Solving PDEs on surfaces

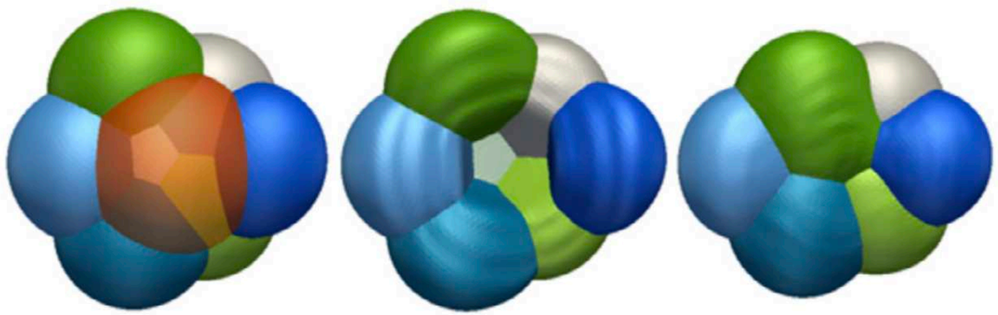
Dan Fortunato

FLATIRON INSTITUTE
Center for Computational Mathematics

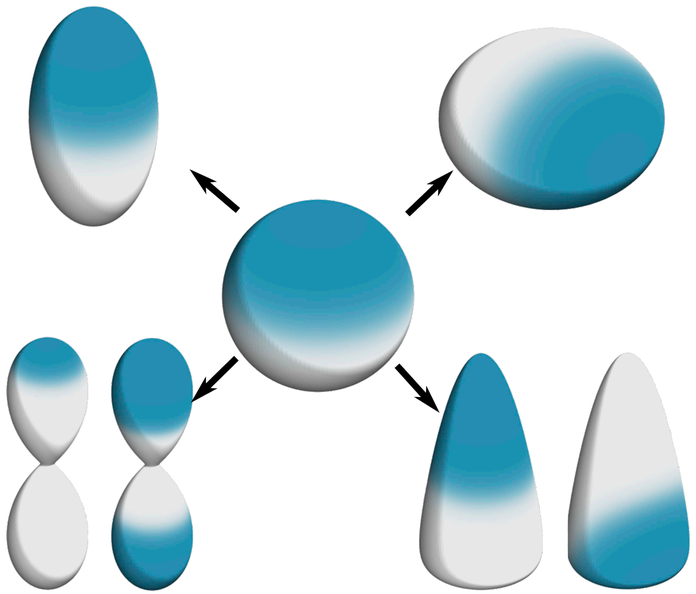
Introduction
Surface PDEs

Surface-bound phenomena arise in many applications.

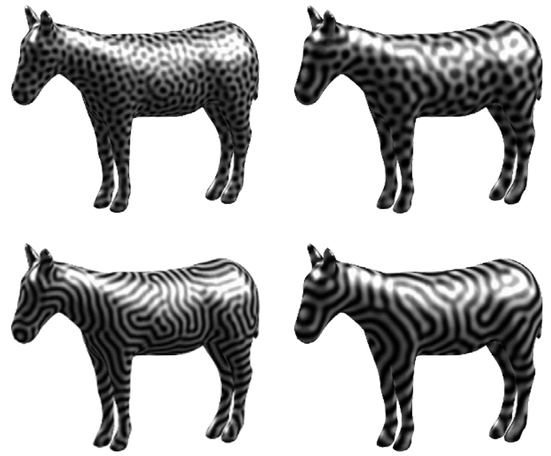
Thin-film hydrodynamics [Saye, 2016]



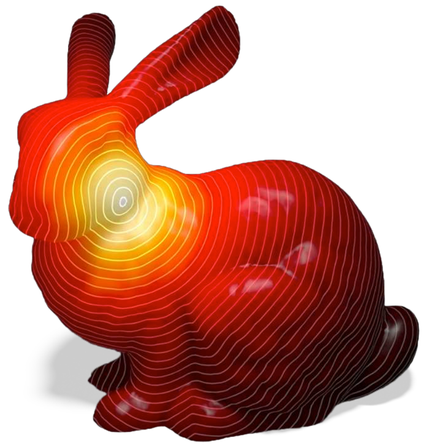
Cell polarization [Miller, F., Muratov, Greengard, & Shvartsman, 2022]



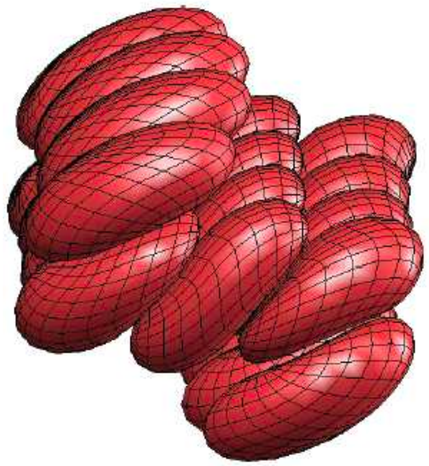
Pattern formation [Jeong, 2017]



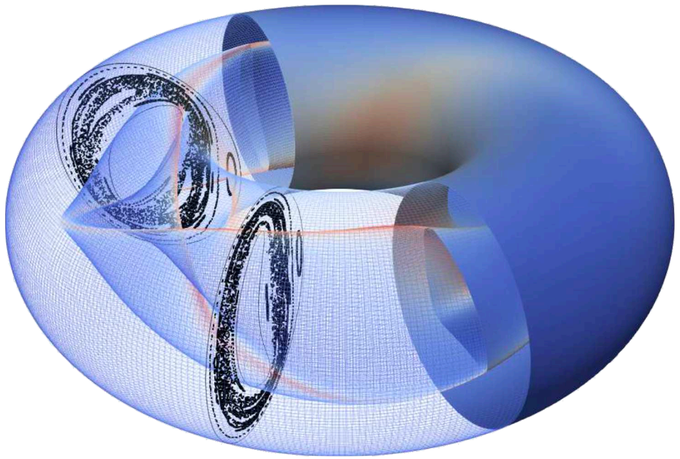
Geodesic distance [Crane et al., 2017]



Vesicle flows [Veerapaneni et al., 2011]



Stellarator design [Malhotra et al., 2019]



Introduction
Surface PDEs

Surface PDEs describe the dynamics of such phenomena.

Steady-state problem

$$\mathcal{L}_\Gamma u(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma$$

- Laplace–Beltrami
- convection–diffusion
- steady Stokes

Time-dependent problem

$$\frac{\partial u}{\partial t} = \underbrace{\mathcal{L}_\Gamma u}_{\text{Linear}} + \underbrace{\mathcal{N}(u)}_{\text{Nonlinear}} \text{ on } \Gamma$$

- reaction–diffusion
- heat
- Navier–Stokes

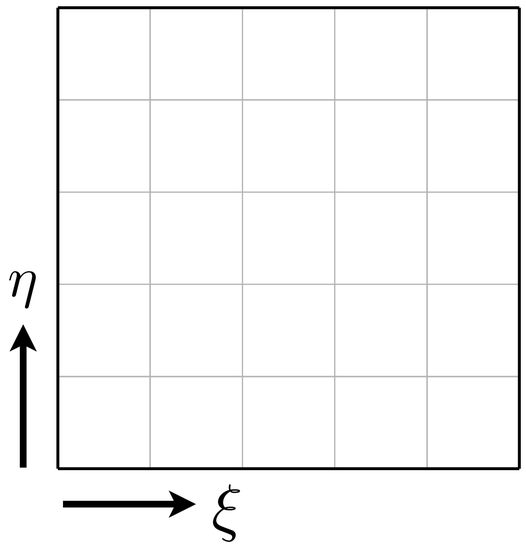
Implicit time discretization:

$$(I - \Delta t \mathcal{L}_\Gamma)\, u^{k+1} = u^k + \Delta t \mathcal{N}(u^k)$$

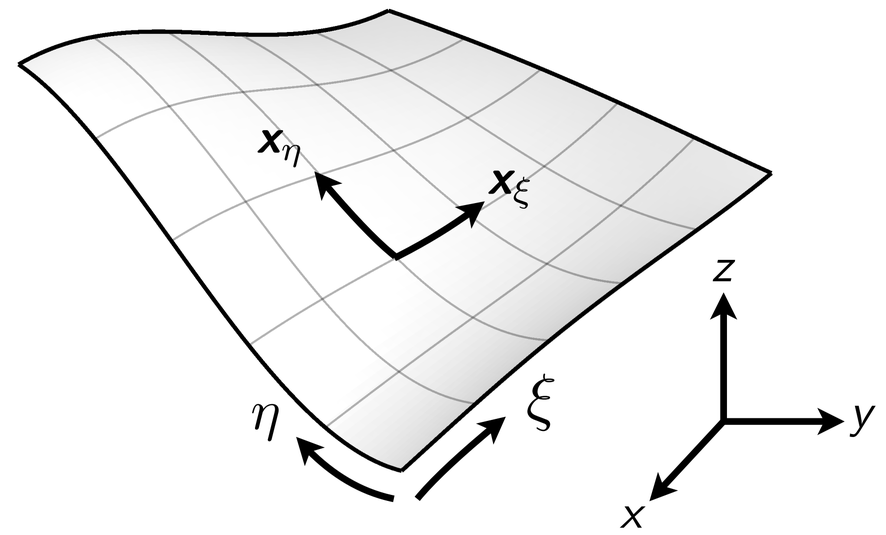
Model surface PDE: $\nabla_\Gamma \cdot (\boldsymbol{A}(\boldsymbol{x}) \nabla_\Gamma u(\boldsymbol{x})) + \nabla_\Gamma \cdot (\boldsymbol{b}(\boldsymbol{x}) u(\boldsymbol{x})) + c(\boldsymbol{x}) u(\boldsymbol{x}) = f(\boldsymbol{x})$

(+ BCs if surface is not closed)

Surface PDEs
Differential operators



$$\boldsymbol{x}(\xi,\eta) = \begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \\ z(\xi,\eta) \end{bmatrix} : \mathbb{R}^2 \to \mathbb{R}^3$$

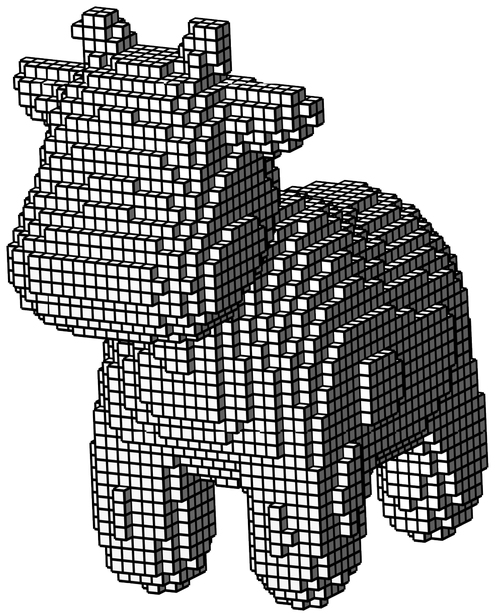


- Metric tensor $g = \begin{bmatrix} \boldsymbol{x}_\xi \cdot \boldsymbol{x}_\xi & \boldsymbol{x}_\xi \cdot \boldsymbol{x}_\eta \\ \boldsymbol{x}_\eta \cdot \boldsymbol{x}_\xi & \boldsymbol{x}_\eta \cdot \boldsymbol{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.
- Surface gradient: $\nabla_\Gamma u = [\boldsymbol{x}_\xi \;\; \boldsymbol{x}_\eta] \; g^{-1} \nabla_{\xi\eta} u$ $\qquad \partial_x^\Gamma = \boldsymbol{e}_x \cdot \nabla_\Gamma$
- Surface divergence: $\nabla_\Gamma \cdot \boldsymbol{u} = \dfrac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot \left(\sqrt{\det g}\, \boldsymbol{u} \right)$ $\qquad \partial_y^\Gamma = \boldsymbol{e}_y \cdot \nabla_\Gamma$
- Laplace–Beltrami: $\Delta_\Gamma u = \nabla_\Gamma \cdot \nabla_\Gamma u = \dfrac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot \left(\sqrt{\det g}\, g^{-1} \nabla_{\xi\eta} u \right)$ $\qquad \partial_z^\Gamma = \boldsymbol{e}_z \cdot \nabla_\Gamma$

Surface representation

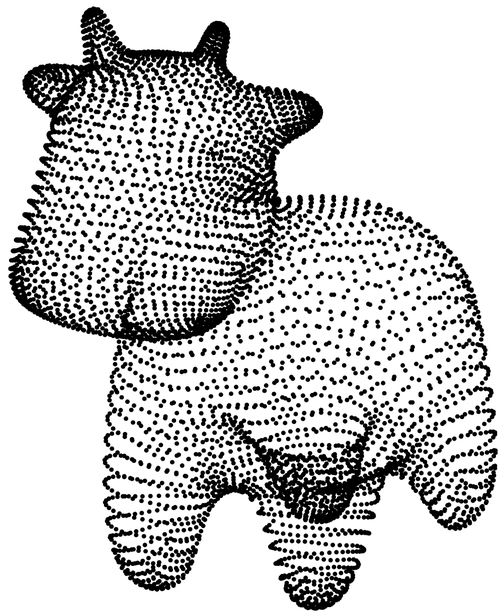
Many ways to represent a surface

Embedded grid



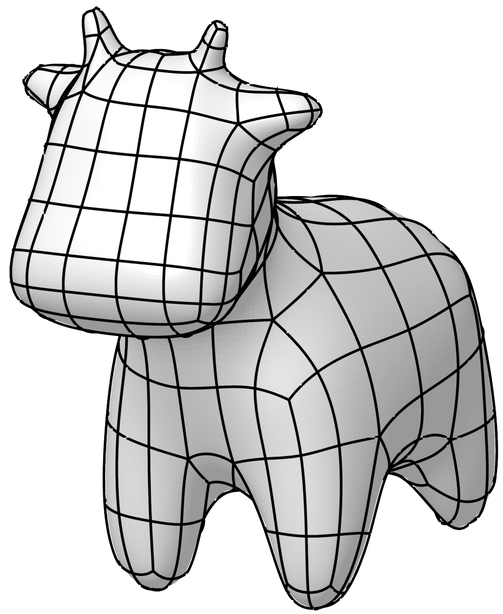
- Closest point methods
  (Macdonald, Greer, Ruuth, …)
- Level set methods
  (Sethian, Osher, …)

Unstructured point cloud



- Radial basis functions
  (Fornberg, Piret, Wendland, Wright, …)

Surface mesh

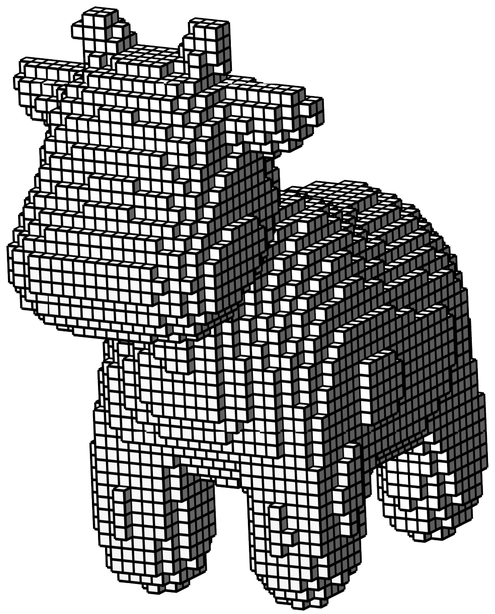


- Finite element methods
  (Dziuk, Elliott, …)
- Integral equation methods
  (O'Neil, Goodwill, Rachh, …)

Surface representation

Many ways to represent a surface

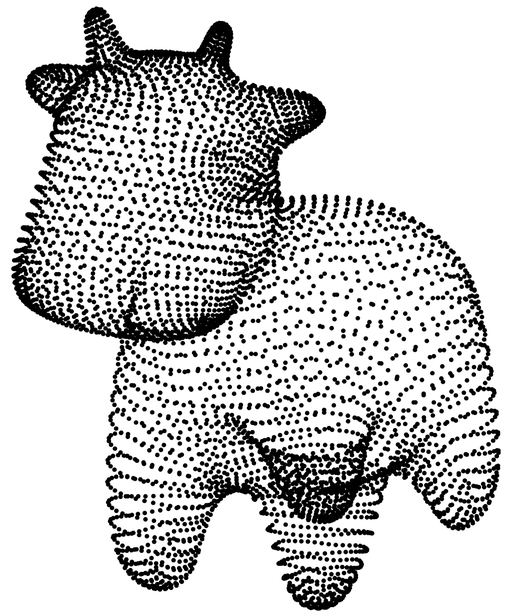
Embedded grid



- Closest point methods
  (Macdonald, Greer, Ruuth, …)
- Level set methods
  (Sethian, Osher, …)

high order → more volume DoFs

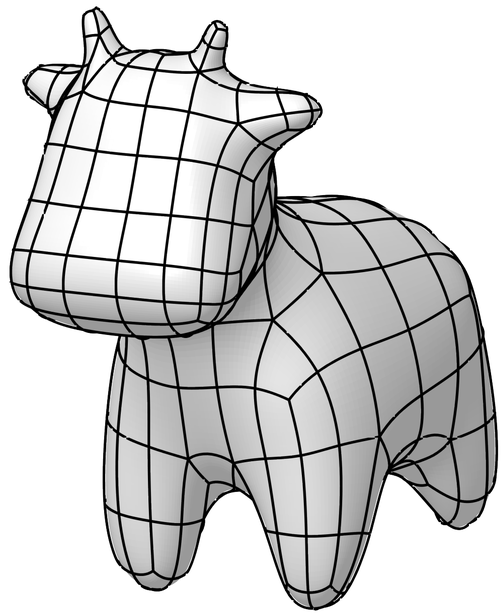
Unstructured point cloud



- Radial basis functions
  (Fornberg, Piret, Wendland, Wright, …)

high order → ill-conditioned

Surface mesh



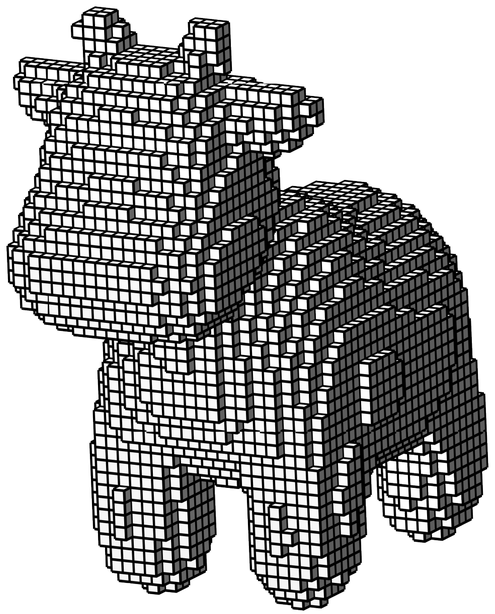
- Finite element methods
  (Dziuk, Elliott, …)
- Integral equation methods
  (O'Neil, Goodwill, Rachh, …)

high order → dense fill-in

Surface representation

Many ways to represent a surface

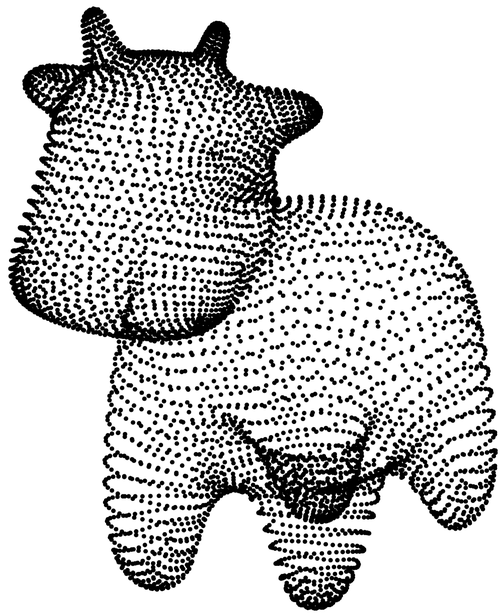
Embedded grid



- Closest point methods
  (Macdonald, Greer, Ruuth, …)
- Level set methods
  (Sethian, Osher, …)

high order → more volume DoFs

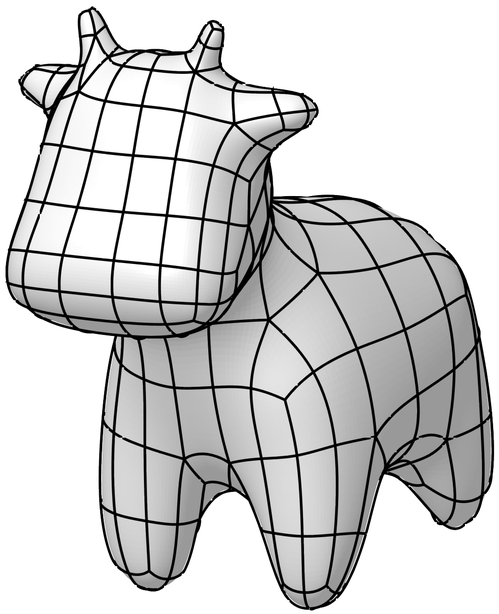
Unstructured point cloud



- Radial basis functions
  (Fornberg, Piret, Wendland, Wright, …)

