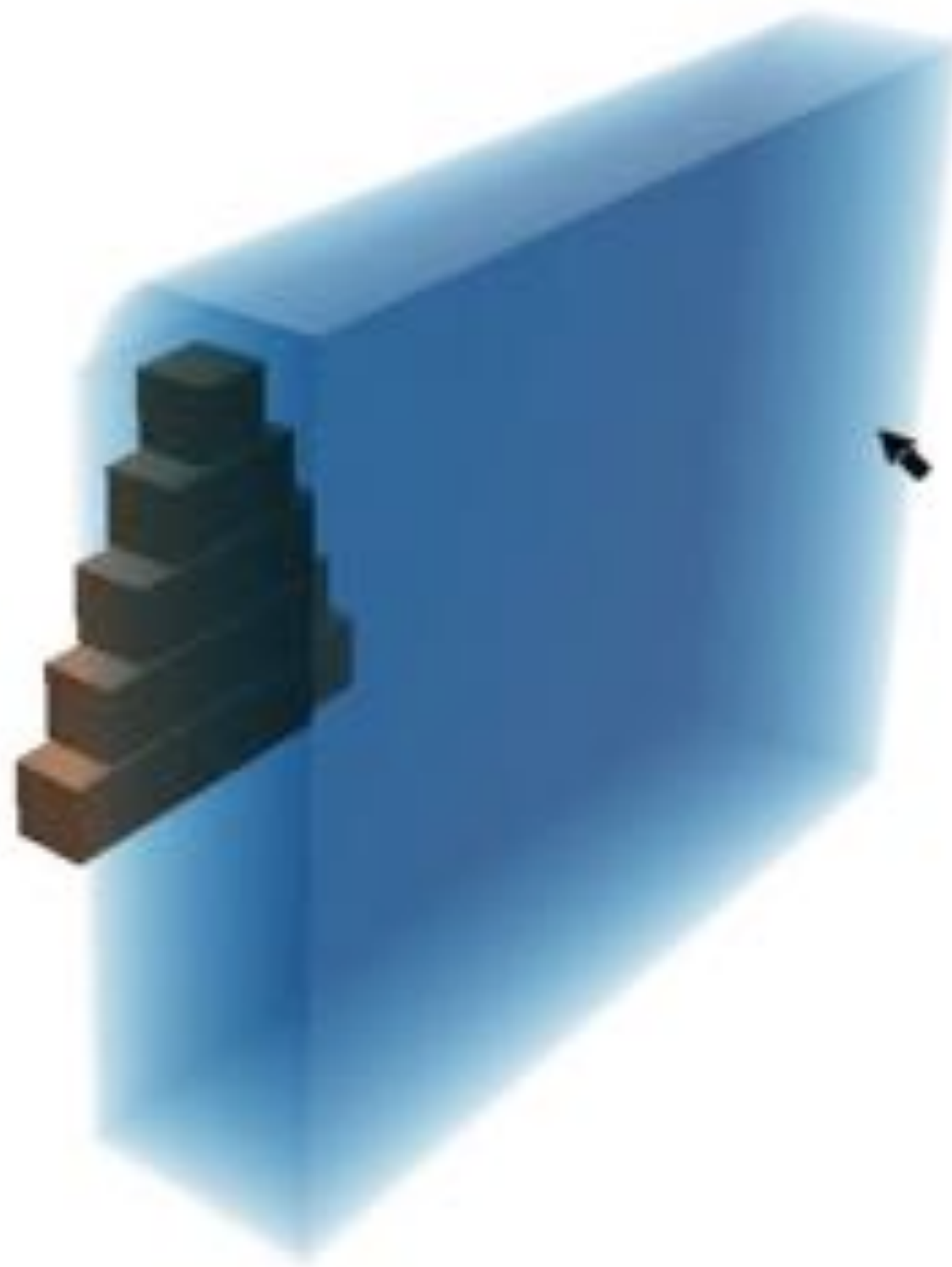
# Numerical Solution of PDEs—Overview

- Like ODEs, many interesting PDEs are difficult/impossible to solve analytically—especially if we want to incorporate data (e.g., user interaction)
- Must instead use numerical integration
- Basic strategy:
  - pick a time discretization (forward Euler, backward Euler...)
  - pick a spatial discretization (TODAY)
  - as with ODEs, run a time-stepping algorithm
- Historically, very expensive—only for “hero shots” in movies
- Computers are ever faster...

# Real Time PDE-Based Simulation (Fire)



# Real Time PDE-Based Simulation (Water)

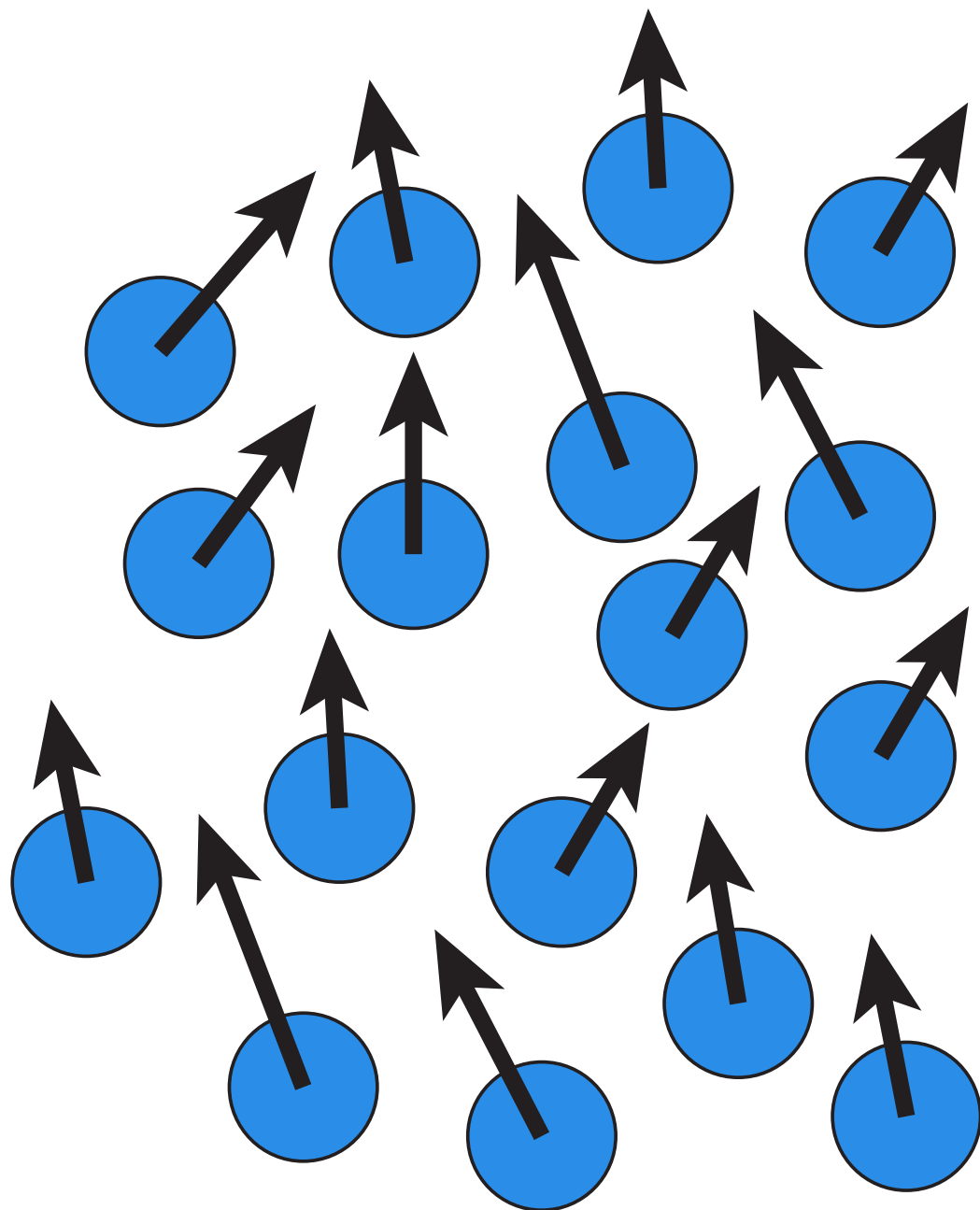


Nuttapong Chentanez, Matthias Müller, *"Real-time Eulerian water simulation using a restricted tall cell grid"*

