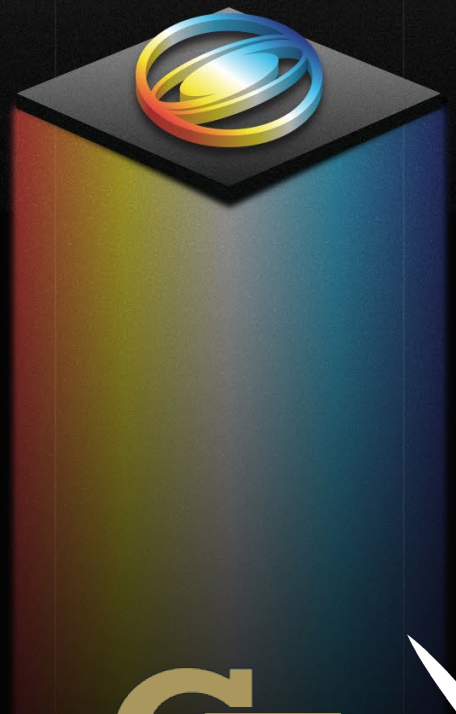
SIGGRAPH 2024
DENVER+ 28 JUL — 1 AUG

THE PREMIER CONFERENCE
& EXHIBITION ON
COMPUTER GRAPHICS &
INTERACTIVE TECHNIQUES

LIGHTNING-FAST METHOD OF FUNDAMENTAL SOLUTIONS

JIONG CHEN, INRIA
FLORIAN SCHÄFER, GEORGIA TECH
MATHIEU DESBRUN, INRIA / ECOLE POLYTECHNIQUE



SOLVING PDES

FINITE ELEMENT METHOD (FEM)

- Generalizable to most types of PDEs
  - Linear or nonlinear PDEs
  - Homogenous or inhomogeneous coefficients
- Requires volumetric discretization
  - Large number of degrees of freedom
  - High-quality volumetric tessellation is often hard to get

[Chen et al. 2018]

[Bargteil et al. 2007]

BOUNDARY ELEMENT METHOD (BEM)

SOLVING PDES

FINITE ELEMENT METHOD (FEM)

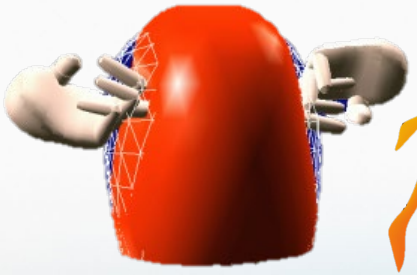
- Generalizable to most types of PDEs
  - Linear or nonlinear PDEs
  - Homogenous or inhomogeneous coefficients
- Requires volumetric discretization
  - Large number of degrees of freedom
  - High-quality volumetric tessellation is often hard to get

[Chen et al. 2018]

[Bargteil et al. 2007]

BOUNDARY ELEMENT METHOD (BEM)

- Only needs boundary discretization
  - Huge reduction in dimensionality
  - Works for infinite or semi-infinite domains as well
- Limited to certain types of problems
  - Only applicable to linear and homogenous problems
  - Involves dense and often asymmetric linear systems
  - **Quite common, though!**



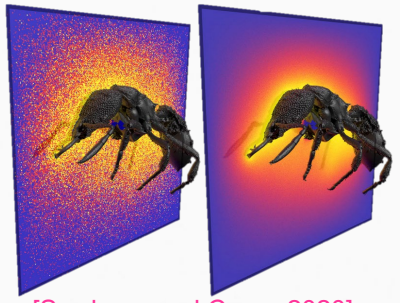
[James and Pai 1999]

[Orzan et al. 2008]

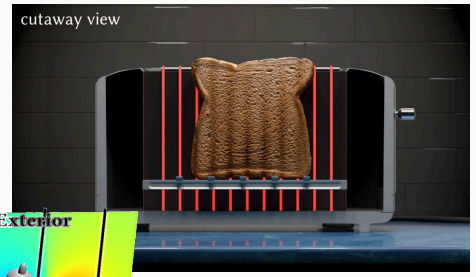
[Sugimoto et al. 2022]

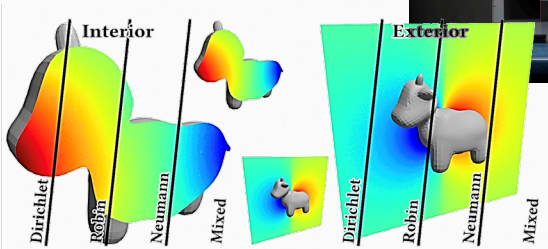
NUMERICAL METHODS FOR BEM

- Stochastic approaches (Walk-on-{Sphere/Star/Boundary})
  - Based on mean-value property of harmonic functions or Neumann series for $(I - A)^{-1}$
  - Fast to evaluate for a single point
  - Easily integrated to rendering code base
  - Yet, slow to converge (in the square root of #paths)



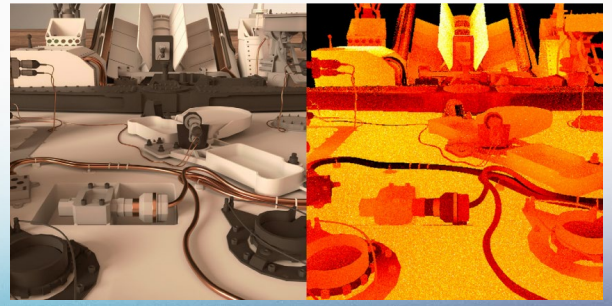
[Sawhney and Crane 2020]





[Sawhney et al. 2023]

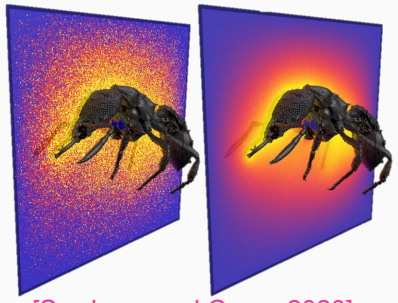
[Sugimoto et al. 2023]



[Miller et al. 2024]

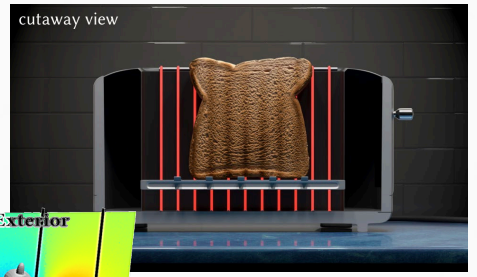
NUMERICAL METHODS FOR BEM

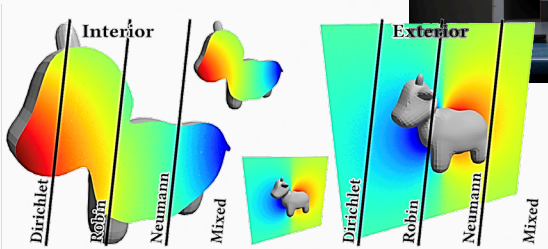
- Stochastic approaches (Walk-on-{Sphere/Star/Boundary})
  - Based on mean-value property of harmonic functions or Neumann series for $(I - A)^{-1}$
  - Fast to evaluate for a single point
  - Easily integrated to rendering code base
  - Yet, slow to converge (in the square root of #paths)



[Sawhney and Crane 2020]

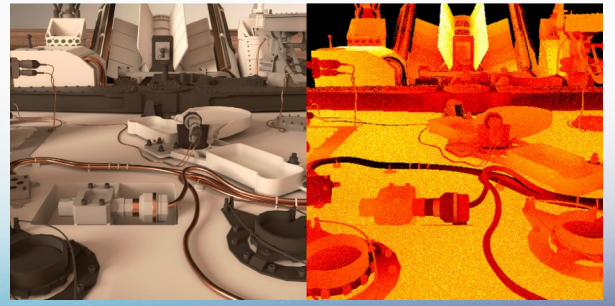
- Deterministic approaches
  - Direct solvers (e.g., LU, SVD)
    - High time/memory complexity
  - Iterative solvers (e.g., GMRES)
    - Slow/unguaranteed convergence





[Sawhney et al. 2023]

[Sugimoto et al. 2023]



[Miller et al. 2024]

RECAP BEM

- Methods to build Boundary Integral Equation (BIE) systems [Costabel 1984]
  - Direct approaches: solve for Dirichlet or Neumann boundary conditions
    - based on Green's third identity or its variants
  - Indirect approaches: solve for an unknown density on the boundary
    - E.g., "charges" for potential problems, "forces" for elasticity

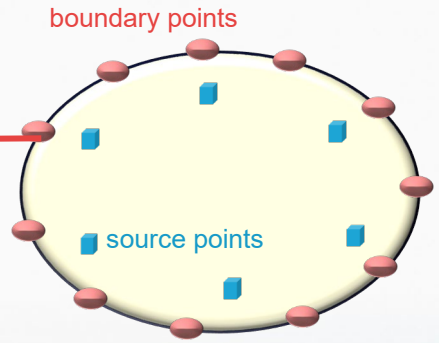
RECAP BEM

- Methods to build Boundary Integral Equation (BIE) systems [Costabel 1984]
  - Direct approaches: solve for Dirichlet or Neumann boundary conditions
    - based on Green's third identity or its variants
  - Indirect approaches: solve for an unknown density on the boundary
    - E.g., "charges" for potential problems, "forces" for elasticity
- Indirect approach: single layer potential for Dirichlet problems

RECAP BEM

- Methods to build Boundary Integral Equation (BIE) systems [Costabel 1984]
  - Direct approaches: solve for Dirichlet or Neumann boundary conditions
    - based on Green's third identity or its variants
  - Indirect approaches: solve for an unknown density on the boundary
    - E.g., "charges" for potential problems, "forces" for elasticity
- Indirect approach: single layer potential for Dirichlet problems
  - **Solve stage:** solve for "charges" that enforce a set of given boundary "potential"

$$\int_{\mathcal{M}} G(z, \boldsymbol{y})\sigma(\boldsymbol{y})\, \mathrm{d}v_{\boldsymbol{y}} = b(z) \quad \forall z \in \mathcal{M}.$$



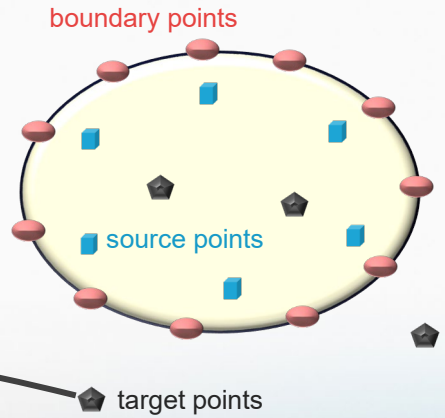
RECAP BEM

- Methods to build Boundary Integral Equation (BIE) systems [Costabel 1984]
  - Direct approaches: solve for Dirichlet or Neumann boundary conditions
    - based on Green's third identity or its variants
  - Indirect approaches: solve for an unknown density on the boundary
    - E.g., "charges" for potential problems, "forces" for elasticity
- Indirect approach: single layer potential for Dirichlet problems
  - **Solve stage:** solve for "charges" that enforce a set of given boundary "potential"

$$\int_{\mathcal{M}} G(z, \boldsymbol{y})\sigma(\boldsymbol{y})\, \mathrm{d}v_{\boldsymbol{y}} = b(z) \quad \forall z \in \mathcal{M}.$$

  - **Evaluation stage:** evaluate the "potential" at any target point in space

$$u(\boldsymbol{x}) = \int_{\mathcal{M}} G(\boldsymbol{x}, \boldsymbol{y})\sigma(\boldsymbol{y})\, \mathrm{d}v_{\boldsymbol{y}}.$$



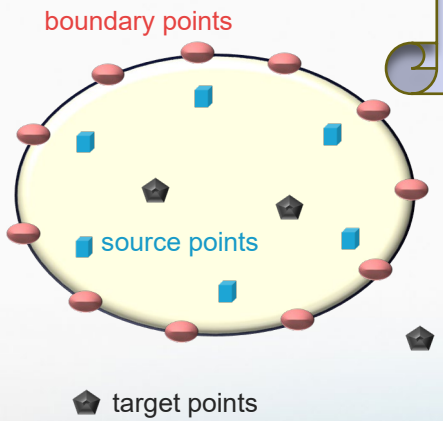
RECAP BEM

- Methods to build Boundary Integral Equation (BIE) systems [Costabel 1984]
  - Direct approaches: solve for Dirichlet or Neumann boundary conditions
    - based on Green's third identity or its variants
  - Indirect approaches: solve for an unknown density on the boundary
    - E.g., "charges" for potential problems, "forces" for elasticity
- Indirect approach: single layer potential for Dirichlet problems
  - **Solve stage:** solve for "charges" that enforce a set of given boundary "potential"

$$\int_{\mathcal{M}} G(z,\boldsymbol{y})\sigma(\boldsymbol{y})\,\mathrm{d}v_{\boldsymbol{y}} = b(z) \quad \forall z \in \mathcal{M}.$$

  - **Evaluation stage:** evaluate the "potential" at any target point in space

$$u(\boldsymbol{x}) = \int_{\mathcal{M}} G(\boldsymbol{x},\boldsymbol{y})\sigma(\boldsymbol{y})\,\mathrm{d}v_{\boldsymbol{y}}.$$

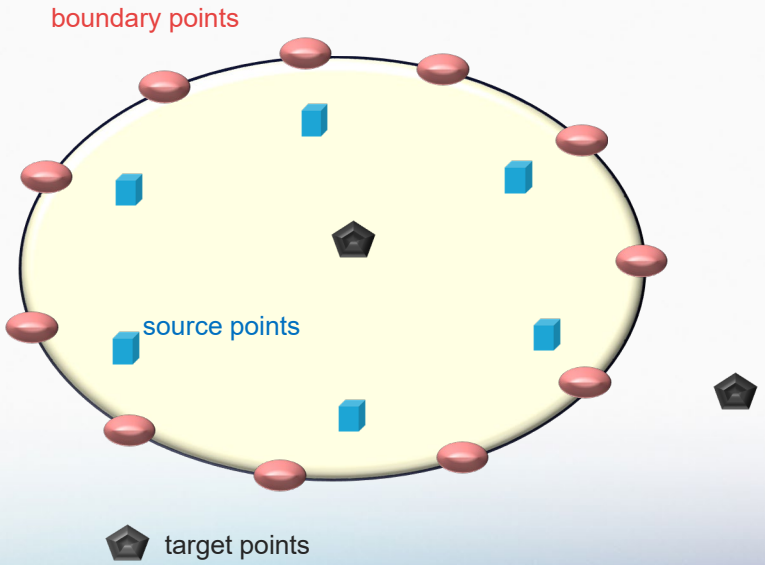


- Results in *Fredholm integral equation of the first kind*, more Ill-posed than the *second kind*
- Need efficient preconditioners
- ***Any symmetric and sparse structures to leverage to get a stable and scalable solver?***

EXPLOIT SYMMETRY

Discretize boundary integral equations (BIE)

$$\int_{\mathcal{M}} G(z, y)\sigma(y)\,\mathrm{d}v_y = b(z) \quad \forall z \in \mathcal{M}.$$



EXPLOIT SYMMETRY

$$K\ s = b$$

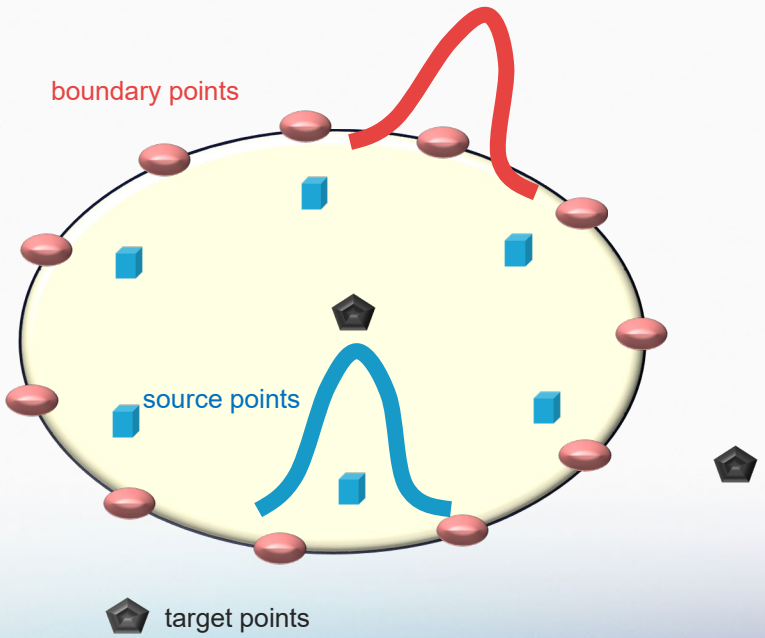
Discretize boundary integral equations (BIE)

$$\int_{\mathcal{M}} G(z, y)\sigma(y)\,\mathrm{d}v_y = b(z) \quad \forall z \in \mathcal{M}.$$

$$\sum_{j=1}^{S}\left(\iint_{\mathcal{M}\times\mathcal{M}} \phi_i(y)G(y, z)\psi_j(z)\,\mathrm{d}v_y\mathrm{d}v_z\right) s_j = \int_{\mathcal{M}} b(y)\phi_i(y)\mathrm{d}v_y$$