high order → ill-conditioned

Surface mesh



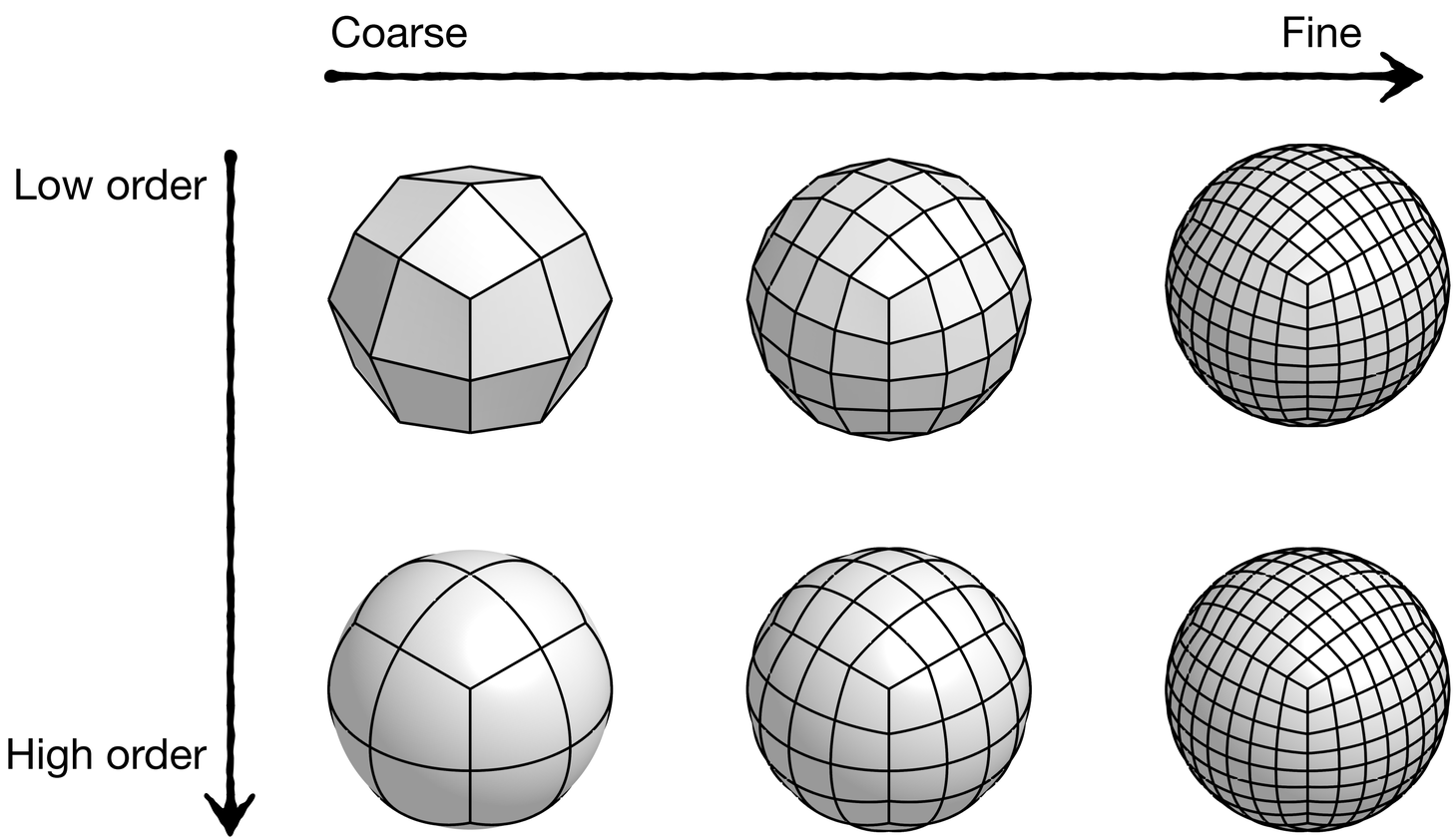
- Finite element methods
  (Dziuk, Elliott, …)
- Integral equation methods
  (O'Neil, Goodwill, Rachh, …)

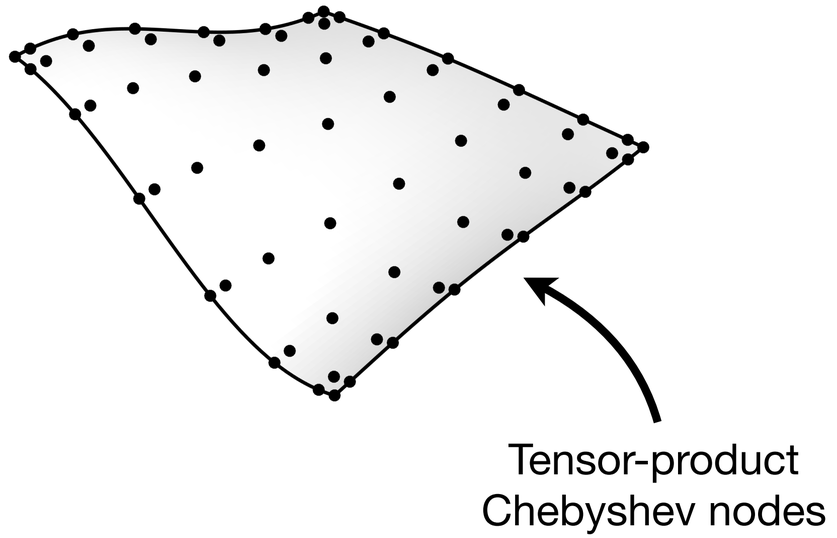
high order → dense fill-in

Developing efficient, high-order accurate solvers on surfaces is challenging!

Surface representation
Low-order vs. high-order

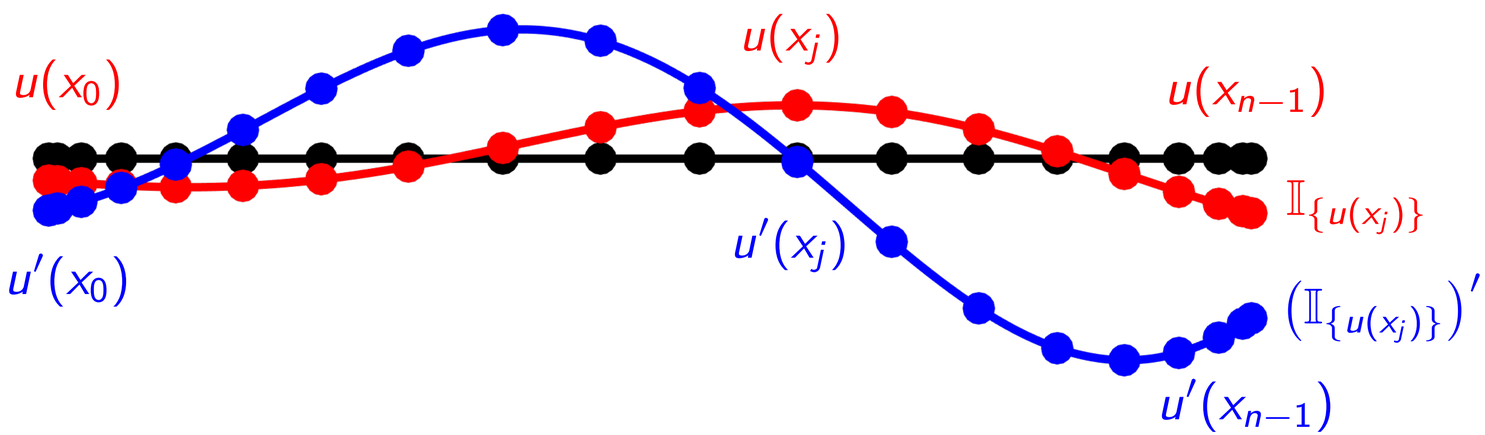
- Meshes are a good choice for CAD-compatibility. We use high-order quadrilateral patches.
- High-order elements allow faster convergence to solution.
- Coordinate maps of a patch are discretized via tabulation at Chebyshev nodes.





High-order discretization
Spectral collocation

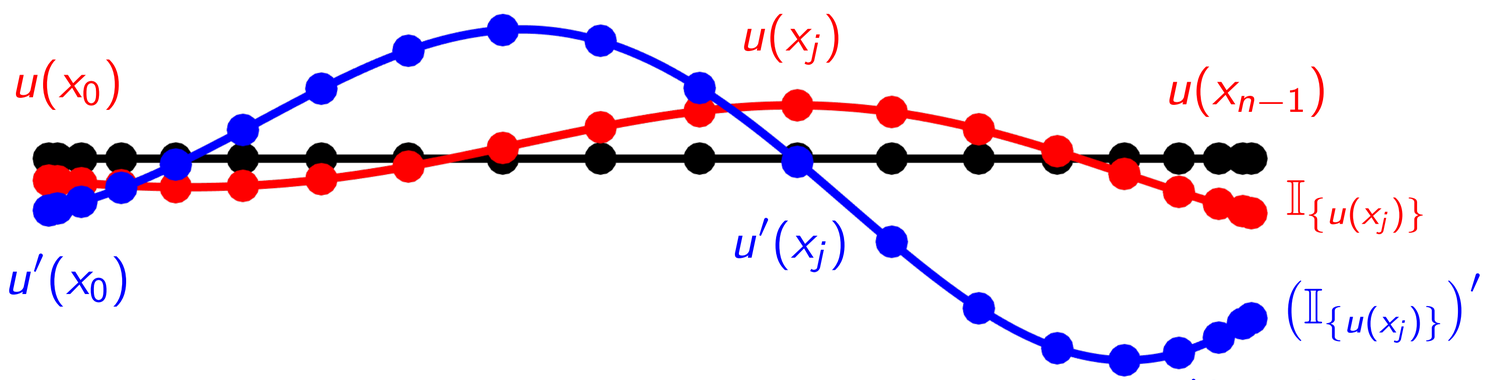
- Function values also stored at Chebyshev nodes.
- Derivatives and metric information (e.g. Jacobian) computed via spectral differentiation.



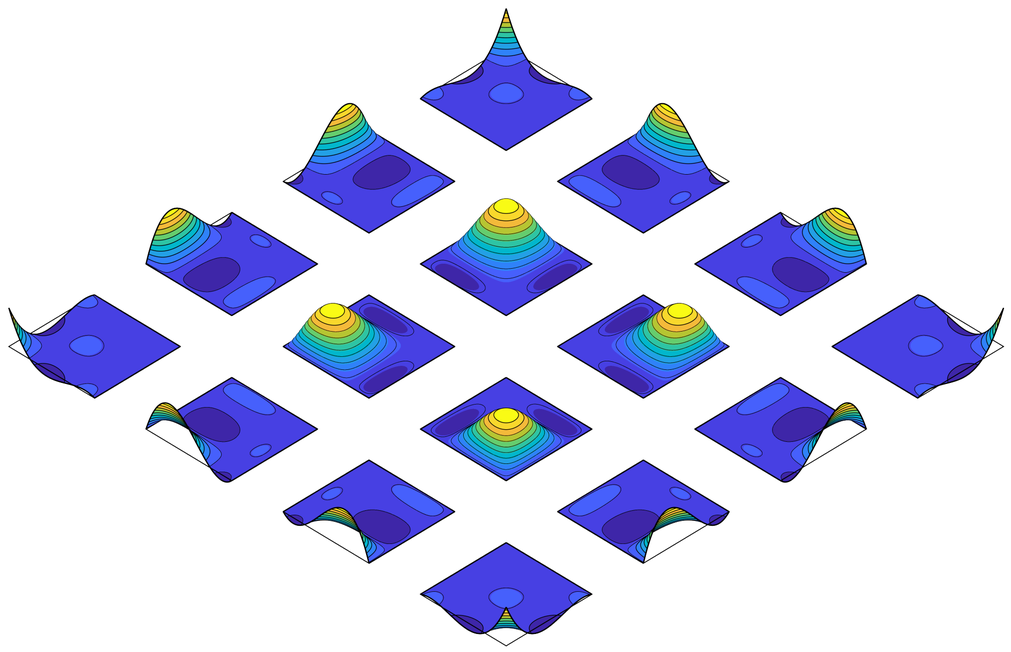
$$D\,u_j \mapsto u_j'$$

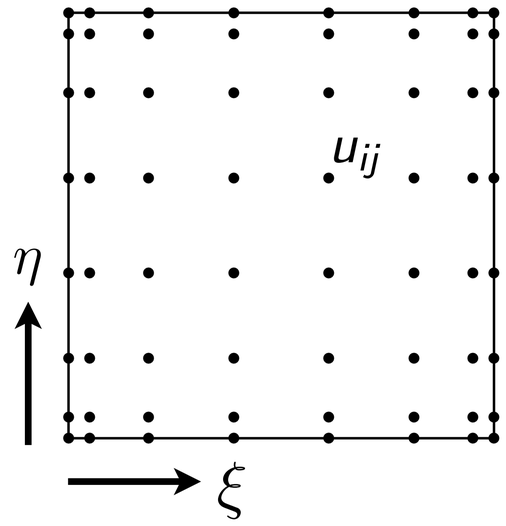
High-order discretization
Spectral collocation

- Function values also stored at Chebyshev nodes.
- Derivatives and metric information (e.g. Jacobian) computed via spectral differentiation.



$$D\,u_j \mapsto u_j'$$

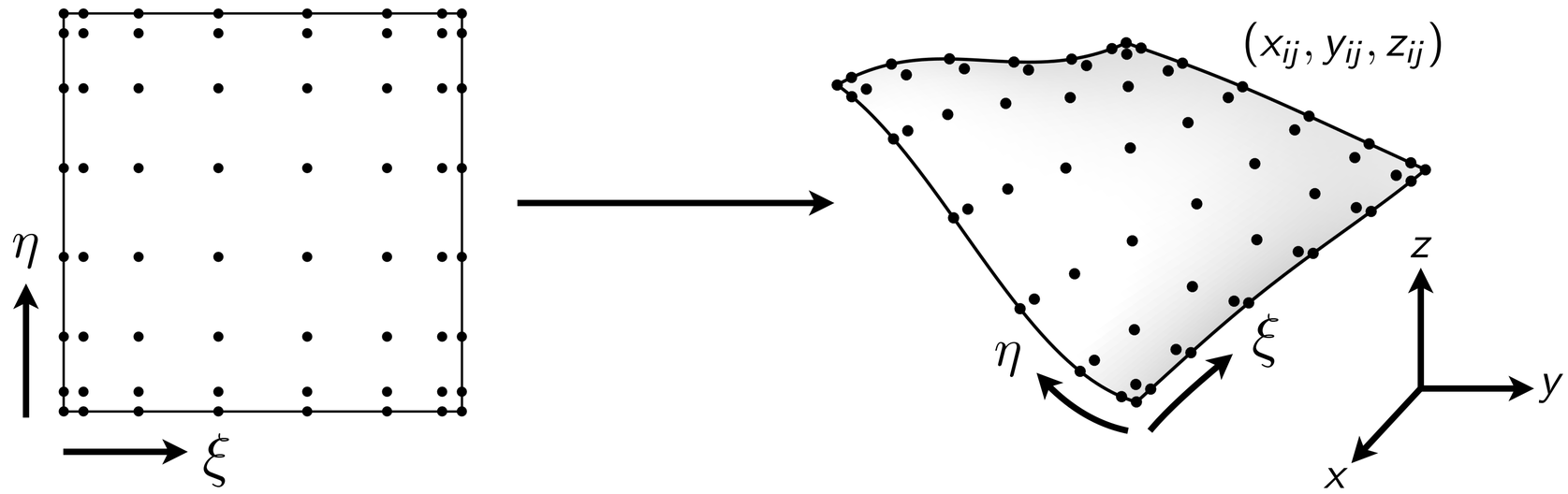




$$(D \otimes I)\,u_{ij} \mapsto \partial u_{ij}/\partial \xi$$
$$(I \otimes D)\,u_{ij} \mapsto \partial u_{ij}/\partial \eta$$

High-order discretization
Spectral collocation on a surface

- PDE is discretized through spectral differentiation and pointwise multiplication.

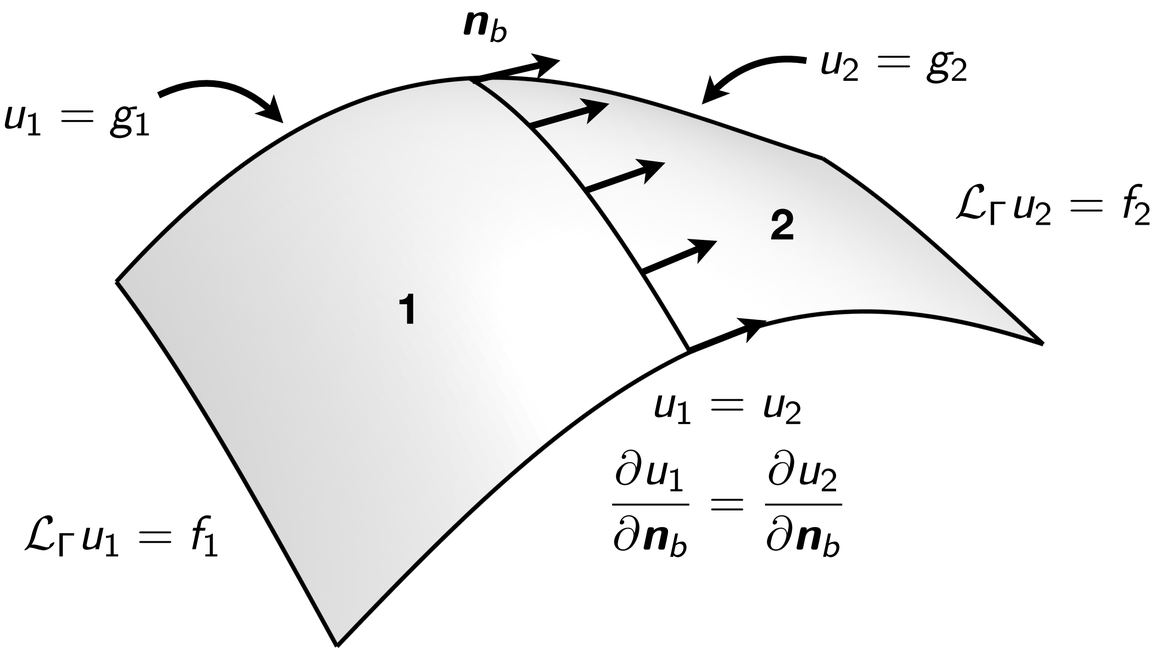


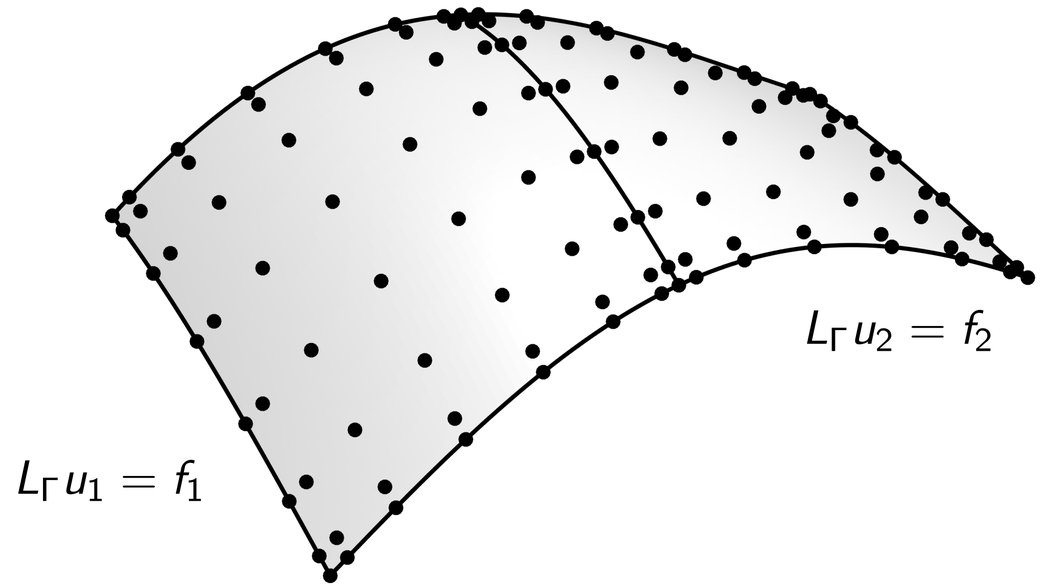
- For example, the discrete tangential x-derivative operator is:

$$D_x = \begin{bmatrix} \ddots & & \\ & (\xi_x)_{ij} & \\ & & \ddots \end{bmatrix} (D \otimes I) \; + \; \begin{bmatrix} \ddots & & \\ & (\eta_x)_{ij} & \\ & & \ddots \end{bmatrix} (I \otimes D)$$

- In general, the PDE results in a $(p+1)^2 \times (p+1)^2$ linear system, $L_\Gamma u = f$, which we can invert directly.

