Convergent Slender Body Theory

Code: https://github.com/dmalhotra/CSBQ

Dhairya Malhotra, Alex Barnett

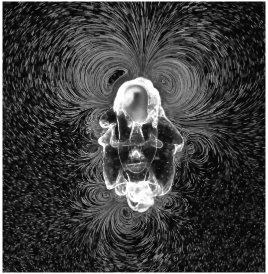
June 13, 2024

Slender Body Theory

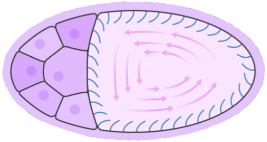
Stokes simulations with fibers are key to modeling complex fluids (suspensions, rheology, industrial, biomedical, cellular biophysics).

Slender Body Theory (SBT):

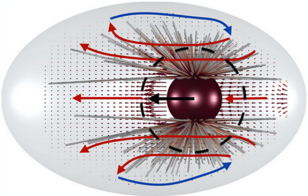
- Asymptotic expansion in radius (ε) as $\varepsilon \to 0$ (Keller-Rubinow '76).
- Doublet correction to make velocity theta-independent (Johnson '80).



Starfish larvae (Gilpin et al. 2016)



Drosophila oocyte (Stein et al. 2021)



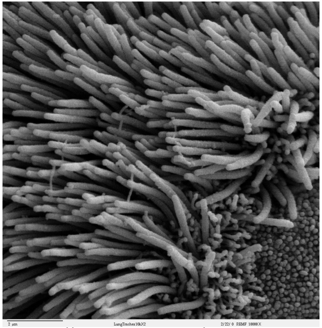
Mitotic spindle (Nazockdast et al. 2015)

Slender Body Theory Error Estimates

Error estimates: Rigorous analysis difficult (few very recent studies)

- classical asymptotics claims: $\varepsilon^2 \log(\varepsilon)$
- rigorous analysis: $\varepsilon \log^{3/2}(\varepsilon)$ (Mori-Ohm-Spirn '19)
- numerical tests: $\varepsilon^{1.7}$ (Mitchell et al. '21 -- verify close-touching breakdown)

  close-to-touching with gap of 10ε, only 2.5-digits in the infty-norm.



Source: http://remf.dartmouth.edu/imagesindex.html

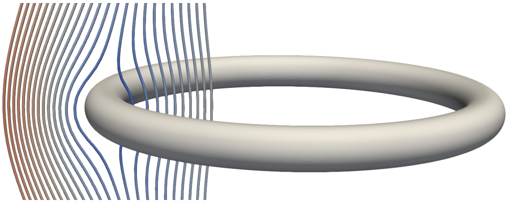
Slender Body Theory Error Estimates

Error estimates: Rigorous analysis difficult (few very recent studies)

- classical asymptotics claims: $\varepsilon^2 \log(\varepsilon)$
- rigorous analysis: $\varepsilon \log^{3/2}(\varepsilon)$ (Mori-Ohm-Spirn '19)
- numerical tests: $\varepsilon^{1.7}$ (Mitchell et al. '21 -- verify close-touching breakdown)

  close-to-touching with gap of 10ε, only 2.5-digits in the infty-norm.

ε	$\boldsymbol{u}_{exact}$	Rel-Error
1e-1	6.14921383598 56e-2	0.5e-2
1e-2	9.09845223245 84e-2	0.1e-3
1e-3	1.20156558899 04e-1	0.2e-5
1e-4	1.49319329075 87e-1	0.2e-7
1e-5	1.78481913130 97e-1	0.3e-9



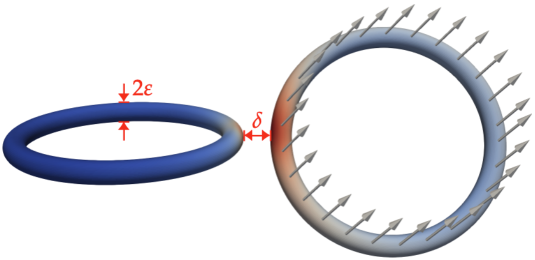
Slender Body Theory Error Estimates

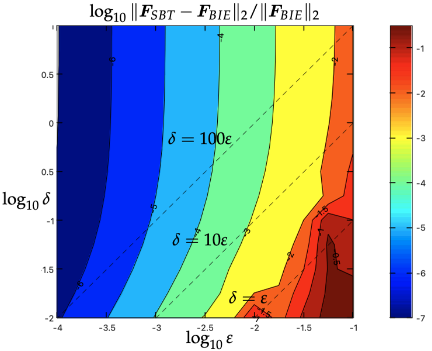
FLATIRON INSTITUTE

Error estimates: Rigorous analysis difficult (few very recent studies)

- classical asymptotics claims: $\varepsilon^2 \log(\varepsilon)$
- rigorous analysis: $\varepsilon \log^{3/2}(\varepsilon)$ (Mori-Ohm-Spirn '19)
- numerical tests: $\varepsilon^{1.7}$ (Mitchell et al. '21 -- verify close-touching breakdown)

close-to-touching with gap of 10ε, only 2.5-digits in the infty-norm.





Convergent Slender Body Theory

Goals: Develop boundary integral methods to solve the slender body BVP

- in a convergent way.
- adaptively when fibers get close.
- efficiently with effort independent of radius.

Convergent Slender Body Theory

Goals: Develop boundary integral methods to solve the slender body BVP

- in a convergent way.
- adaptively when fibers get close.
- efficiently with effort independent of radius.

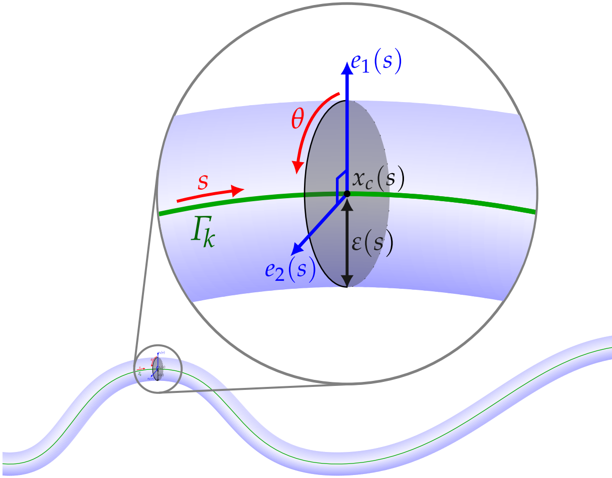
Focus on rigid fibers in this talk -- flexible fibers for future.

