

An introduction to Surfacefun

Dan Fortunato
Flatiron Institute

<https://surfacefun.readthedocs.io/>

Introduction

Today's plan

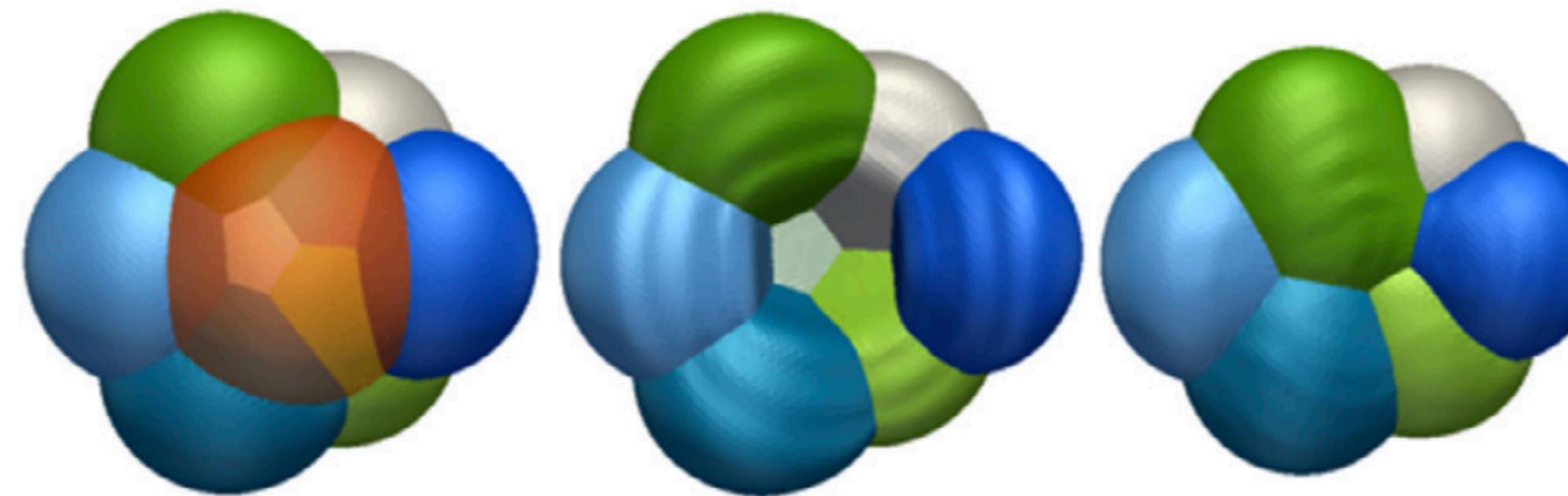
- Mathematical background
- The algorithms behind Surfacefun
- Software tutorial and live demos

Introduction

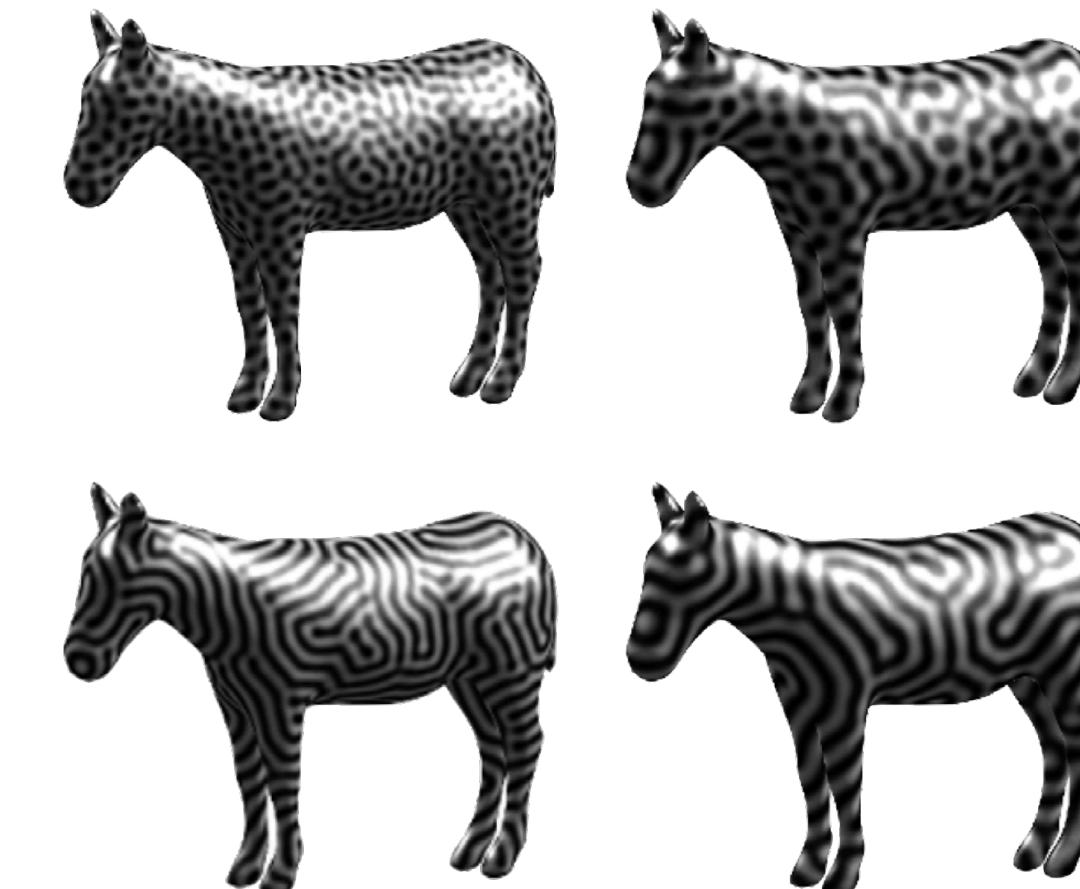
Why solve Surface PDEs?

Surface-bound phenomena arise in many applications.

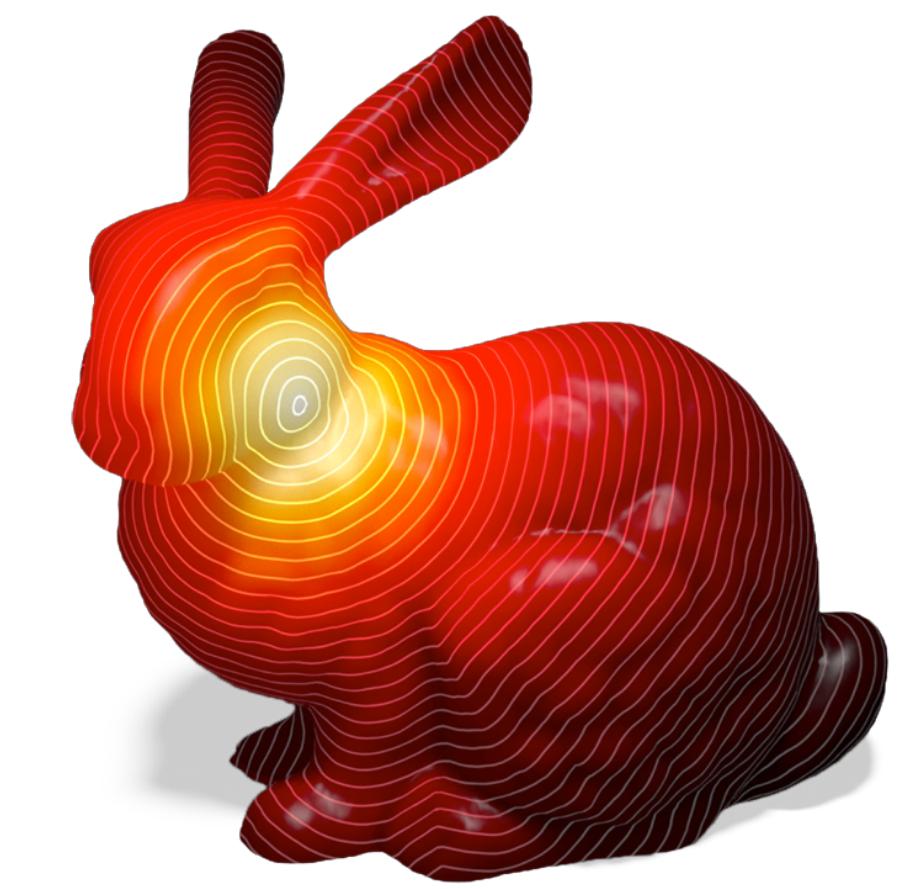
Thin-film hydrodynamics [Saye, 2016]



Pattern formation [Jeong, 2017]



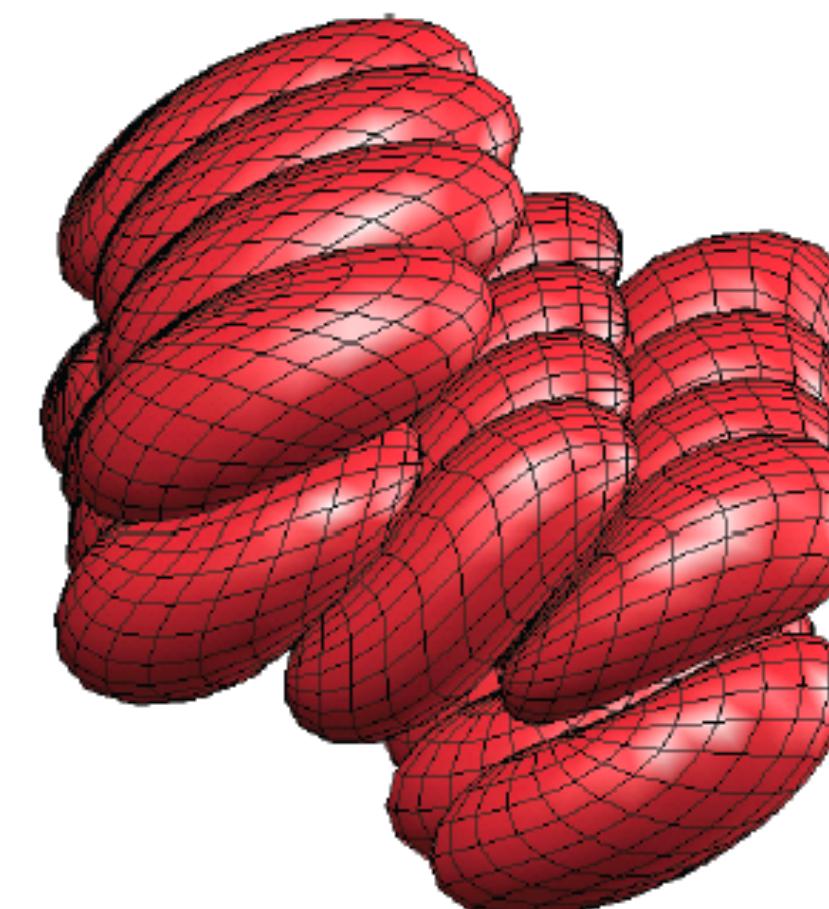
Geodesic distance [Crane et al., 2017]



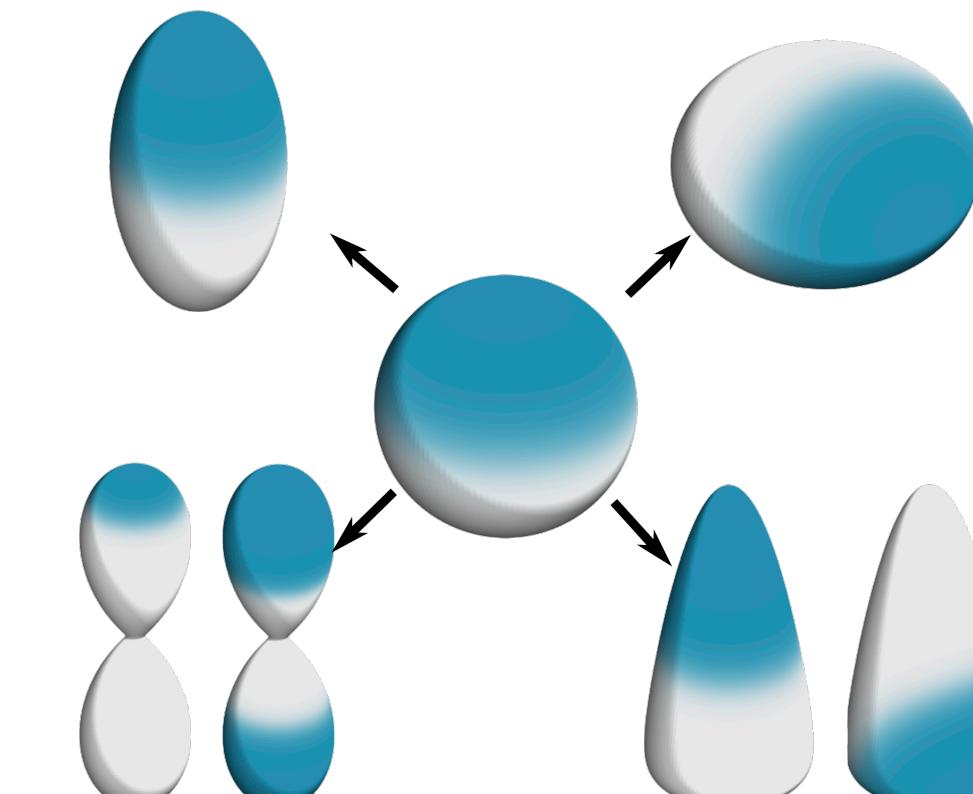
Embryonic furrow development [HHMI]



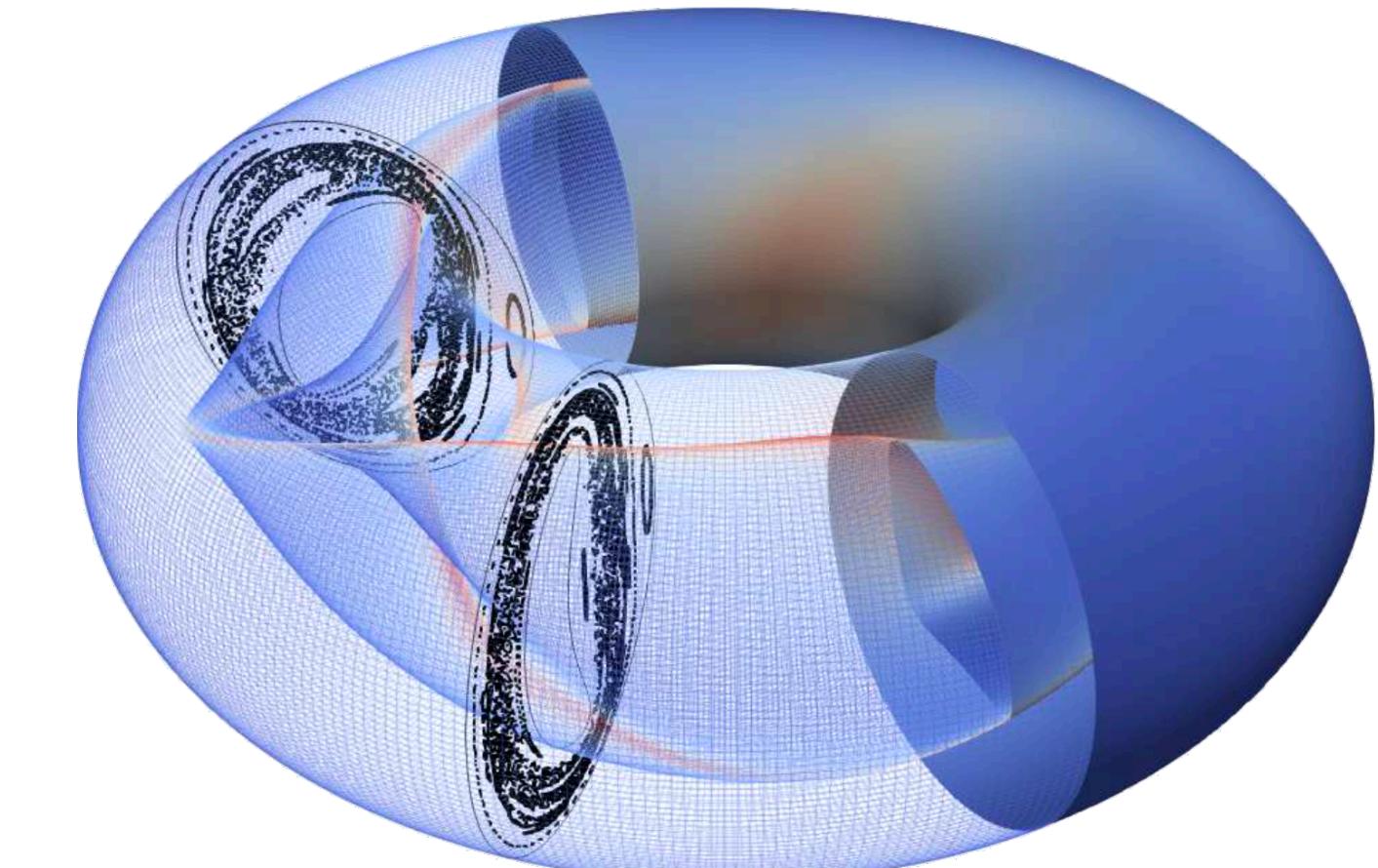
Vesicle flows [Veerapaneni et al., 2011]



Cell polarization [Miller, F., et al, 2022]



Stellarator design [Malhotra et al., 2019]