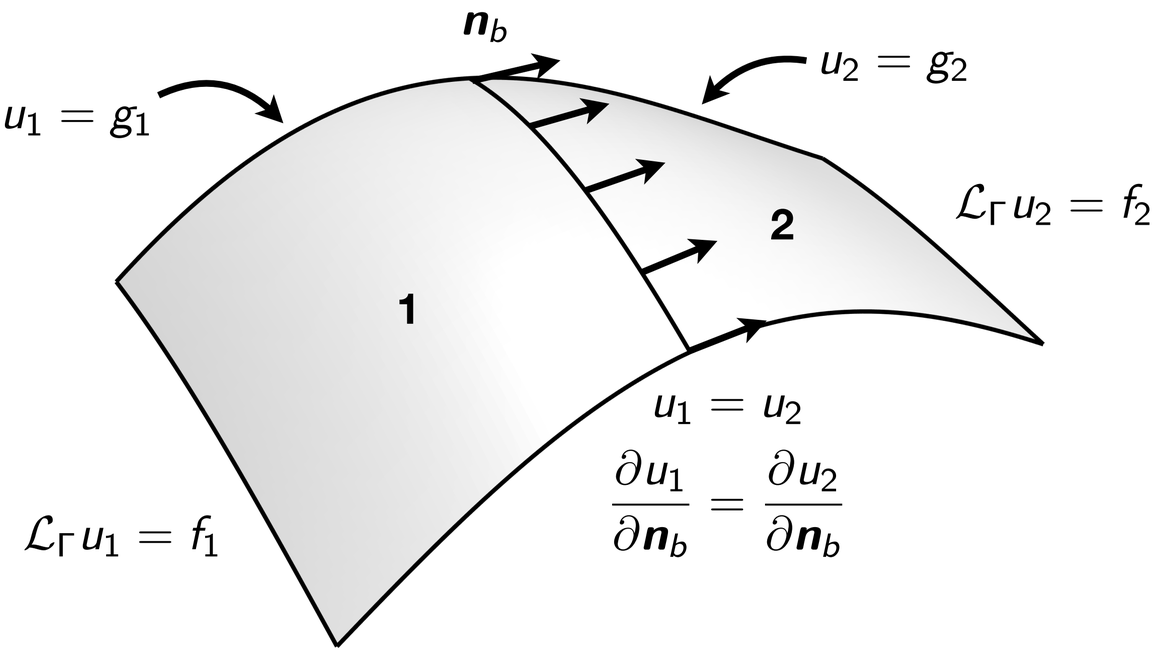
High-order discretization
Two glued patches

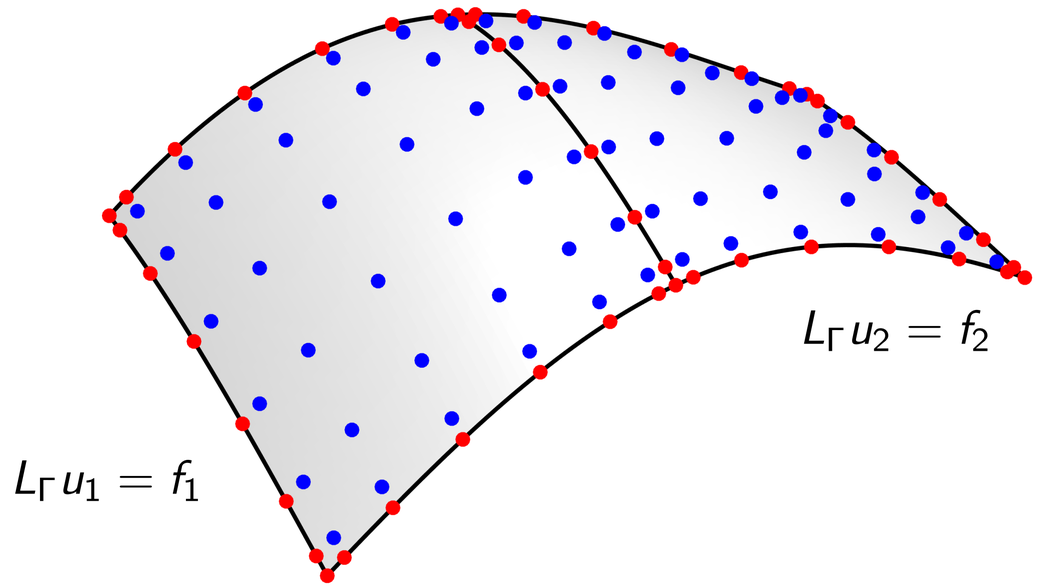




[F., 2022]

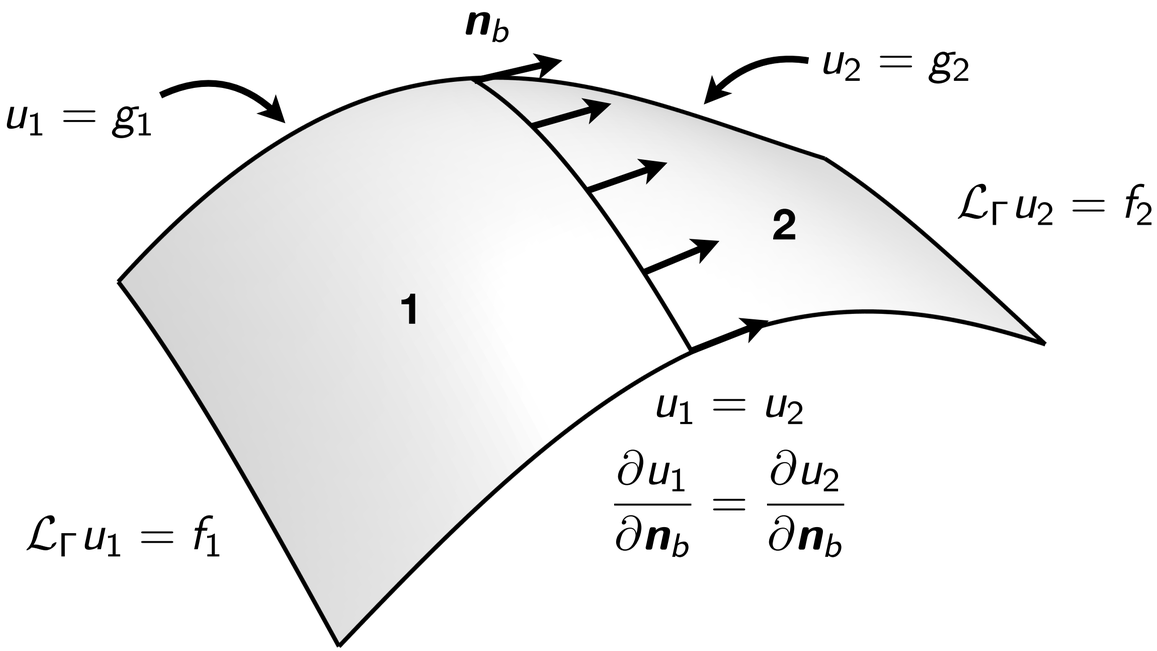
High-order discretization
Two glued patches

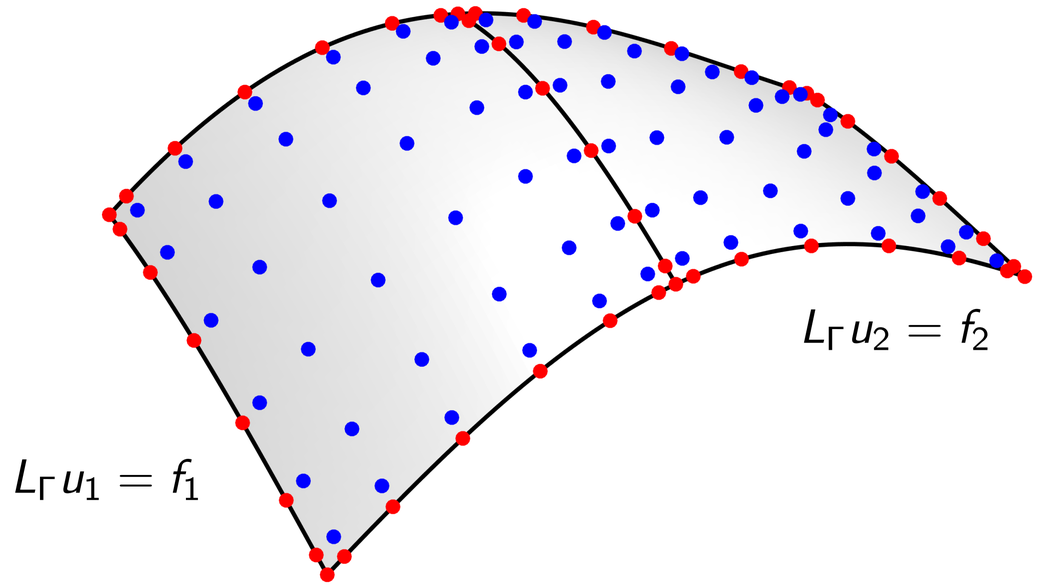




[F., 2022]

High-order discretization
Two glued patches



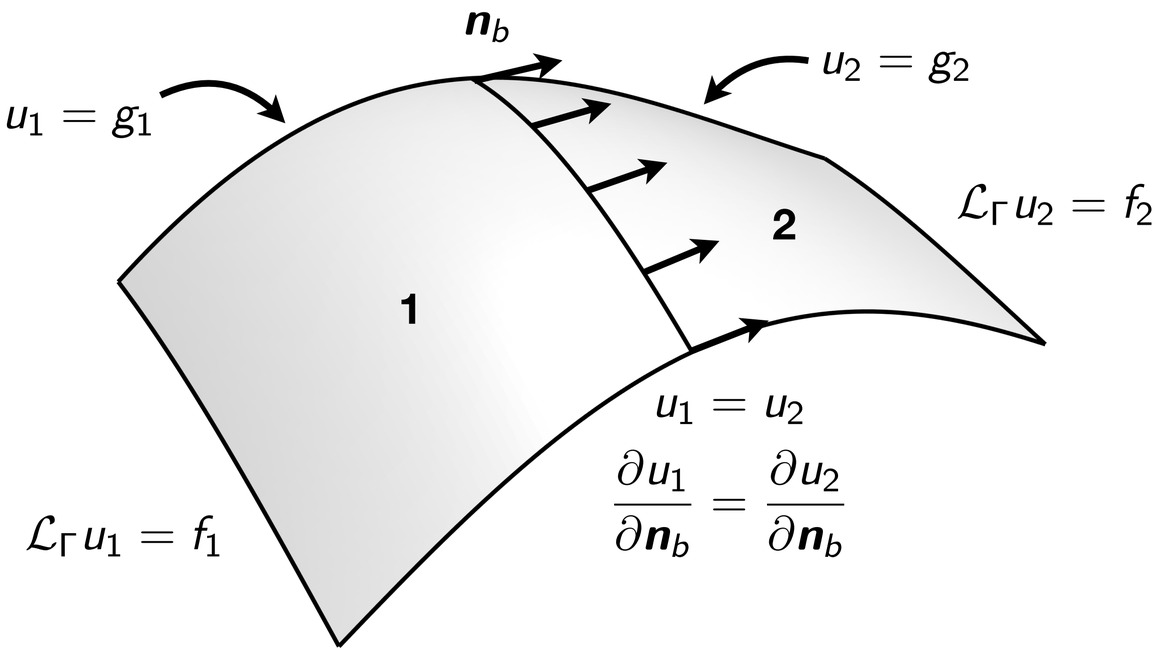


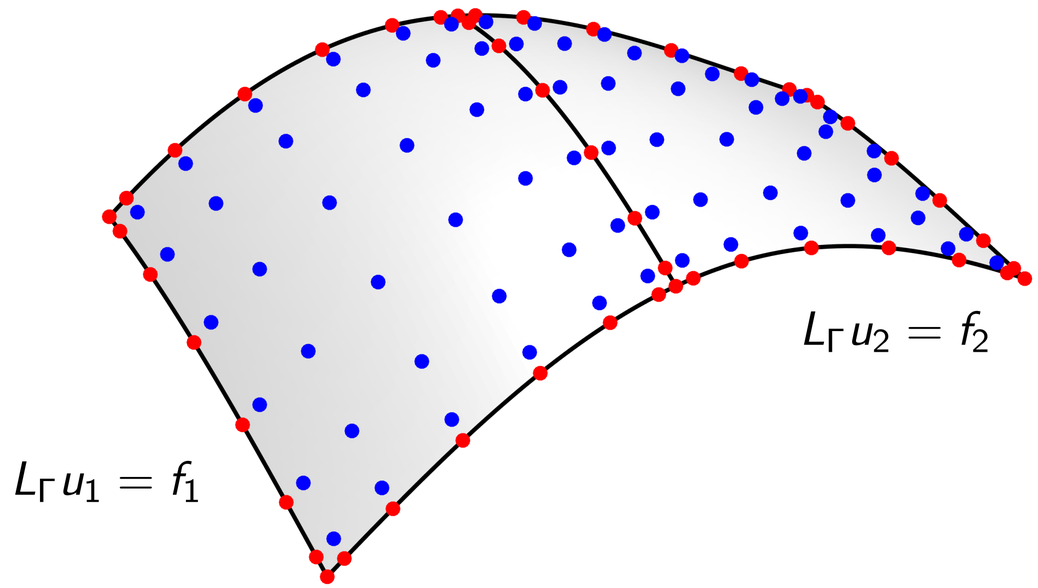
- Know how to do local solves on each element: “Solution operator”

$$S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \qquad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$$

[F., 2022]

High-order discretization
Two glued patches





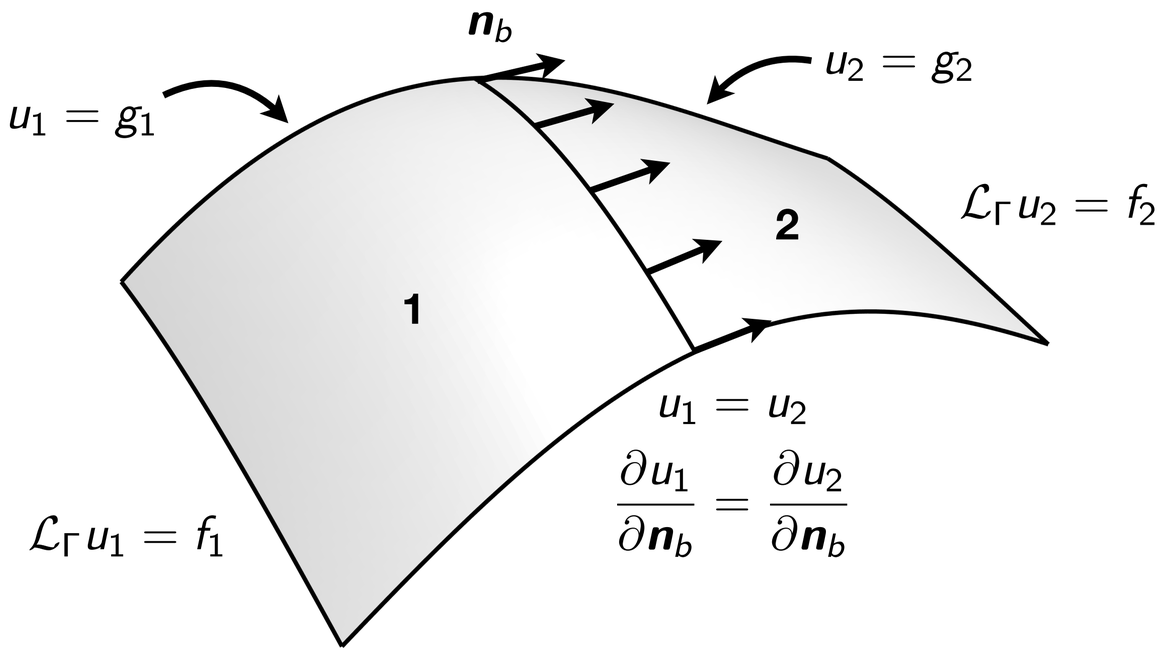
- Know how to do local solves on each element: “Solution operator”
- Know how information flows out of each element: “Dirichlet-to-Neumann map”

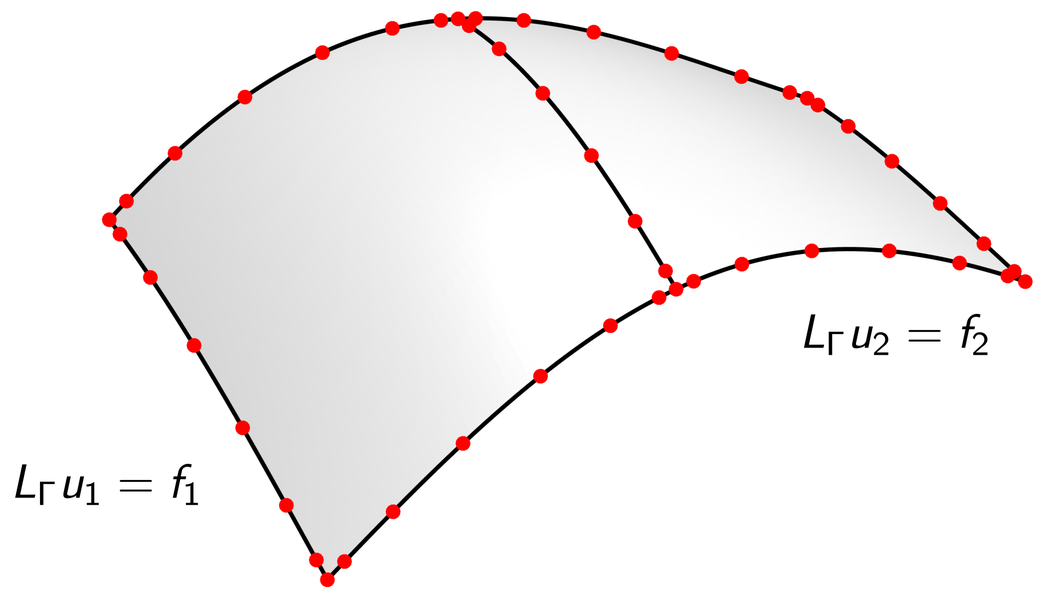
$$S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \qquad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$$

$$DtN_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial n_b} \qquad DtN_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial n_b}$$

[F., 2022]

High-order discretization
Two glued patches





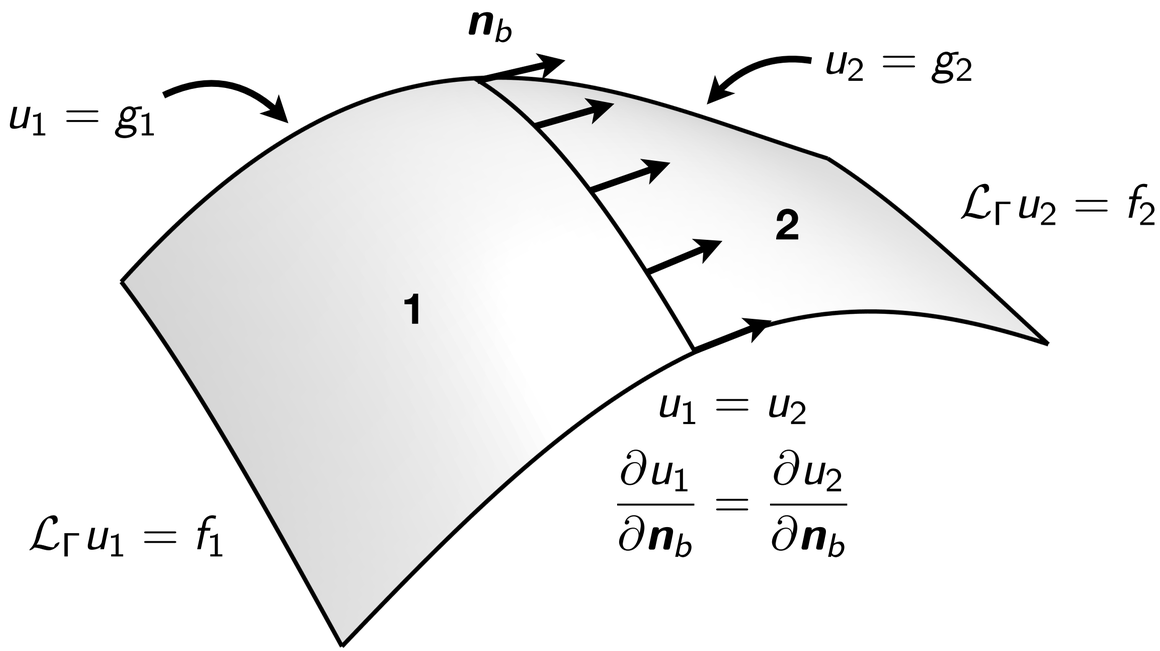
- Know how to do local solves on each element: "Solution operator"
- Know how information flows out of each element: "Dirichlet-to-Neumann map"

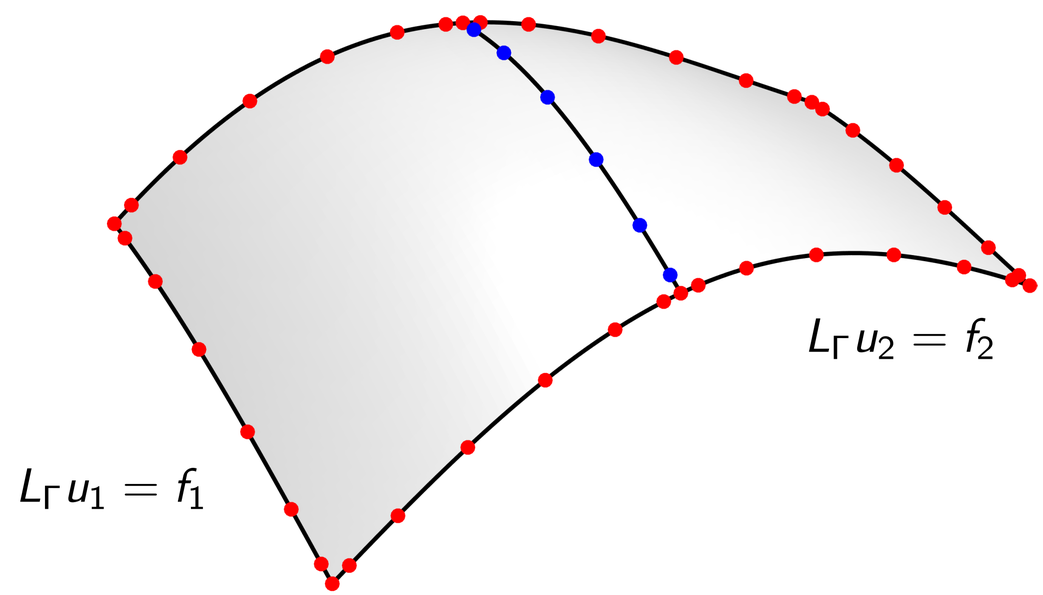
$$S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \qquad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$$

$$DtN_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial n_b} \qquad DtN_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial n_b}$$

[F., 2022]

High-order discretization
Two glued patches





- Know how to do local solves on each element: “Solution operator”
- Know how information flows out of each element: “Dirichlet-to-Neumann map”
- Take Schur complement to eliminate interior degrees of freedom:

$$S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \qquad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$$