Related work: Mitchell et al, '21 (mixed-BVP corresponding to flexible fiber loop)

Discretization

Geometry description:

- parameterization s along fiber length
- coordinates $x_c(s)$ of centerline curve
- circular cross-section with radius $\varepsilon(s)$
- orientation vector $e_1(s)$



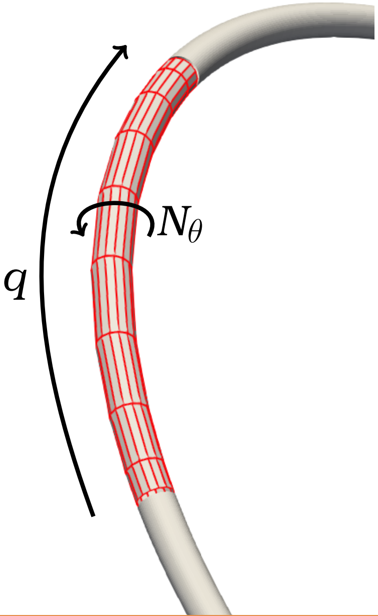
Discretization

Geometry description:

- parameterization s along fiber length
- coordinates $x_c(s)$ of centerline curve
- circular cross-section with radius $\varepsilon(s)$
- orientation vector $e_1(s)$

Discretization:

- piecewise Chebyshev (order q) discretization in s for $x_c(s)$, $\varepsilon(s)$, $e_1(s)$
- Collocation nodes: tensor product of Chebyshev and Fourier discretization in angle with order N_θ.



Boundary Quadratures

$$u(x) \ = \int_\Gamma \mathcal{K}(x-y)\,\sigma(y)\,da(y)$$

Boundary Quadratures

$$u(x) \; = \int_{\Gamma} \mathcal{K}(x-y)\, \sigma(y)\, da(y) \; = \sum_{k=1}^{N_{panel}} \int_{\Gamma_k} \mathcal{K}(x-y)\, \sigma(y)\, da(y)$$

Boundary Quadratures

$$u(x) = \int_{\Gamma} \mathcal{K}(x-y)\,\sigma(y)\,da(y) = \sum_{k=1}^{N_{panel}} \int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y)$$

$$= \underbrace{\sum_{x \notin \mathcal{N}(\Gamma_k)} \int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y)}_{\text{far-field}} + \underbrace{\sum_{x \in \mathcal{N}(\Gamma_k)} \int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y)}_{\text{near interactions}}$$

Boundary Quadratures

$$u(x) = \int_\Gamma \mathcal{K}(x-y)\,\sigma(y)\,da(y) = \sum_{k=1}^{N_{panel}} \int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y)$$
$$= \underbrace{\sum_{x\notin\mathcal{N}(\Gamma_k)} \int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y)}_{\text{far-field}} + \underbrace{\sum_{x\in\mathcal{N}(\Gamma_k)} \int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y)}_{\text{near interactions}}$$

Far-field

- Tensor product quad:
  Gauss-Legendre $\times$ PTR
- Accelerate with PVFMM
  $\mathcal{O}(N^2) \to \mathcal{O}(N)$

Near interactions

- Build special quadrature rules!

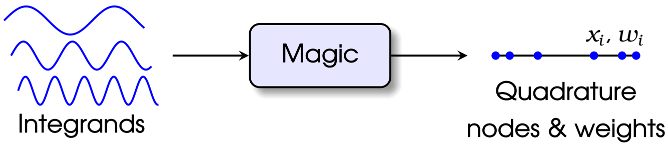
Generating Special Quadrature Rules

Integrands

Quadrature
nodes & weights

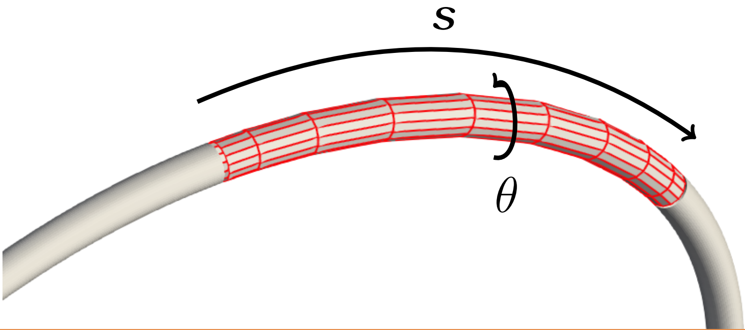
Generalized Gaussian Quad.
Bremer, Gimbutas and
Rokhlin - SISC 2010

Generating Special Quadrature Rules

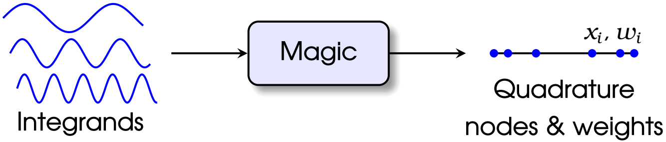


Generalized Gaussian Quad.
Bremer, Gimbutas and
Rokhlin - SISC 2010

$$\int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y) = \int_s \int_\theta \mathcal{K}(x-y(s,\theta))\,\sigma(s,\theta)\,J(s,\theta)\,d\theta\,ds$$

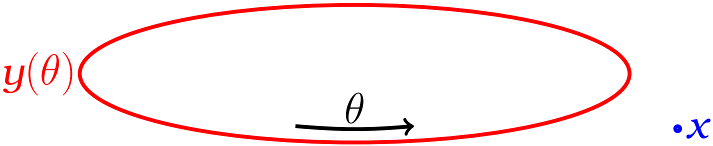


Generating Special Quadrature Rules

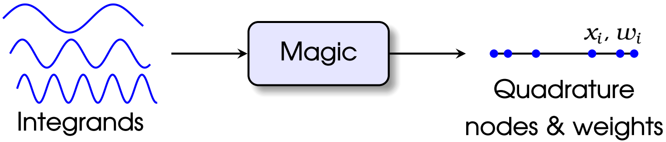


Generalized Gaussian Quad.
Bremer, Gimbutas and
Rokhlin - SISC 2010

$$\int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y) \;=\; \int_s \int_\theta \mathcal{K}(x-y(s,\theta))\,\sigma(s,\theta)\,J(s,\theta)\,d\theta\,ds$$

