A fast direct solver for surface PDEs

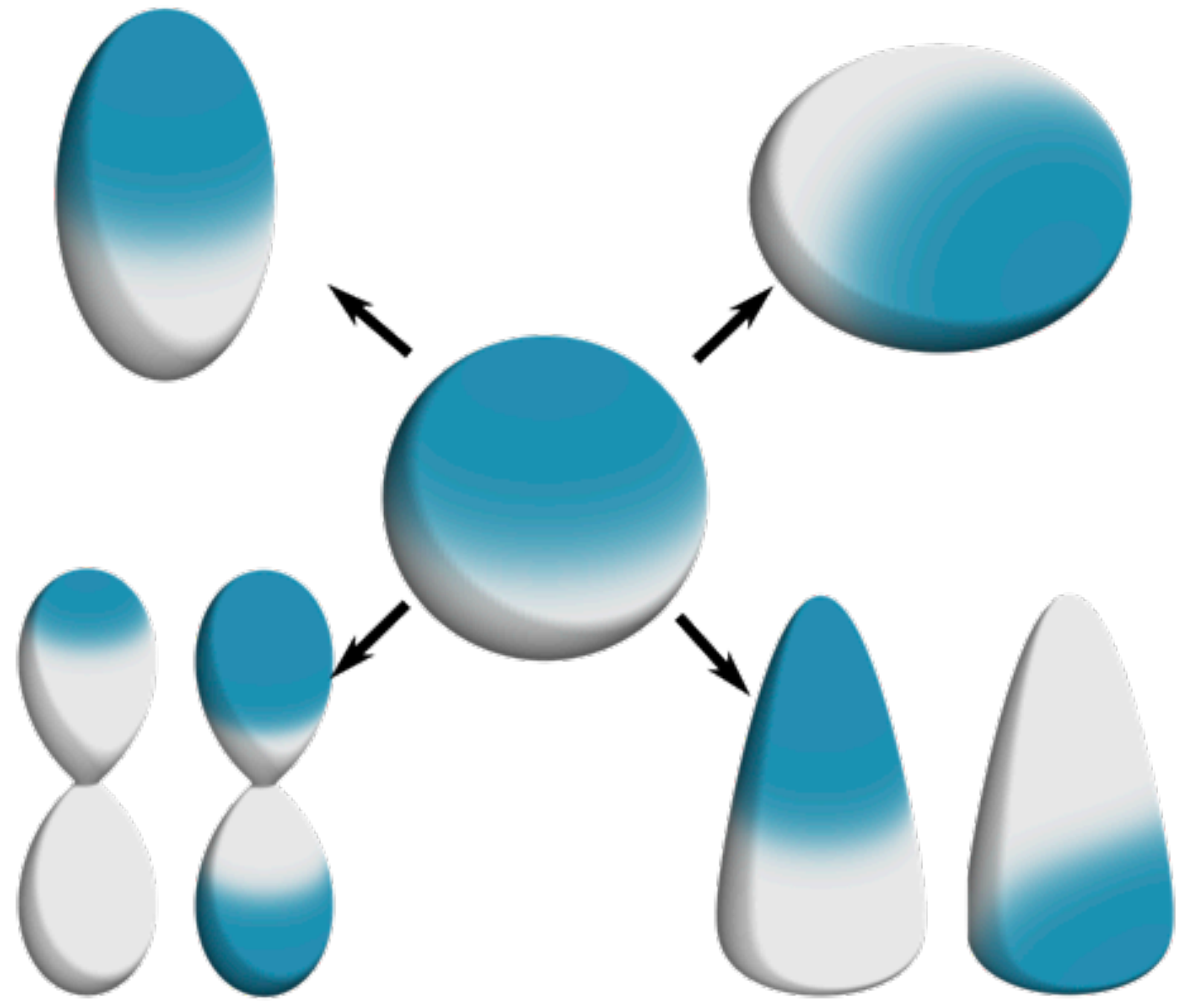
Dan Fortunato
Flatiron Institute

FLATIRON INSTITUTE
Center for Computational Mathematics

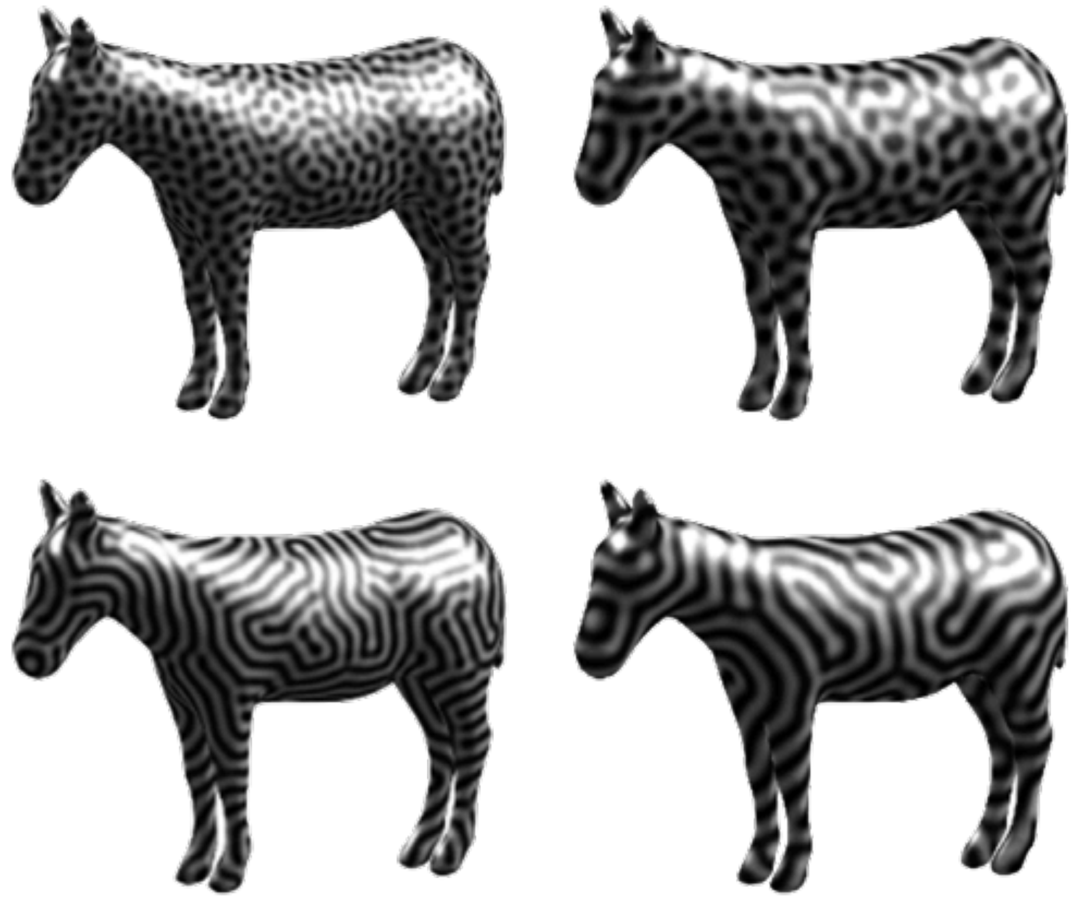
Surface-bound phenomena arise in many applications.

Thin-film hydrodynamics [Saye, 2016]

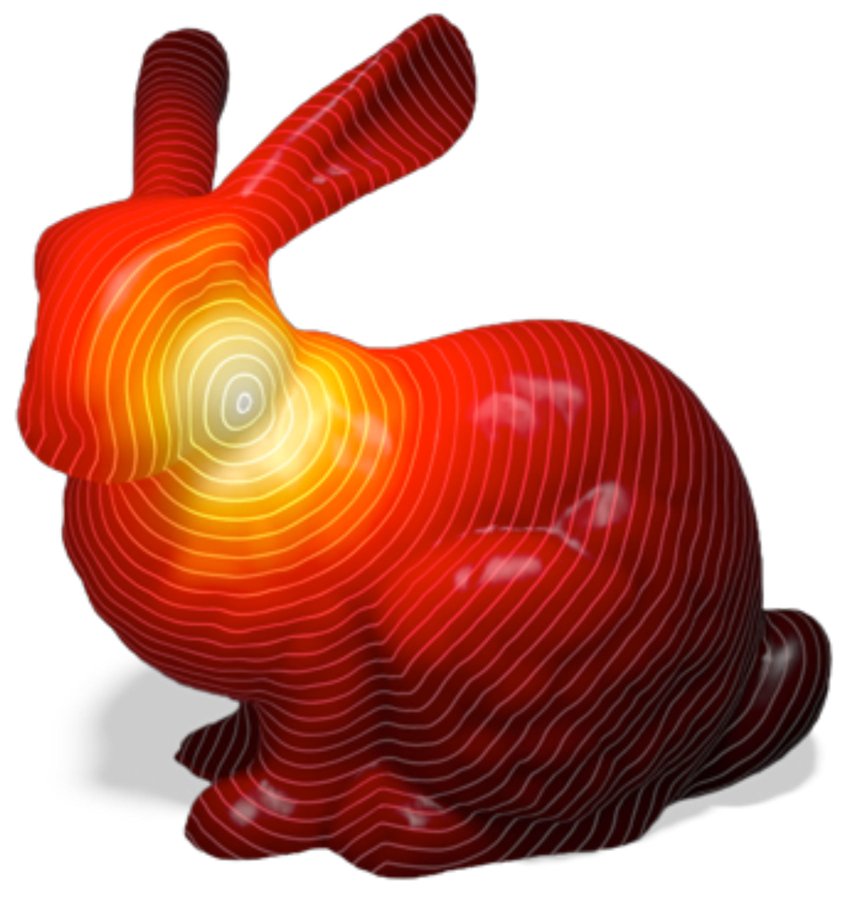
Cell polarization [Miller, F., Muratov, Greengard, Shvartsman, 2021]



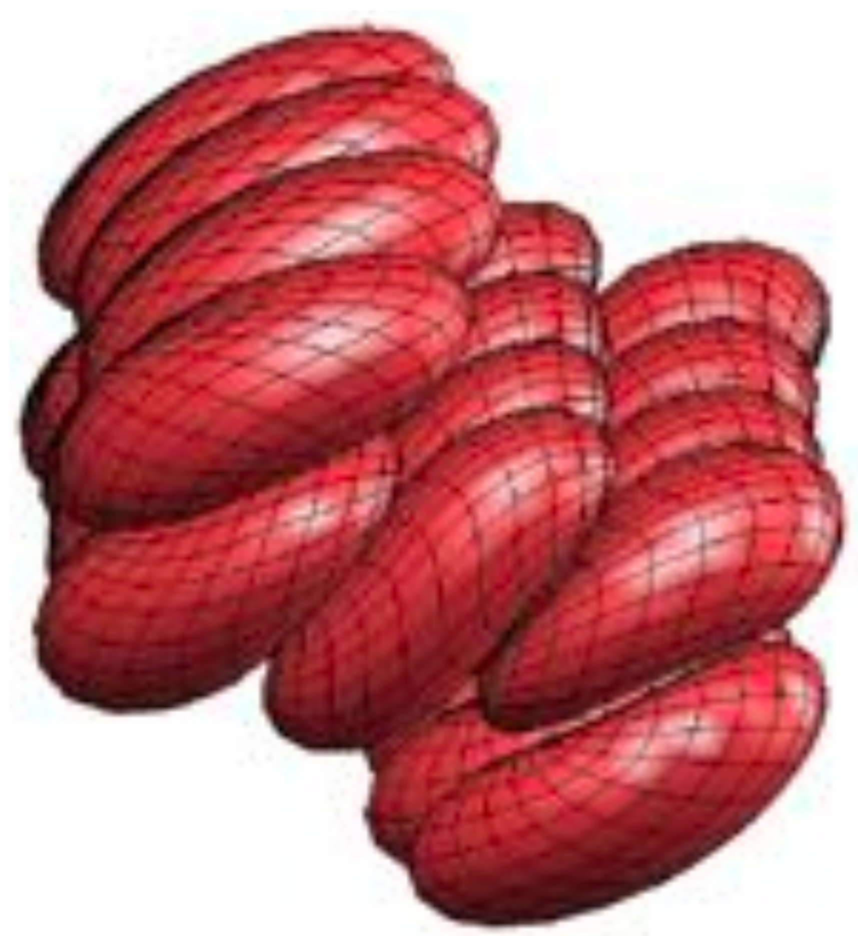
Pattern formation [Jeong, 2017]



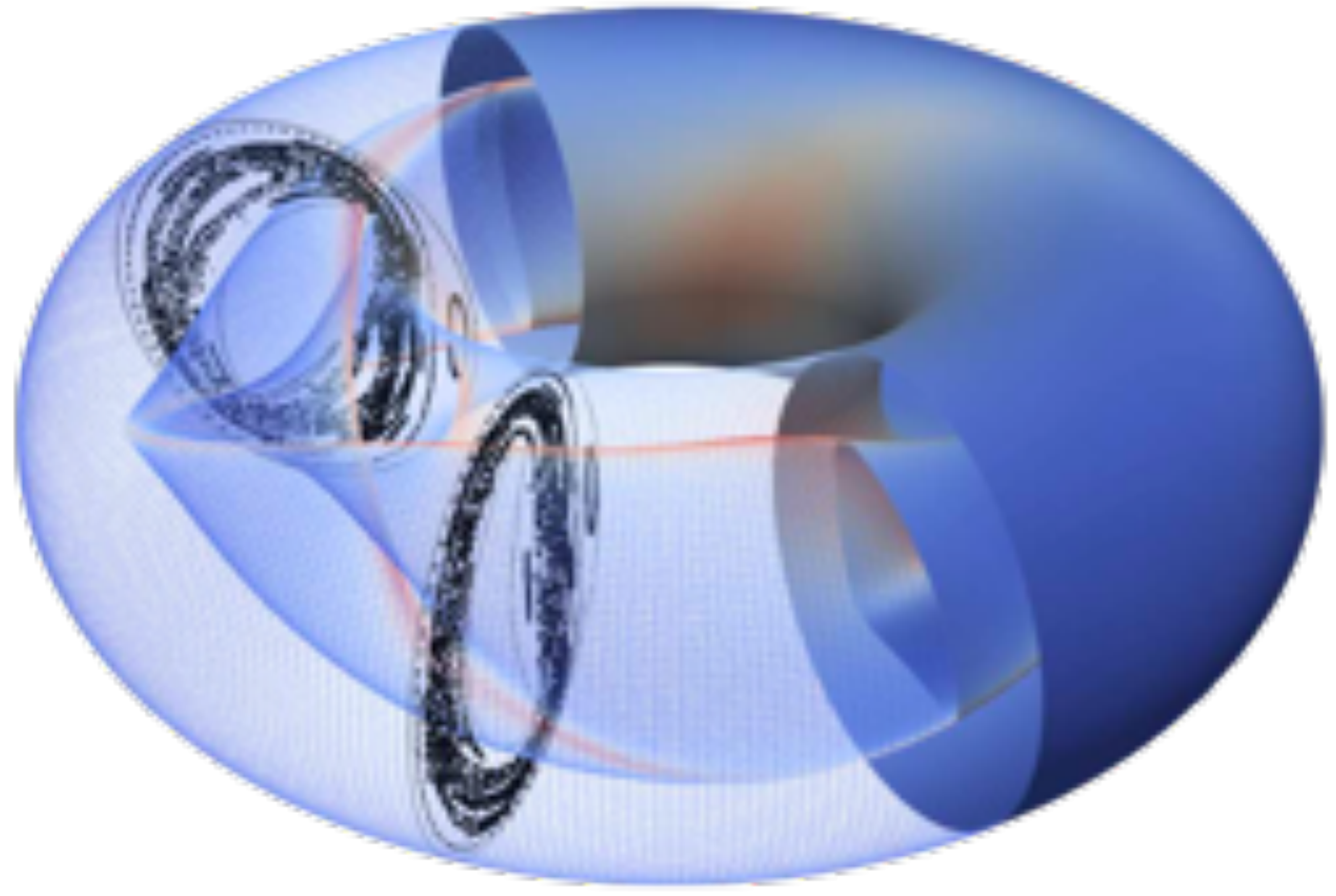
Geodesic distance [Crane et al., 2017]



Vesicle flows [Veerapaneni et al., 2011]



Stellarator design [Malhotra et al., 2019]



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

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

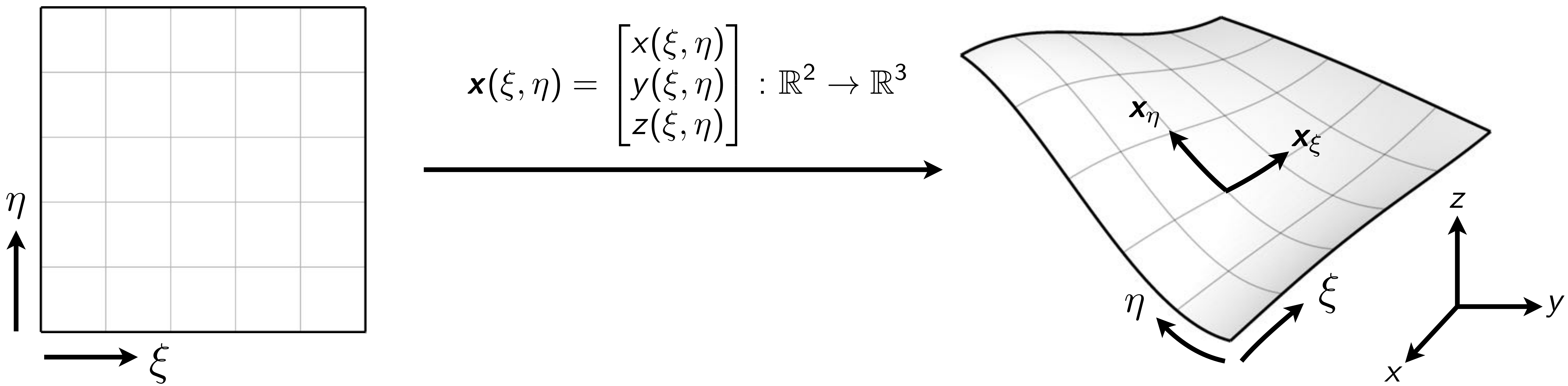
- reaction–diffusion
- heat
- Navier–Stokes

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)

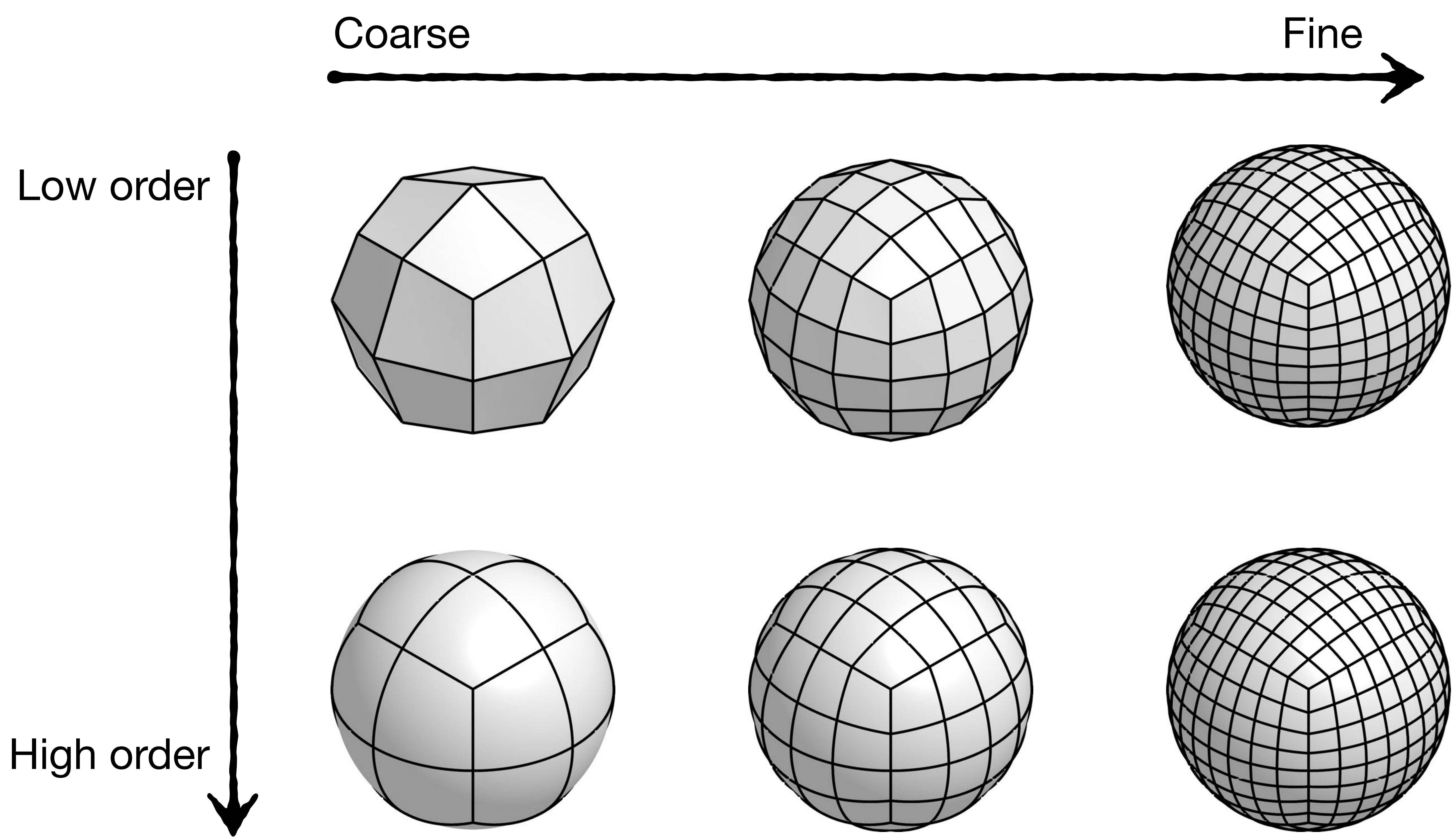


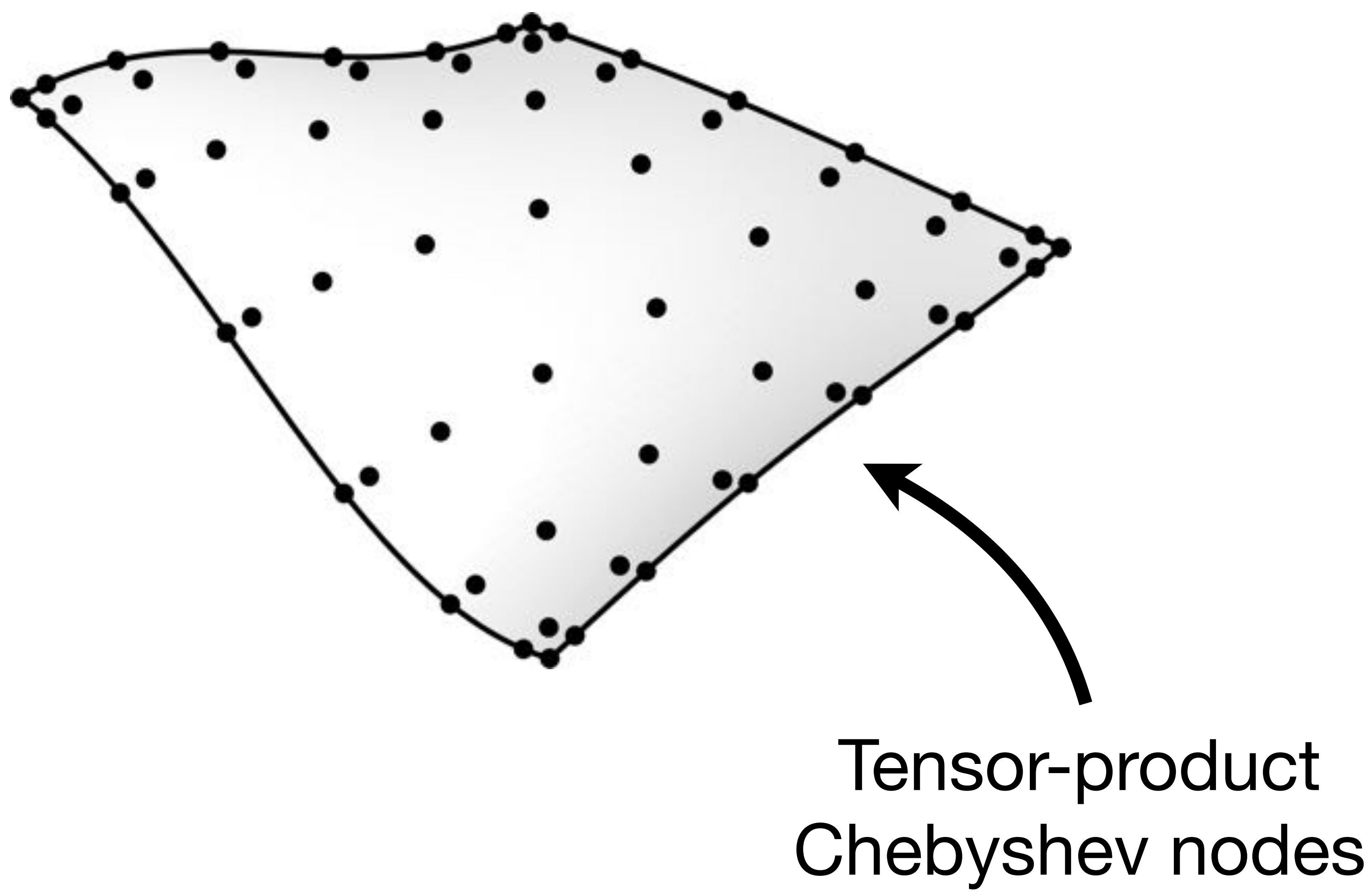
- Metric tensor $g = \begin{bmatrix} \boldsymbol{x}_\xi \cdot \boldsymbol{x}_\xi & \boldsymbol{x}_\xi \cdot \boldsymbol{x}_\eta \\ \boldsymbol{x}_\eta \cdot \boldsymbol{x}_\xi & \boldsymbol{x}_\eta \cdot \boldsymbol{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.
- Surface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

Surface representation

Low-order vs. high-order

- Many ways to represent a surface. Meshes are a good choice for CAD-compatibility.
- High-order elements allow faster convergence to solution.
- Coordinate maps of a patch are discretized via tabulation at Chebyshev nodes.