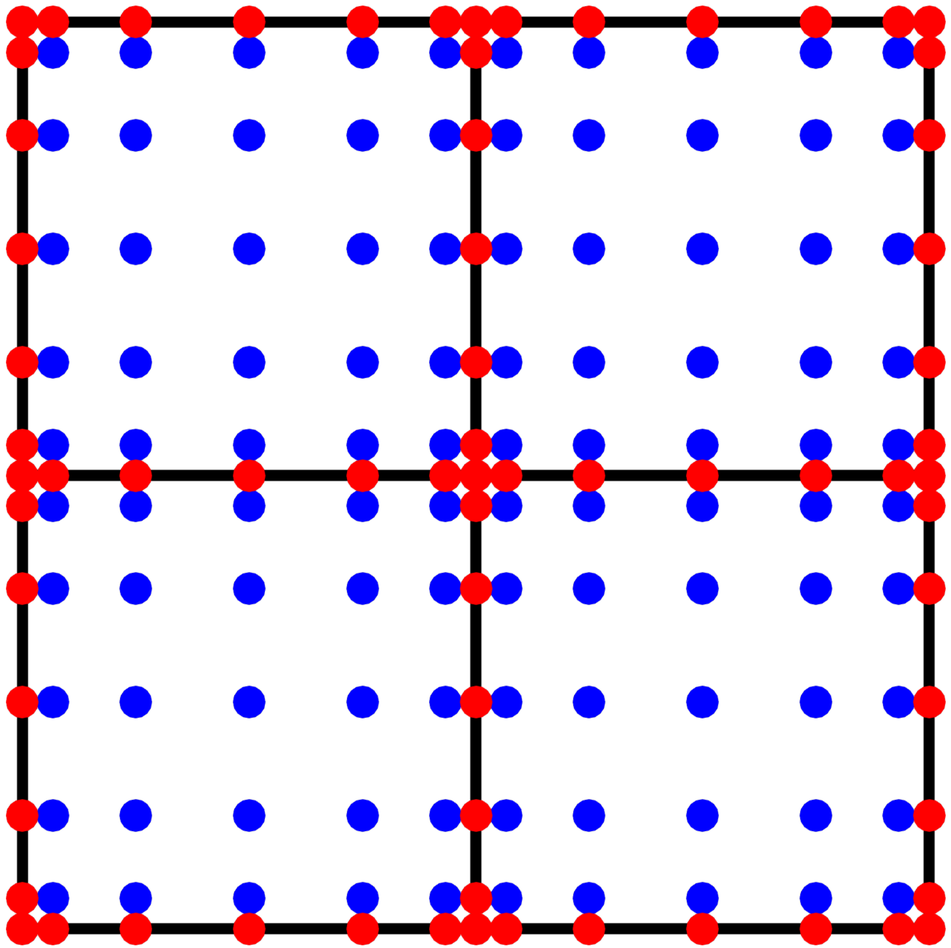
$$DtN_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial n_b} \qquad DtN_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial n_b}$$

$$S_{\text{glue}} = -\left(DtN_1^{\text{glue}} + DtN_2^{\text{glue}}\right)^{-1} \begin{bmatrix} DtN_1^{\text{glue},1} & DtN_2^{\text{glue},2} \end{bmatrix} \qquad S_{\text{glue}} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = u_{\text{glue}}$$

[F., 2022]

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)



[F., 2022]

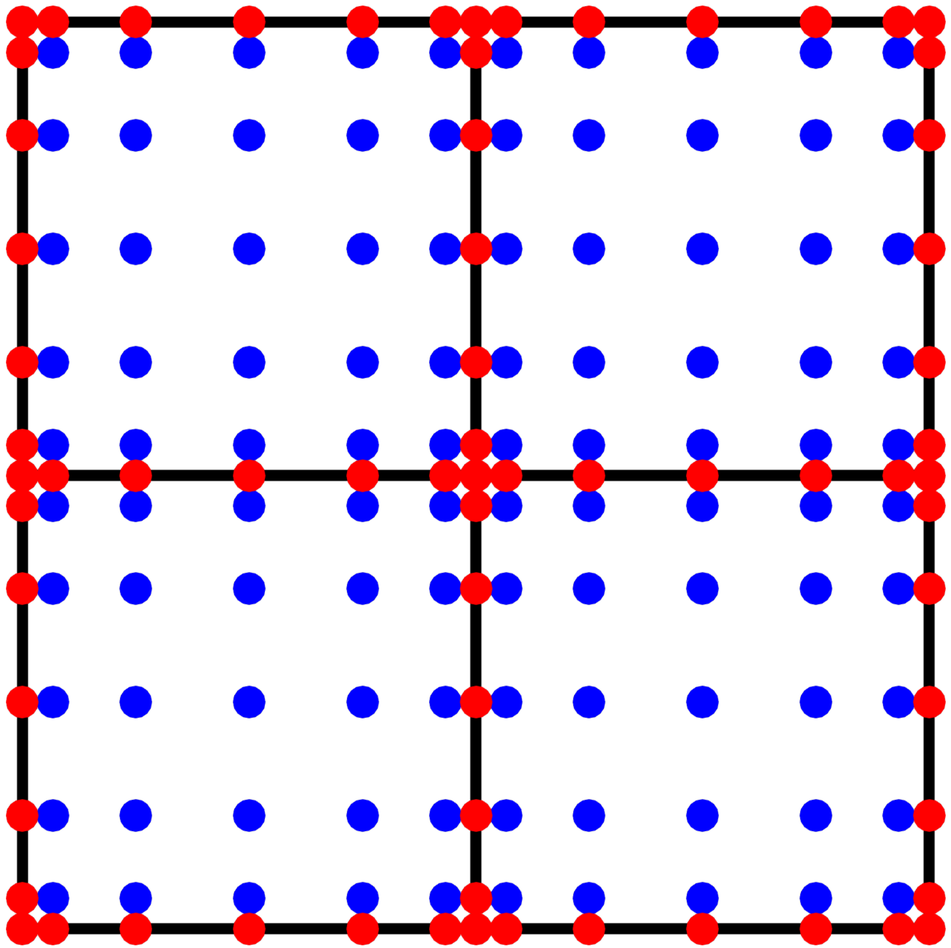
[Martinsson, 2013]

[Gillman & Martinsson, 2014]

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Build element operators $\mathcal{O}(Np^6)$



[F., 2022]

[Martinsson, 2013]

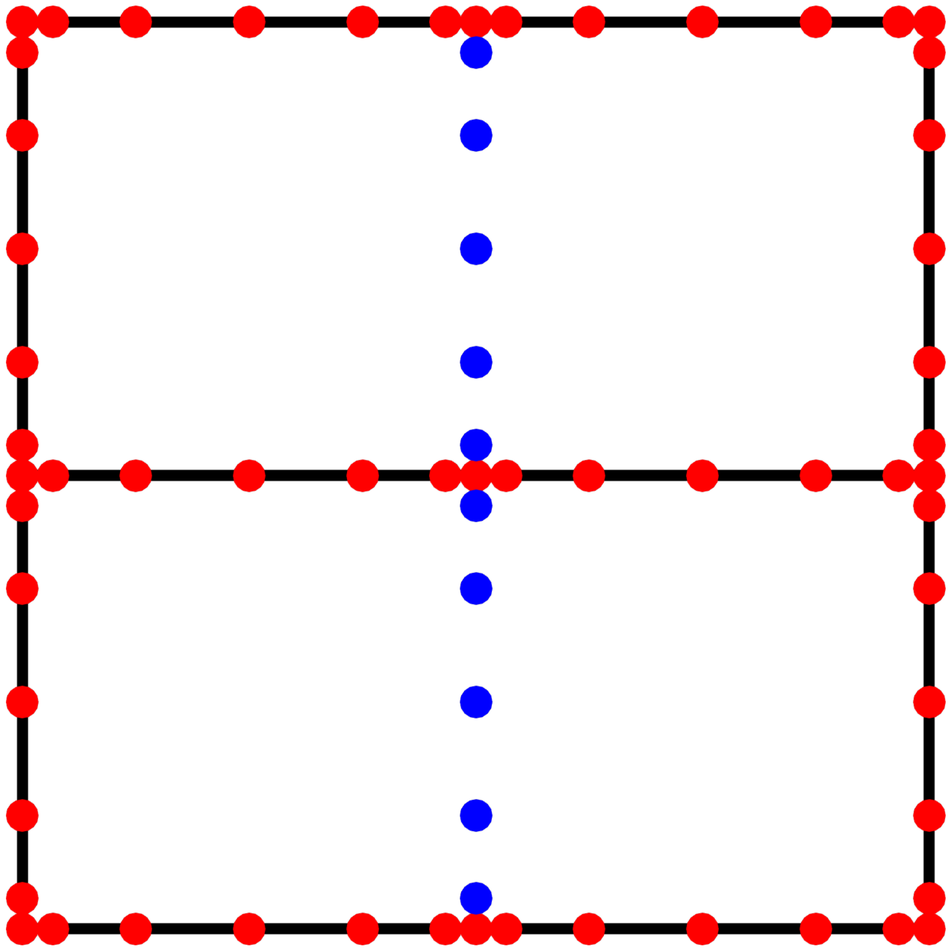
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Merge operators $\mathcal{O}(N^{3/2})$



[F., 2022]

[Martinsson, 2013]

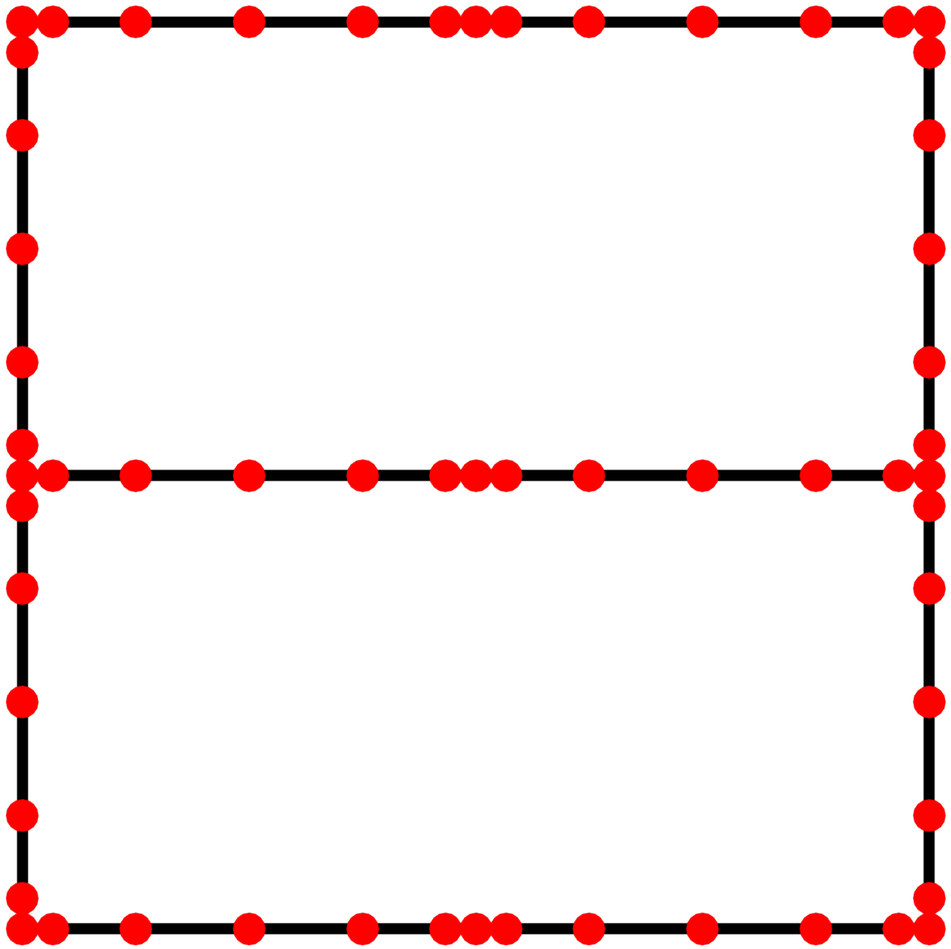
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Merge operators $\mathcal{O}(N^{3/2})$



[F., 2022]

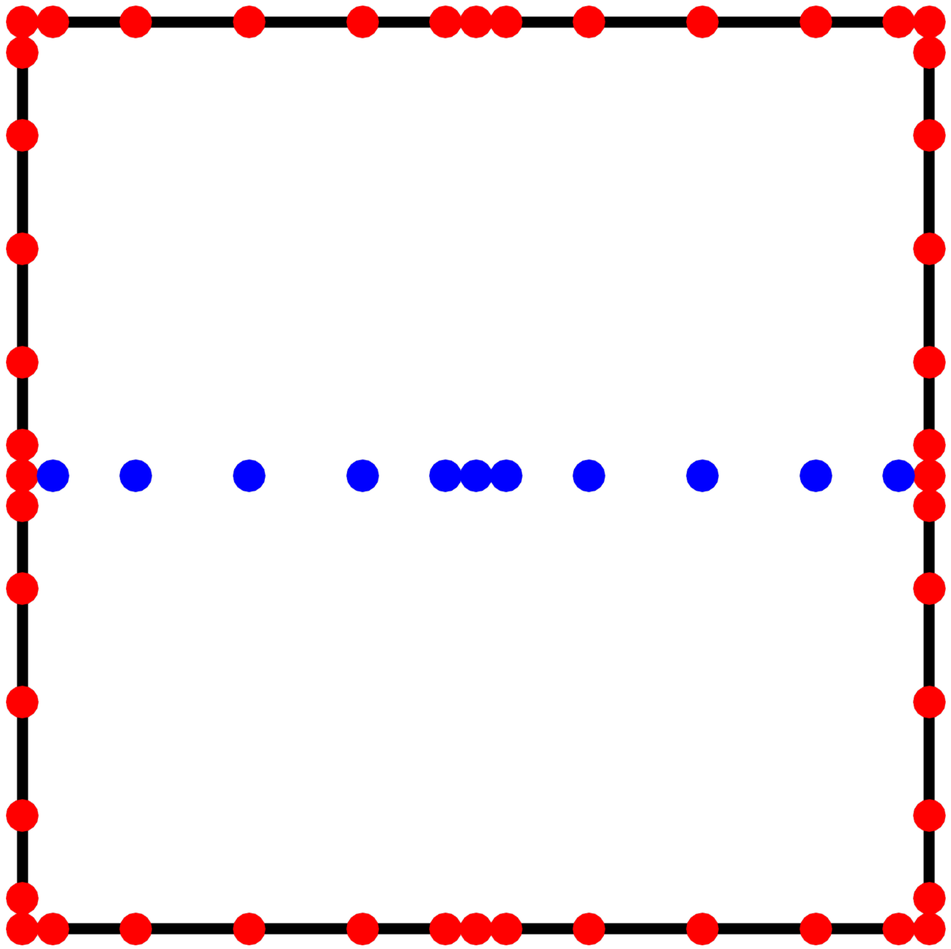
[Martinsson, 2013]

[Gillman & Martinsson, 2014]

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Merge operators $\mathcal{O}(N^{3/2})$



[F., 2022]

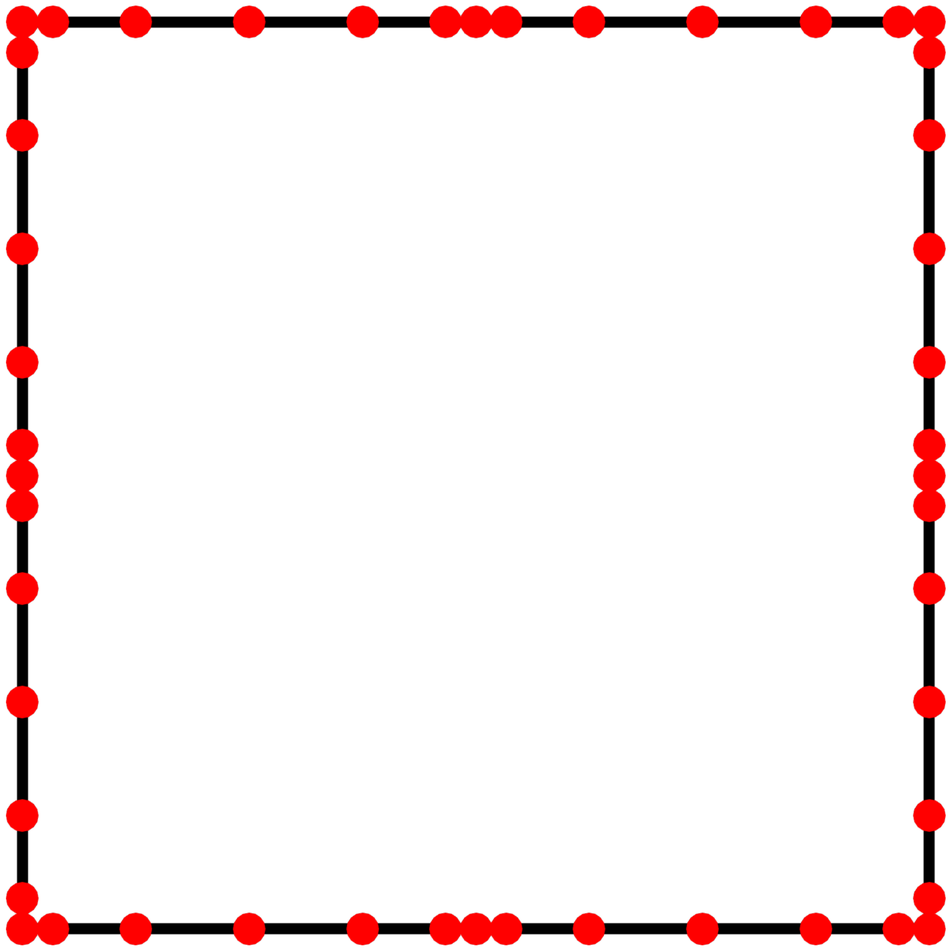
[Martinsson, 2013]

[Gillman & Martinsson, 2014]

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Merge operators $\mathcal{O}(N^{3/2})$



[F., 2022]

[Martinsson, 2013]

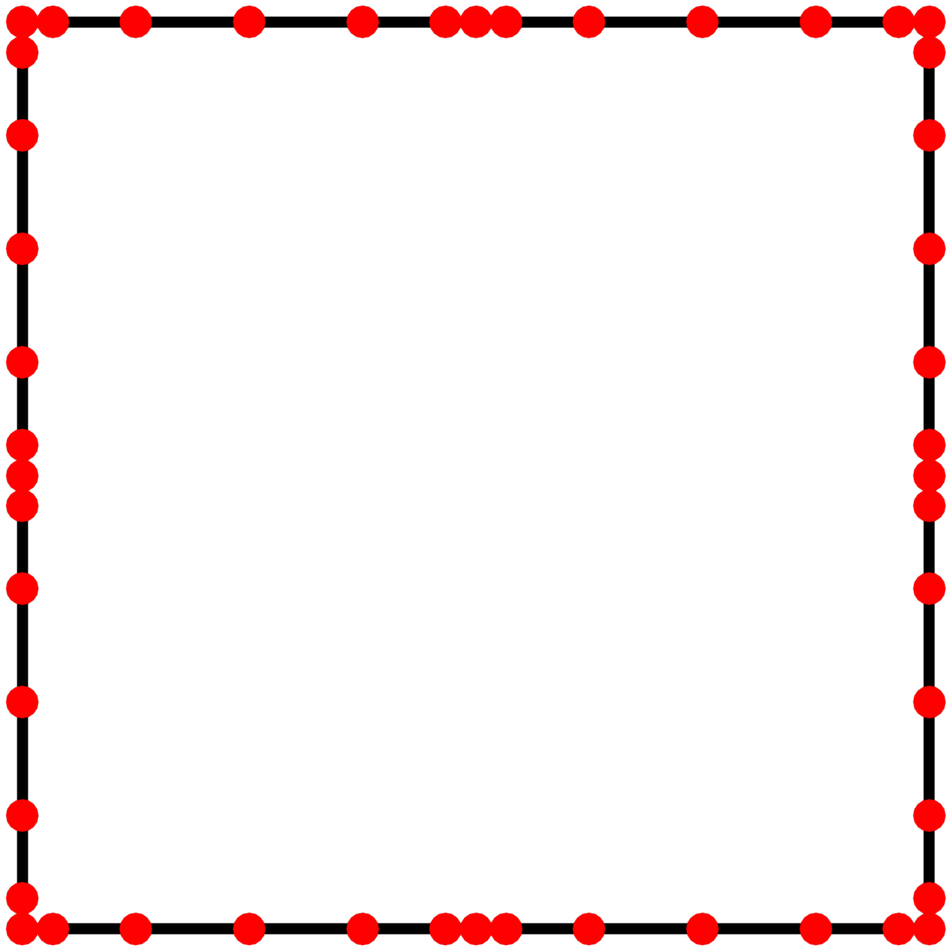
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



[F., 2022]

[Martinsson, 2013]