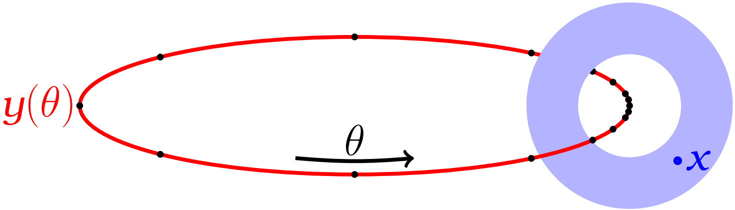


Generating Special Quadrature Rules



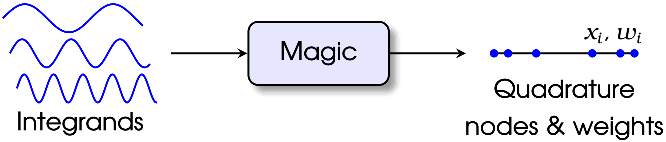
Generalized Gaussian Quad.
Bremer, Gimbutas and
Rokhlin - SISC 2010

$$\int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y) \;=\; \int_s \int_\theta \mathcal{K}(x-y(s,\theta))\,\sigma(s,\theta)\,J(s,\theta)\,d\theta\,ds$$



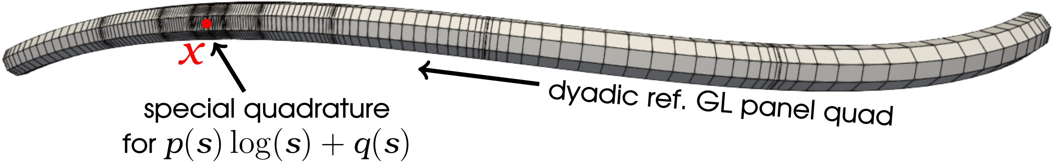
~ 48 quadrature nodes for $n_0 = 8$ and 10-digits accuracy.
$\sim 26M$ modal Green's function evaluations/sec/core (Skylake 2.4GHz)

Generating Special Quadrature Rules

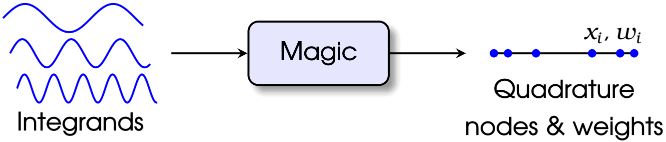


Generalized Gaussian Quad.
Bremer, Gimbutas and Rokhlin - SISC 2010

$$\int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y) = \int_s \int_\theta \mathcal{K}(x-y(s,\theta))\,\sigma(s,\theta)\,J(s,\theta)\,d\theta\,ds$$

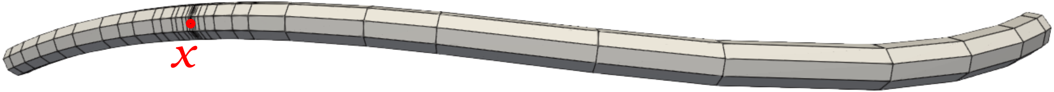


Generating Special Quadrature Rules



Generalized Gaussian Quad.
Bremer, Gimbutas and Rokhlin - SISC 2010

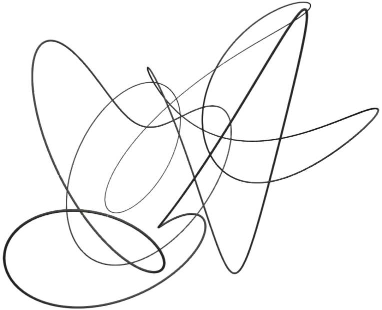
$$\int_{\Gamma_k} \mathcal{K}(x-y)\,\sigma(y)\,da(y) = \int_s \int_\theta \mathcal{K}(x-y(s,\theta))\,\sigma(s,\theta)\,J(s,\theta)\,d\theta\,ds$$

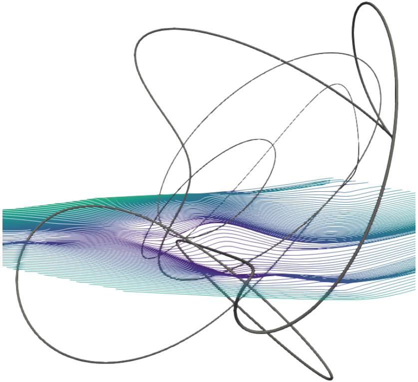


Instead build special quadrature rules!

- replace composite panel quadratures with a single quadrature.
- Separate rules for different aspect ratios (1 -- 10^4 in powers of 2)

Numerical Results - Stokes BVP





Exterior Stokes Dirichlet BVP:

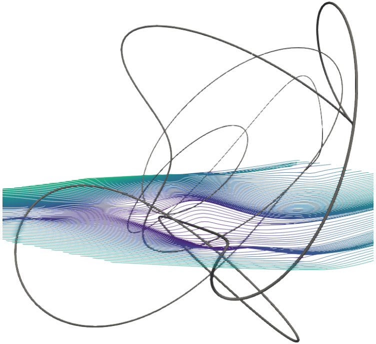
$$\Delta \boldsymbol{u} - \nabla p = 0, \qquad \boldsymbol{u}|_\Gamma = \boldsymbol{u}_0,$$
$$\nabla \cdot \boldsymbol{u} = 0, \qquad u(x) \to 0 \text{ as } |x| \to 0,$$

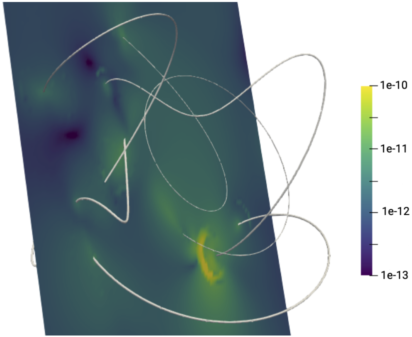
wire radius = 1.5e-3 to 4e-3

wire length = 16

BIE formulation: $(I/2 + D + S \,/\, (2\varepsilon \log \varepsilon^{-1}))\, \boldsymbol{\sigma} = \boldsymbol{u}_0$

