Scalable Simulation of Realistic Volume Fraction Red Blood Cell Flows through Vascular Networks

Matthew J. Morse

Courant Institute, NYU

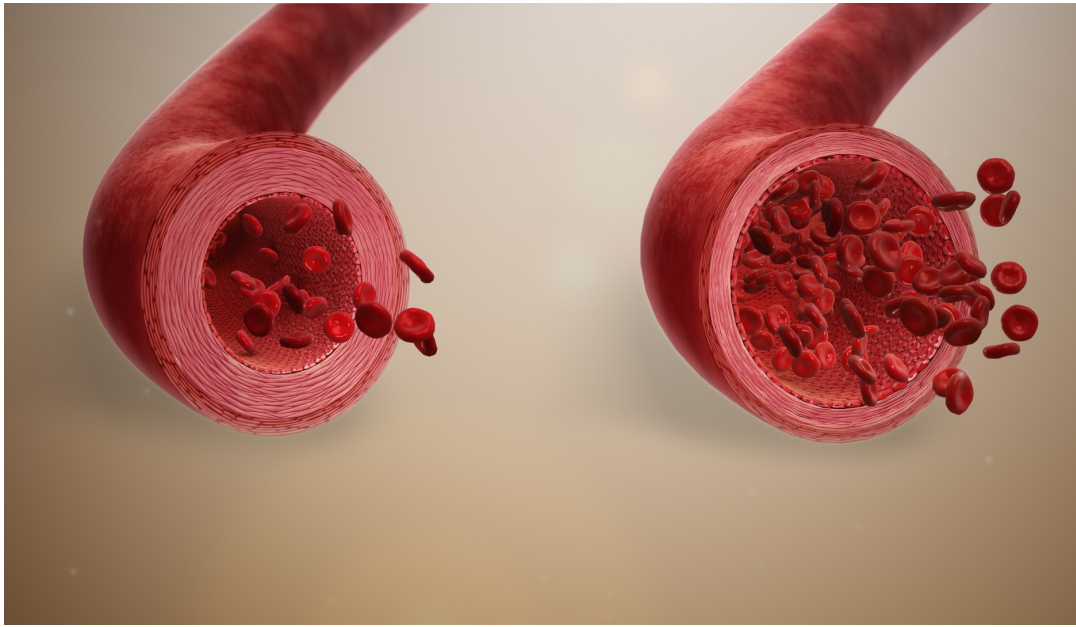
Joint work with Libin Lu, Abtin Rahimian, Georg Stadler, Denis Zorin

Outline

- **Motivation**
- Formulation, Numerics, Algorithms
- Results

Goal: Simulate red blood cell flow in capillaries

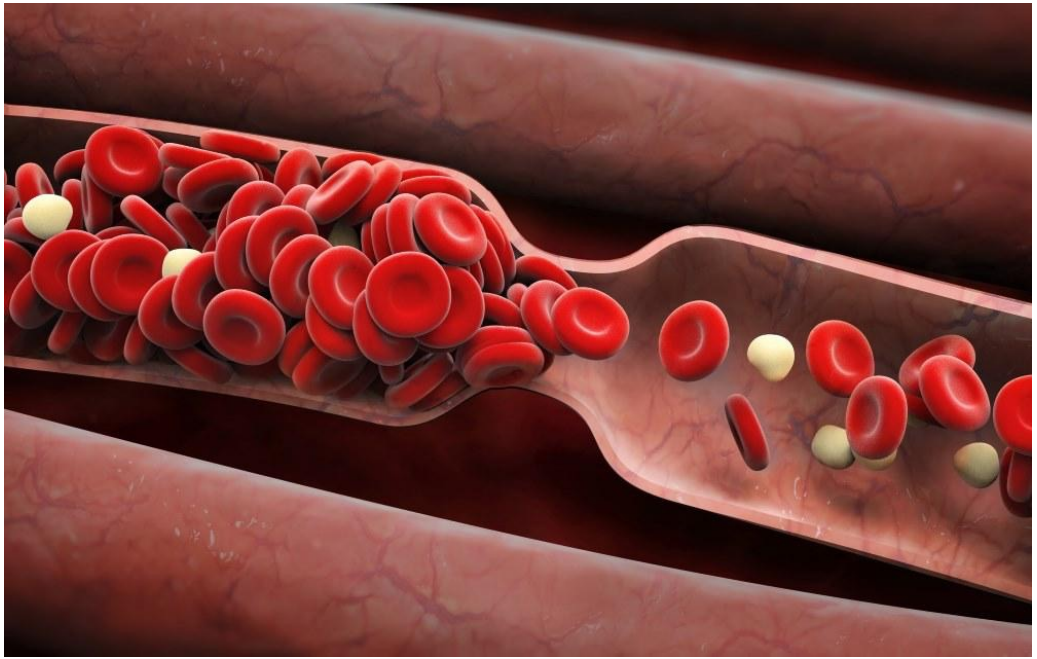
- Important biophysical phenomena:
  - vasoconstriction
  - vasodilation



https://en.wikipedia.org/wiki/Vasodilation

Goal: Simulate red blood cell flow in capillaries

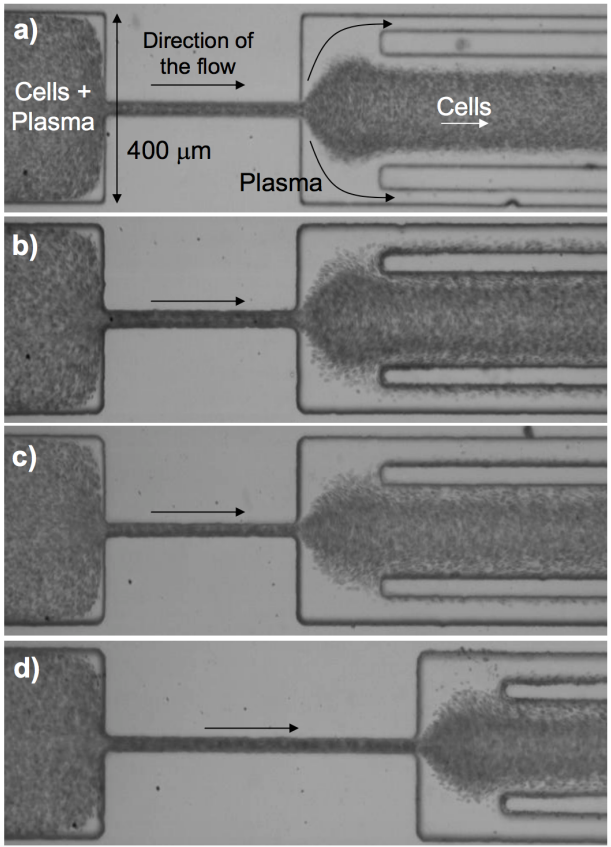
- Important biophysical phenomena:
  - vasoconstriction
  - vasodilation
  - blood clotting



shutterstock.com

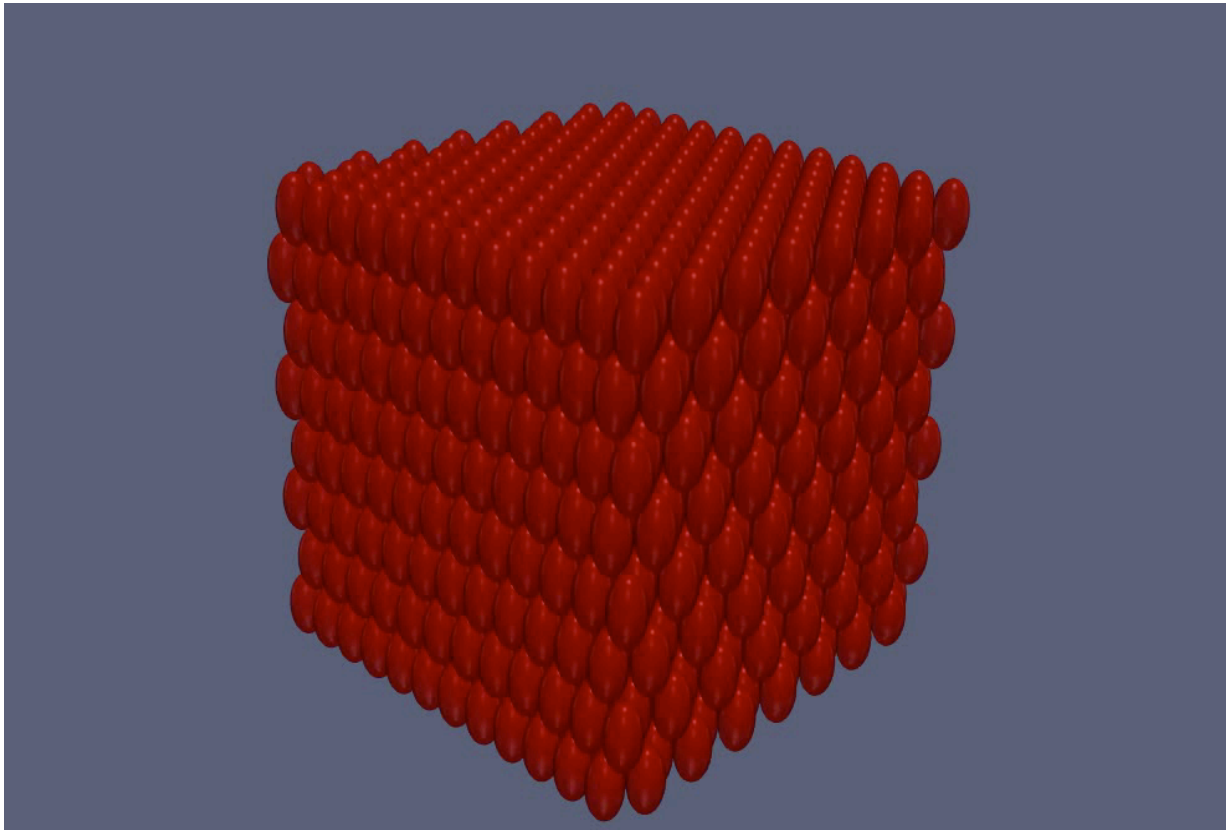
Goal: Simulate red blood cell flow in capillaries

- Important biophysical phenomena:
  - vasoconstriction
  - vasodilation
  - blood clotting
  - microfluidic device design



