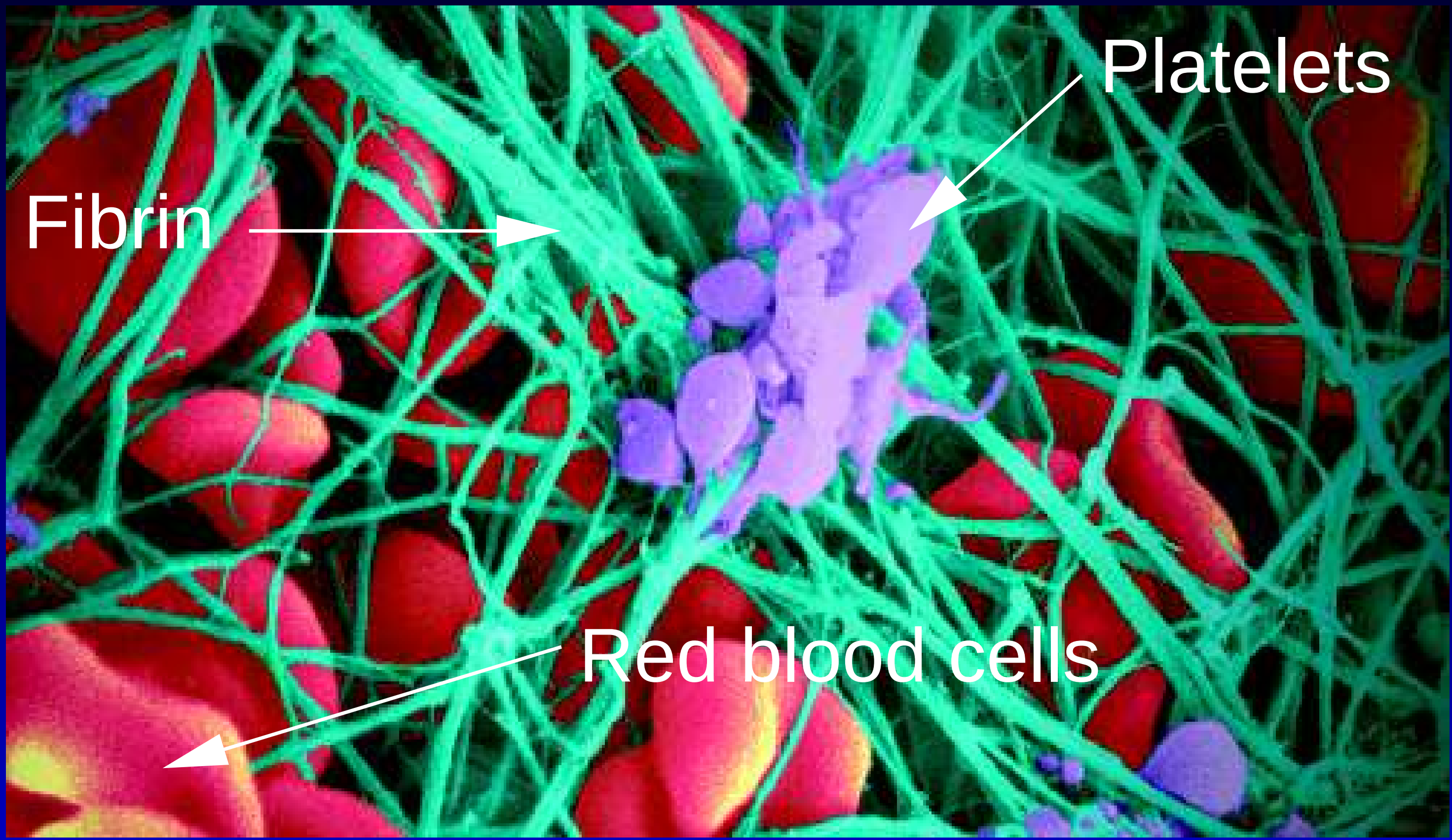
# A Computational Method for Simulating the Interaction between Fluid and Elastic Structures