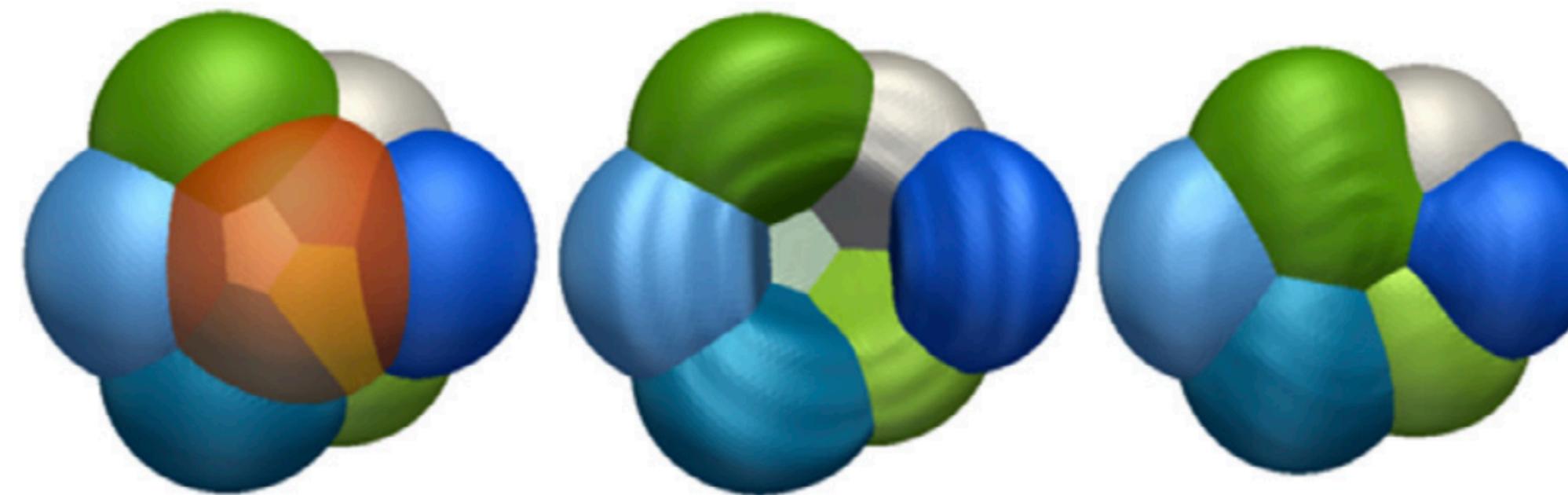


Introduction

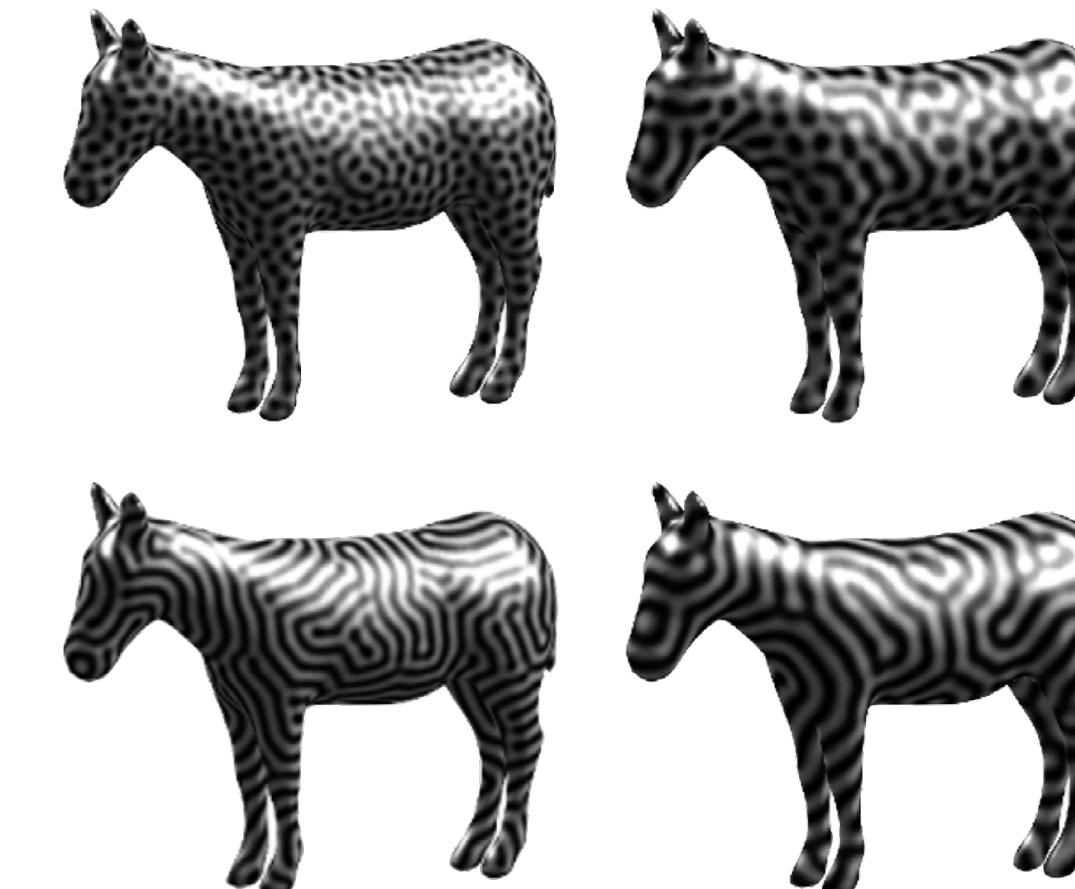
Why solve Surface PDEs?

Surface-bound phenomena arise in many applications.

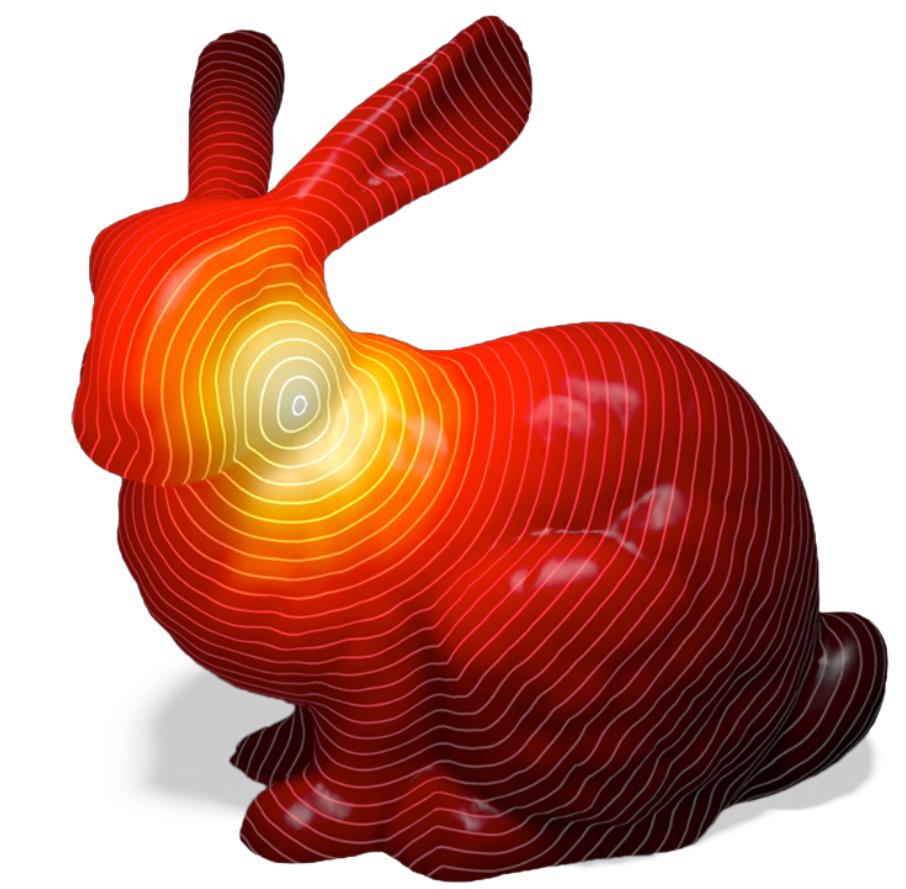
Thin-film hydrodynamics [Saye, 2016]



Pattern formation [Jeong, 2017]



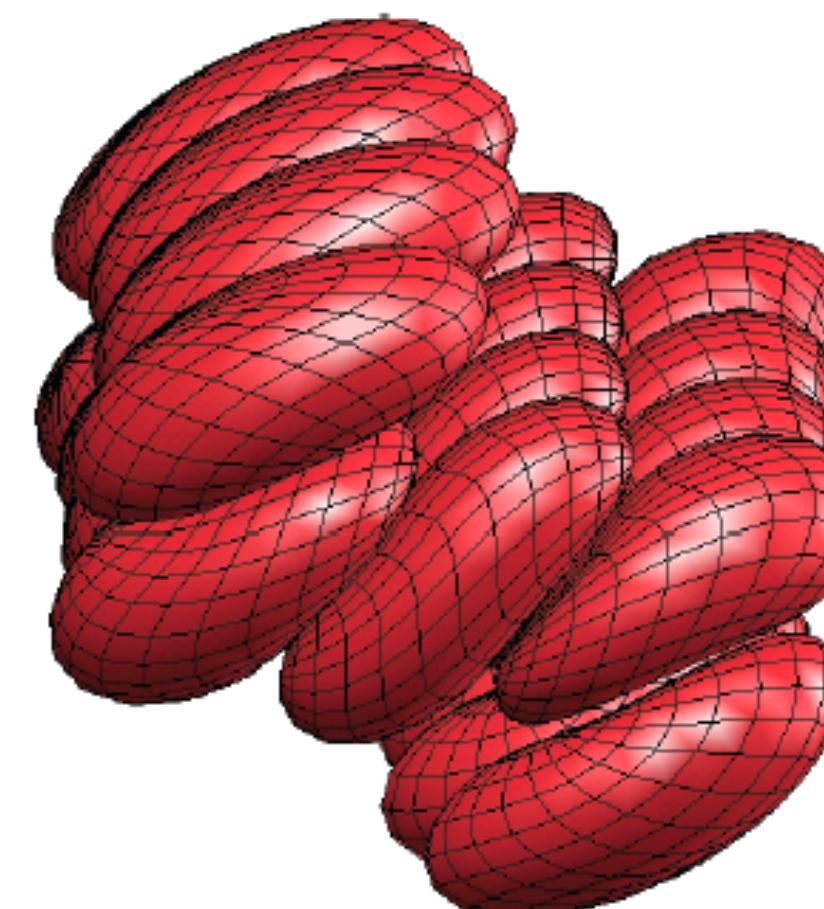
Geodesic distance [Crane et al., 2017]



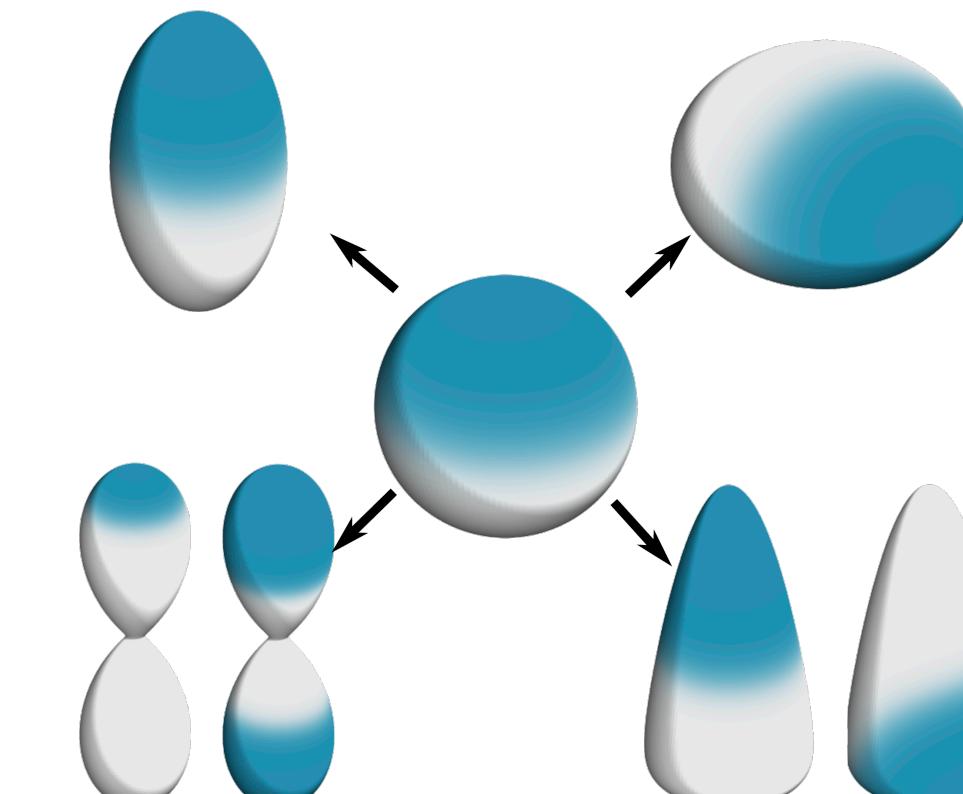
Embryonic furrow development [HHMI]



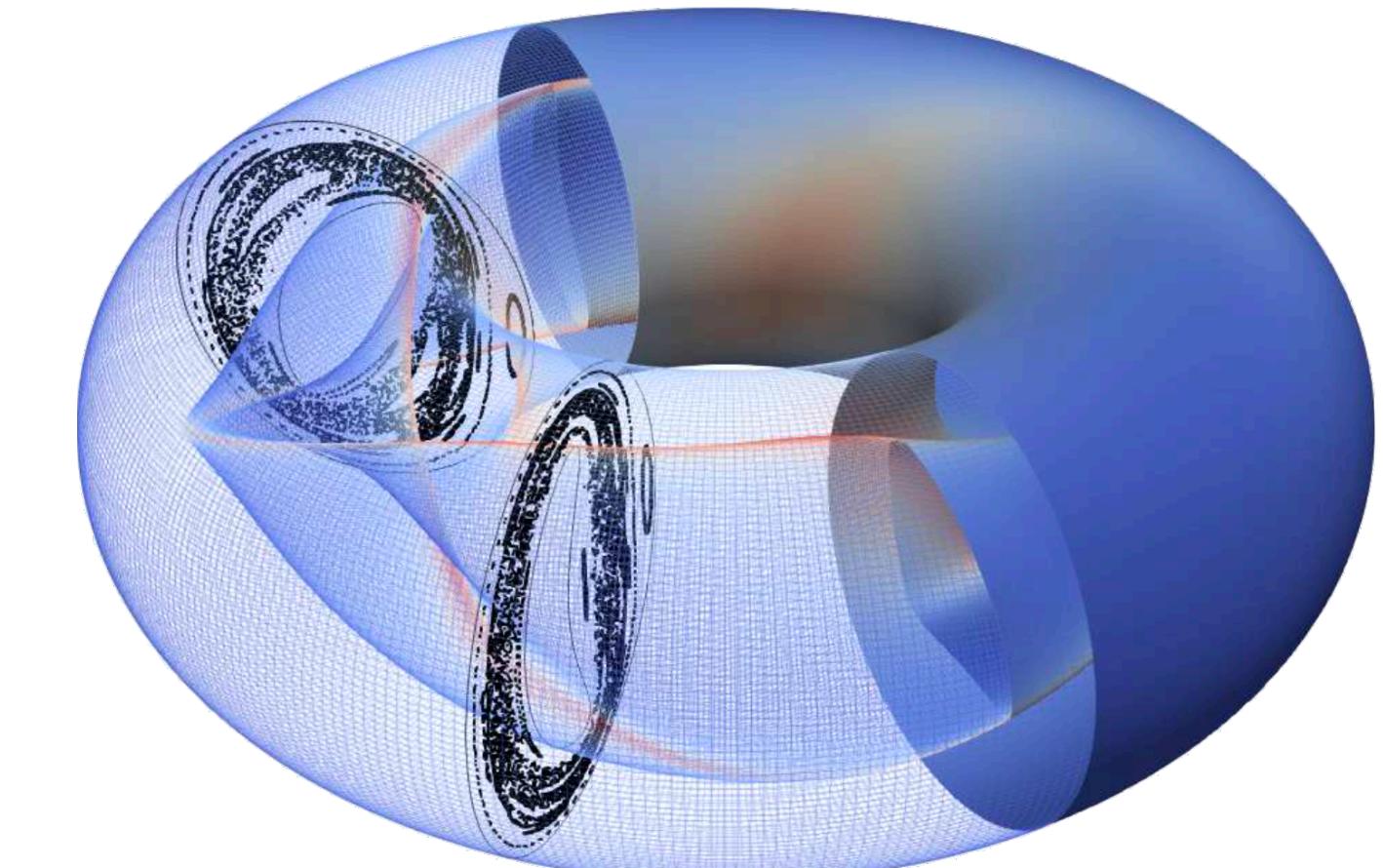
Vesicle flows [Veerapaneni et al., 2011]



Cell polarization [Miller, F., et al, 2022]



Stellarator design [Malhotra et al., 2019]



Introduction

Surface PDEs

Surface PDEs describe the dynamics of such phenomena.

Steady-state problem

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

Time-dependent problem

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

- Laplace–Beltrami
- convection–diffusion
- steady Stokes



Implicit time discretization:

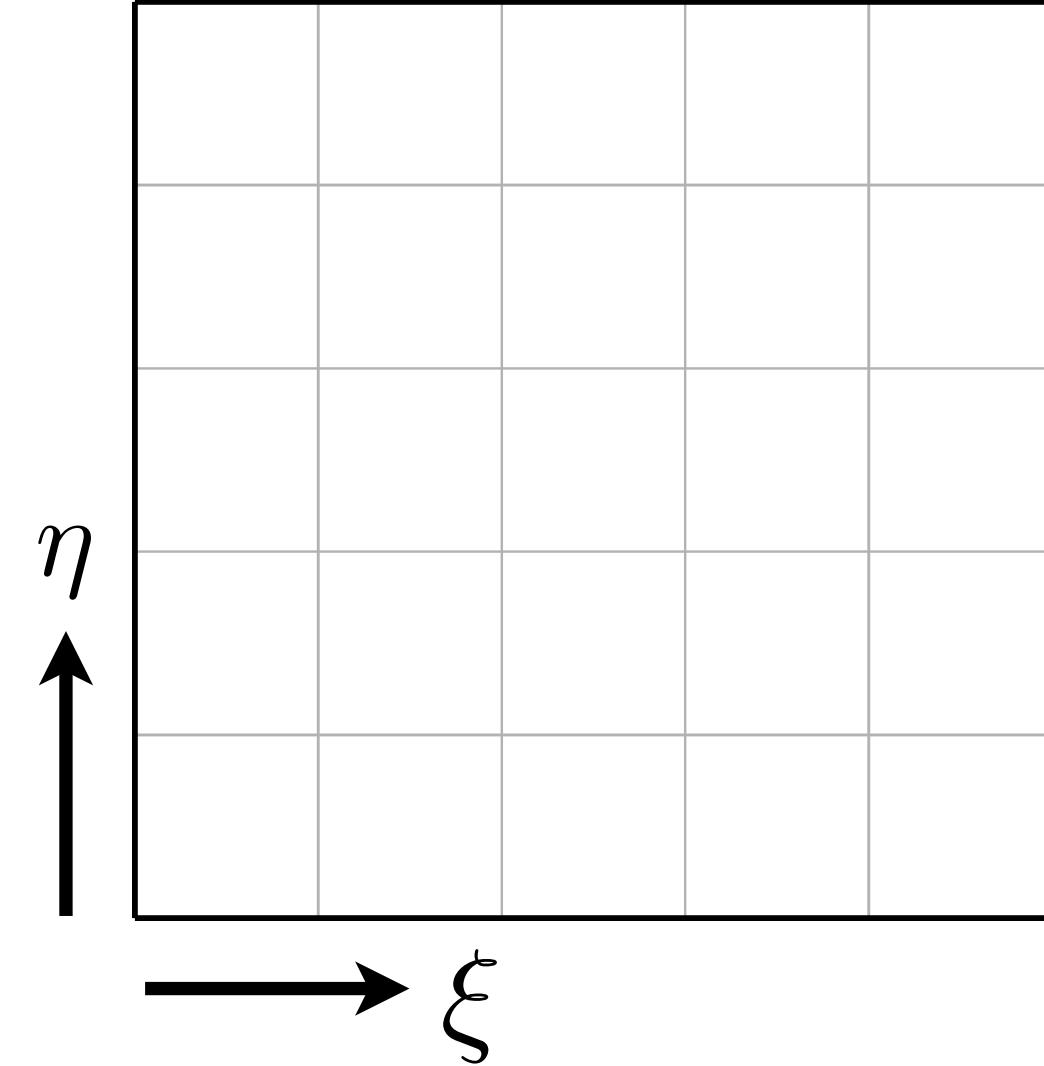
$$(I - \Delta t \mathcal{L}_\Gamma) u^{k+1} = u^k + \Delta t \mathcal{N}(u^k)$$