Discretize boundary data $b(z) = \sum_i \psi_i(z)b_i$

Discretize sources $\sigma(y) = \sum_j \phi_j(y)s_j$



EXPLOIT SYMMETRY

$$K\ s = b$$

Discretize boundary integral equations (BIE)

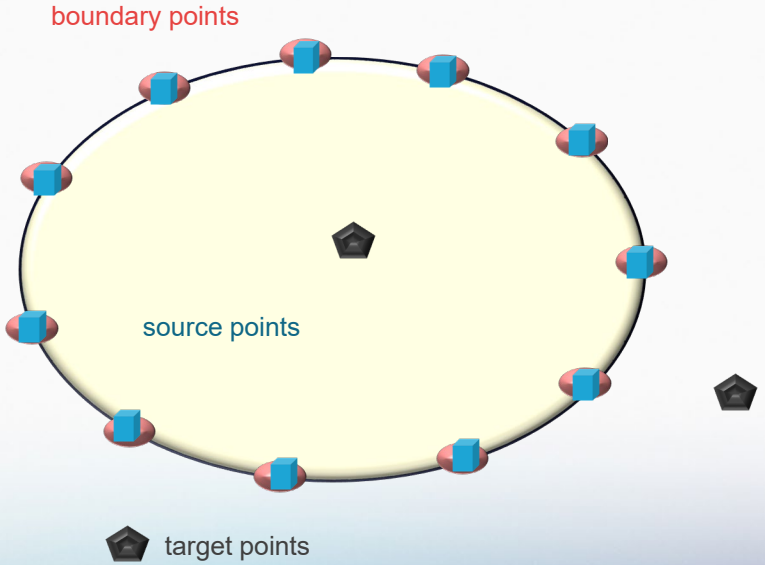
$$\int_{\mathcal{M}} G(z,y)\sigma(y)\,\mathrm{d}v_y = b(z) \quad \forall z \in \mathcal{M}.$$

$$\sum_{j=1}^{S}\left(\iint_{\mathcal{M}\times\mathcal{M}} \phi_i(y)G(y,z)\psi_j(z)\,\mathrm{d}v_y\mathrm{d}v_z\right)s_j = \int_{\mathcal{M}} b(y)\phi_i(y)\mathrm{d}v_y$$

Discretize boundary data $b(z) = \sum_i \psi_i(z)b_i$

Discretize sources $\sigma(y) = \sum_j \phi_j(y)s_j$

- To obtain a symmetric discrete BIE
  - Either identical basis functions for collocated source and boundary points
  - Or solve least-squares problem $\boldsymbol{K^T K s = K^T b}$



EXPLOIT SYMMETRY

Discretize boundary integral equations (BIE)

$$K\,s = b$$

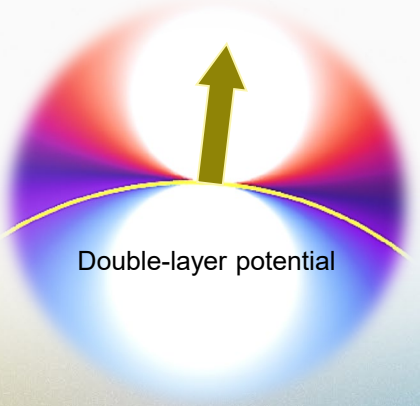
$$\int_{\mathcal{M}} G(z,y)\sigma(y)\,\mathrm{d}\nu_y = b(z) \quad \forall z \in \mathcal{M}.$$

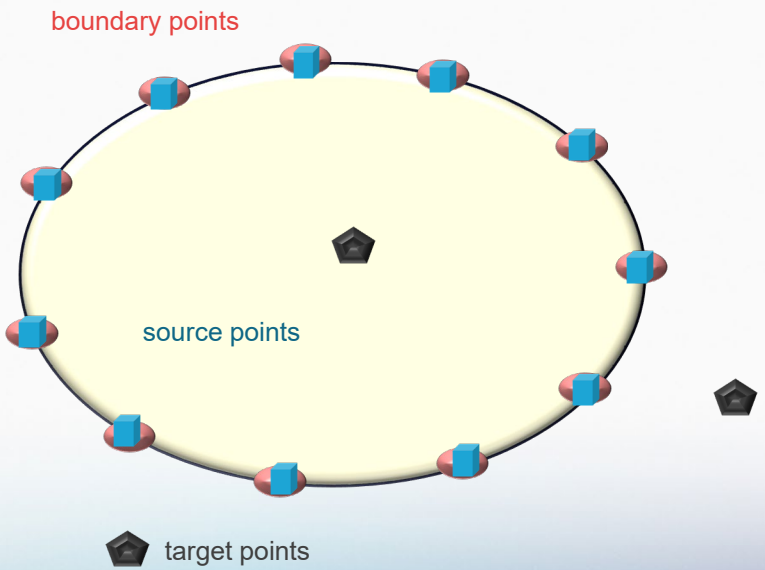
$$\sum_{j=1}^{S}\left(\iint_{\mathcal{M}\times\mathcal{M}} \phi_i(y)G(y,z)\psi_j(z)\,\mathrm{d}\nu_y\mathrm{d}\nu_z\right)s_j = \int_{\mathcal{M}} b(y)\phi_i(y)\mathrm{d}\nu_y$$

Discretize boundary data $b(z) = \sum_i \psi_i(z)b_i$

Discretize sources $\sigma(y) = \sum_j \phi_j(y)s_j$

- To obtain a symmetric discrete BIE
  - Either identical basis functions for collocated source and boundary points
  - Or solve least-squares problem $\boldsymbol{K^T K s = K^T b}$
    - e.g., Fredholm integral equations of the second kind
      - Double-layer potential for Dirichlet problems
      - Single-layer potential for Neumann problems





EXPLOIT SPARSITY

- Directly applying incomplete Cholesky to factorize K

  [Chen et al. 2021]

  - *Accuracy issue:* Numerous entries must be dropped out for efficiency
  - *Stability issue:* Loss of positive definiteness causes breakdowns

$$Ks = b$$

$$K \approx LL^T$$

EXPLOIT SPARSITY

- Directly applying incomplete Cholesky to factorize K

  [Chen et al. 2021]

  - *Accuracy issue:* Numerous entries must be dropped out for efficiency
  - *Stability issue:* Loss of positive definiteness causes breakdowns
- However, boundary integral operators are conceptually close to the inverse of their differential operator
  - Green function is the solution subject to a singular impulse
  - E.g., in elasticity, a BIE matrix acts like the inverse of stiffness, or compliance

Compliance

$$Ks = b$$

Forces Displacements

$$K \approx LL^T$$

EXPLOIT SPARSITY

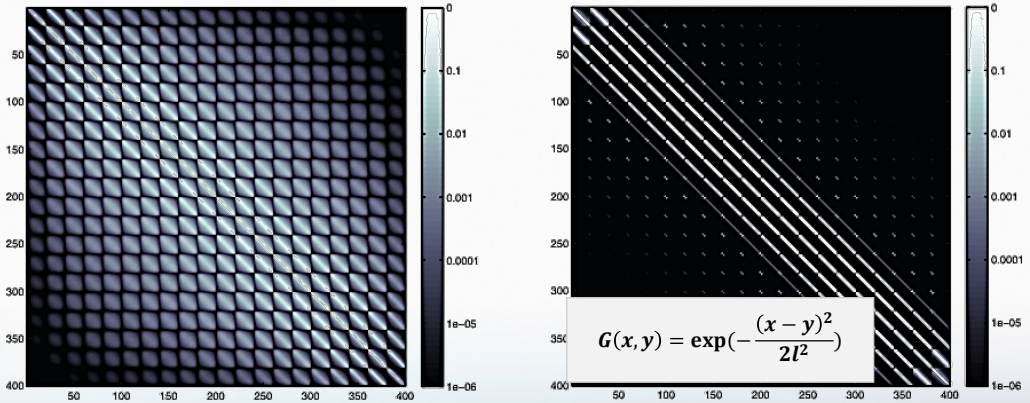
- Directly applying incomplete Cholesky to factorize K

  [Chen et al. 2021]

  - *Accuracy issue:* Numerous entries must be dropped out for efficiency
  - *Stability issue:* Loss of positive definiteness causes breakdowns
- However, boundary integral operators are conceptually close to the inverse of their differential operator
  - Green function is the solution subject to a singular impulse
  - E.g., in elasticity, a BIE matrix acts like the inverse of stiffness, or compliance
- So, the inverse of BIE matrices could be sparse
  - True for many covariance matrices assembled by fast-decaying kernel functions in Gaussian Process
  - Similar for Green's functions as well

$$Ks = b$$

$$K \approx LL^T$$



(a) Inverse Laplacian matrix (b) Inverse exponential covariance matrix

[Chow and Saad 2014]

INVERSE CHOLESKY PRECONDITIONER

- We leverage inverse Cholesky factorization to precondition BIE matrices

$$K^{-1} \approx L_S L_S^T$$

INVERSE CHOLESKY PRECONDITIONER

- We leverage inverse Cholesky factorization to precondition BIE matrices

$$Ks = b \quad \boxed{K^{-1} \approx L_S L_S^T} \quad s \approx L_S L_S^T b$$

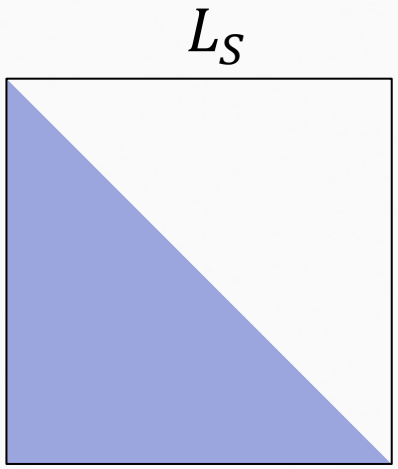
INVERSE CHOLESKY PRECONDITIONER

- We leverage inverse Cholesky factorization to precondition BIE matrices

$$Ks = b \quad \boxed{K^{-1} \approx L_S L_S^T} \quad s \approx L_S L_S^T b$$

- Kaporin's construction for L_S [Kaporin 1994]

$$\boldsymbol{L}_{S_j,j} = \frac{\boldsymbol{K}_{S_j,S_j}^{-1}\mathbb{e}_j}{\sqrt{\mathbb{e}_j^{\mathsf{T}}\boldsymbol{K}_{S_j,S_j}^{-1}\mathbb{e}_j}}, \quad \forall j = 1..B,$$



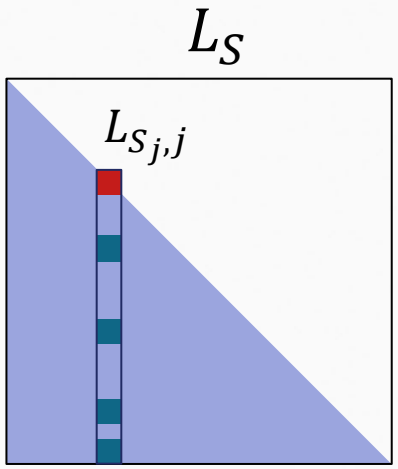
INVERSE CHOLESKY PRECONDITIONER

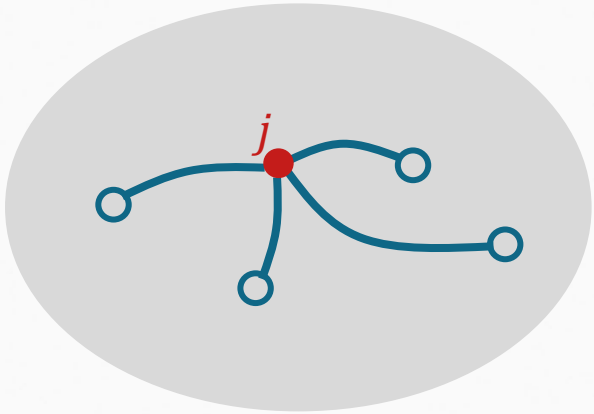
- We leverage inverse Cholesky factorization to precondition BIE matrices

$$Ks = b \quad \boxed{K^{-1} \approx L_S L_S^T} \quad s \approx L_S L_S^T b$$

- Kaporin's construction for L_S [Kaporin 1994]

$$L_{S_j,j} = \frac{K^{-1}_{S_j,S_j} e_j}{\sqrt{e_j^\top K^{-1}_{S_j,S_j} e_j}}, \quad \forall j = 1..B,$$





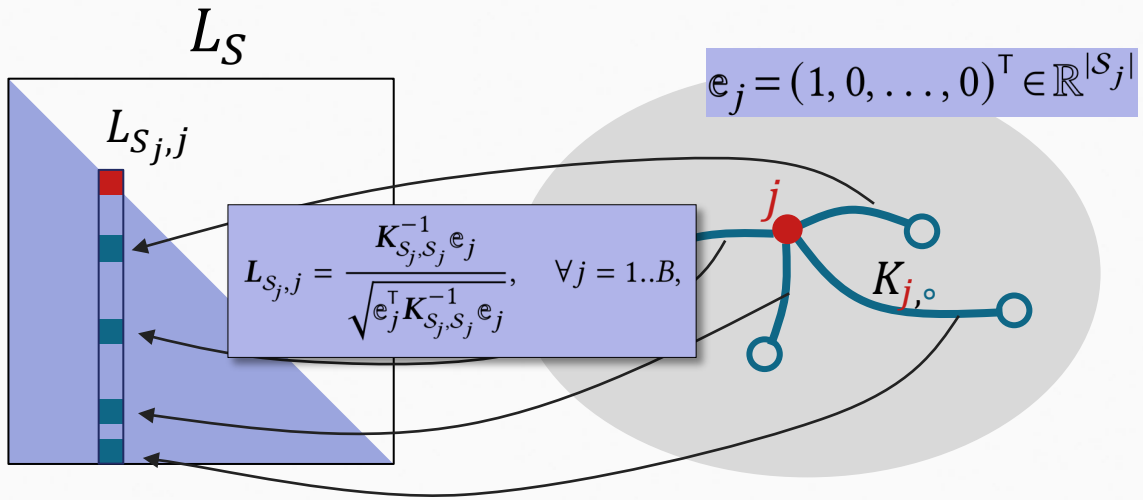
INVERSE CHOLESKY PRECONDITIONER

- We leverage inverse Cholesky factorization to precondition BIE matrices

$$Ks = b \quad \boxed{K^{-1} \approx L_S L_S^T} \quad s \approx L_S L_S^T b$$

- Kaporin's construction for L_S [Kaporin 1994]

$$L_{S_j,j} = \frac{K^{-1}_{S_j,S_j} e_j}{\sqrt{e_j^\top K^{-1}_{S_j,S_j} e_j}}, \quad \forall j = 1..B,$$