Elijah Newren

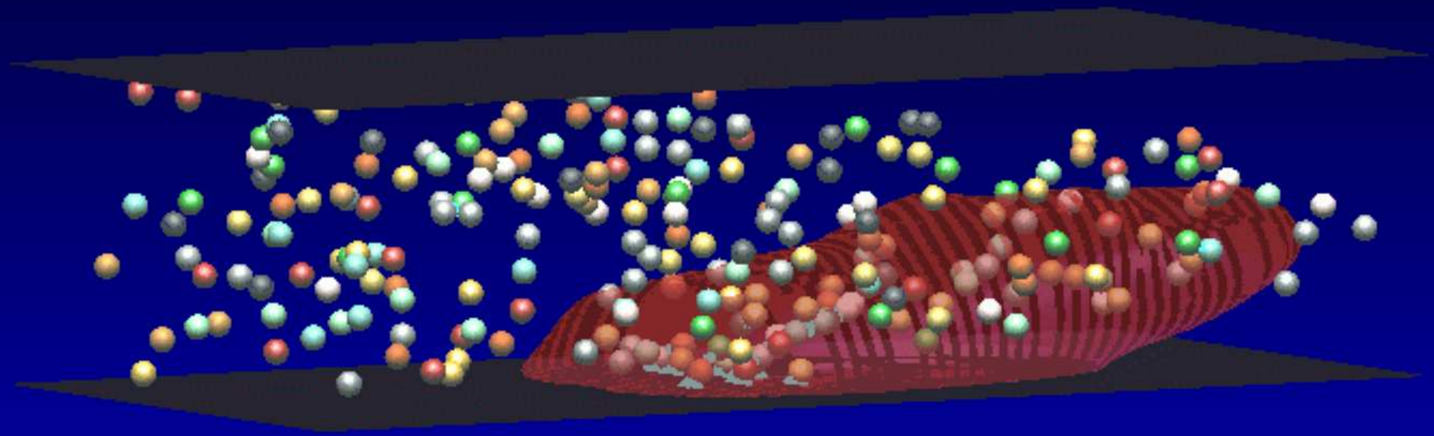
Department of Mathematics
University of Utah

# Motivation




A colorized scanning electron micrograph of a blood clot formed *in vitro* without flow (Image by J. Weisel)

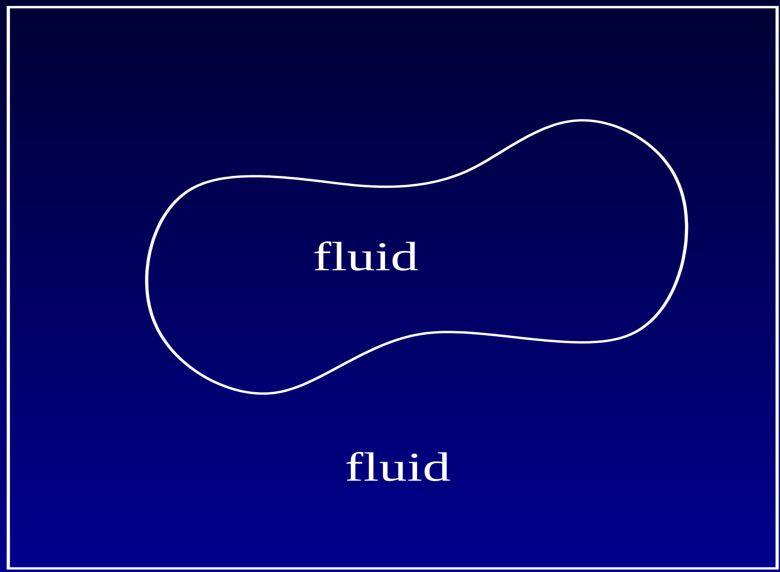
# Motivation



Simulation of Platelet Aggregation by H. Yu and A. Fogelson

# Simple Model Problem

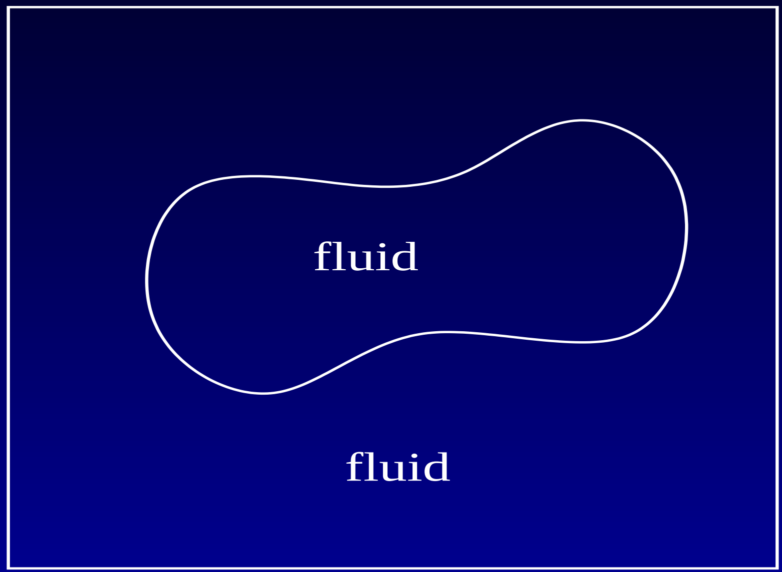
Think “2D water balloon”