- reaction–diffusion
- heat
- Navier–Stokes

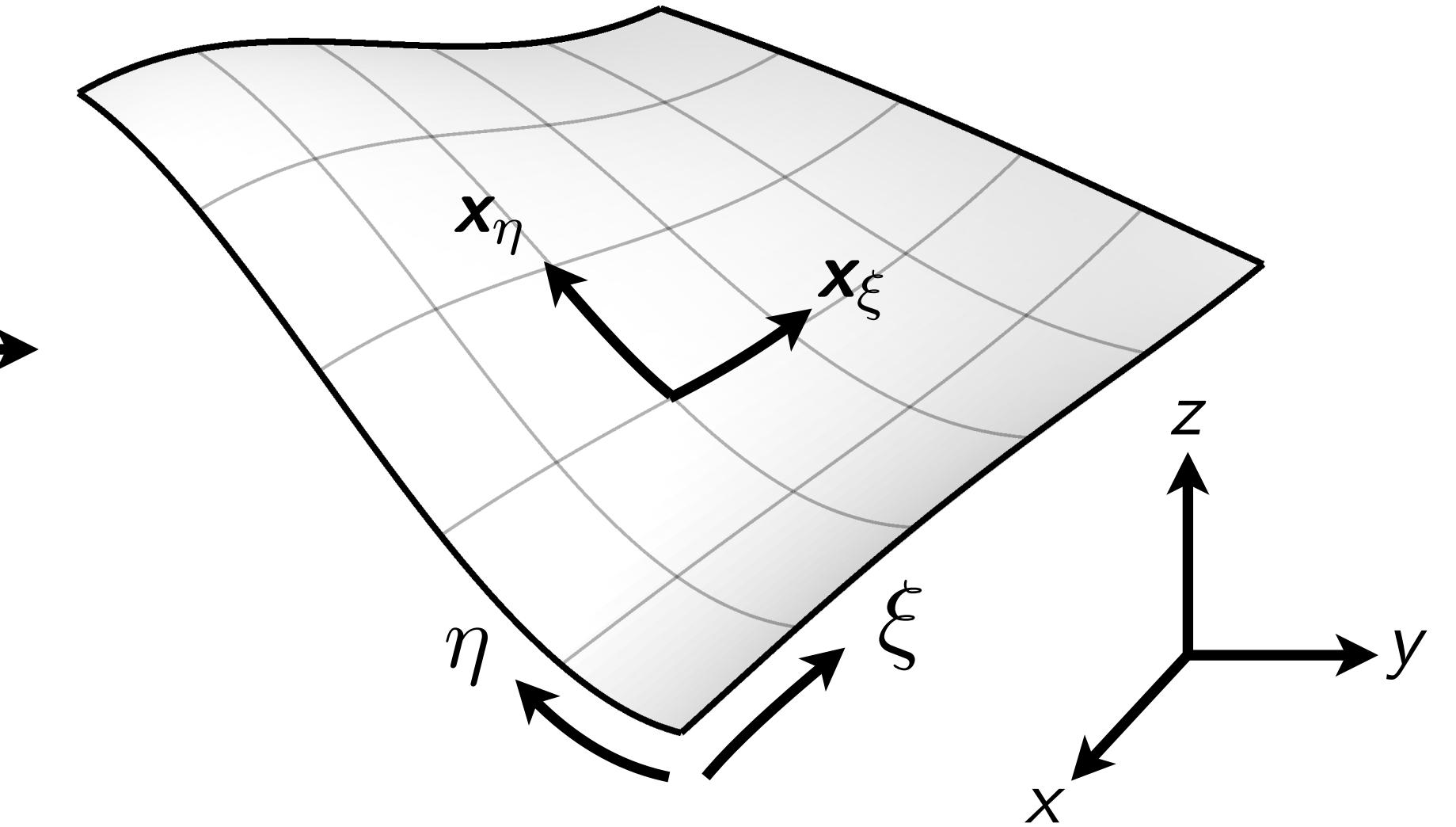
Model surface PDE: $\nabla_\Gamma \cdot (\mathbf{A}(\mathbf{x}) \nabla_\Gamma u(\mathbf{x})) + \nabla_\Gamma \cdot (\mathbf{b}(\mathbf{x}) u(\mathbf{x})) + c(\mathbf{x}) u(\mathbf{x}) = f(\mathbf{x})$
(+ BCs if surface is not closed)

Surface PDEs

Differential operators



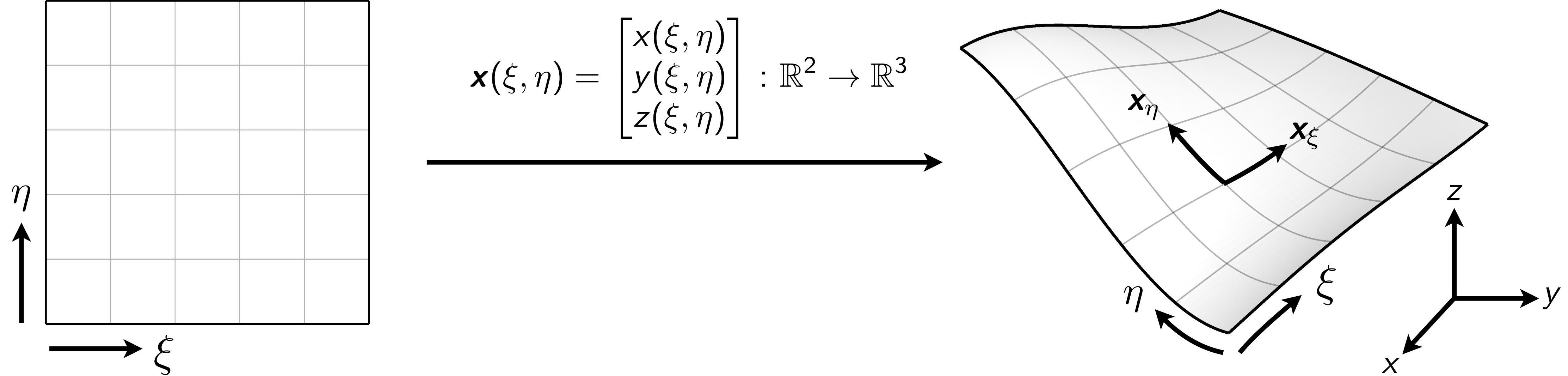
$$\mathbf{x}(\xi, \eta) = \begin{bmatrix} x(\xi, \eta) \\ y(\xi, \eta) \\ z(\xi, \eta) \end{bmatrix} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



- Metric tensor $g = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.

Surface PDEs

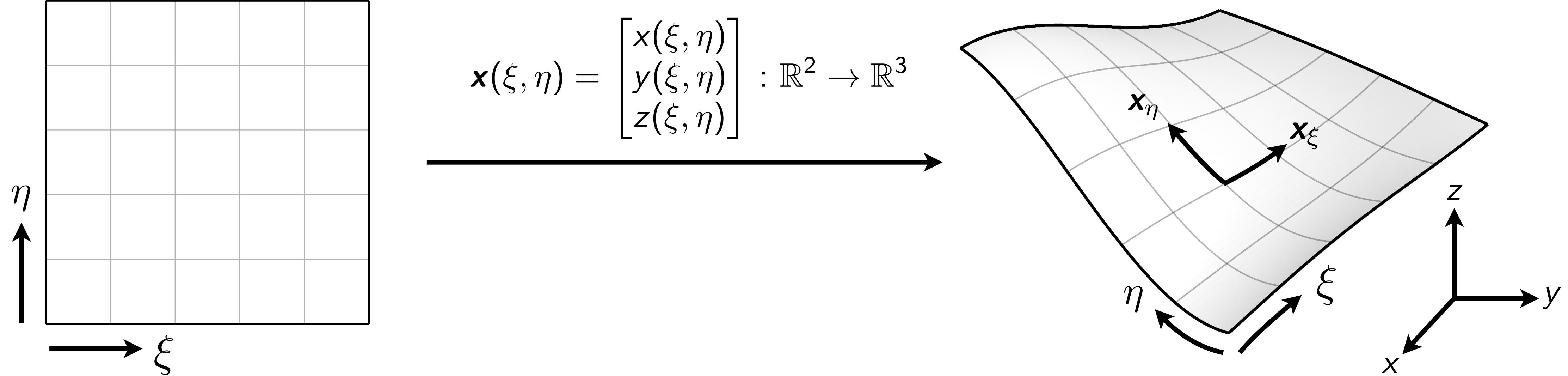
Differential operators



- Metric tensor $g = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.
- Surface gradient: $\nabla_{\Gamma} u = [\mathbf{x}_\xi \ \mathbf{x}_\eta] g^{-1} \nabla_{\xi\eta} u$

Surface PDEs

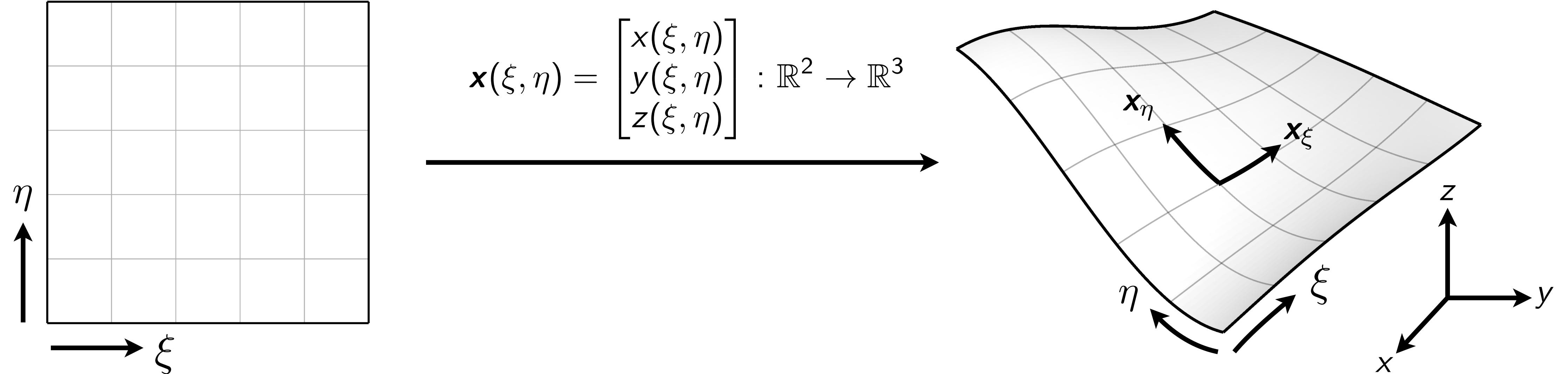
Differential operators



- Metric tensor $g = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.
- Surface gradient: $\nabla_{\Gamma} u = [\mathbf{x}_\xi \ \mathbf{x}_\eta] g^{-1} \nabla_{\xi\eta} u$
- Surface divergence: $\nabla_{\Gamma} \cdot \mathbf{u} = \frac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot (\sqrt{\det g} \ \mathbf{u})$

Surface PDEs

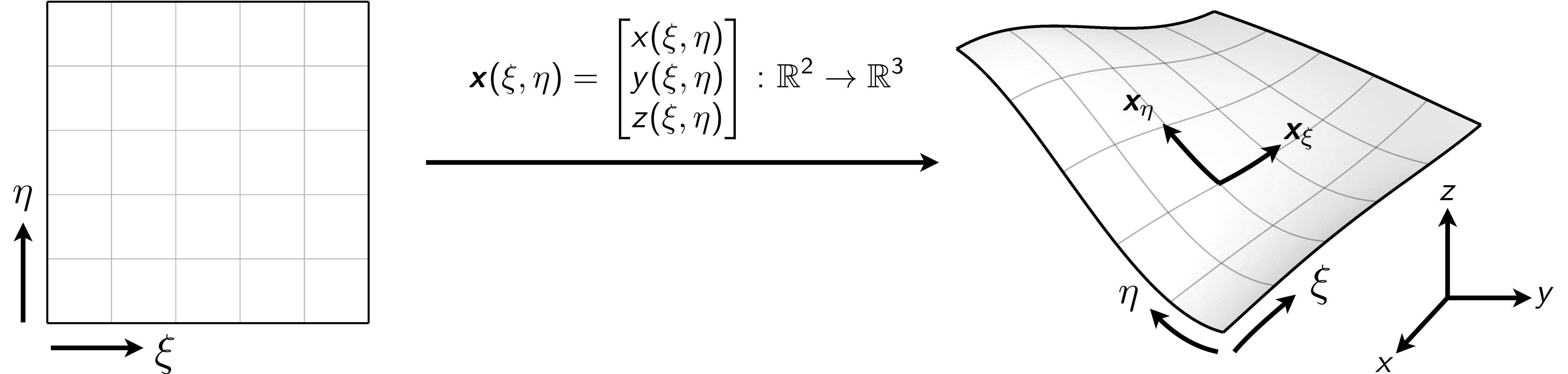
Differential operators



- Metric tensor $g = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.
- Surface gradient: $\nabla_\Gamma u = [\mathbf{x}_\xi \ \mathbf{x}_\eta] g^{-1} \nabla_{\xi\eta} u$
- Surface divergence: $\nabla_\Gamma \cdot \mathbf{u} = \frac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot (\sqrt{\det g} \mathbf{u})$
- Laplace–Beltrami: $\Delta_\Gamma u = \nabla_\Gamma \cdot \nabla_\Gamma u = \frac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot (\sqrt{\det g} g^{-1} \nabla_{\xi\eta} u)$

Surface PDEs

Differential operators

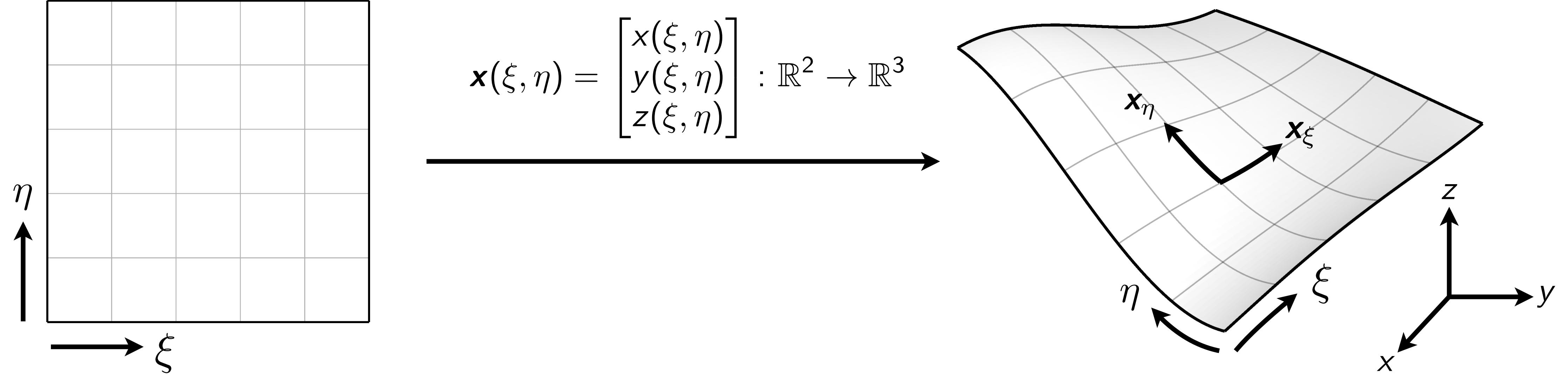


- Metric tensor $g = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix}$ encodes how lengths and angles change along surface.
- Surface gradient: $\nabla_\Gamma u = [\mathbf{x}_\xi \ \mathbf{x}_\eta] g^{-1} \nabla_{\xi\eta} u$
- Surface divergence: $\nabla_\Gamma \cdot \mathbf{u} = \frac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot (\sqrt{\det g} \mathbf{u})$
- Laplace–Beltrami: $\Delta_\Gamma u = \nabla_\Gamma \cdot \nabla_\Gamma u = \frac{1}{\sqrt{\det g}} \nabla_{\xi\eta} \cdot (\sqrt{\det g} g^{-1} \nabla_{\xi\eta} u)$

$$\begin{aligned}\partial_x^\Gamma &= \mathbf{e}_x \cdot \nabla_\Gamma \\ \partial_y^\Gamma &= \mathbf{e}_y \cdot \nabla_\Gamma \\ \partial_z^\Gamma &= \mathbf{e}_z \cdot \nabla_\Gamma\end{aligned}$$

Surface PDEs

Differential operators



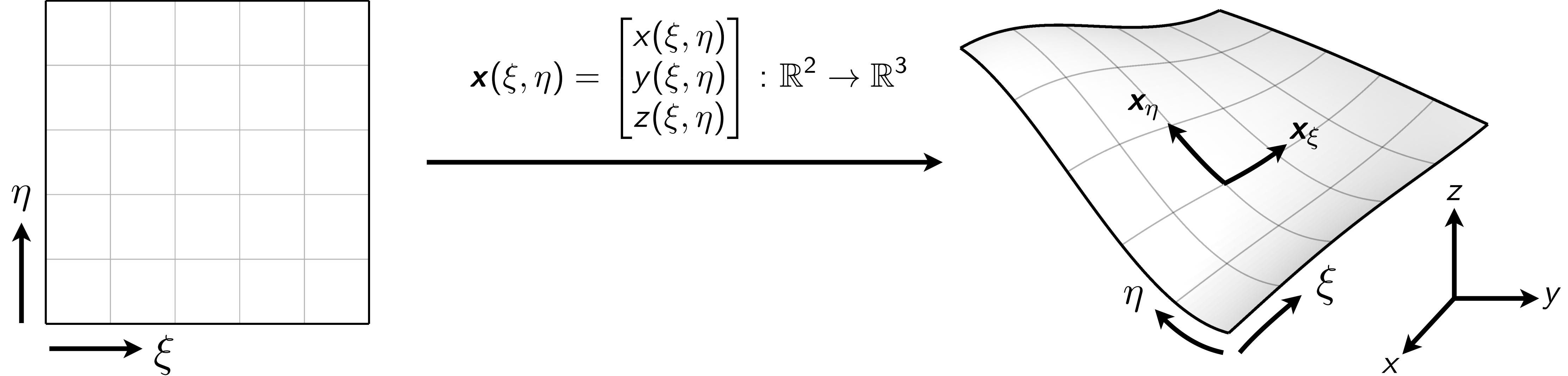
- Working in the embedding space makes things much simpler:

$$\mathcal{L}_\Gamma u(\mathbf{x}) = \sum_{i=1}^3 \sum_{j=1}^3 a_{ij}(\mathbf{x}) \partial_i^\Gamma \partial_j^\Gamma u(\mathbf{x}) + \sum_{i=1}^3 b_i(\mathbf{x}) \partial_i^\Gamma u(\mathbf{x}) + c(\mathbf{x}) u(\mathbf{x})$$

$$\boxed{\begin{aligned}\partial_x^\Gamma &= \mathbf{e}_x \cdot \nabla_\Gamma \\ \partial_y^\Gamma &= \mathbf{e}_y \cdot \nabla_\Gamma \\ \partial_z^\Gamma &= \mathbf{e}_z \cdot \nabla_\Gamma\end{aligned}}$$