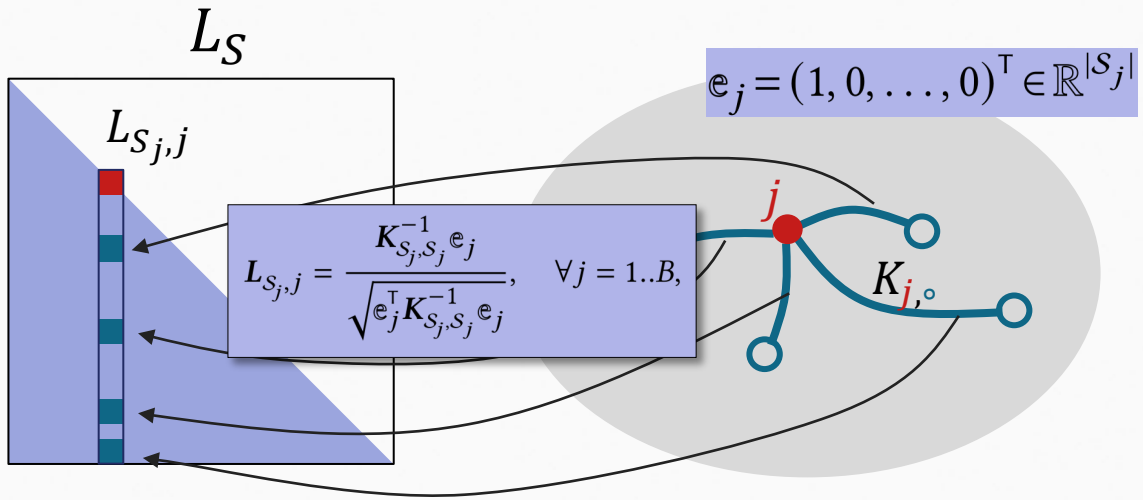
INVERSE CHOLESKY PRECONDITIONER

- We leverage inverse Cholesky factorization to precondition BIE matrices

$$Ks = b \quad \xrightarrow{K^{-1} \approx L_S L_S^T} \quad s \approx L_S L_S^T b$$

- Kaporin's construction for L_S [Kaporin 1994]

$$L_{S_j,j} = \frac{K_{S_j,S_j}^{-1} e_j}{\sqrt{e_j^\top K_{S_j,S_j}^{-1} e_j}}, \quad \forall j = 1..B,$$



- Properties
  - **Massively parallel:** each column of L_S is computed independently to others. Good for GPUs!
  - **Memory efficient:** no need to assemble the global BIE matrix.
  - **Stable:** no breakdowns will occur
  - **Variational interpretation(s):** minimizing Kaporin's condition number*, KL-divergence, and a constrained quadratic form

$$* \quad \kappa_{\mathrm{Kap}}(M) = \frac{1}{B} \frac{\mathrm{tr}(M)}{\det(M)^{1/B}}$$

OVERVIEW

Precompute (CPU)

Needs boundary meshes

or just points

Compute (GPU)

Needs PDEs and

associated Green function

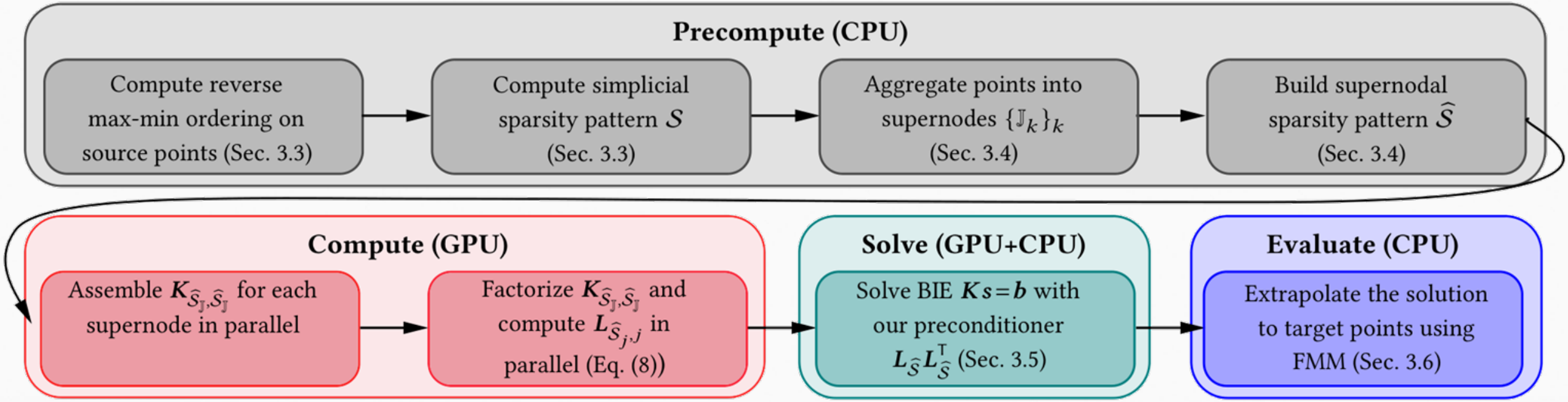
Solve (GPU+CPU)

Needs boundary conditions

Evaluate (CPU)

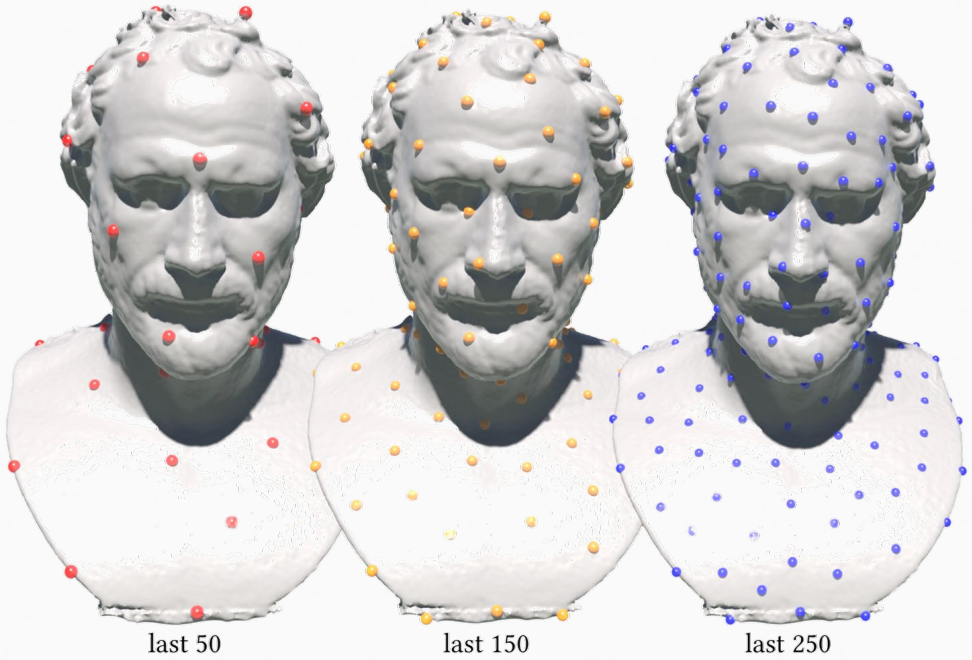
Needs interpolate/extrapolation points

OVERVIEW


Precompute (CPU)
Compute reverse max-min ordering on source points (Sec. 3.3)
Compute simplicial sparsity pattern $\mathcal{S}$ (Sec. 3.3)
Aggregate points into supernodes $\{\mathbb{J}_k\}_k$ (Sec. 3.4)
Build supernodal sparsity pattern $\widehat{\mathcal{S}}$ (Sec. 3.4)
Compute (GPU)
Assemble $K_{\widehat{\mathcal{S}}_{\mathbb{J}},\widehat{\mathcal{S}}_{\mathbb{J}}}$ for each supernode in parallel
Factorize $K_{\widehat{\mathcal{S}}_{\mathbb{J}},\widehat{\mathcal{S}}_{\mathbb{J}}}$ and compute $L_{\widehat{\mathcal{S}}_{j,j}}$ in parallel (Eq. (8))
Solve (GPU+CPU)
Solve BIE $Ks=b$ with our preconditioner $L_{\widehat{\mathcal{S}}}L_{\widehat{\mathcal{S}}}^{\mathsf{T}}$ (Sec. 3.5)
Evaluate (CPU)
Extrapolate the solution to target points using FMM (Sec. 3.6)

REORDERING DOFS

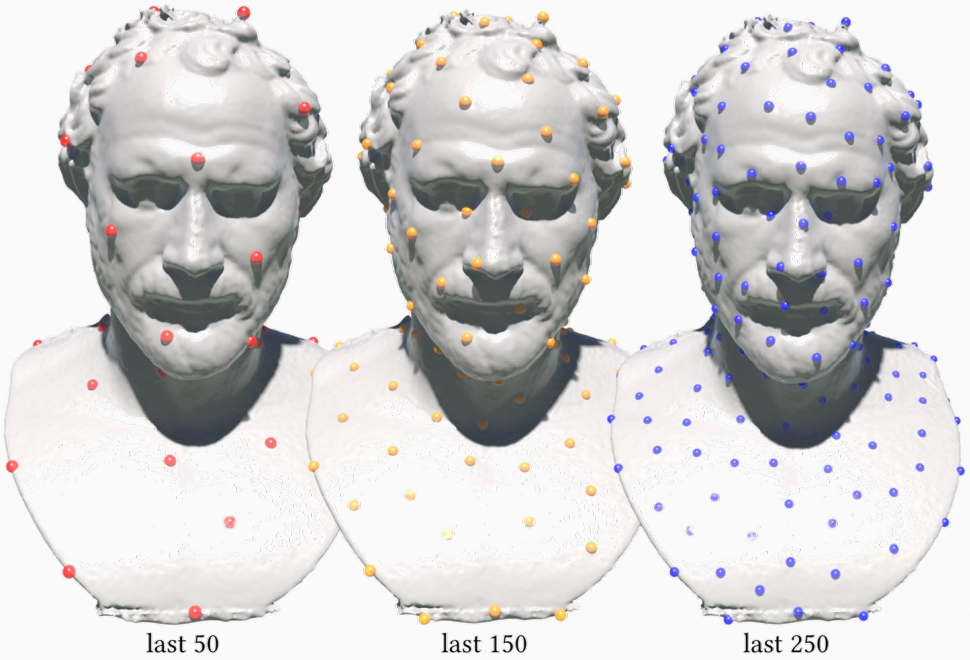
- Fine-to-coarse ordering by farthest point sampling [Chen et al. 2021]
  - Max-min ordering $i_k = \underset{q}{\operatorname{argmax}} \min_{p \in \{0,k-1\}} \operatorname{dist}(\boldsymbol{y}_q, \boldsymbol{y}_{i_p})$,
  - Reverse max-min ordering $\boldsymbol{P} = \{i_{B-1}, ..., i_1, i_0\}$, i.e., fine-to-coarse



A fine-to-coarse reordering

REORDERING DOFS

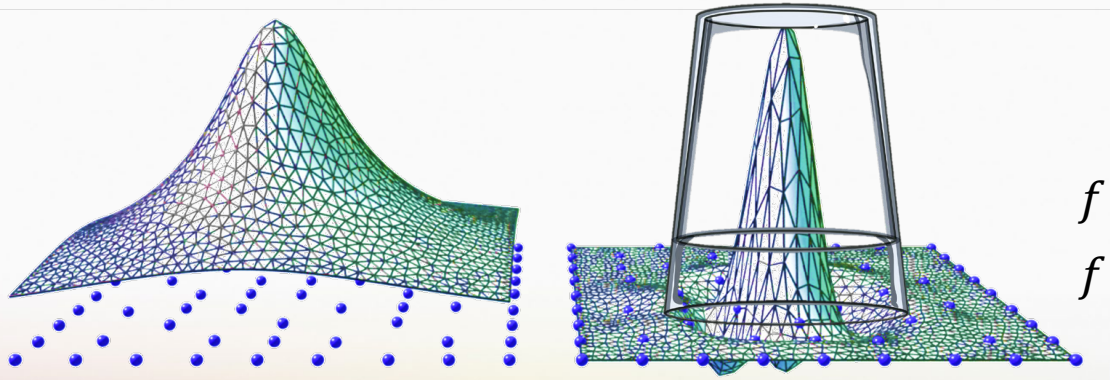
- Fine-to-coarse ordering by farthest point sampling [Chen et al. 2021]
  - Max-min ordering $i_k = \underset{q}{\operatorname{argmax}} \min_{p \in \{0,k-1\}} \operatorname{dist}(\boldsymbol{y}_q, \boldsymbol{y}_{i_p})$,
  - Reverse max-min ordering $\boldsymbol{P} = \{i_{B-1}, ..., i_1, i_0\}$, i.e., fine-to-coarse
- Intuition
  - Make sampling points space uniformly within each scale
  - The screening effect in kriging [Stein 2002]

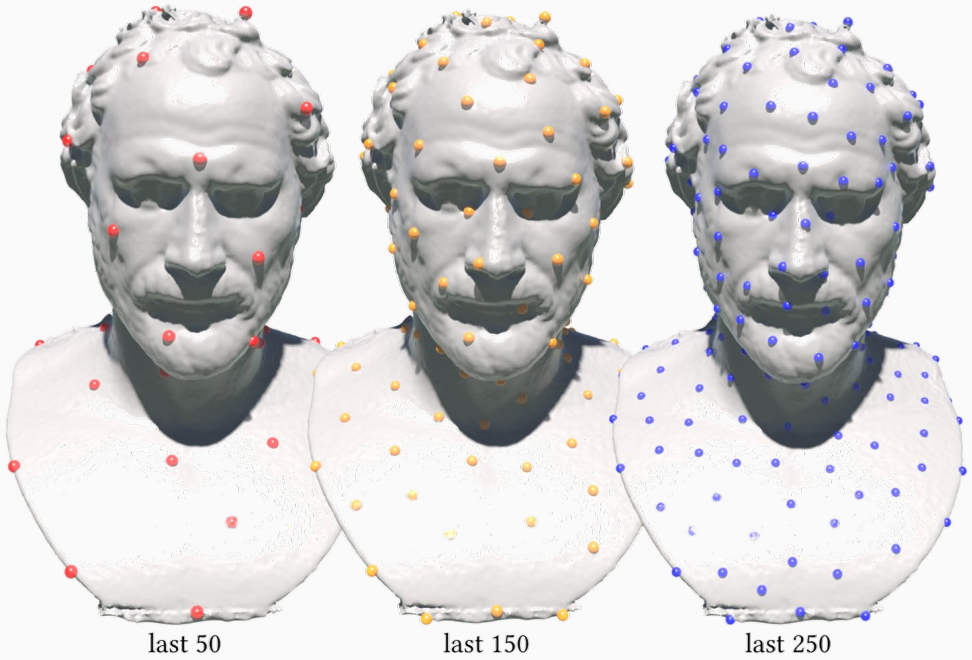


A fine-to-coarse reordering

REORDERING DOFS

- Fine-to-coarse ordering by farthest point sampling [Chen et al. 2021]
  - Max-min ordering $i_k = \underset{q}{\operatorname{argmax}} \min_{p \in \{0,k-1\}} \operatorname{dist}(\boldsymbol{y}_q, \boldsymbol{y}_{i_p})$,
  - Reverse max-min ordering $\boldsymbol{P} = \{i_{B-1}, ..., i_1, i_0\}$, i.e., fine-to-coarse
- Intuition
  - Make sampling points space uniformly within each scale
  - The screening effect in kriging [Stein 2002]
    - GP: conditioning a subset of points results in localized correlations