# Lagrangian vs. Eulerian

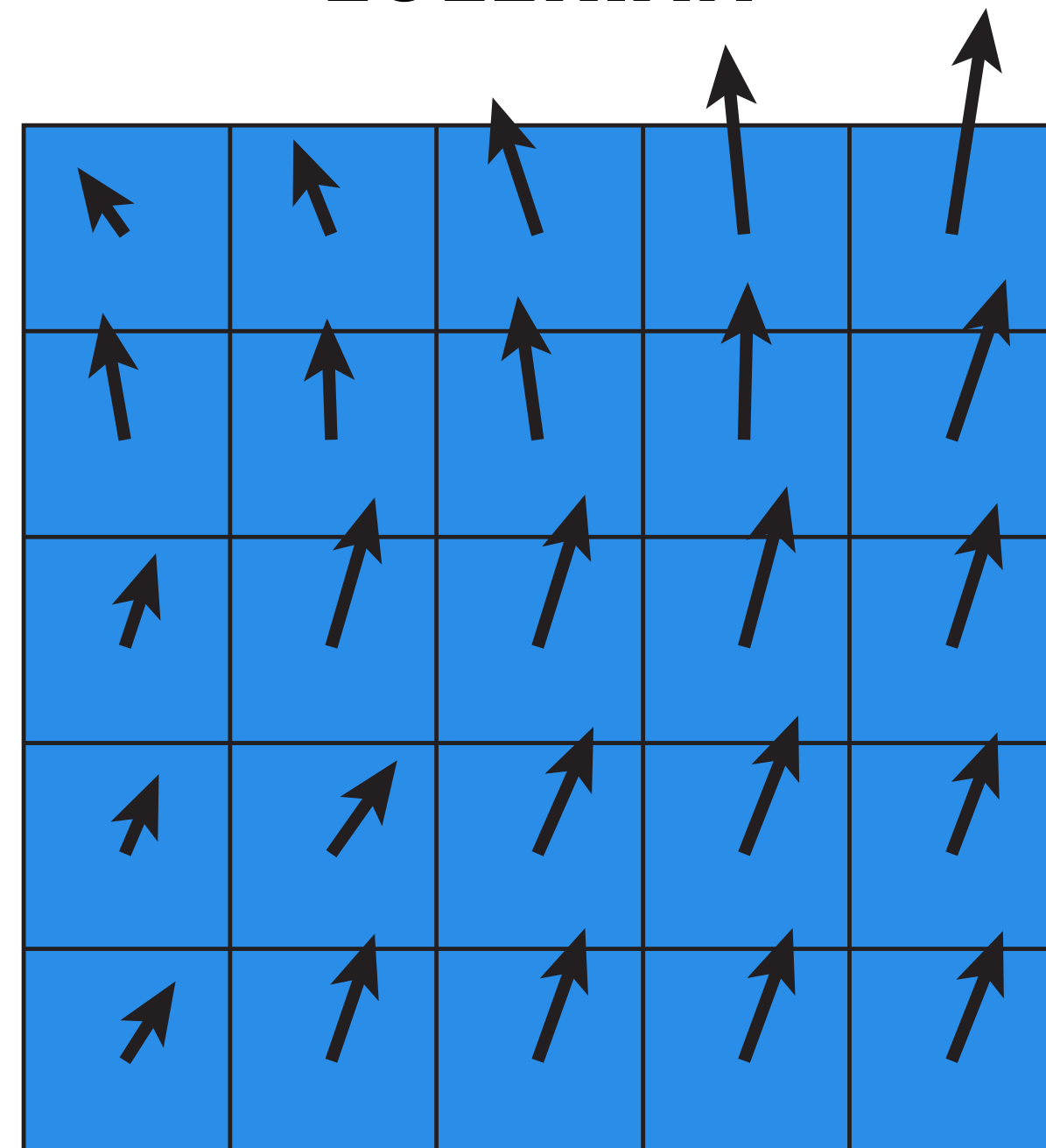
- Two basic ways to discretize space: Lagrangian & Eulerian
- E.g., suppose we want to encode the motion of a fluid