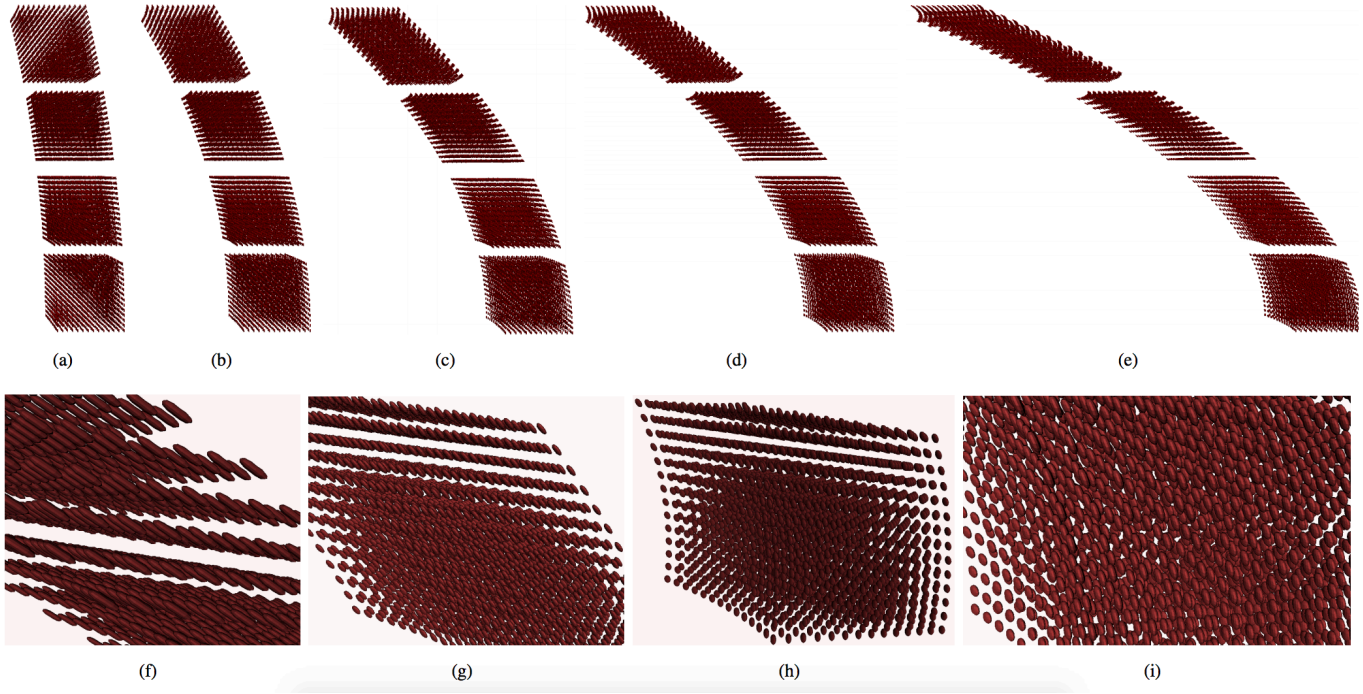
Drops, vesicles and red blood cells: Deformability and behavior under flow - Magalie Faivre, Thesis 2006

There are many free space codes...



Parallel contact-aware simulations of deformable particles in 3D Stokes flow, L. Lu et. al., arxiv 2018

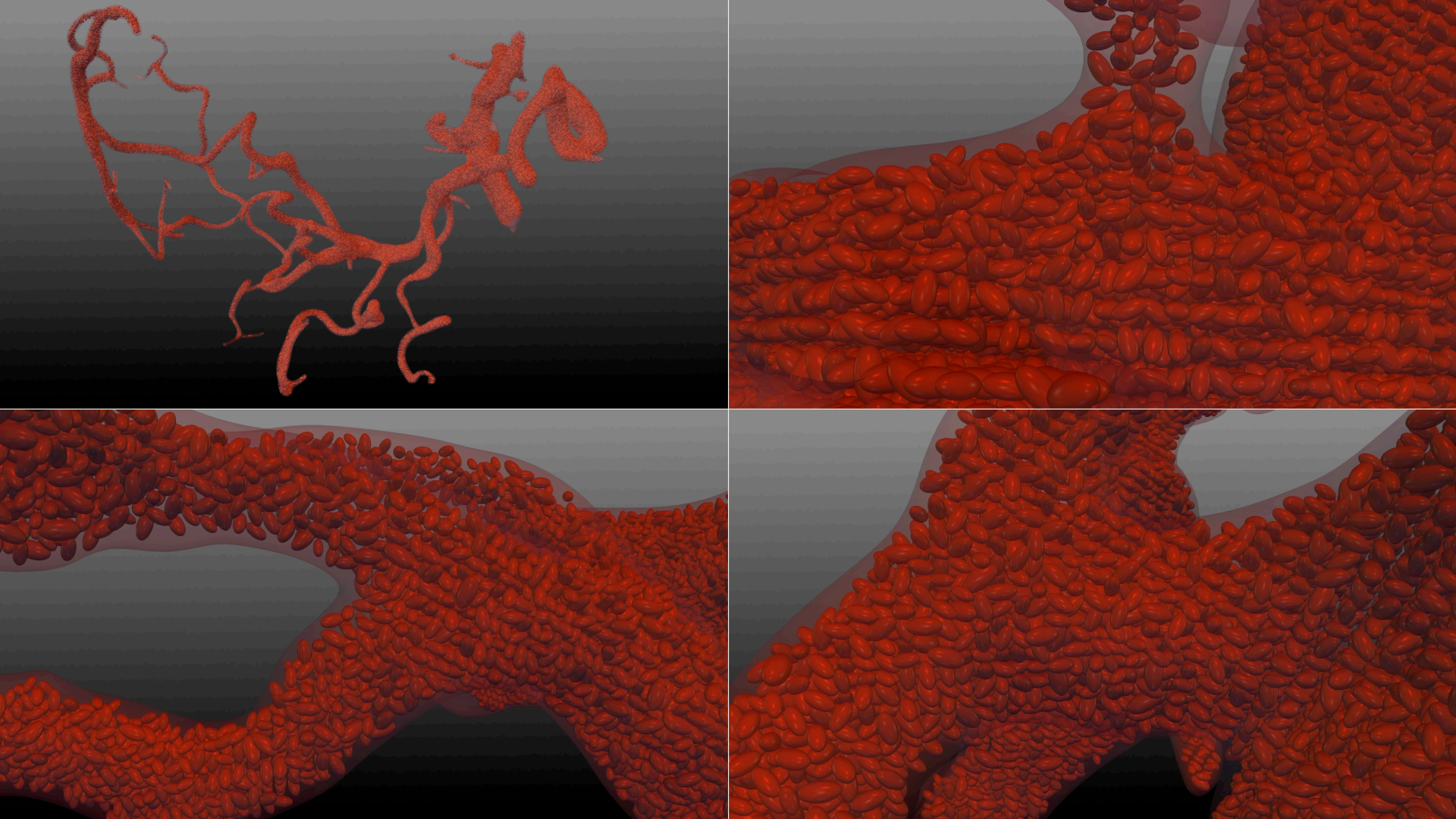
There are many free space codes…



"Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures." Rahimian et. al., SC 2010

… but few with boundaries

- Either:
  - Large scale, low-order accurate
  - Small scale, high-order accurate



Contribution

- Parallel Stokes boundary solver on complex geometry
- Parallel collision handling between RBCs + blood vessel
- Scaled to ~35k cores (resource limited)

Outline

- Motivation
- **Formulation, Numerics, Algorithms**
- Results

Stokes flow on RBCs

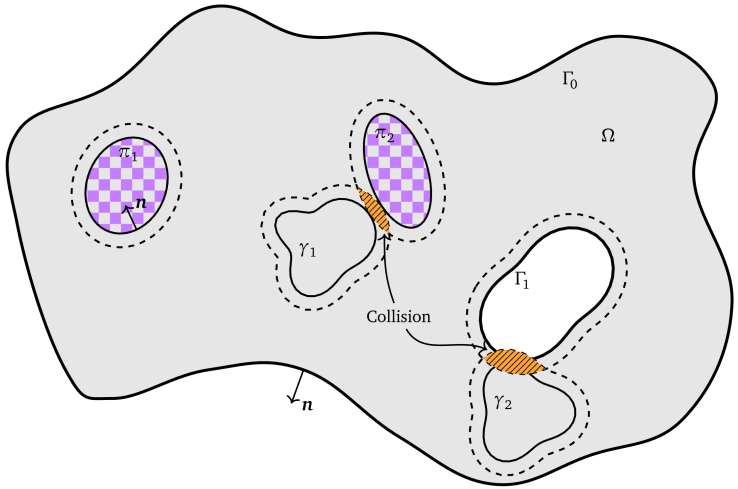
$$\mu\Delta\mathbf{u}(\mathbf{x}) + \nabla p(\mathbf{x}) = \mathbf{F}(\mathbf{x}), \quad \Delta \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega$$

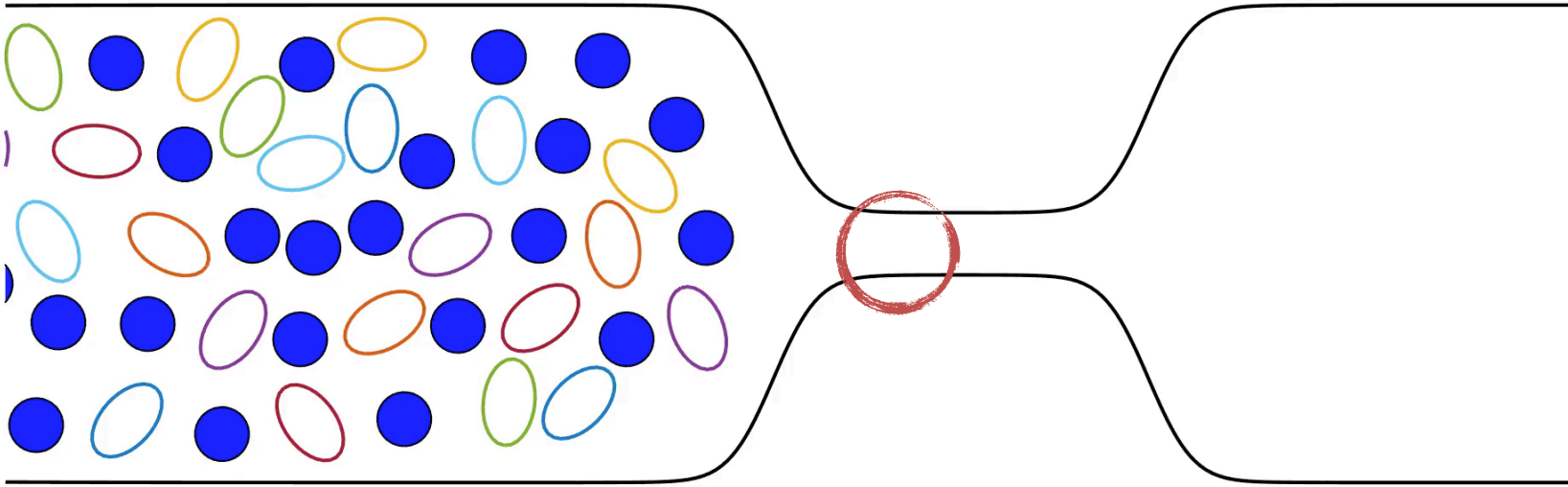
$$\mathbf{X}_t = \mathbf{u}(\mathbf{X}), \quad \mathbf{X} \in \gamma_i$$

$$\mathbf{F}(\mathbf{x}) = \sum_i \int_{\gamma_i} \mathbf{f}(\mathbf{y})\delta(\mathbf{x}-\mathbf{y})d\mathbf{y}$$

$$\mathbf{f}(\mathbf{y}) = \mathbf{f}_\sigma + \mathbf{f}_b$$



Collision handling is hard!