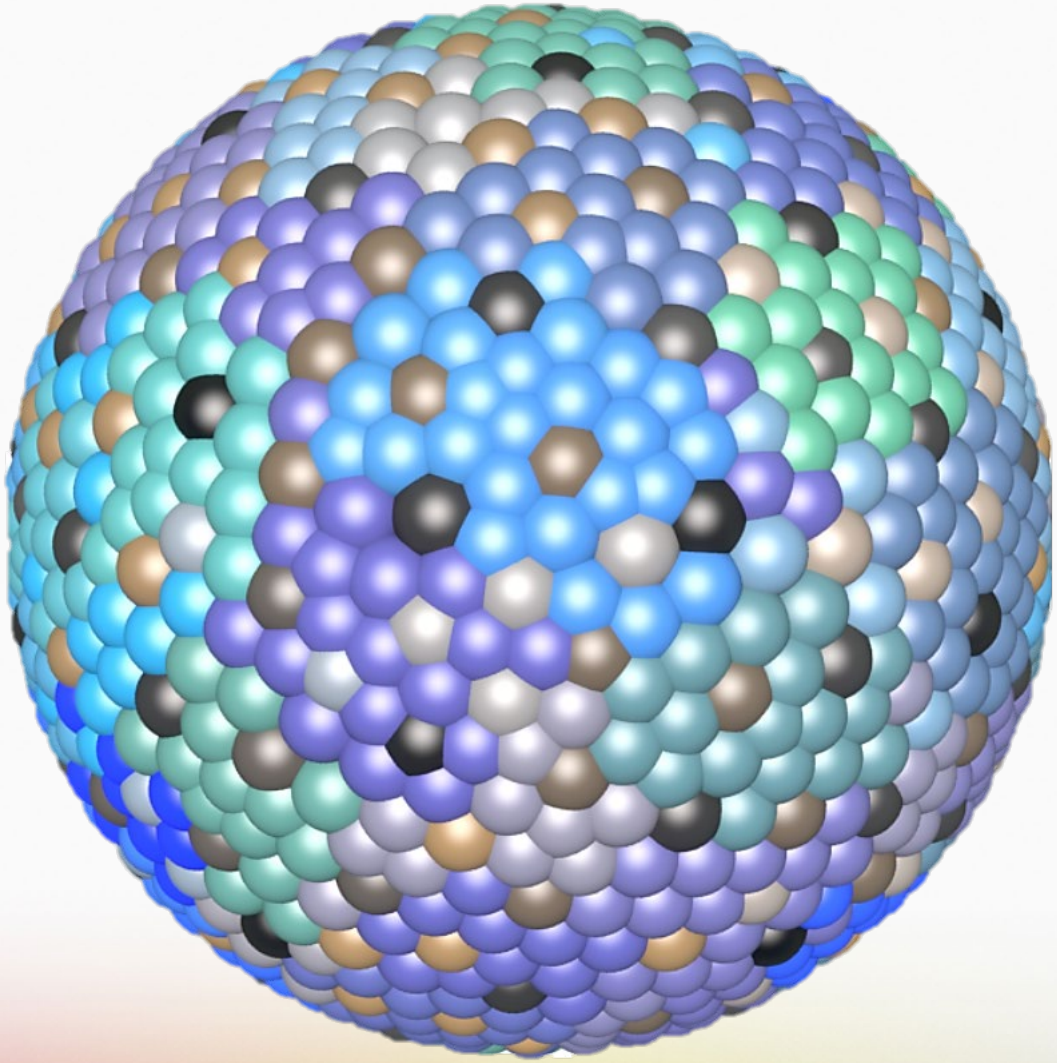
A fine-to-coarse reordering

$$f(A,B,C,D) = f(A)f(B|A)f(C|A,B)f(D|A,B,C) = N(0,\Sigma)$$

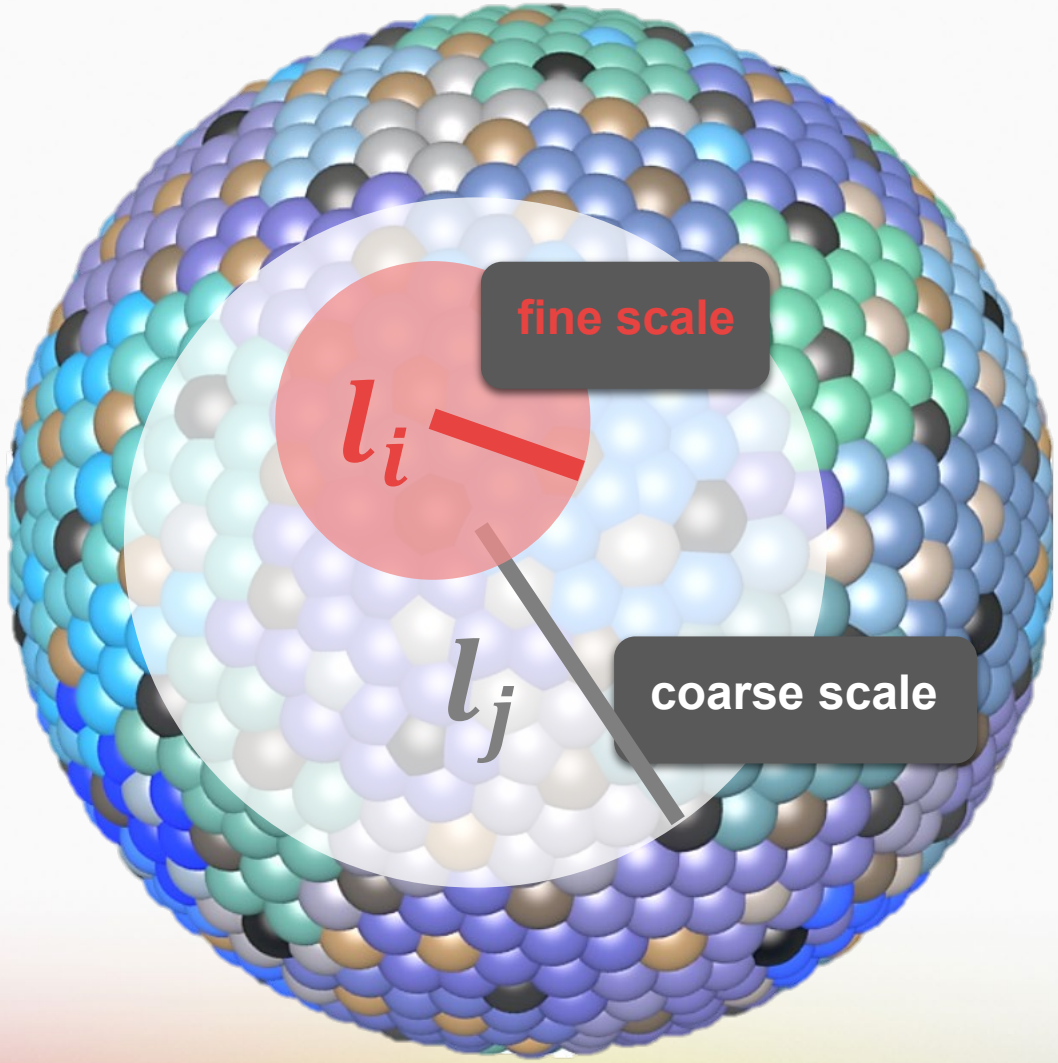
$$f(A,B,C,D) \approx f(A)f(B|A)f(C|A,\cancel{B})f(D|A,B,\cancel{C}) = N(0,(LL^{\mathrm{T}})^{-1})$$

Too far Too far

CONSTRUCTING SPARSITY PATTERN



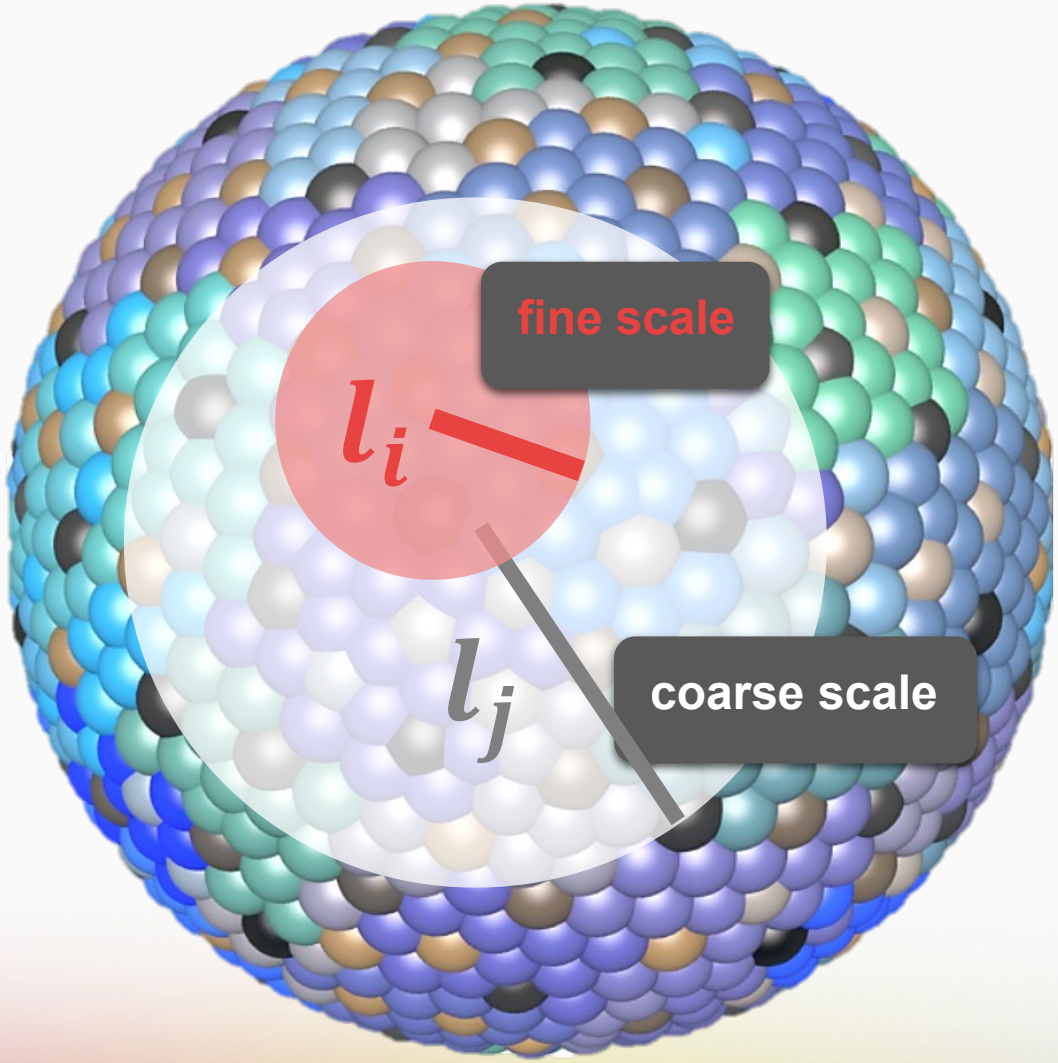
CONSTRUCTING SPARSITY PATTERN



- Length scale returned in coarse-to-fine ordering

$$\ell_{i_k} = \min_{p \in \{0,k-1\}} \mathrm{dist}(\boldsymbol{y}_{i_k}, \boldsymbol{y}_{i_p})$$

CONSTRUCTING SPARSITY PATTERN



- Length scale returned in coarse-to-fine ordering

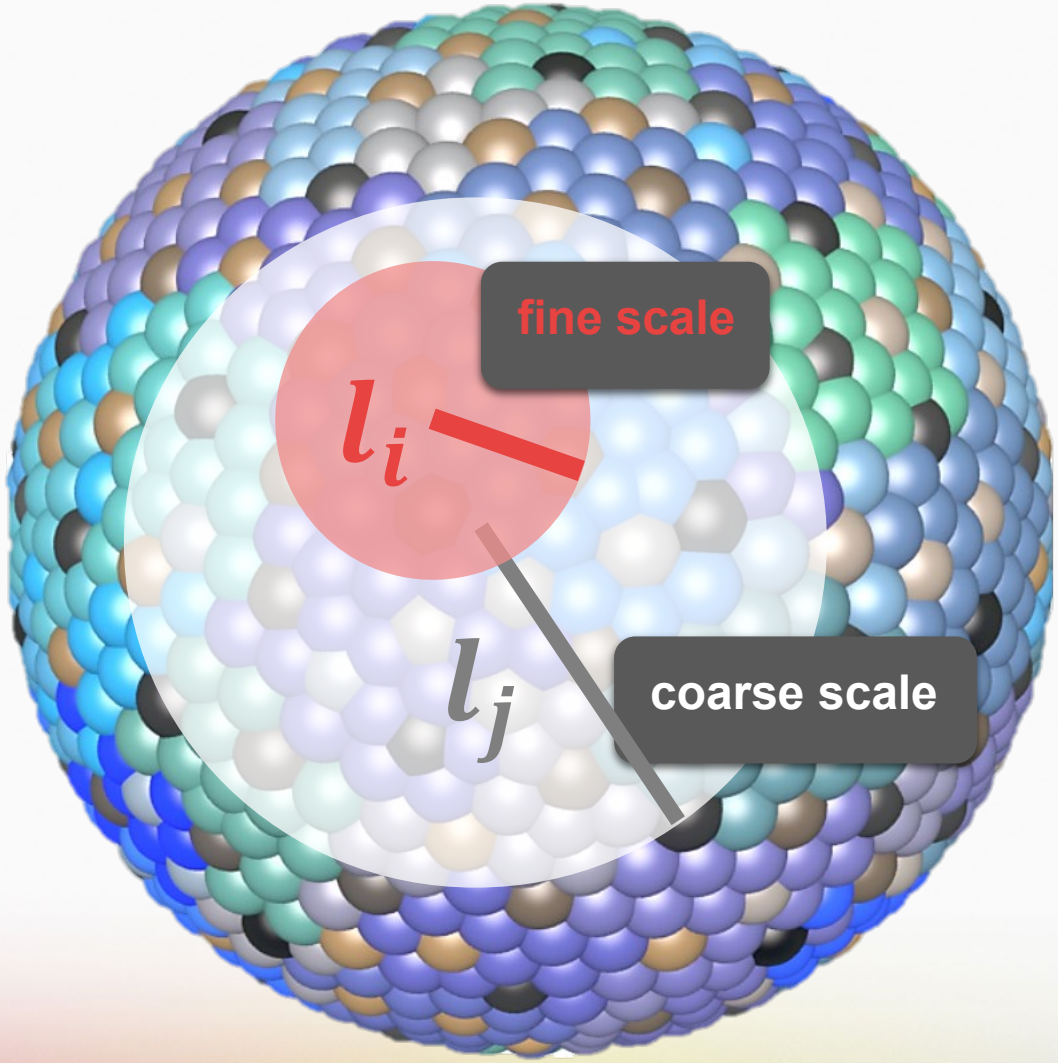
$$\ell_{i_k} = \min_{p \in \{0,k-1\}} \operatorname{dist}(\boldsymbol{y}_{i_k}, \boldsymbol{y}_{i_p})$$

- Lower-triangular sparsity patter

$$\mathcal{S} := \{(i,j) | i \geq j \text{ and } \operatorname{dist}(x_i, x_j) \leq \rho \min(\ell_i, \ell_j)\}$$

- **Again, screening effect:** a fine-scale point is unlikely to be correlated to distant points on coarser scales

CONSTRUCTING SPARSITY PATTERN



- Length scale returned in coarse-to-fine ordering

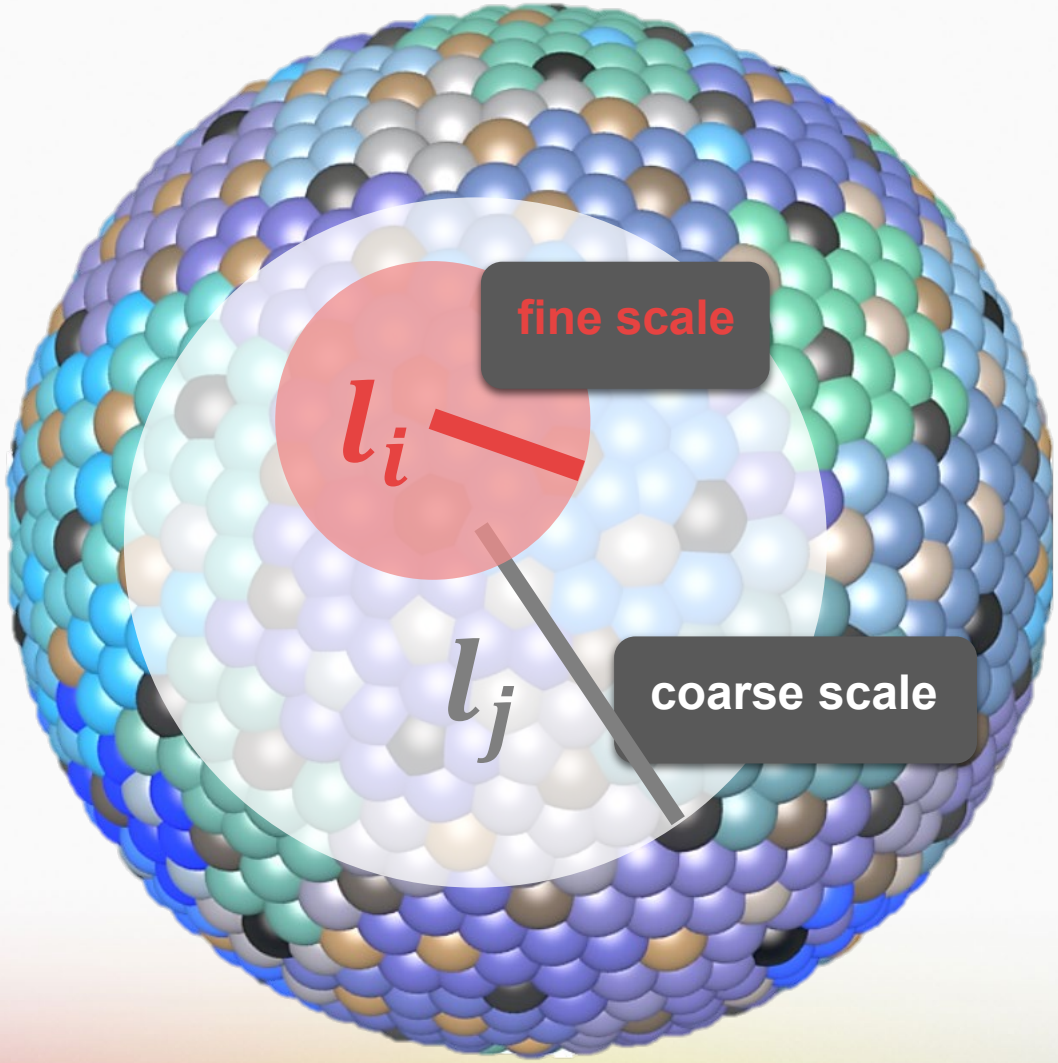
$$\ell_{i_k} = \min_{p \in \{0,k-1\}} \operatorname{dist}(\boldsymbol{y}_{i_k}, \boldsymbol{y}_{i_p})$$