## LAGRANGIAN



**track position & velocity  
of moving particles**

## EULERIAN

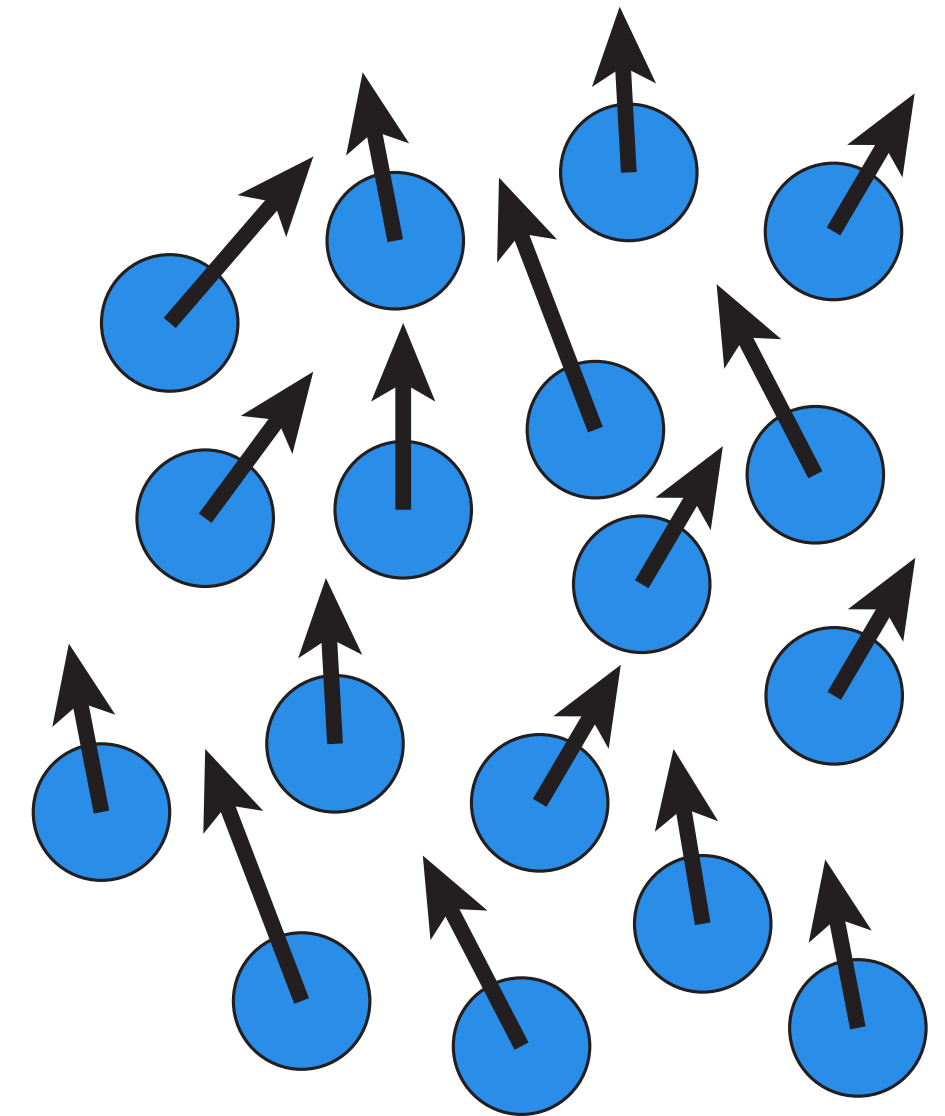


**track velocity (or *flux*)  
at fixed grid locations**

# Lagrangian vs. Eulerian—Trade-Offs

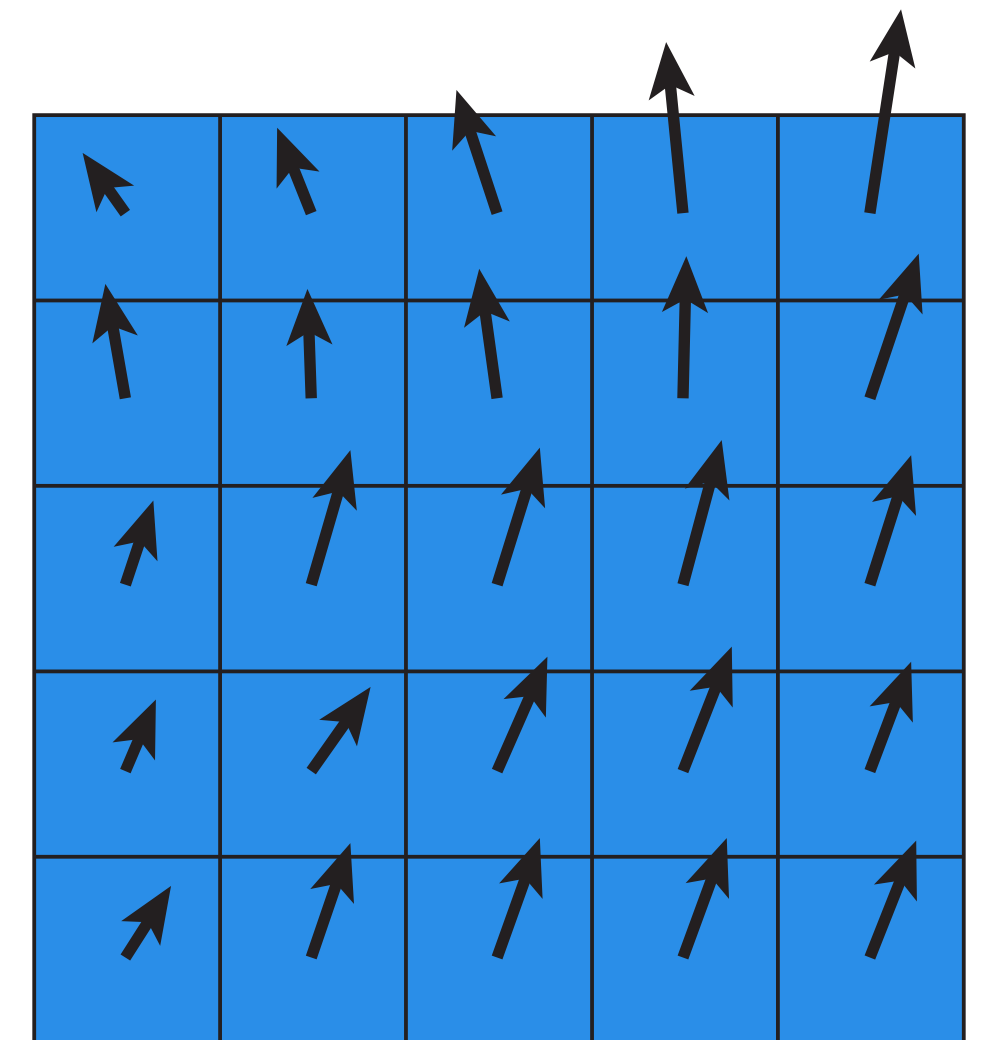
## ■ Lagrangian

- **conceptually easy (like polygon soup!)**
- **resolution/domain not limited by grid**
- **good particle distribution can be tough**
- **finding neighbors can be expensive**



## ■ Eulerian

- **fast, regular computation**
- **easy to represent, e.g., smooth surfaces**
- **simulation “trapped” in grid**
- **grid causes “numerical diffusion” (blur)**
- **need to understand PDEs (but you will!)**





# Mixing Lagrangian & Eulerian

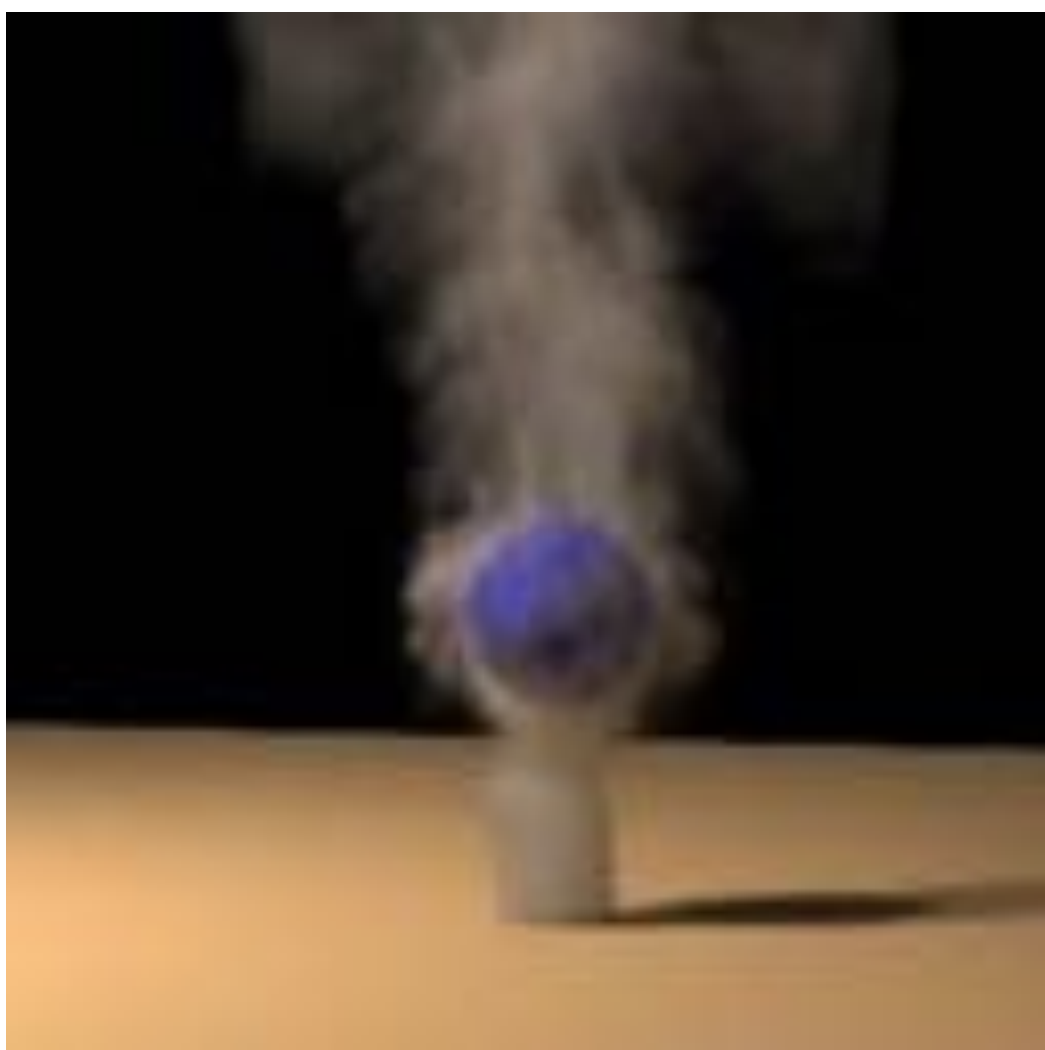
- Of course, no reason you have to choose just one!
- Many modern methods mix Lagrangian & Eulerian:
  - PIC/FLIP, particle level sets, mesh-based surface tracking, Voronoi-based, arbitrary Lagrangian-Eulerian (ALE), ...

*Maya Bifrost*



# Aside: Which Quantity Do We Solve For?

- Many PDEs have mathematically equivalent formulations in terms of different quantities
- E.g., incompressible fluids:
  - *velocity*—how fast is each particle moving?
  - *vorticity*—how fast is fluid “spinning” at each point?
- Computationally, can make a big difference
- *Pick the right tool for the job!*



**Ok, but we're getting *way* ahead of ourselves.  
How do we solve *easy* PDEs?**

# Numerical PDEs—Basic Strategy

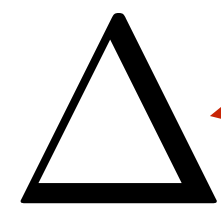
- **Pick PDE formulation**
  - Which quantity do we want to solve for?
  - E.g., velocity or vorticity?
- **Pick spatial discretization**
  - How do we approximate derivatives in space?
- **Pick time discretization**
  - How do we approximate derivatives in time?
  - When do we evaluate forces?
  - Forward Euler, backward Euler, symplectic Euler, ...
- **Finally, we have an *update rule***
- **Repeatedly solve to generate an animation**



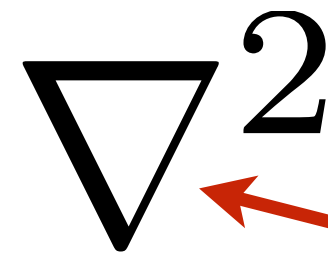
**Richard Courant**

# The Laplace Operator

- All of our model equations used the Laplace operator
- Different conventions for symbol:



← same symbol used for “change”



← same symbol used for Hessian!

- *Unbelievably* important object showing up everywhere across physics, geometry, signal processing, ...
- Ok, but what does it mean?
- *Differential operator*: eats a function, spits out its “2nd derivative”

- What does that mean for a function  $u: \mathbb{R}^n \rightarrow \mathbb{R}$ ?

- divergence of gradient

$$\Delta u = \overset{\text{div}}{\nabla} \cdot \overset{\text{grad}}{\nabla} u$$