Collision handling is hard!

- Small time steps
- Fine spatial discretization
  - Too expensive
- Maintain accurate physics?
  - requires solving non-linear complementarity problem (NCP)

Collision handling is harder in parallel!

- Need to detect colliding geometry on other processors
- Need scalable algorithms to solve resulting non-linear complementarity problem

Stokes flow on RBCs

$$\mu\Delta\mathbf{u}(\mathbf{x}) + \nabla p(\mathbf{x}) = \mathbf{F}(\mathbf{x}), \quad \Delta \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega$$

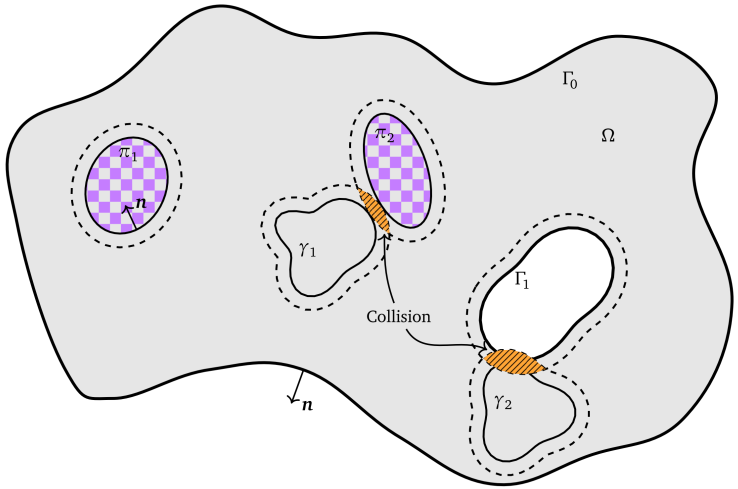
$$\mathbf{u}(\mathbf{x}) = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega$$

$$\mathbf{X}_t = \mathbf{u}(\mathbf{X}), \quad \mathbf{X} \in \gamma_i$$

$$\mathbf{F}(\mathbf{x}) = \sum_i \int_{\gamma_i} \mathbf{f}(\mathbf{y})\delta(\mathbf{x}-\mathbf{y})d\mathbf{y}$$

$$\mathbf{f}(\mathbf{y}) = \mathbf{f}_\sigma + \mathbf{f}_b + \mathbf{f}_c$$

$\mathbf{f}_c$: [Harmon et. al. 2011]: space-time interference volume



Boundary Integral Formulation

$$\mathbf{u} = \sum_i \mathbf{u}^{\gamma_i} + \mathbf{u}^{\Gamma}$$

- RBCs:

$$\mathbf{u}^{\gamma_i}(\mathbf{x}) = \int_{\gamma_i} \mathcal{S}(\mathbf{x}-\mathbf{y})\mathbf{f}(\mathbf{y})d\mathbf{y}, \quad \mathcal{S}(\mathbf{r}) = \frac{1}{8\mu\pi}\left(\frac{1}{\mathbf{r}} + \frac{\mathbf{r}\otimes\mathbf{r}}{\mathbf{r}^3}\right)$$

- Existing methods: [Veerapeneni et. al. JCP 2011], [Malhotra et. al. arxiv 2017, Lu et. al. arxiv 2018]

Boundary Integral Formulation

$$\mathbf{u} = \sum_i \mathbf{u}^{\gamma_i} + \mathbf{u}^{\Gamma}$$

- Vessel:

$$\mathbf{u}^{\Gamma}(\mathbf{x}) = \int_{\Gamma} \mathcal{D}(\mathbf{x}-\mathbf{y})\phi(\mathbf{y})d\mathbf{y}, \quad \mathcal{D}(\mathbf{r}) = \frac{\partial \mathcal{S}(\mathbf{r})}{\partial n}$$

$$\left(\frac{1}{2}I + D + N\right)\phi = \mathbf{g} - \sum_i \mathbf{u}^{\gamma_i}, \quad \mathbf{x} \in \Gamma$$

BIE Representation + Discretization

- RBCs:
  - Spherical harmonic representation
  - Semi-implicit time stepping
- Vessel:
  - Bezier polynomial patches
- Nyström discretization with spectrally accurate quadrature rules

Evaluating velocity

$$\mathbf{u}^{\Gamma}(\mathbf{x}) = \int_{\Gamma} \mathcal{D}(\mathbf{x}-\mathbf{y})\phi(\mathbf{y})d\mathbf{y}$$

$$\mathbf{u}^{\Gamma}(\mathbf{x}) = \sum_{i=1}^{N} \int_{P_i} \mathcal{D}(\mathbf{x}-\mathbf{y})\phi(\mathbf{y})d\mathbf{y}$$

$$\mathbf{u}^{\Gamma}(\mathbf{x}) \approx \sum_{i=1}^{N} \sum_{j=1}^{q} \mathcal{D}(\mathbf{x}-\mathbf{y}_i)\phi(\mathbf{y}_i)w_i$$

N-body sum

N-body sum —> FMM

- Accelerate with fast multipole method (FMM)
- All hydrodynamic interaction through FMMs
- Many high performance parallel FMMs exist; we choose PVFMM [Malhotra et. al., CCP 2015]

BIE: Pros and Cons

Pros:

- Linear Complexity
- High-order accuracy
- Dimension reduction
- No volume mesh
- Parallel scalability

Cons:

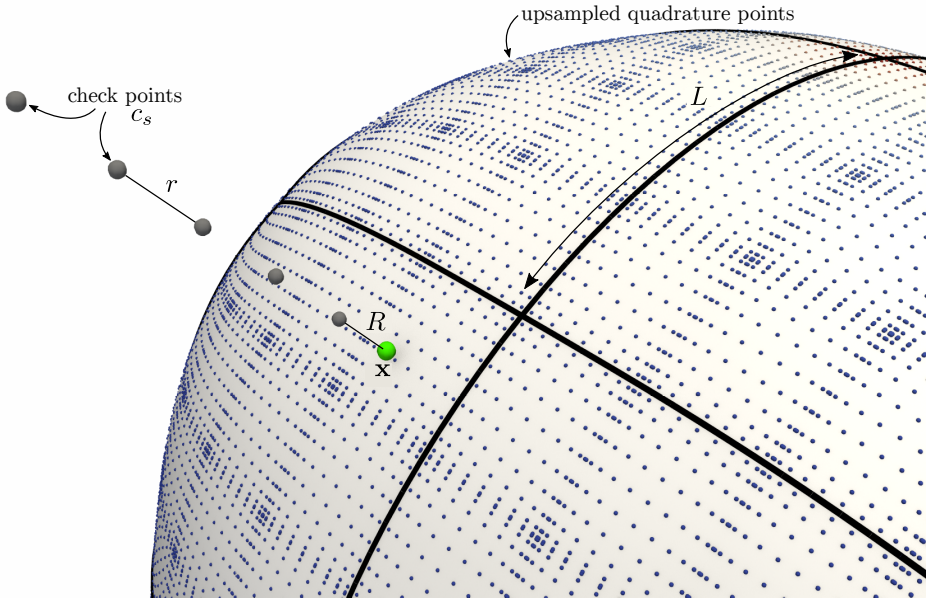
- Requires singular integration
- Only valid for Re << 1 (i.e. Stokes)

Challenges for Parallel RBC Simulation

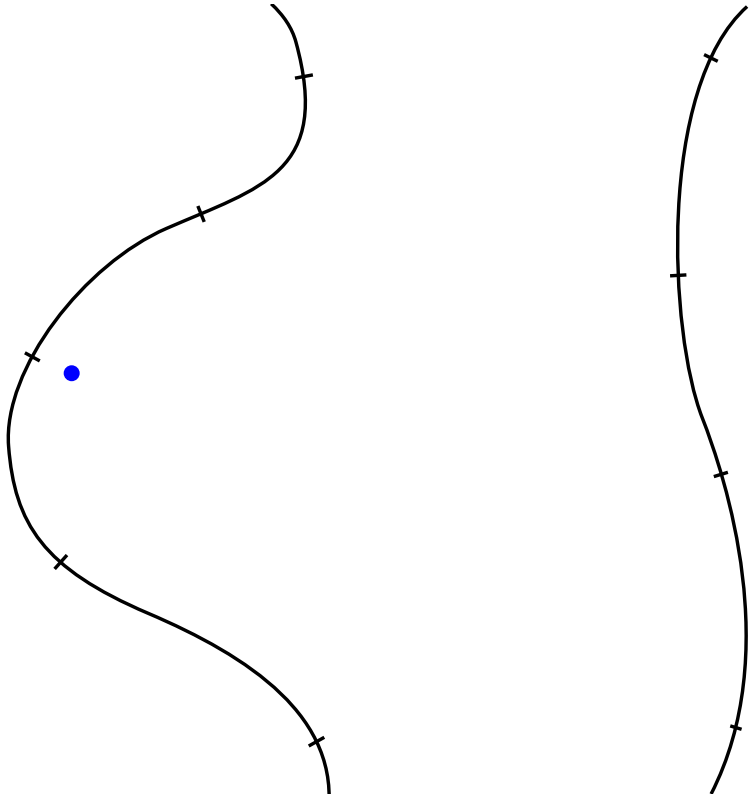
- Evaluate fluid velocity in parallel (FMM)
- Parallel singular quadrature
- Guarantee collision-free state across non-local geometry