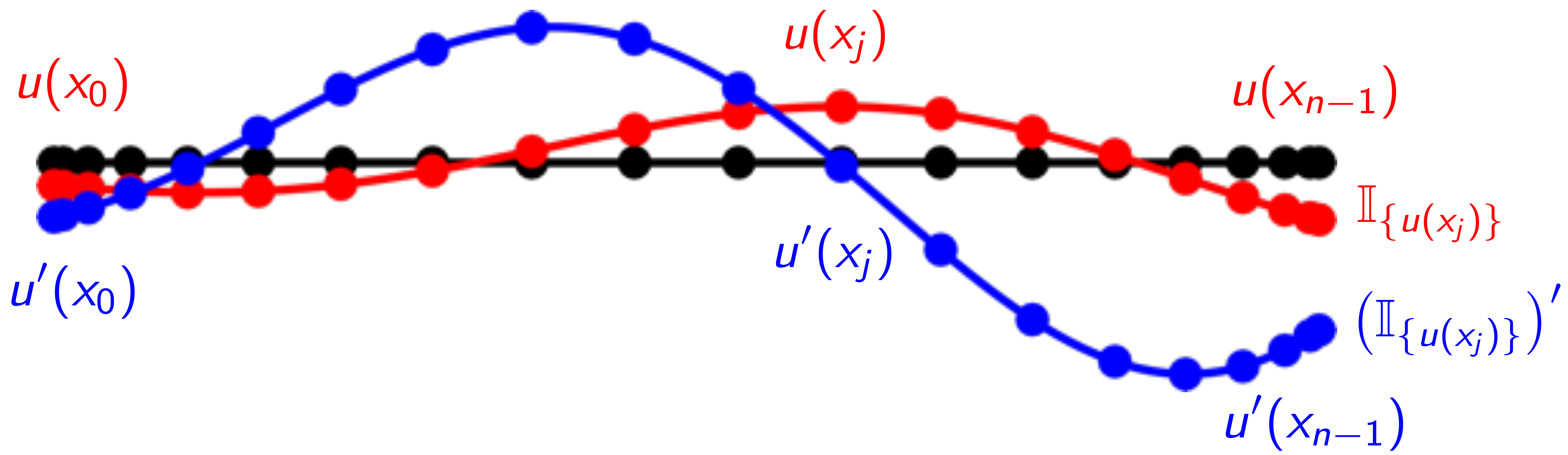




High-order discretization

Spectral collocation

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

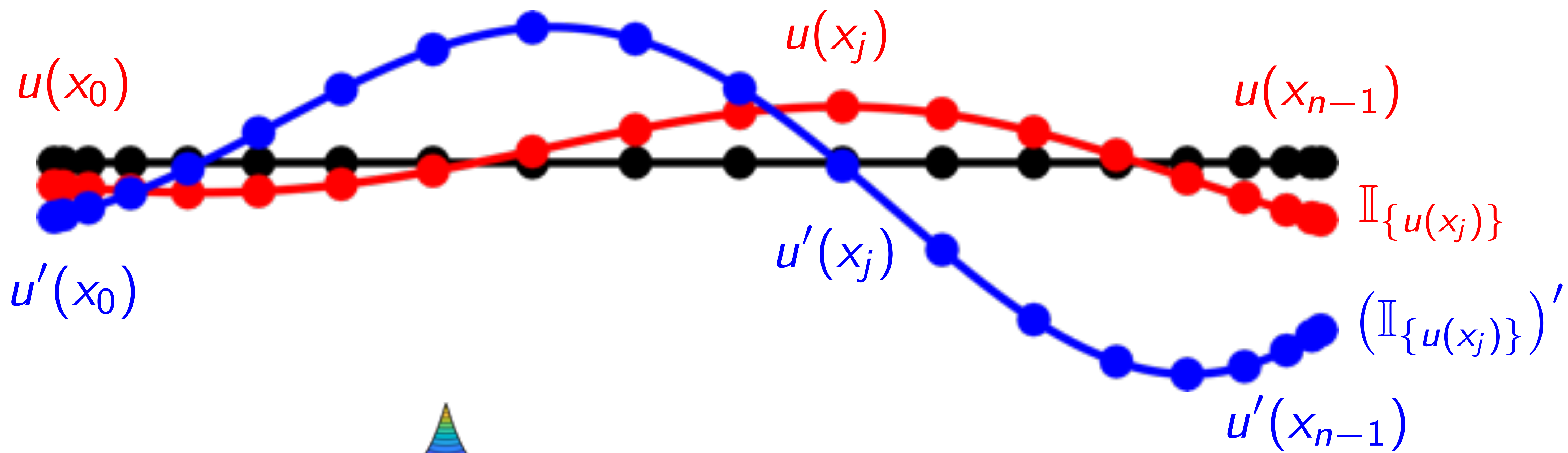


$$D u_j \mapsto u_j'$$

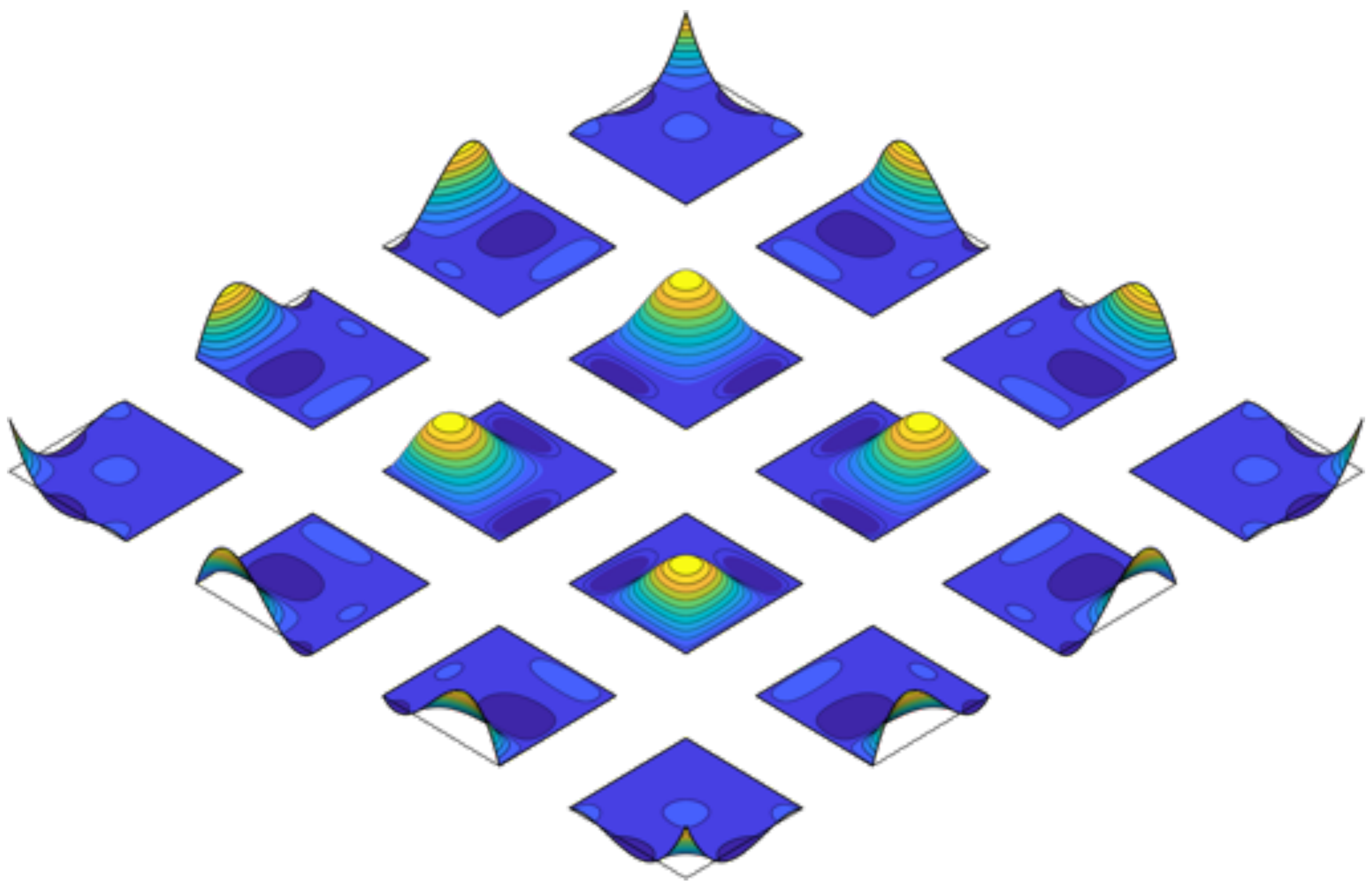
High-order discretization

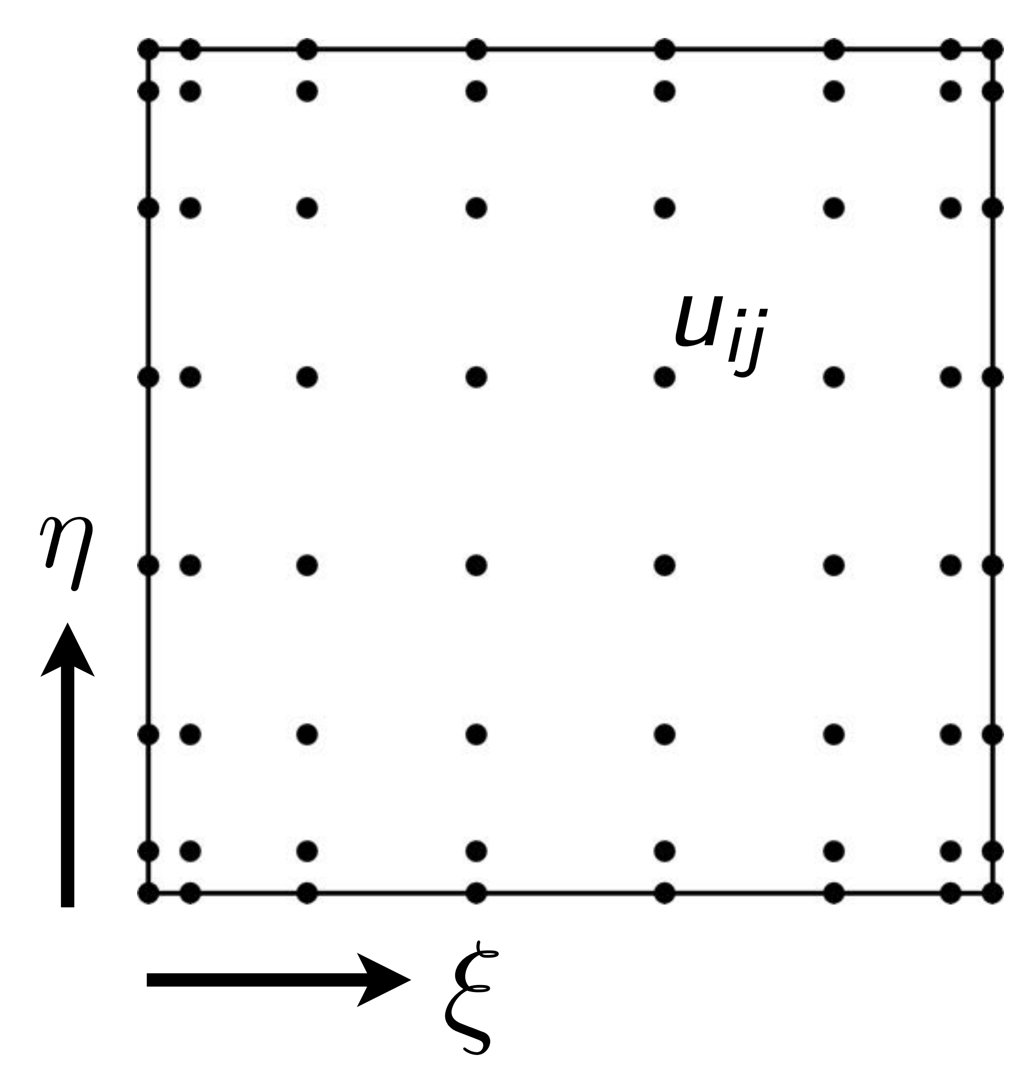
Spectral collocation

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



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



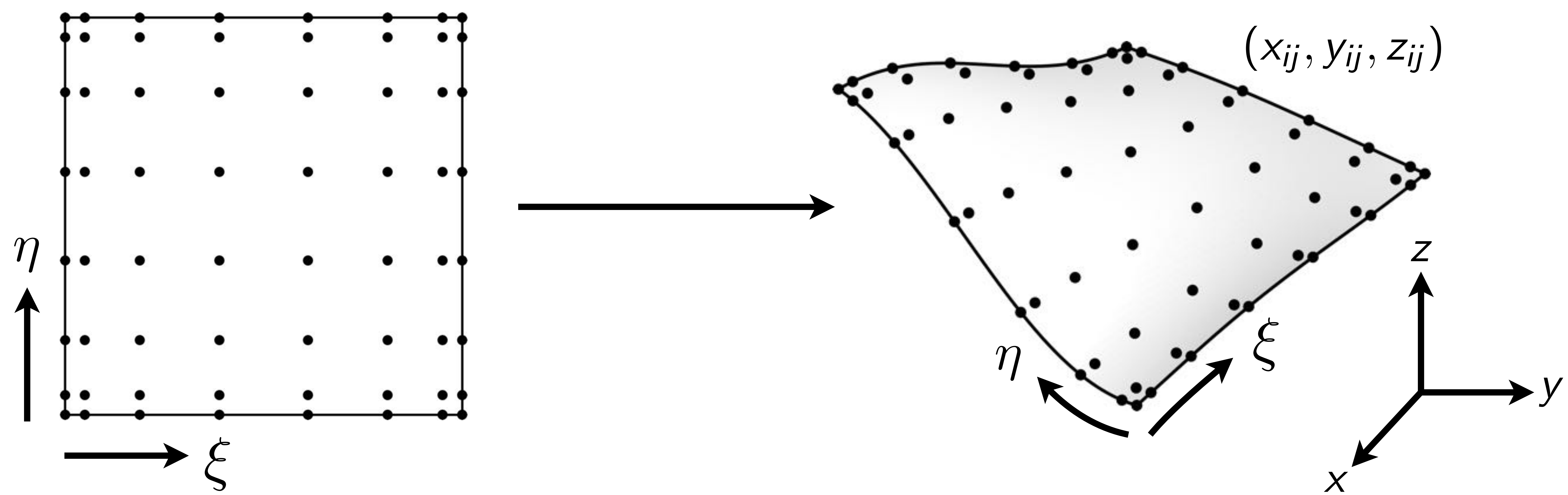


$$(D \otimes I)\,u_{ij} \mapsto \partial u_{ij}/\partial \xi$$

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

High-order discretization
Spectral collocation on a surface

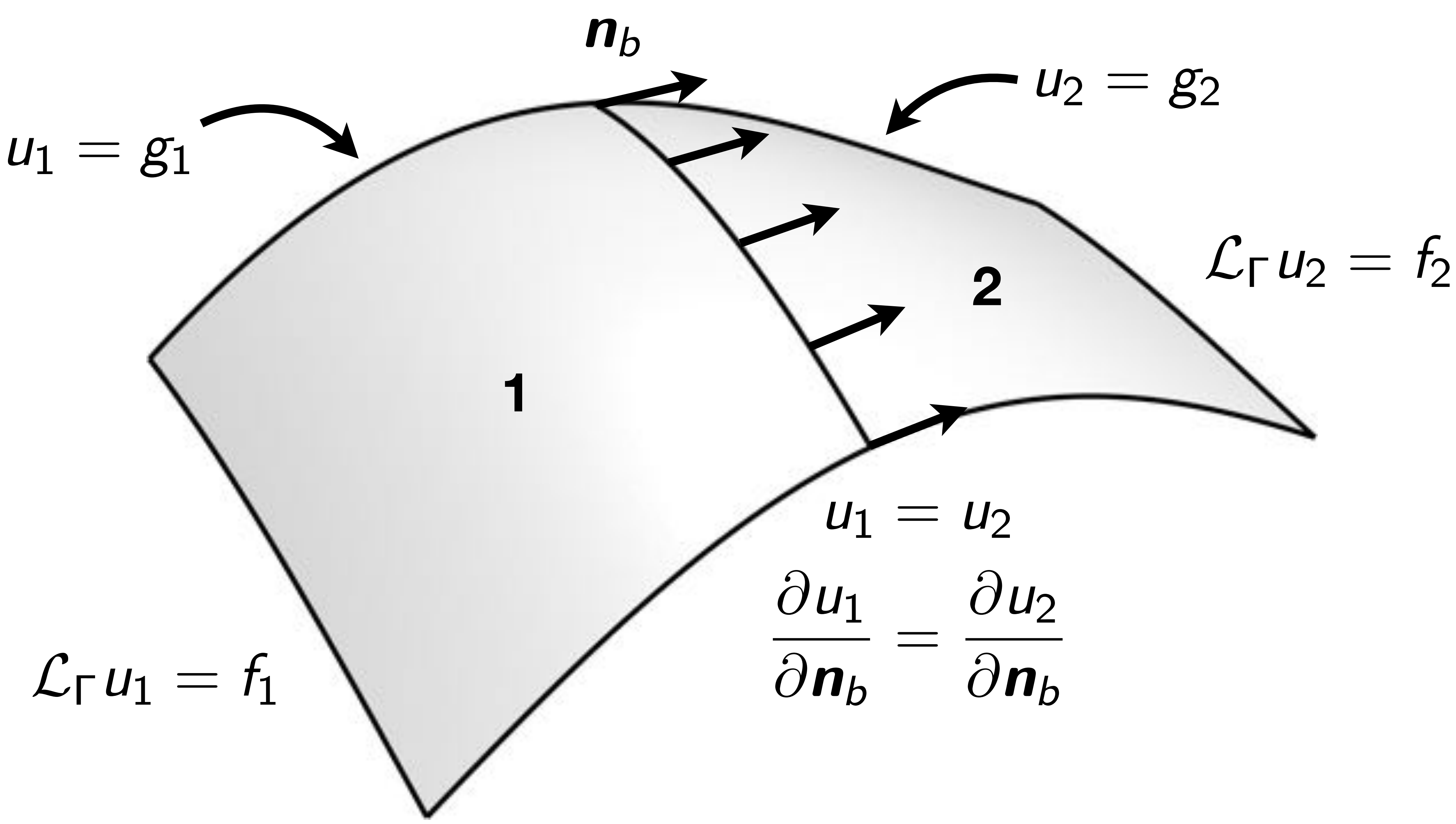
- PDE is discretized through spectral differentiation and pointwise multiplication.



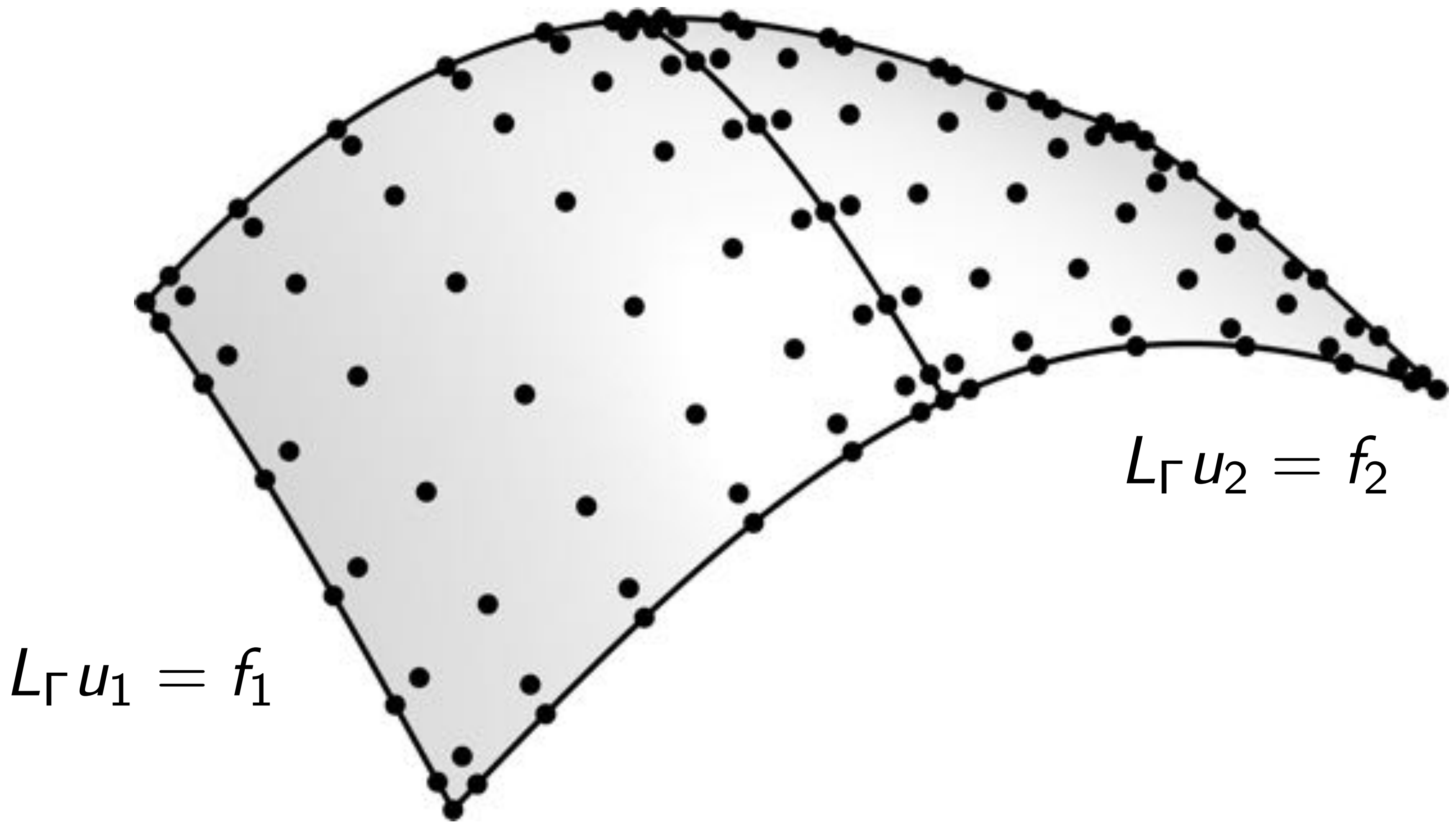
- For example, the discrete tangential x-derivative operator is:

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

- 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.



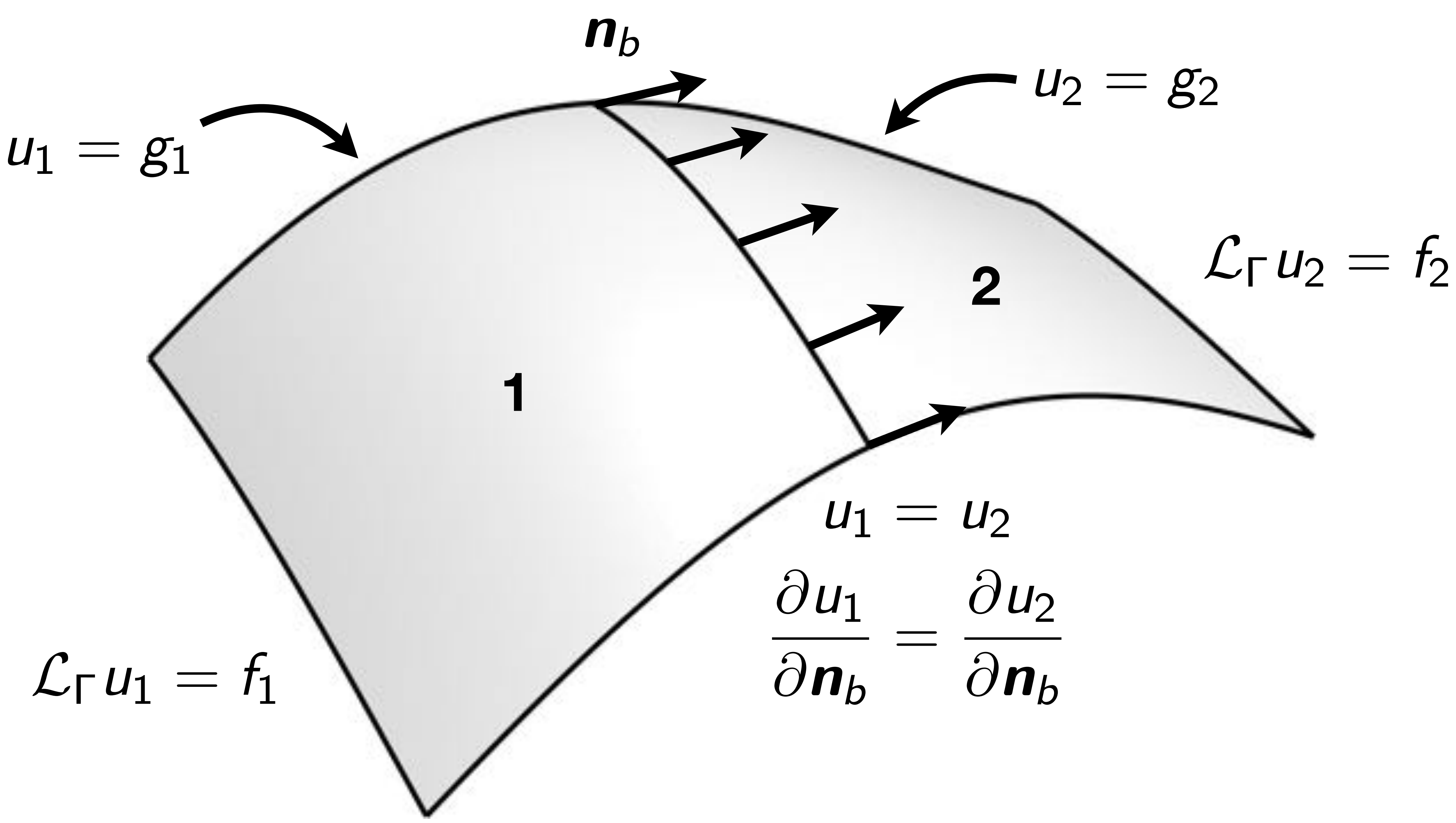
$\boldsymbol{n}_b$
$u_2 = g_2$
$u_1 = g_1$
2
$\mathcal{L}_\Gamma u_2 = f_2$
1
$u_1 = u_2$
$\frac{\partial u_1}{\partial \boldsymbol{n}_b} = \frac{\partial u_2}{\partial \boldsymbol{n}_b}$
$\mathcal{L}_\Gamma u_1 = f_1$