- sum of second derivatives

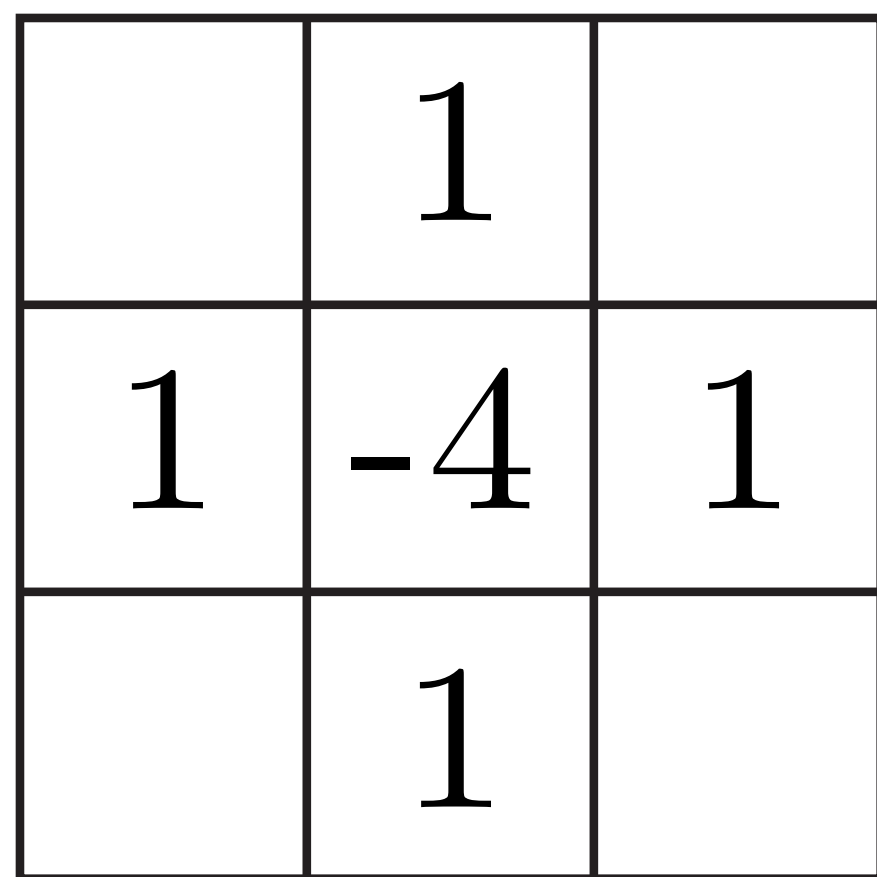
$$\Delta u = \frac{\partial^2 u}{\partial x_1^2} + \dots + \frac{\partial^2 u}{\partial x_n^2}$$

- “average” curvature

# Discretizing the Laplacian

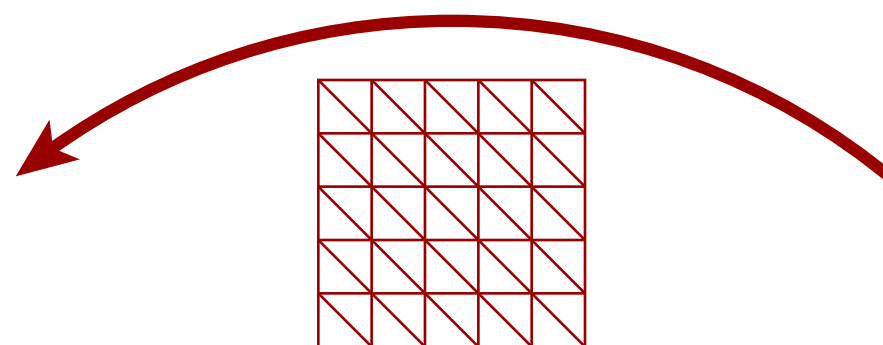
- How do we approximate the Laplacian?
- Depends on discretization (Eulerian, Lagrangian, grid, mesh, ...)

GRID  $h$

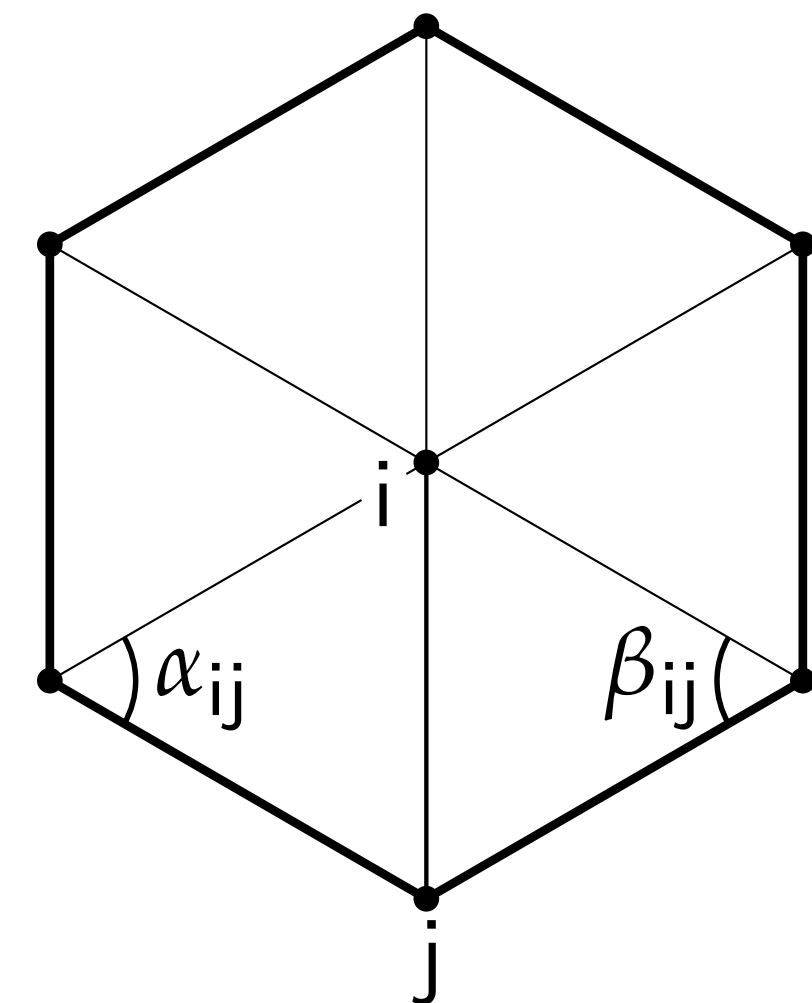


$$\frac{4u_{ij} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}}{h^2}$$

(actually, this becomes that)



TRIANGLE MESH



$$\frac{1}{2} \sum_j (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)$$

- Also not too hard on point clouds, polygon meshes, ...

# Numerically Solving the Laplace Equation

- Want to solve  $\Delta u = 0$
- Plug in one of our discretizations, e.g.,

	$c$	
$d$	$a$	$b$
	$e$	

$$\frac{4a - b - c - d - e}{h} = 0$$
$$\iff a = \frac{1}{4}(b + c + d + e)$$

- Oh: if we have a solution, then each value must be the average of the neighboring values.
- How do we solve this?
- One idea: keep averaging with neighbors! (“Jacobi method”)
- Correct, but *slow*. We’ll discuss faster methods next lecture.

# Boundary Conditions for Discrete Laplace

- What values do we use to compute averages near the

	$c$	
?	$a$	$b$
	$e$	

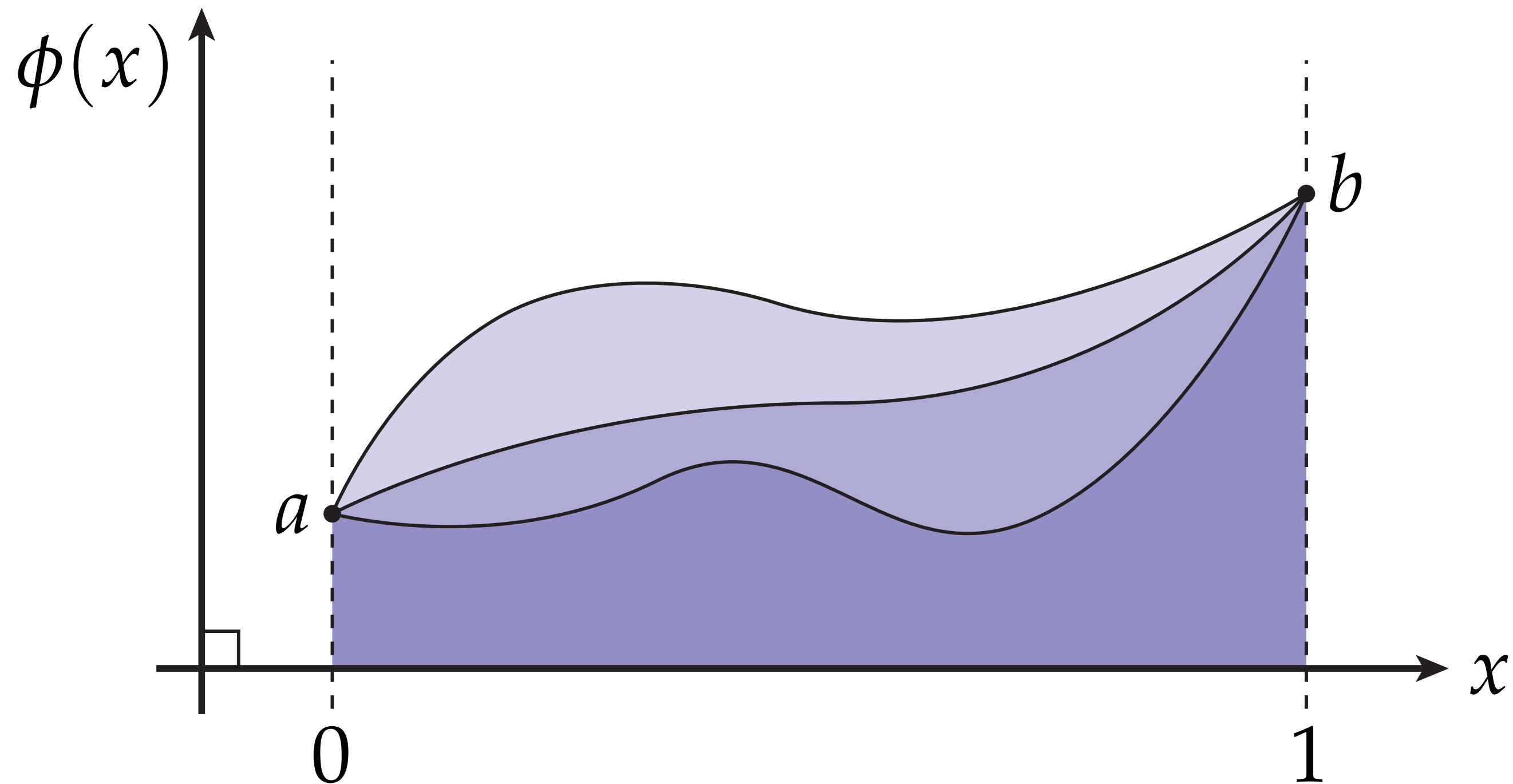
$$a = \frac{1}{4} (b + c + ? + e)$$

- **A: We get to choose**—this is the data we want to interpolate!
- **Two basic boundary conditions:**
  1. ***Dirichlet***—boundary data always set to fixed values
  2. ***Neumann***—specify derivative (difference) across boundary
- **Also mixed (*Robin*) boundary conditions (and more, in general)**