Singular quadrature

- Upsample boundary data
- Find closest point
- Construct check points
- Evaluate velocity at check points
- Extrapolate velocity

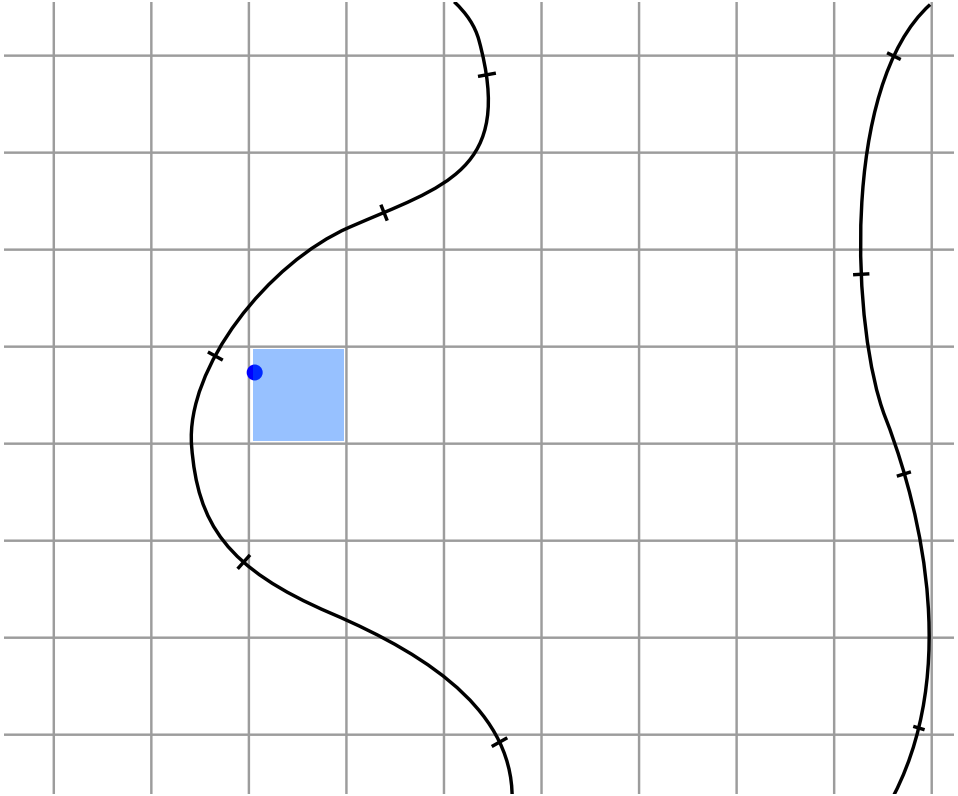


Closest point

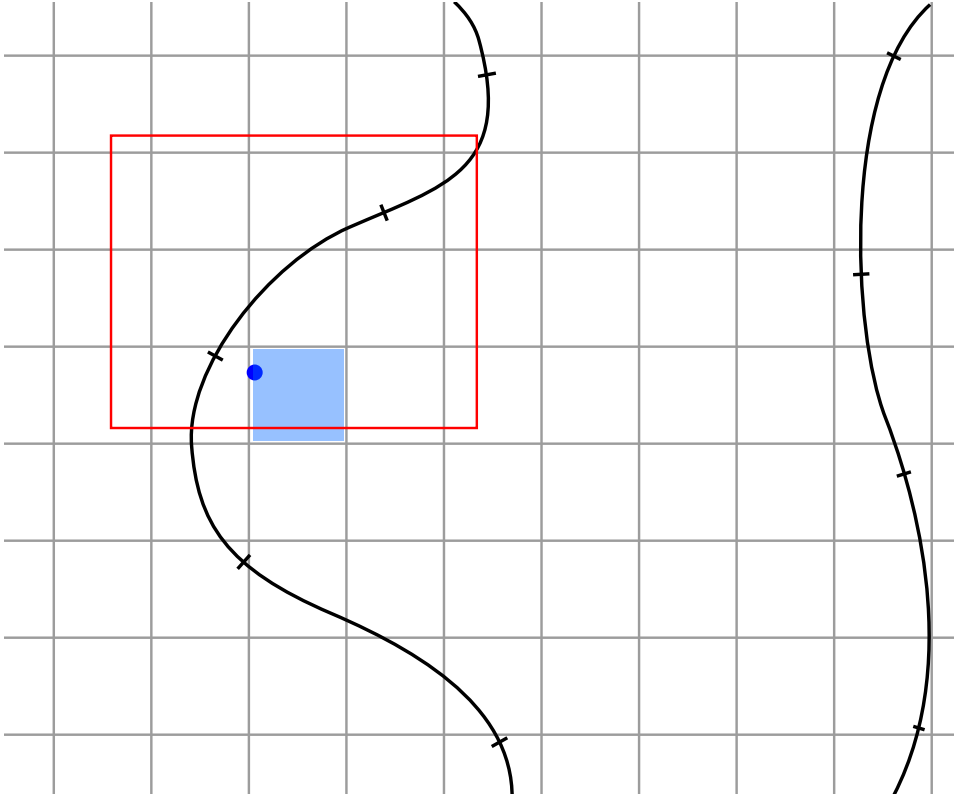


Parallel contact-aware simulations of deformable particles in 3D Stokes flow, Lu et. al., arxiv 2018