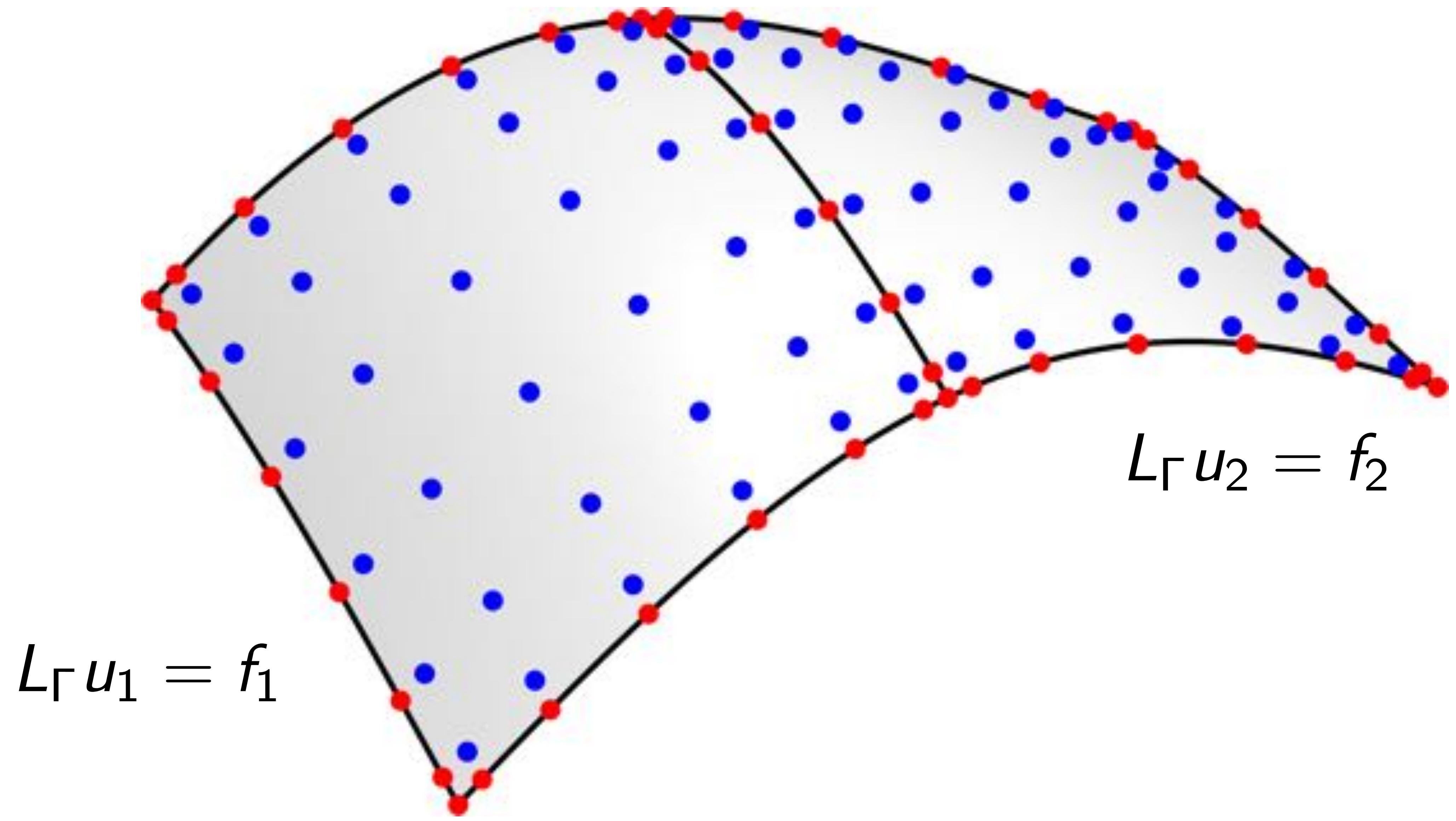


$L_\Gamma u_2 = f_2$
$L_\Gamma u_1 = f_1$

High-order discretization

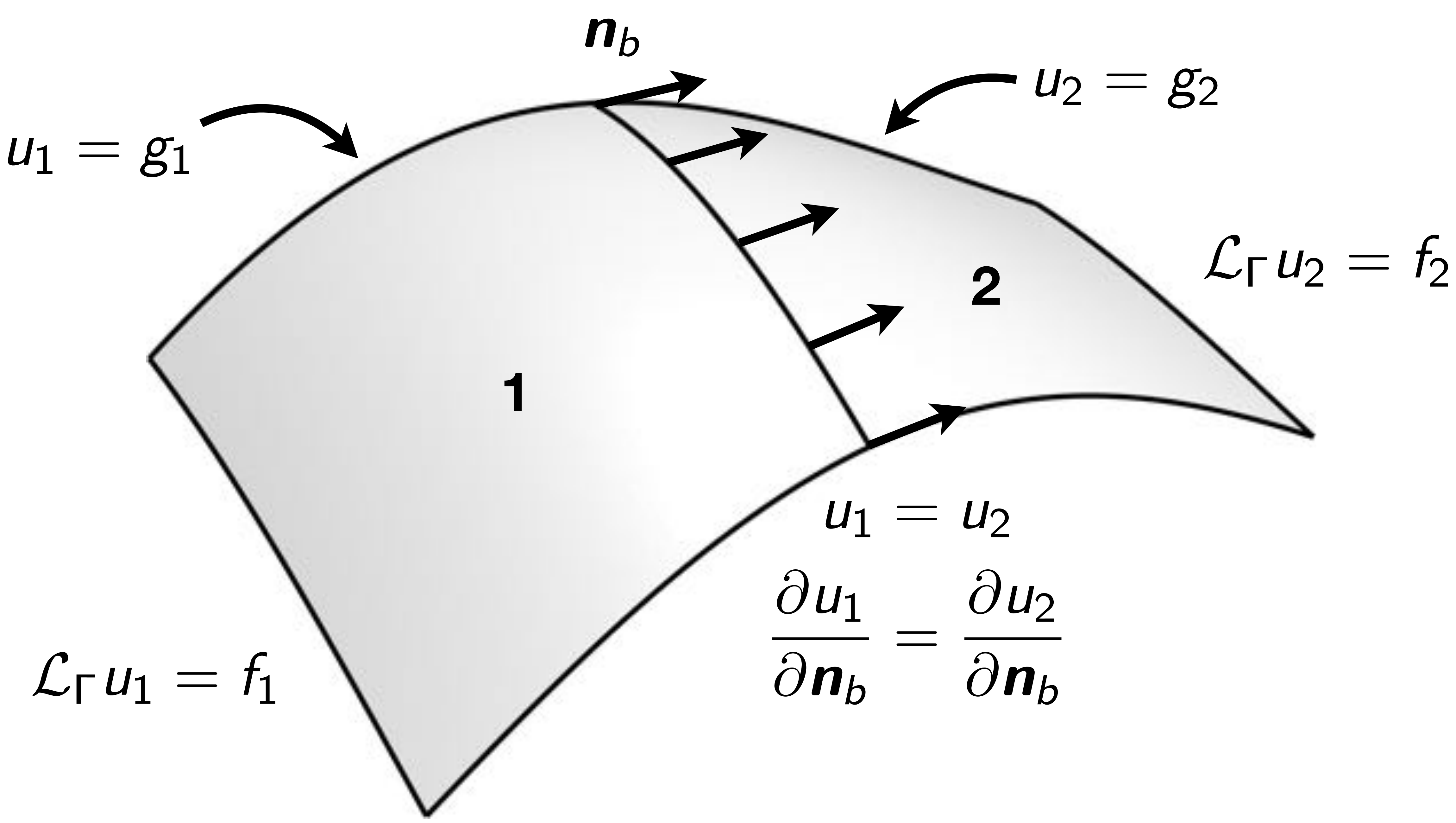
Two glued patches

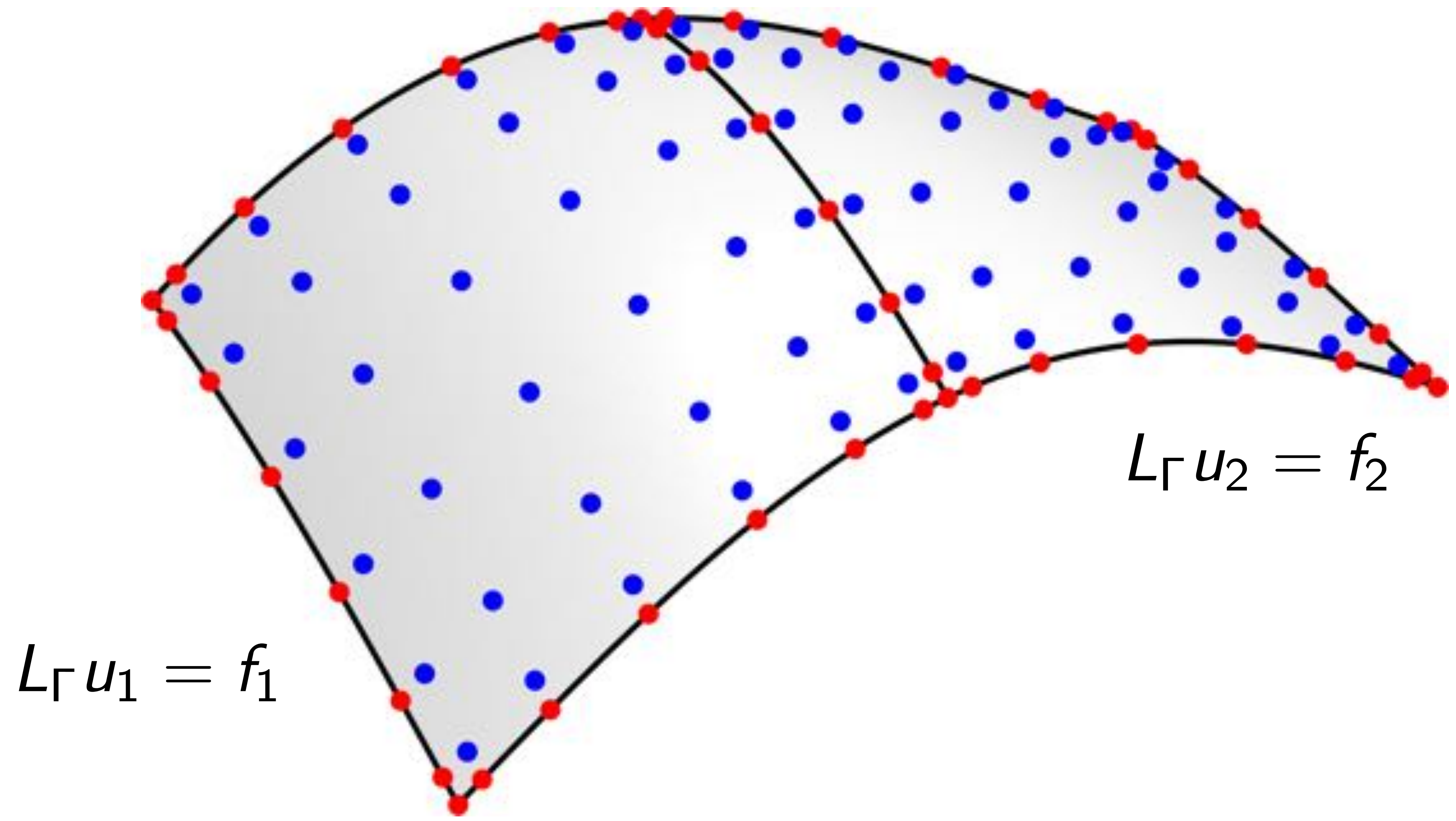




High-order discretization

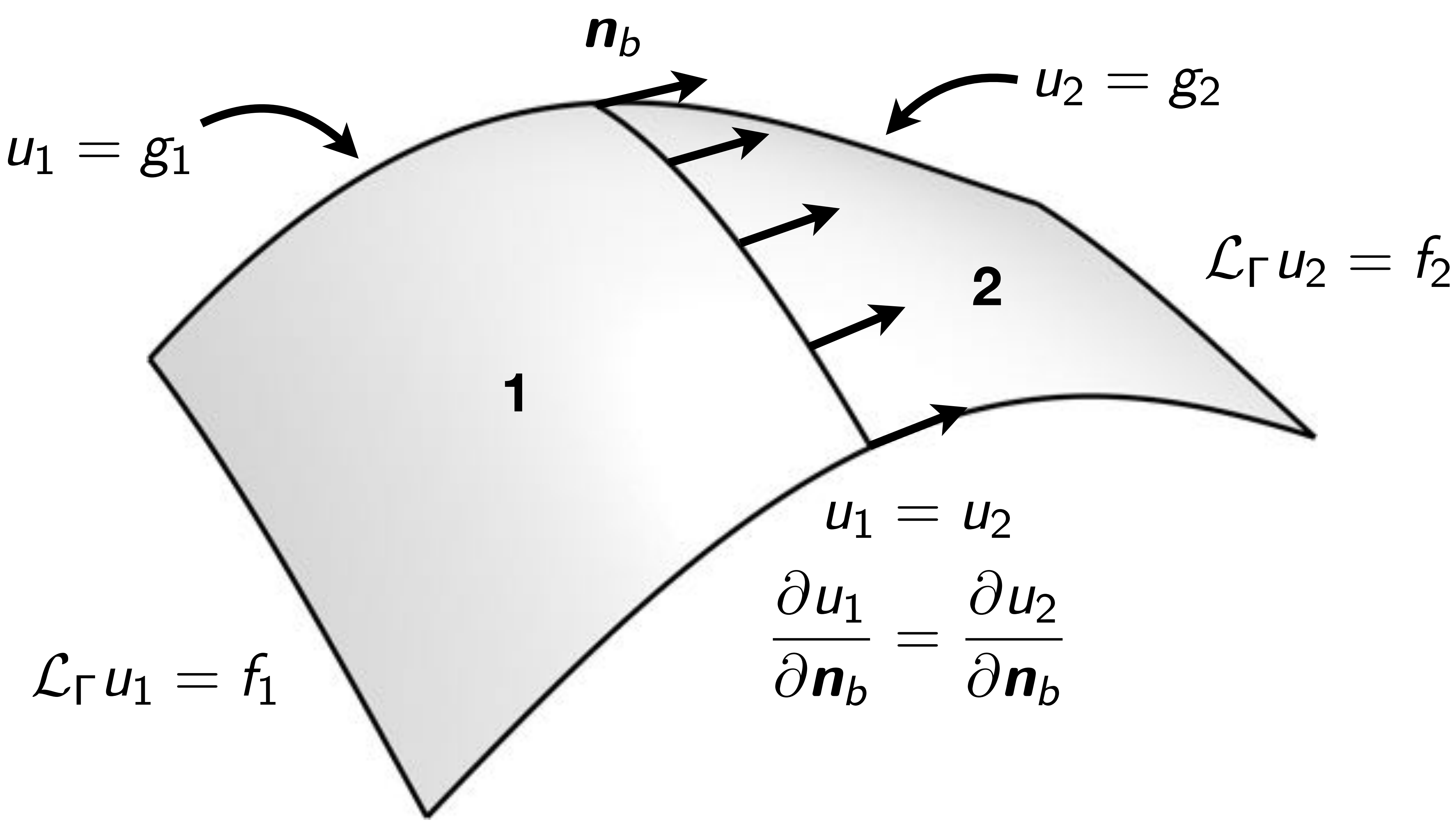
Two glued patches

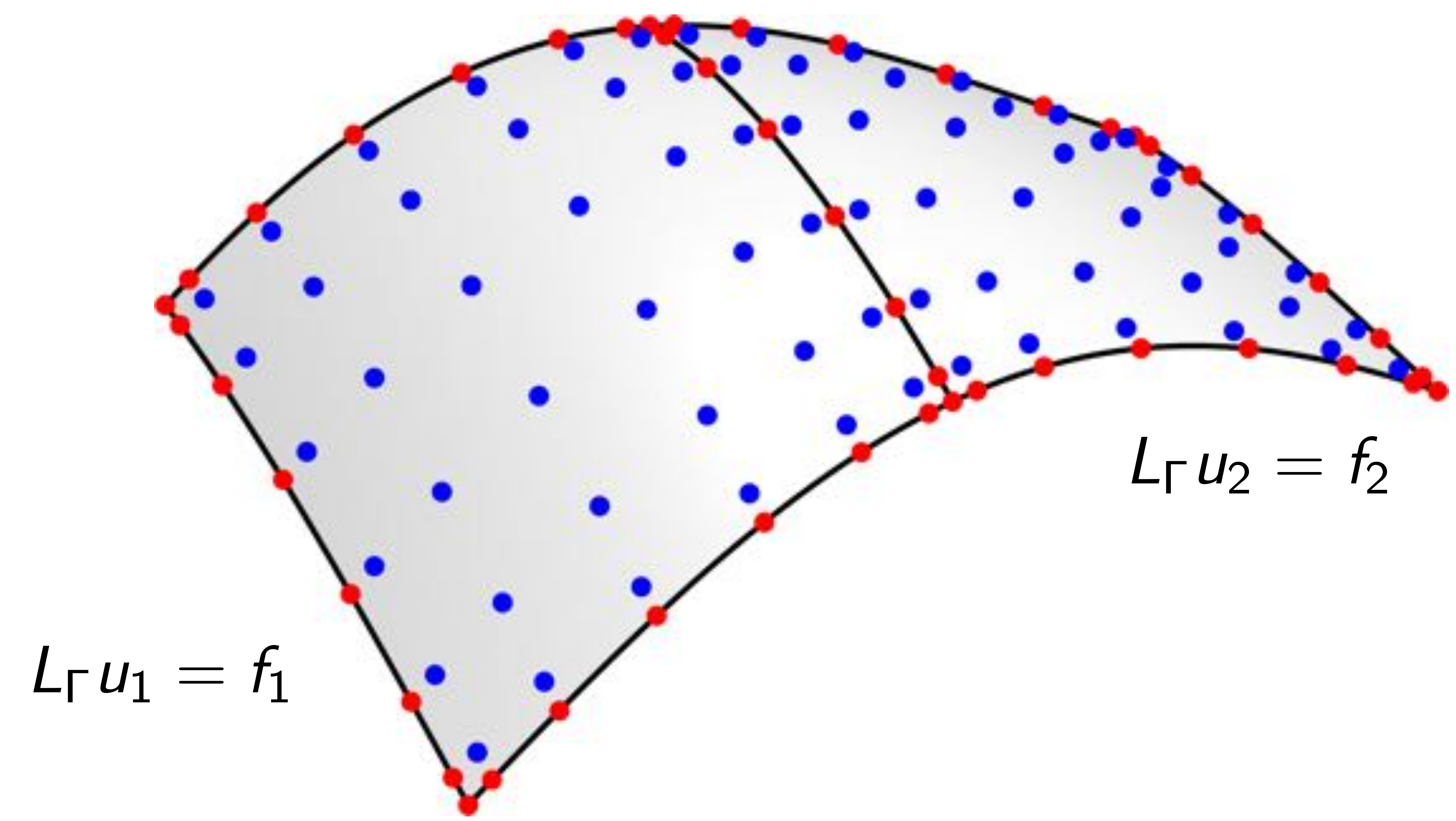




- Know how to do local solves on each element: “Solution operator”

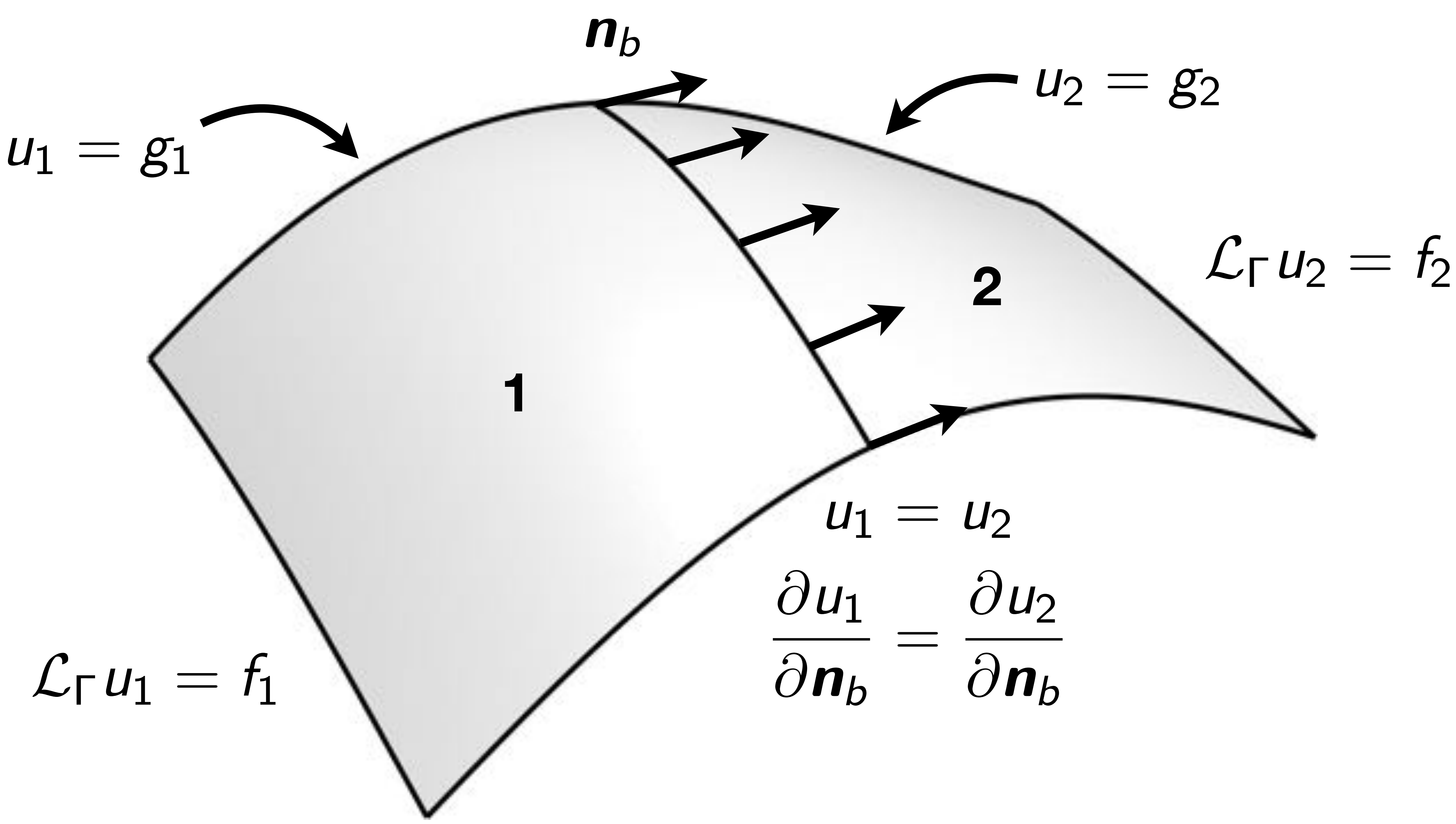
$$S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \qquad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$$

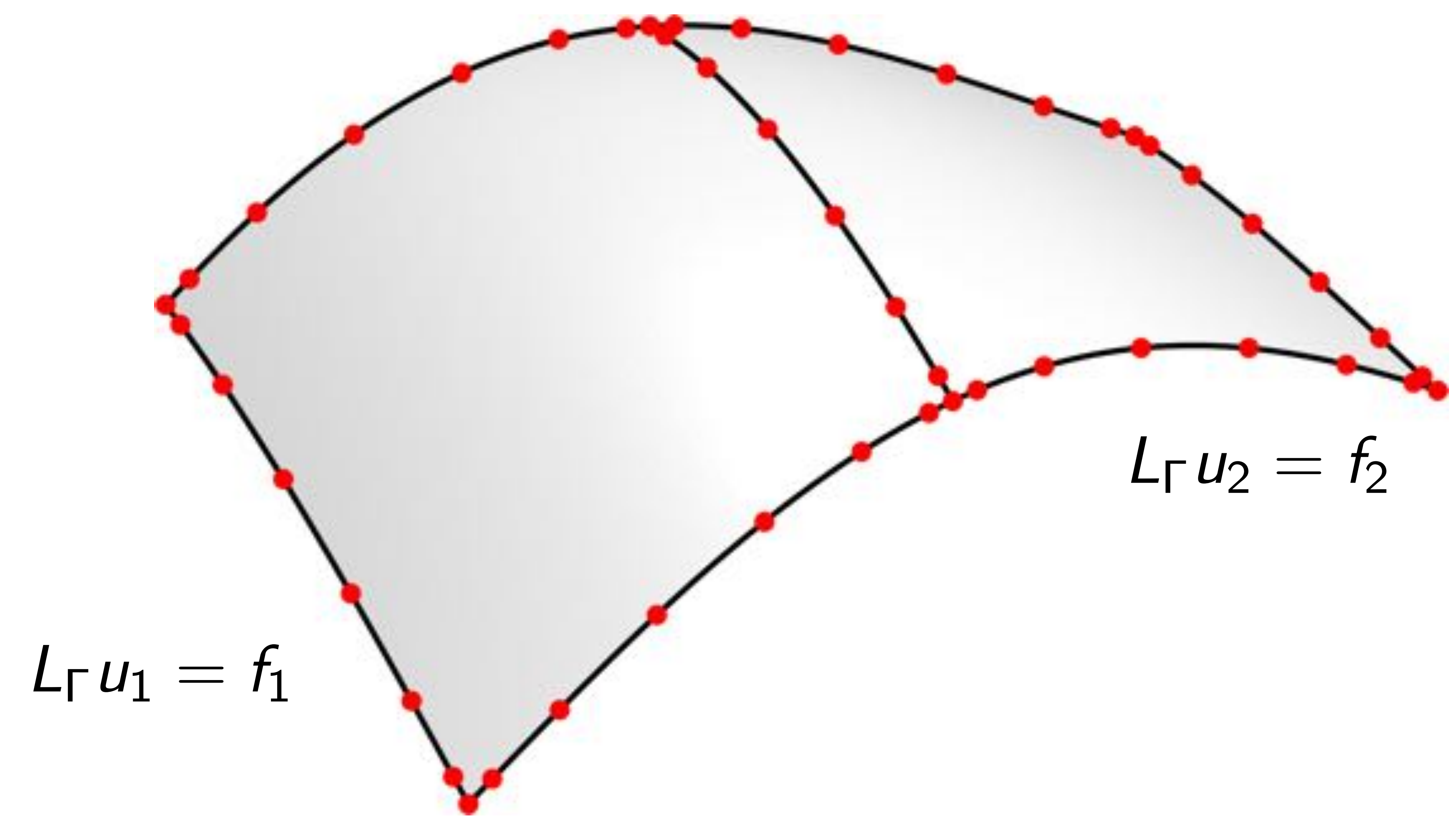




- Know how to do local solves on each element: "Solution operator" $S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \quad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$
- Know how information flows out of each element: "Dirichlet-to-Neumann map" $DtN_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial \boldsymbol{n}_b} \quad DtN_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial \boldsymbol{n}_b}$