- Lower-triangular sparsity patter

$$\mathcal{S} := \{(i,j) | i \geq j \text{ and } \operatorname{dist}(x_i, x_j) \leq \rho \min(\ell_i, \ell_j)\}$$

- **Again, screening effect:** a fine-scale point is unlikely to be correlated to distant points on coarser scales

CONSTRUCTING SPARSITY PATTERN



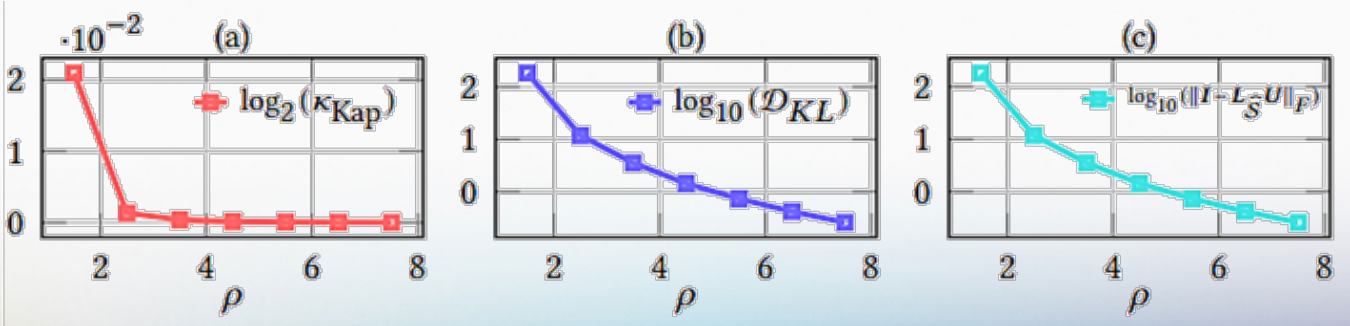
- Length scale returned in coarse-to-fine ordering

$$\ell_{i_k} = \min_{p \in \{0,k-1\}} \operatorname{dist}(\boldsymbol{y}_{i_k}, \boldsymbol{y}_{i_p})$$

- Lower-triangular sparsity patter

$$\mathcal{S} := \{(i,j) | i \geq j \text{ and } \operatorname{dist}(x_i, x_j) \leq \rho \min(\ell_i, \ell_j)\}$$

- **Again, screening effect:** a fine-scale point is unlikely to be correlated to distant points on coarser scales



EFFICIENT IMPLEMENTATION

FOR PRECONDTIONER

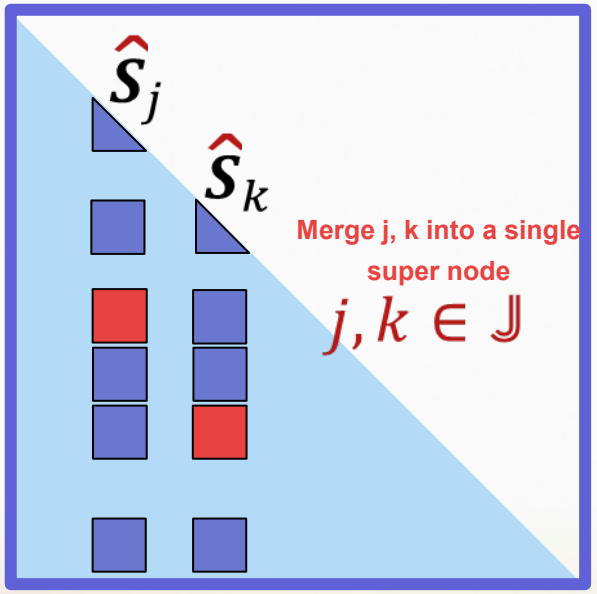
FOR PCG ITERATIONS

EFFICIENT IMPLEMENTATION

FOR PRECONDTIONER

- Supernode mode to reuse local factorizations as much as possible

Supernodal sparsity pattern

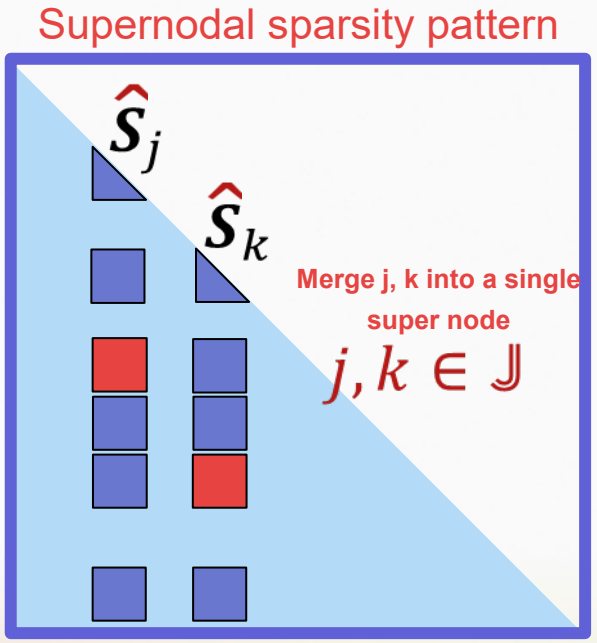


FOR PCG ITERATIONS

EFFICIENT IMPLEMENTATION

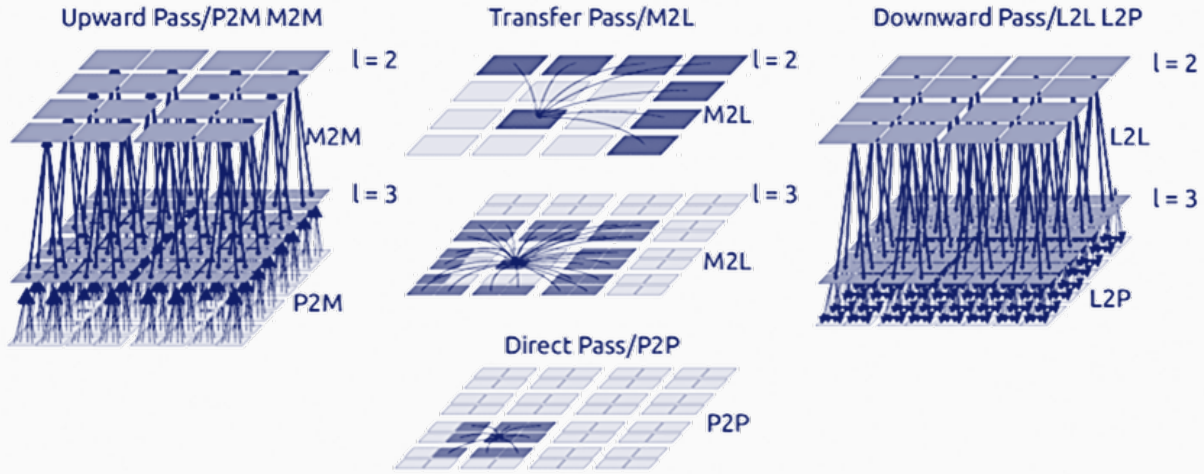
FOR PRECONDTIONER

- Supernode mode to reuse local factorizations as much as possible



FOR PCG ITERATIONS

- Fast Multipole Method to evaluate matrix-vector products



EXAMPLES OF APPLICATION

LAPLACE'S EQUATION

$$\Delta u = 0$$

$$G(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} -\dfrac{1}{2\pi} \ln(r), & \text{in 2D} \\ \dfrac{1}{4\pi r}, & \text{in 3D} \end{cases}$$

LINEAR ELASTICITY

$$\Delta u + \frac{1}{1-2\nu} \nabla(\nabla \cdot u) = 0,$$

$$G(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \dfrac{a-b}{r} \ln(1/r) \boldsymbol{I} + \dfrac{b}{r^2} \boldsymbol{r}\boldsymbol{r}^{\mathsf{T}}, & \text{in 2D} \\ \dfrac{a-b}{r} \boldsymbol{I} + \dfrac{b}{r^3} \boldsymbol{r}\boldsymbol{r}^{\mathsf{T}}, & \text{in 3D} \end{cases}$$

HELMHOLTZ EQUATION

$$\Delta u + k^2 u = 0,$$

$$G(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \dfrac{\mathbb{i}}{4} H_0^{(1)}(kr), & \text{in 2D,} \\ \dfrac{\exp(\mathbb{i}kr)}{4\pi r}, & \text{in 3D,} \end{cases}$$

EXAMPLES OF APPLICATION

LAPLACE'S EQUATION

$$\Delta u = 0$$

$$G(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} -\dfrac{1}{2\pi} \ln(r), & \text{in 2D} \\ \dfrac{1}{4\pi r}, & \text{in 3D} \end{cases}$$

LINEAR ELASTICITY

$$\Delta u + \frac{1}{1-2\nu} \nabla(\nabla \cdot u) = 0,$$

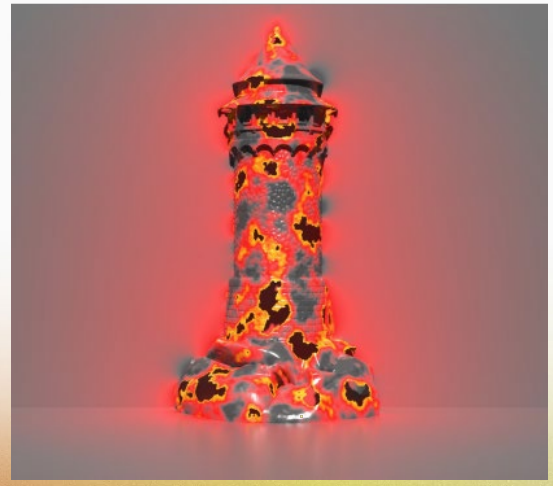
$$G(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \dfrac{a-b}{r} \ln(1/r) \boldsymbol{I} + \dfrac{b}{r^2} \boldsymbol{r}\boldsymbol{r}^{\mathsf{T}}, & \text{in 2D} \\ \dfrac{a-b}{r} \boldsymbol{I} + \dfrac{b}{r^3} \boldsymbol{r}\boldsymbol{r}^{\mathsf{T}}, & \text{in 3D} \end{cases}$$

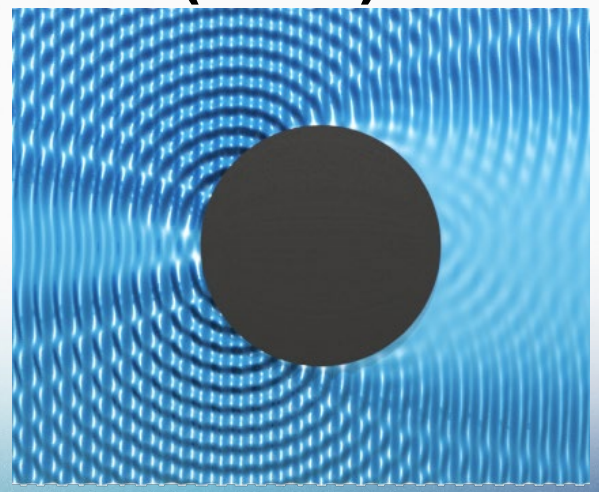
HELMHOLTZ EQUATION

$$\Delta u + k^2 u = 0,$$

$$G(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \dfrac{\mathbb{i}}{4} H_0^{(1)}(kr), & \text{in 2D,} \\ \dfrac{\exp(\mathbb{i}kr)}{4\pi r}, & \text{in 3D,} \end{cases}$$

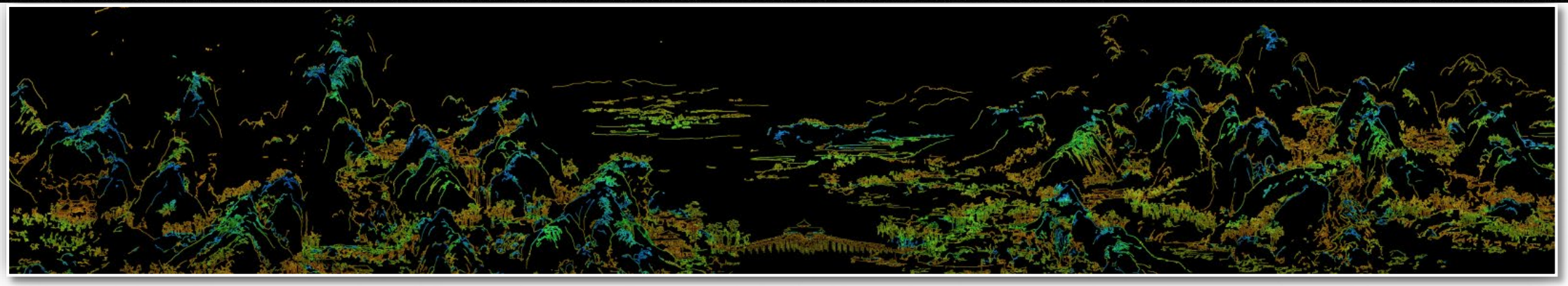
The method of fundamental solutions (MFS)