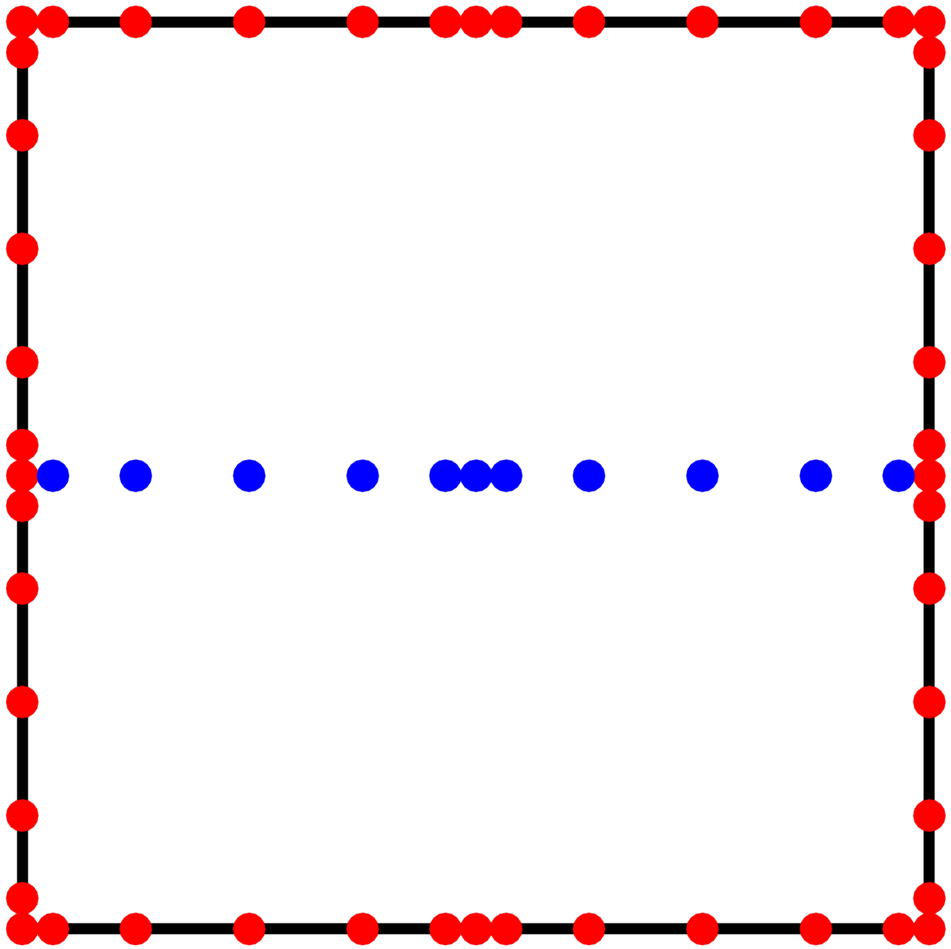
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



[F., 2022]

[Martinsson, 2013]

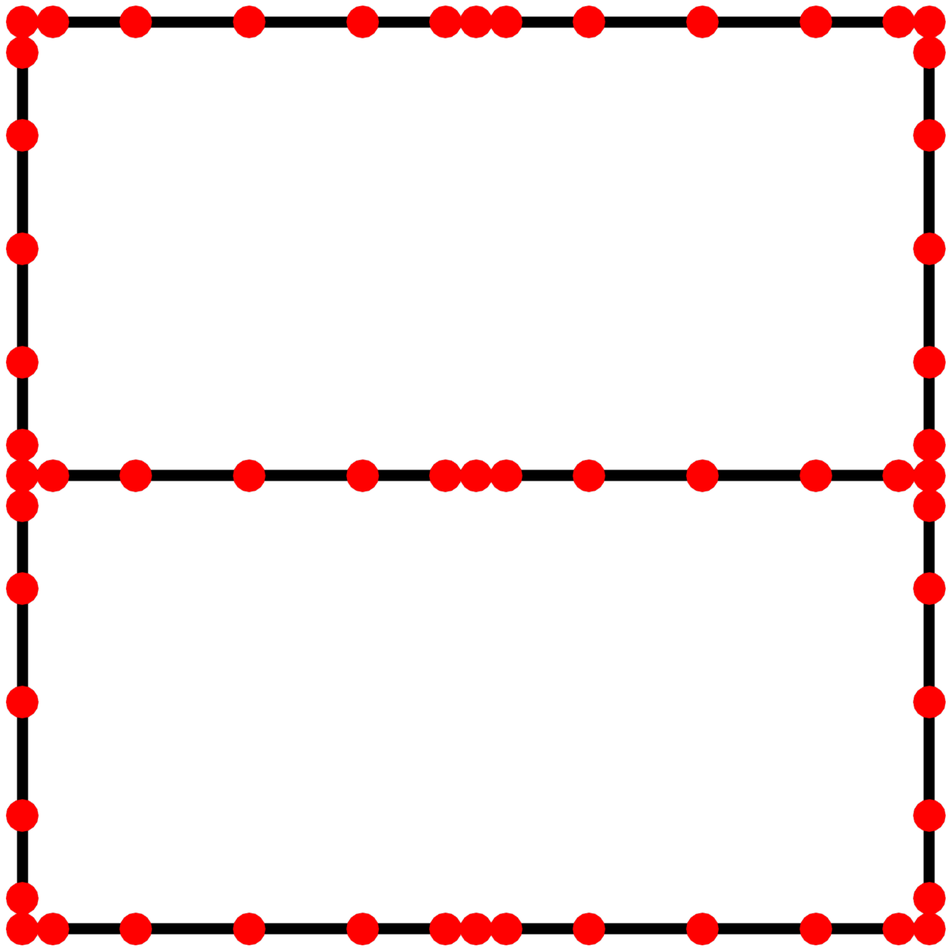
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



[F., 2022]

[Martinsson, 2013]

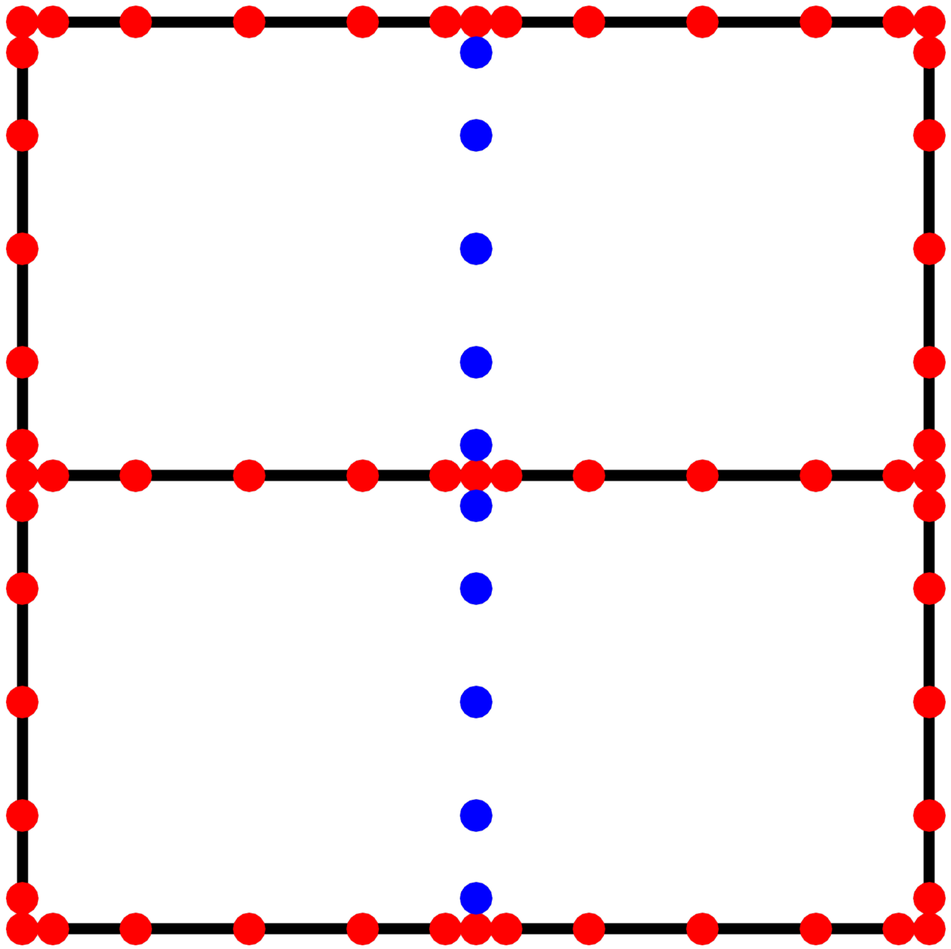
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



[F., 2022]

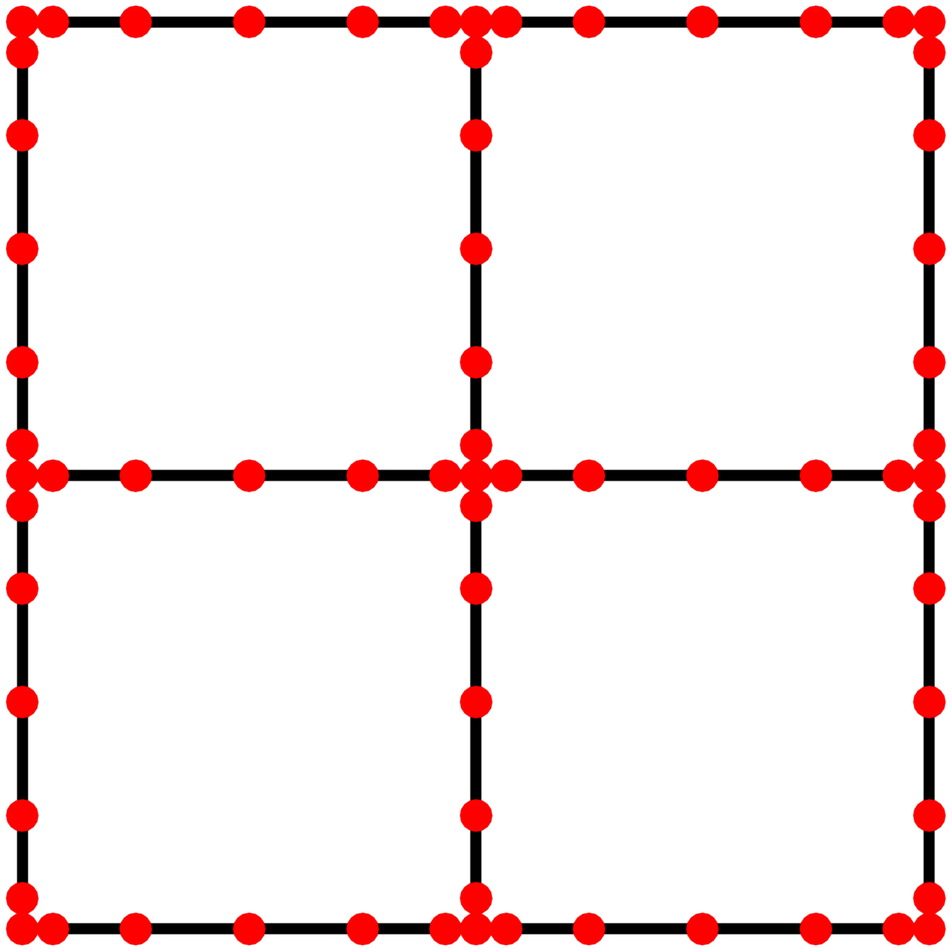
[Martinsson, 2013]

[Gillman & Martinsson, 2014]

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



[F., 2022]

[Martinsson, 2013]

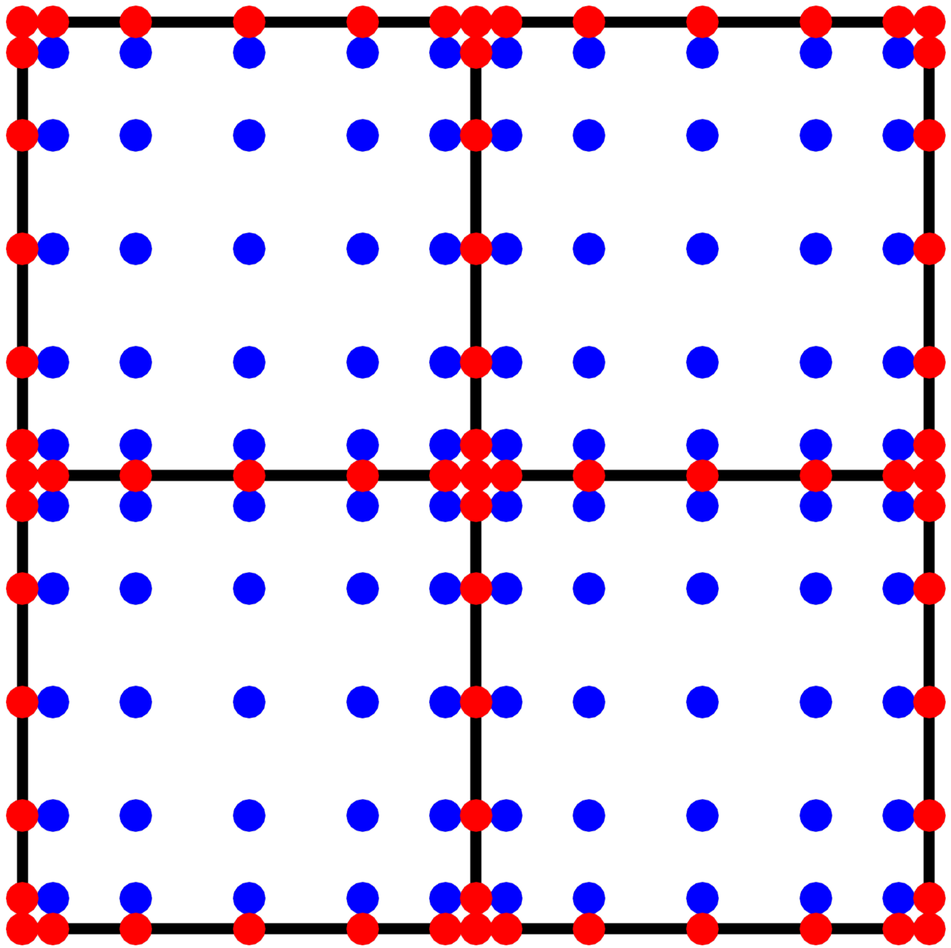
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



[F., 2022]

[Martinsson, 2013]

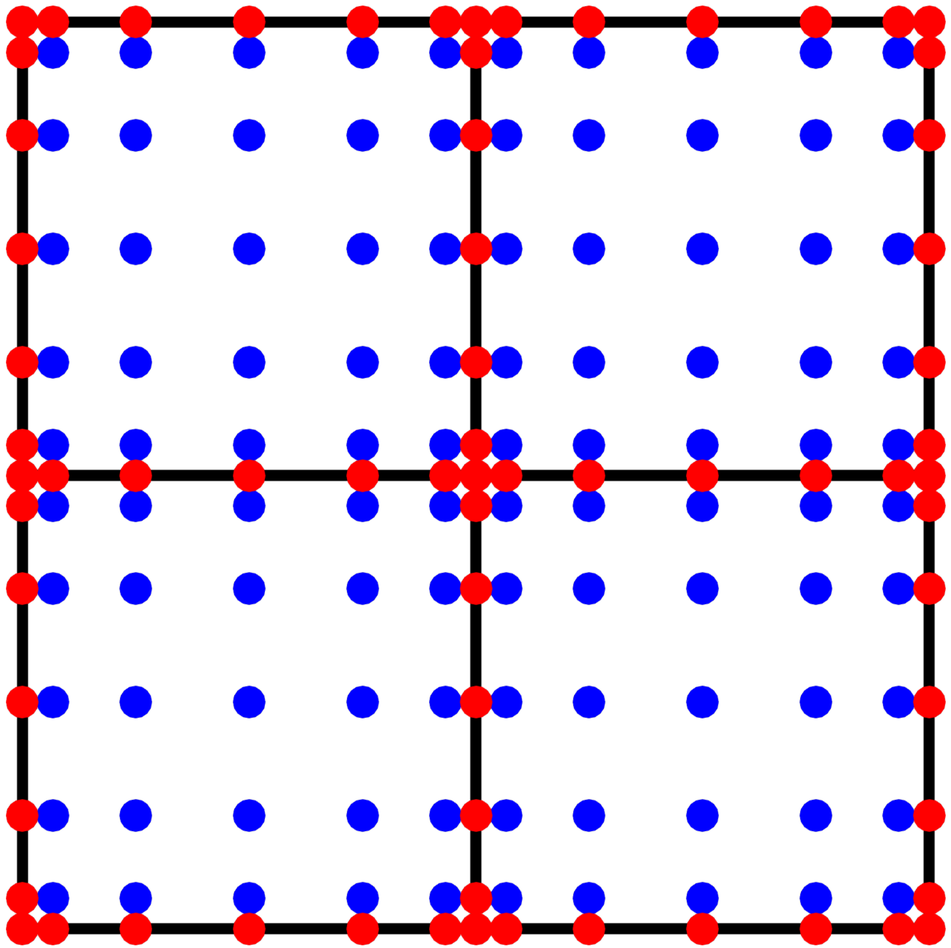
[Gillman & Martinsson, 2014]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)

Apply solution operators $\mathcal{O}(N \log N)$



$$\underbrace{N^{3/2}}_{\text{Factorization}} + \underbrace{N \log N}_{\text{Solve}}$$

Factorization results in a hierarchy of solution operators stored in memory, so repeated solves are fast.

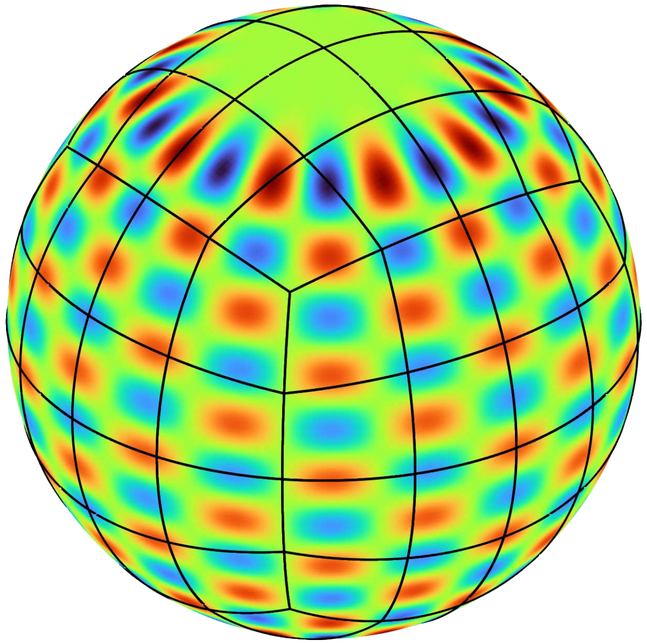
[F., 2022]

[Martinsson, 2013]

[Gillman & Martinsson, 2014]

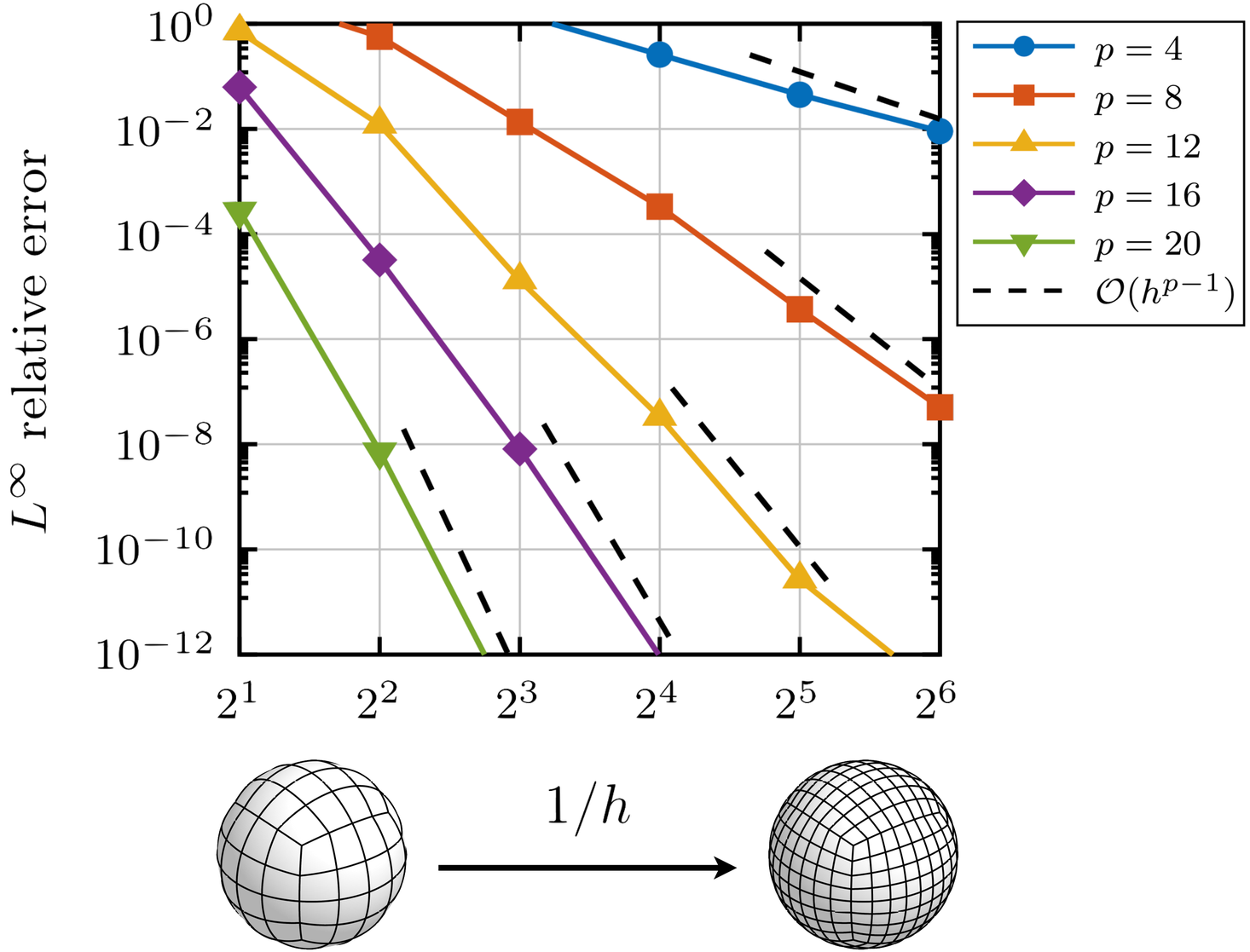
Examples
Laplace–Beltrami and convergence