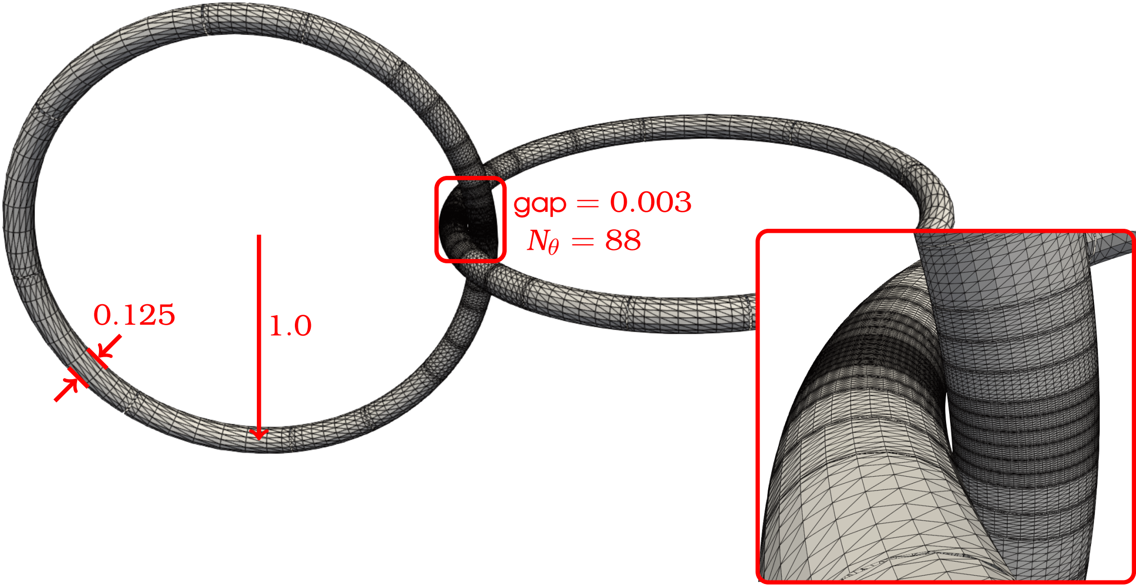
Numerical Results - Stokes BVP



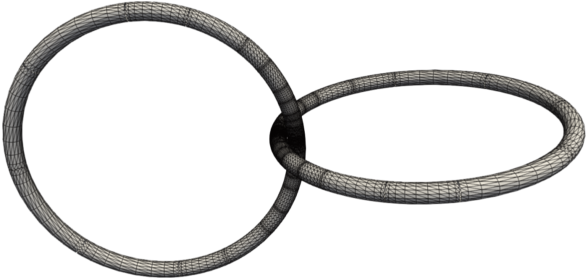


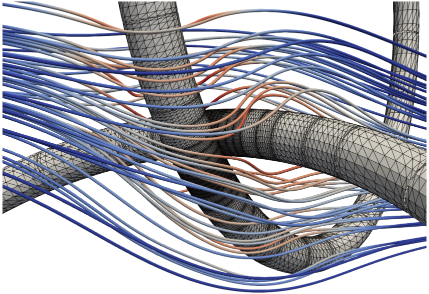
						1-core			40-cores	
N	N_{panel}	N_θ	ϵ_{GMRES}	N_{iter}	$\lVert e \rVert_\infty$	T_{setup}	(N/T_{setup})	T_{solve}	T_{setup}	T_{solve}
1.0e4	49	8	1e-02	5	3.5e-02	0.193	(5.4e4)	0.130	0.042	0.017
2.6e4	103	12	1e-05	22	5.5e-05	0.572	(4.5e4)	4.039	0.045	0.215
4.7e4	157	20	1e-07	33	6.6e-07	1.416	(3.3e4)	19.518	0.134	1.162
8.3e4	227	24	1e-08	38	4.5e-08	3.623	(2.3e4)	78.907	0.324	3.689
2.2e5	457	40	1e-10	49	2.9e-10	21.949	(1.0e4)	746.966	4.458	48.494

Numerical Results - close-to-touching



Numerical Results - close-to-touching





				1-core			40-cores	
N	ϵ_{GMRES}	N_{iter}	$\|e\|_\infty$	T_{setup}	(N/T_{setup})	T_{solve}	T_{setup}	T_{solve}
6.5e4	1e-02	4	2.1e-02	8.1	(8.0e+3)	6.5	1.28	1.4
6.5e4	1e-05	24	2.4e-03	16.8	(3.8e+3)	42.9	2.50	7.7
6.5e4	1e-07	43	2.8e-06	23.5	(2.7e+3)	81.6	3.31	12.8
6.5e4	1e-10	59	5.4e-08	35.6	(1.8e+3)	122.9	4.06	19.2
6.5e4	1e-13	72	1.3e-10	49.9	(1.3e+3)	162.6	5.27	23.2

Mobility problem

- n rigid bodies $\Omega = \sum_{i=1}^{n} \Omega_i$

  with velocities $\boldsymbol{V}(\boldsymbol{x}) = \boldsymbol{v}_i + \boldsymbol{\omega}_i \times (\boldsymbol{x} - \boldsymbol{x}_i^c)$,

  and given forces $\boldsymbol{F}_i$, torques $\boldsymbol{T}_i$ abount $\boldsymbol{x}_i^c$.

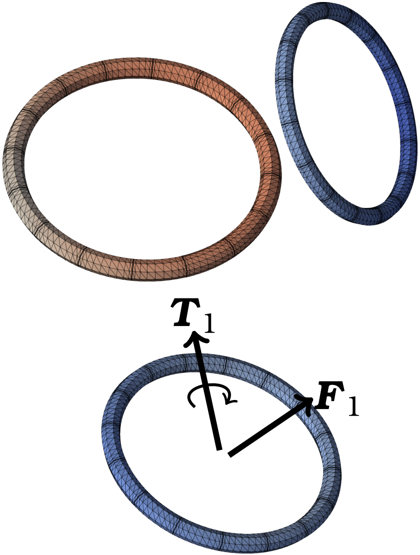
- Stokesian fluid in $\mathbb{R}^3 \setminus \Omega$

$$\Delta \boldsymbol{u} - \nabla p = 0, \quad \nabla \cdot \boldsymbol{u} = 0,$$

$$\boldsymbol{u} \to 0 \text{ as } \boldsymbol{x} \to \infty.$$

- Boundary conditions on $\partial\Omega$,

$$\boldsymbol{u} = \boldsymbol{V} + \boldsymbol{u}_s.$$



Mobility problem

- n rigid bodies $\quad \Omega = \sum_{i=1}^{n} \Omega_i$

  with velocities $\boldsymbol{V}(\boldsymbol{x}) = \boldsymbol{v}_i + \boldsymbol{\omega}_i \times (\boldsymbol{x} - \boldsymbol{x}_i^c)$,

  and given forces $\boldsymbol{F}_i$, torques $\boldsymbol{T}_i$ abount $\boldsymbol{x}_i^c$.