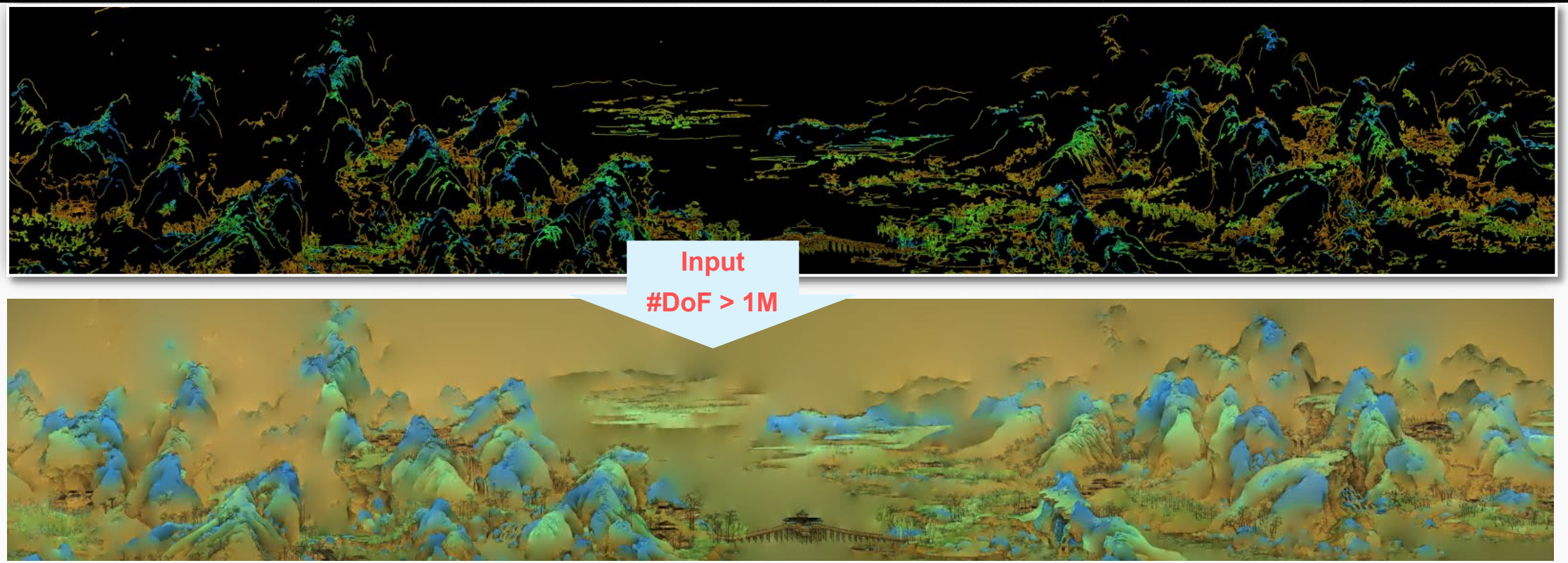


LAPLACE'S EQUATION

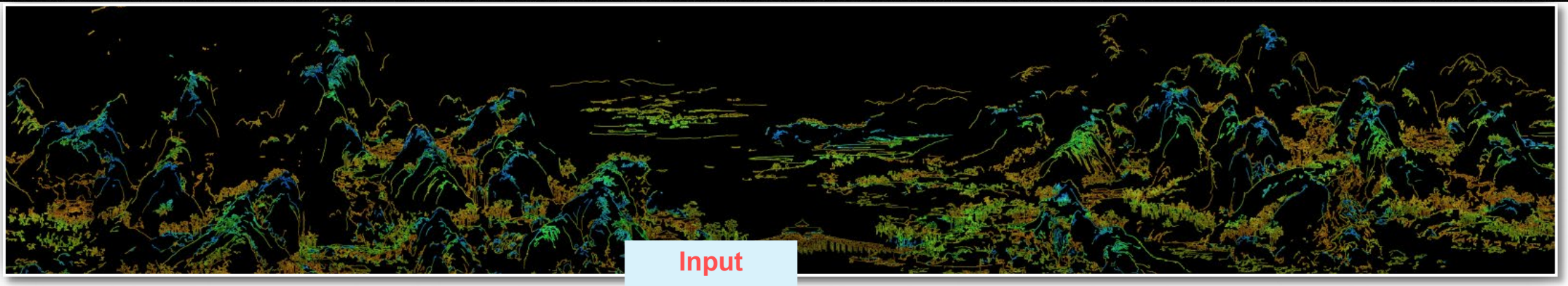
LAPLACE'S EQUATION



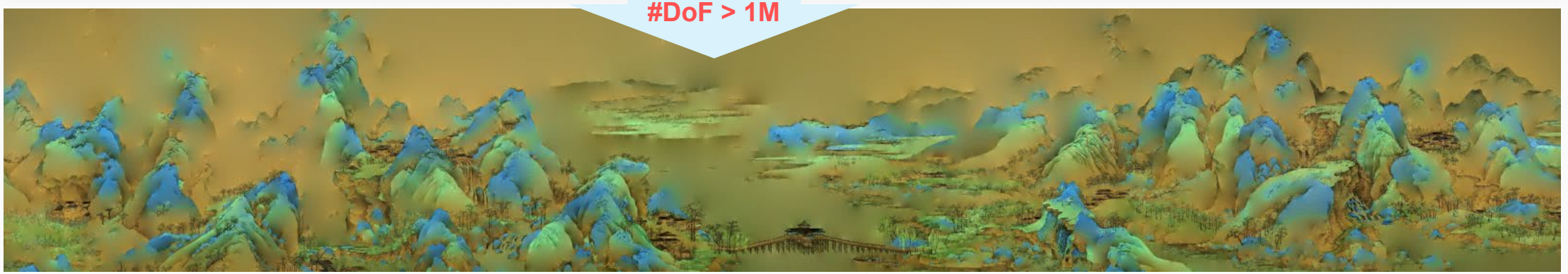
LAPLACE'S EQUATION

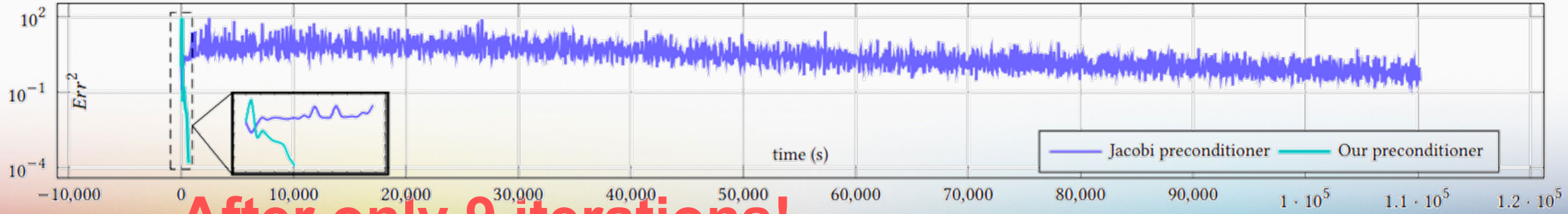


LAPLACE'S EQUATION



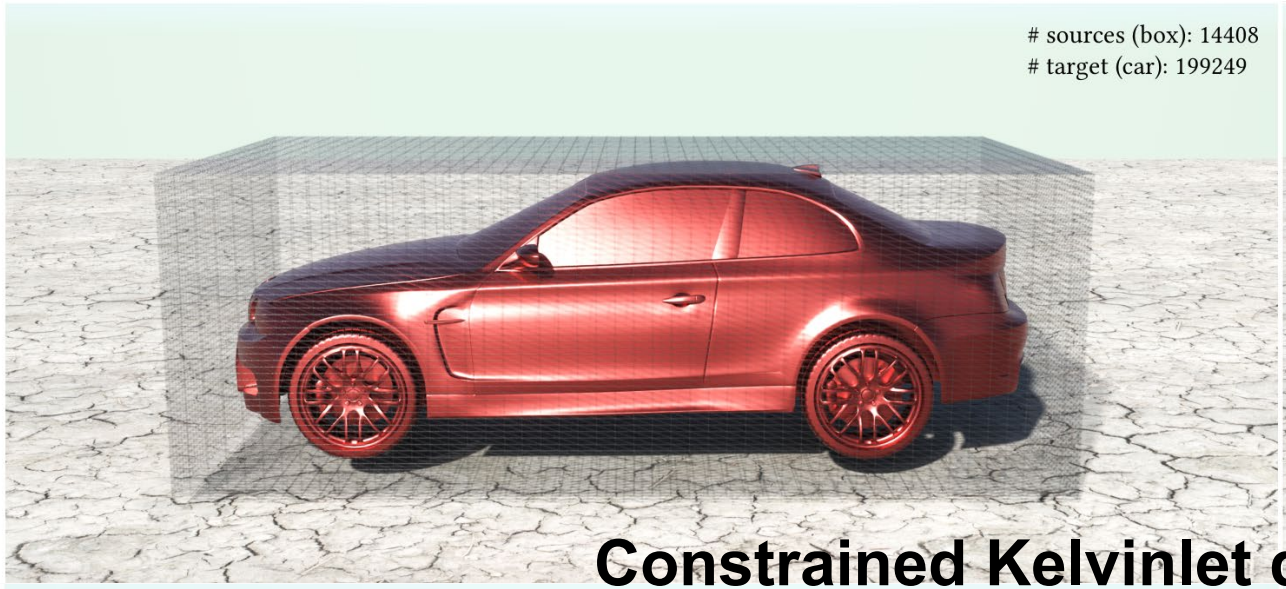
Input
#DoF > 1M

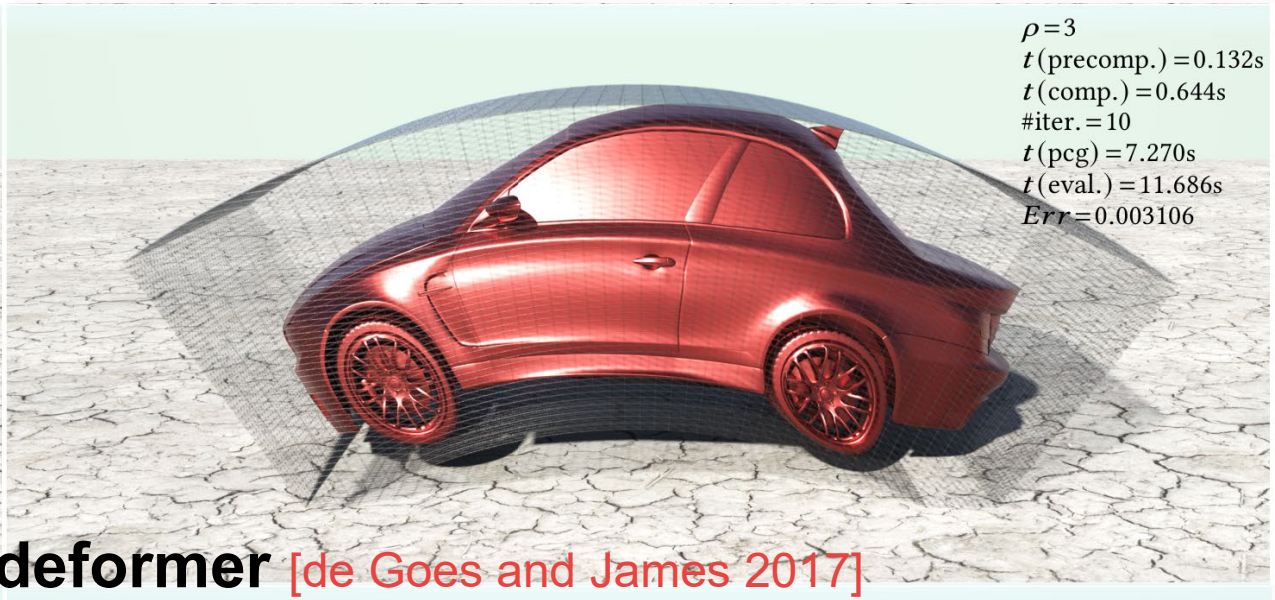




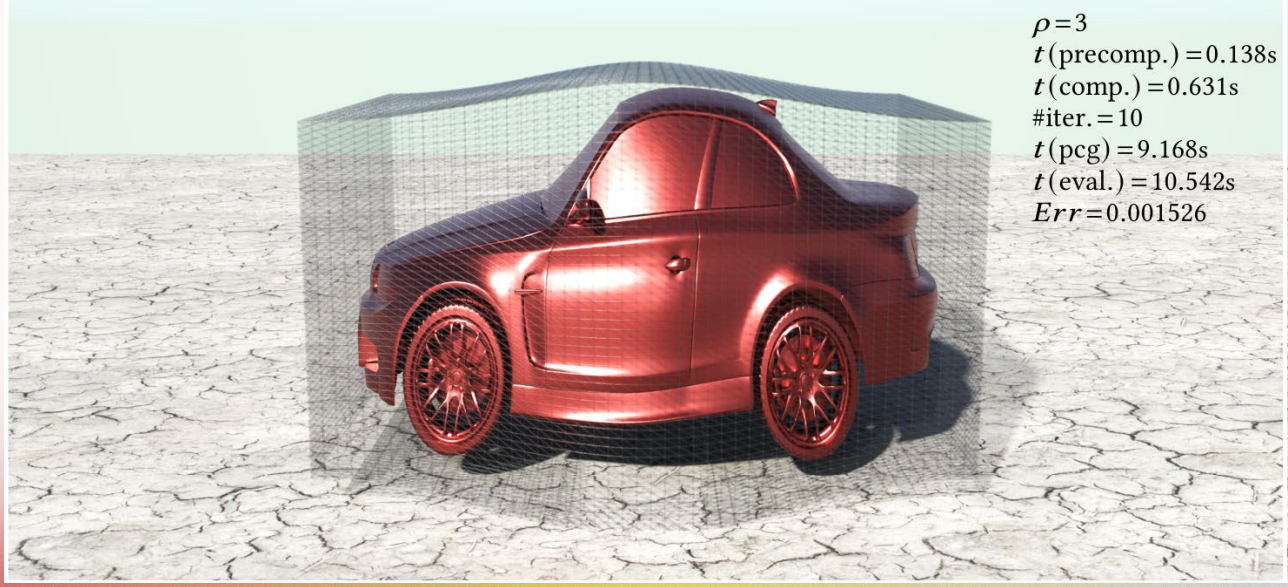
After only 9 iterations!

LINEAR ELASTICITY

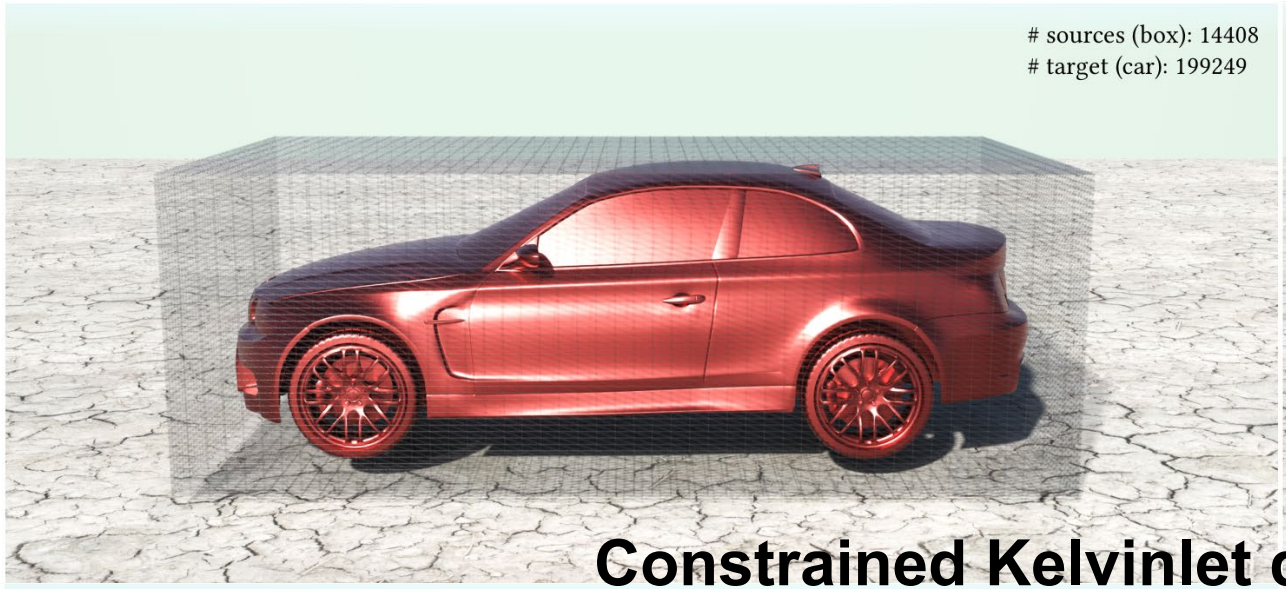


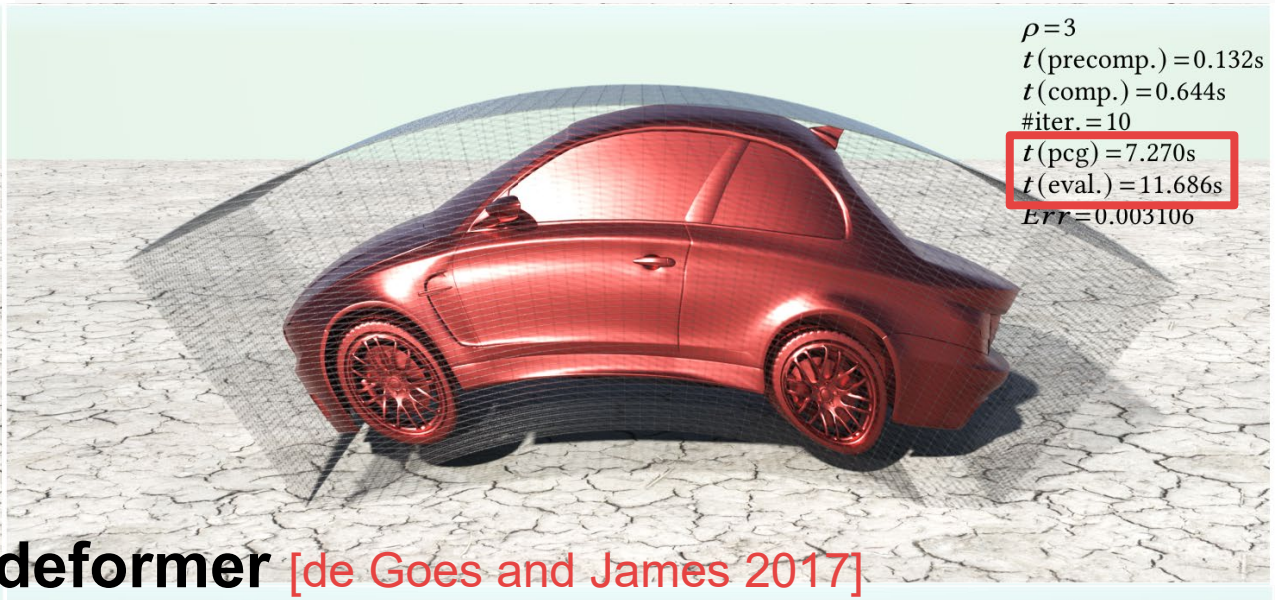


Constrained Kelvinlet deformer [de Goes and James 2017]