$\Delta_\Gamma u = f, \quad \Gamma = \text{sphere}$



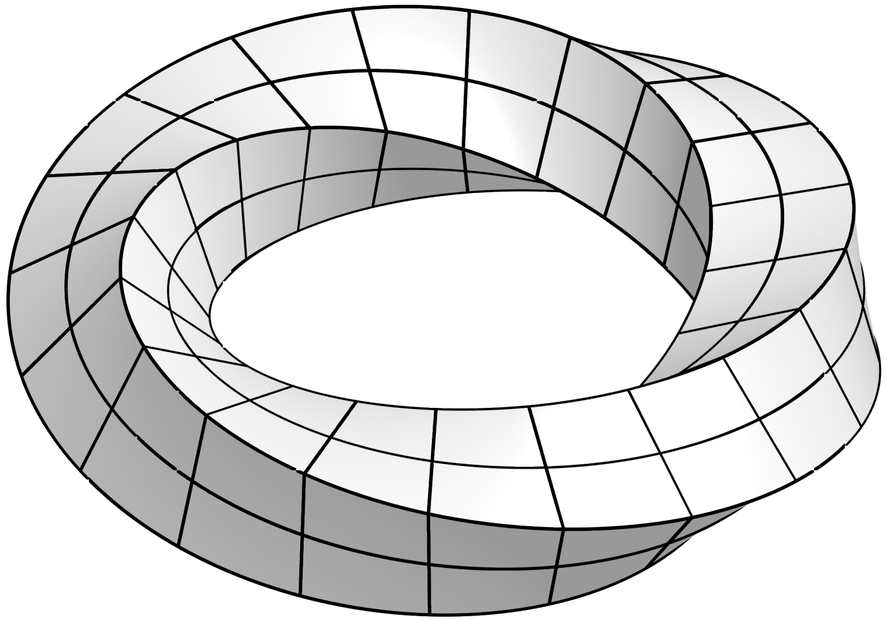
$u(\boldsymbol{x}) = \text{spherical harmonic, } Y_\ell^m(\boldsymbol{x})$

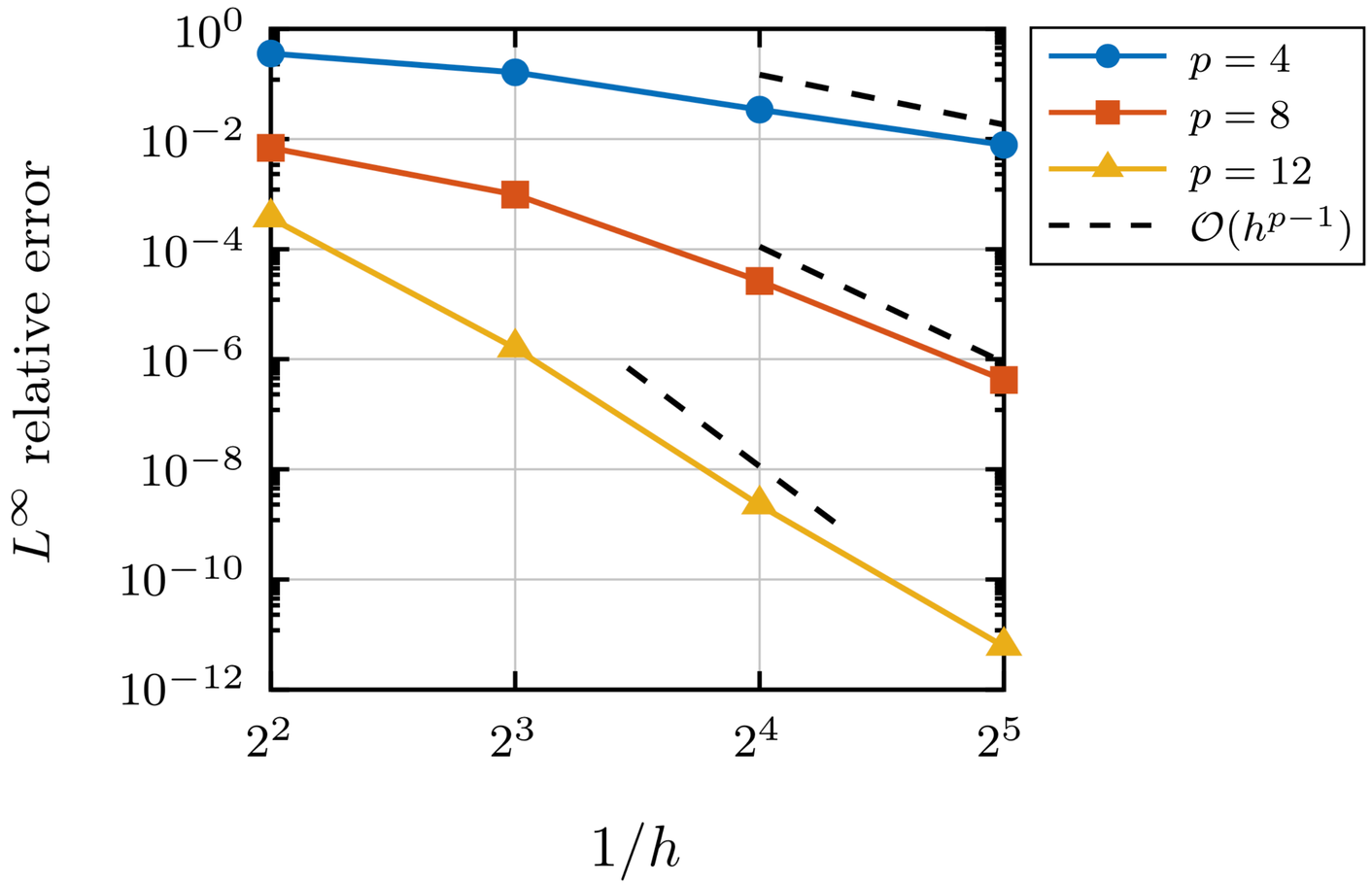
$f(\boldsymbol{x}) = -\ell(\ell+1)Y_\ell^m(\boldsymbol{x})$



Examples
Laplace–Beltrami with edges

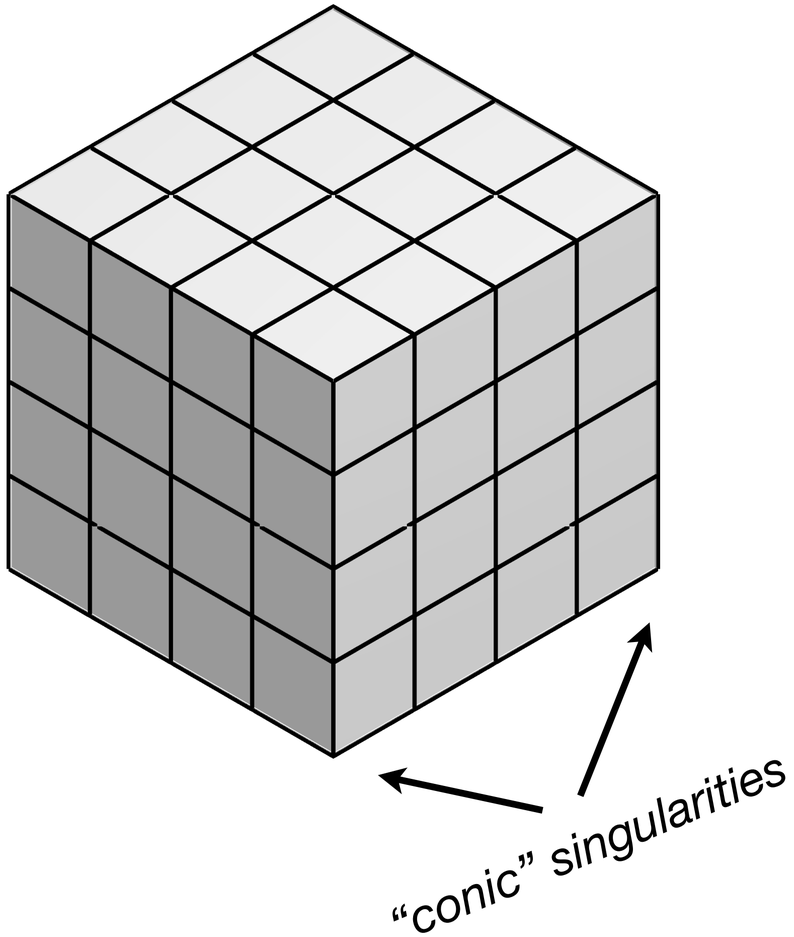
Glue conditions also allow for sharp interfaces and corners.

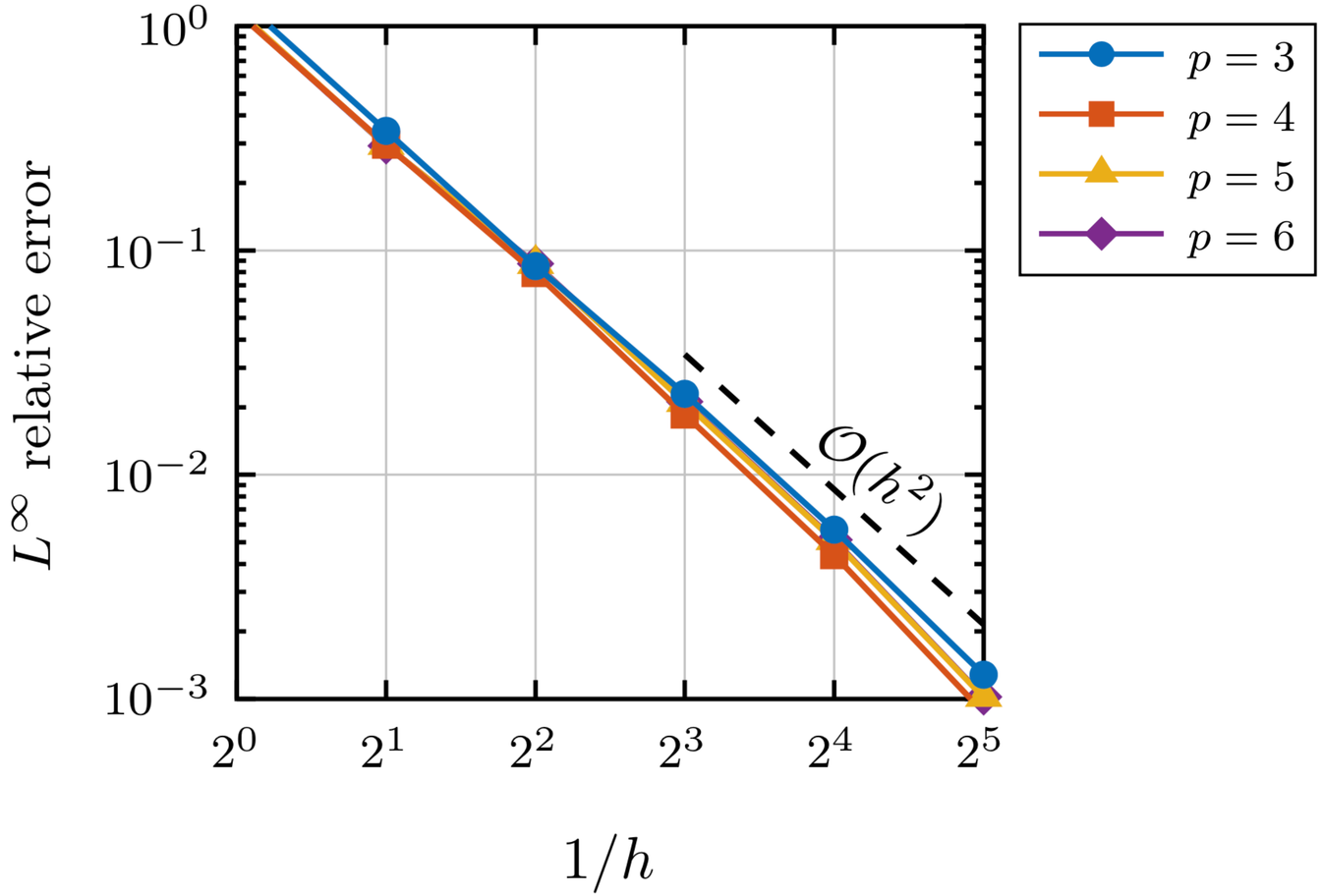




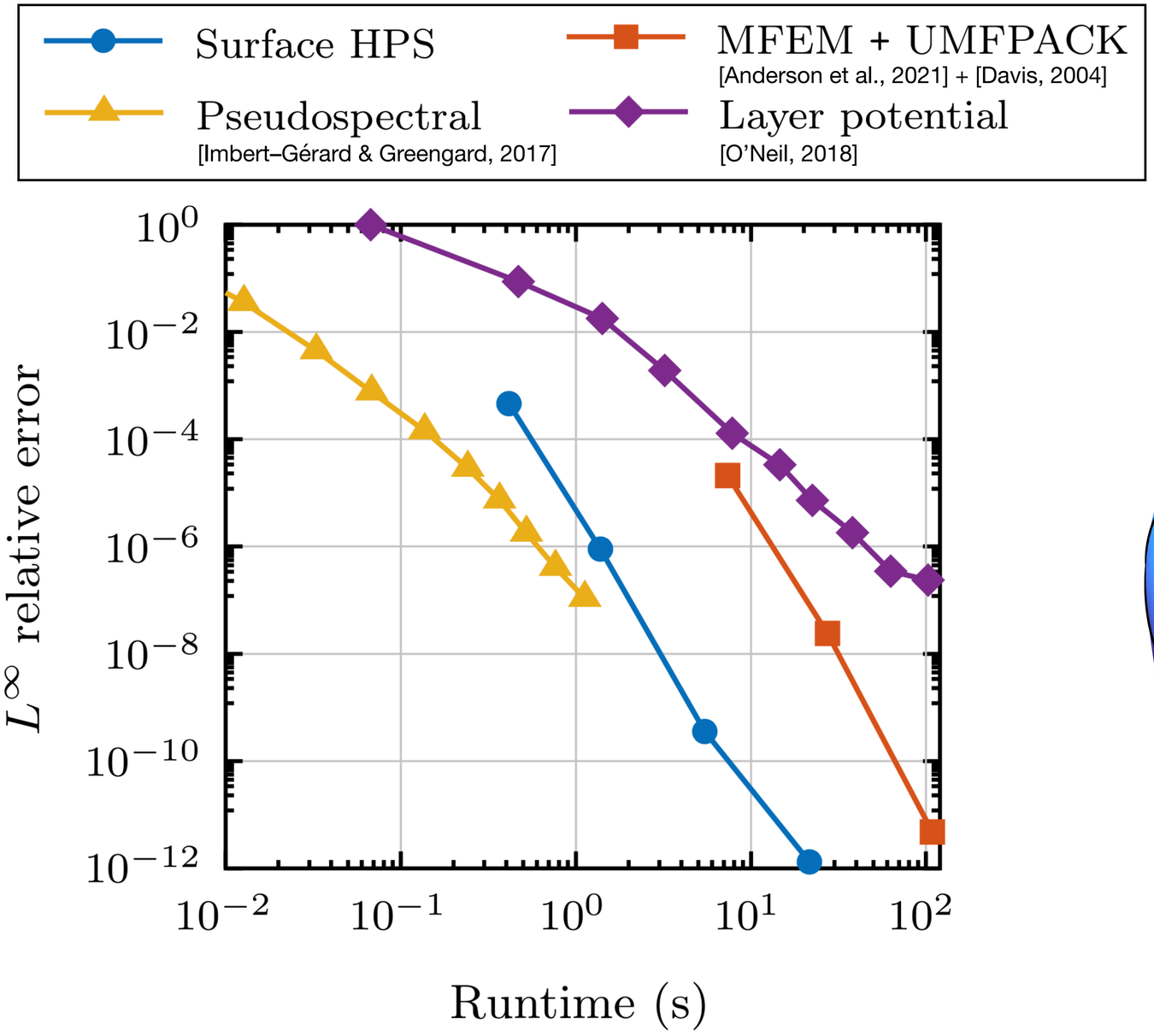
Examples
Laplace–Beltrami with corners

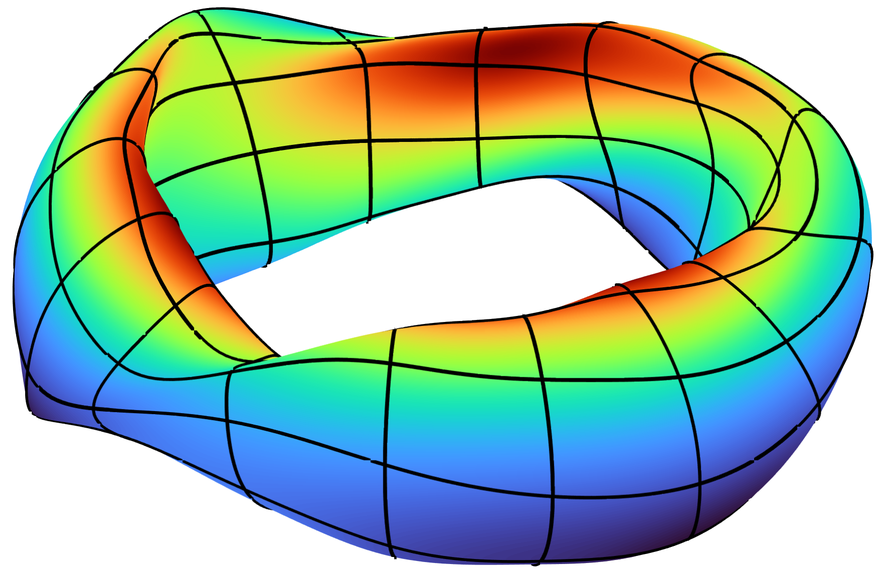
Glue conditions also allow for sharp interfaces and corners… but high-order convergence may be lost.





Examples
Laplace–Beltrami: *“accuracy vs. effort”*





16th order
8-core Intel i9 Macbook Pro
64GB RAM

Examples
Hodge decomposition

Any smooth vector field $\boldsymbol{f}$ tangent to a surface can be written as:

$$\boldsymbol{f} = \underbrace{\nabla_\Gamma u}_{\textit{curl-free}} + \underbrace{\boldsymbol{n} \times \nabla_\Gamma v}_{\textit{div-free}} + \underbrace{\boldsymbol{w}}_{\textit{harmonic}}$$

where $\boldsymbol{w}$ satisfies $\nabla_\Gamma \cdot \boldsymbol{w} = 0$ and $\nabla_\Gamma \cdot (\boldsymbol{n} \times \boldsymbol{w}) = 0$.

Such decompositions play an important role in integral representations for computational electromagnetics.

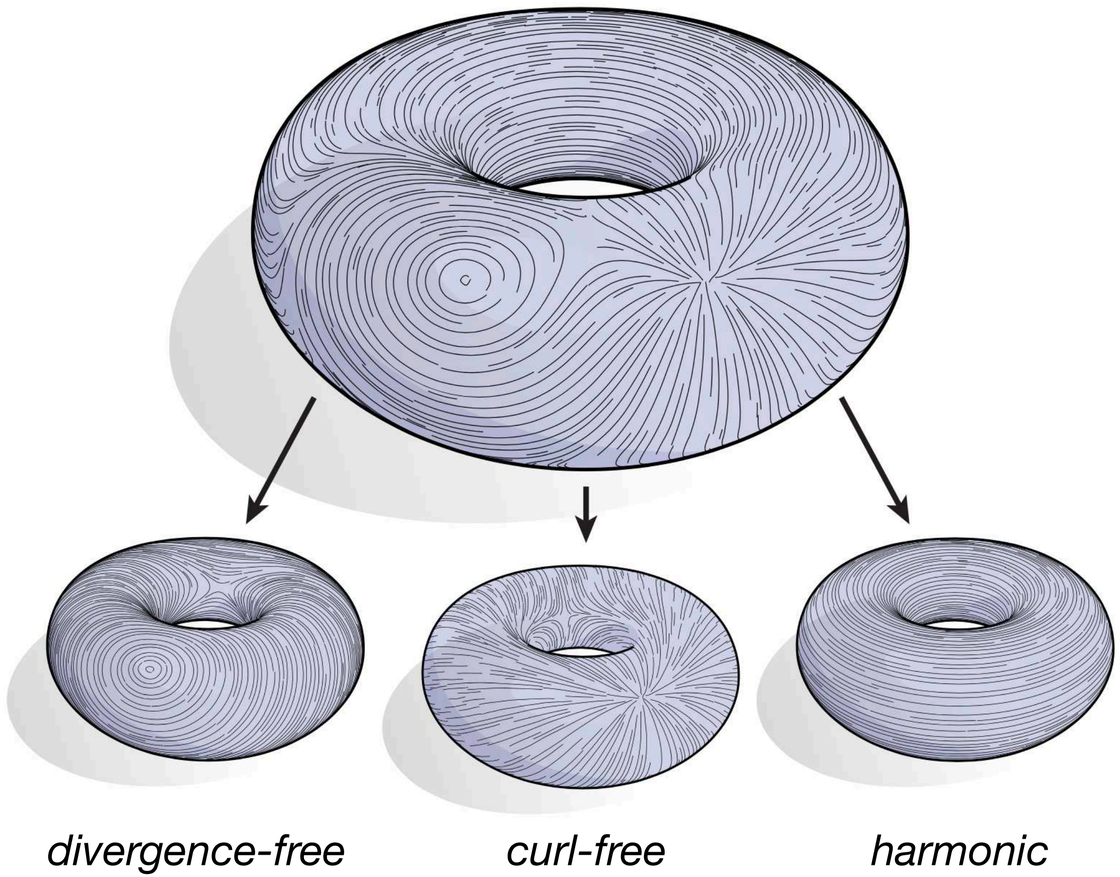


Illustration by Keenan Crane

Examples
Hodge decomposition

Any smooth vector field $\boldsymbol{f}$ tangent to a surface can be written as:

$$\boldsymbol{f} = \underbrace{\nabla_\Gamma u}_{\textit{curl-free}} + \underbrace{\boldsymbol{n} \times \nabla_\Gamma v}_{\textit{div-free}} + \underbrace{\boldsymbol{w}}_{\textit{harmonic}}$$

where $\boldsymbol{w}$ satisfies $\nabla_\Gamma \cdot \boldsymbol{w} = 0$ and $\nabla_\Gamma \cdot (\boldsymbol{n} \times \boldsymbol{w}) = 0$.