# Simple Model Problem

Think "2D water balloon"

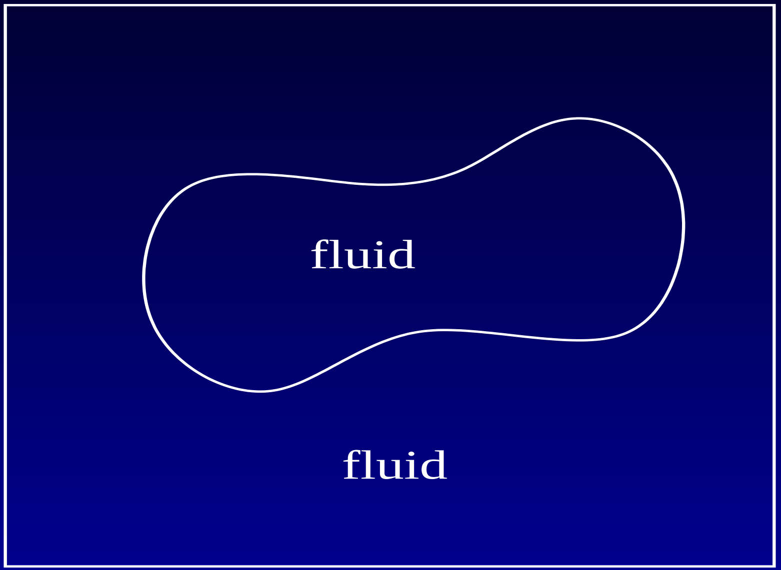
- Flow past multiple, moving, deformable objects



# Simple Model Problem

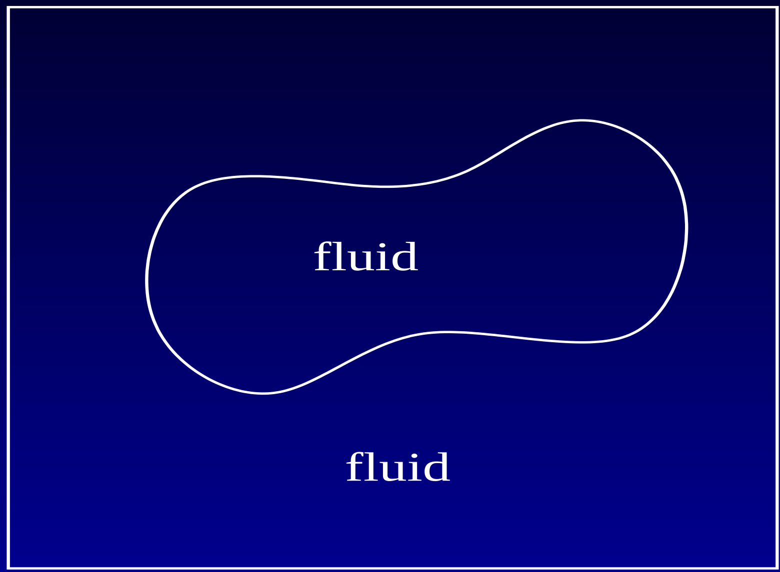
Think "2D water balloon"

- Flow past multiple, moving, deformable objects
- One possiblity: curve fitting grids



# Simple Model Problem

Think "2D water balloon"




- Flow past multiple, moving, deformable objects
- One possiblity: curve fitting grids
- Much simpler alternatives: Immersed Boundary and Immersed Interface Methods

# Key Idea

Replace elastic structures by a singular force exerted on the surrounding fluid

# Key Idea

Replace elastic structures by a singular force exerted on the surrounding fluid

- Allows a single set of fluid dynamics equations to hold in the entire domain with no internal boundary conditions

# Key Idea

Replace elastic structures by a singular force exerted on the surrounding fluid

- Allows a single set of fluid dynamics equations to hold in the entire domain with no internal boundary conditions
- Results in two grids being used