Surface PDEs

Differential operators



- Working in the embedding space makes things much simpler:

$$\mathcal{L}_\Gamma u(\mathbf{x}) = \sum_{i=1}^3 \sum_{j=1}^3 a_{ij}(\mathbf{x}) \partial_i^\Gamma \partial_j^\Gamma u(\mathbf{x}) + \sum_{i=1}^3 b_i(\mathbf{x}) \partial_i^\Gamma u(\mathbf{x}) + c(\mathbf{x}) u(\mathbf{x})$$

- For instance, the surface Laplacian formula now looks like the 3D Laplacian:

$$\mathcal{L}_\Gamma = \partial_x^\Gamma \partial_x^\Gamma + \partial_y^\Gamma \partial_y^\Gamma + \partial_z^\Gamma \partial_z^\Gamma$$

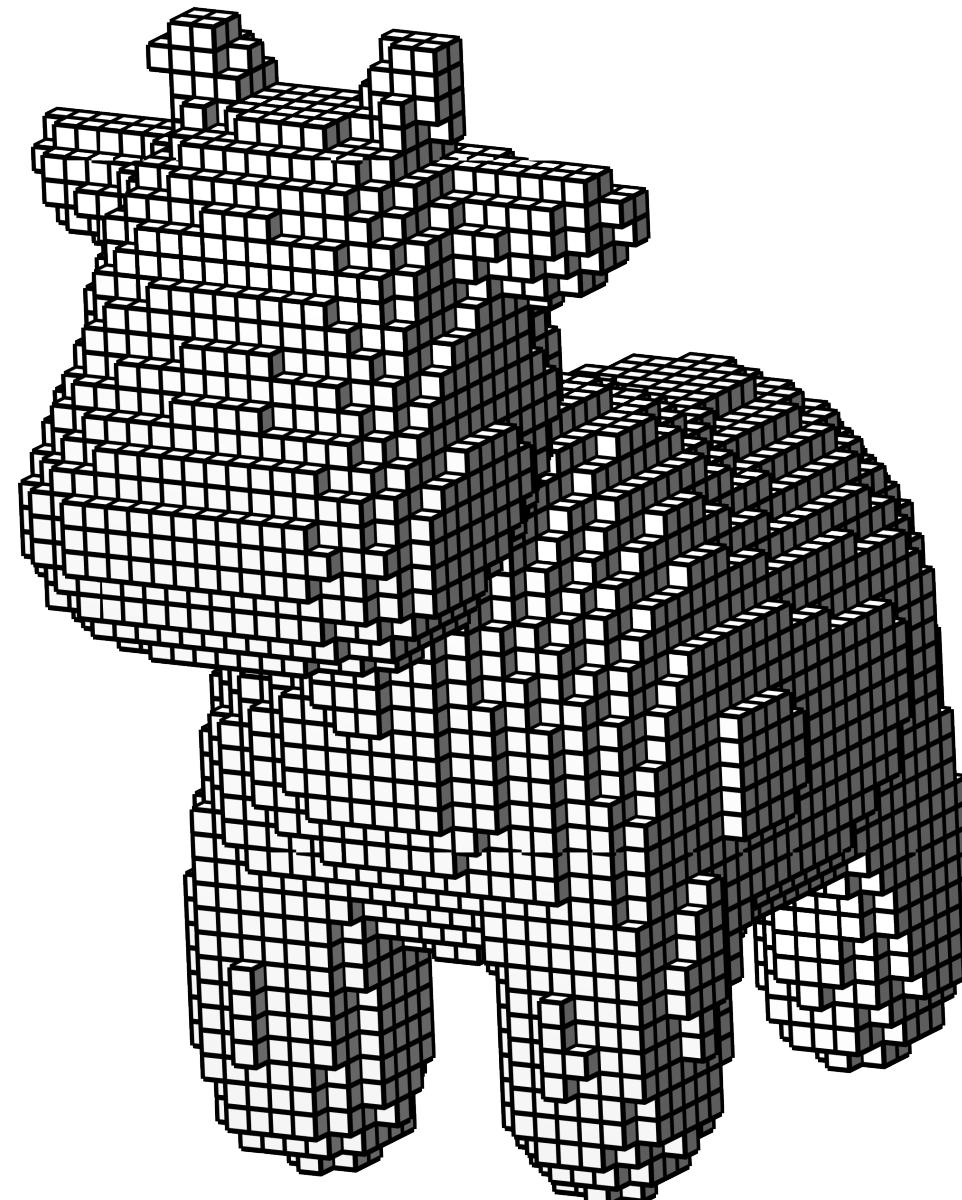
$$\boxed{\begin{aligned}\partial_x^\Gamma &= \mathbf{e}_x \cdot \nabla_\Gamma \\ \partial_y^\Gamma &= \mathbf{e}_y \cdot \nabla_\Gamma \\ \partial_z^\Gamma &= \mathbf{e}_z \cdot \nabla_\Gamma\end{aligned}}$$

The algorithms behind Surfacefun

Surface representation

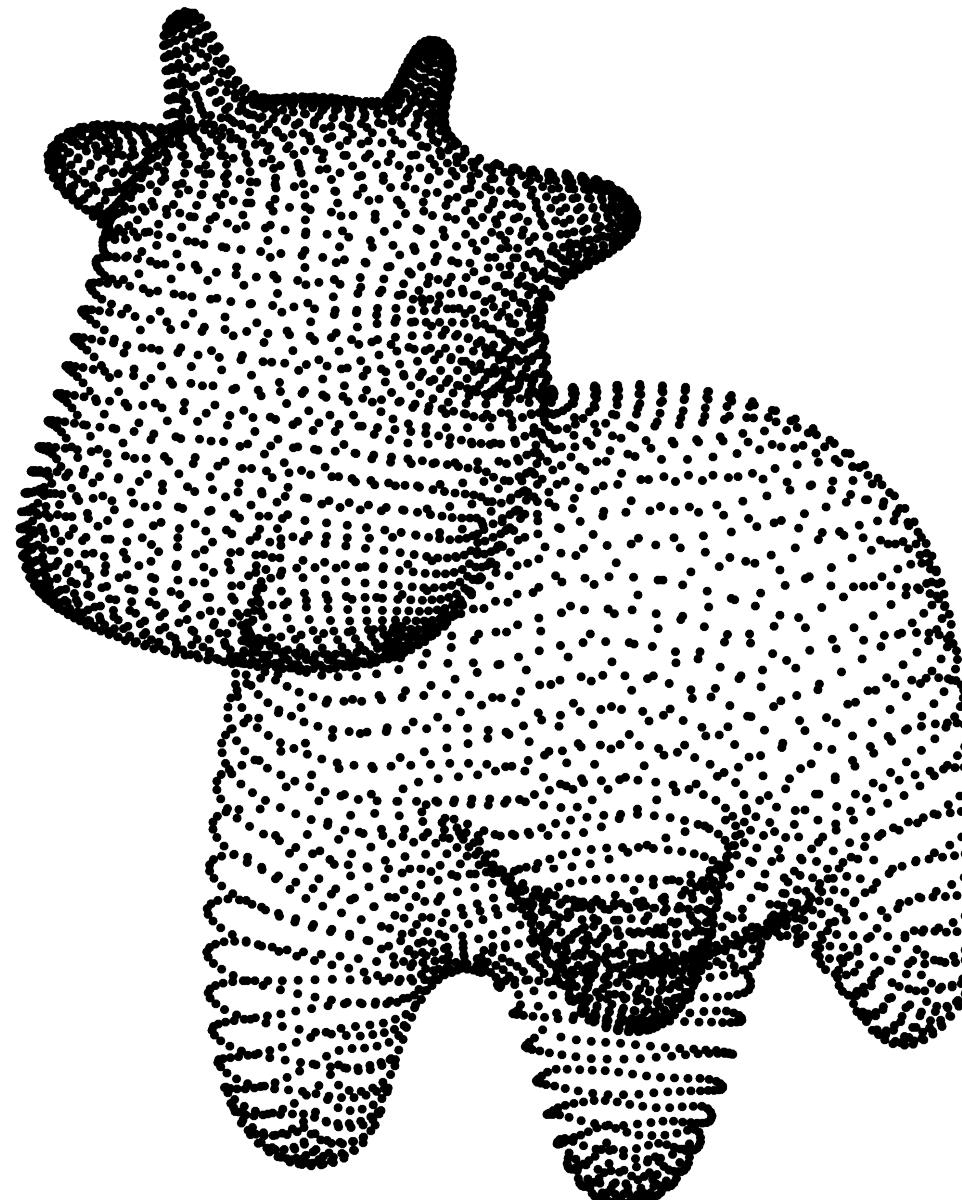
Many ways to represent a surface

Embedded grid



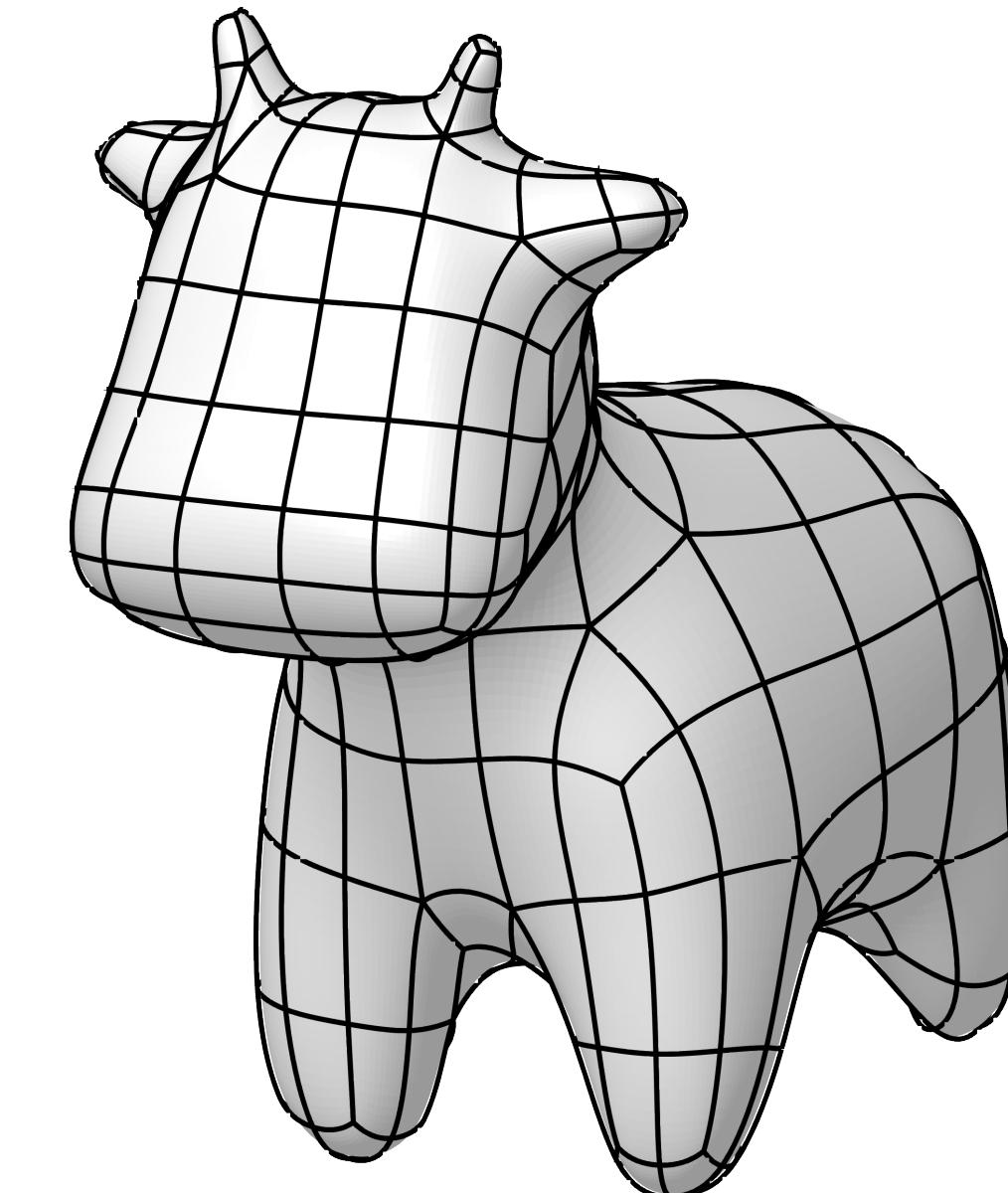
- Closest point methods
(Macdonald, Greer, Ruuth, ...)
- Level set methods
(Sethian, Osher, ...)

Unstructured point cloud



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

Surface mesh

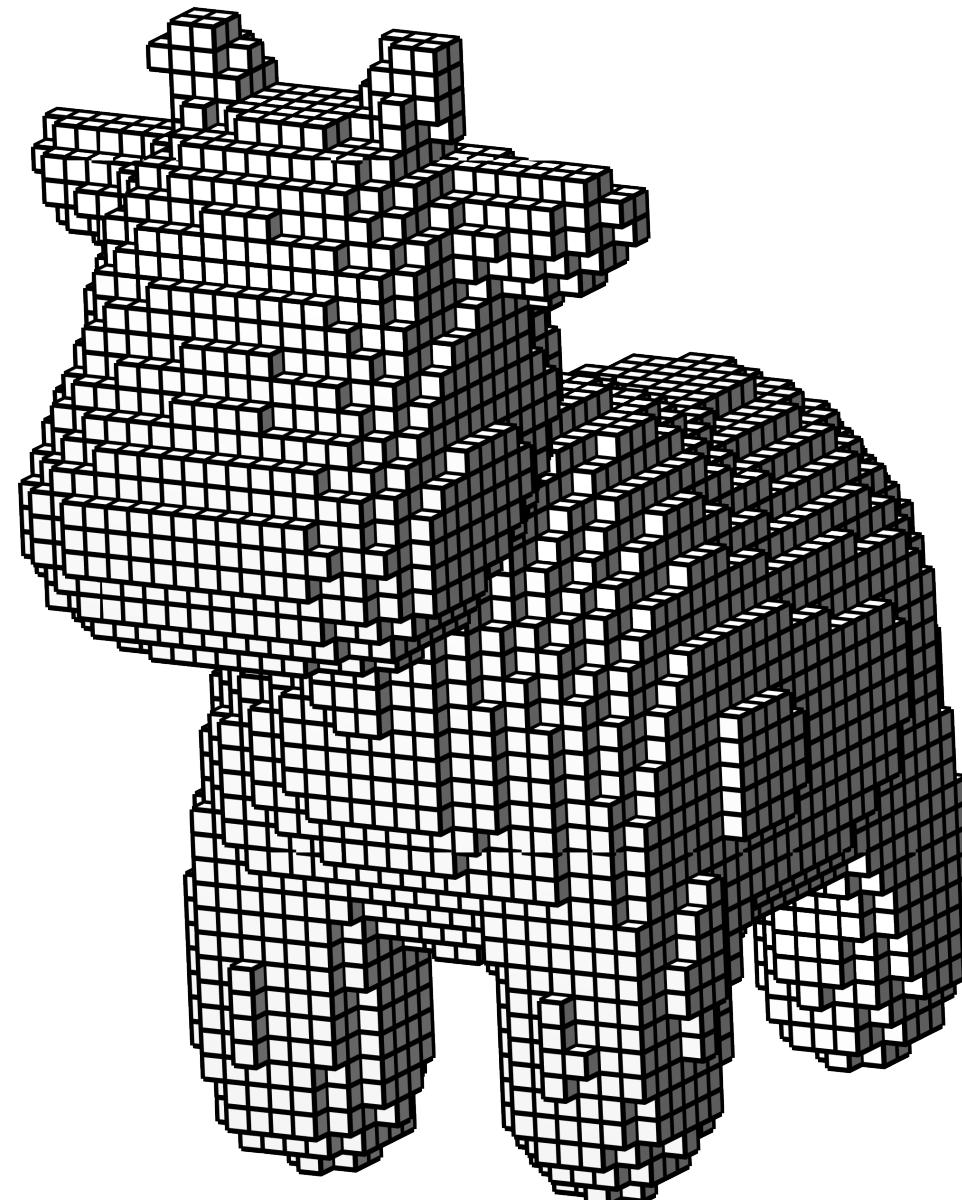


- Finite element methods
(Dziuk, Elliott, ...)
- Integral equation methods
(O'Neil, Goodwill, Rachh, ...)