# Dirichlet Boundary Conditions

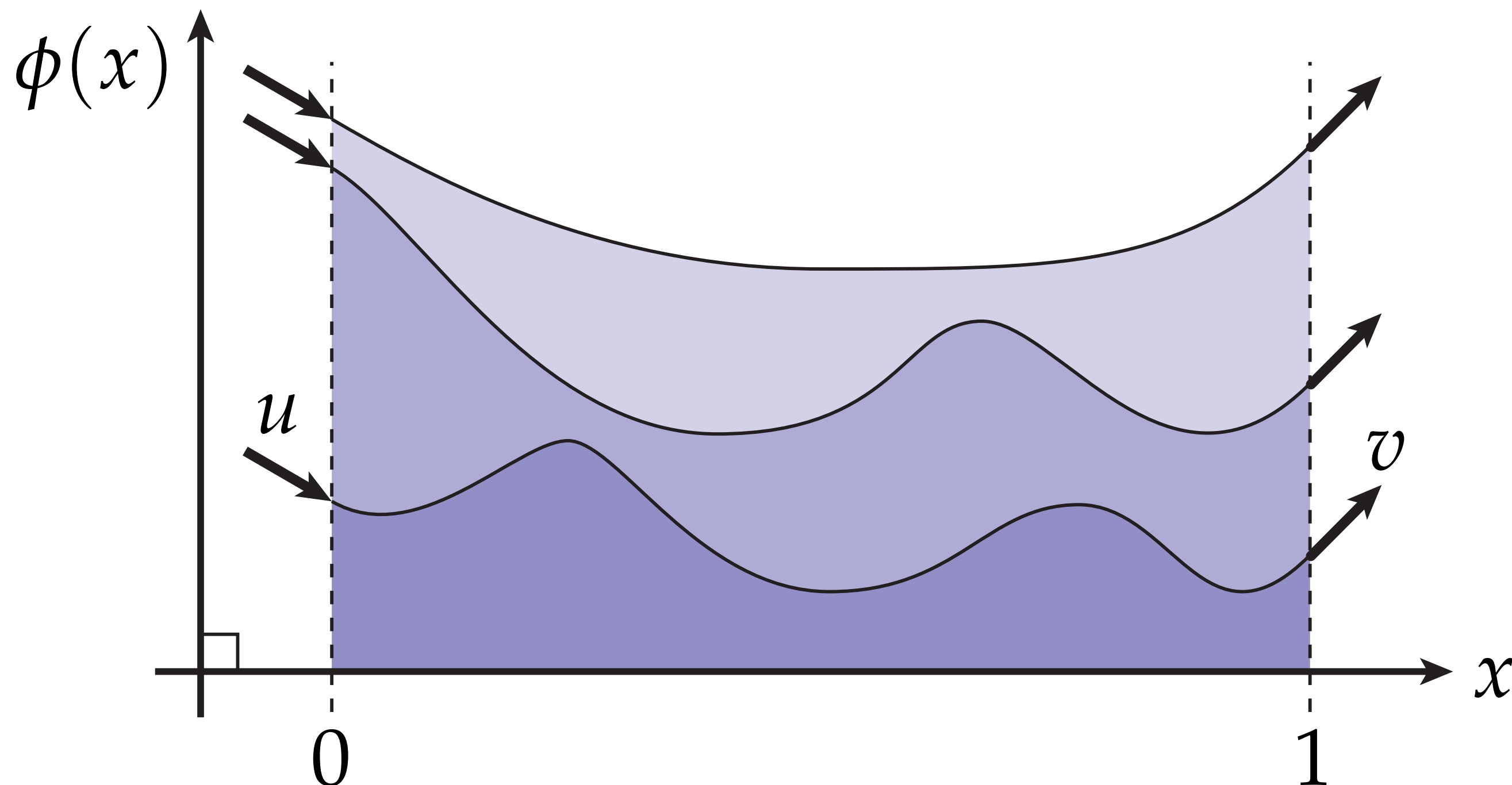
- Let's go back to smooth setting, function on real line
- *Dirichlet* means "prescribe values"
- E.g.,  $\Phi(0)=a$ ,  $\Phi(1) = b$



- Many possible functions "in between"!

# Neumann Boundary Conditions

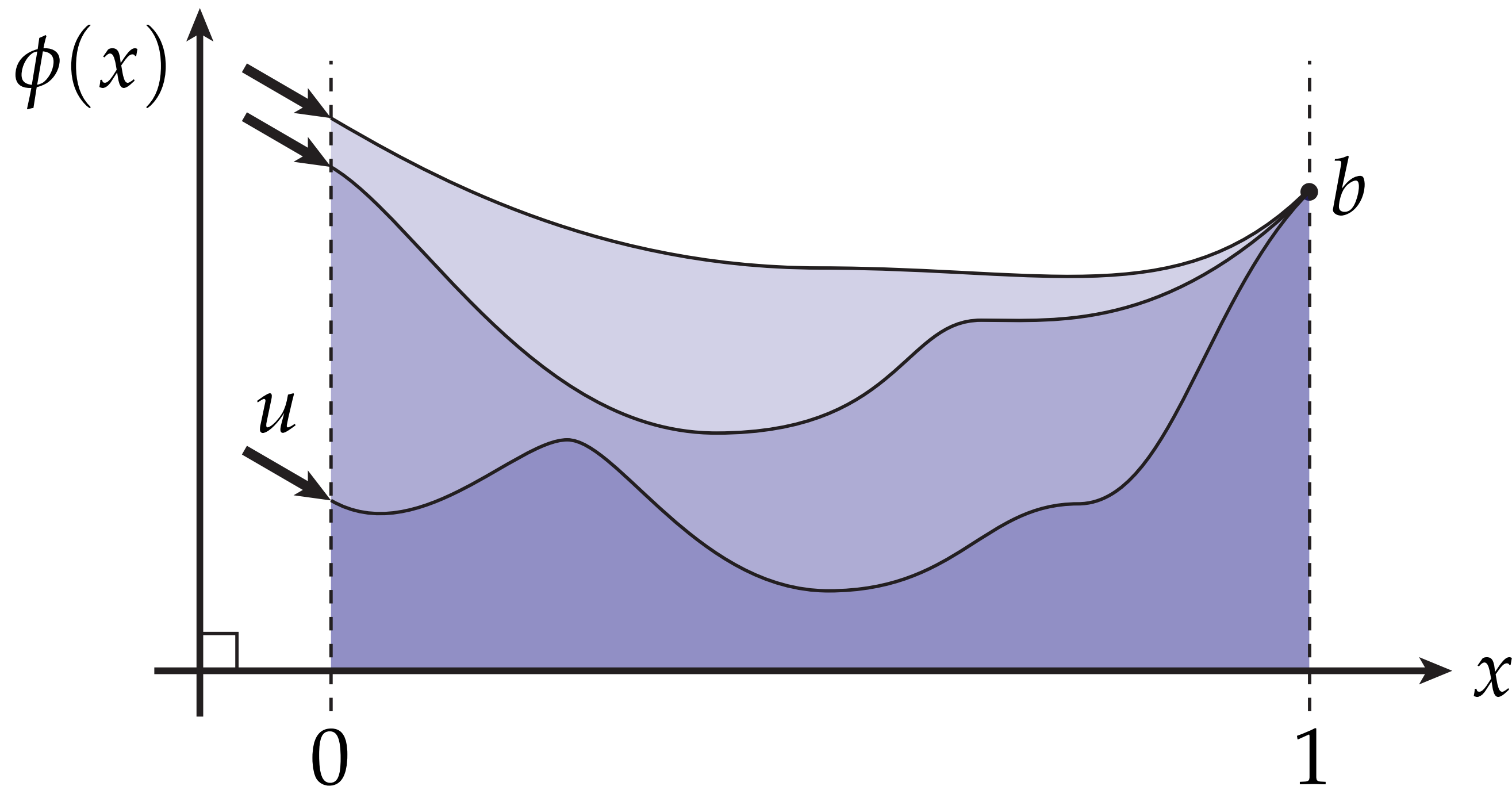
- *Neumann* means “prescribe derivatives”
- E.g.,  $\Phi'(0)=u$ ,  $\Phi'(1) = v$



- Again, many possible functions!