# Key Idea

Replace elastic structures by a singular force exerted on the surrounding fluid

- Allows a single set of fluid dynamics equations to hold in the entire domain with no internal boundary conditions
- Results in two grids being used
  - Fluid variables tracked on a structured Cartesian grid

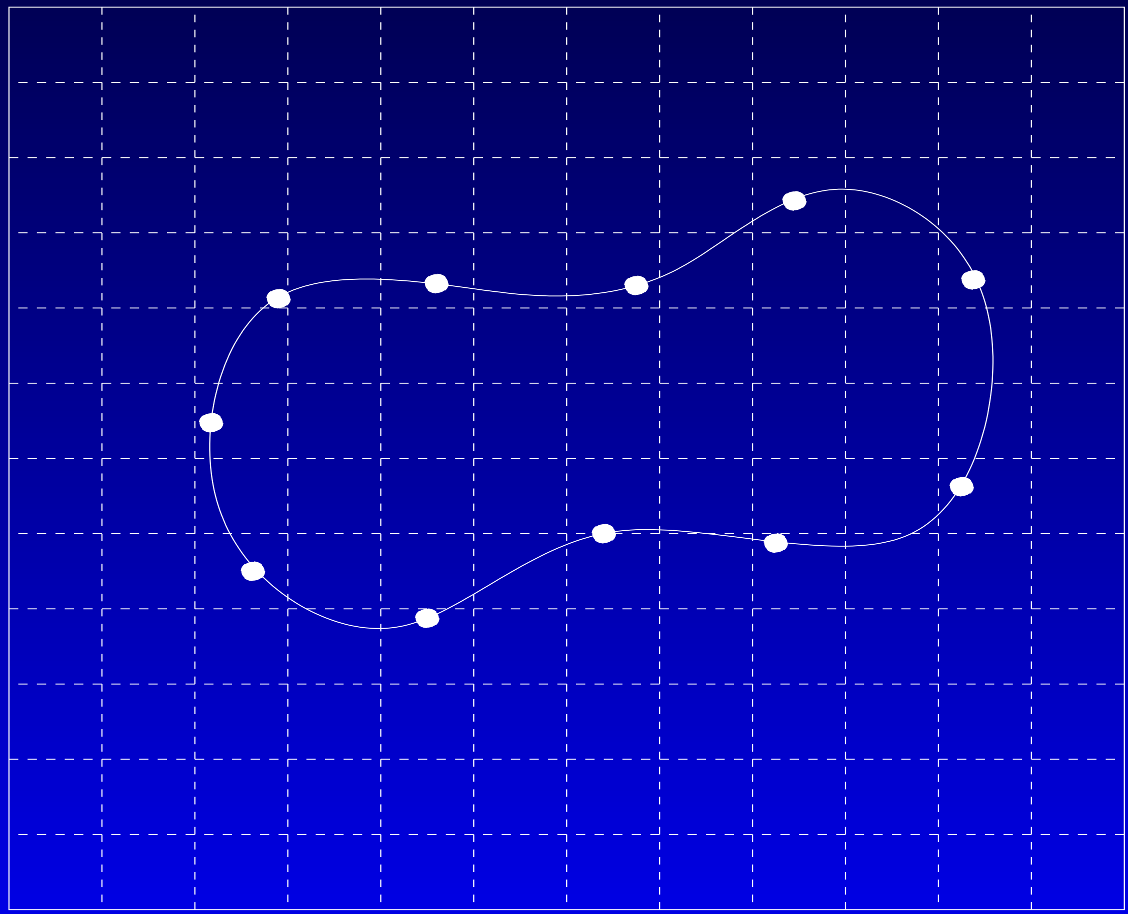
# Key Idea

Replace elastic structures by a singular force exerted on the surrounding fluid

- Allows a single set of fluid dynamics equations to hold in the entire domain with no internal boundary conditions
- Results in two grids being used
    - Fluid variables tracked on a structured Cartesian grid
    - Structure variables tracked on a *moving* irregular surface mesh

# Simple Model Problem

Mixed Eulerian-Lagrangian method; fixed Cartesian grid for fluid variables, *moving* irregular grid for the elastic structure.