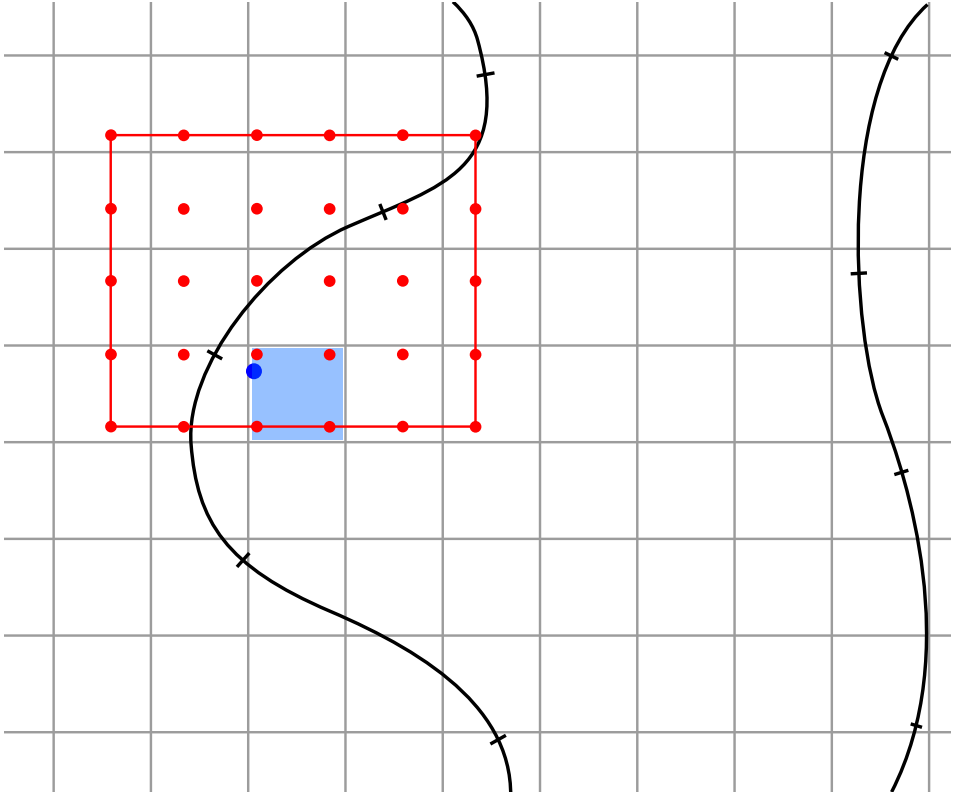
Step 1: Form spatial hash



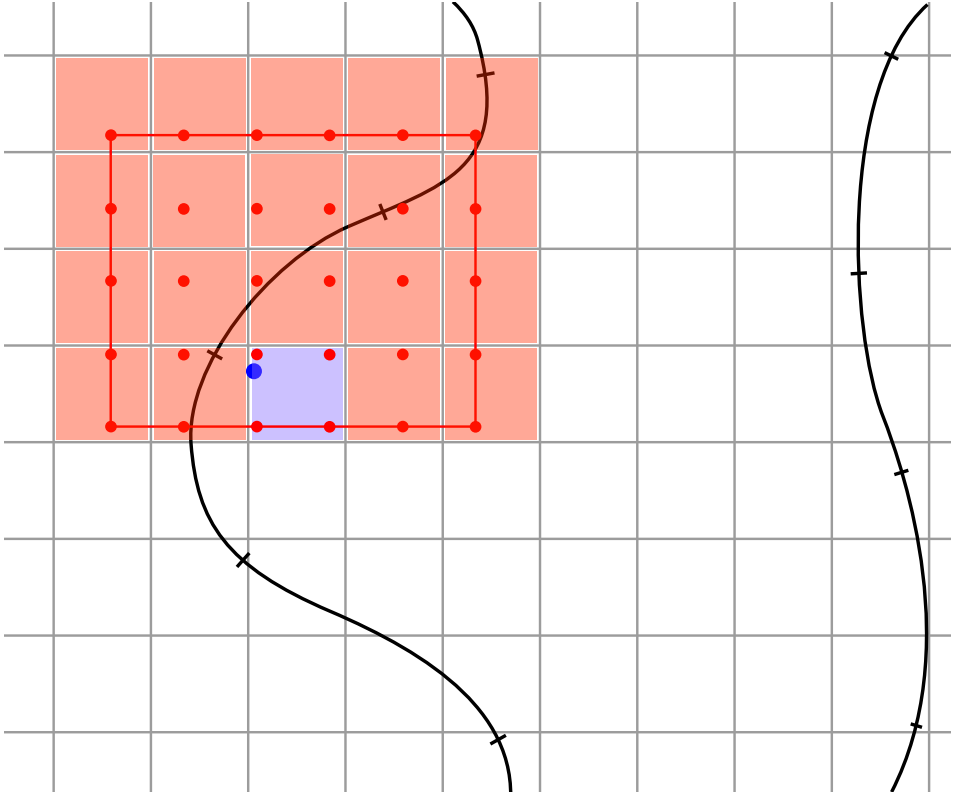
Step 2: Compute bounding box



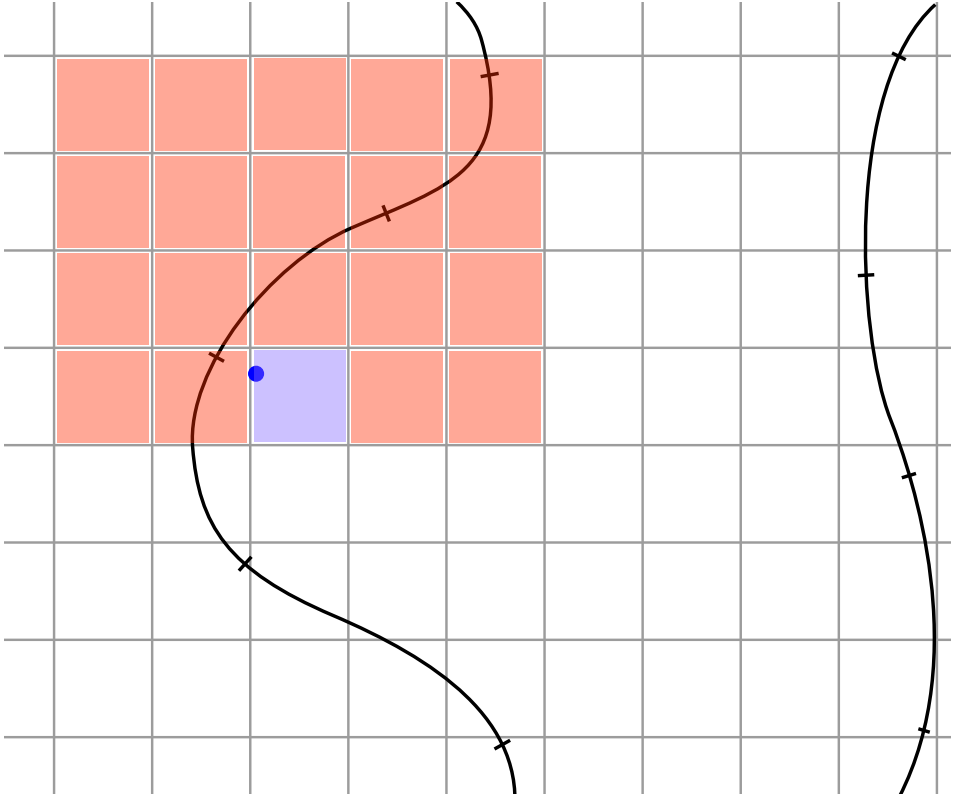
Step 3: Sample bounding box



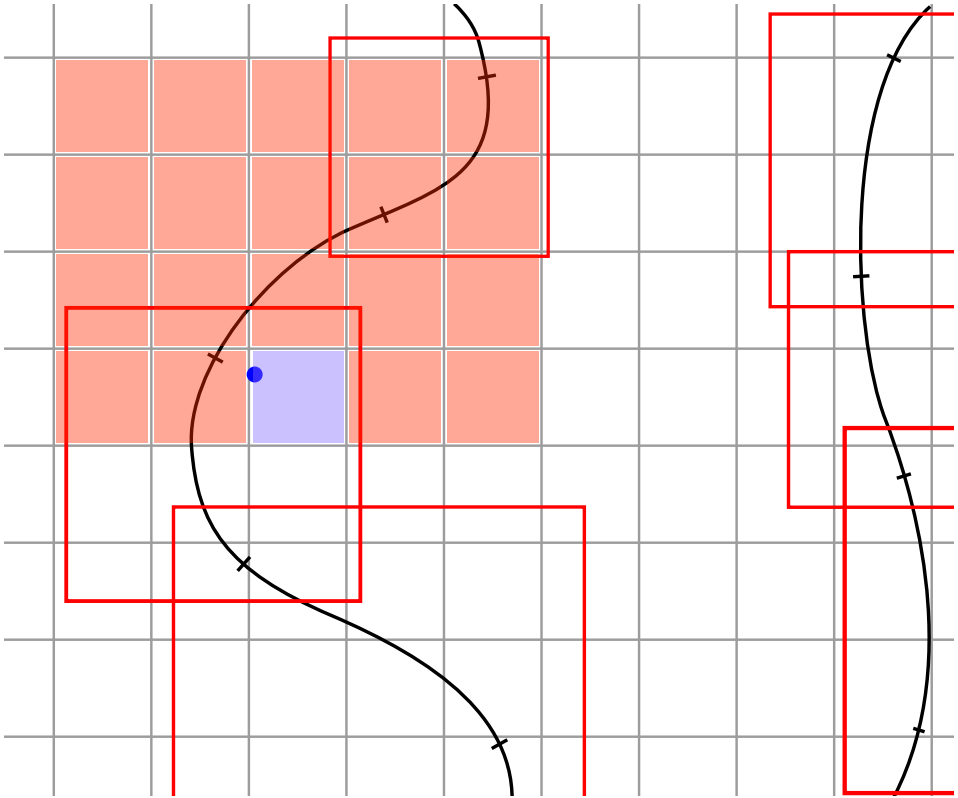
Step 4: Hash bounding box samples



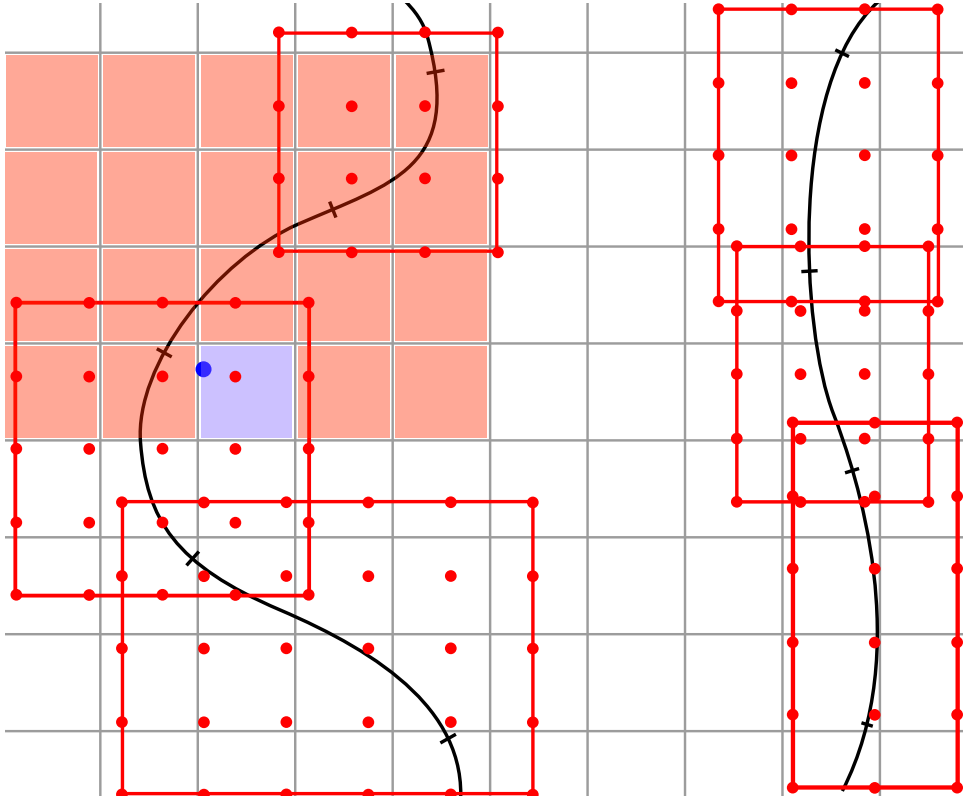
Step 5: Parallel sort + scatter