# Both Neumann & Dirichlet

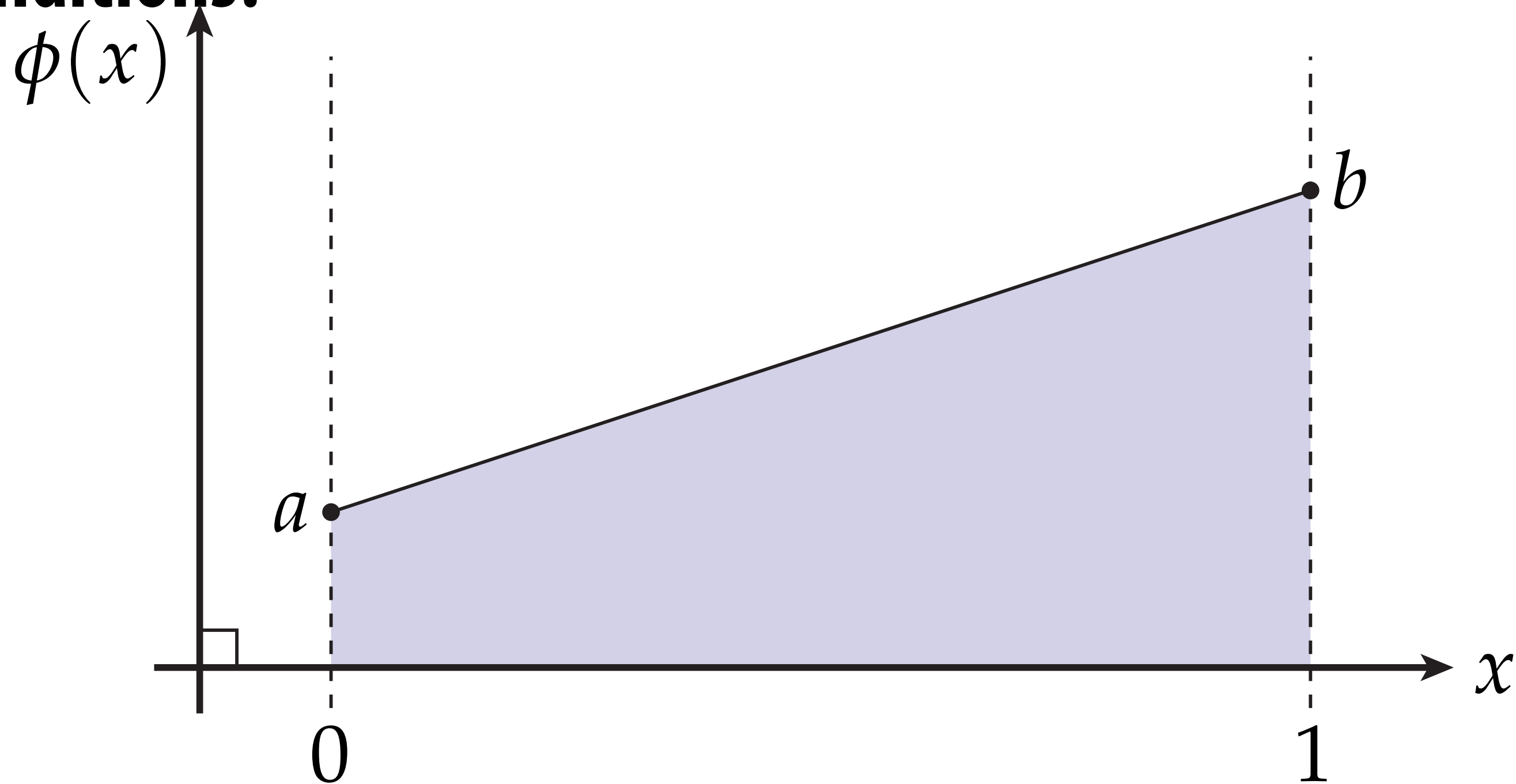
- Or: prescribe some values, some derivatives
- E.g.,  $\Phi'(0)=u, \Phi(1) = b$



- Q: What about  $\Phi'(1)=v, \Phi(1) = b$ ? Does that work?
- Q: What about  $\Phi'(0) + \Phi(0) = p, \Phi'(1) + \Phi(1) = q$ ? (*Robin*)

# 1D Laplace w/ Dirichlet BCs

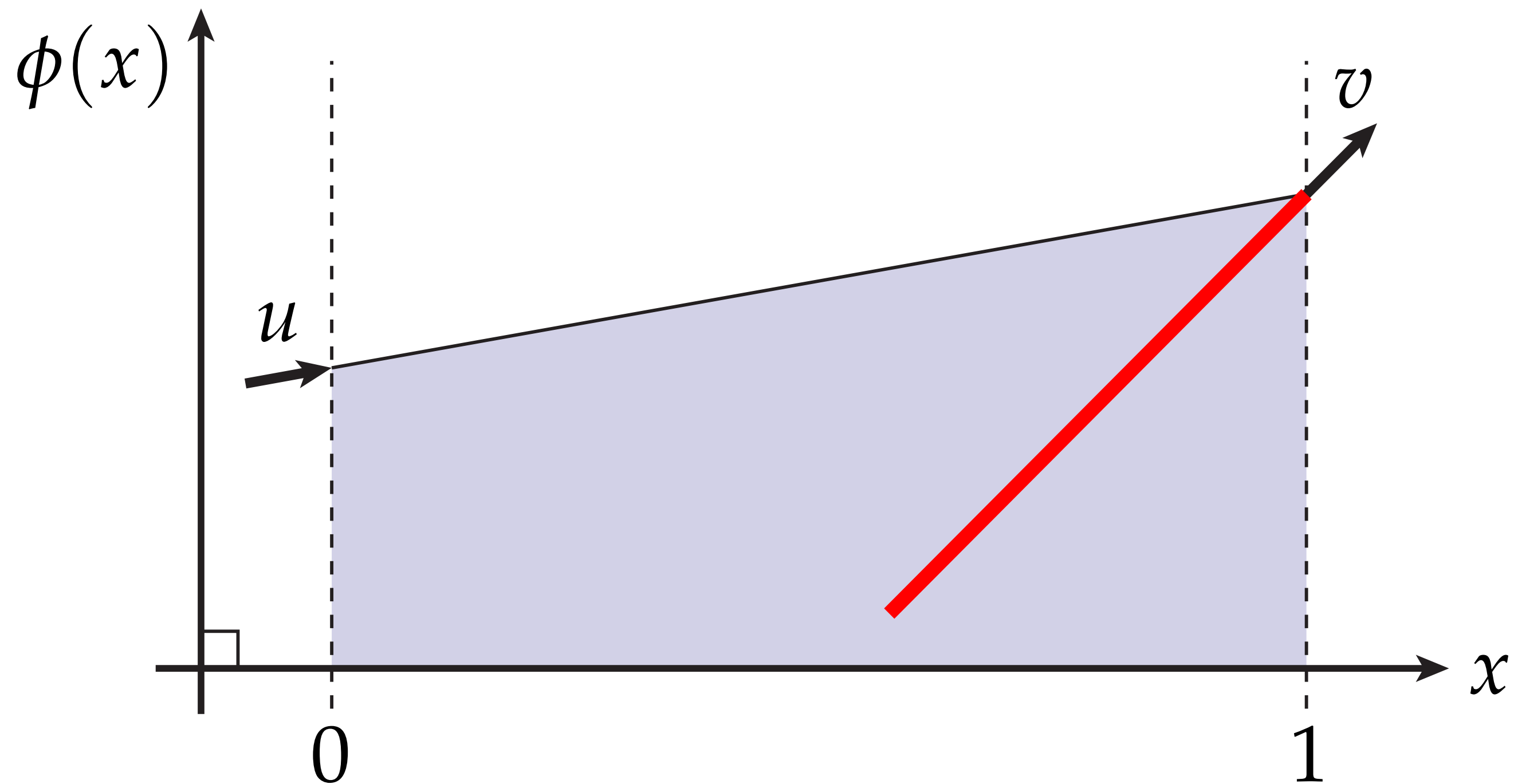
- 1D Laplace:  $\partial^2\Phi/\partial x^2 = 0$
- Solutions:  $\Phi(x) = cx + d$
- Q: Can we *always* satisfy given Dirichlet boundary conditions?



- Yes: a line can interpolate any two points.