- Stokesian fluid in $\mathbb{R}^3 \setminus \Omega$

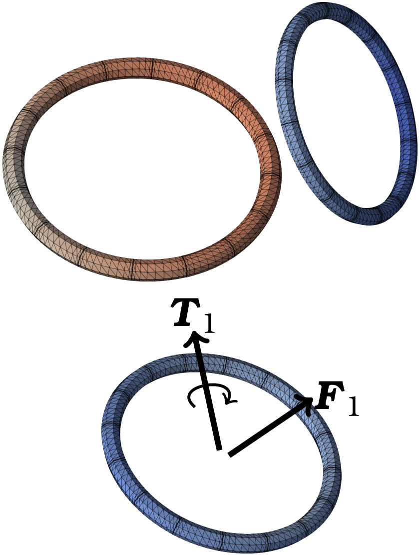
$$\Delta \boldsymbol{u} - \nabla p = 0, \quad \nabla \cdot \boldsymbol{u} = 0,$$

$$\boldsymbol{u} \to 0 \ \text{ as } \ \boldsymbol{x} \to \infty.$$

- Boundary conditions on $\partial\Omega$,

$$\boldsymbol{u} = \boldsymbol{V} + \boldsymbol{u}_s.$$

unknown: $\boldsymbol{V}(\boldsymbol{u}_i, \boldsymbol{\omega}_i)$



Mobility problem - double-layer formulation

Represent fluid velocity: $\boldsymbol{u} = \mathcal{S}[\boldsymbol{\nu}(\boldsymbol{F}_i, \boldsymbol{T}_i)] + \mathcal{D}[\boldsymbol{\sigma}]$

and rigid body velocity: $\boldsymbol{V} = -\sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}$

Applying boundary conditions $(\boldsymbol{u} = \boldsymbol{V} + \boldsymbol{u}_s \text{ on } \partial\Omega)$,

$$(I/2 + D)\,\boldsymbol{\sigma} + \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma} = \boldsymbol{u}_s - S\,\boldsymbol{\nu}$$

(Pozrikidis - Boundary Integral and Singularity Methods for Linearized Viscous Flow)

Mobility problem - double-layer formulation

Represent fluid velocity: $\boldsymbol{u} = \mathcal{S}[\boldsymbol{\nu}(\boldsymbol{F}_i, \boldsymbol{T}_i)] + \mathcal{D}[\boldsymbol{\sigma}]$

and rigid body velocity: $\boldsymbol{V} = -\sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}$

Applying boundary conditions $(\boldsymbol{u} = \boldsymbol{V} + \boldsymbol{u}_s$ on $\partial\Omega)$,

$$(I/2 + D)\,\boldsymbol{\sigma} + \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma} = \boldsymbol{u}_s - S\,\boldsymbol{\nu}$$

(Pozrikidis - Boundary Integral and Singularity Methods for Linearized Viscous Flow)

Second kind integral equation $\cdots$ but doesn't work for slender bodies!

Mobility problem - double-layer formulation

Represent fluid velocity: $\boldsymbol{u} = \mathcal{S}[\boldsymbol{\nu}(\boldsymbol{F}_i, \boldsymbol{T}_i)] + \mathcal{D}[\boldsymbol{\sigma}]$

and rigid body velocity: $\boldsymbol{V} = -\sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}$

Applying boundary conditions $(\boldsymbol{u} = \boldsymbol{V} + \boldsymbol{u}_s$ on $\partial\Omega)$,

$$(I/2 + D)\,\boldsymbol{\sigma} + \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma} = \boldsymbol{u}_s - S\,\boldsymbol{\nu}$$

(Pozrikidis - Boundary Integral and Singularity Methods for Linearized Viscous Flow)

Second kind integral equation $\cdots$ but doesn't work for slender bodies!

$\kappa(I/2 + D) \sim 1/(\varepsilon^2 \log \varepsilon^{-1})$

Mobility problem - combined field formulation

Represent fluid velocity: $\boldsymbol{u} = \mathcal{S}[\boldsymbol{\nu}(\boldsymbol{F}_i, \boldsymbol{T}_i)] + \mathcal{K}[\boldsymbol{\sigma}]$

and rigid body velocity: $\boldsymbol{V} = -\sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}$

where, $\mathcal{K} = \mathcal{D} + \mathcal{S}/(2\varepsilon \log \varepsilon^{-1})$.

Mobility problem - combined field formulation

Represent fluid velocity: $\boldsymbol{u} = \mathcal{S}[\boldsymbol{\nu}(\boldsymbol{F}_i, \boldsymbol{T}_i)] + \mathcal{K}[\boldsymbol{\sigma} - \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}]$

and rigid body velocity: $\boldsymbol{V} = -\sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}$

where, $\mathcal{K} = \mathcal{D} + \mathcal{S}/(2\varepsilon \log \varepsilon^{-1})$.

Mobility problem - combined field formulation

Represent fluid velocity: $\boldsymbol{u} = \mathcal{S}[\boldsymbol{\nu}(\boldsymbol{F}_i, \boldsymbol{T}_i)] + \mathcal{K}[\boldsymbol{\sigma} - \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}]$

and rigid body velocity: $\boldsymbol{V} = - \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}$

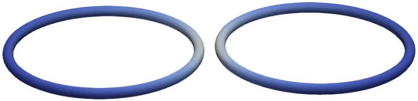
where, $\mathcal{K} = \mathcal{D} + \mathcal{S}/(2\varepsilon \log \varepsilon^{-1})$.

Applying boundary conditions,

$$(\mathcal{I}/2 + \mathcal{K})[\boldsymbol{\sigma} - \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma}] + \sum_{i=1}^{6n} \mathfrak{v}_i \mathfrak{v}_i^T \boldsymbol{\sigma} = \boldsymbol{u}_s - \mathcal{S}[\boldsymbol{\nu}]$$

Second kind integral equation and well-conditioned!