Surface representation

Many ways to represent a surface

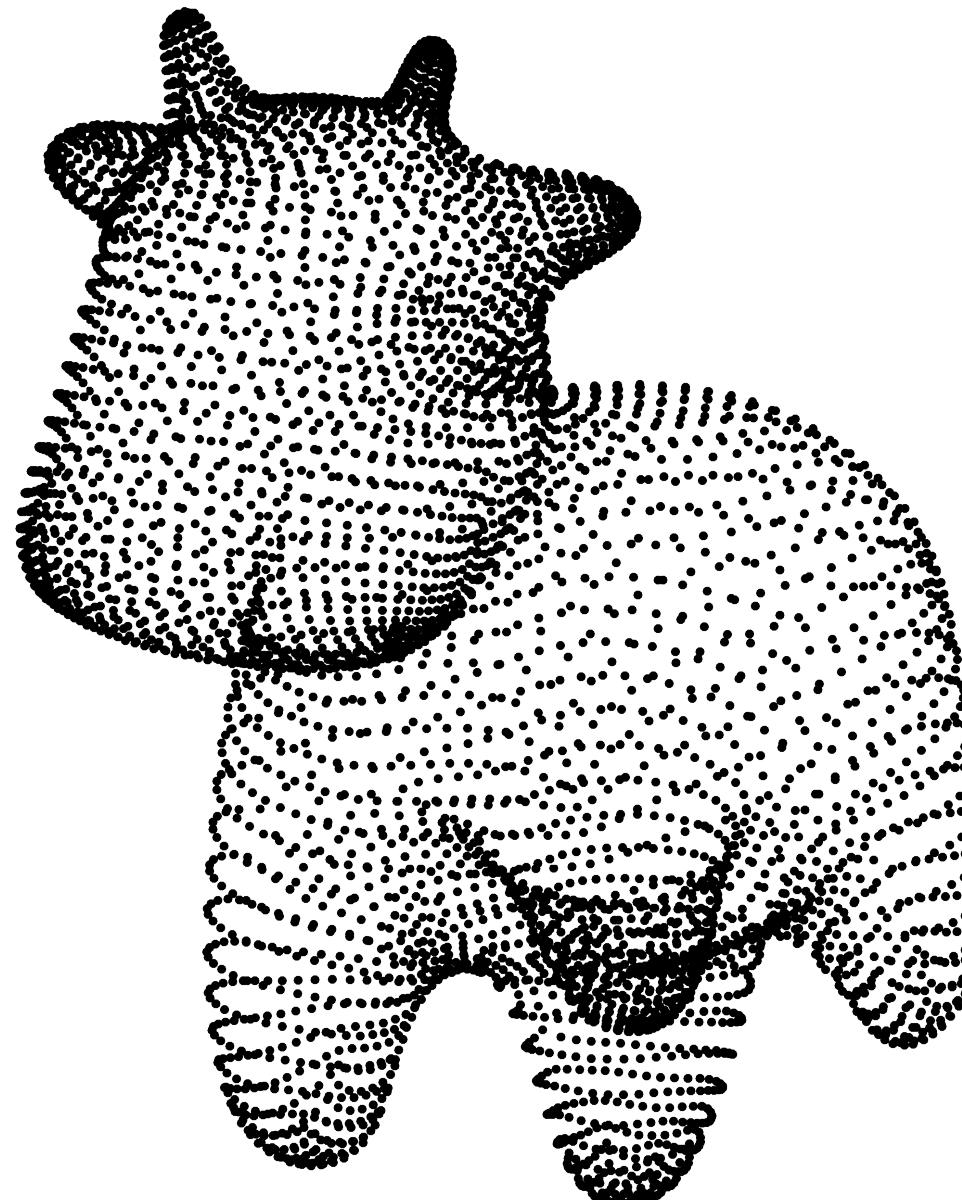
Embedded grid



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

high order → more volume DoFs

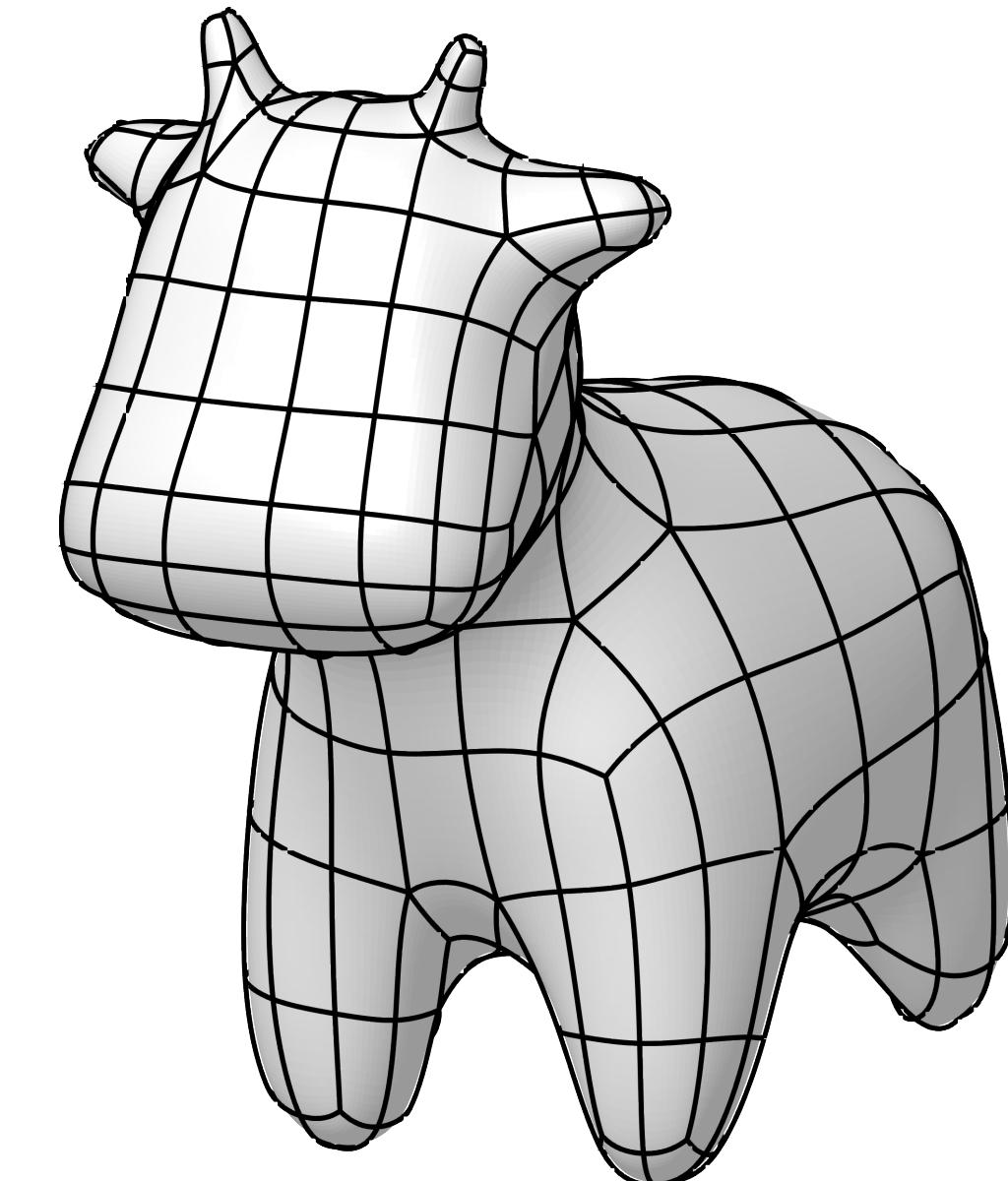
Unstructured point cloud



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

high order → ill-conditioned

Surface mesh



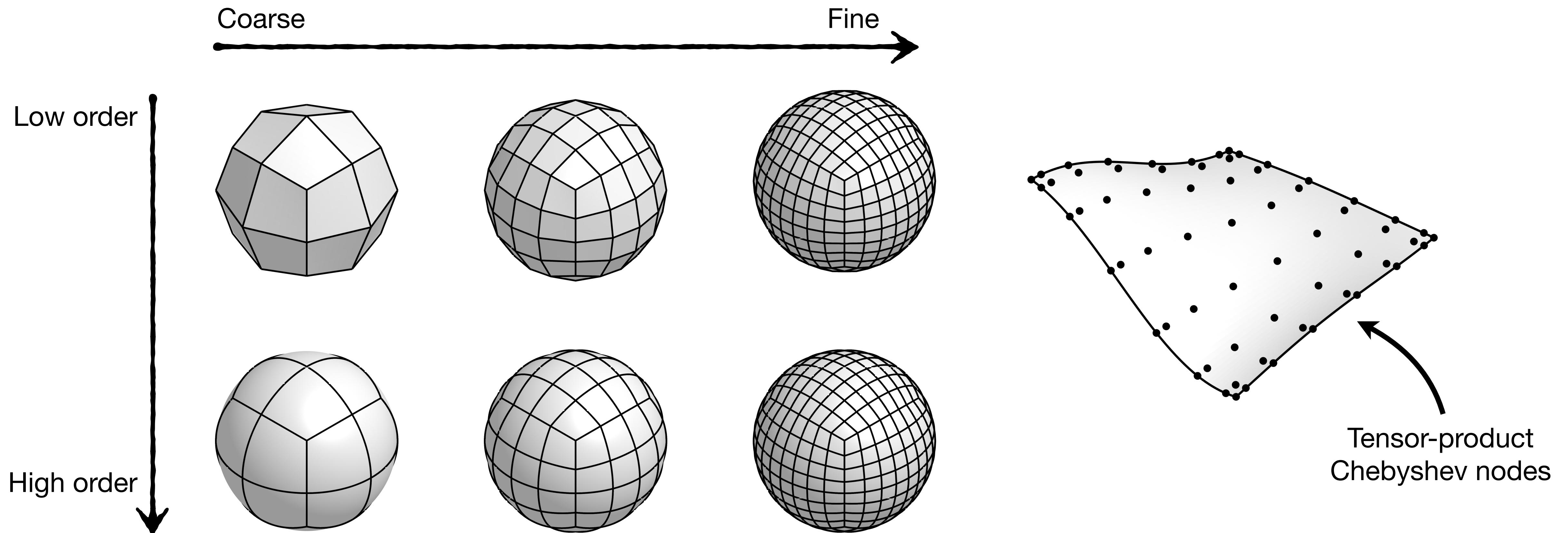
- Finite element methods
(Dziuk, Elliott, ...)
- Integral equation methods
(O'Neil, Goodwill, Rachh, ...)

high order → dense fill-in

Surface representation

Low-order vs. high-order

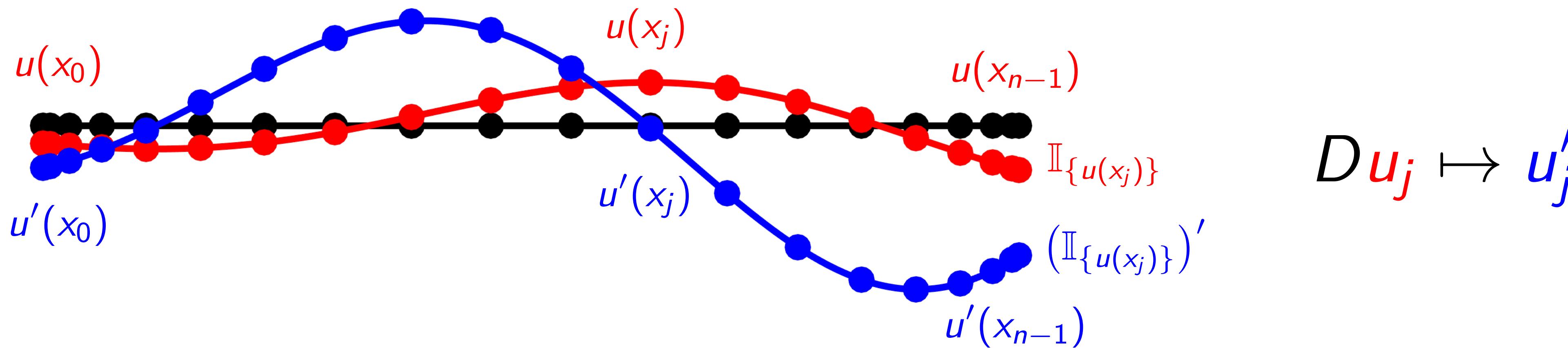
- Meshes are a good choice for CAD-compatibility. Surfacefun uses high-order quadrilateral patches.
- High-order elements allow faster convergence to solution.
- Coordinate maps of a patch are discretized via tabulation at Chebyshev nodes.



High-order discretization

Spectral collocation

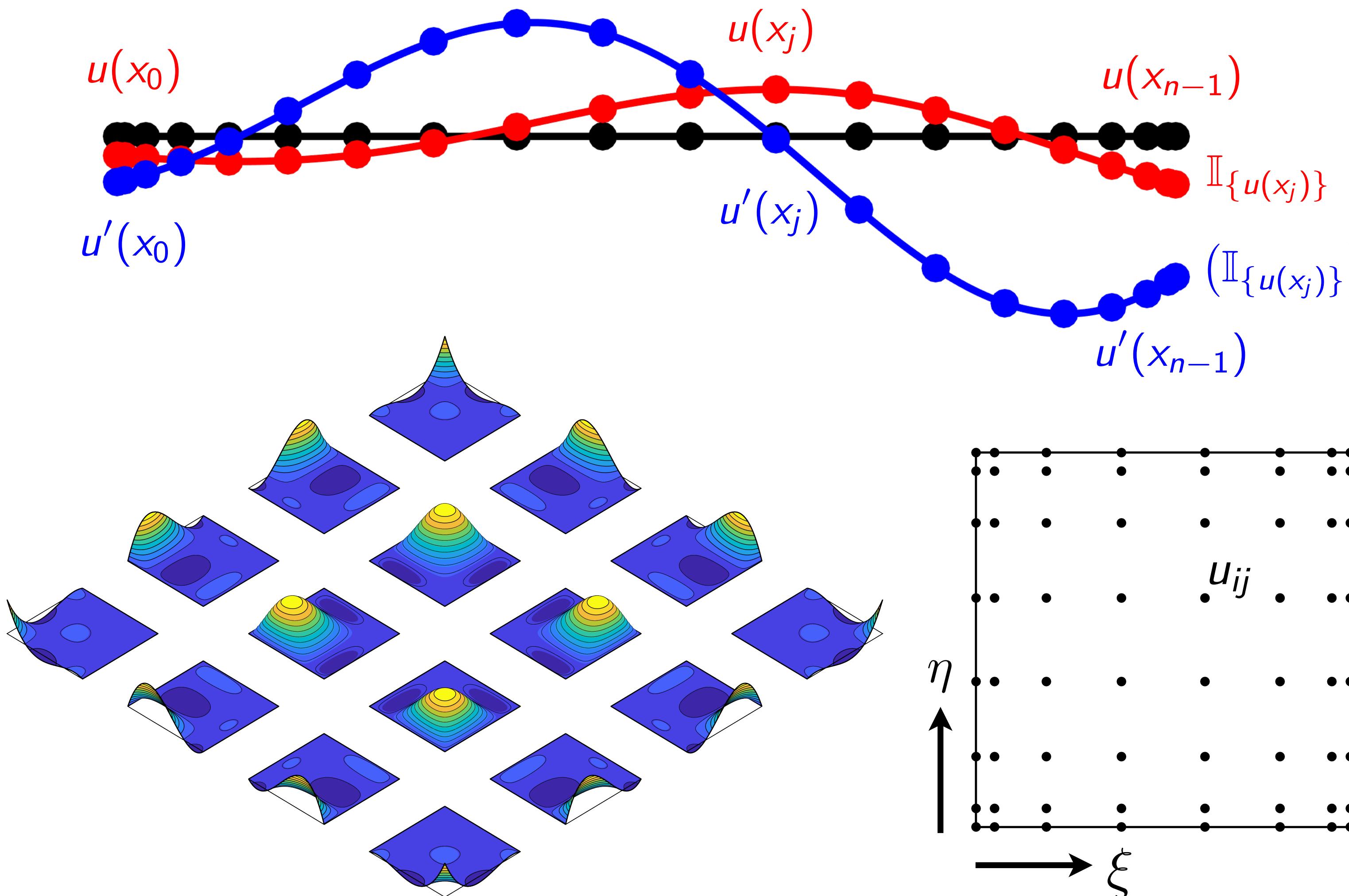
- Function values also stored at Chebyshev nodes.
- Derivatives and metric information (e.g. Jacobian) computed via spectral differentiation.



High-order discretization

Spectral collocation

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



$$D \mathbf{u}_j \mapsto \mathbf{u}'_j$$

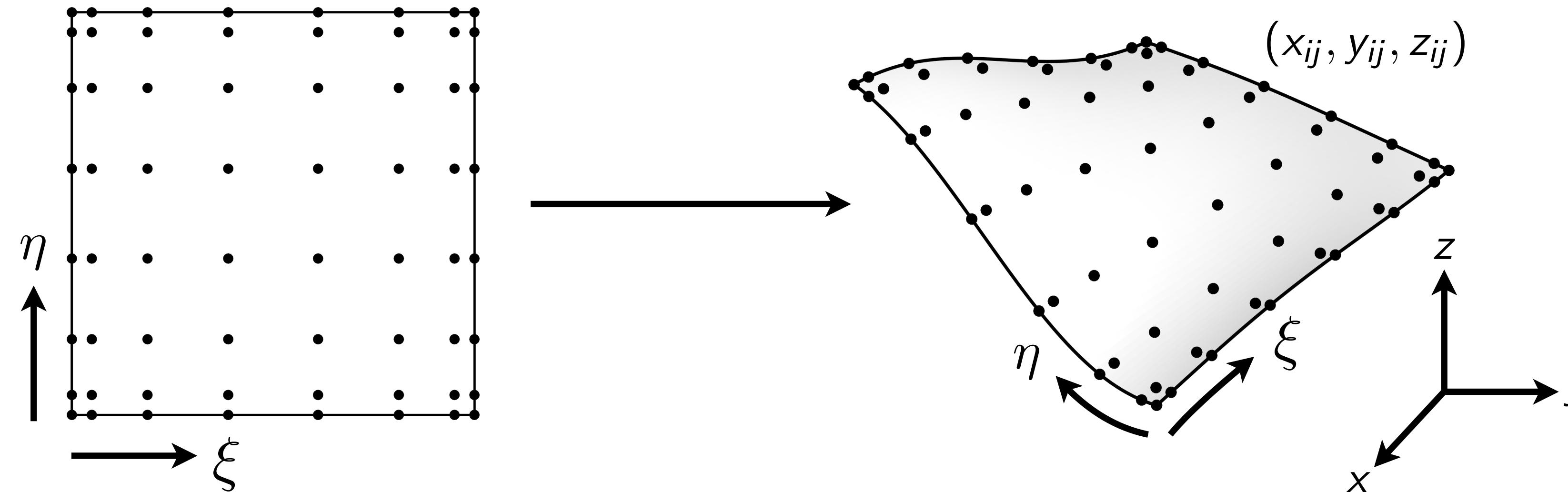
$$(D \otimes I) \mathbf{u}_{ij} \mapsto \partial u_{ij} / \partial \xi$$

$$(I \otimes D) \mathbf{u}_{ij} \mapsto \partial u_{ij} / \partial \eta$$

High-order discretization

Spectral collocation on a surface

- PDE is discretized through spectral differentiation and pointwise multiplication.