# 1D Laplace w/ Neumann BCs

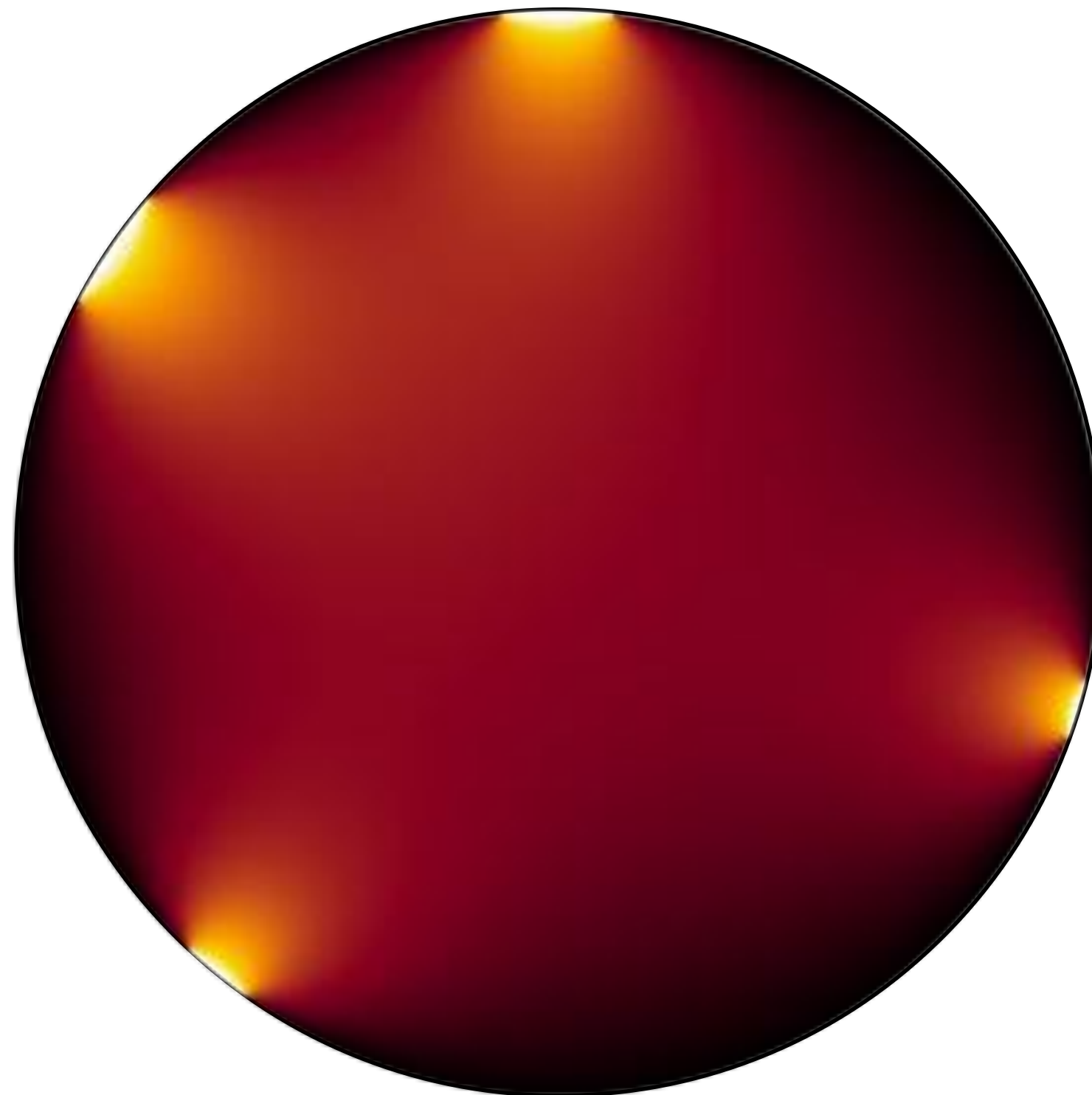
- What about Neumann BCs?
- Q: Can we prescribe the derivative at both ends?



- No! A line has only one slope.
- In general, solution to a PDE may not exist for given BCs.

# 2D Laplace w/ Dirichlet BCs

- 2D Laplace:  $\Delta \Phi = 0$
- Q: Can satisfy any Dirichlet BCs? (given data along boundary)



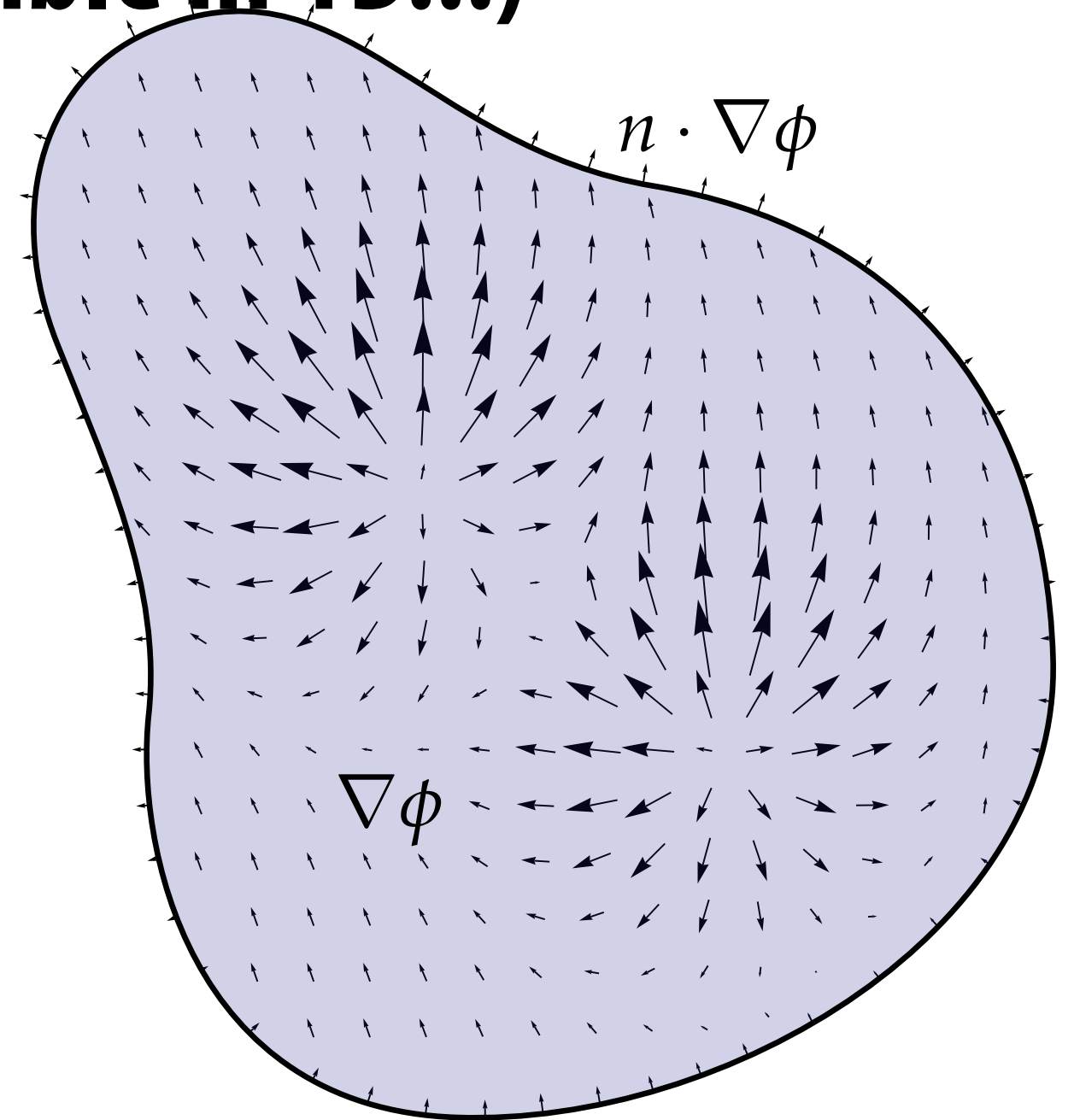
- Yes: Laplace is long-time solution to heat flow
- Data is “heat” at boundary. Then just let it flow...

# 2D Laplace w/ Neumann BCs

- What about Neumann BCs for  $\Delta\Phi=0$ ?
- Neumann BCs prescribe derivative in normal direction:  
 $n \cdot \nabla\Phi$
- Q: Can it always be done? (Wasn't possible in 1D...)

$$\int_{\partial\Omega} n \cdot \nabla\phi = \int_{\Omega} \nabla \cdot \nabla\phi = \int_{\Omega} \Delta\phi \stackrel{!}{=} 0$$

- Should be called, "what goes in must come out theorem!"
- Can't have a solution unless the net flux through the boundary is zero.
- Numerical software will *not* always tell you if there's a problem! (Especially if you wrote it yourself...)