Numerical Results - Sedimentation Flow

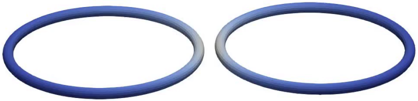


Time-stepping: 5-th order adaptive SDC

8-digits accuracy in quadratures, GMRES solve, and time-stepping.

40 CPU cores

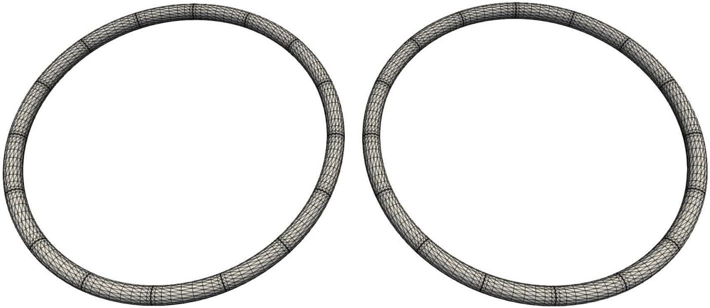
Numerical Results - Sedimentation Flow



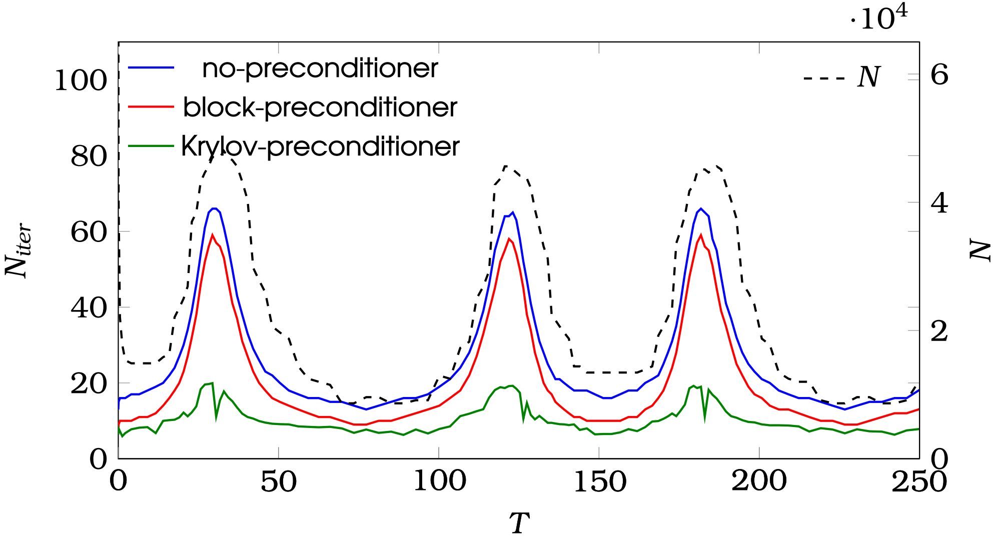
Time-stepping: 5-th order adaptive SDC

8-digits accuracy in quadratures, GMRES solve, and time-stepping.

40 CPU cores



Numerical Results - Sedimentation Flow



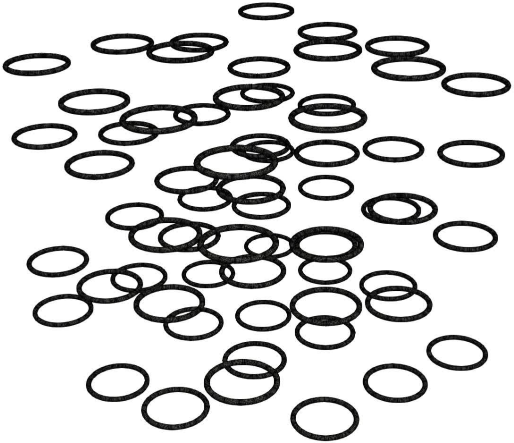
Numerical Results - Sedimentation Flow

5-th order adaptive SDC

8-digits accuracy in quadratures, GMRES solve, and time-stepping.

0.5 million unknowns
64 rings.

160 CPU cores



CSBQ library

Code: https://github.com/dmalhotra/CSBQ

(remember to checkout submodule SCTL)

Requirements: C++11 compiler with OpenMP

Build system: none (header only)

```
#include <csbq.hpp>
```

CSBQ library

Code: https://github.com/dmalhotra/CSBQ
(remember to checkout submodule SCTL)

Requirements: C++11 compiler with OpenMP

Build system: none (header only)

```
#include <csbq.hpp>
```

Optional dependencies: BLAS, LAPACK, FFTW, MPI, and PVFMM

Library overview

SCTL: Scientific Computing Template Library

BoundaryIntegralOp **Kernel** **FMM** **Quadrature**

GMRES **SDC** **Profile** **Vec (SIMD)** **Comm**

Vector **Matrix** **VTUData** **Interpolation** **Tree**

Library overview

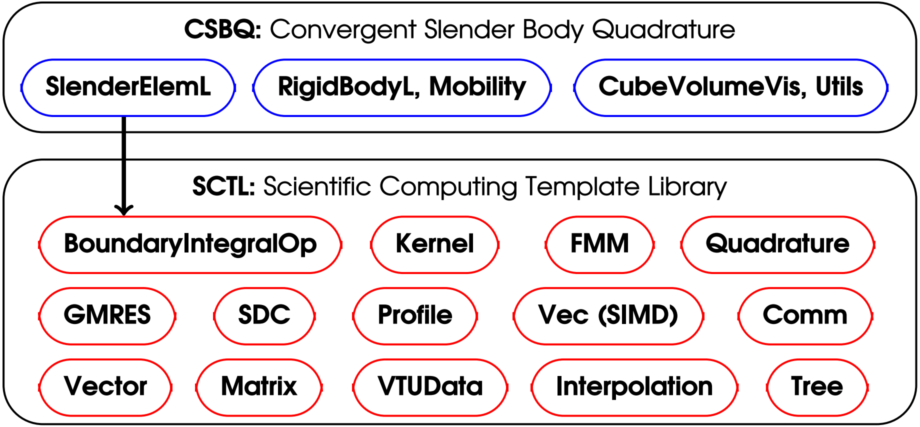
CSBQ: Convergent Slender Body Quadrature