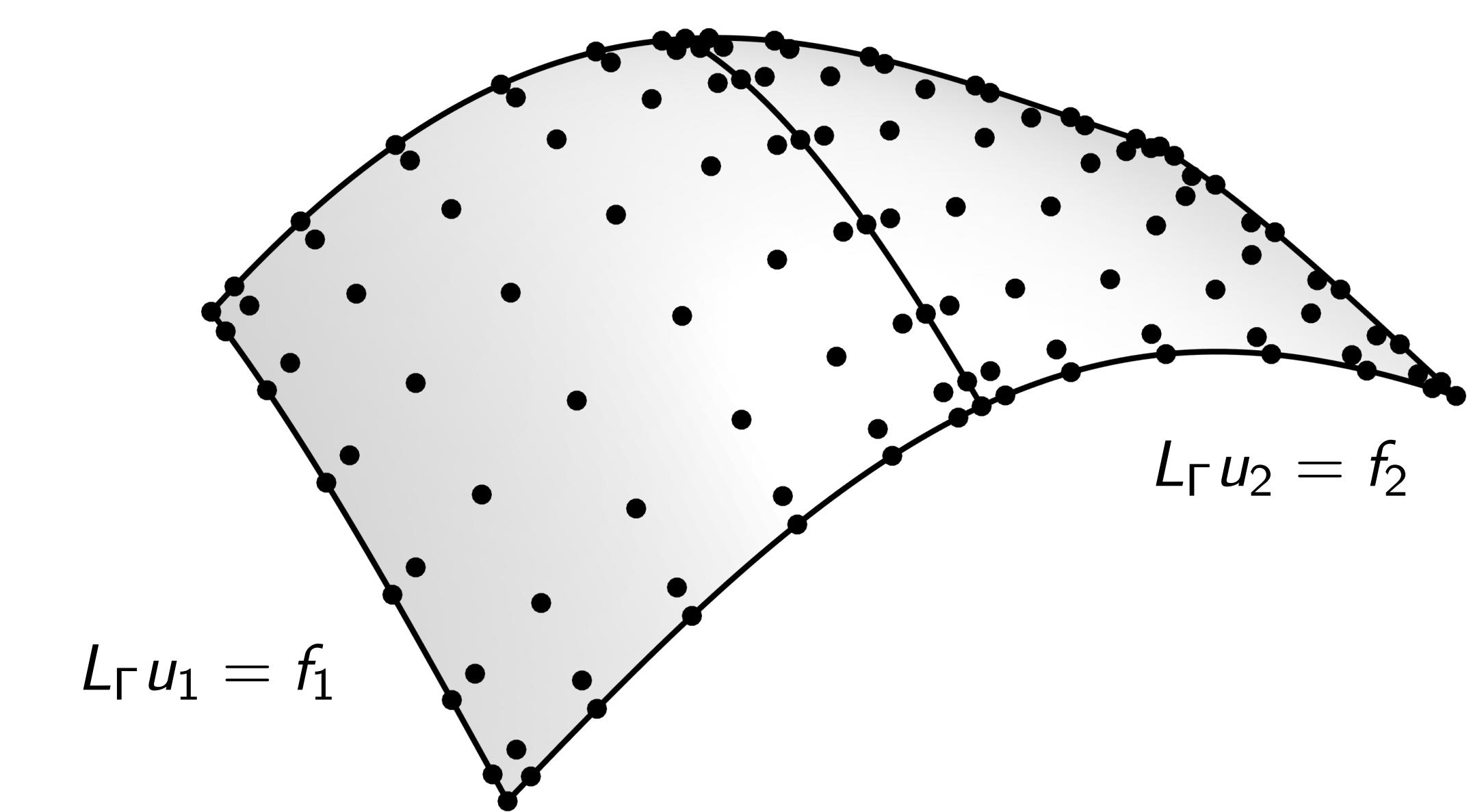
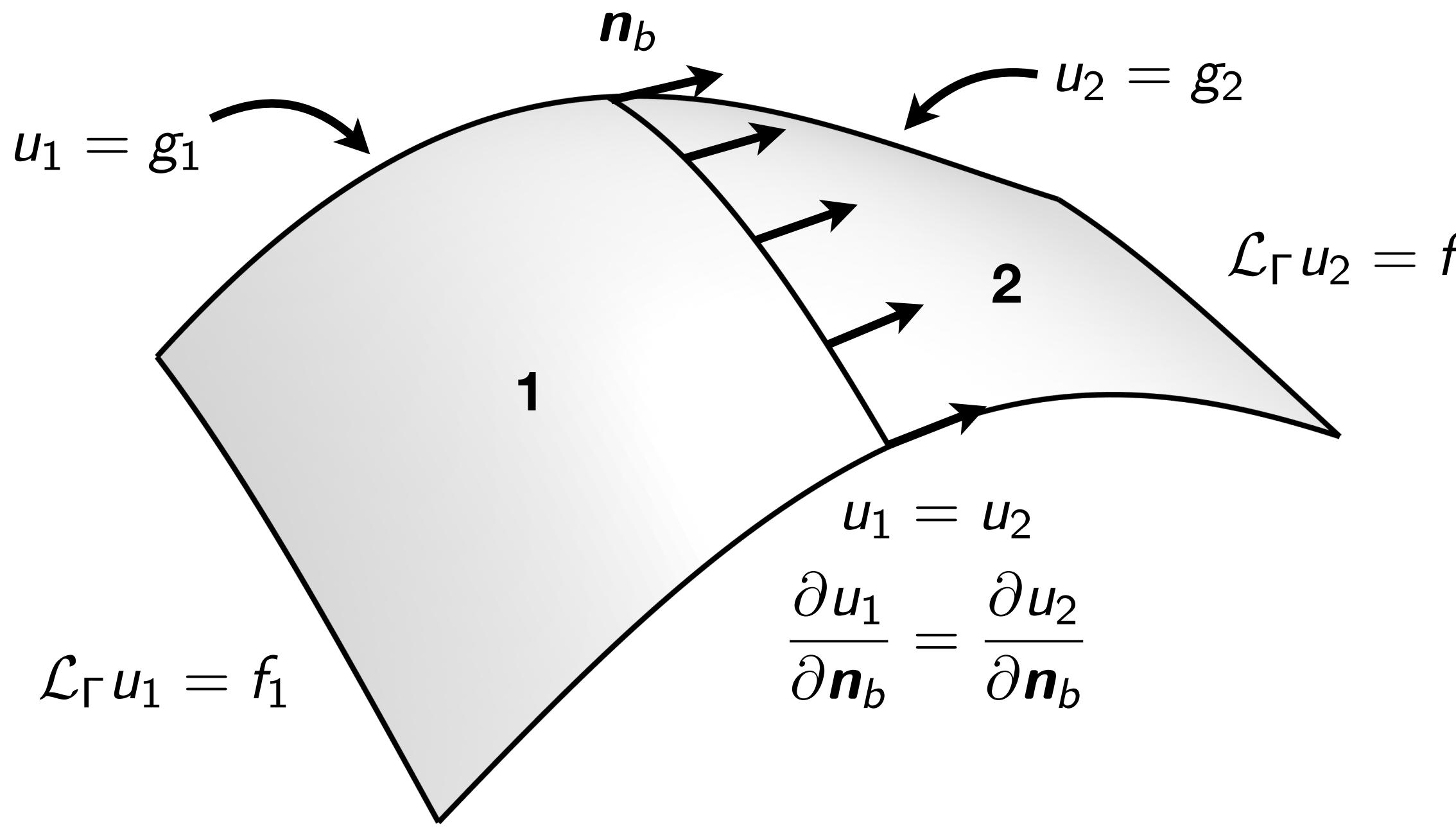
- For example, the discrete ∂_x^Γ operator is:

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

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

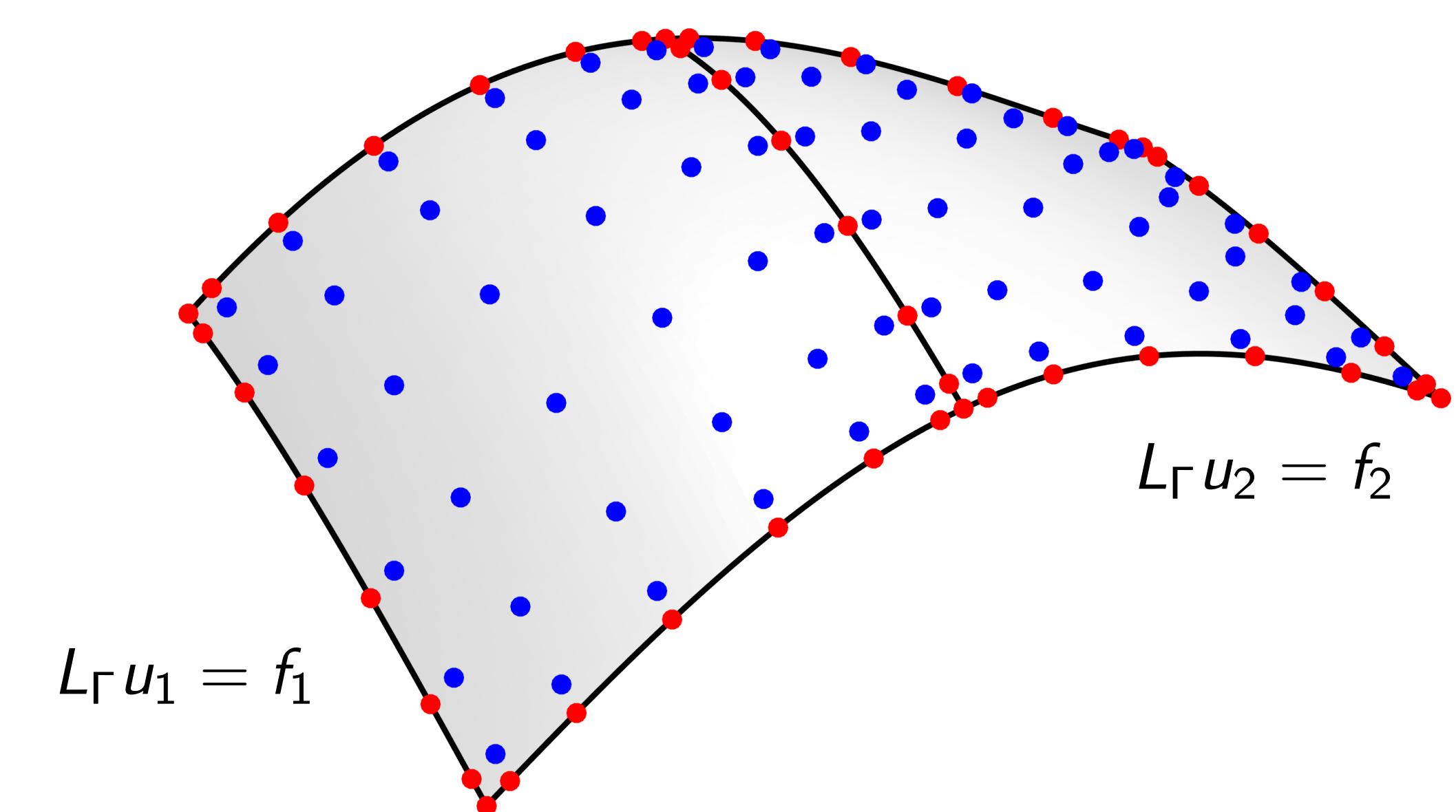
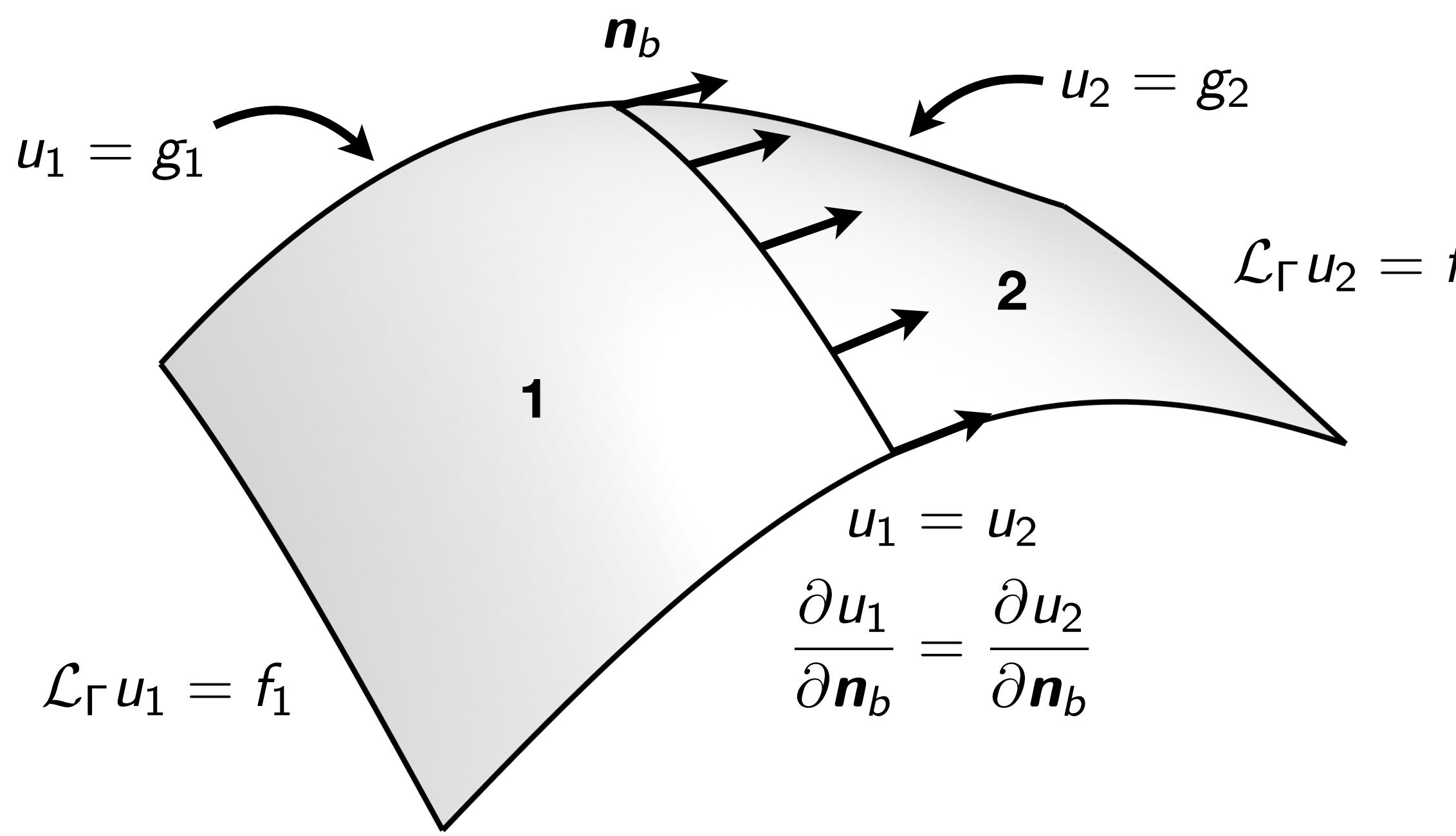
High-order discretization

Two glued patches



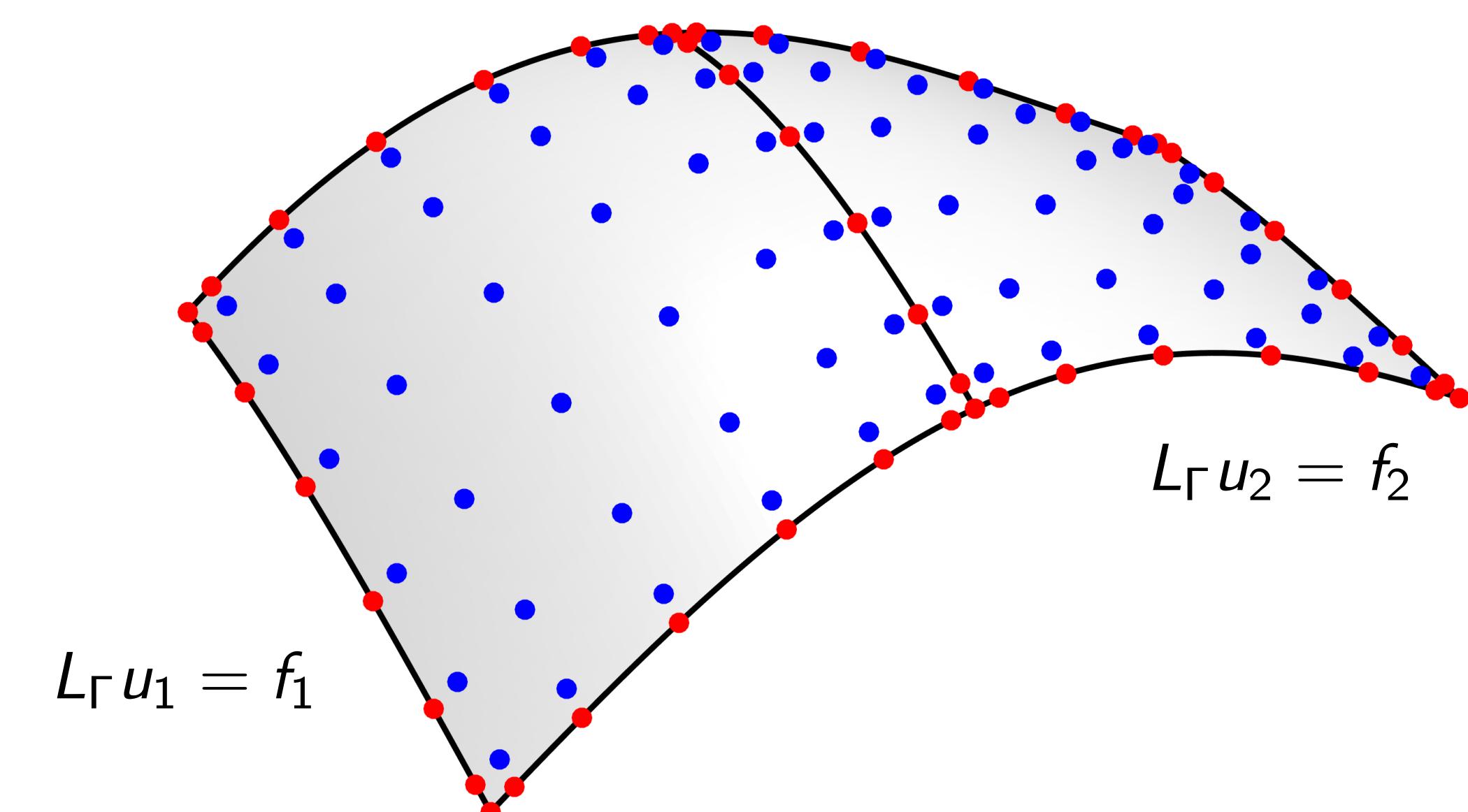
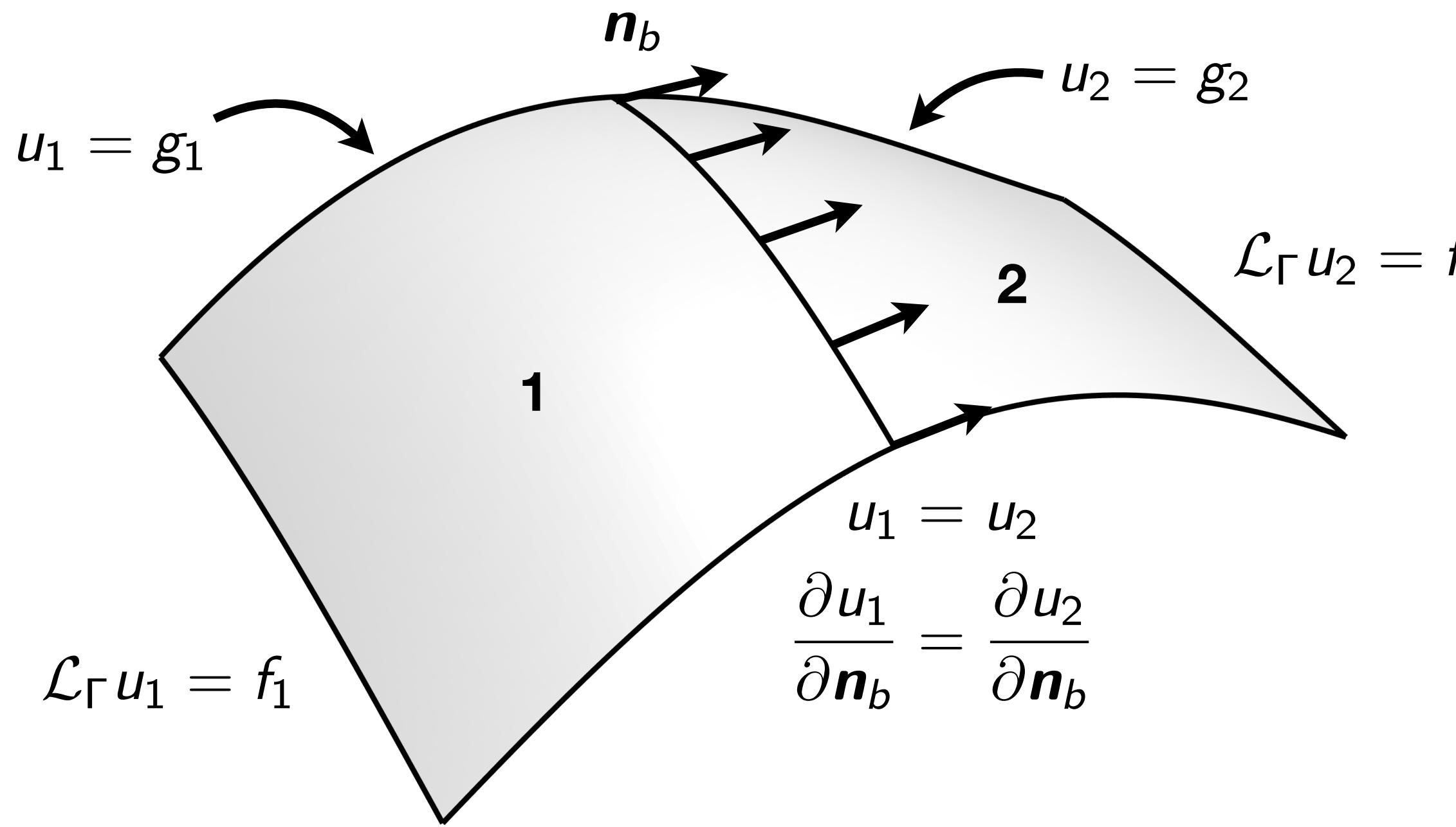
High-order discretization