Such decompositions play an important role in integral representations for computational electromagnetics.

One may compute this decomposition by solving

$$\Delta_\Gamma u = \nabla_\Gamma \cdot \boldsymbol{f}$$

$$\Delta_\Gamma v = -\nabla_\Gamma \cdot (\boldsymbol{n} \times \boldsymbol{f})$$

and then setting $\boldsymbol{w} = \boldsymbol{f} - \nabla_\Gamma u - \boldsymbol{n} \times \nabla_\Gamma v$.

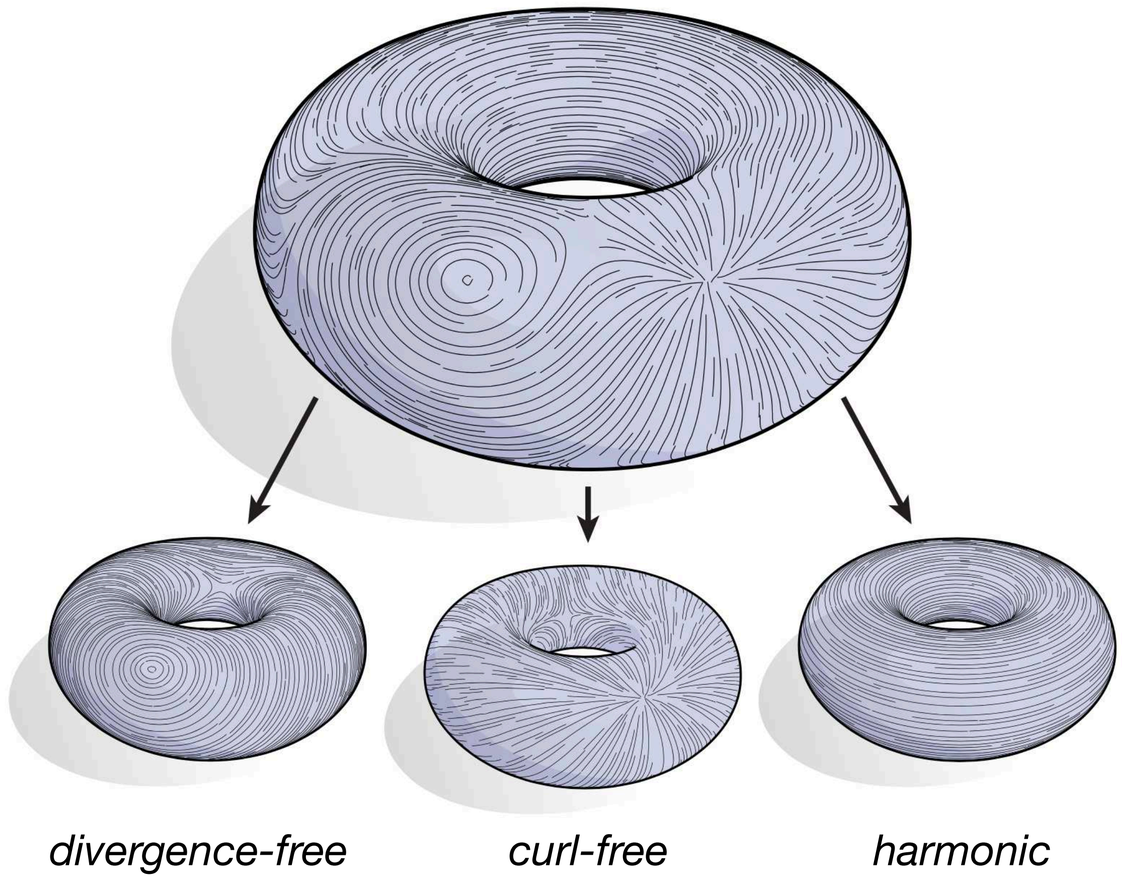
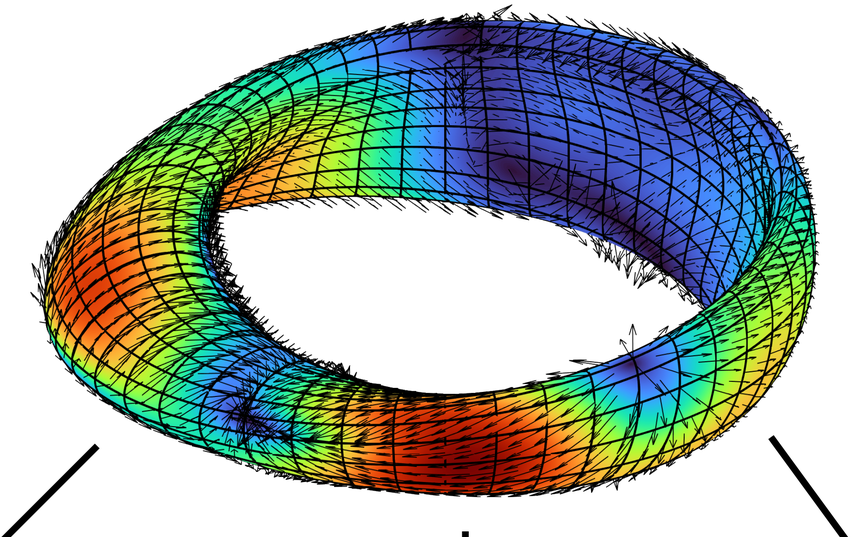


Illustration by Keenan Crane

Examples
Hodge decomposition

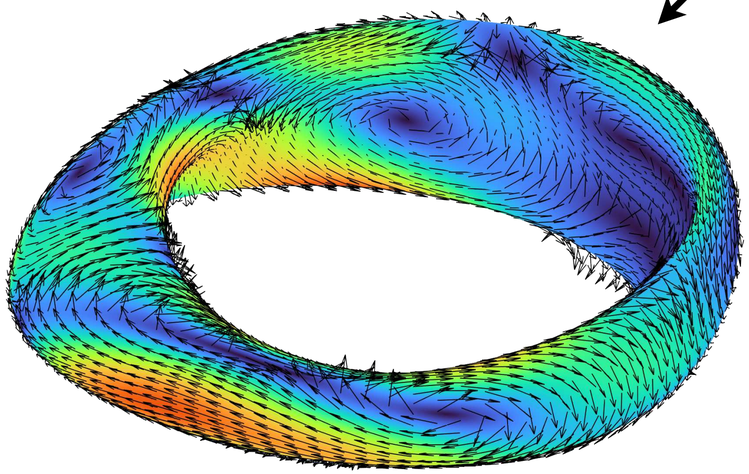


200k unknowns, 16th order

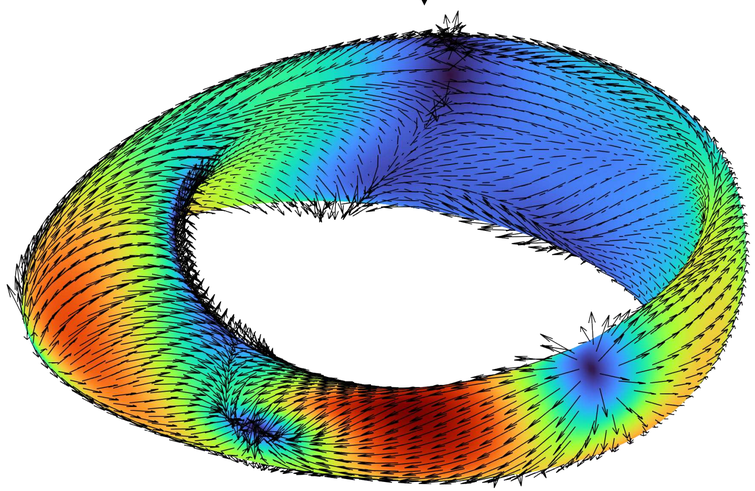
4 seconds

$\|\nabla_\Gamma \cdot \boldsymbol{w}\|_2 \approx 4 \times 10^{-7}$

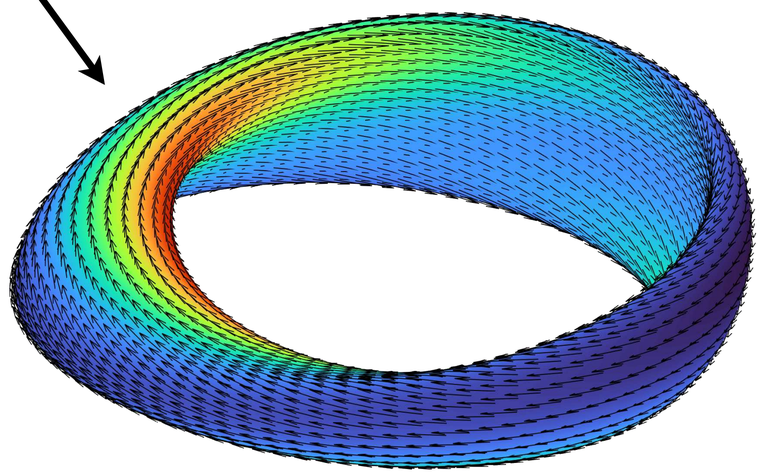
$\|\nabla_\Gamma \cdot (\boldsymbol{n} \times \boldsymbol{w})\|_2 \approx 2 \times 10^{-6}$



divergence-free



curl-free



harmonic

Examples
Eigenvalue problems

$$\Delta_\Gamma u = \lambda u$$


Mengjian Hua
(advisee)


Dhairya Malhotra

Simultaneous inverse iteration:

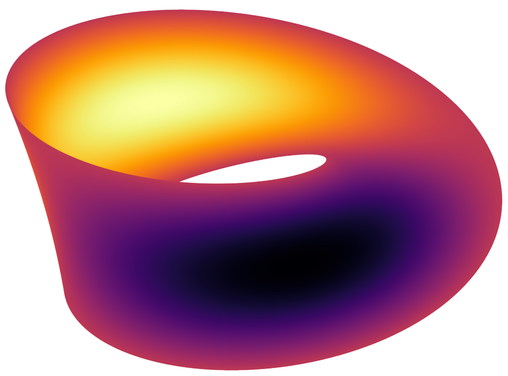
$$Q^{(0)} = \texttt{rand}(N, m)$$

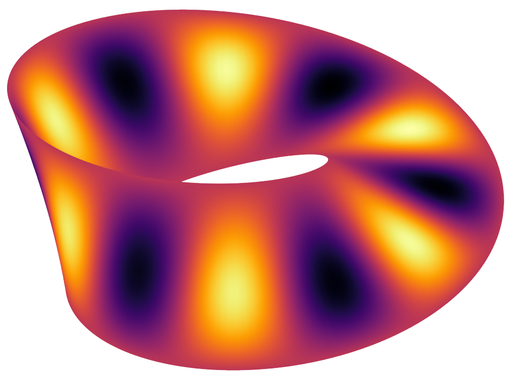
for $k = 1, 2, \ldots$

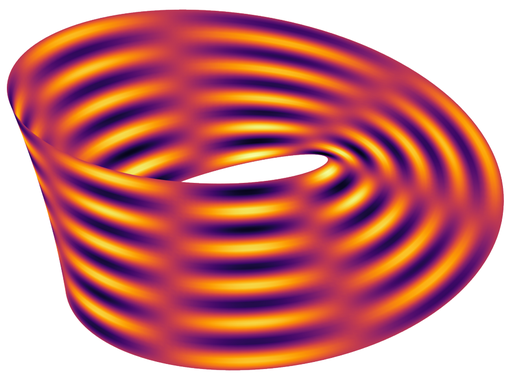
Stored in RAM, very fast apply

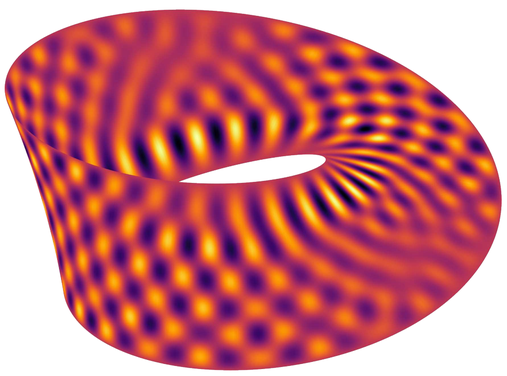
$$Z^{(k-1)} = \Delta_\Gamma^{-1} Q^{(k-1)}$$

$$Q^{(k)} R^{(k)} = Z^{(k-1)}$$


$\lambda = -10.93...$

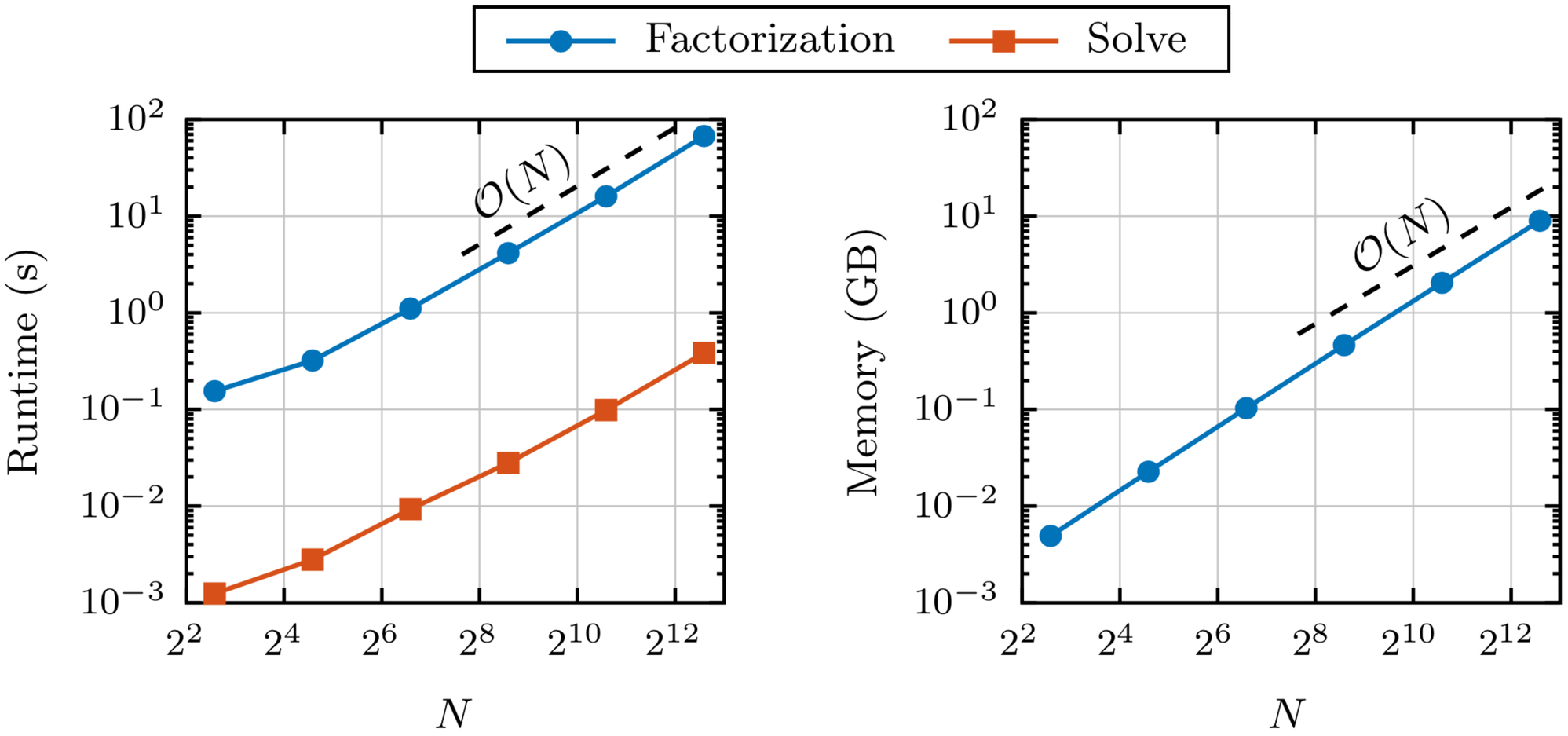

$\lambda = -44.66...$


$\lambda = -1000.56...$


$\lambda = -998.11...$

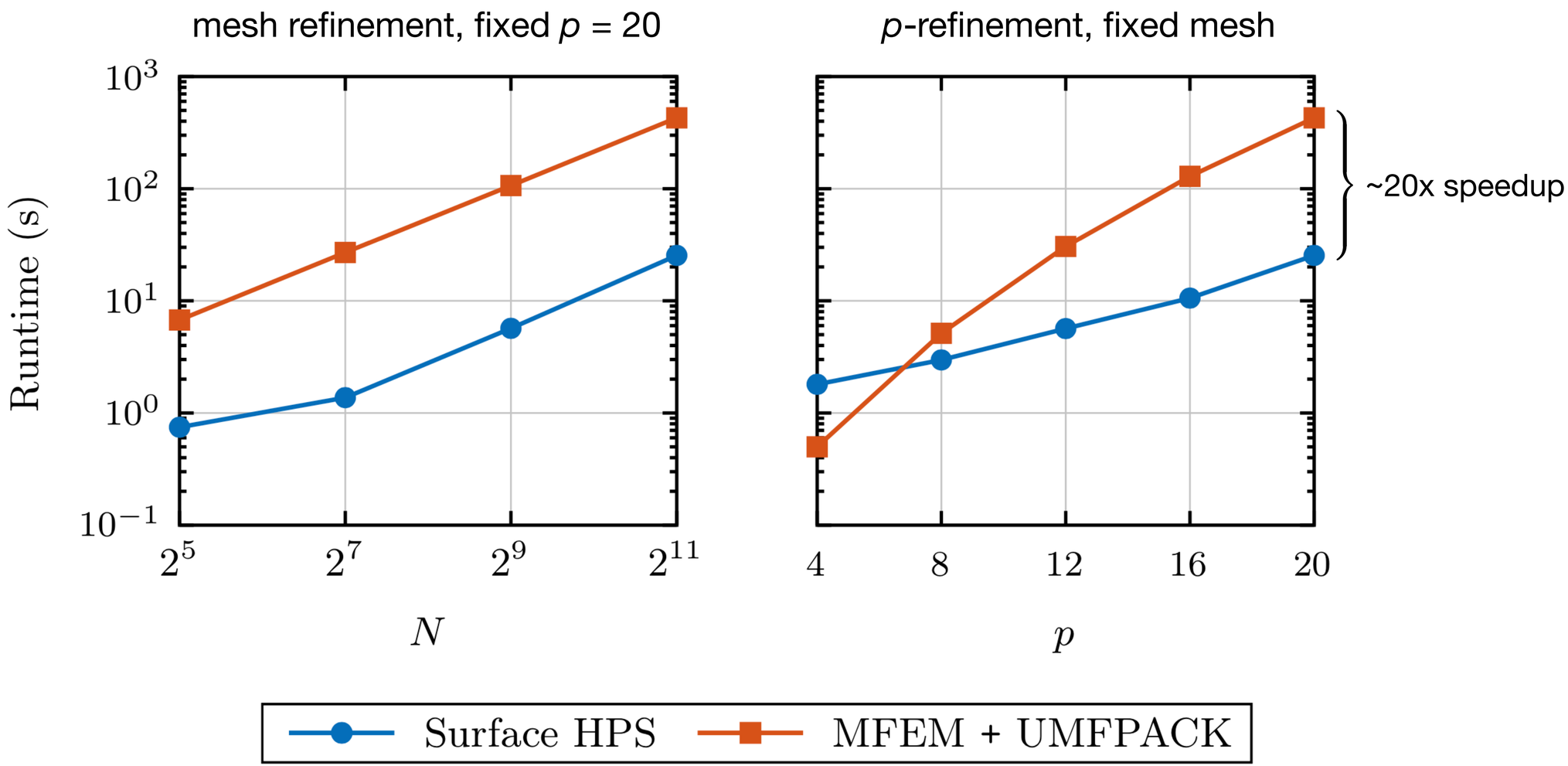
Examples
Performance

Computational complexity: $\mathcal{O}(N^{3/2})$ factorization, but $\mathcal{O}(N)$ is observed pre-asymptotically.



Examples
Performance

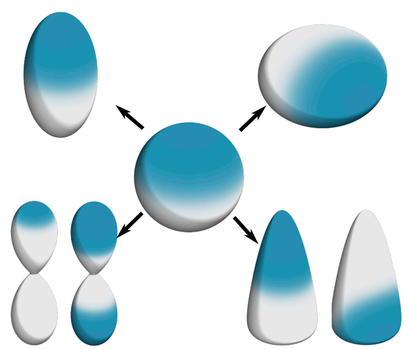
Significant speedups at high order when compared to FEM with a sparse direct solver.