- SlenderElemL
- RigidBodyL, Mobility
- CubeVolumeVis, Utils

SCTL: Scientific Computing Template Library

- BoundaryIntegralOp
- Kernel
- FMM
- Quadrature
- GMRES
- SDC
- Profile
- Vec (SIMD)
- Comm
- Vector
- Matrix
- VTUData
- Interpolation
- Tree

Library overview



Library overview

Applications

CSBQ: Convergent Slender Body Quadrature

- **SlenderElemL**
- **RigidBodyL, Mobility**
- **CubeVolumeVis, Utils**

SCTL: Scientific Computing Template Library

- **BoundaryIntegralOp**
- **Kernel**
- **FMM**
- **Quadrature**
- **GMRES**
- **SDC**
- **Profile**
- **Vec (SIMD)**
- **Comm**
- **Vector**
- **Matrix**
- **VTUData**
- **Interpolation**
- **Tree**

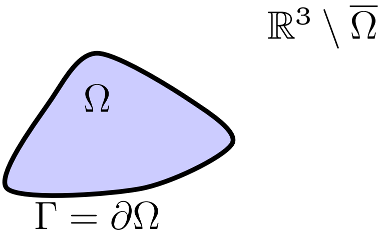
Step 1: Stokes Dirichlet boundary value problem (exterior)

- Stokes equations:

$$\begin{aligned} \Delta \mathbf{u} - \nabla p = 0 \qquad & \text{in } \mathbb{R}^3 \setminus \overline{\Omega}, \\ \nabla \cdot \mathbf{u} = 0 \qquad & \text{in } \mathbb{R}^3 \setminus \overline{\Omega}, \end{aligned}$$

- Boundary conditions:

$$\begin{aligned} \mathbf{u}|_\Gamma = \mathbf{u}_0 \qquad & \text{on } \Gamma, \\ \mathbf{u}(\mathbf{x}) \to \mathbf{0} \qquad & \text{as } |\mathbf{x}| \to \infty \end{aligned}$$



Solving BIEs with CSBQ

Step 2: Integral Equation Formulation

- Boundary integral representation:

$$\mathbf{u} = \mathcal{D}[\sigma] + \eta\, \mathcal{S}[\sigma] \quad \text{in } \mathbb{R}^3 \setminus \overline{\Omega},$$

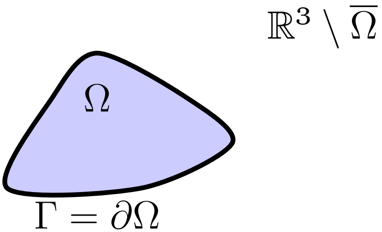
where

$$\mathcal{S}[\sigma](\mathbf{x}) := \int_\Gamma S(\mathbf{x}-\mathbf{y})\,\sigma(\mathbf{y})\,dS_{\mathbf{y}},$$
$$\mathcal{D}[\sigma](\mathbf{x}) := \int_\Gamma D(\mathbf{x}-\mathbf{y})\,\sigma(\mathbf{y})\,dS_{\mathbf{y}},$$
$$\eta = 1/(\epsilon \log \epsilon^{-1})$$

- Enforce boundary conditions:

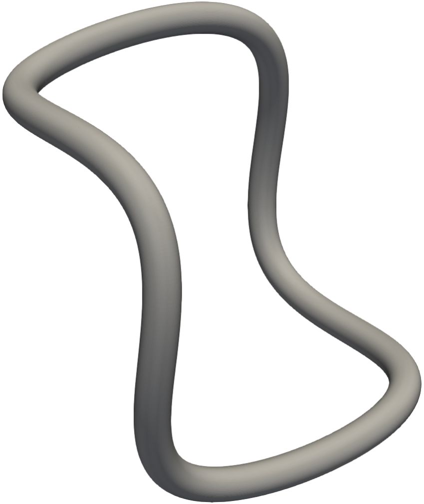
$$(I/2 + D + \eta\, S)\sigma = \mathbf{u}_0 \quad \text{on } \Gamma,$$

Solve for unknown σ



Solving BIEs with CSBQ

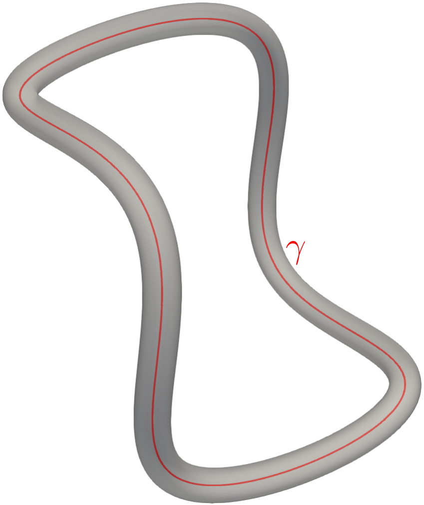
Step 3: Discretize the geometry



Solving BIEs with CSBQ

Step 3: Discretize the geometry

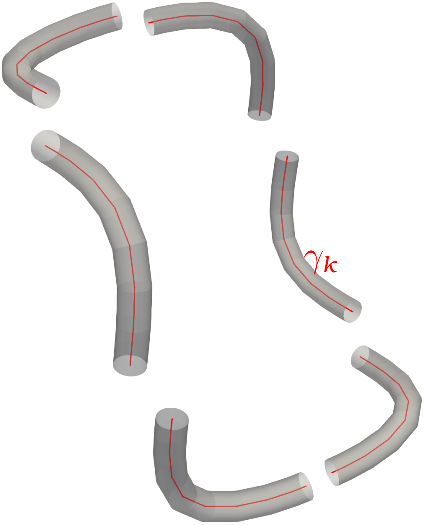
- Geometry described by centerline γ.



Solving BIEs with CSBQ

Step 3: Discretize the geometry

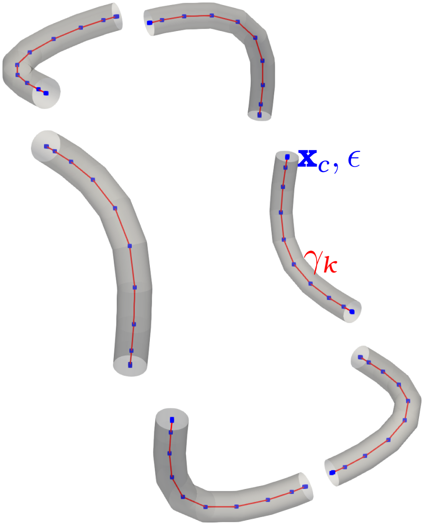
- Geometry described by centerline γ.
- Partition into panels.
  - Choose panel order and Fourier order.