Two glued patches



High-order discretization

Two glued patches



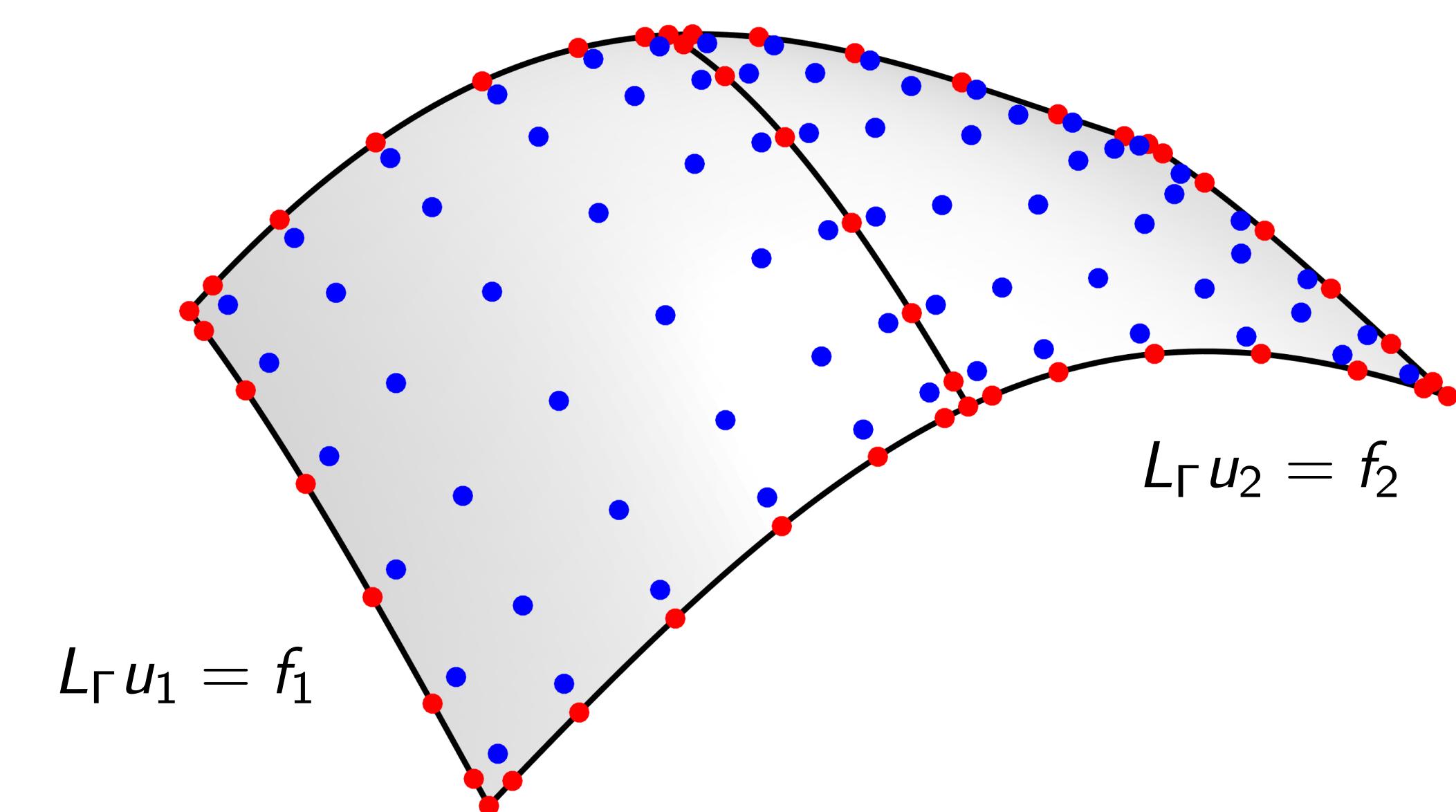
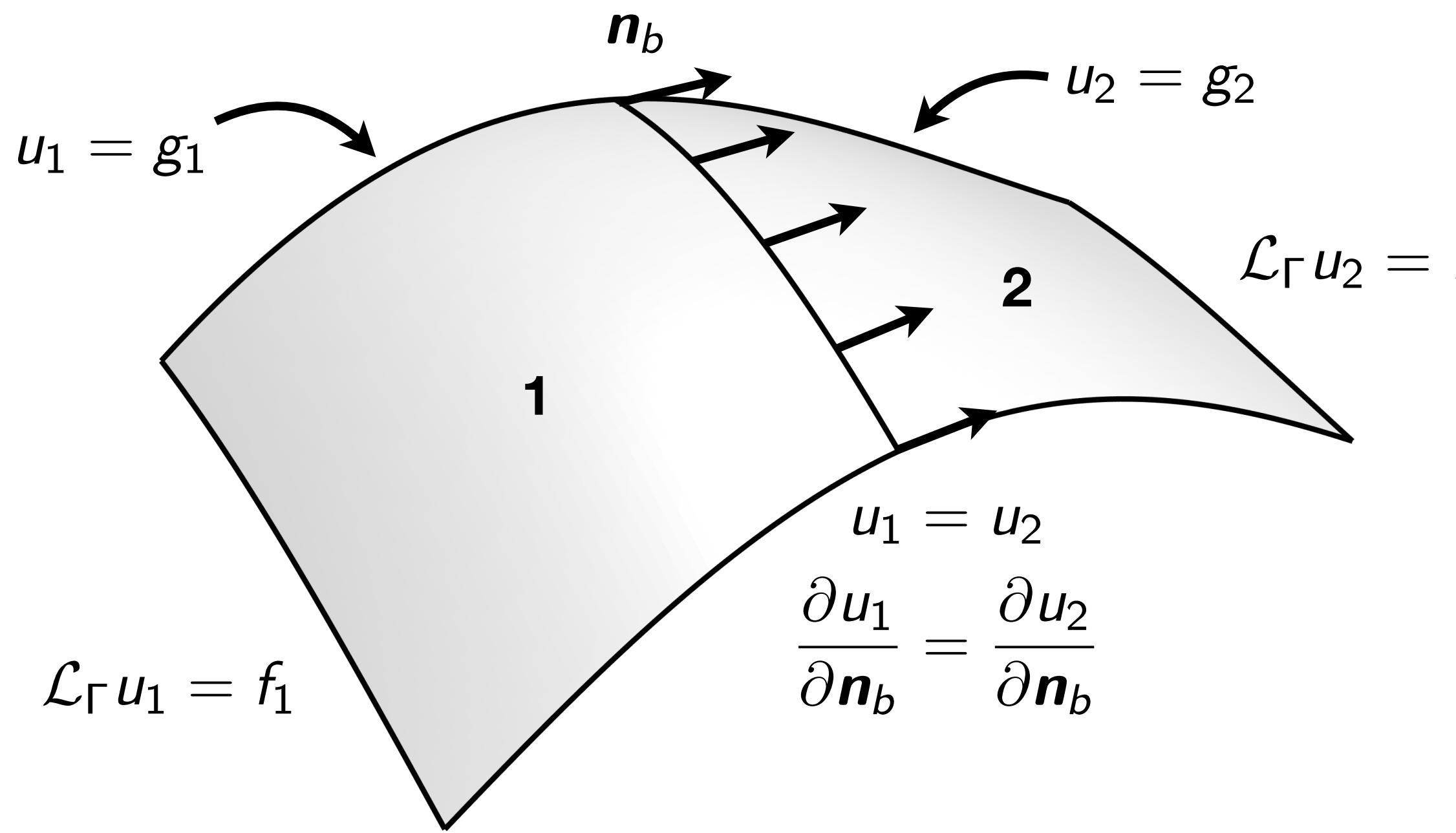
- Know how to do **local** solves on each element:

“Solution operator”

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

High-order discretization

Two glued patches

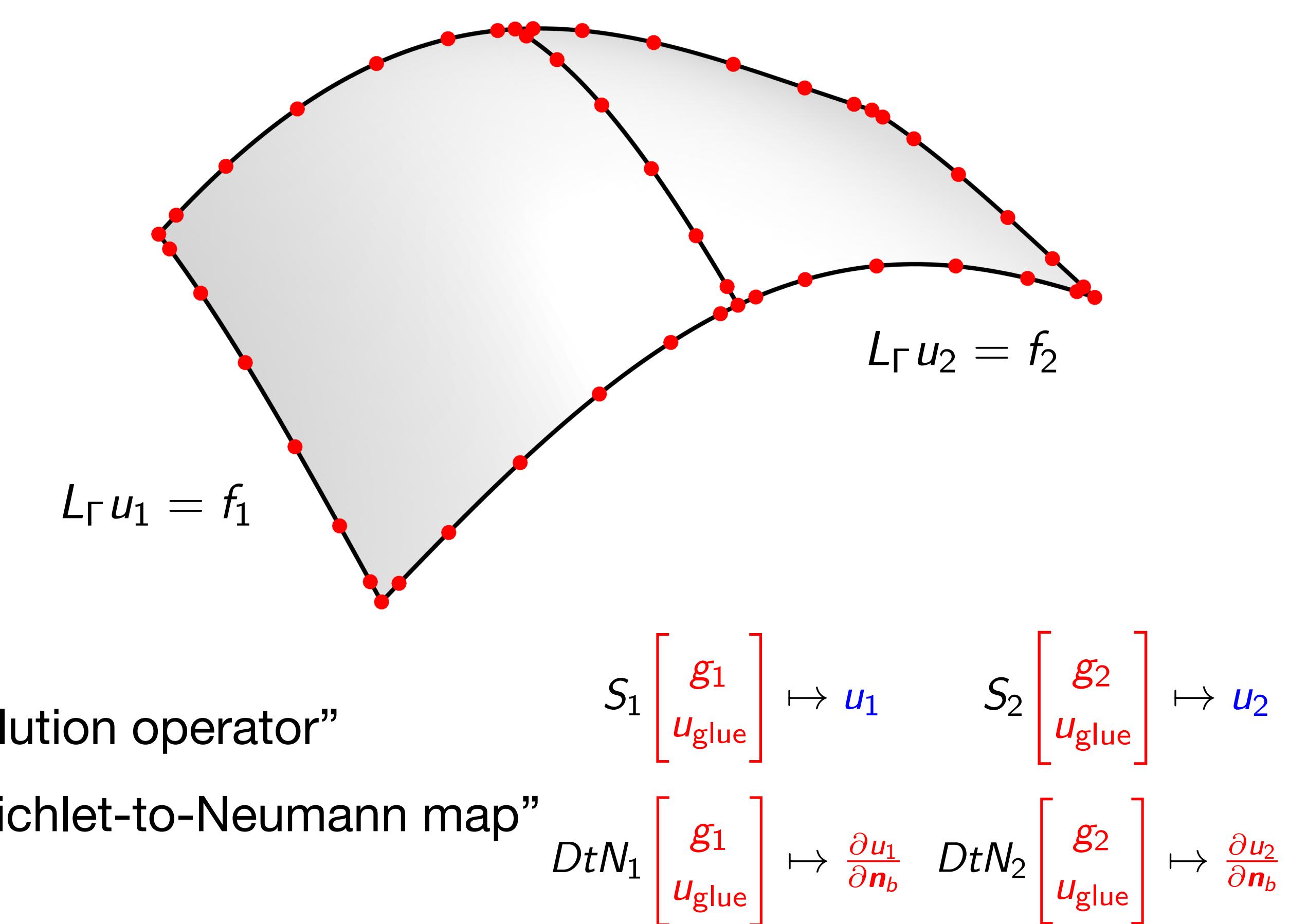
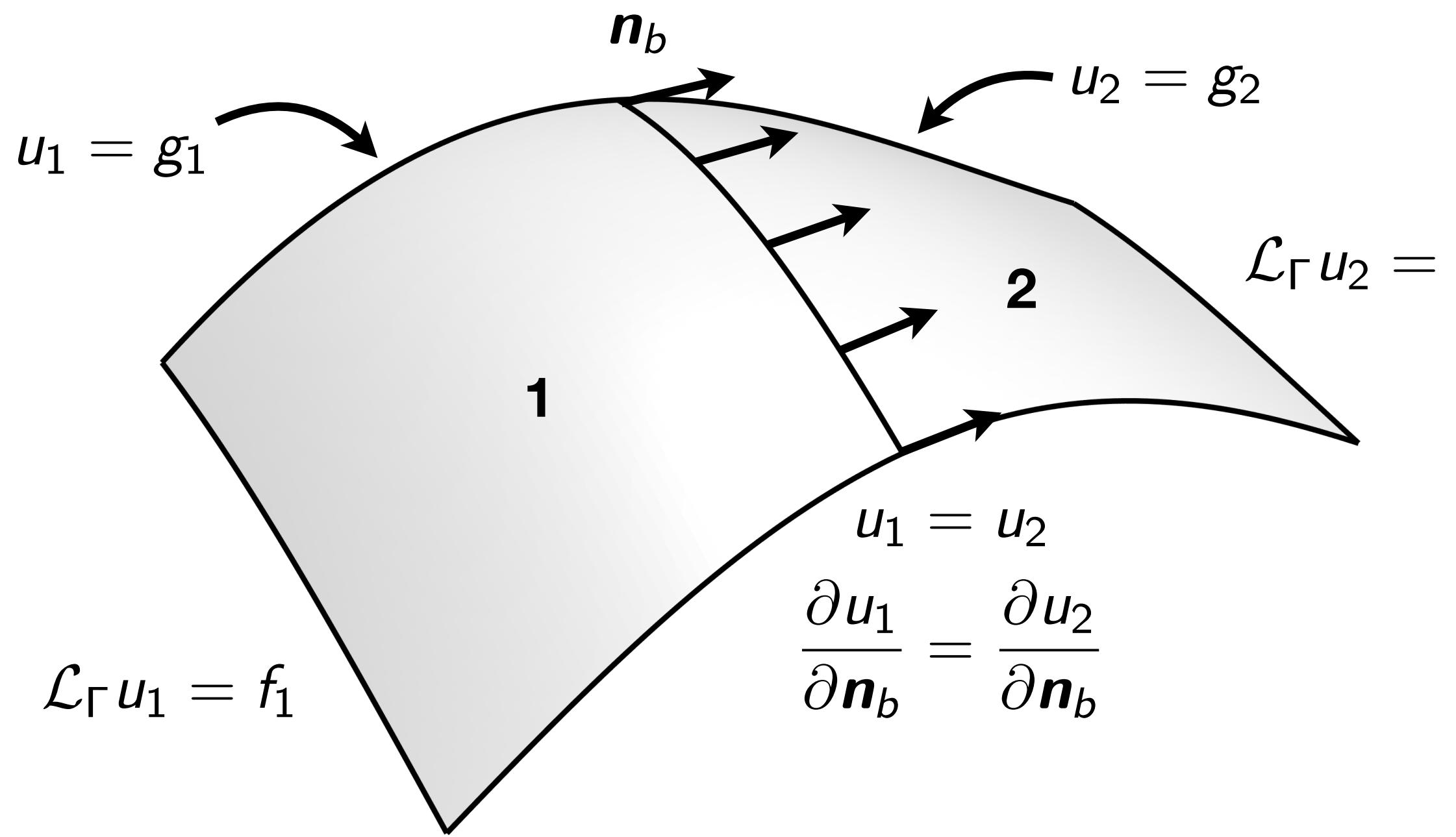


- Know how to do **local** solves on each element: “Solution operator”
- Know how information **flows out** of each element: “Dirichlet-to-Neumann map”

$$S_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_1 \quad S_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto u_2$$
$$DtN_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial \mathbf{n}_b} \quad DtN_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial \mathbf{n}_b}$$