# IB/II Method

$$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$$

# IB/II Method

$$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$$

Communicate forces, $\mathbf{F}$, to Eulerian grid

# IB/II Method

$$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$$

Communicate forces, $\mathbf{F}$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

# IB/II Method

$$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$$

Communicate forces, $\mathbf{F}$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X})$$

# IB/II Method

$$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$$

Communicate forces, $\mathbf{F}$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X})$$

Update structure position using $\mathbf{U}$

# IB/II Method

$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$

Communicate forces, $\mathbf{F}$, to Eulerian grid

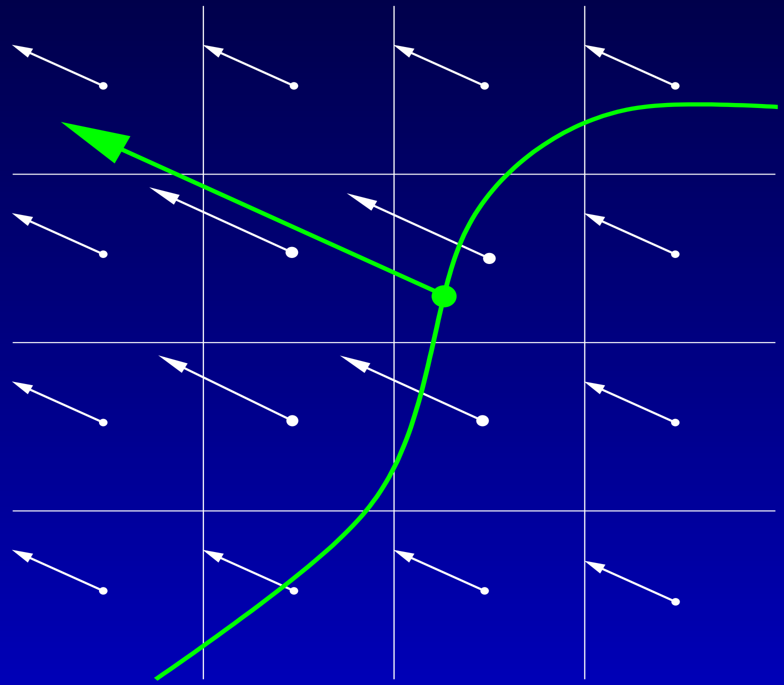
Solve N.S. Equations to get new velocity, $\mathbf{u}$