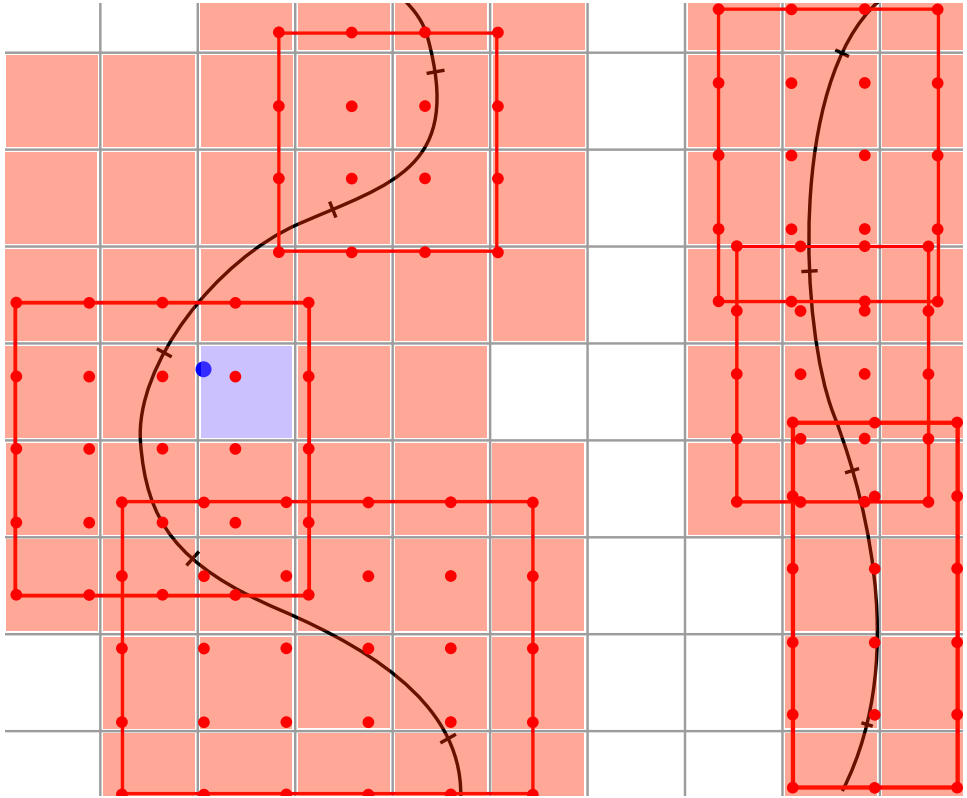
Repeat for all patches…



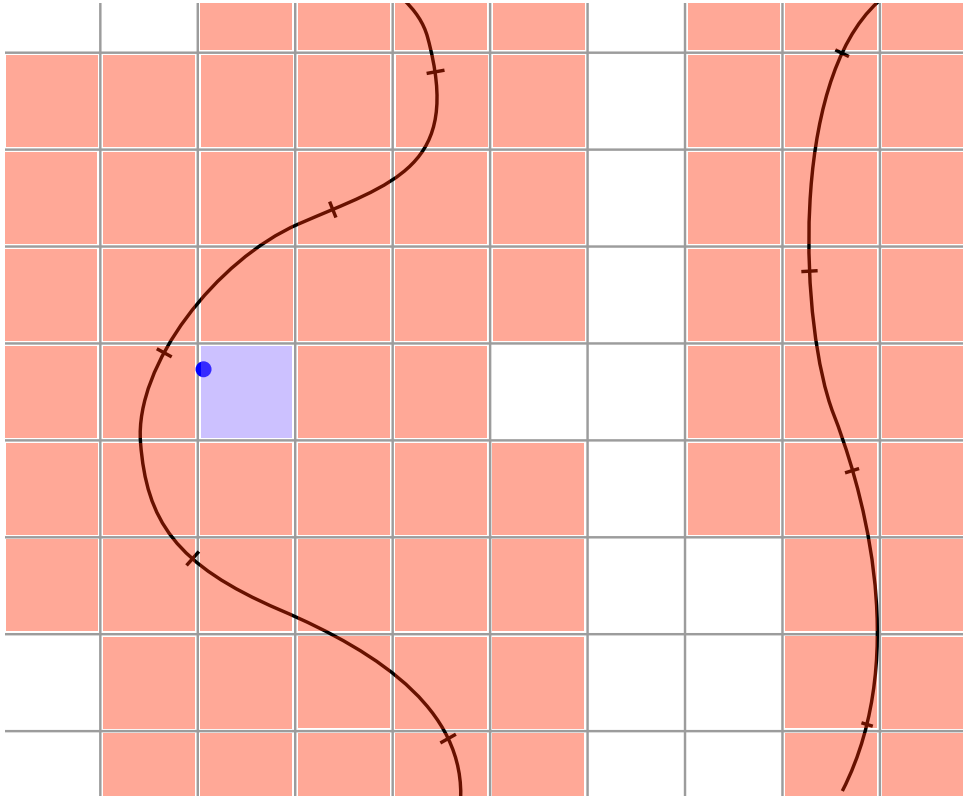
Repeat for all patches…



Repeat for all patches…

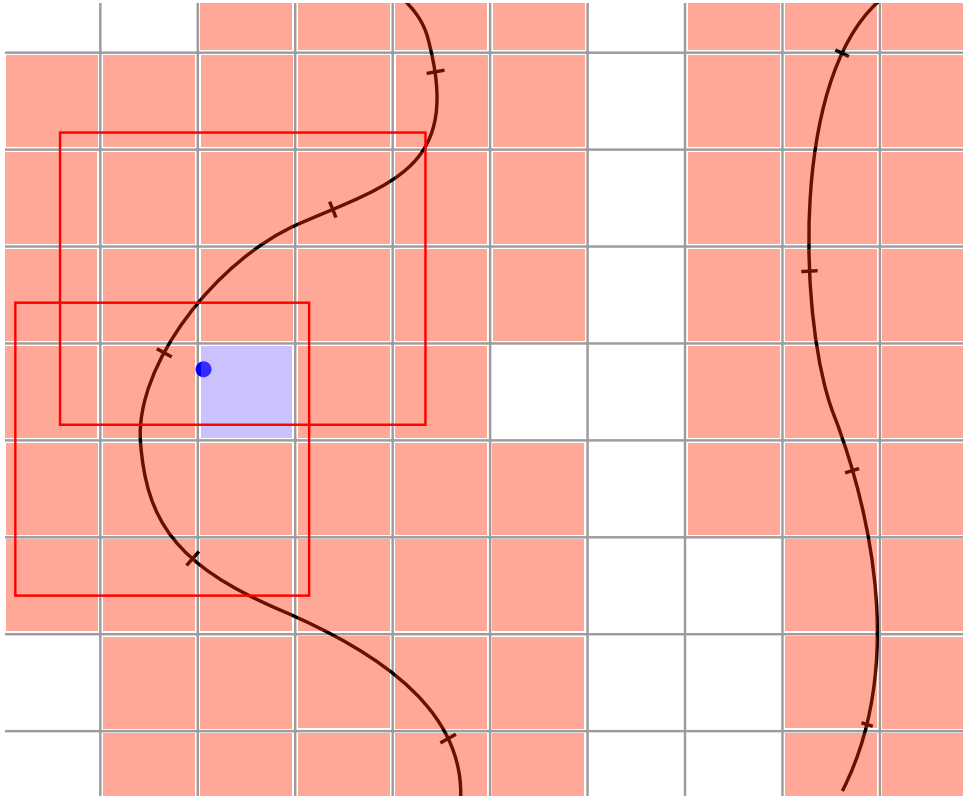


Repeat for all patches…

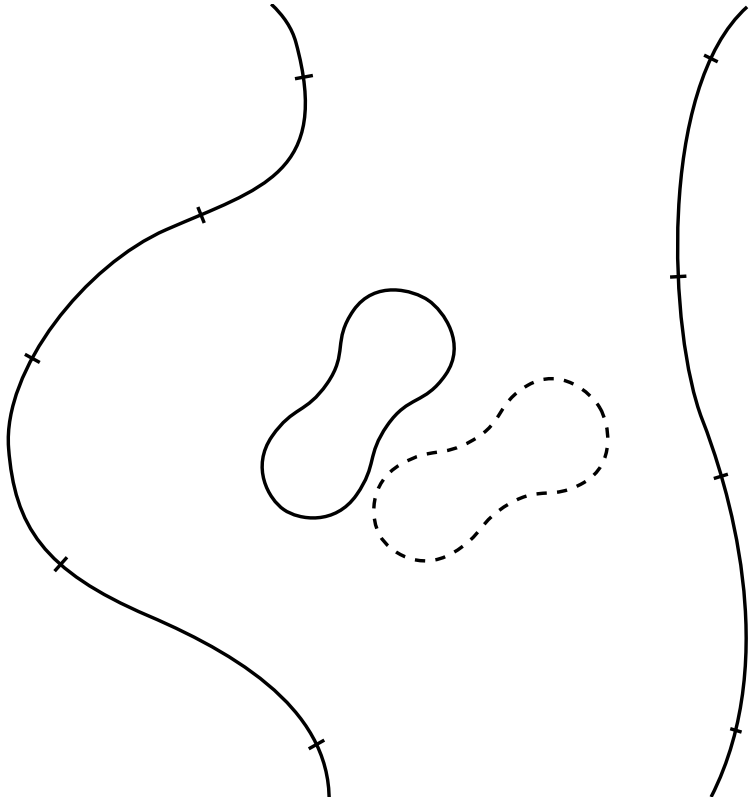


Hyksort: a new variant of hypercube quicksort on distributed memory architectures, H. Sundar et. al.; ICS 2016