High-order discretization
Two glued patches



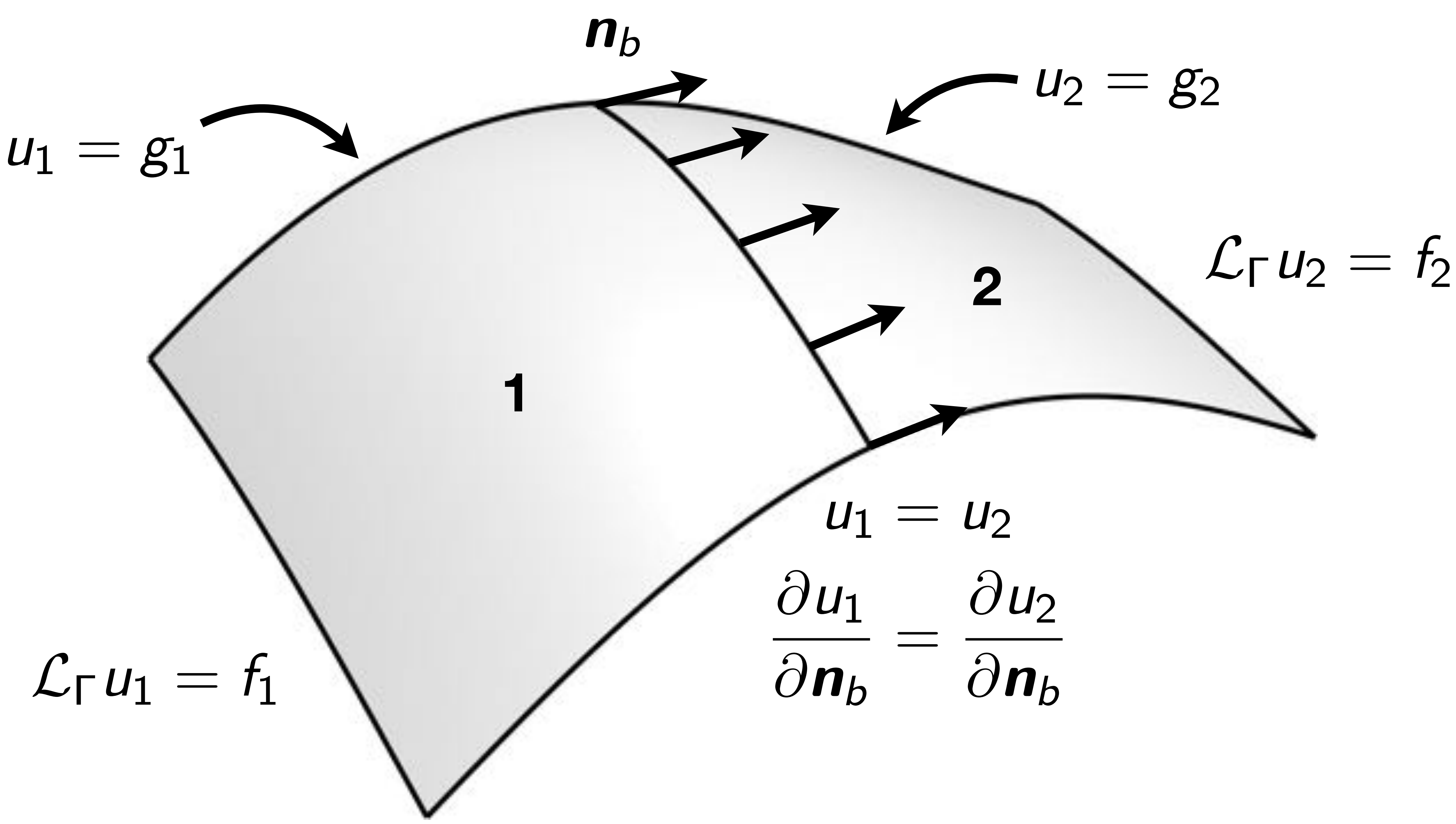


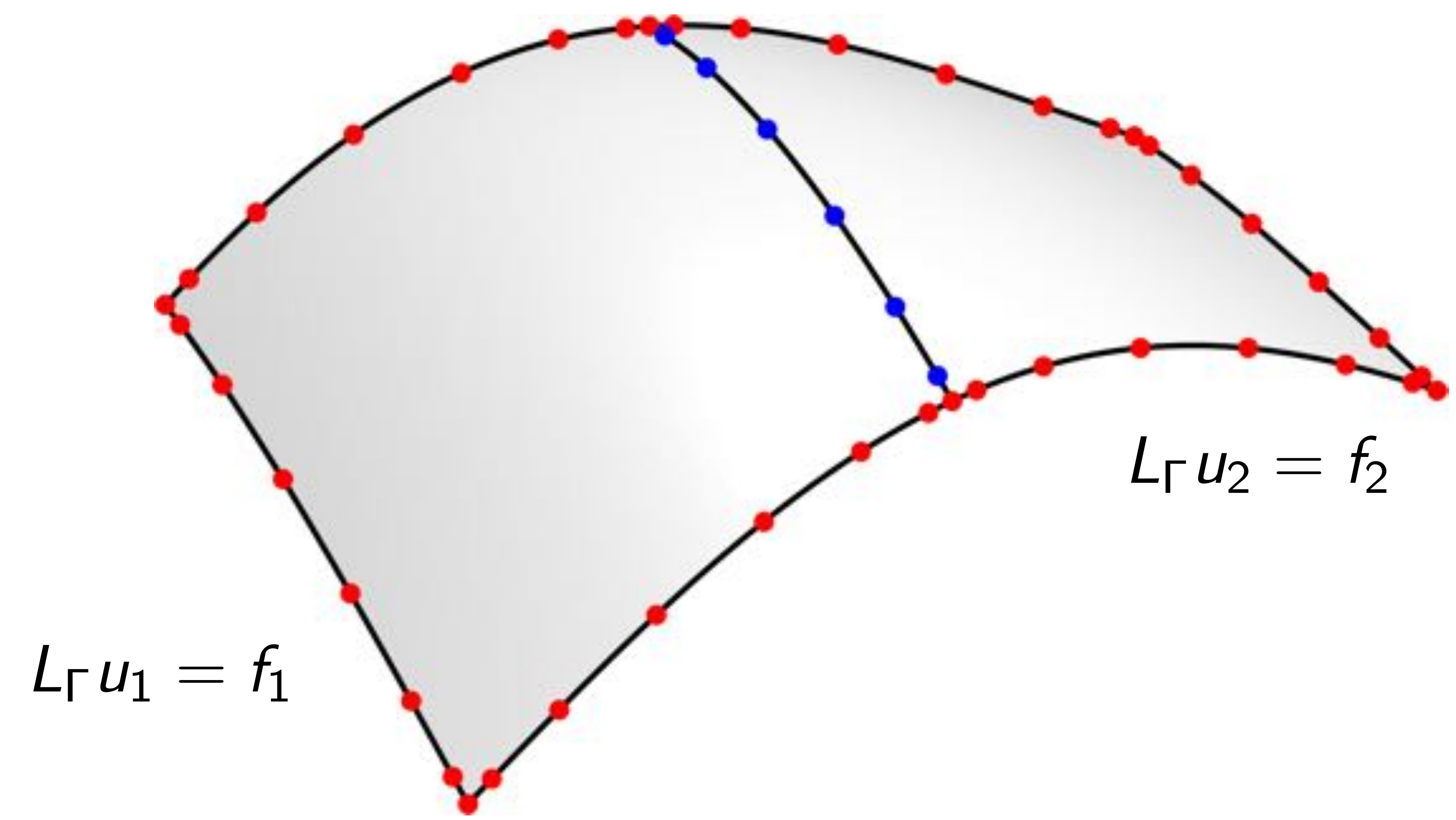
- Know how to do local solves on each element: "Solution operator"
- Know how information flows out of each element: "Dirichlet-to-Neumann map"

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

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

High-order discretization
Two glued patches





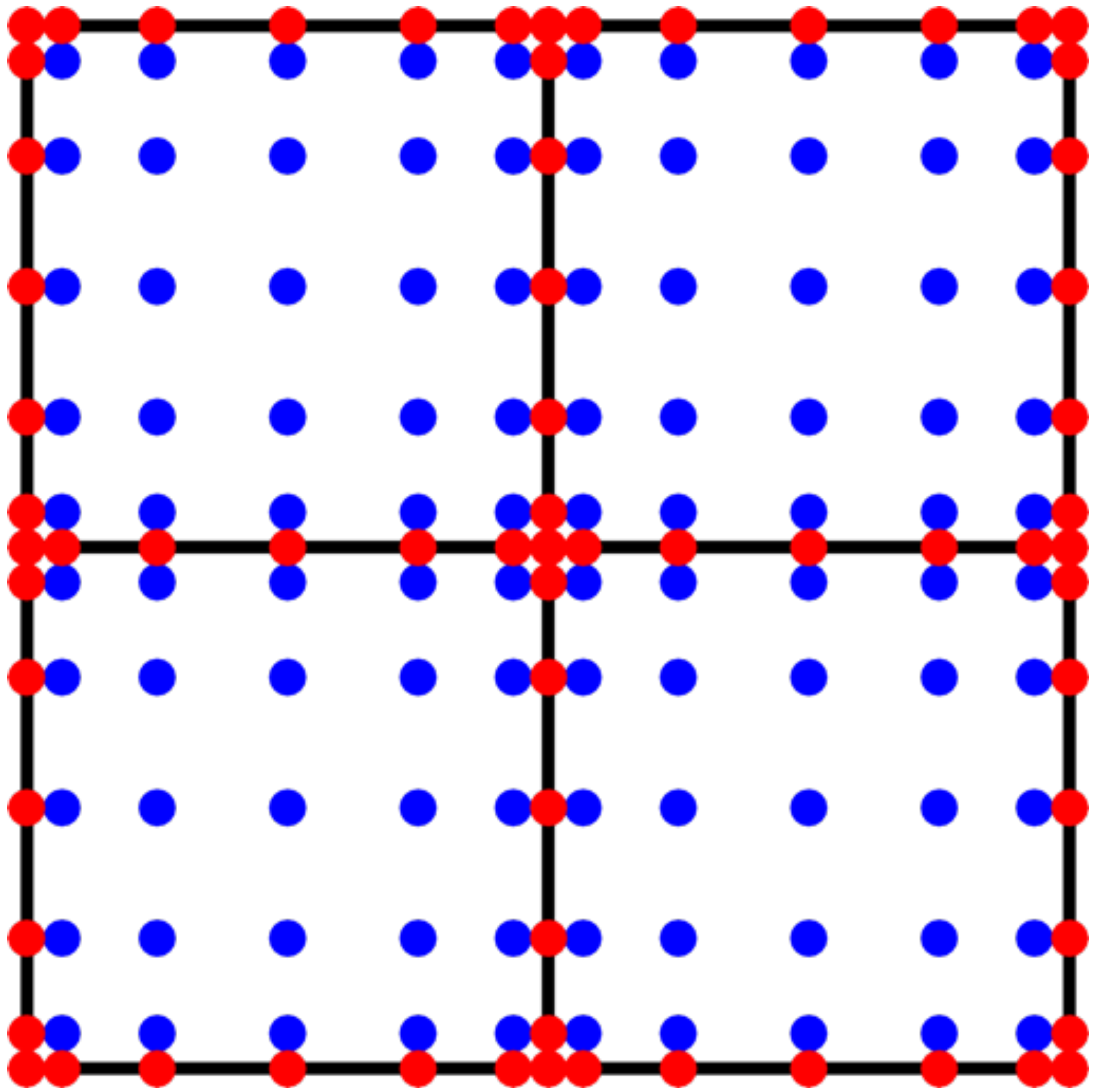
- Know how to do local solves on each element: "Solution operator" $S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \qquad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$
- Know how information flows out of each element: "Dirichlet-to-Neumann map" $DtN_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial \boldsymbol{n}_b} \qquad DtN_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial \boldsymbol{n}_b}$
- Take Schur complement to eliminate interior degrees of freedom:

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

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

<u>Key idea</u>: Recursively glue elements together in a hierarchy.



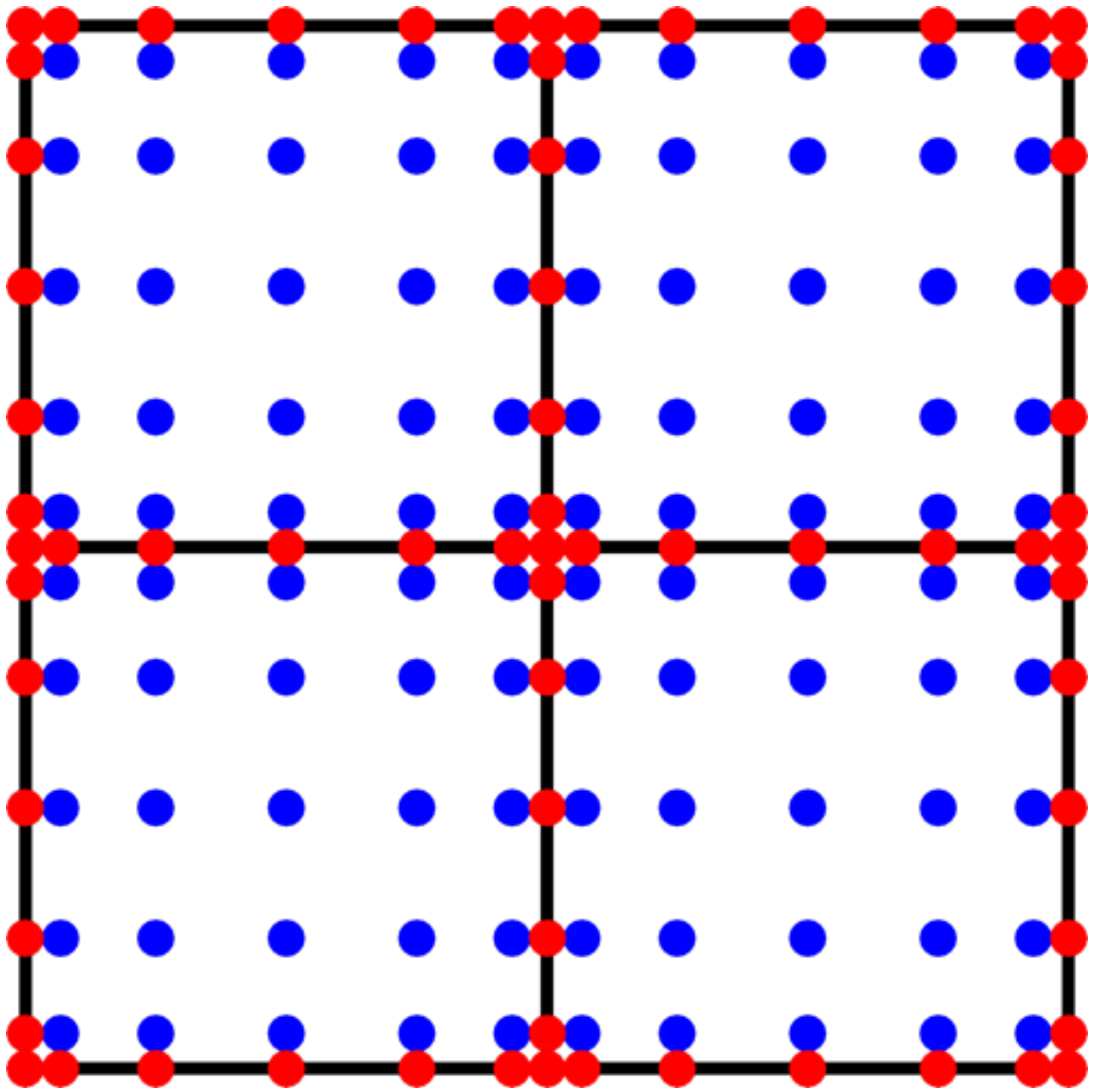
Gunnar Martinsson

Adrianna Gillman

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

<ins>Key idea</ins>: Recursively glue elements together in a hierarchy.

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

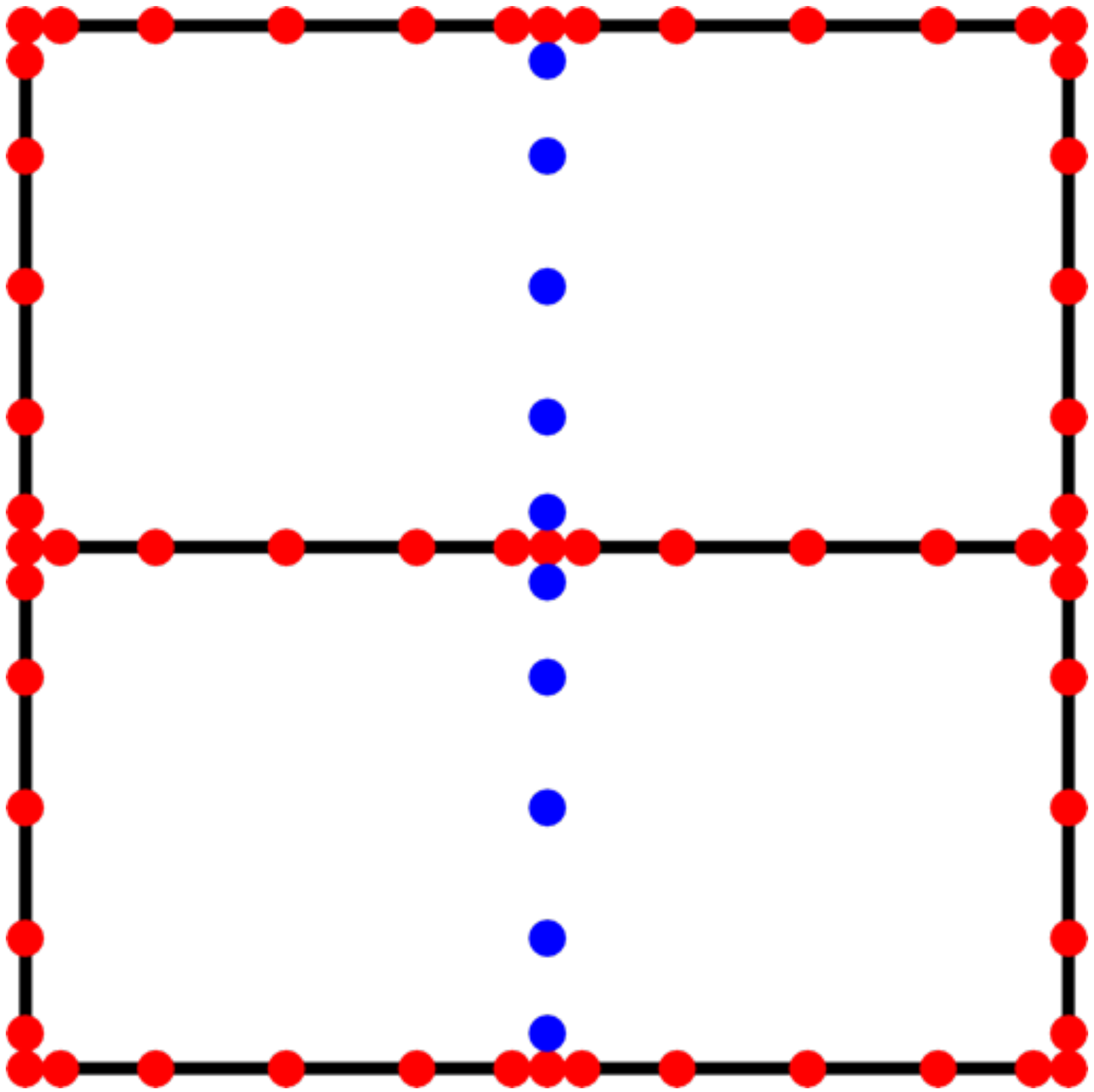


Gunnar Martinsson

Adrianna Gillman

Key idea: Recursively glue elements together in a hierarchy.

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

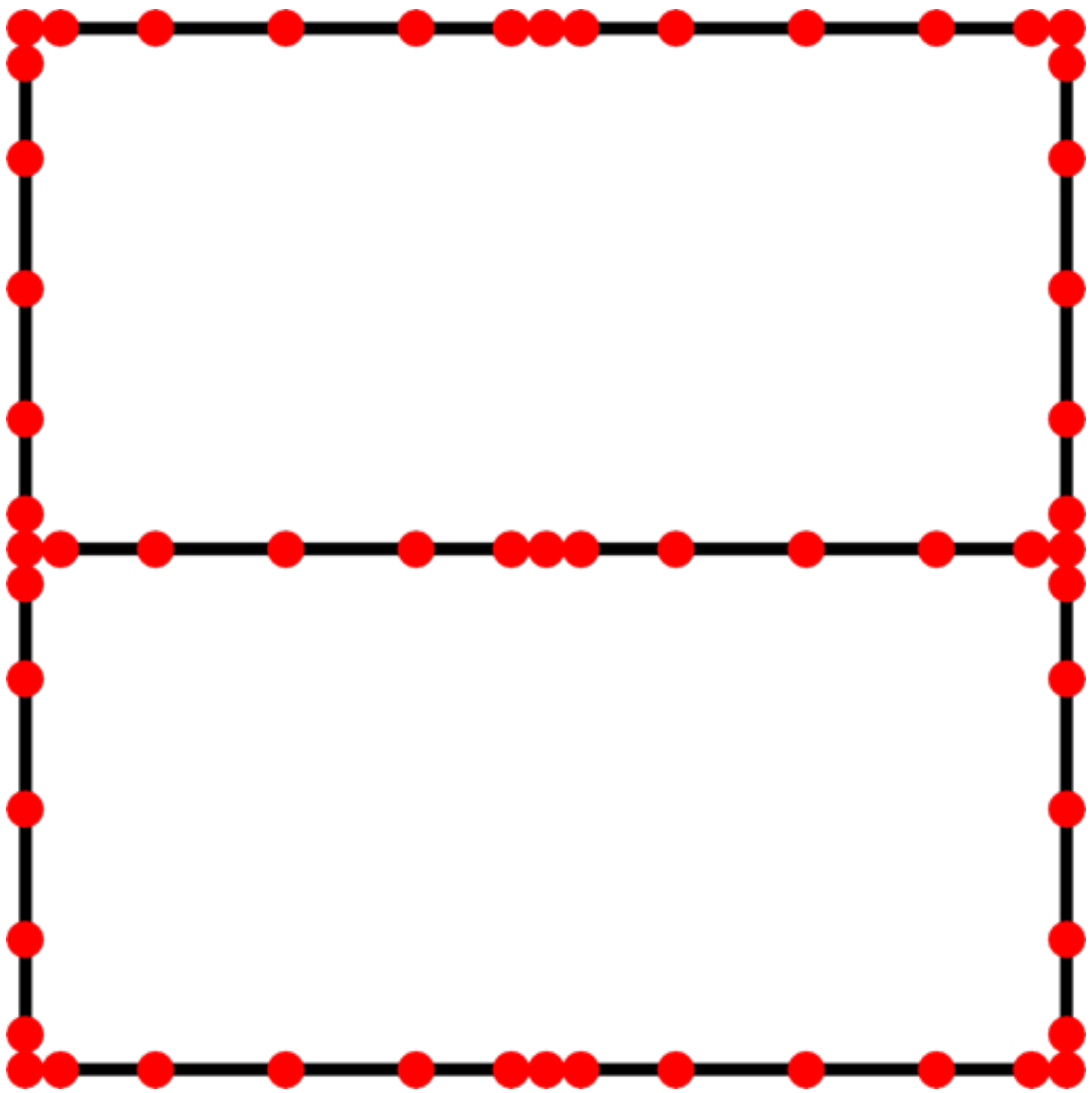


Gunnar Martinsson

Adrianna Gillman

Key idea: Recursively glue elements together in a hierarchy.

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



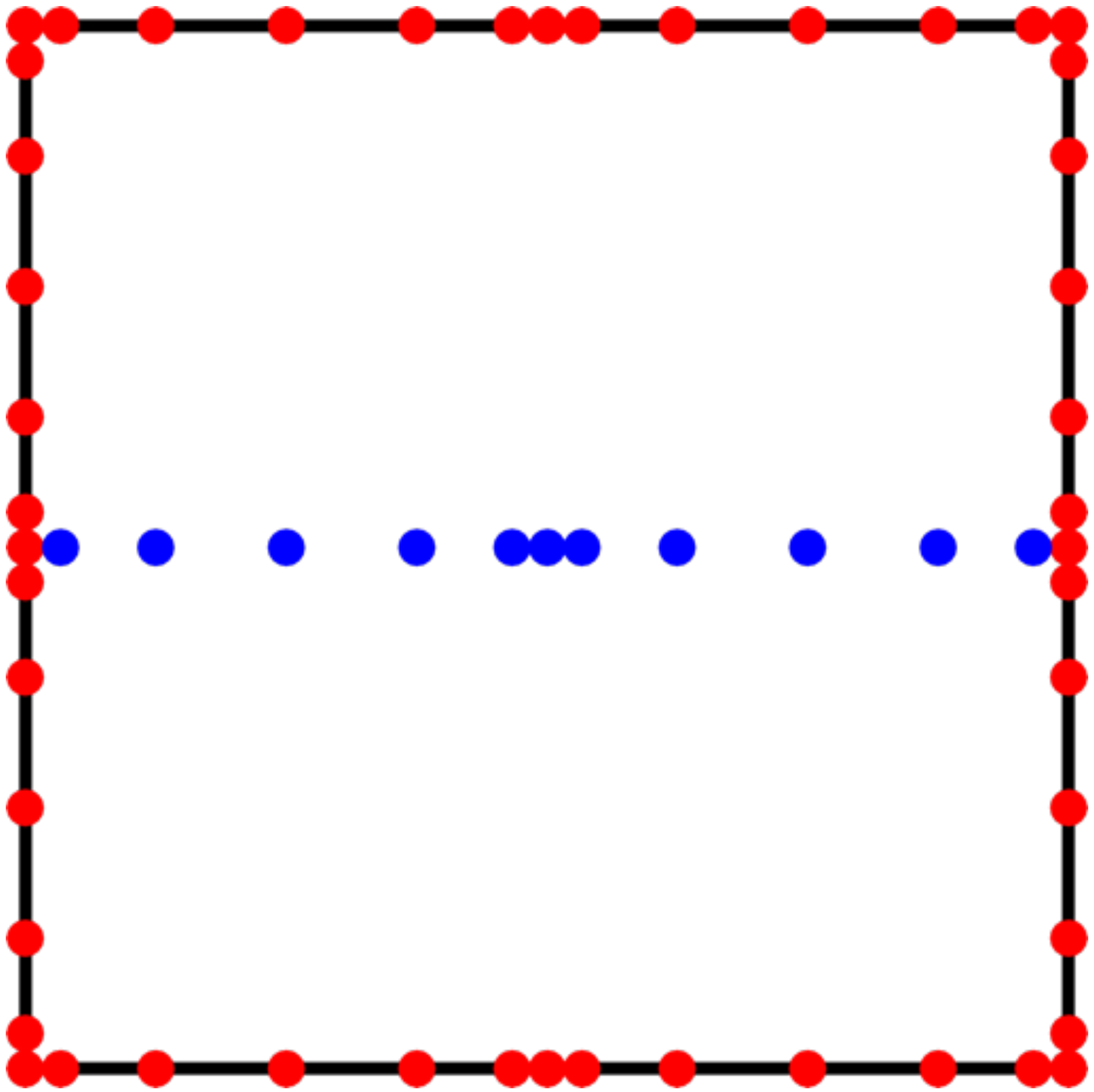

Gunnar Martinsson


Adrianna Gillman

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

<u>Key idea</u>: Recursively glue elements together in a hierarchy.