Examples
Reaction–diffusion systems

- Reaction–diffusion processes are ubiquitous in biology.

$$\frac{\partial u}{\partial t} = \underbrace{\mathcal{L}_\Gamma u}_{\text{Diffusion}} + \underbrace{\mathcal{N}(u)}_{\text{Reaction}} \quad \text{on } \Gamma$$

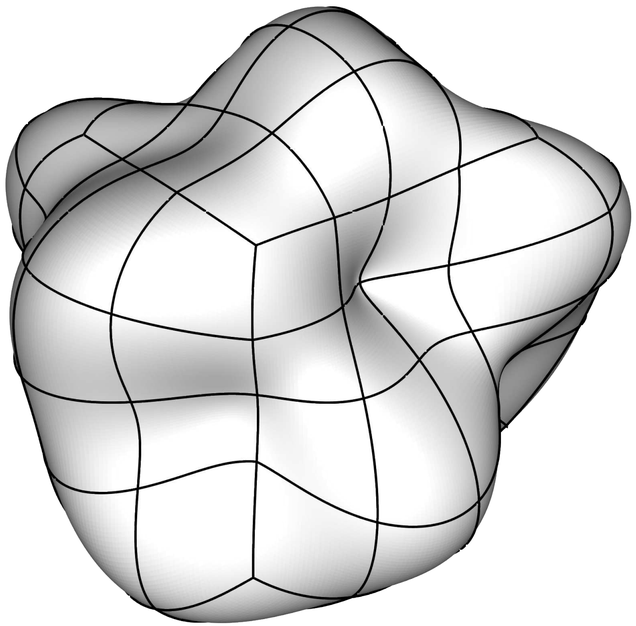


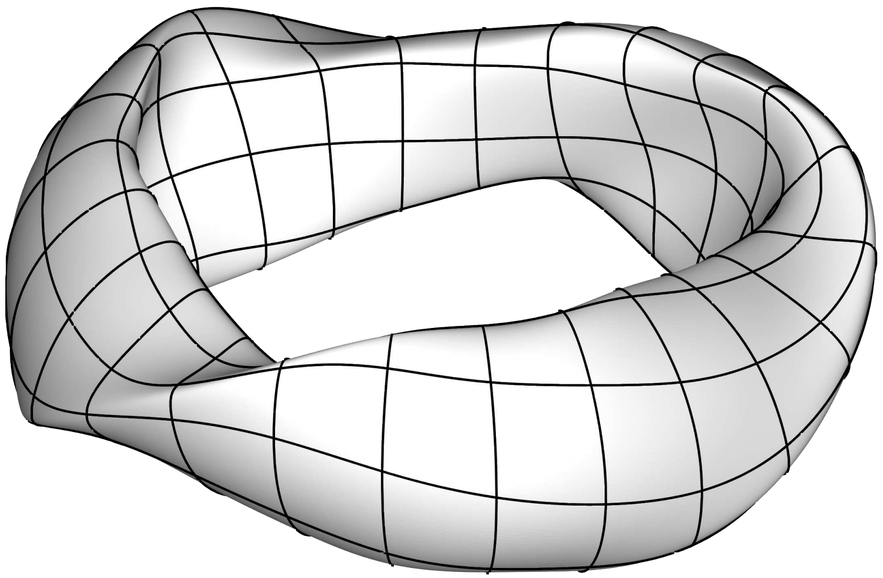
- Reaction and diffusion timescales are often orders of magnitude different.
- Implicit time-stepping can alleviate stability issues (e.g., backward Euler or IMEX-BDF4)

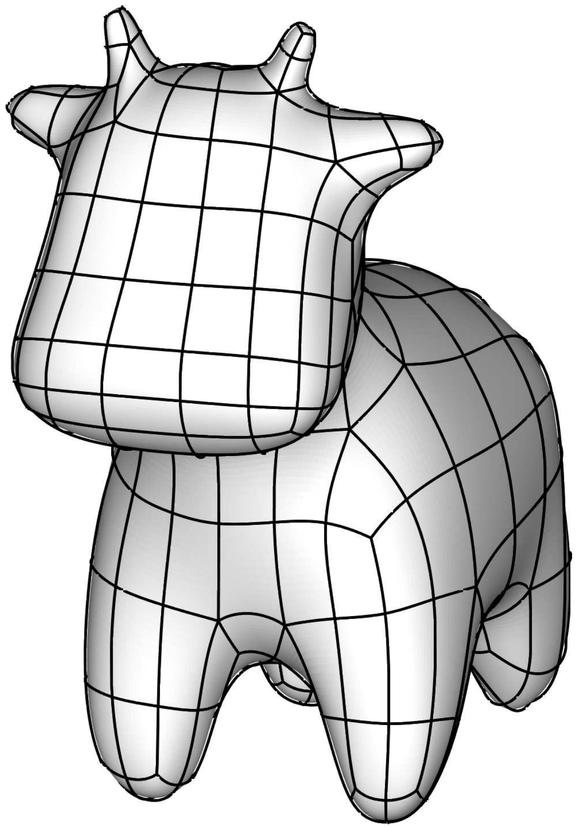
$$\frac{\partial u}{\partial t} = \mathcal{L}_\Gamma u + \mathcal{N}(u) \xrightarrow{\text{(e.g. backward Euler)}} u^{k+1} = \underbrace{(I - \Delta t \mathcal{L}_\Gamma)^{-1}}_{\substack{\text{Stored in RAM,} \\ \text{very fast apply}}} \left(u^k + \Delta t \mathcal{N}(u^k)\right)$$

- If geometry, time step, and parameters do not change with time, we can precompute a solver once and reuse it at every step.

Examples
Reaction–diffusion systems

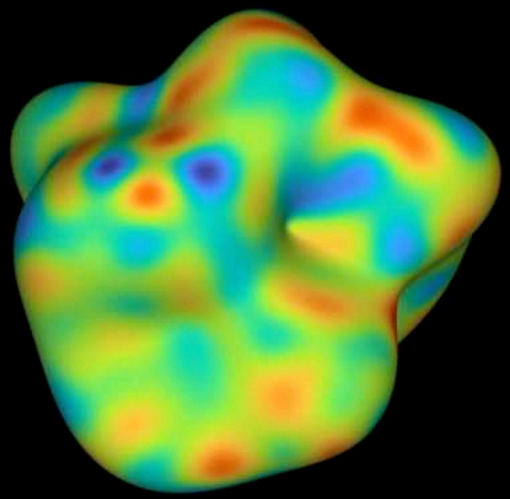






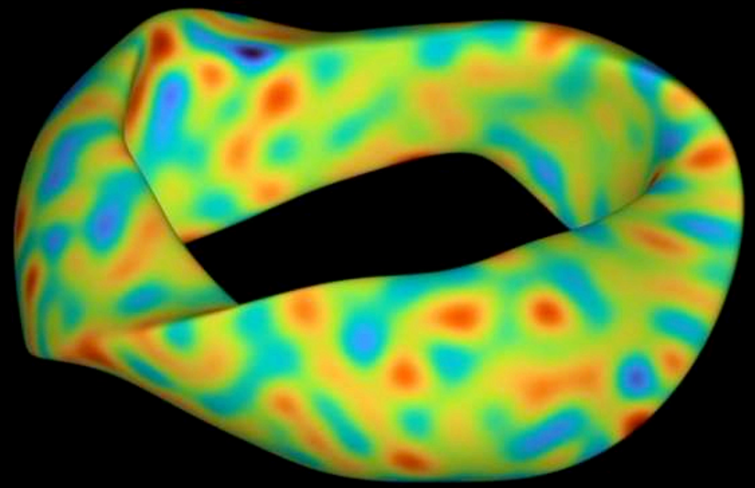
16th order
8-core Intel i9 Macbook Pro
64GB RAM

$$\frac{\partial u}{\partial t} = \delta_u^2 \Delta_\Gamma u + \alpha u(1 - \tau_1 v^2) + v(1 - \tau_2 u)$$
$$\frac{\partial v}{\partial t} = \delta_v^2 \Delta_\Gamma v + \beta v(1 + \alpha \tau_1 uv/\beta) + u(\gamma + \tau_2 v)$$



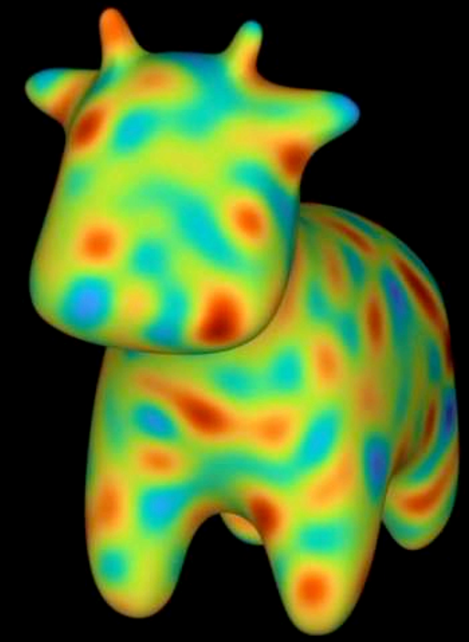
25k unknowns

~20 fps



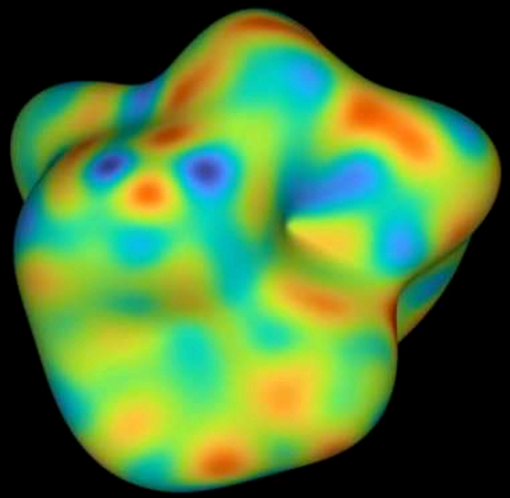
50k unknowns

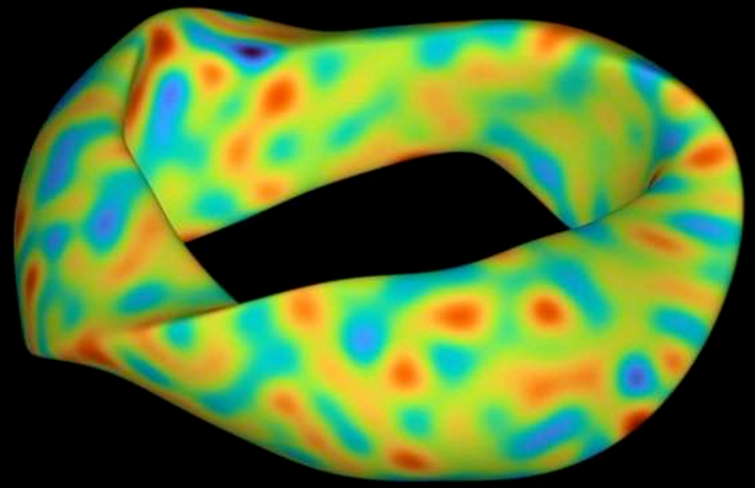
~13 fps



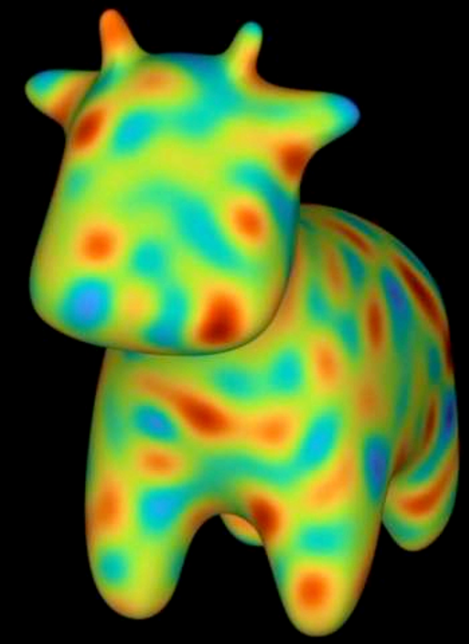
135k unknowns

~4 fps



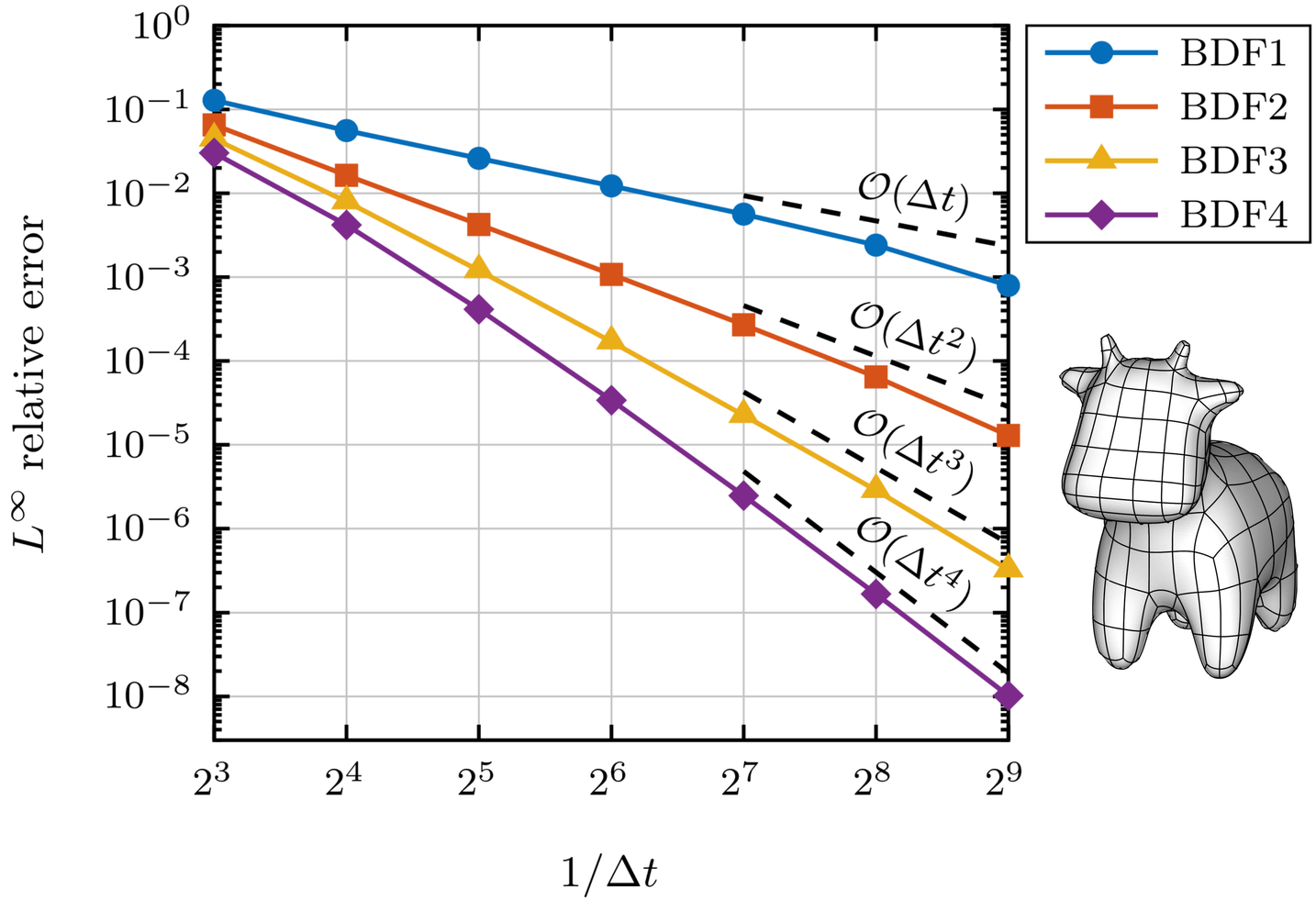


$$\frac{\partial u}{\partial t} = \delta^2 \Delta_\Gamma u + u - (1 + ci)u|u|^2$$



Examples
Reaction–diffusion systems

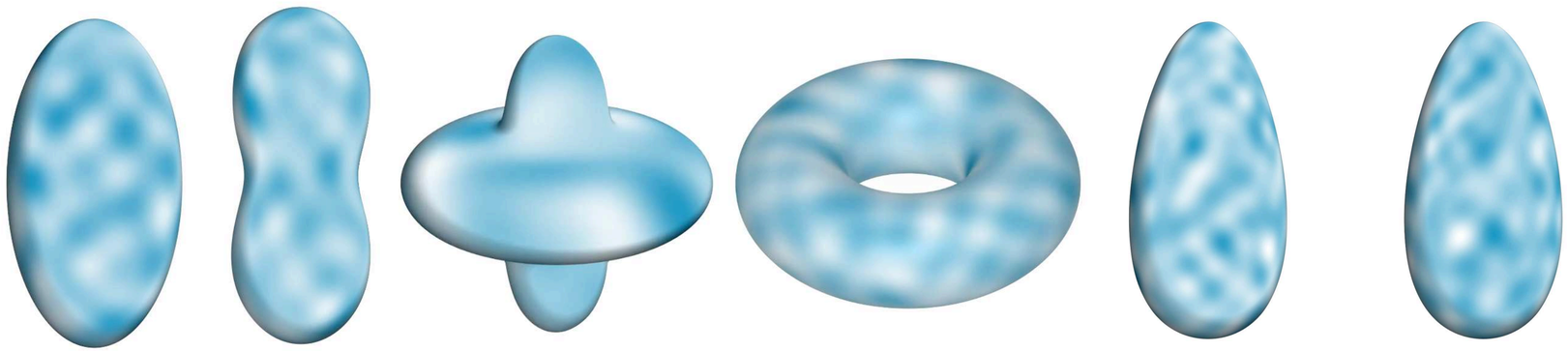
IMEX-BDF methods can achieve high-order accuracy in time.

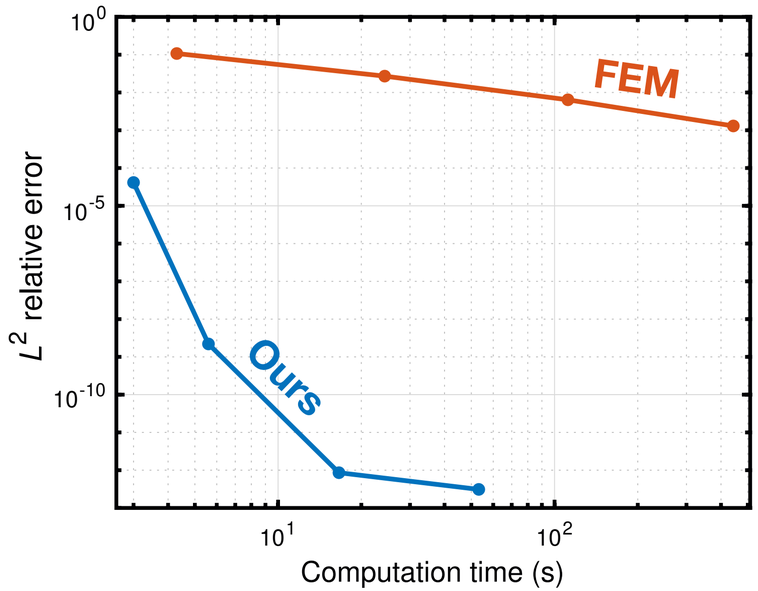


Examples
Cell polarization

Surface curvature can drive cell polarization. Fast solvers allowed us to explore how symmetry is broken over a wide range of parameters.

$$\frac{\partial u}{\partial t} = \delta^2 \Delta_\Gamma u - u + \left(\beta + \frac{u^\nu}{\gamma^\nu + u^\nu}\right)\left(1 - \alpha \int_\Gamma u \, ds\right) \quad \text{on } \Gamma$$






Pearson Miller


Cyrill Muratov


Leslie Greengard


Stas Shvartsman

[Miller*, F.*, Muratov, Greengard, & Shvartsman, 2022]
[Miller, F., Novaga, Shvartsman, & Muratov, 2022]

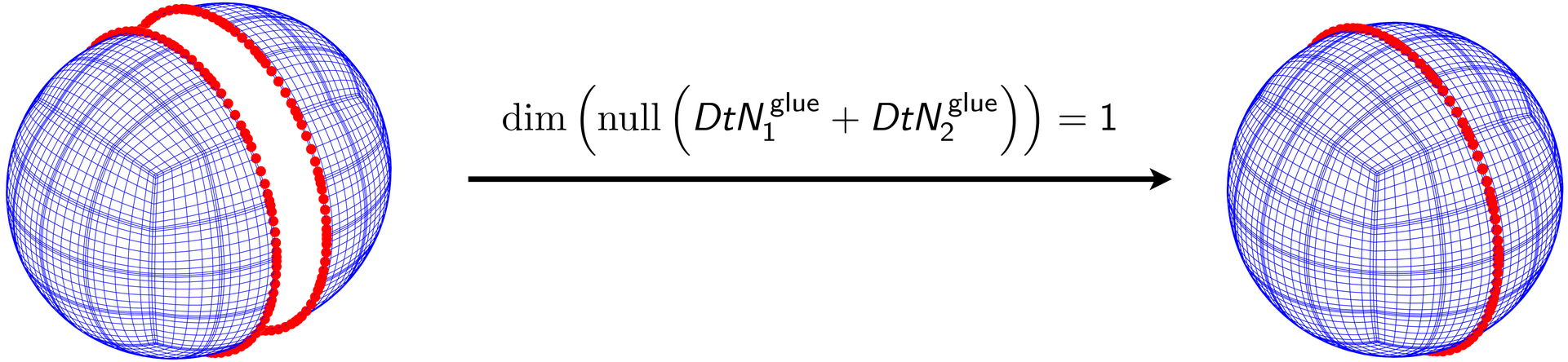
A fast direct solver on surfaces
Laplace–Beltrami and rank deficiency

$$\Delta_\Gamma u = f$$

- The Laplace–Beltrami problem on a closed surface is rank-one deficient, but is uniquely solvable under the mean-zero conditions:

$$\int_\Gamma u = 0 \qquad \text{and} \qquad \int_\Gamma f = 0$$

- In HPS, this rank deficiency is only seen in the final gluing:



- We add the mean-zero constraint to fix the rank deficiency at the top level:

$$\dim\left(\text{null}\left(DtN_1^{\text{glue}} + DtN_2^{\text{glue}} + \mathbf{q}\mathbf{q}^T\right)\right) = 0$$