$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X})$

Update structure position using $\mathbf{U}$

# Setting up grid from forces

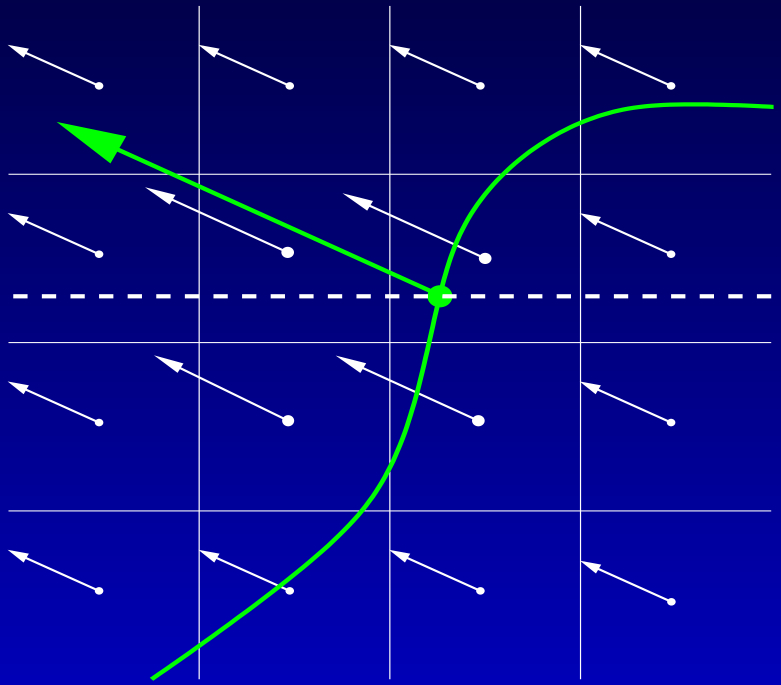
Immersed Boundary



Spreading the singular force

# Setting up grid from forces

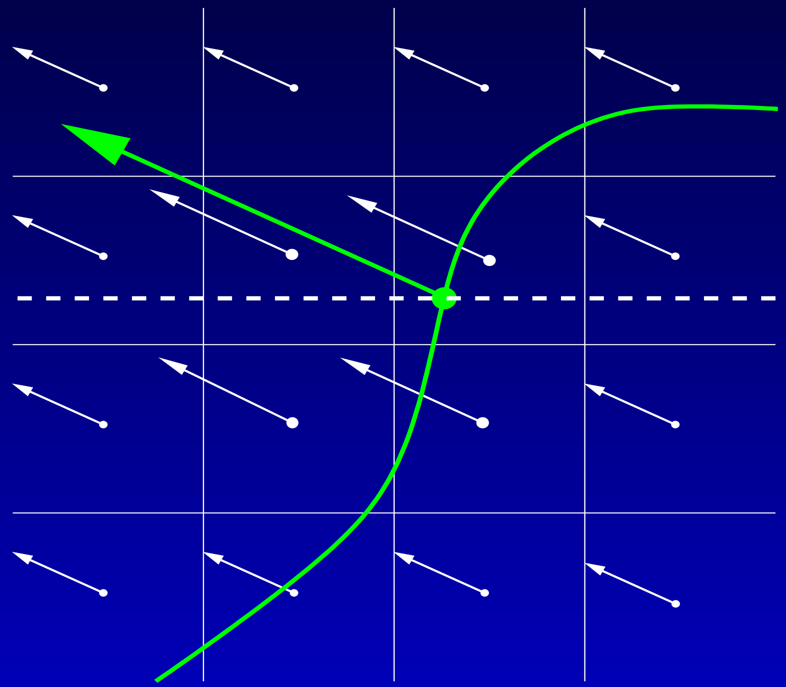
Immersed Boundary



Spreading the singular force

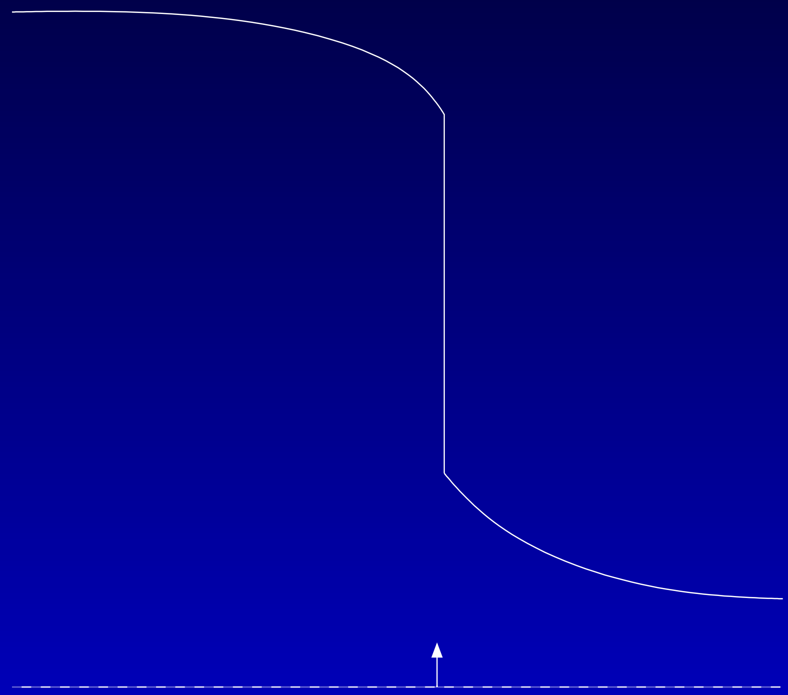
# Setting up grid from forces

Immersed Boundary



Spreading the singular force

Immersed Interface



Handling known discontinuities (e.g. in pressure) due to forces

# Making the solver implicit

# Making the solver implicit

$$\mathbf{F} = \text{Compute Forces}(\mathbf{X})$$

Communicate forces, $\mathbf{F}$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X})$$

Update structure position using $\mathbf{U}$

# Making the solver implicit

$\mathbf{F} = \text{Compute Forces}(\mathbf{X}^n)$

Communicate forces, $\mathbf{F}$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X})$

Update structure position using $\mathbf{U}$

Usage of old positions results in explicit method

# Making the solver implicit

$\mathbf{F}^n = \text{Compute Forces}(\mathbf{X}^n)$

Communicate forces, $\mathbf{F}^n$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X})$

Update structure position using $\mathbf{U}$

Usage of old positions results in explicit method

# Making the solver implicit

$$\mathbf{F}^n = \text{Compute Forces}(\mathbf{X}^n)$$

Communicate forces, $\mathbf{F}^n$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$$\text{U} = \text{interpolate}(\mathbf{u}, \mathbf{X}^n)$$