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

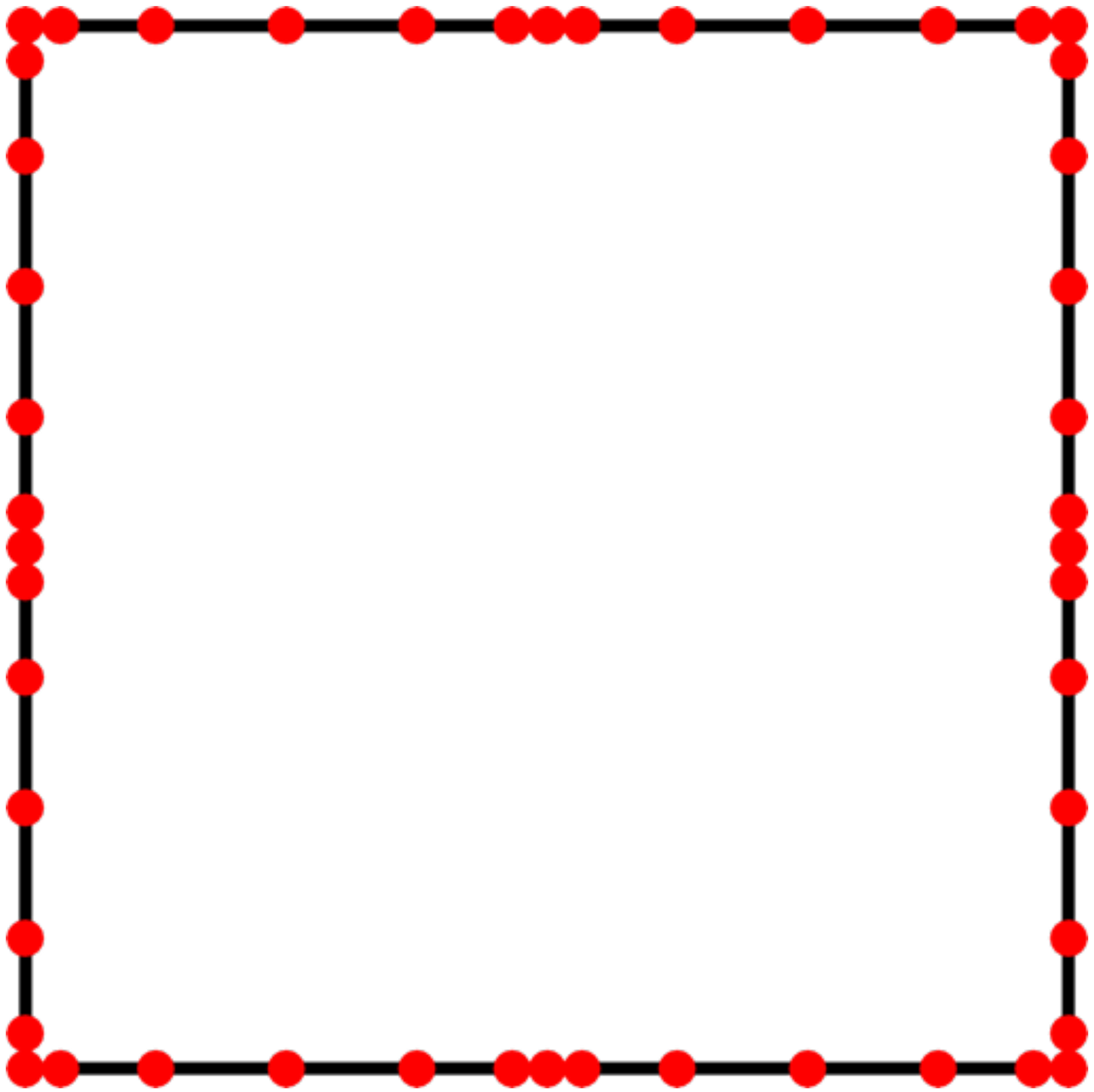


Gunnar Martinsson

Adrianna Gillman

Key idea: Recursively glue elements together in a hierarchy.

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

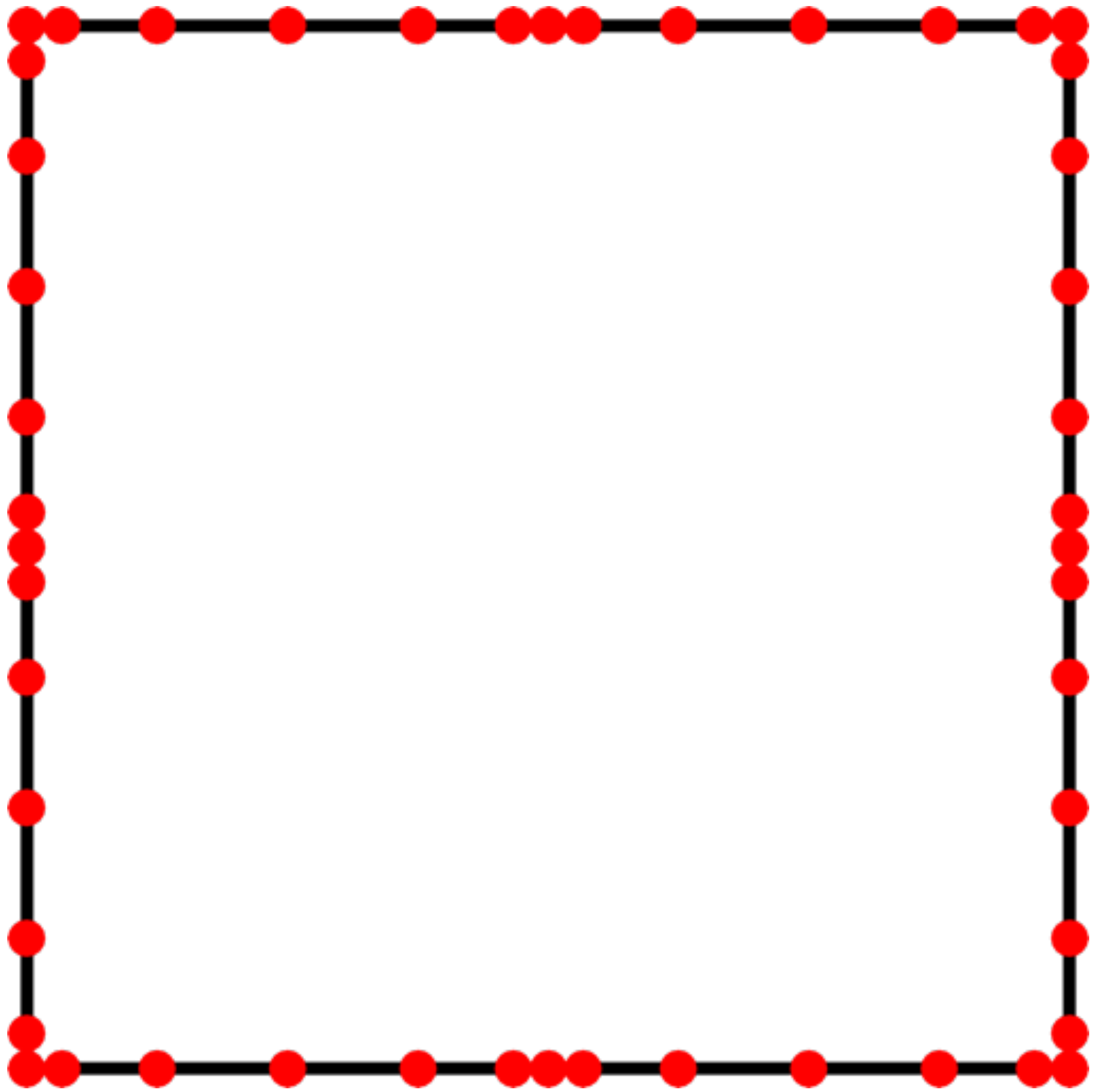


Gunnar Martinsson

Adrianna Gillman

Key idea: Recursively glue elements together in a hierarchy.

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



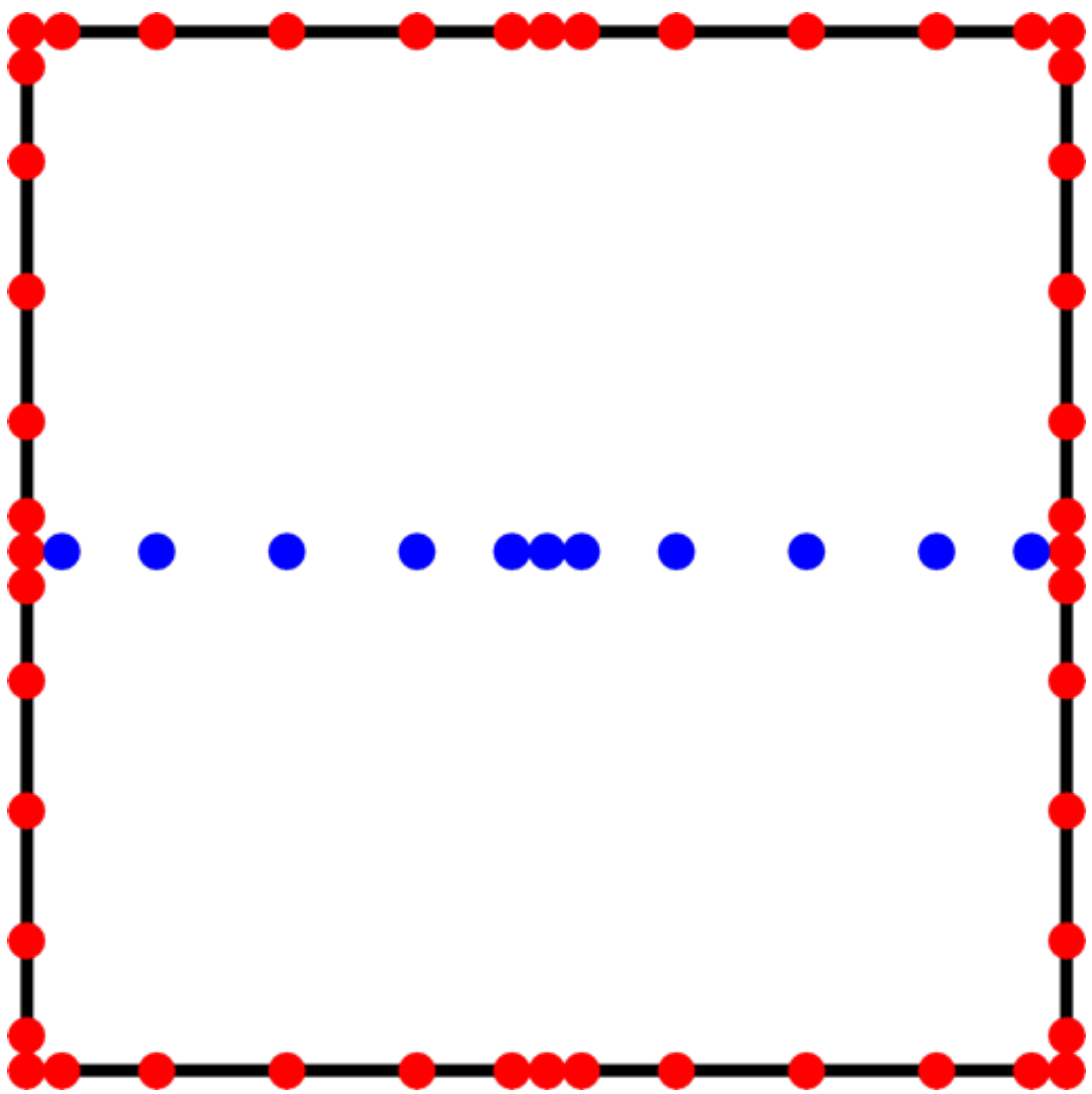

Gunnar Martinsson


Adrianna Gillman

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy.

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



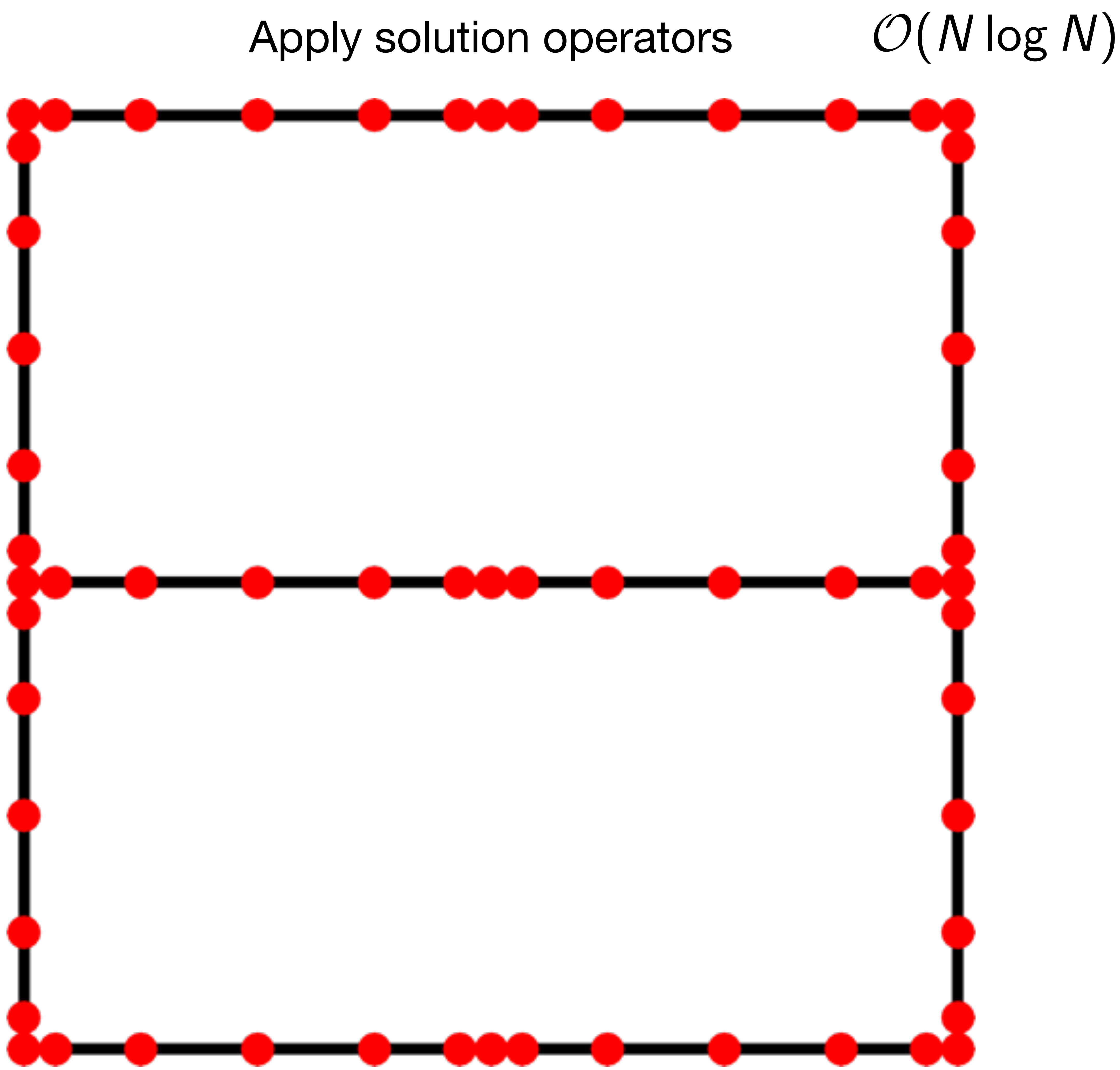

Gunnar Martinsson


Adrianna Gillman

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy.



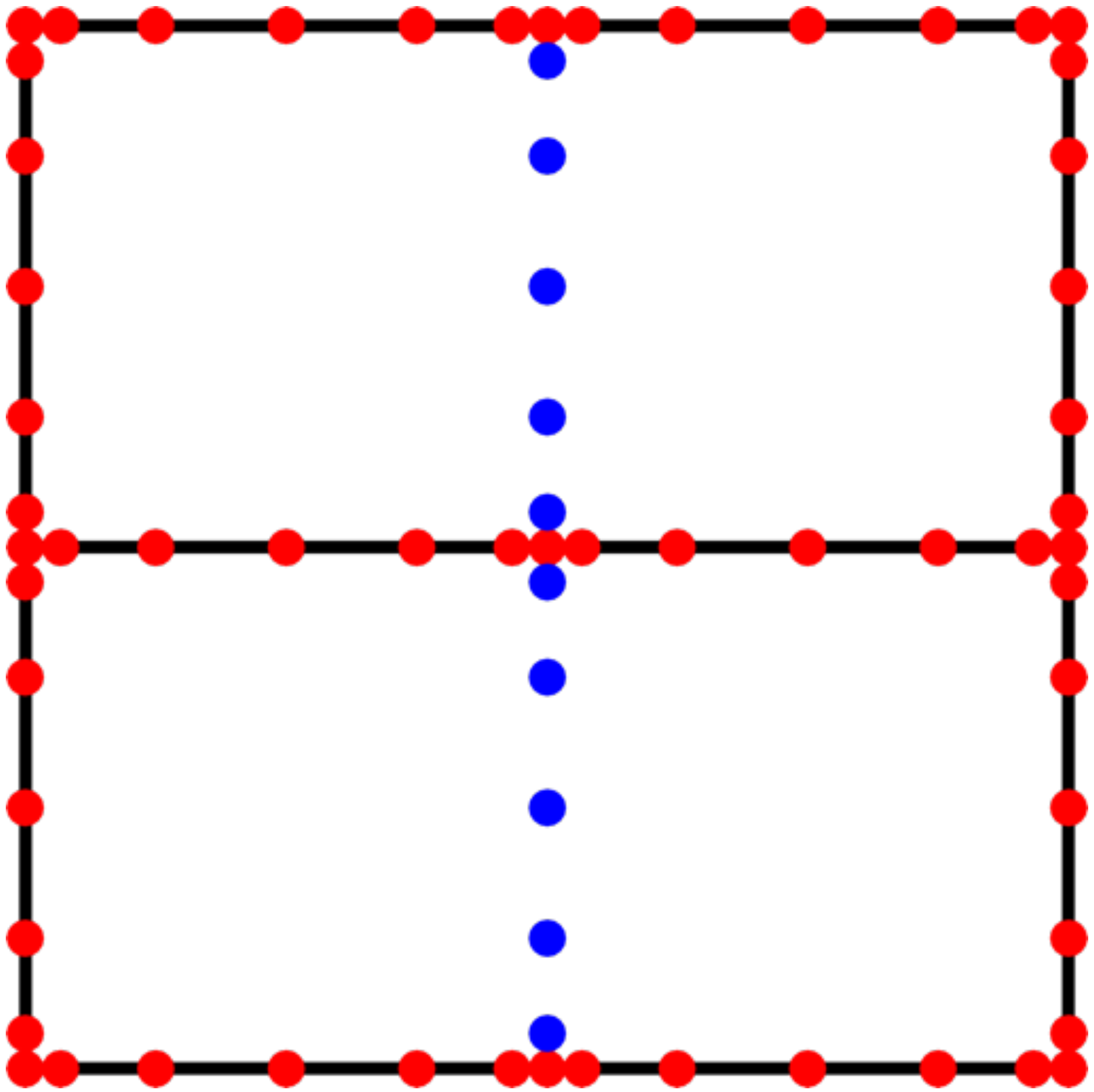
Gunnar Martinsson

Adrianna Gillman

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy.

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



Gunnar Martinsson

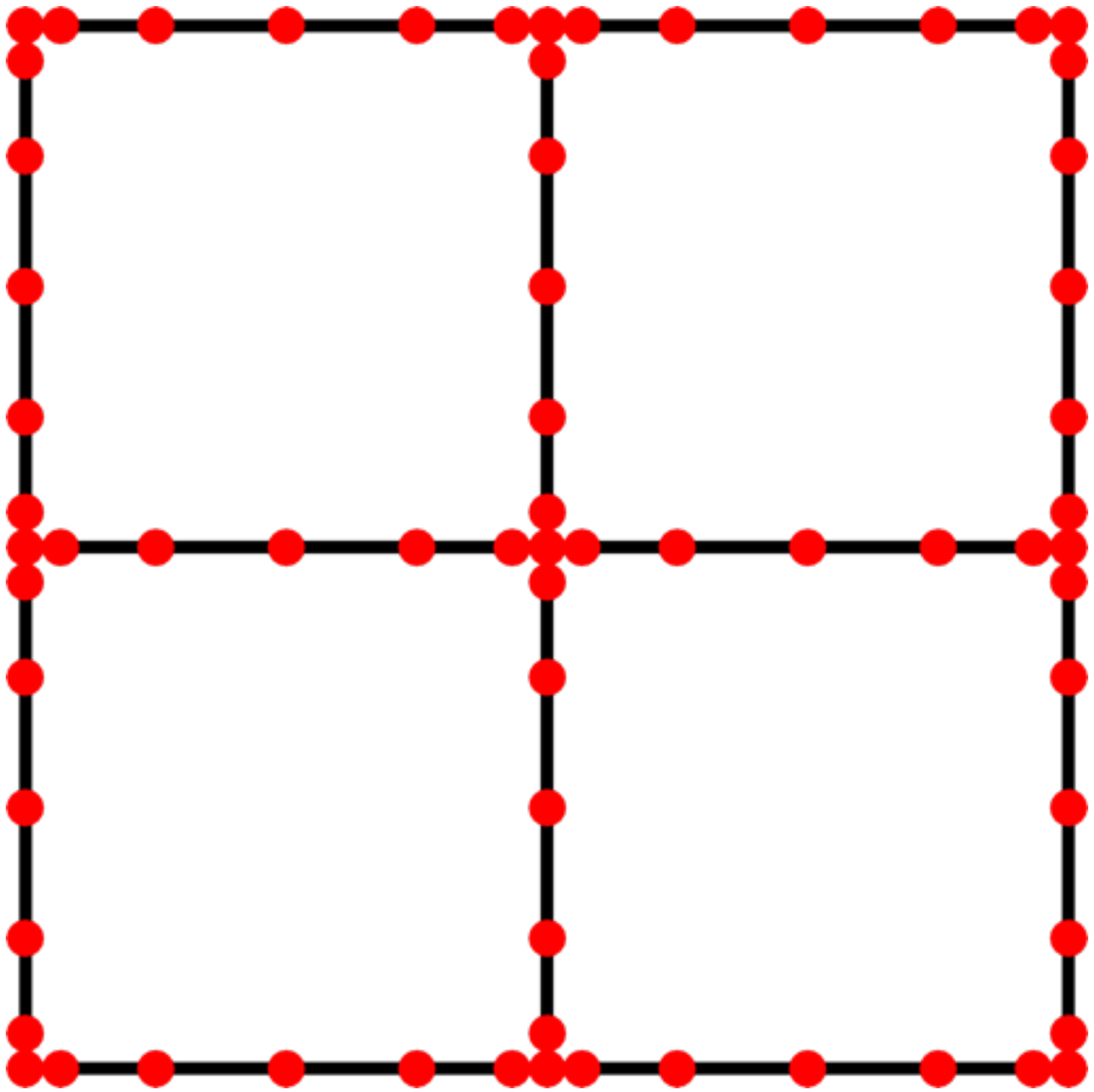
Adrianna Gillman

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

<u>Key idea</u>: Recursively glue elements together in a hierarchy.