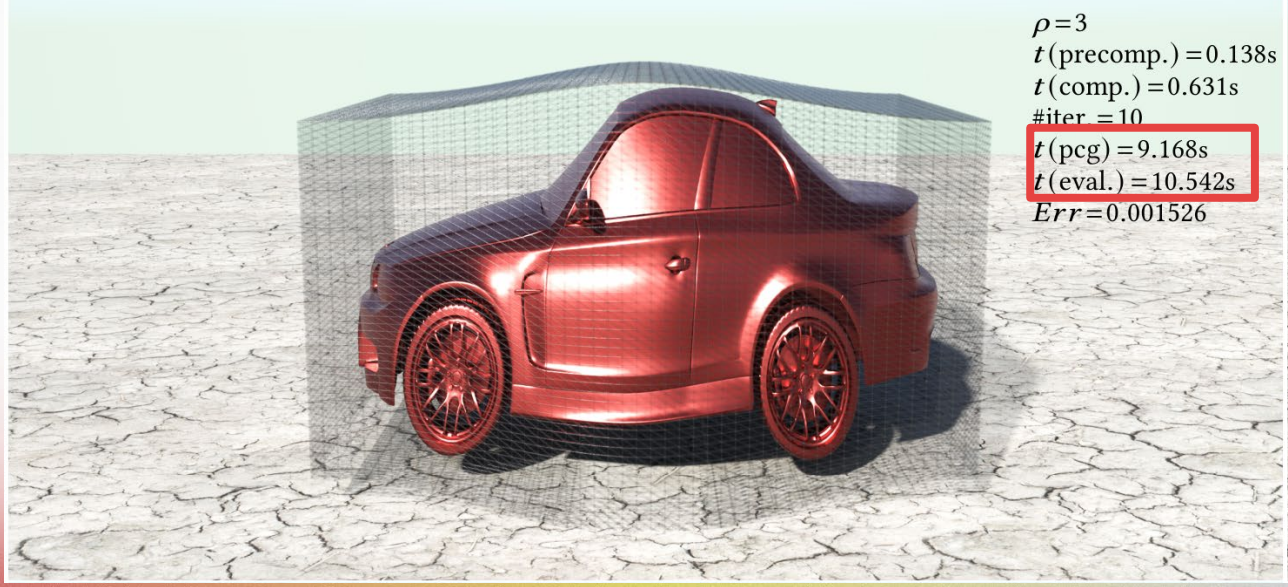


LINEAR ELASTICITY

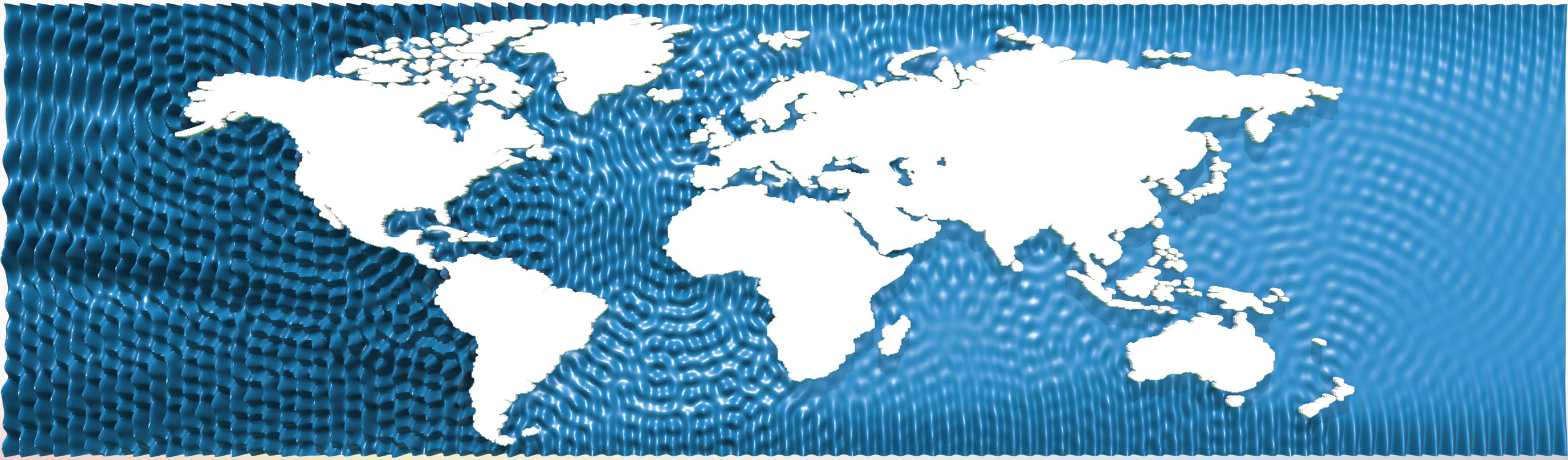




Constrained Kelvinlet deformer [de Goes and James 2017]

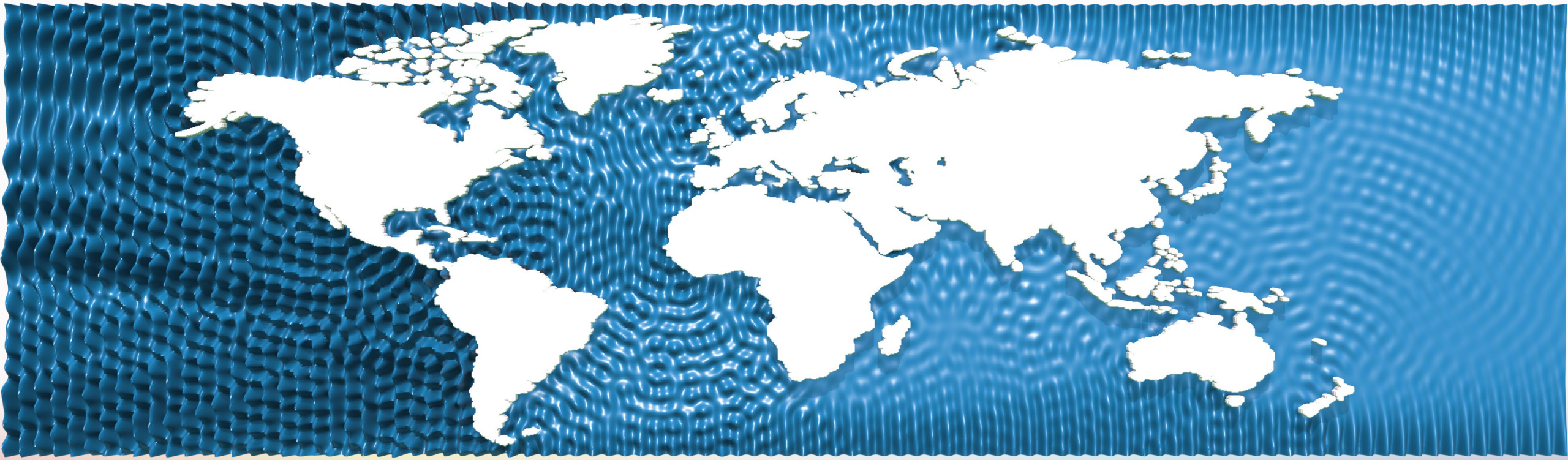


HELMHOLTZ EQUATION



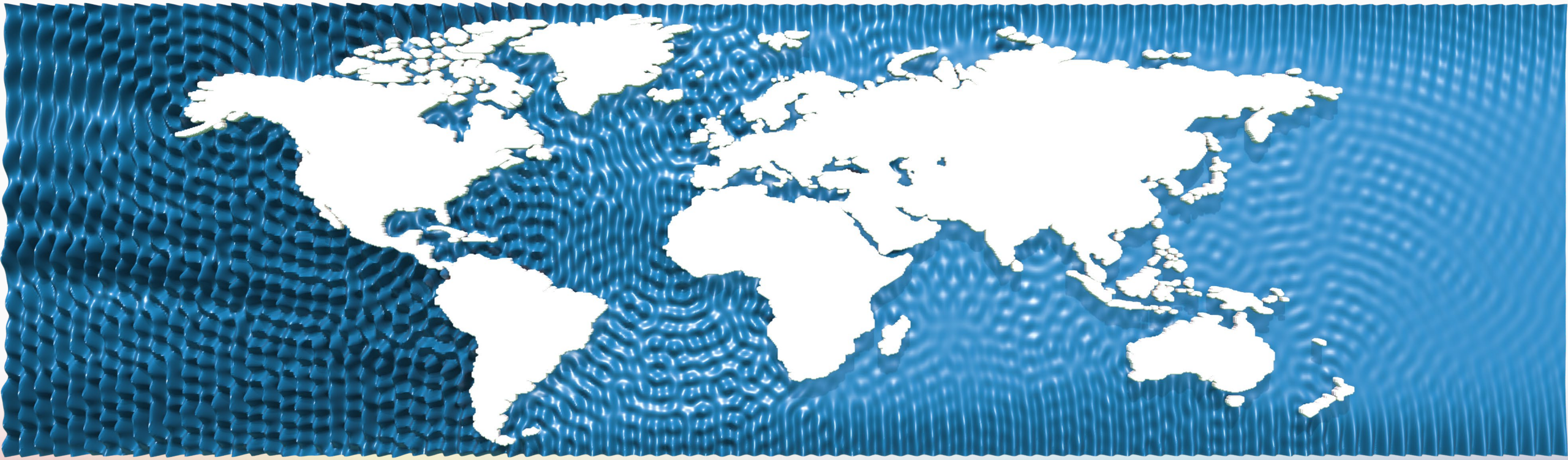
HELMHOLTZ EQUATION

- Least-squares solves are always needed for Helmholtz equations
  - The BIE/MFS matrices are complex symmetric, but not Hermitian



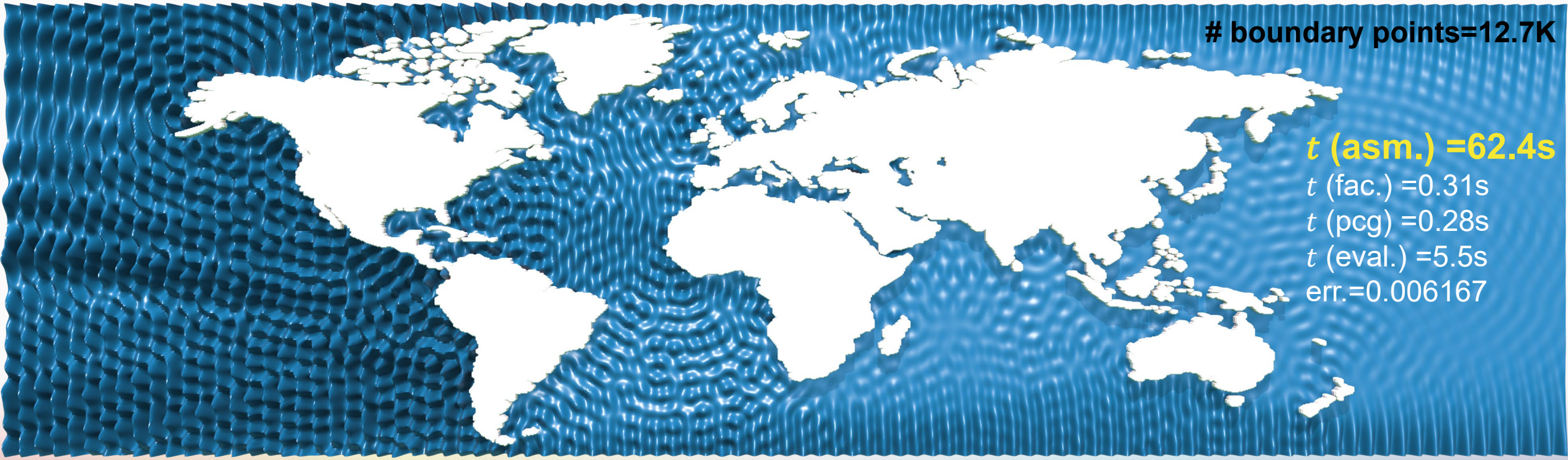
HELMHOLTZ EQUATION

- Least-squares solves are always needed for Helmholtz equations
  - The BIE/MFS matrices are complex symmetric, but not Hermitian
  - Cholesky factorization does no exist



HELMHOLTZ EQUATION

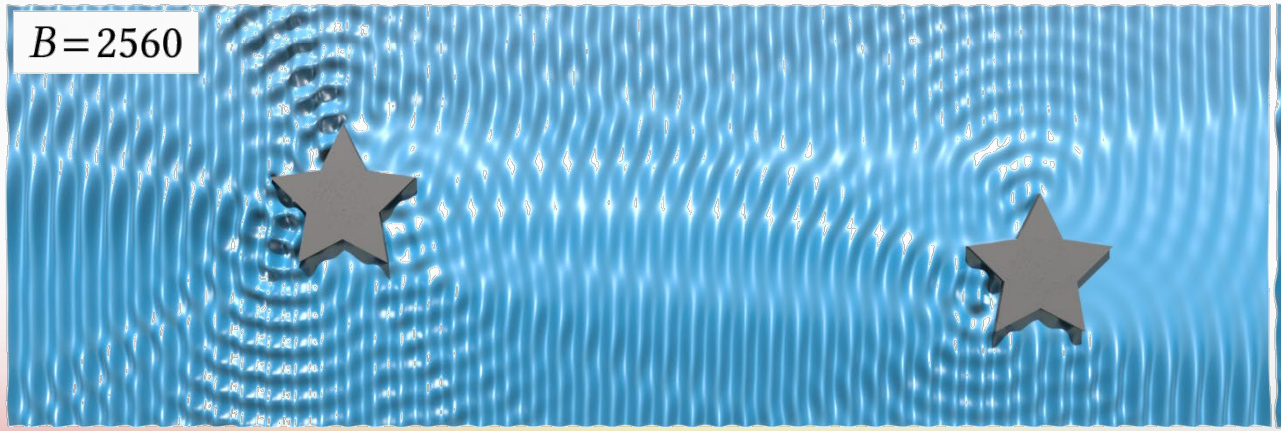
- Least-squares solves are always needed for Helmholtz equations
  - The BIE/MFS matrices are complex symmetric, but not Hermitian
  - Cholesky factorization does no exist