# Solving the Heat Equation

- Back to our three model equations, want to solve *heat eqn.*

$$\dot{u} = \Delta u$$

- Just saw how to discretize Laplacian
- Also know how to do time (forward Euler, backward Euler, ...)
- E.g., forward Euler:

$$u^{k+1} = u^k + \Delta u^k$$

- Q: On a grid, what's our overall update now at  $u_{i,j}$ ?

$$u_{i,j}^{k+1} = u^k + \frac{\tau}{h^2} (4u_{i,j}^k - u_{i+1,j}^k - u_{i-1,j}^k - u_{i,j+1}^k - u_{i,j-1}^k)$$

- *Not* hard to implement! Loop over grid, add up some neighbors.



# Solving the Wave Equation

- Finally, wave equation:

$$\ddot{u} = \Delta u$$

- Not much different; now have 2nd derivative in time
- By now we've learned two different techniques:

- Convert to two 1st order (in time) equations:

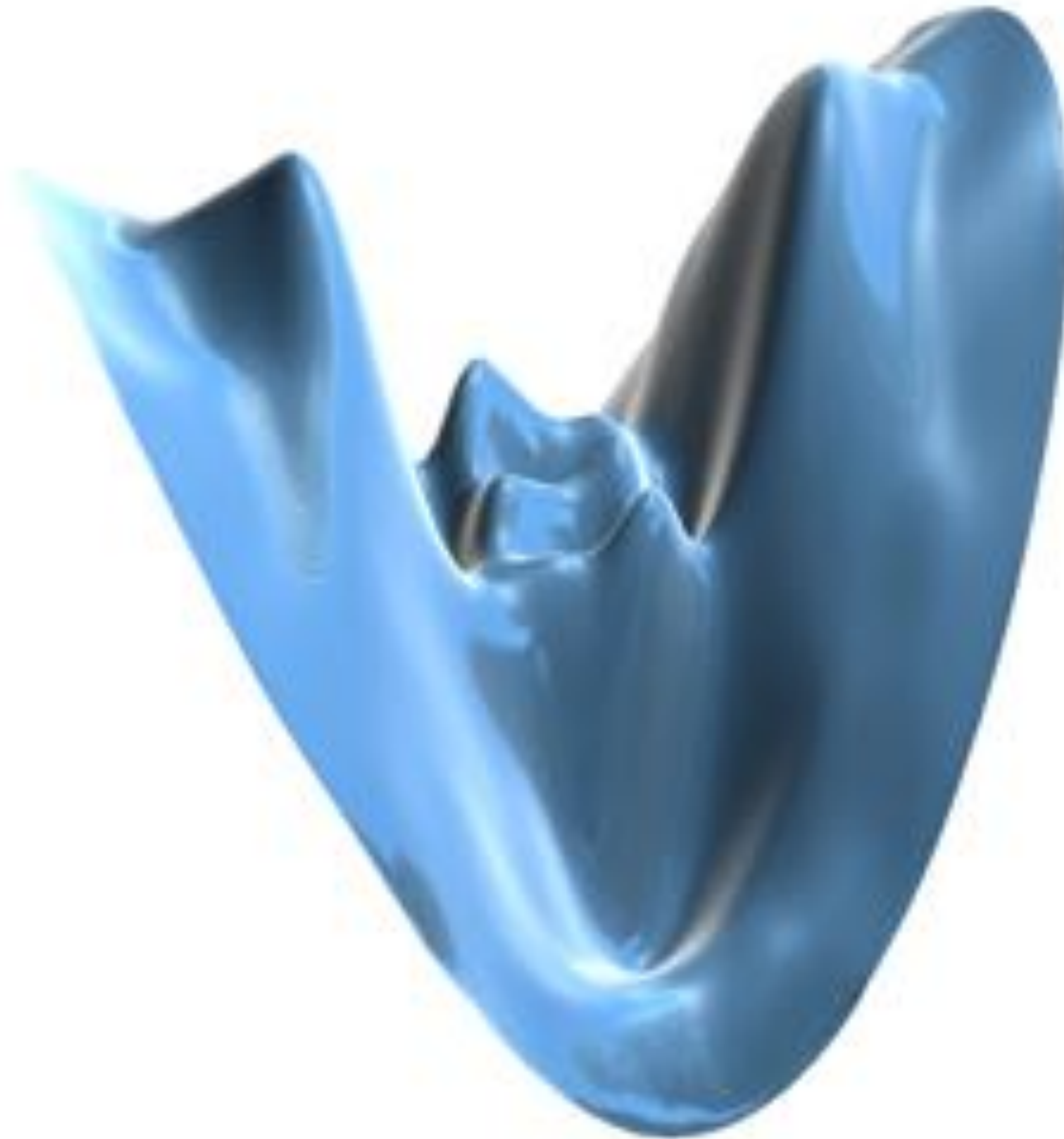
$$\dot{u} = v, \quad \dot{v} = \Delta u$$

- Or, use centered difference (like Laplace) in time:

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{\tau^2} = \Delta u^k$$

- Plus all our choices about how to discretize Laplacian.
- *So many choices!* And many, many (*many*) more we didn't discuss.

# DEMO: Model PDEs on a Triangle Mesh

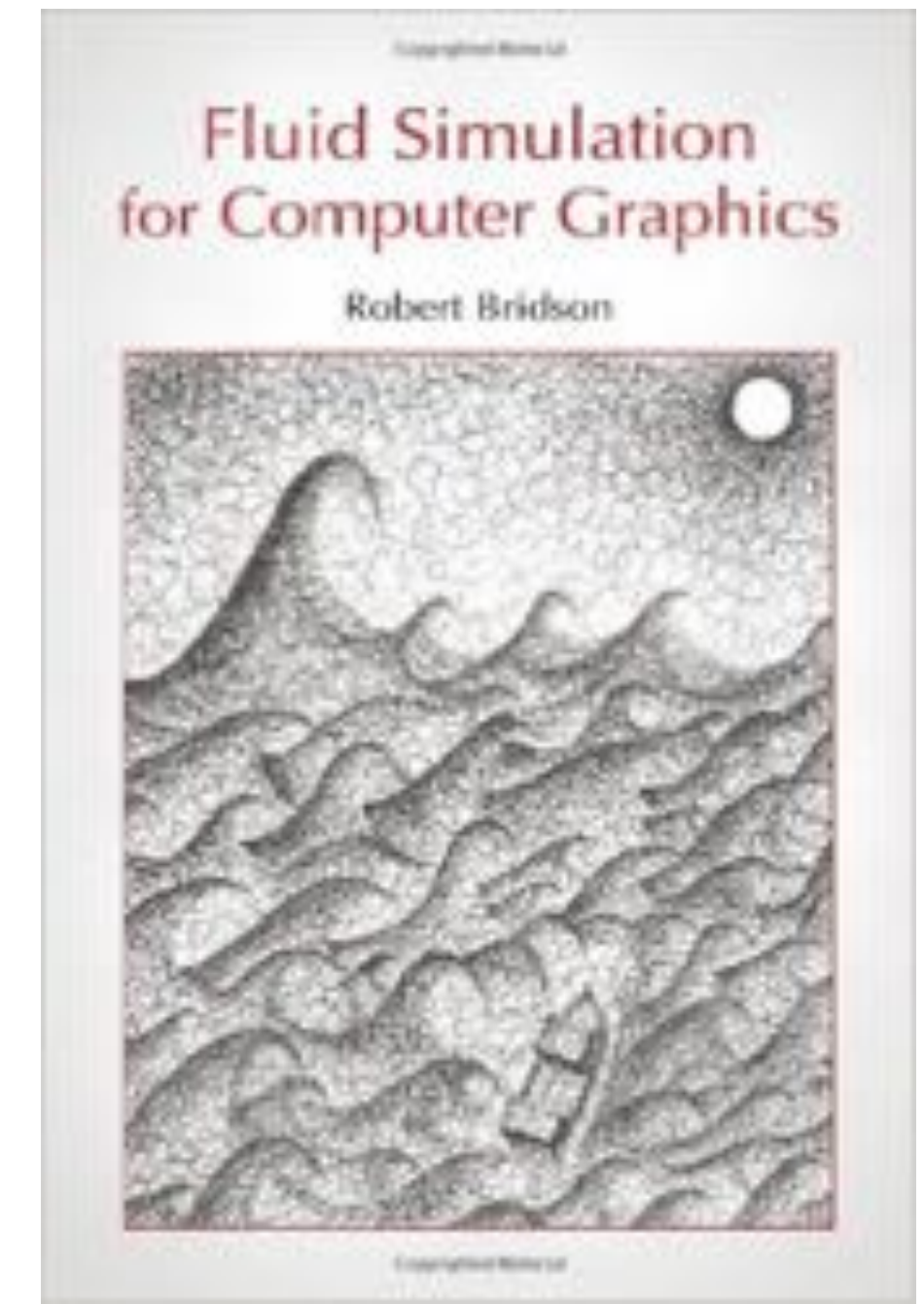


**Wait, what about all those  
cool fluids and stuff?**

# Want to Know More?

- There are some good books:
- And papers:

<http://www.physicsbasedanimation.com/>



- Also, what did the folks who *wrote* these books & papers read?