Step 6: Compute distance locally

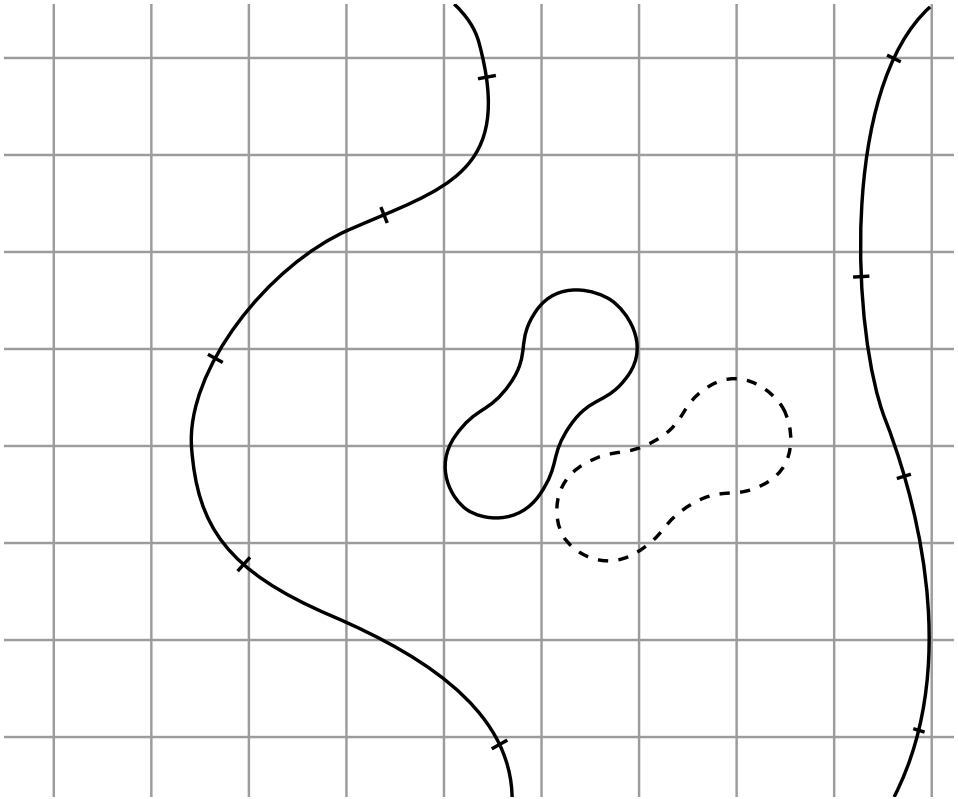


Collision handling

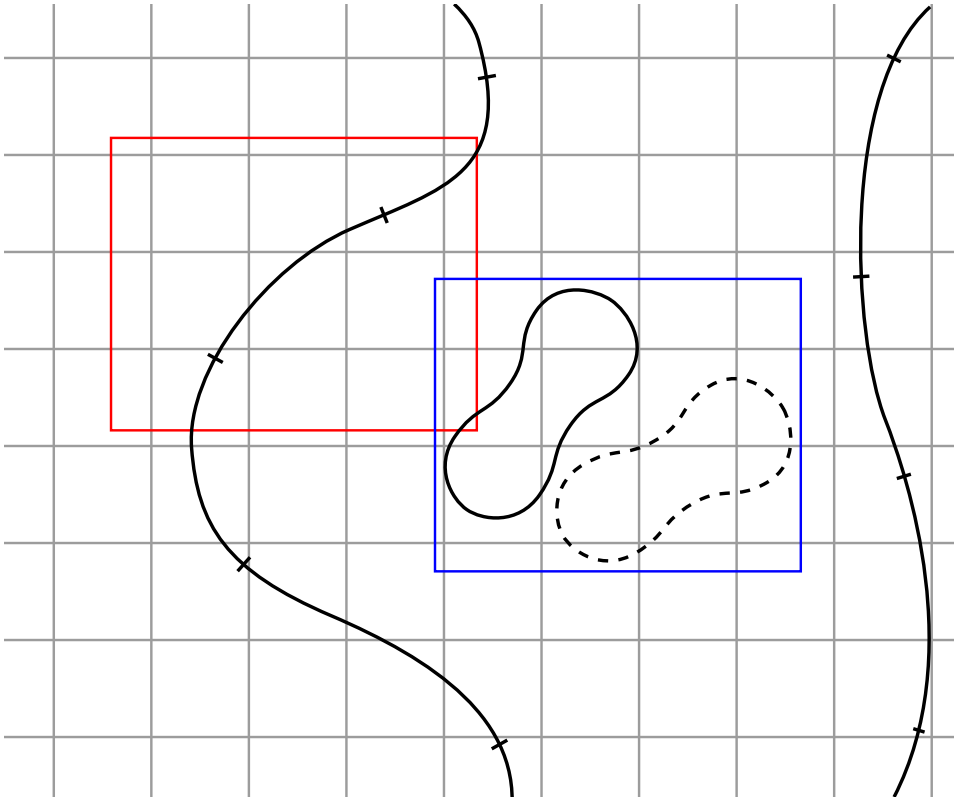


L. Lu et. al. Parallel contact-aware simulations of deformable particles in 3D Stokes flow, arxiv 2018

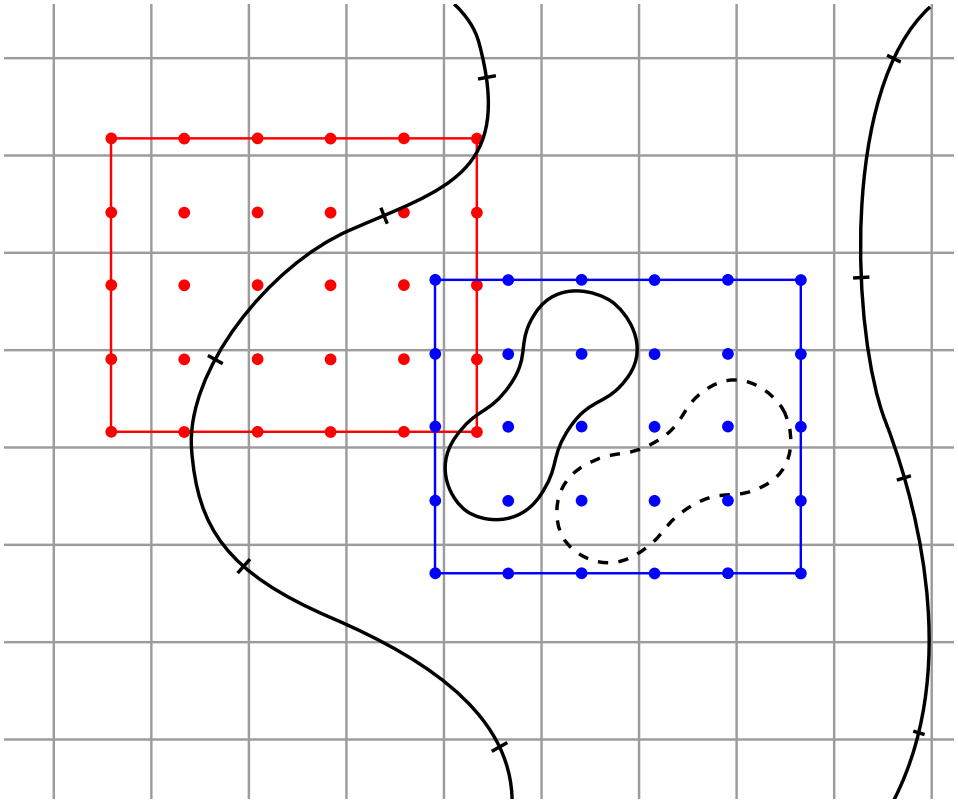
Step 1: Form a spatial hash



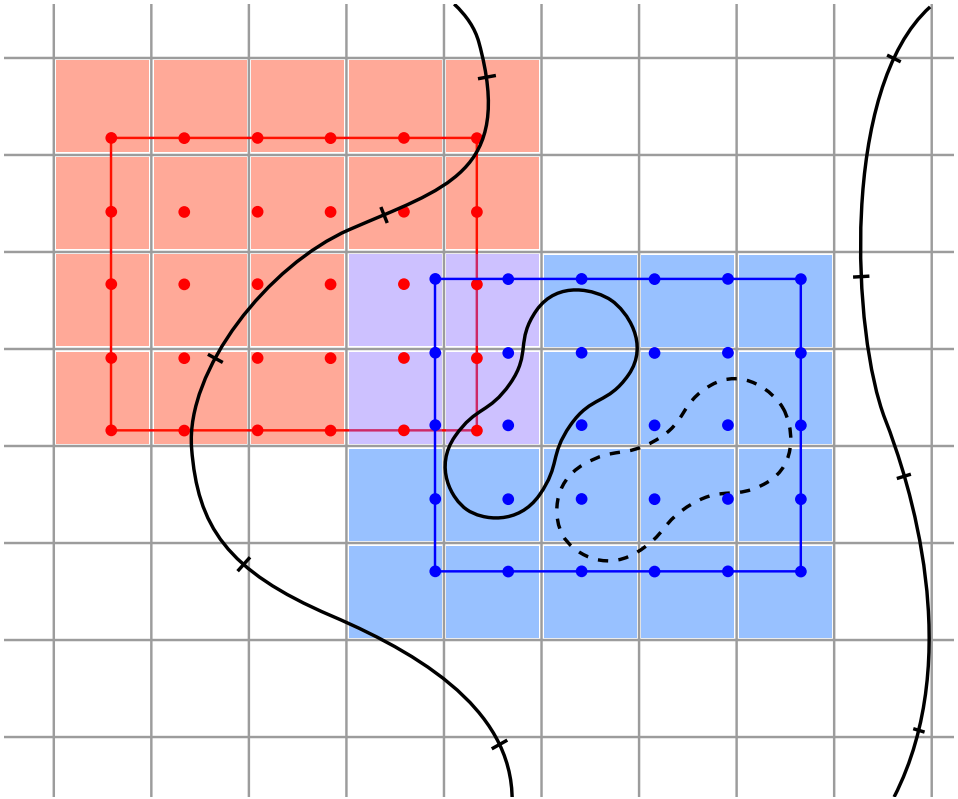
Step 2: Compute space-time bounding boxes



Step 3: Sample bounding boxes

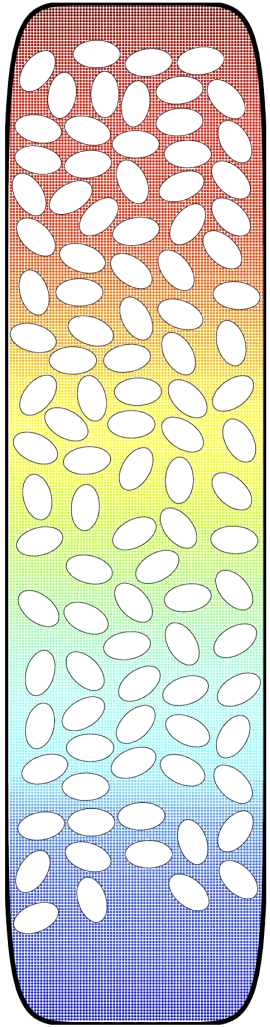


Step 4: Hash samples and parallel sort



Step 5: Iteratively solve NCP; repeat until collision-free

- Follow [Lu et. al. arxiv 2018]
- Solve NCP by a sequence of parallel linear complementarity problems (LCPs)



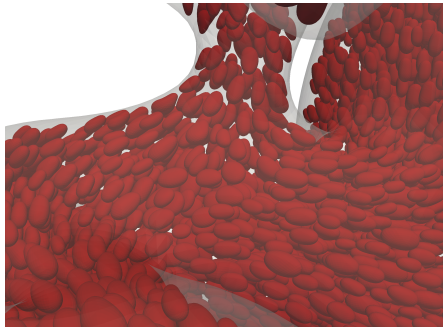
Outline

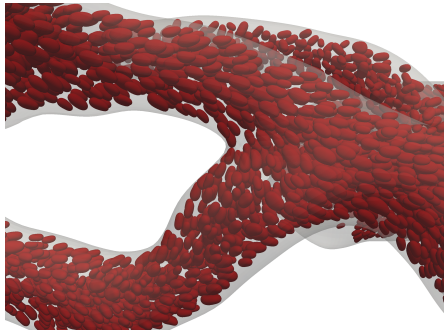
- Motivation
- Formulation, Numerics, Algorithms
- **Results**

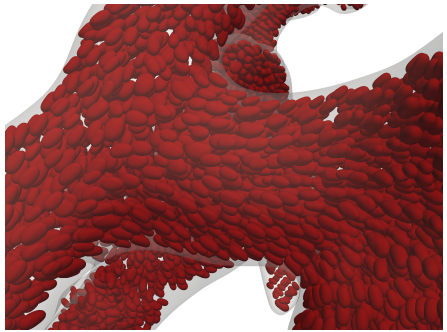
Setup

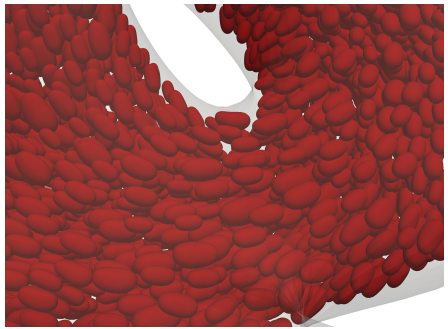
- Stampede2 at TACC
- Skylake (SKX): dual socket 24 core 2.1 GHz CPU, 192GB RAM
- Knights Landing (KNL): 68 core 1.4 GHz CPU, 96 GB RAM + 16 GB high-speed MCDRAM