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



Gunnar Martinsson

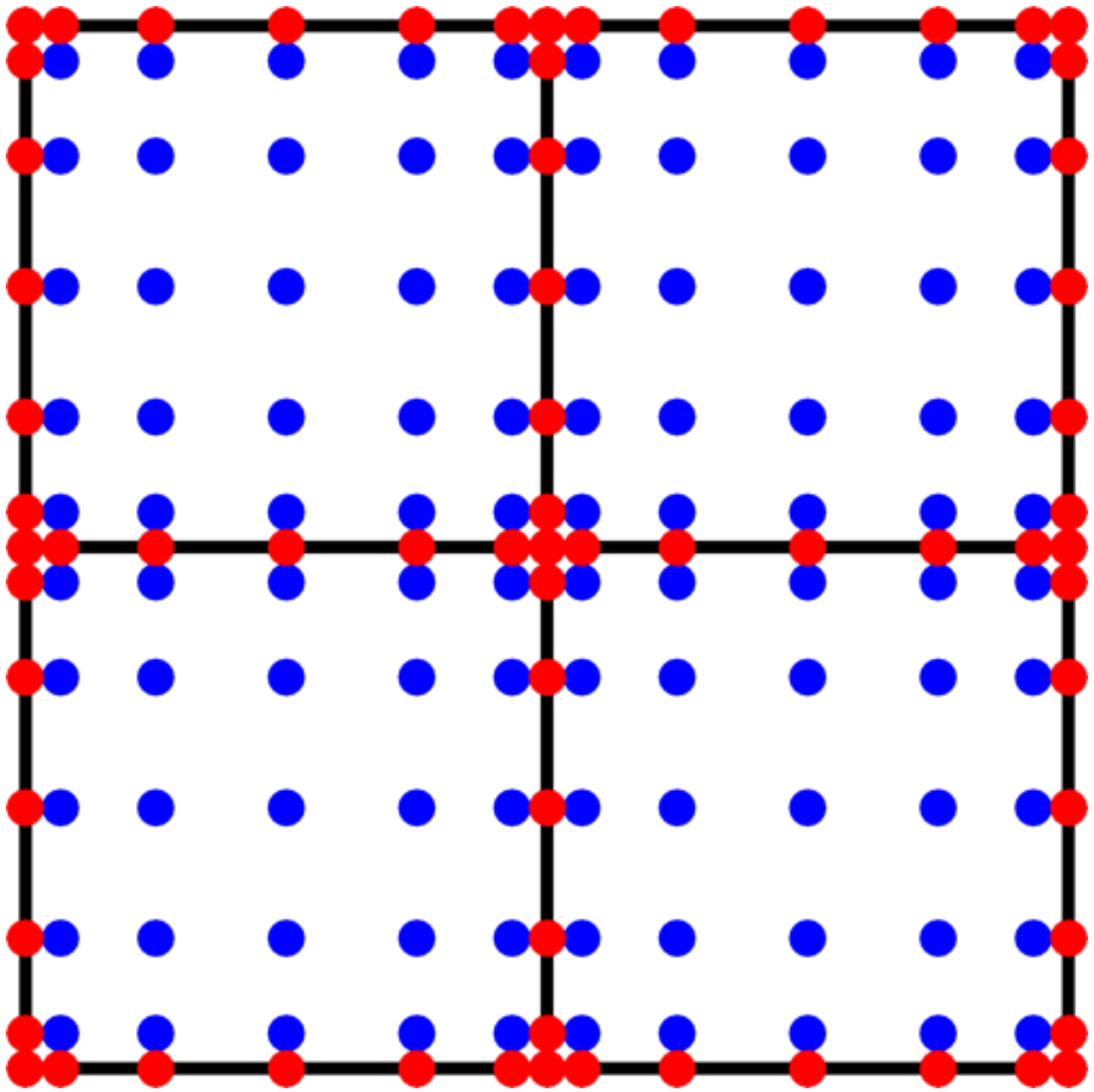
Adrianna Gillman

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

Key idea: Recursively glue elements together in a hierarchy.

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



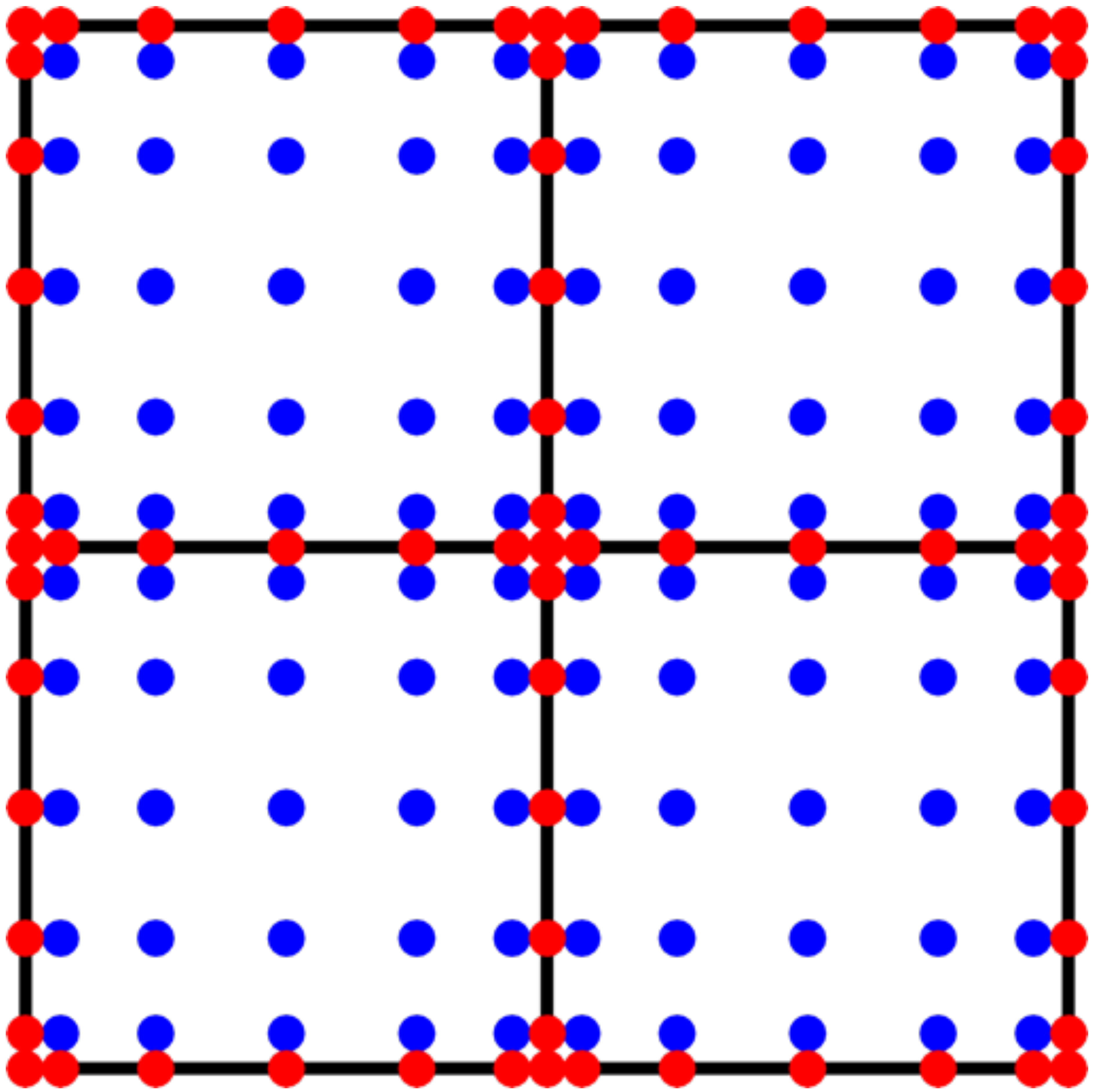
Gunnar Martinsson

Adrianna Gillman

A fast direct solver on surfaces
Hierarchical Poincaré–Steklov method

<u>Key idea</u>: Recursively glue elements together in a hierarchy.

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



Gunnar Martinsson

Adrianna Gillman

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

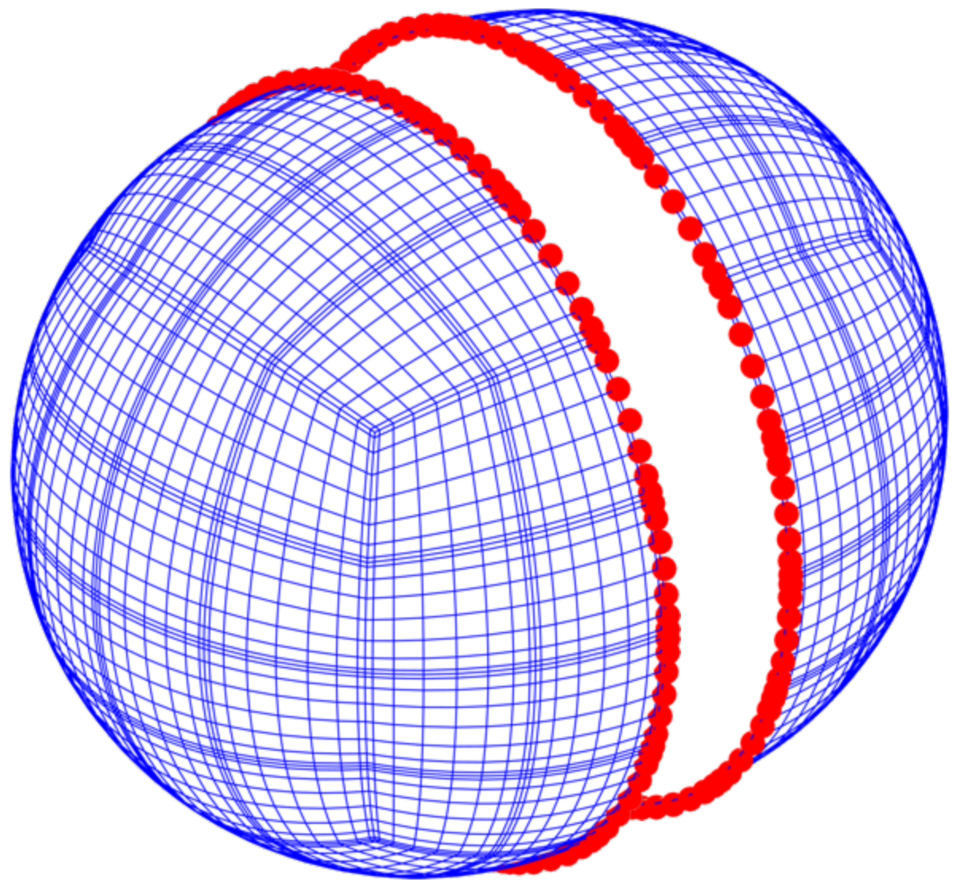
Factorization results in a hierarchy of solution operators stored in memory, so repeated solves are fast.

$$\Delta_\Gamma u = f$$

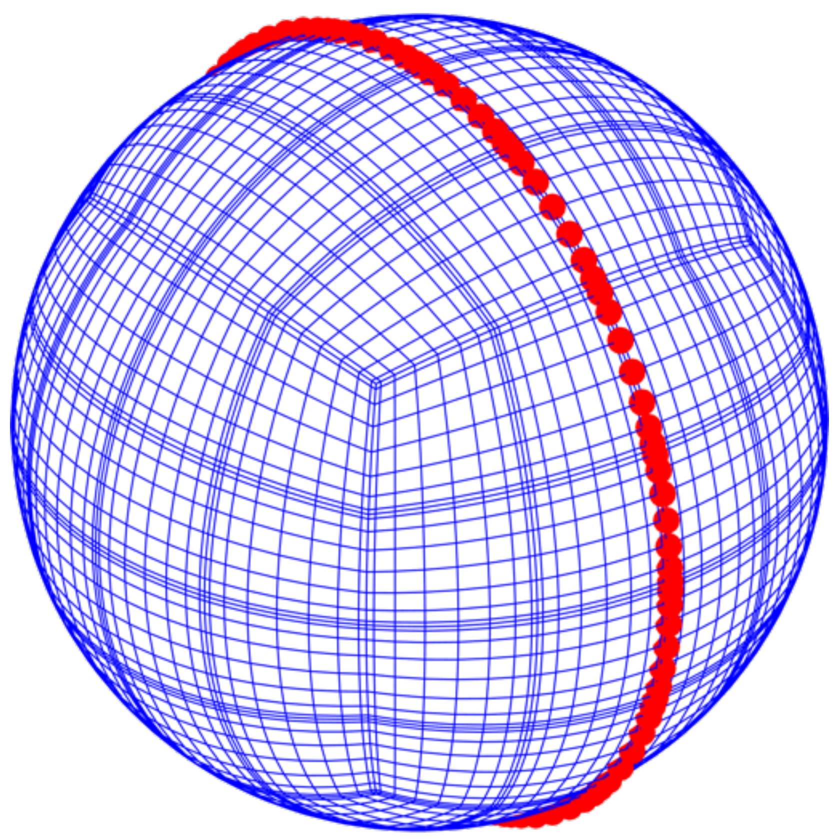
- The Laplace–Beltrami problem on a closed surface is rank-one deficient, but is uniquely solvable under the mean-zero conditions:

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

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



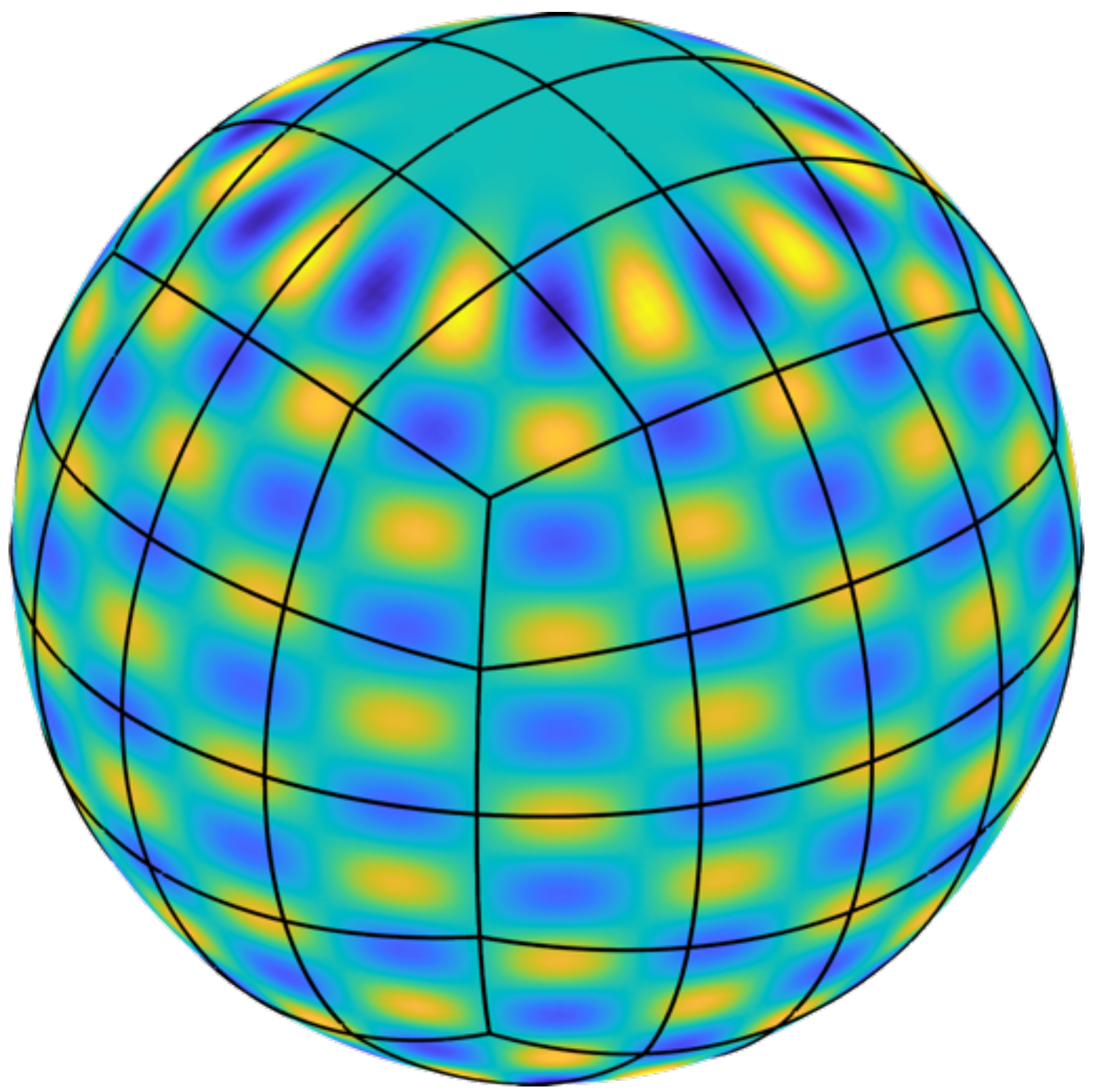
$$\dim\left(\mathrm{null}\left(DtN_1^{\mathsf{glue}} + DtN_2^{\mathsf{glue}}\right)\right) = 1$$



- We use the "ones matrix" to fix the rank deficiency at the top level:

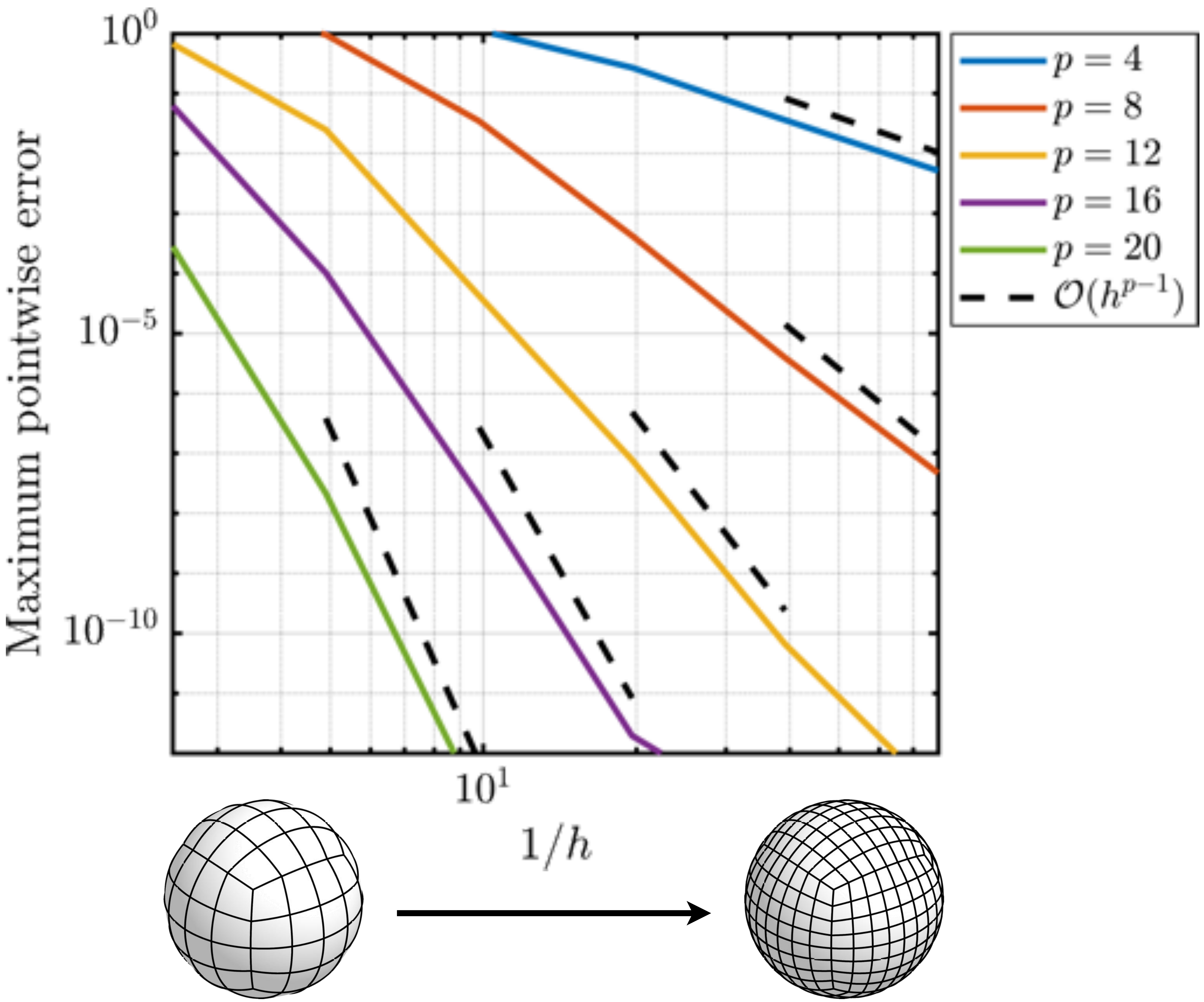
$$\dim\left(\mathrm{null}\left(DtN_1^{\mathsf{glue}} + DtN_2^{\mathsf{glue}} + \mathbf{1}\mathbf{1}^T\right)\right) = 0$$

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



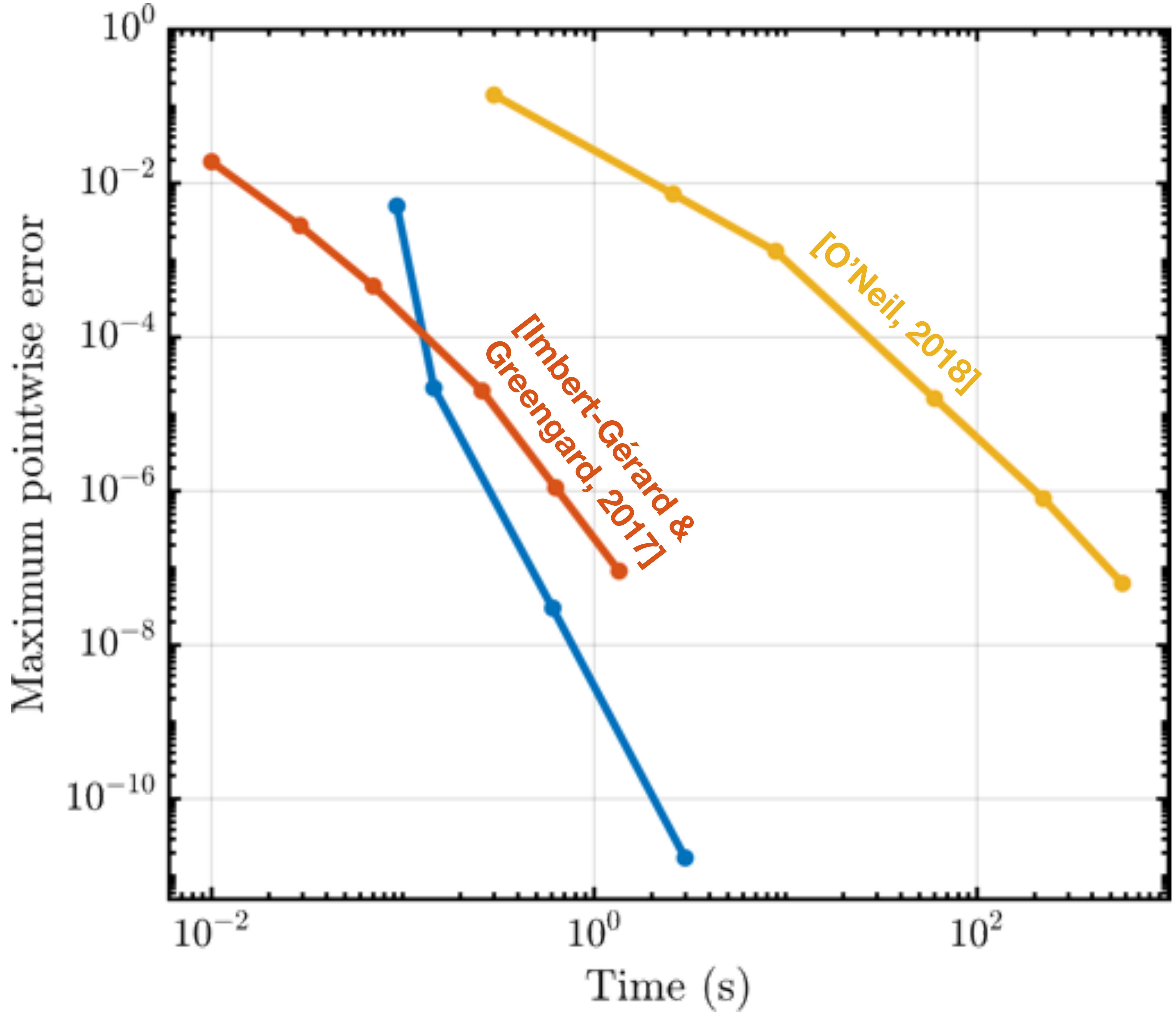
$$u(\boldsymbol{x}) = \text{spherical harmonic, } Y_\ell^m(\boldsymbol{x})$$

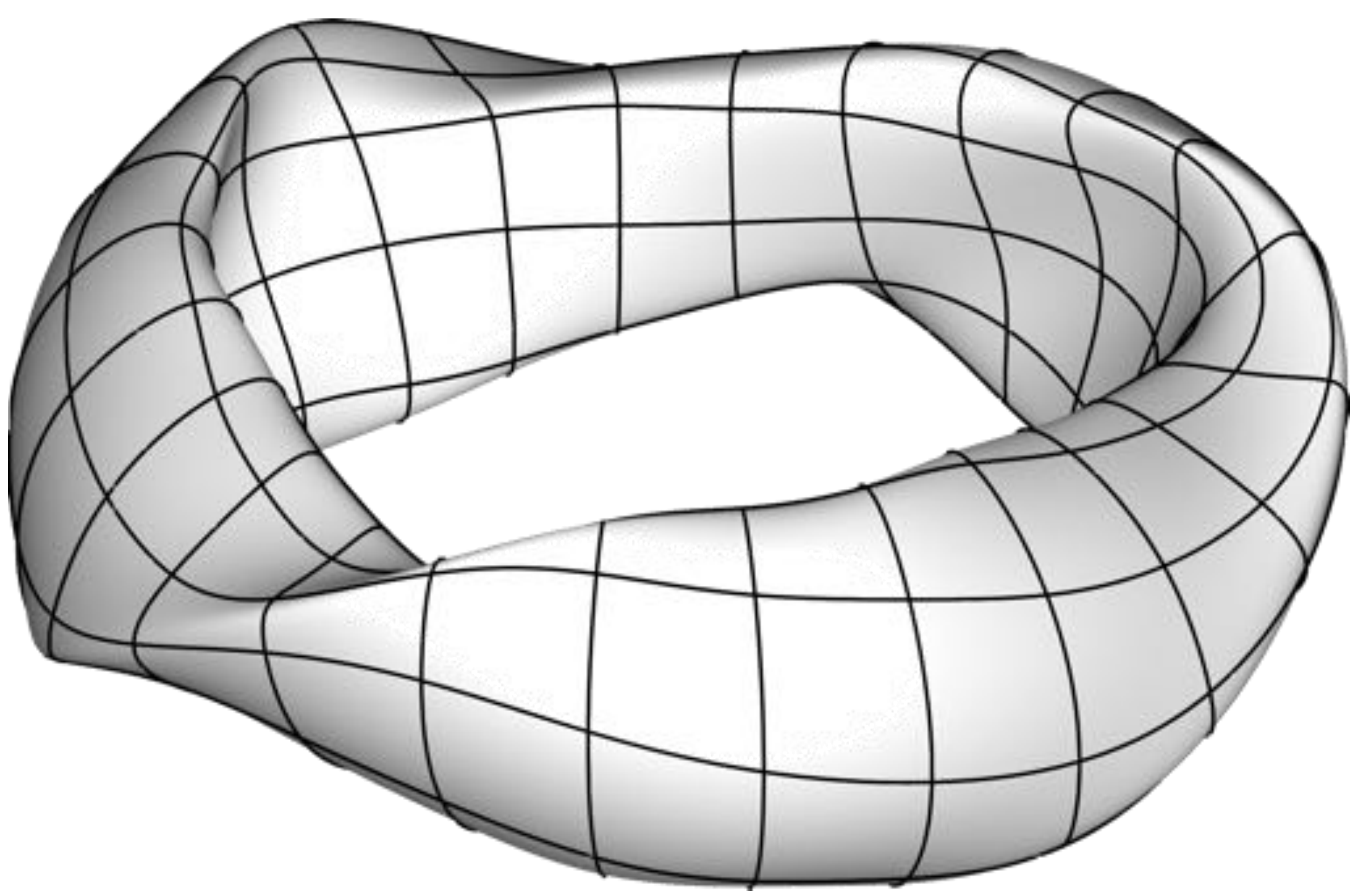
$$f(\boldsymbol{x}) = -\ell(\ell+1)Y_\ell^m(\boldsymbol{x})$$