Update structure position using U

Usage of old positions results in explicit method

# Making the solver implicit

$$\mathbf{F}^{n+1} = \text{Compute Forces}(\mathbf{X}^{n+1})$$

Communicate forces, $\mathbf{F}^{n+1}$, to Eulerian grid

Solve N.S. Equations to get new velocity, $\mathbf{u}$

$$\text{U} = \text{interpolate}(\mathbf{u}, \mathbf{X}^{n+1})$$

Update structure position using U

# Making the solver implicit

$$\mathbf{F}^{n+1} = \text{Compute Forces}(\mathbf{X}^{n+1})$$

Communicate forces, $\mathbf{F}^{n+1}$, to Eulerian grid

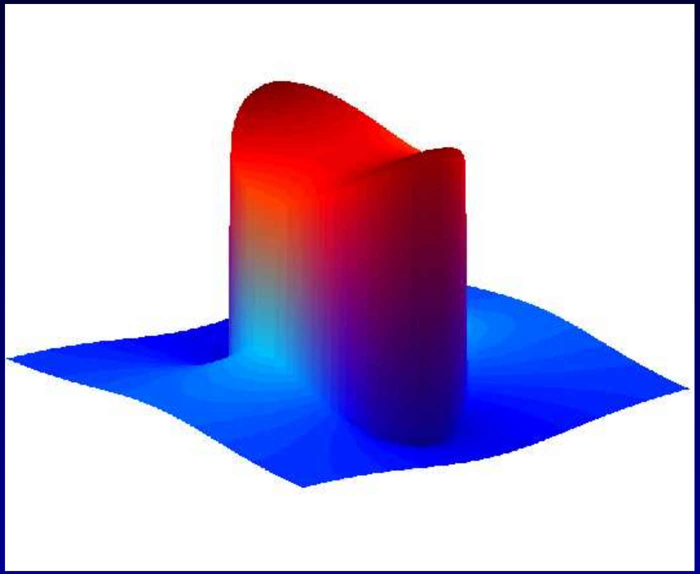
Solve N.S. Equations to get new velocity, $\mathbf{u}$

$$\mathbf{U} = \text{interpolate}(\mathbf{u}, \mathbf{X}^{n+1})$$

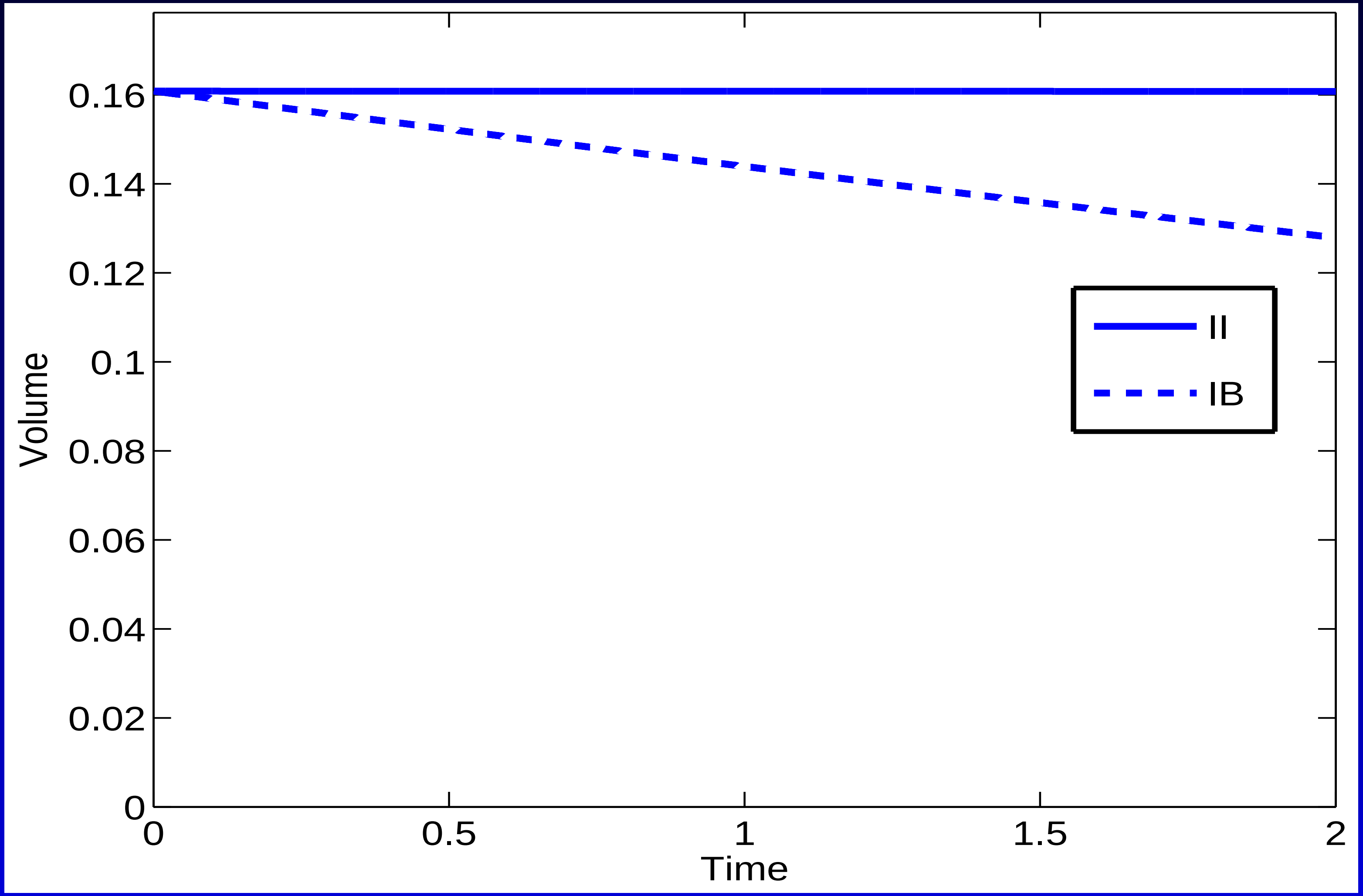
Update structure position using $\mathbf{U}$