Strong scaling



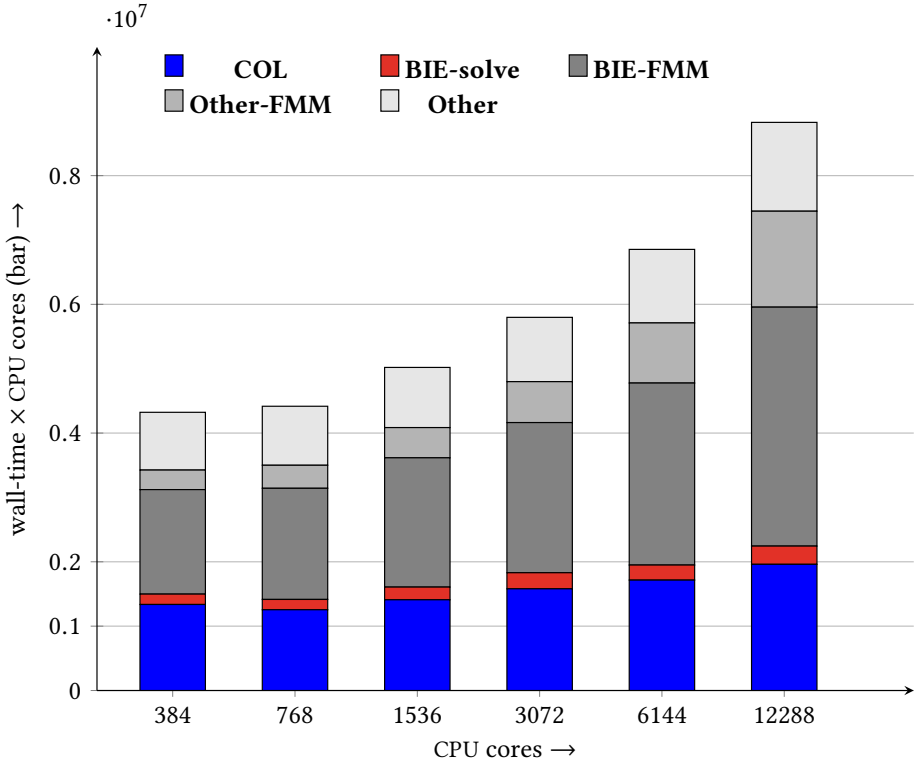




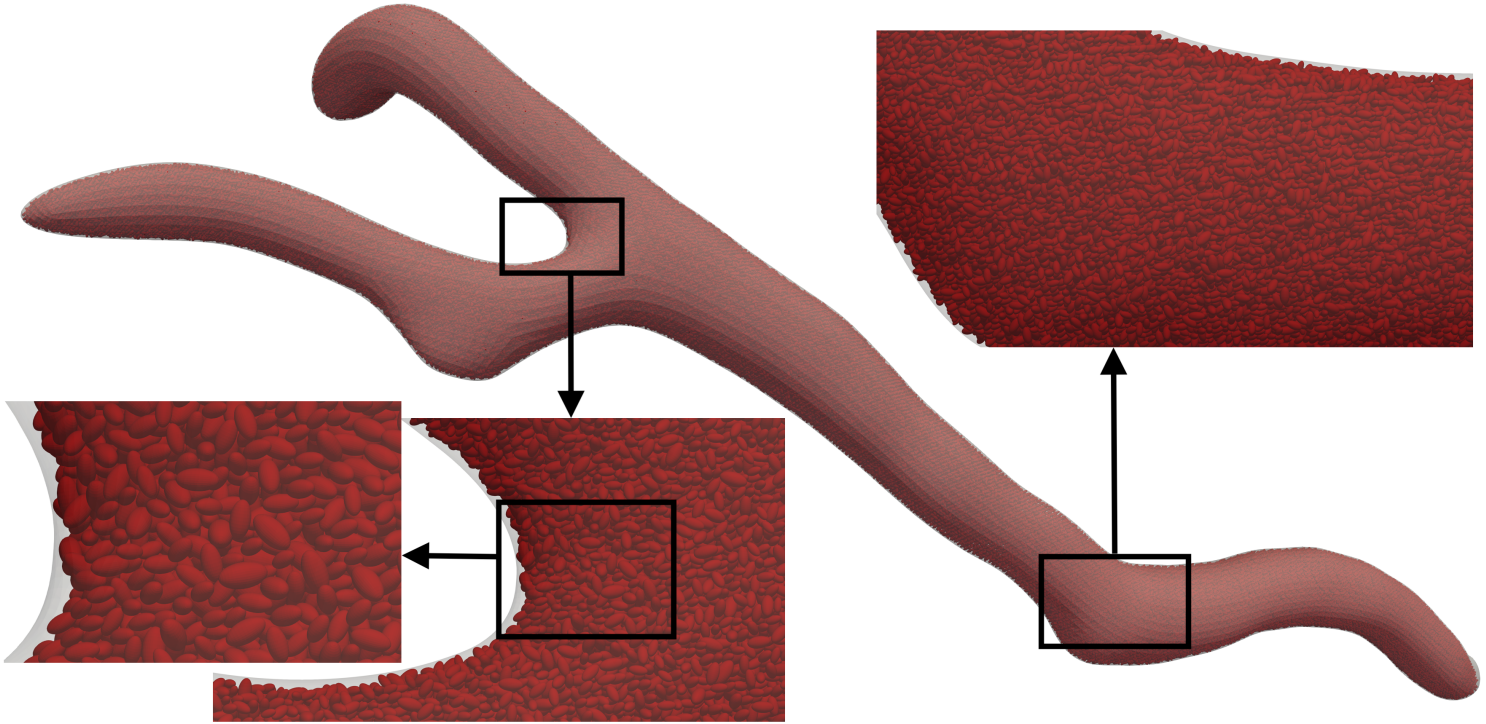


Strong scaling

- Skylake
- 40,960 RBCs with 40,960 patch blood vessel
- ~89 million DOF for RBCs
- ~15 million DOF for vessel
- 15.7x speed-up from 384 to 12288 cores
- 49% overall parallel efficiency
- 66% efficiency of collision handling + singular integration

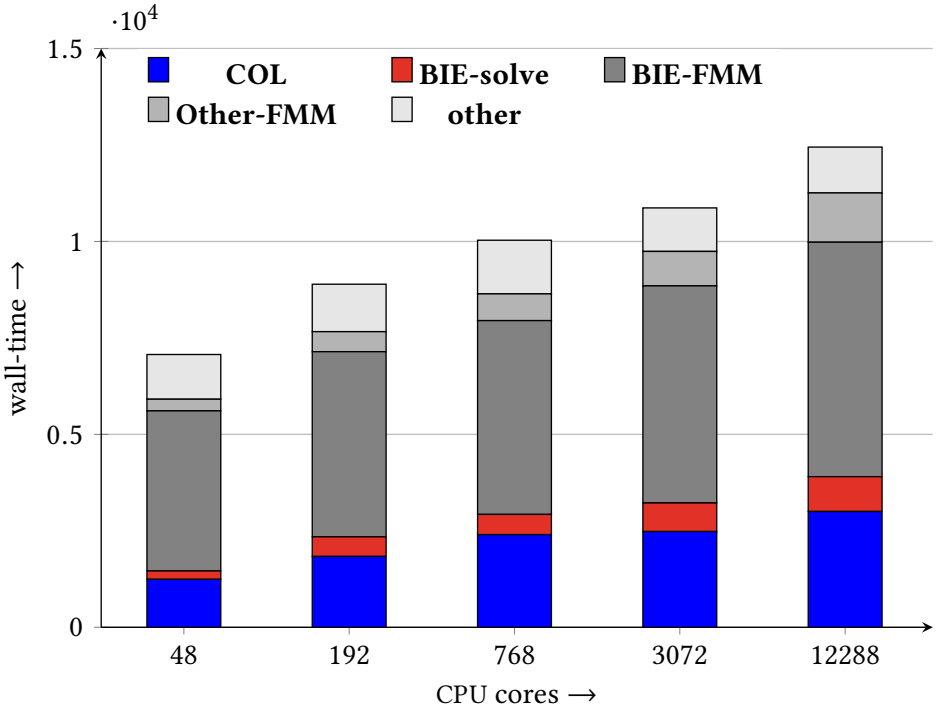


Weak scaling



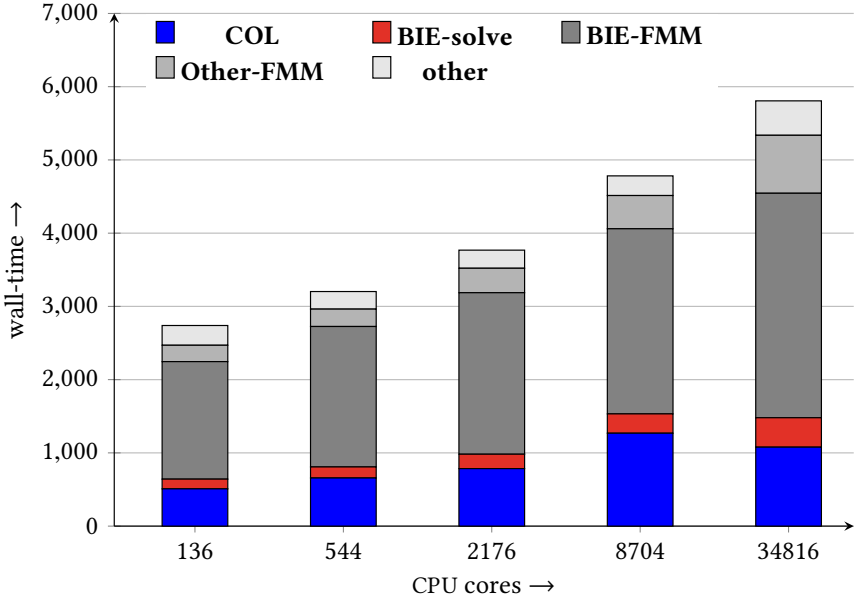
Weak Scaling: SKX

- Skylake
- 71% overall parallel efficiency
- 60% efficiency of collision handling + singular integration



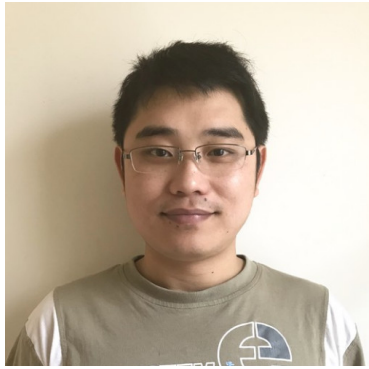
Weak Scaling: KNL

- Skylake
- 47% overall parallel efficiency
- 43% efficiency of collision handling + singular integration
- Largest simulation: 1 million RBCs and 2 million patches on vessel
- 3 billion total DOF
- Maintain collision-free state among 4.1 billion surface elements



Credits

Abtin Rahimian



Libin Lu

Denis Zorin

Dhairya Malhotra

Michael Shelley

Georg Stadler

Thanks!