Examples

Laplace–Beltrami: *"accuracy vs. effort"*



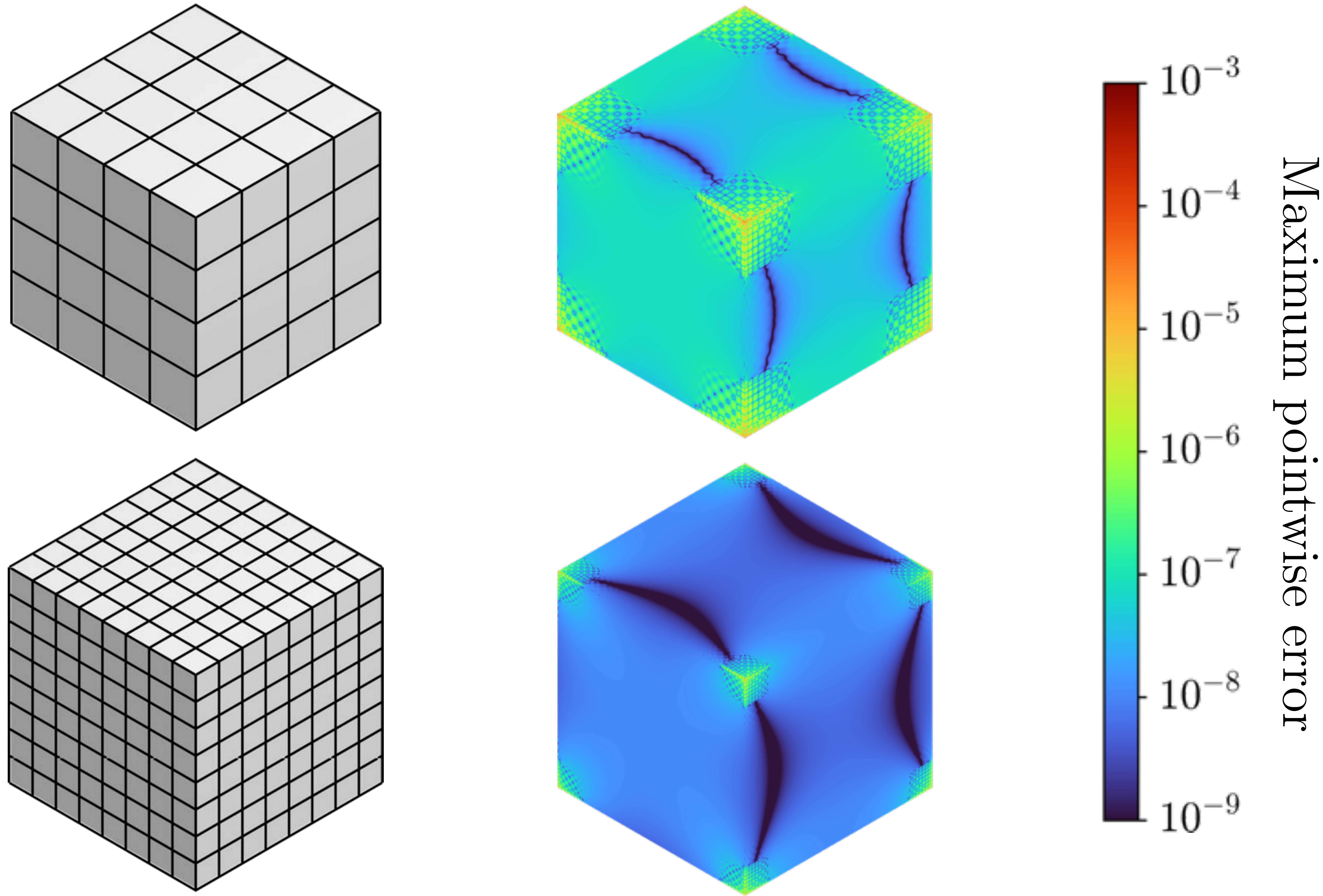


Data from [Malhotra et al., 2019]

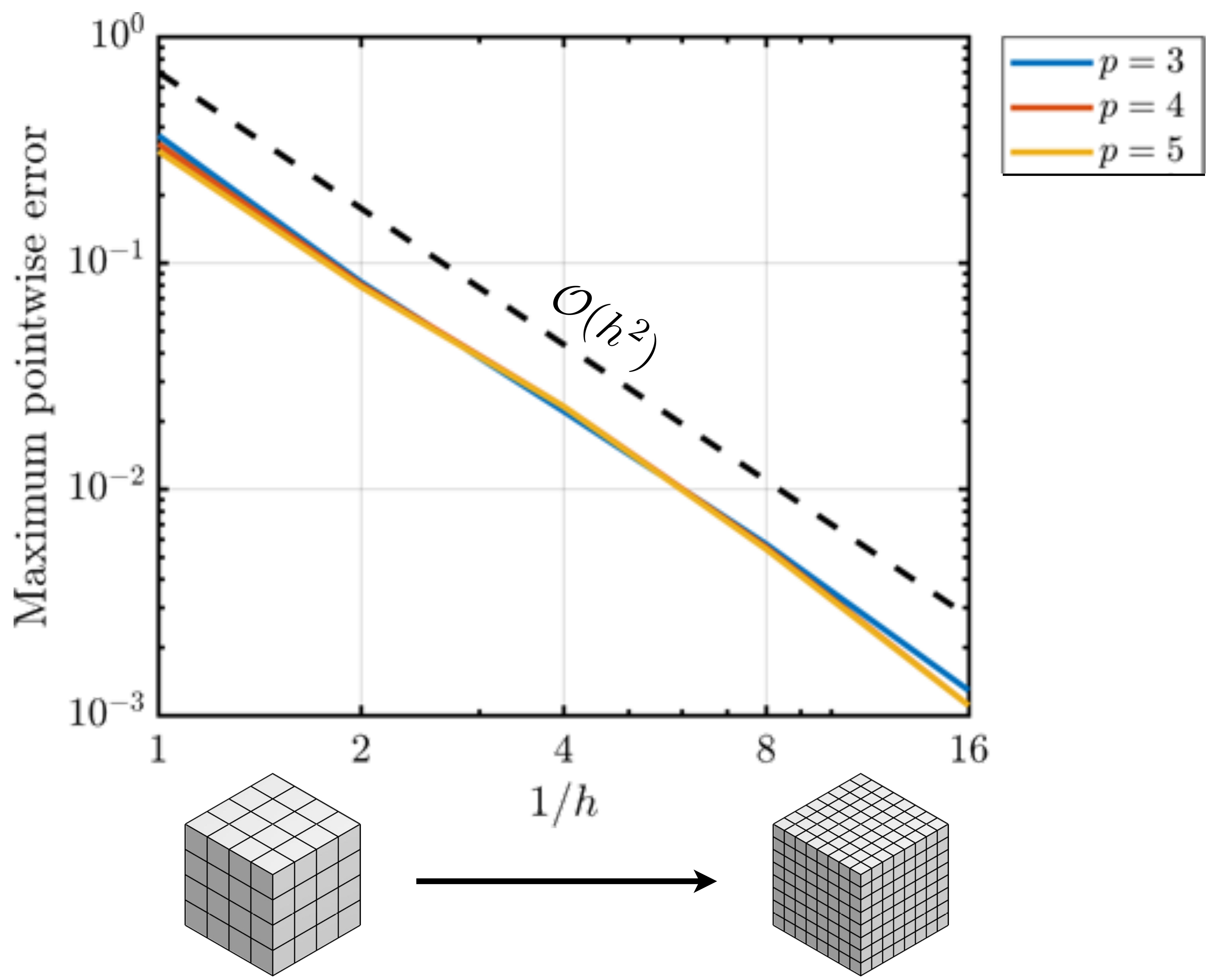
Examples

Laplace–Beltrami with corners

Glue conditions also allow for sharp interfaces and corners.



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



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.

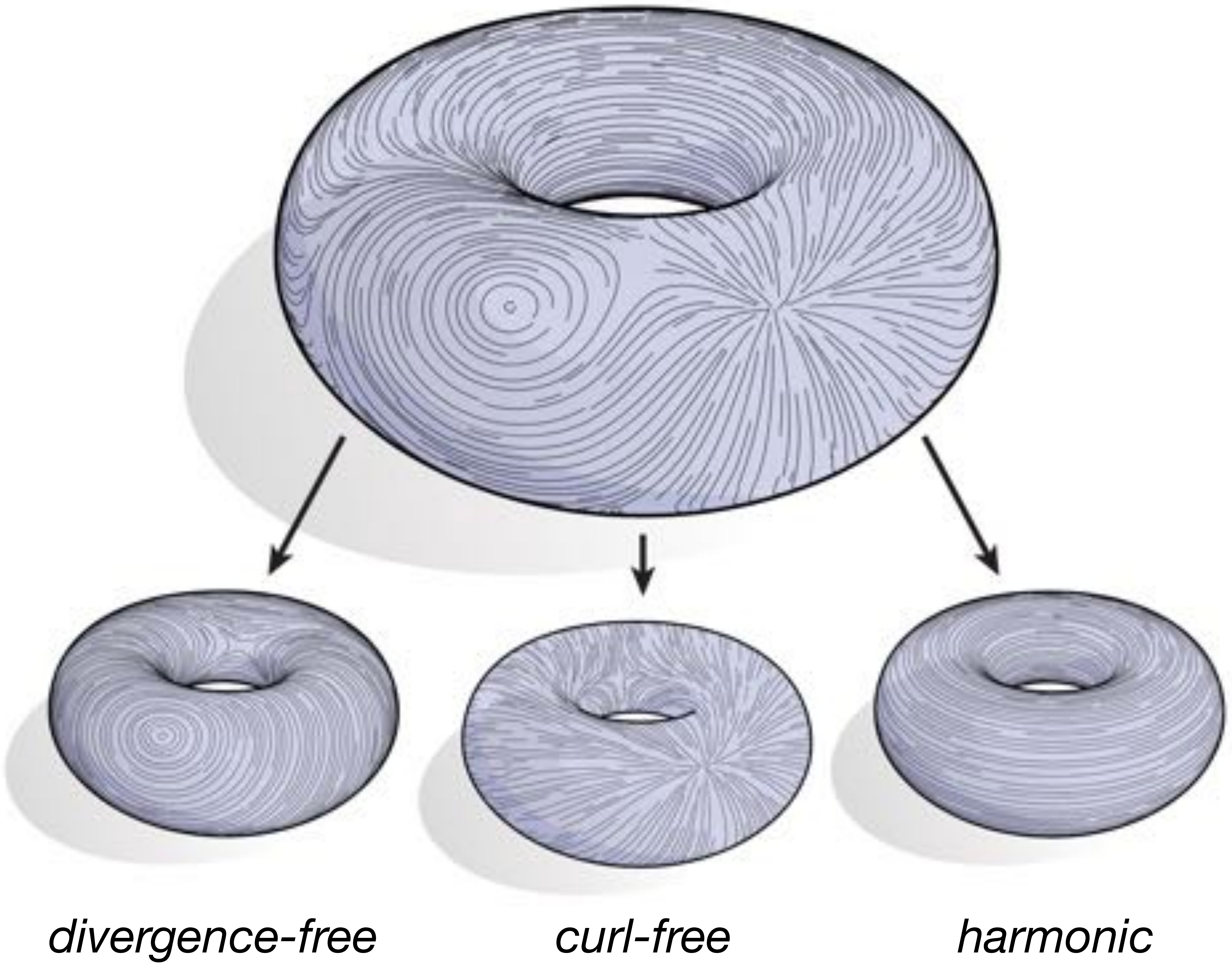


Illustration by Keenan Crane

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

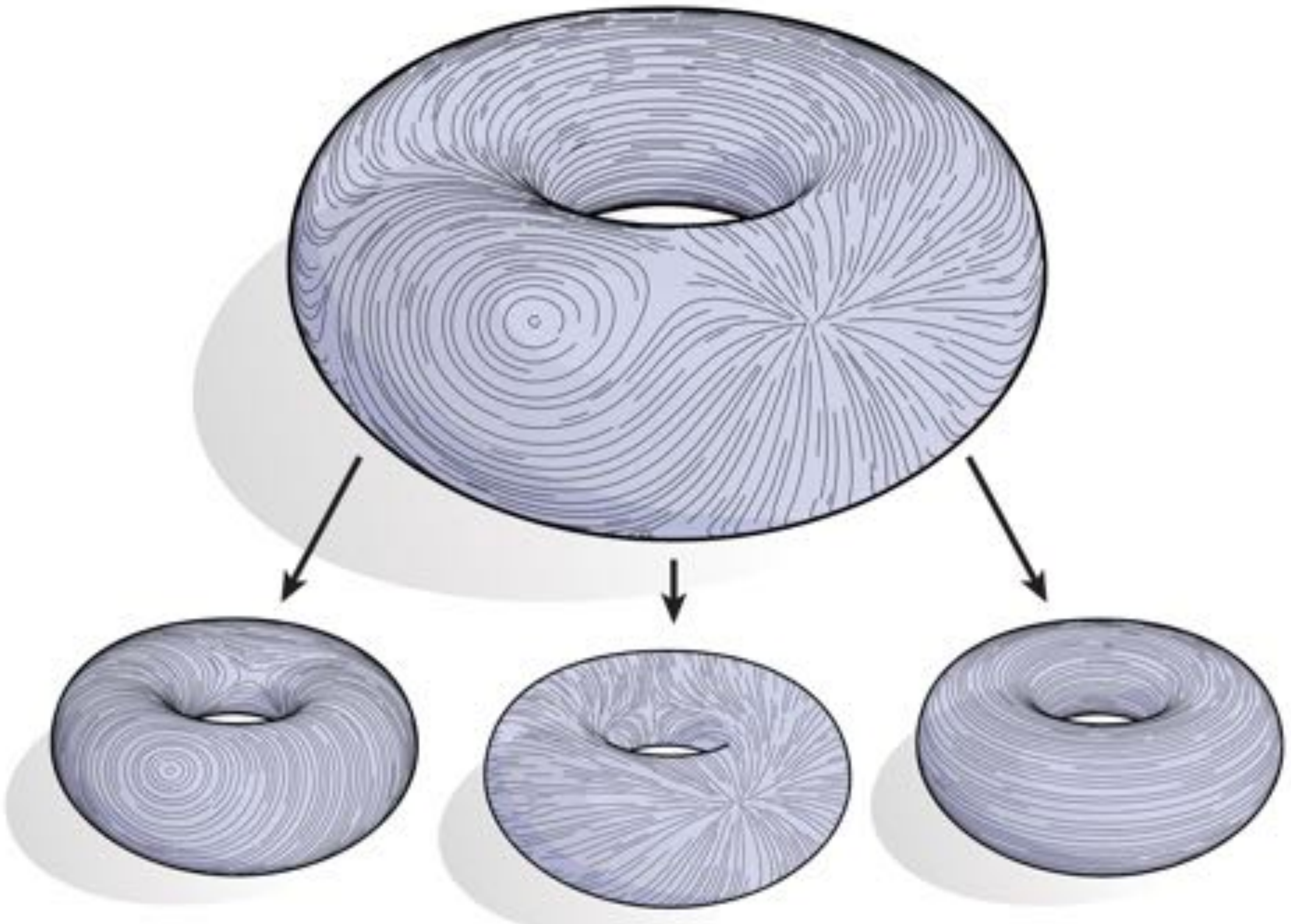
Such decompositions play an important role in integral representations for computational electromagnetics.

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

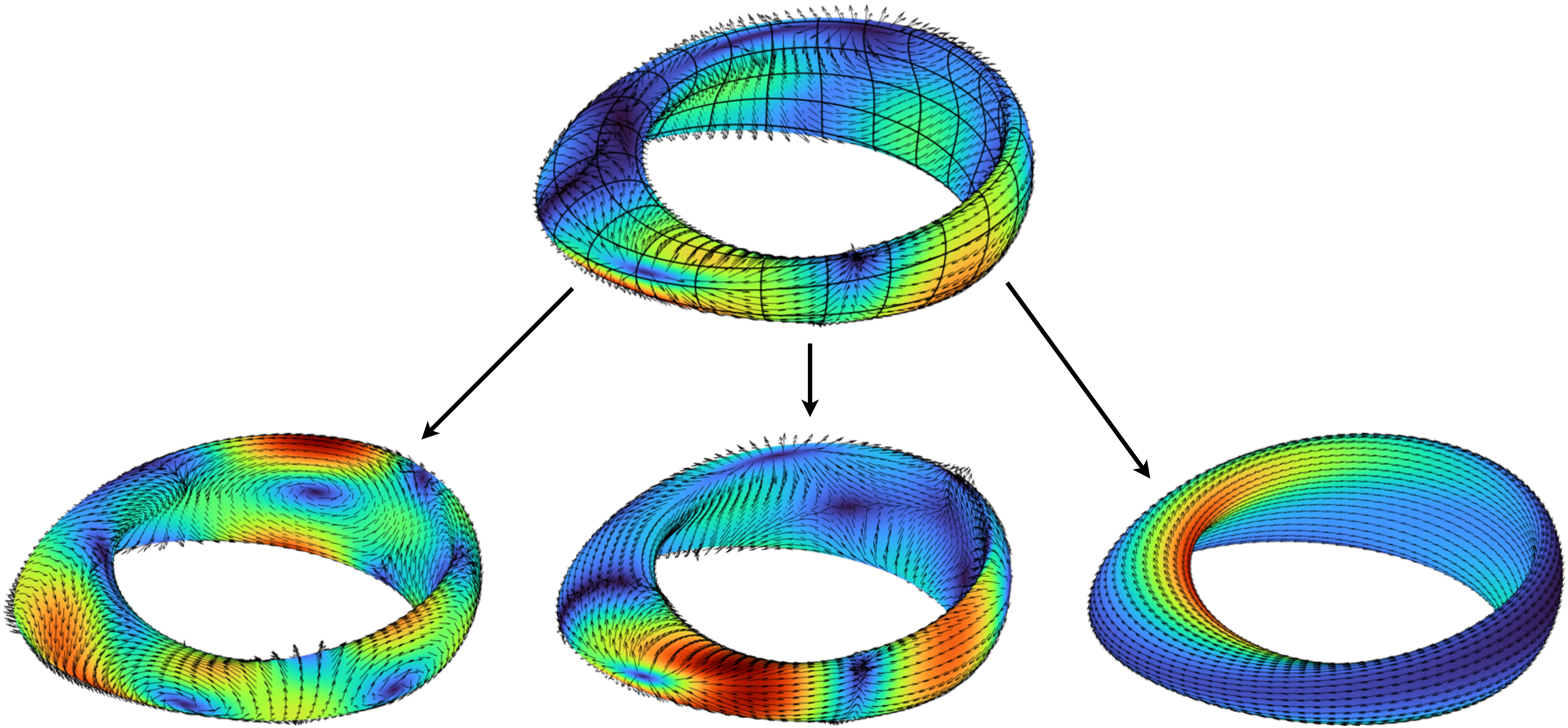


divergence-free *curl-free* *harmonic*

Illustration by Keenan Crane

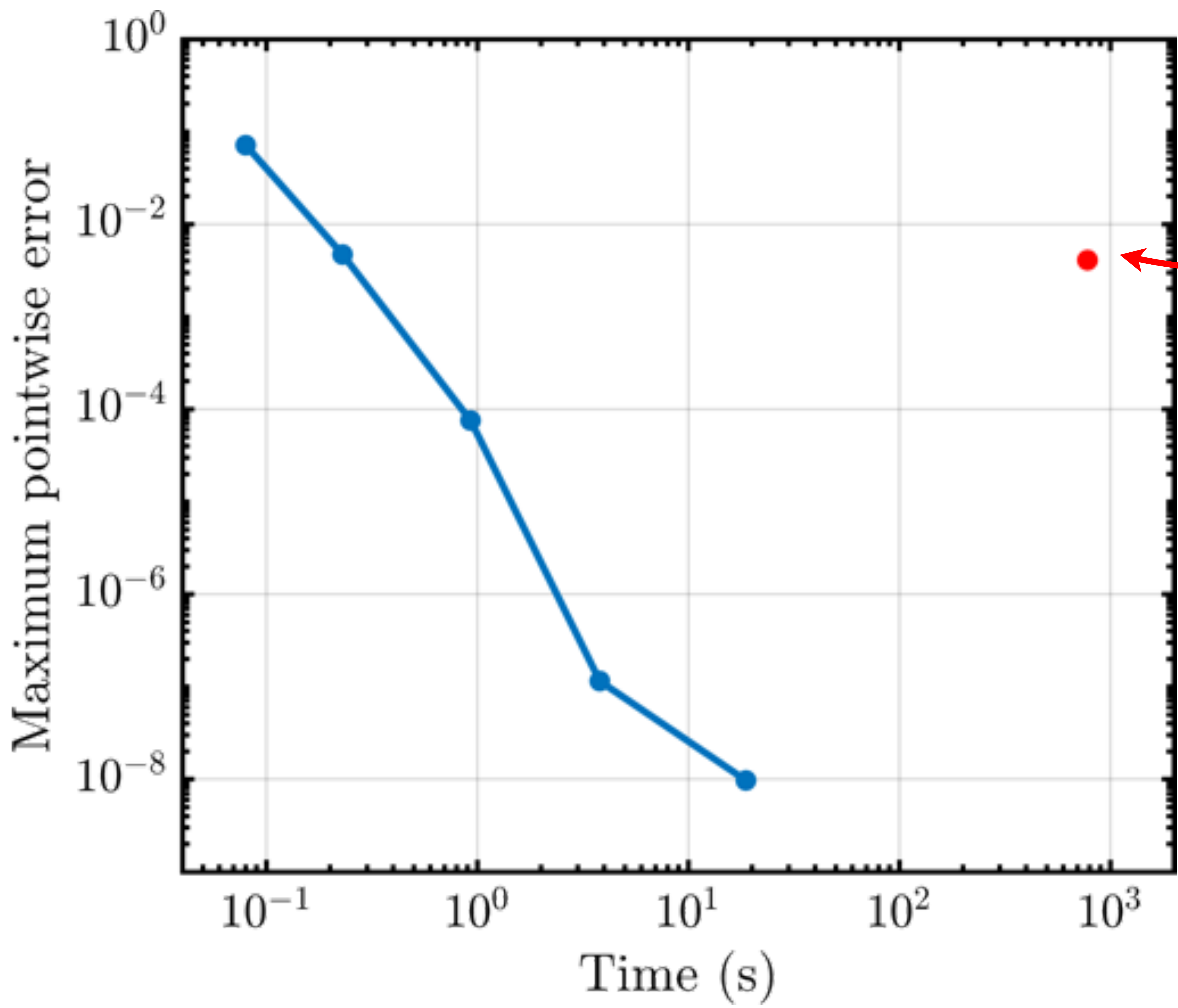
Examples

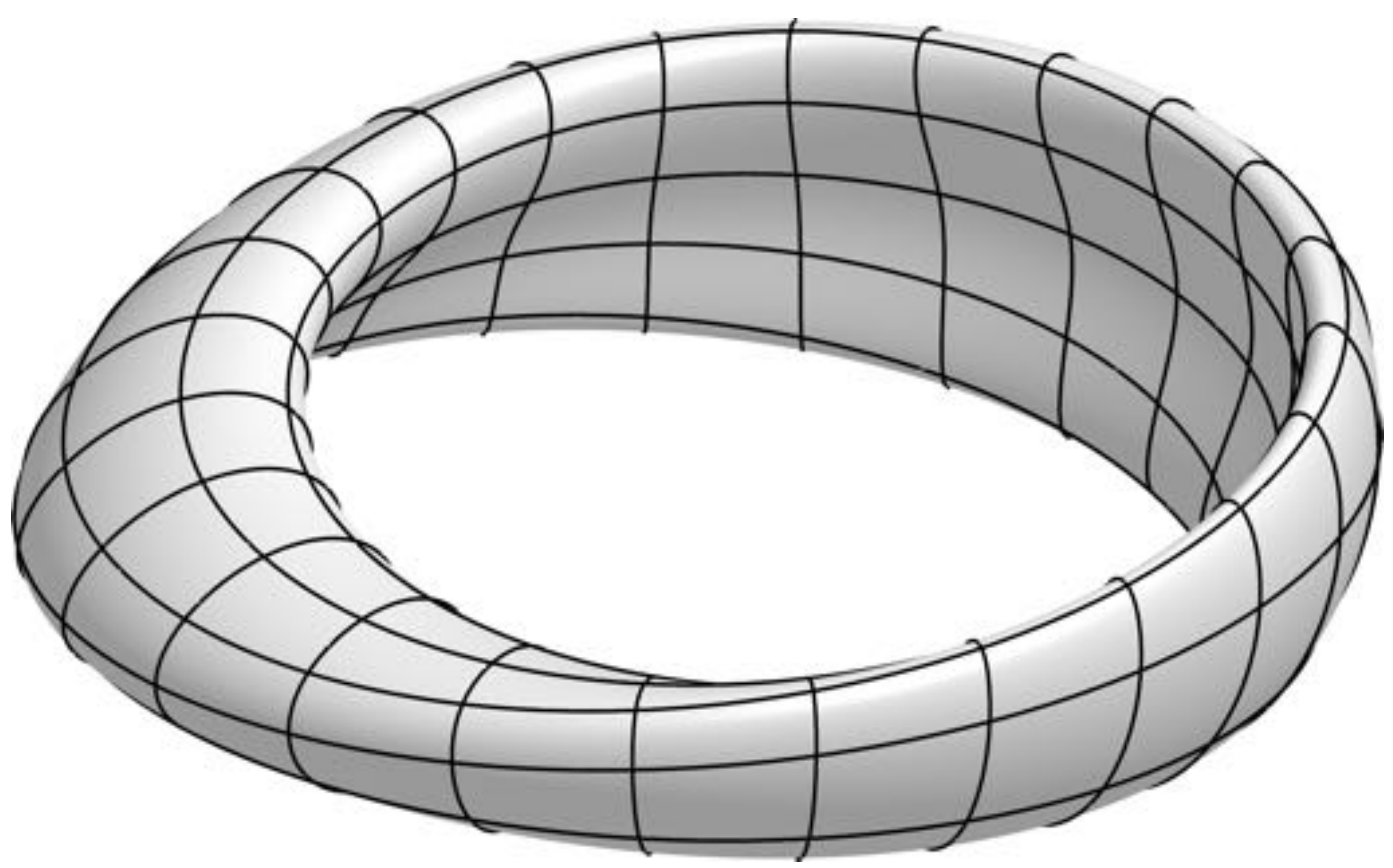
Hodge decomposition



divergence-free *curl-free* *harmonic*

Hodge decomposition: “*accuracy vs. effort*”





Data from Manas Rachh

Examples
Reaction–diffusion systems

- Reaction and diffusion timescales are often orders of magnitude different.