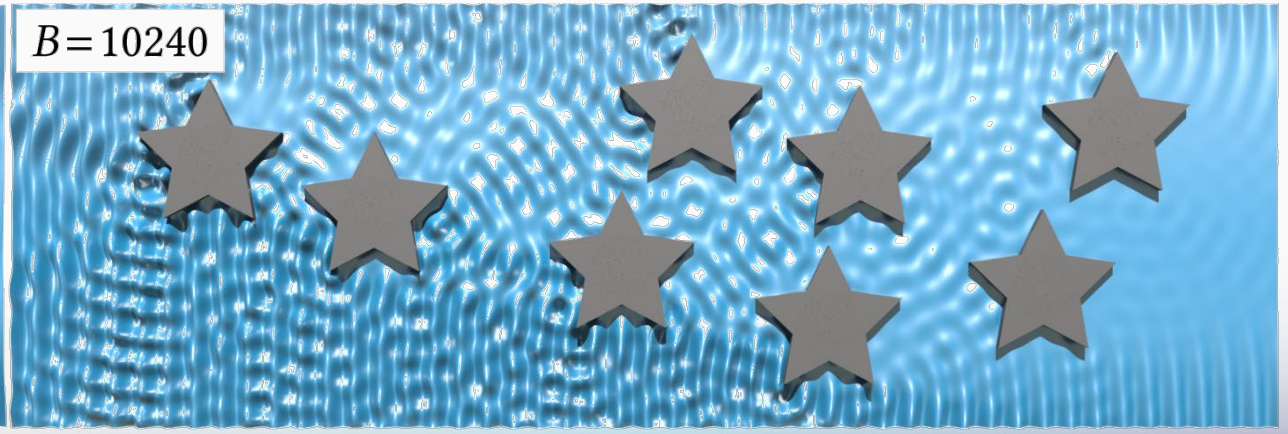


COMPARISON WITH SVD

Boundary size

B	SVD		Ours					
	t(fac.)	t(slv.)	t(precomp.)	t(comp.)	#iters	t(pcg)	t(total)	Err
1280	4.864	0.003	0.004	0.419	15	0.005	0.427	0.000706
2560	33.757	0.011	0.007	0.715	15	0.013	0.735	0.000679
5120	261.454	0.045	0.013	1.270	15	0.048	1.331	0.004405
7680	911.212	0.156	0.023	3.478	15	0.099	3.600	0.003497
10240	2405.59	0.303	0.032	7.170	15	0.167	7.369	0.003665

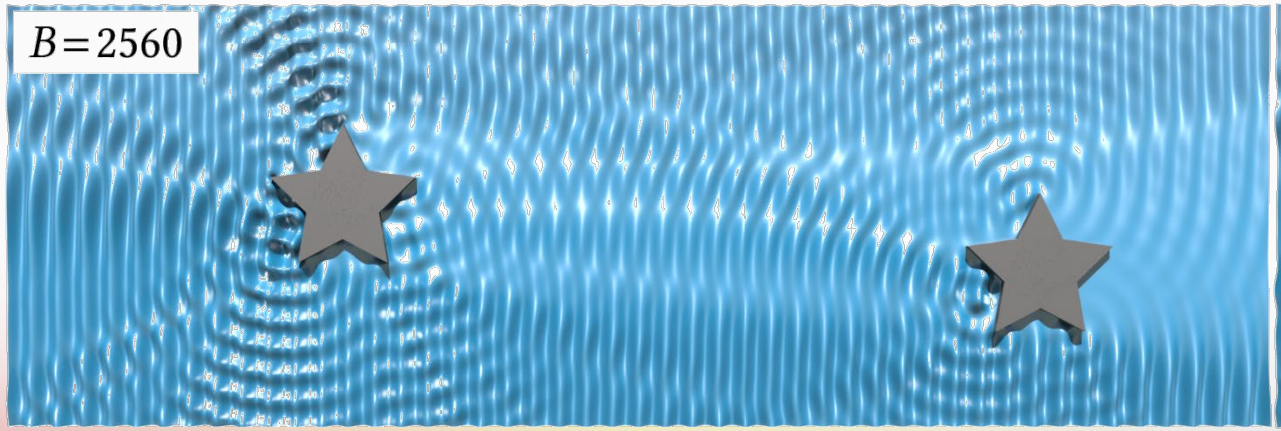


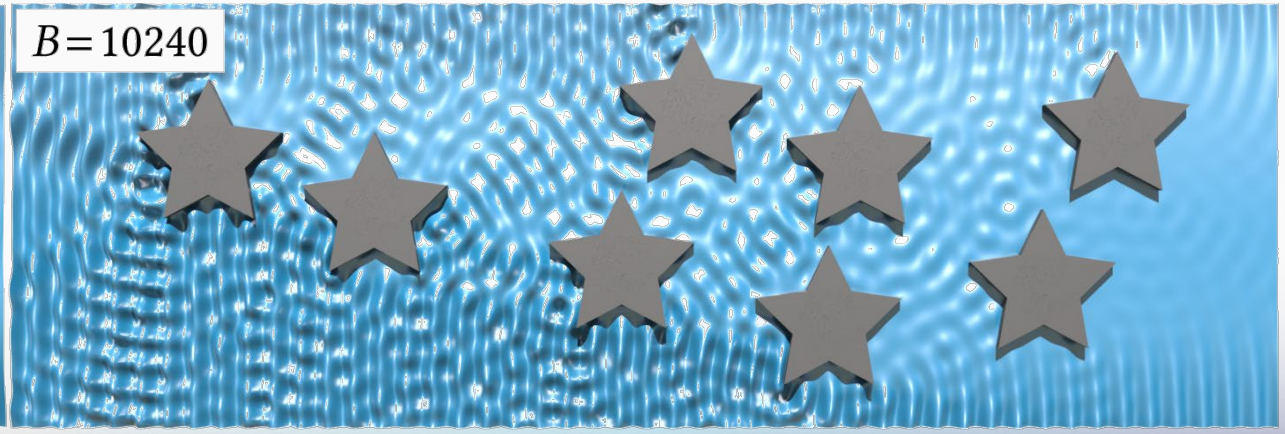


COMPARISON WITH SVD

Boundary size

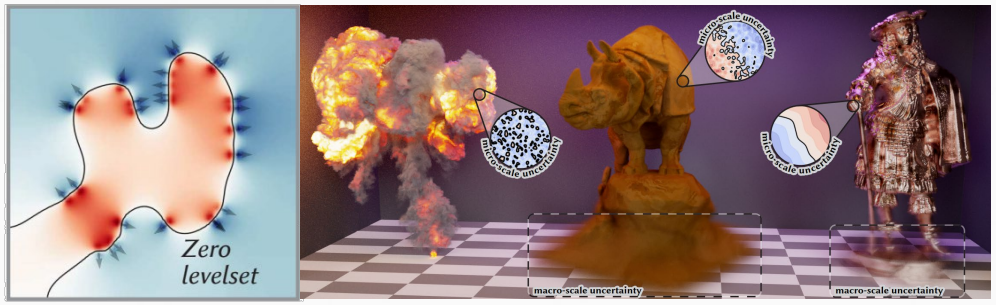
B	SVD		Ours					
	t(fac.)	t(slv.)	t(precomp.)	t(comp.)	#iters	t(pcg)	t(total)	Err
1280	4.864	0.003	0.004	0.419	15	0.005	0.427	0.000706
2560	33.757	0.011	0.007	0.715	15	0.013	0.735	0.000679
5120	261.454	0.045	0.013	1.270	15	0.048	1.331	0.004405
7680	911.212	0.156	0.023	3.478	15	0.099	3.600	0.003497
10240	2405.59	0.303	0.032	7.170	15	0.167	7.369	0.003665





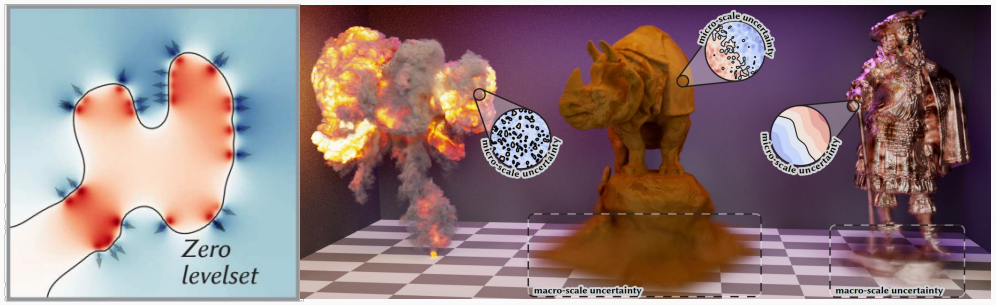
BEM FROM GAUSSIAN PROCESS VIEWPOINT

- Formulate stochasticity in Computer Graphics
  - Geometry processing, e.g., surface reconstruction [Sellán and Jacobson 2022]
  - Rendering, e.g., light transport [Seyb et al. 2024]



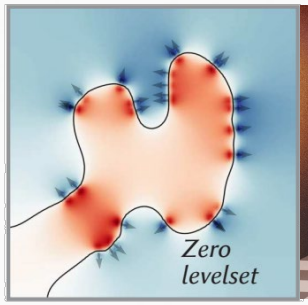
BEM FROM GAUSSIAN PROCESS VIEWPOINT

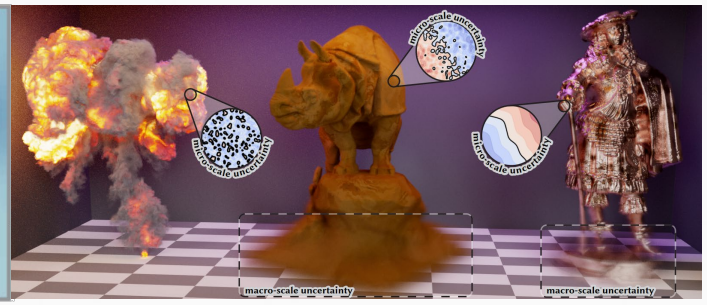
- Formulate stochasticity in Computer Graphics
  - Geometry processing, e.g., surface reconstruction [Sellán and Jacobson 2022]
  - Rendering, e.g., light transport [Seyb et al. 2024]
- Boundary value problems from a statistical point of view
  - Investigate the distribution of all possible solutions, not just a single one!



BEM FROM GAUSSIAN PROCESS VIEWPOINT

- Formulate stochasticity in Computer Graphics
  - Geometry processing, e.g., surface reconstruction [Sellán and Jacobson 2022]
  - Rendering, e.g., light transport [Seyb et al. 2024]
- Boundary value problems from a statistical point of view
  - Investigate the distribution of all possible solutions, not just a single one!
- Gaussian-process based inference *v.s.* MFS





Gaussian Process	MFS
Kernel function	Green's function
Observation	Boundary condition
Conditional mean	Solution
Prediction	Evaluation

BEM FROM GAUSSIAN PROCESS VIEWPOINT

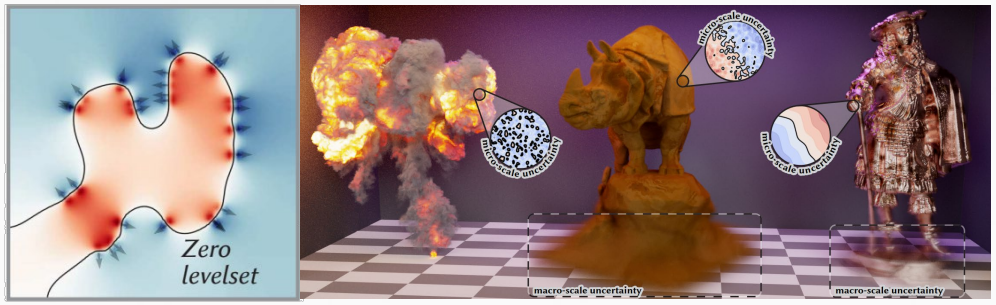
- Formulate stochasticity in Computer Graphics
  - Geometry processing, e.g., surface reconstruction [Sellán and Jacobson 2022]
  - Rendering, e.g., light transport [Seyb et al. 2024]
- Boundary value problems from a statistical point of view
  - Investigate the distribution of all possible solutions, not just a single one!
- Gaussian-process based inference *v.s.* MFS
  - Beyond conditional mean

$$\mu\left(f(\boldsymbol{x}) \mid \boldsymbol{y}, f(\boldsymbol{y})\right) = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{K}(\boldsymbol{y}, \boldsymbol{y})^{-1} f(\boldsymbol{y}),$$

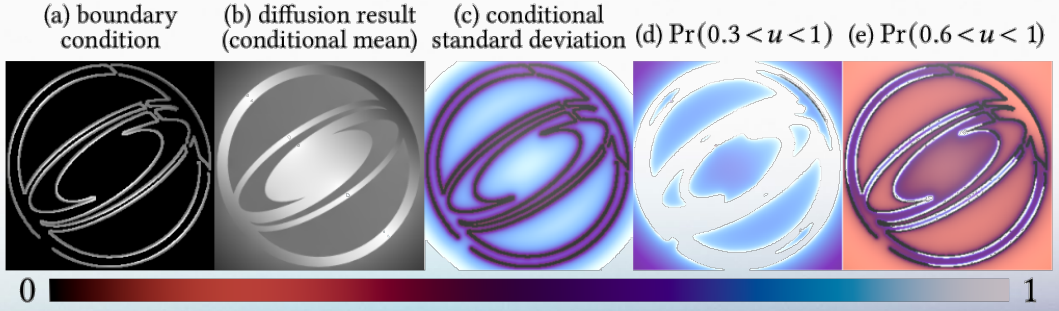
  - Conditional variance for uncertainty quantification

$$\sigma^2_{\boldsymbol{y}_i} = \boldsymbol{K}(\boldsymbol{y}_i, \boldsymbol{y}_i) - \boldsymbol{K}(\boldsymbol{y}_i, \boldsymbol{x})\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x})^{-1}\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}_i).$$

  - Tell the probability of the solution falling within a given range



Gaussian Process	MFS
Kernel function	Green's function
Observation	Boundary condition
Conditional mean	Solution
Prediction	Evaluation



Uncertainty quantification of BIE solves

FUTURE WORKS

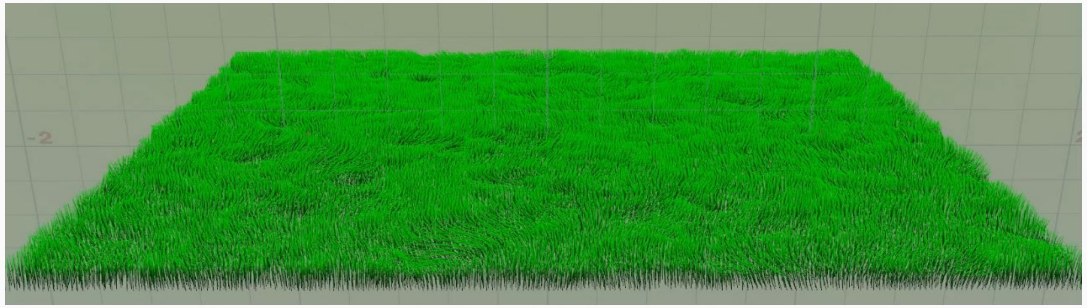
- Generalize the idea to …
  - Asymmetric systems from elliptic PDEs or non-elliptic PDEs without least-squares solves, e.g., wave equations [Schreck et al. 2019]
  - Nonlinear problems, e.g., Gaussian process hydrodynamics [Owhadi 2023]

FUTURE WORKS

- Generalize the idea to …
  - Asymmetric systems from elliptic PDEs or non-elliptic PDEs without least-squares solves, e.g., wave equations [Schreck et al. 2019]
  - Nonlinear problems, e.g., Gaussian process hydrodynamics [Owhadi 2023]
- Strategies for further speedup
  - Problem-adapted FMM instead of “black-box” FMM
  - Faster matrix-vector product, e.g., H-matrix evaluation

FUTURE WORKS

- Generalize the idea to …
  - Asymmetric systems from elliptic PDEs or non-elliptic PDEs without least-squares solves, e.g., wave equations [Schreck et al. 2019]
  - Nonlinear problems, e.g., Gaussian process hydrodynamics [Owhadi 2023]
- Strategies for further speedup
  - Problem-adapted FMM instead of “black-box” FMM
  - Faster matrix-vector product, e.g., H-matrix evaluation
- Simulation meets stochasticity
  - Make use of uncertainty quantification for adaptive simulation
  - Develop stochastic representation to account for the uncertainty of a complex dynamical system




SIGGRAPH 2024
DENVER+ 28 JUL — 1 AUG
THANKS!