Jacobian is large, dense, and involves nonlinear PDE solve

# Example problem



# Volume Conservation



# Extending to 3D

- Need to be able to evaluate tangents, normals, and various order derivatives anywhere along the interface

# Extending to 3D

- Need to be able to evaluate tangents, normals, and various order derivatives anywhere along the interface
- Using splines for a surface in 3D is somewhat messy (perhaps we're just lazy?)

# Extending to 3D

- Need to be able to evaluate tangents, normals, and various order derivatives anywhere along the interface
- Using splines for a surface in 3D is somewhat messy (perhaps we're just lazy?)
- Level Set method

# Extending to 3D

- Need to be able to evaluate tangents, normals, and various order derivatives anywhere along the interface
- Using splines for a surface in 3D is somewhat messy (perhaps we're just lazy?)
- Level Set method
  - Coalescing of objects

# Extending to 3D

- Need to be able to evaluate tangents, normals, and various order derivatives anywhere along the interface
- Using splines for a surface in 3D is somewhat messy (perhaps we're just lazy?)
- Level Set method
  - Coalescing of objects
  - Doesn't handle links between objects

# Extending to 3D

- Need to be able to evaluate tangents, normals, and various order derivatives anywhere along the interface
- Using splines for a surface in 3D is somewhat messy (perhaps we're just lazy?)
- Level Set method
  - Coalescing of objects
  - Doesn't handle links between objects
- Implementation underway using Radial Basis Functions

# Solution components

# Solution components

- (Approximate) Projection Method

# Solution components

- (Approximate) Projection Method
- Multigrid

# Solution components

- (Approximate) Projection Method
- Multigrid
- SAMRAI

# Solution components

- (Approximate) Projection Method
- Multigrid
- SAMRAI
- MPI

# Solution components

- (Approximate) Projection Method
- Multigrid
- SAMRAI
- MPI
- Splines $\longrightarrow$ Radial Basis Functions

# Solution components

- (Approximate) Projection Method
- Multigrid
- SAMRAI
- MPI
- Splines $\longrightarrow$ Radial Basis Functions
- Quasi-Newton Solver

# Solution components

- (Approximate) Projection Method
- Multigrid
- SAMRAI
- MPI
- Splines → Radial Basis Functions
- Quasi-Newton Solver
- Lapack, Valgrind, others

# Acknowledgements


- Aaron Fogelson, Grady Wright, Bob Guy, Mike Kirby
- Rich Hornung (LLNL)
- Mike Pernice (LANL)
- Krell Institute, DOE CSGF program