Solving BIEs with CSBQ

Step 3: Discretize the geometry

- Geometry described by centerline γ.
- Partition into panels.
  - Choose panel order and Fourier order.
- Determine nodes $\mathbf{x}_c$ and radius ϵ.



Solving BIEs with CSBQ

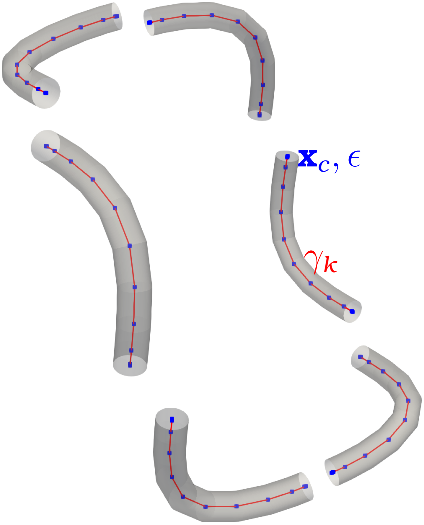
Step 3: Discretize the geometry

- Geometry described by centerline γ.
- Partition into panels.
  - Choose panel order and Fourier order.
- Determine nodes $\mathbf{x}_c$ and radius ϵ.

Code:

```
Vector<Long> ElemOrder, FourierOrder;
Vector<double> Xc, eps;
// set vector data ...

SlenderElemList<double> elem_lst(ElemOrder,
                                 FourierOrder, Xc, eps);
```



Step 4: Discretize Integral Operator

- Stokes combined field kernel:

```
struct Stokes3D_CF_ {
    template <Integer digits, class VecType>
    static void uKerMatrix(VecType (&u)[3][3], const VecType (&r)[3],
                           const VecType (&n)[3], const void* ctx_ptr) {
        // u[i][j] = [Stokes-SL] + [Stokes-DL];
    }
    // other member functions ...
};
```

Solving BIEs with CSBQ

Step 4: Discretize Integral Operator

- Stokes combined field kernel:

```
struct Stokes3D_CF_ {
    template <Integer digits, class VecType>
    static void uKerMatrix(VecType (&u)[3][3], const VecType (&r)[3],
                           const VecType (&n)[3], const void* ctx_ptr) {
        // u[i][j] = [Stokes-SL] + [Stokes-DL];
    }
    // other member functions ...
};
```

- Boundary integral operator:

```
Stokes3D_CF ker;
BoundaryIntegralOp<double,Stokes3D_CF> LayerPotenOp(ker);
LayerPotenOp.AddElemList(elem_lst); // add geometry to operator
LayerPotenOp.SetAccuracy(tol);      // set accuracy

const auto BIO = [&LayerPotenOp](Vector<double>* U,
                                 const Vector<double>& sigma) {
    LayerPotenOp.ComputePotential(*U, sigma);
    (*U) += sigma * 0.5;
};
```

$$\mathbf{u}(\mathbf{x}) = (\mathcal{D}+\mathcal{S})[\sigma](\mathbf{x})$$

$$K\,\sigma := (I/2+D+S)\,\sigma$$

Solving BIEs with CSBQ

FLATIRON INSTITUTE

Step 5: Solve Integral Equation

```
Vector<double> sigma, U0(LayerPotenOp.Dim(0));
// set boundary condition U0

GMRES<double> solver;
solver(&sigma, BIO, U0, tol);
```

Solving BIEs with CSBQ

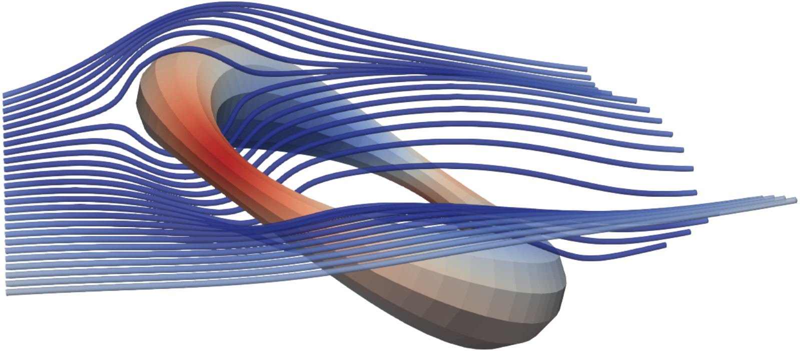
Step 6: Post-process

```
Vector<double> U;
LayerPotenOp.SetTargetCoord(Xtrg); // Set targets for LayerPotenOp
LayerPotenOp.ComputePotential(U, sigma); // Evaluate solution
//... Write to VTK file
```

Solving BIEs with CSBQ

Step 6: Post-process

```
Vector<double> U;
LayerPotenOp.SetTargetCoord(Xtrg); // Set targets for LayerPotenOp
LayerPotenOp.ComputePotential(U, sigma); // Evaluate solution
//... Write to VTK file
```



Conclusions

- Convergent boundary integral formulation for slender bodies,
  - unlike SBT, boundary conditions enforced to high accuracy.
- Special quadrature - efficient for aspect ratios as large as 10^5.
  - quadrature setup rates $\sim 20{,}000$ unknowns/s/core (at 7-digits).
- Combined field BIE formulations,
  - well-conditioned for slender-body geometries.

Conclusions

- Convergent boundary integral formulation for slender bodies,
  - unlike SBT, boundary conditions enforced to high accuracy.
- Special quadrature - efficient for aspect ratios as large as 10^5.
  - quadrature setup rates $\sim 20{,}000$ unknowns/s/core (at 7-digits).
- Combined field BIE formulations,
  - well-conditioned for slender-body geometries.

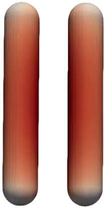
Limitations and ongoing work:

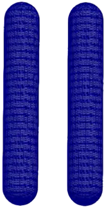
- Flexible fibers -- applications in biological fluids.

Extra

Extra

Numerical Results - Sedimentation Flow





Numerical Results - Sedimentation Flow