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

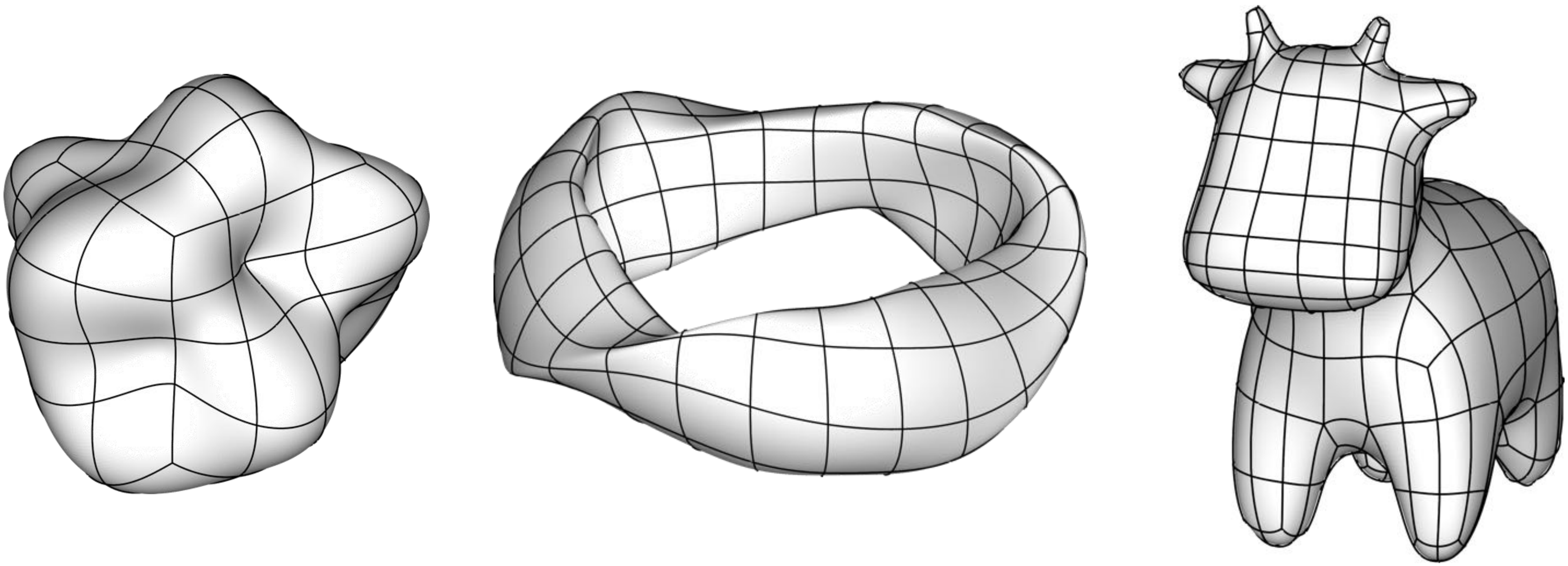
- Implicit time-stepping can alleviate stability issues (e.g., backward Euler or IMEX-BDF4)

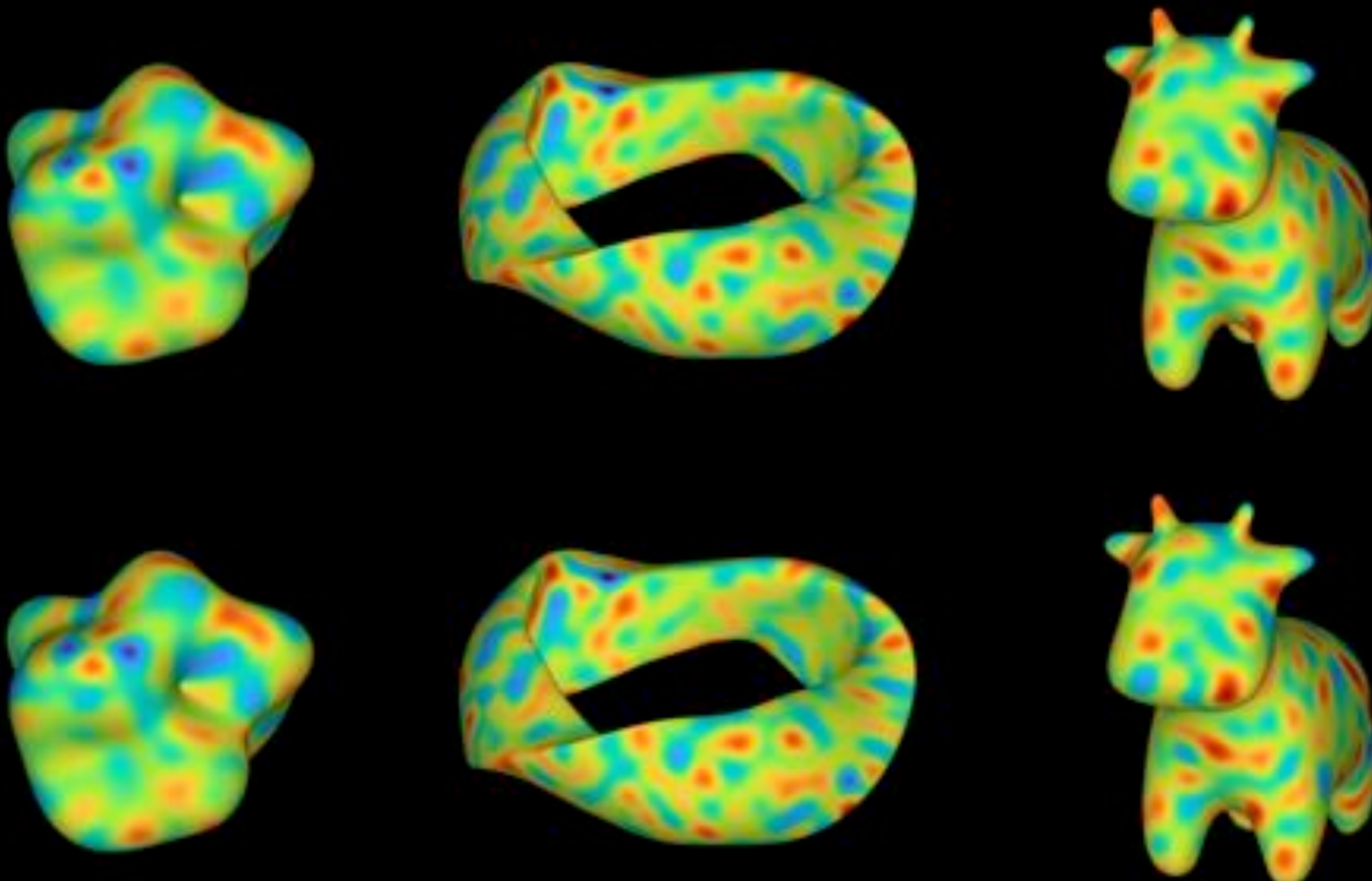
$$\frac{\partial u}{\partial t} = \mathcal{L}_\Gamma u + \mathcal{N}(u) \xrightarrow{\text{(e.g. backward Euler)}} u^{k+1} = \underbrace{(I - \Delta t \mathcal{L}_\Gamma)^{-1}}_{\substack{\text{Stored in RAM,} \\ \text{very fast apply}}} \left(u^k + \Delta t \mathcal{N}(u^k)\right)$$

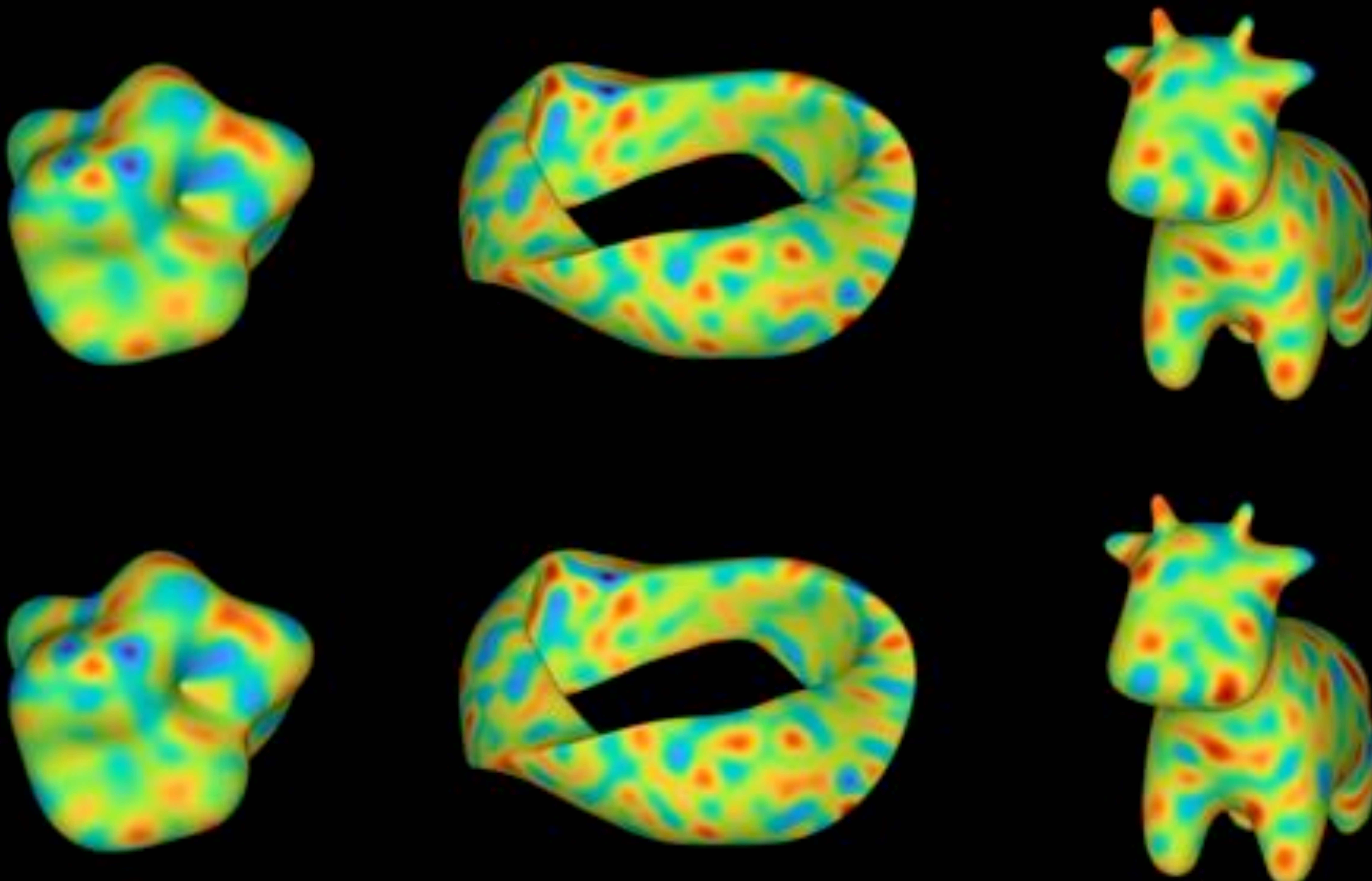
- If geometry, time step, and parameters do not change with time, we can precompute a solver once and reuse it at every step.

Examples

Reaction–diffusion systems







Examples

Eigenvalue problems

with Mengjian Hua (NYU) and Dhairya Malhotra (Flatiron)

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

<u>Simultaneous inverse iteration:</u>

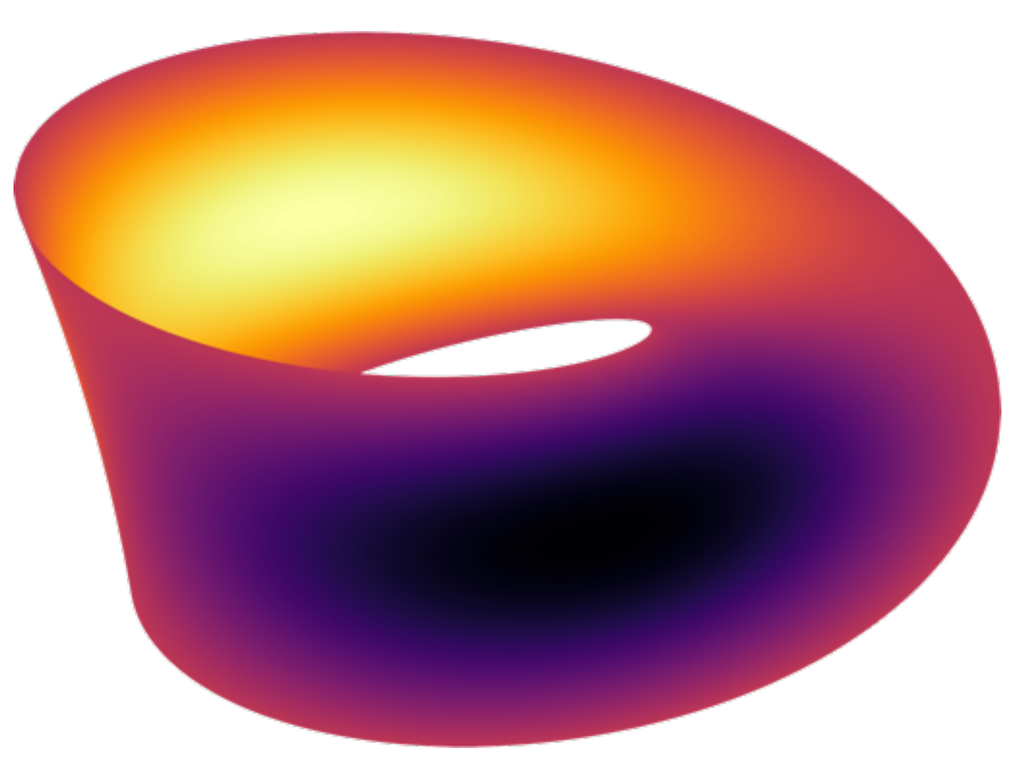
$$Q^{(0)} = \texttt{rand}(N, m)$$

for $k = 1, 2, \ldots$

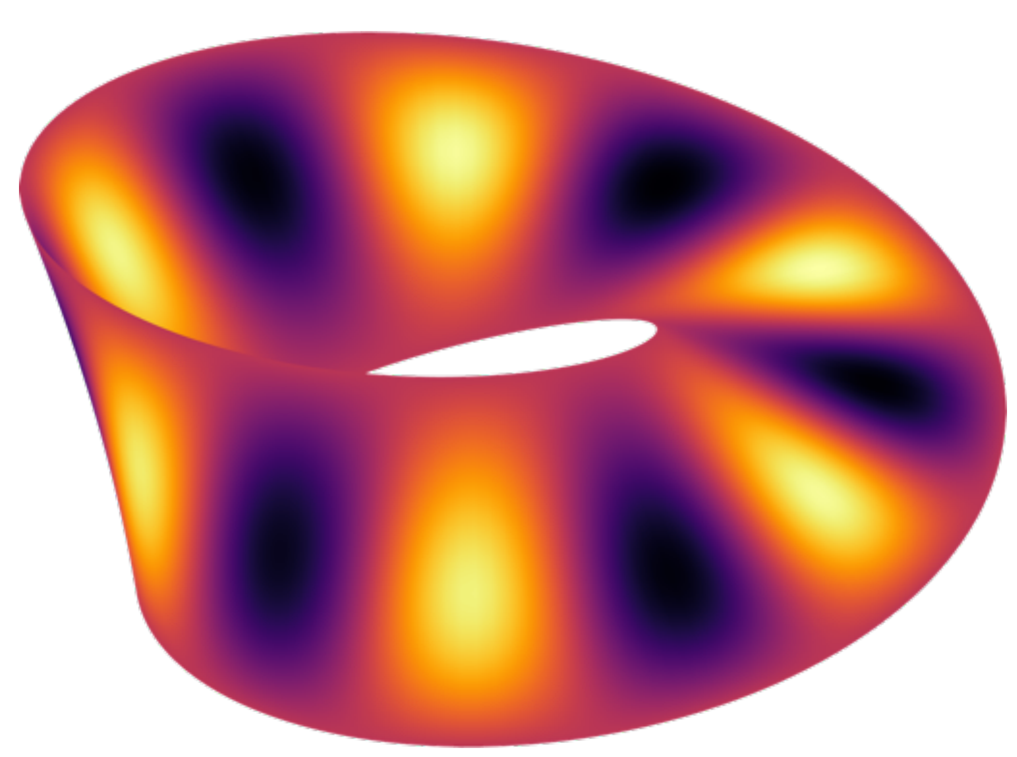
$$Z^{(k-1)} = \Delta_\Gamma^{-1} Q^{(k-1)}$$

Stored in RAM, very fast apply

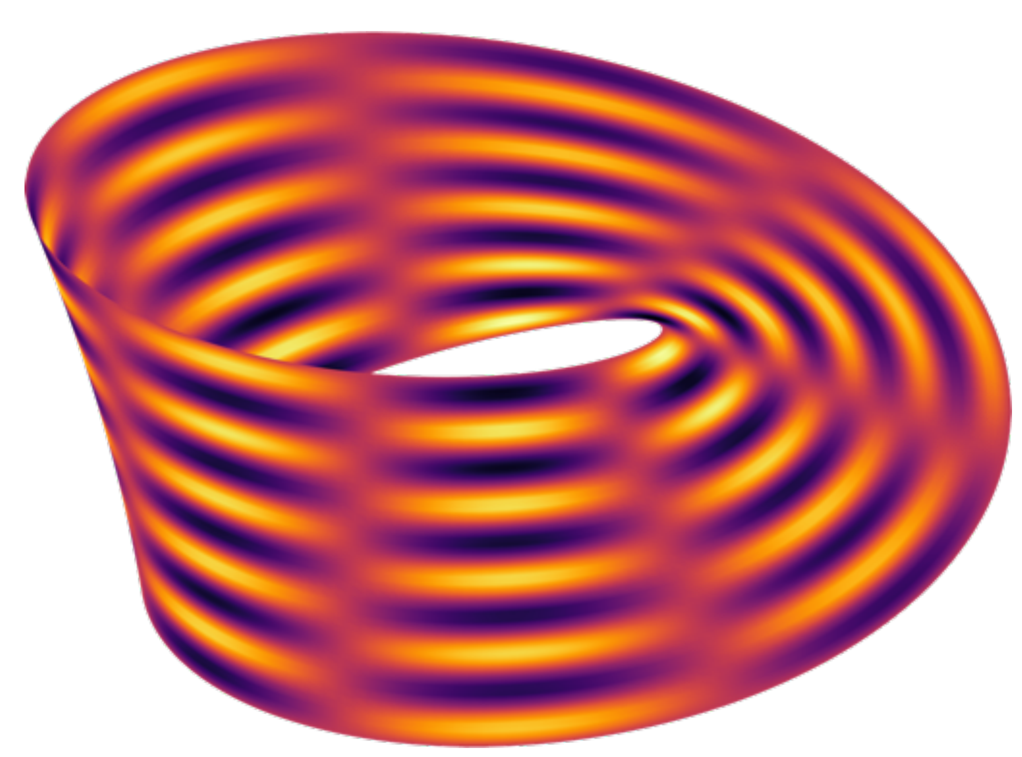
$$Q^{(k)} R^{(k)} = Z^{(k-1)}$$



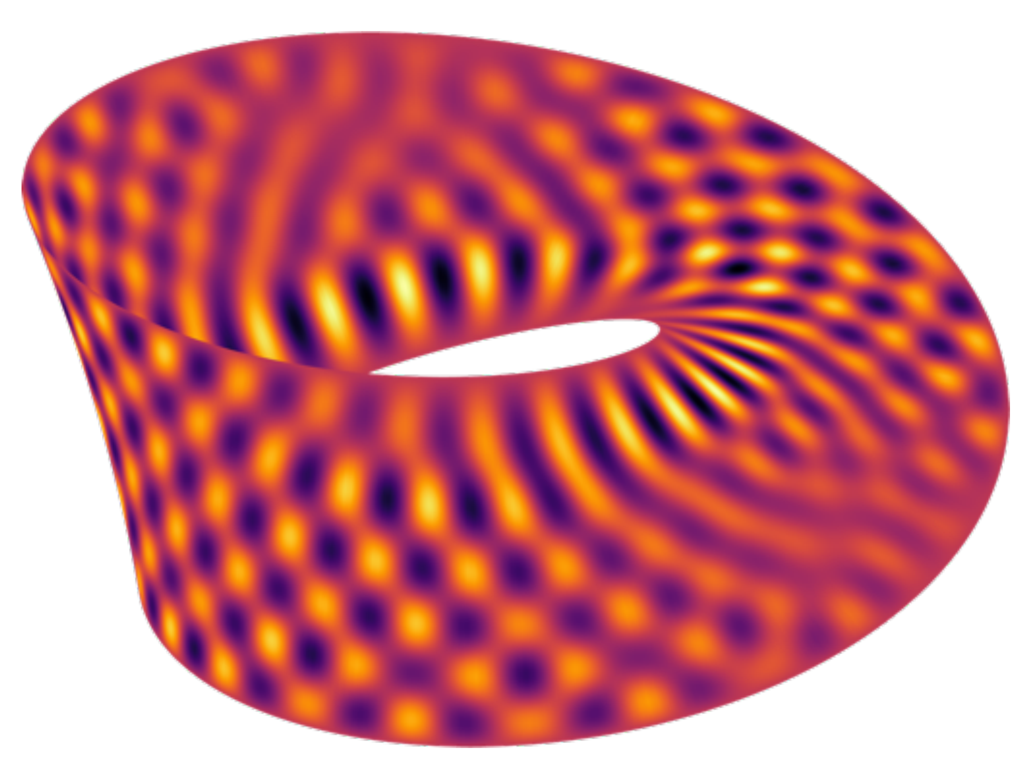
$\lambda = -10.93...$



$\lambda = -44.66...$



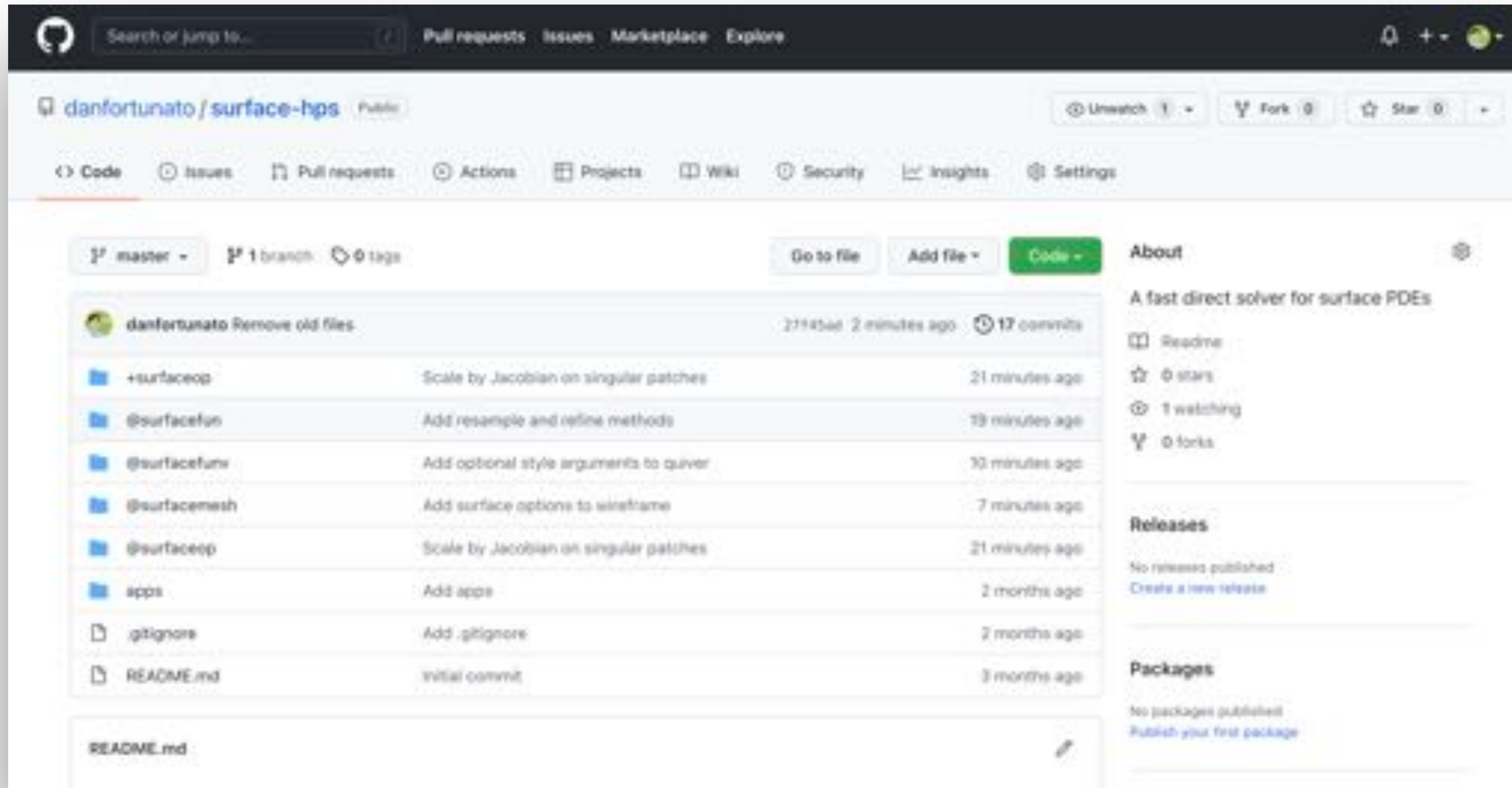
$\lambda = -1000.56...$



$\lambda = -998.11...$

Software
`github.com/danfortunato/surface-hps`

- Provides abstractions for computing with functions on surfaces in MATLAB.



Thank you