High-order discretization

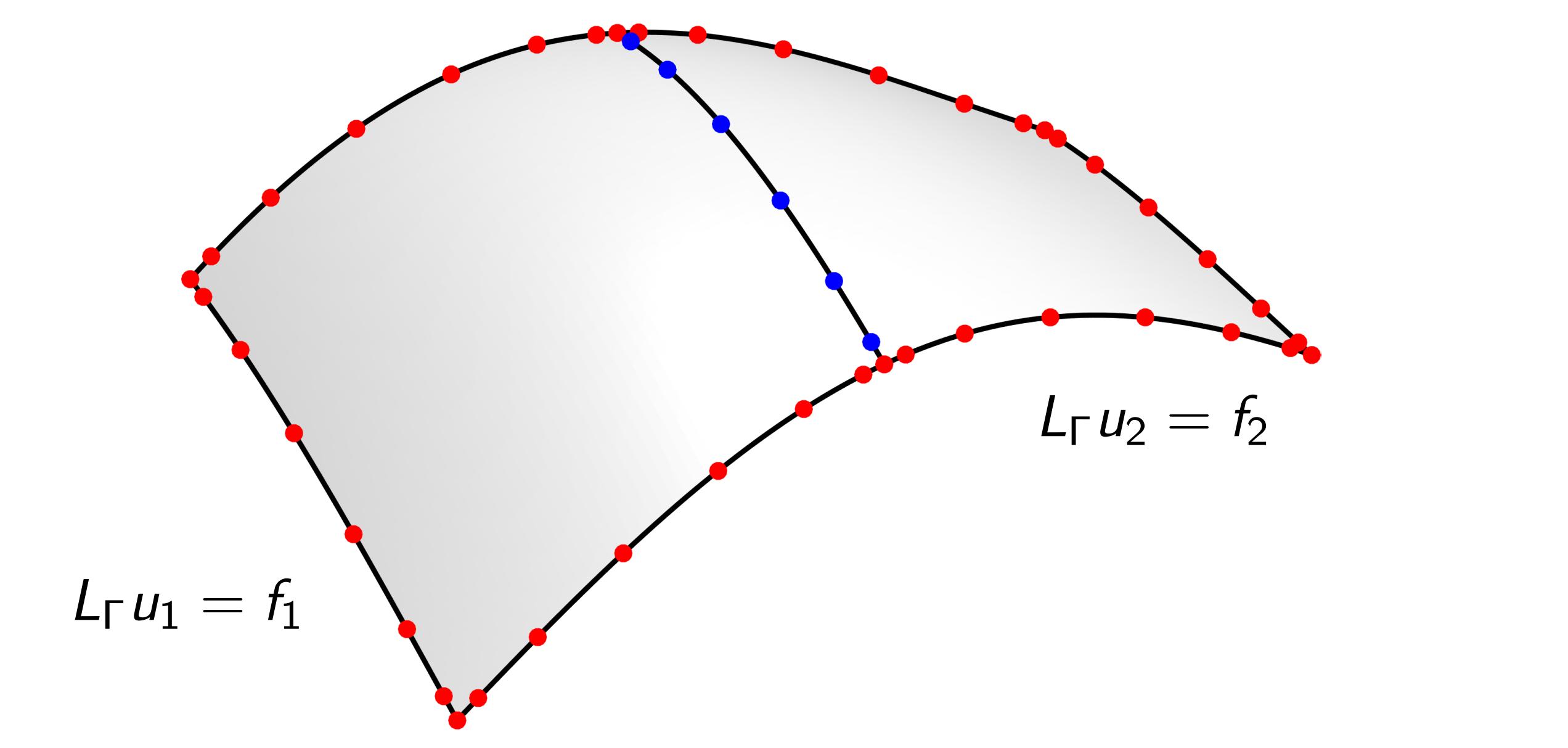
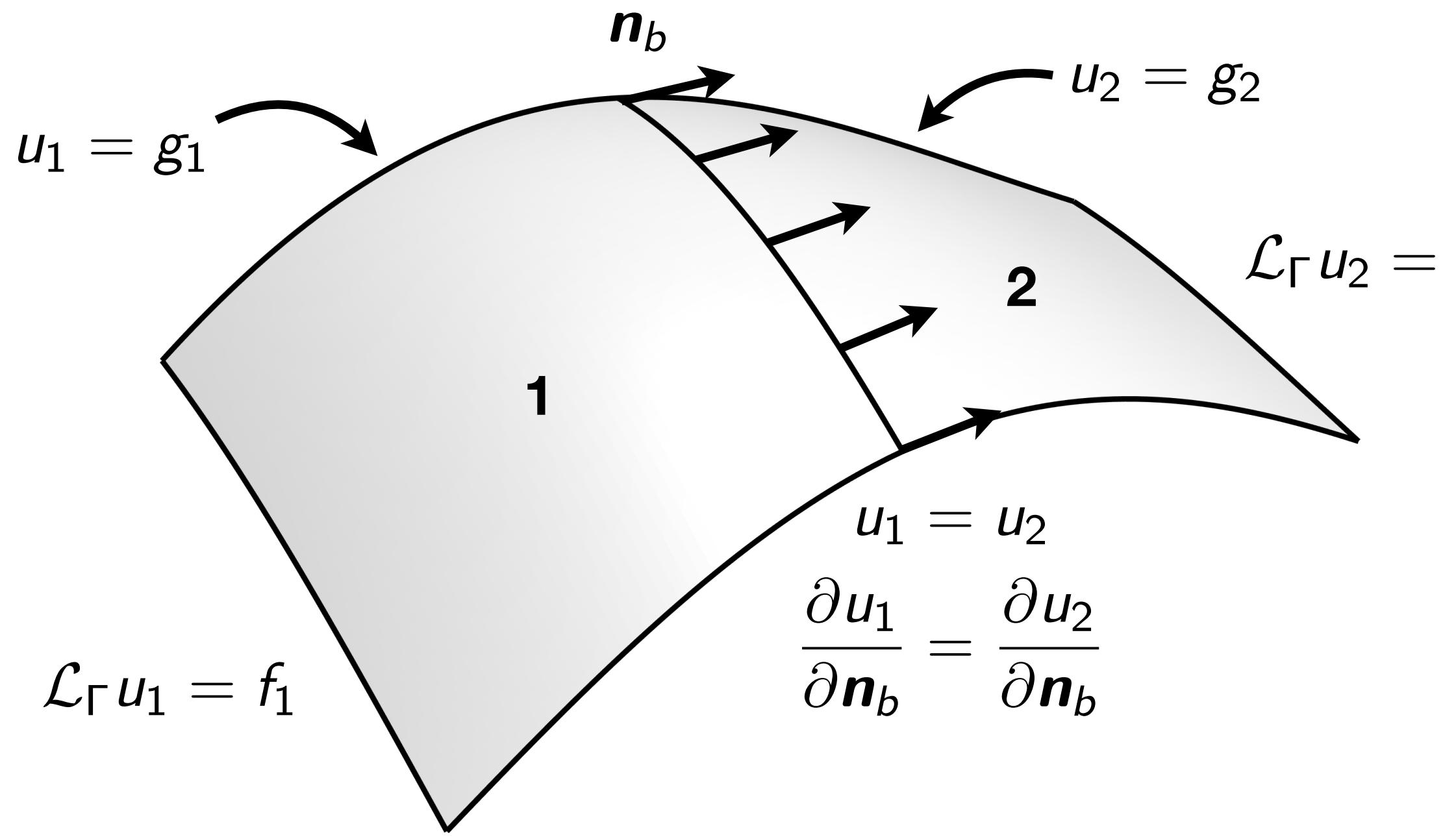
Two glued patches



[F., 2022]

High-order discretization

Two glued patches



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

$$S_{\text{glue}} = - \left(D t N_1^{\text{glue}} + D t N_2^{\text{glue}} \right)^{-1} \begin{bmatrix} D t N_1^{\text{glue},1} & D t N_2^{\text{glue},2} \end{bmatrix}$$

$$S_{\text{glue}} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = u_{\text{glue}}$$

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

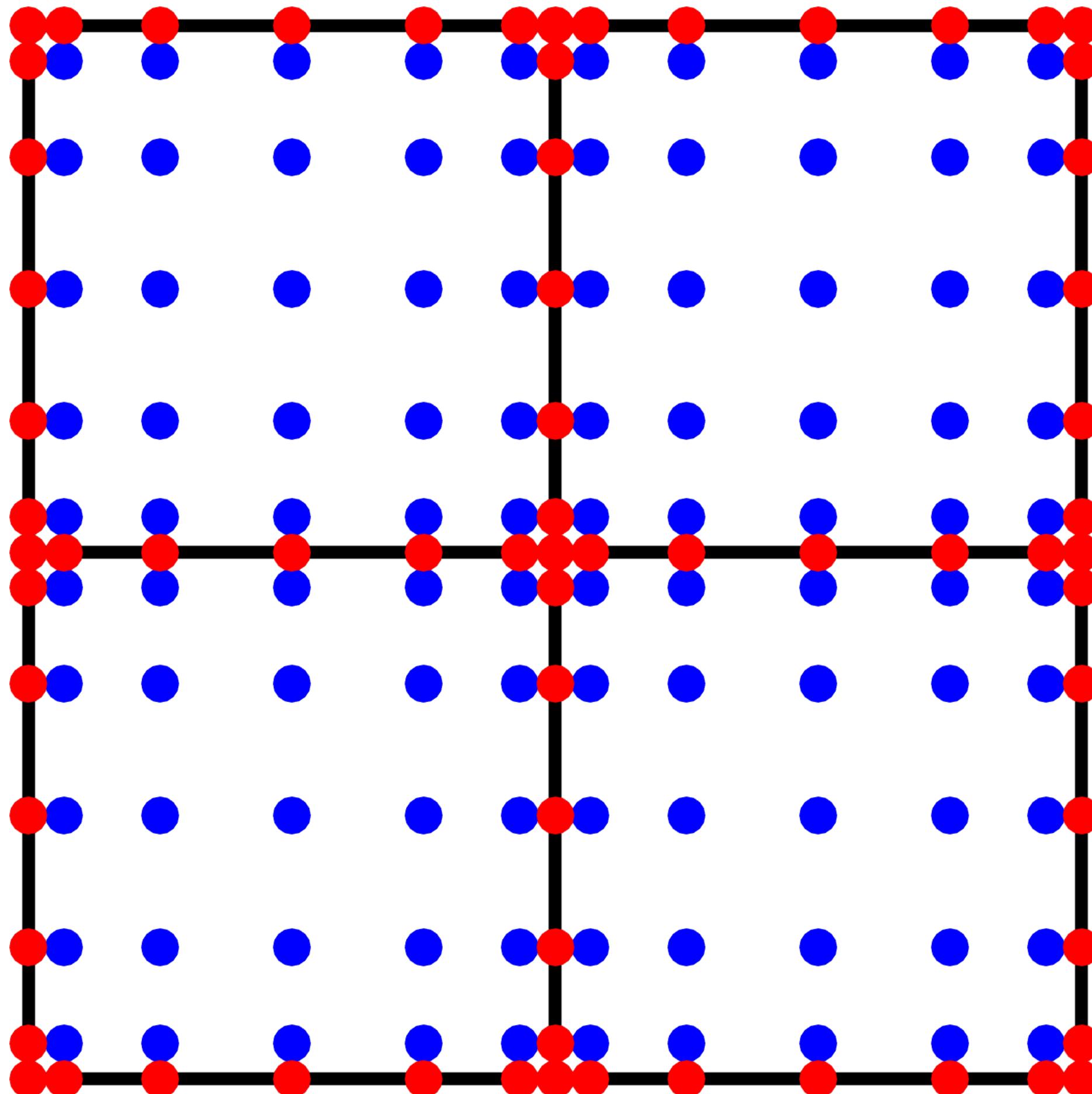
$$D t N_1 \begin{bmatrix} g_1 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_1}{\partial \mathbf{n}_b} \quad D t N_2 \begin{bmatrix} g_2 \\ u_{\text{glue}} \end{bmatrix} \mapsto \frac{\partial u_2}{\partial \mathbf{n}_b}$$

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

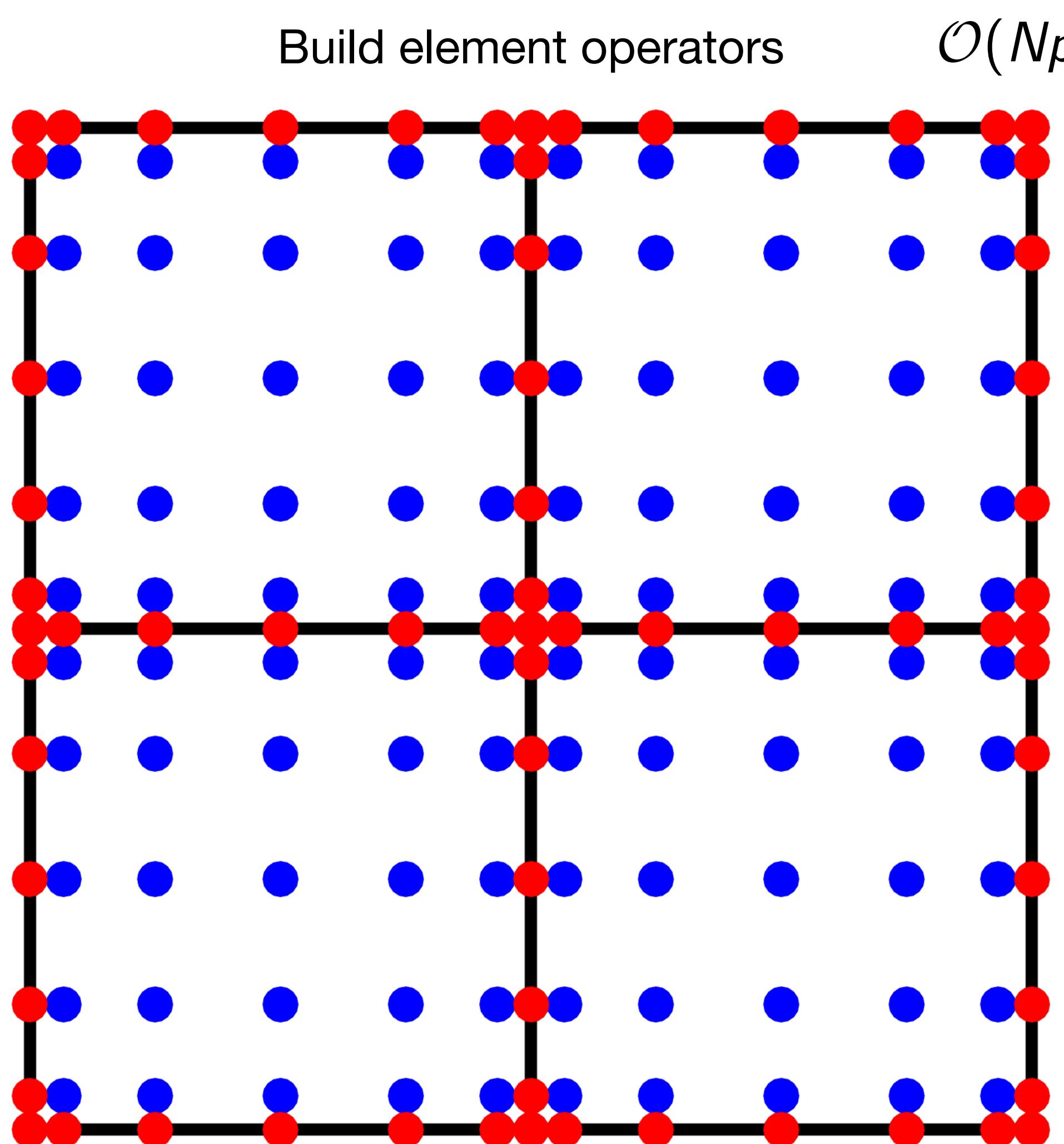
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

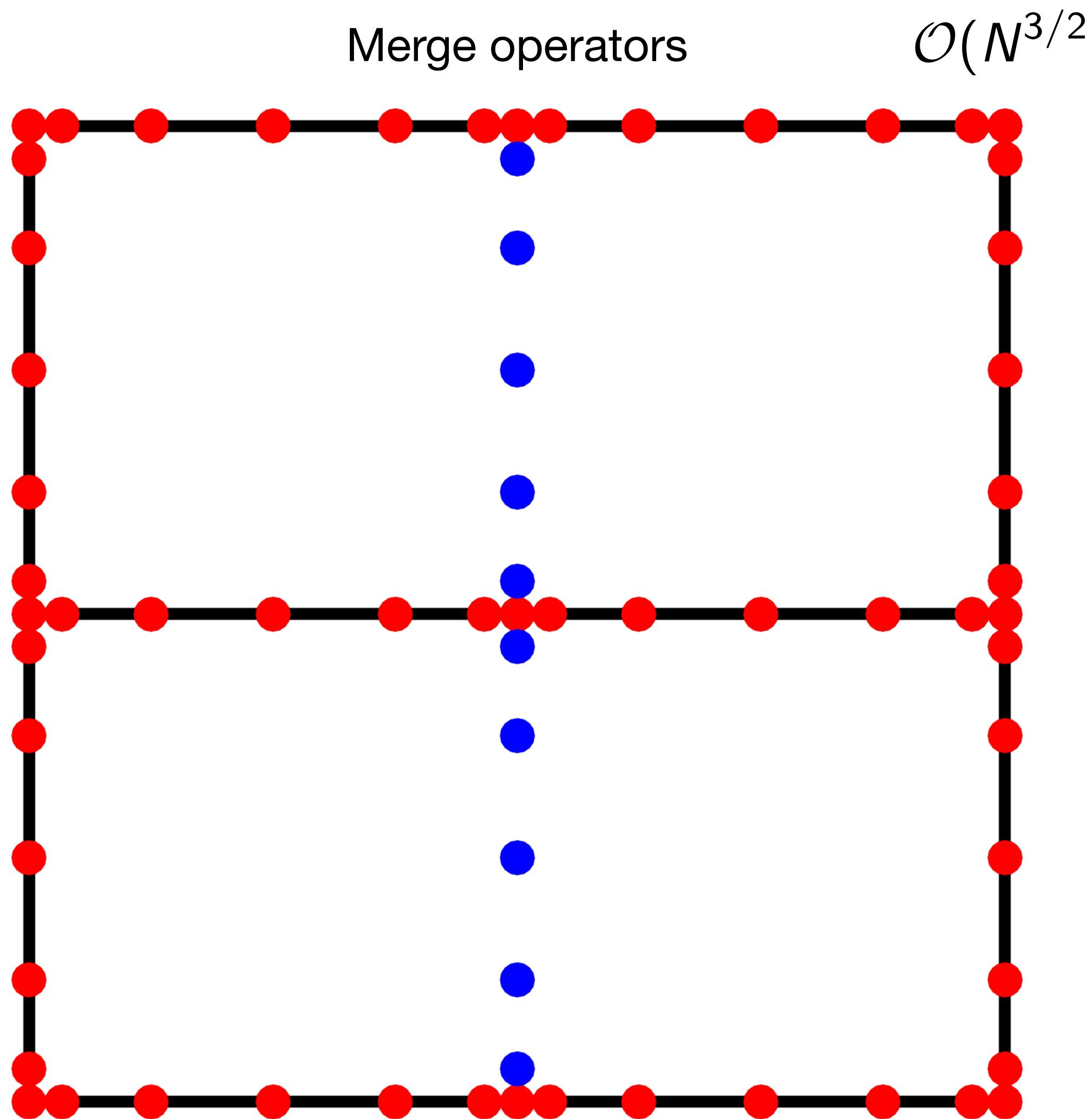
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

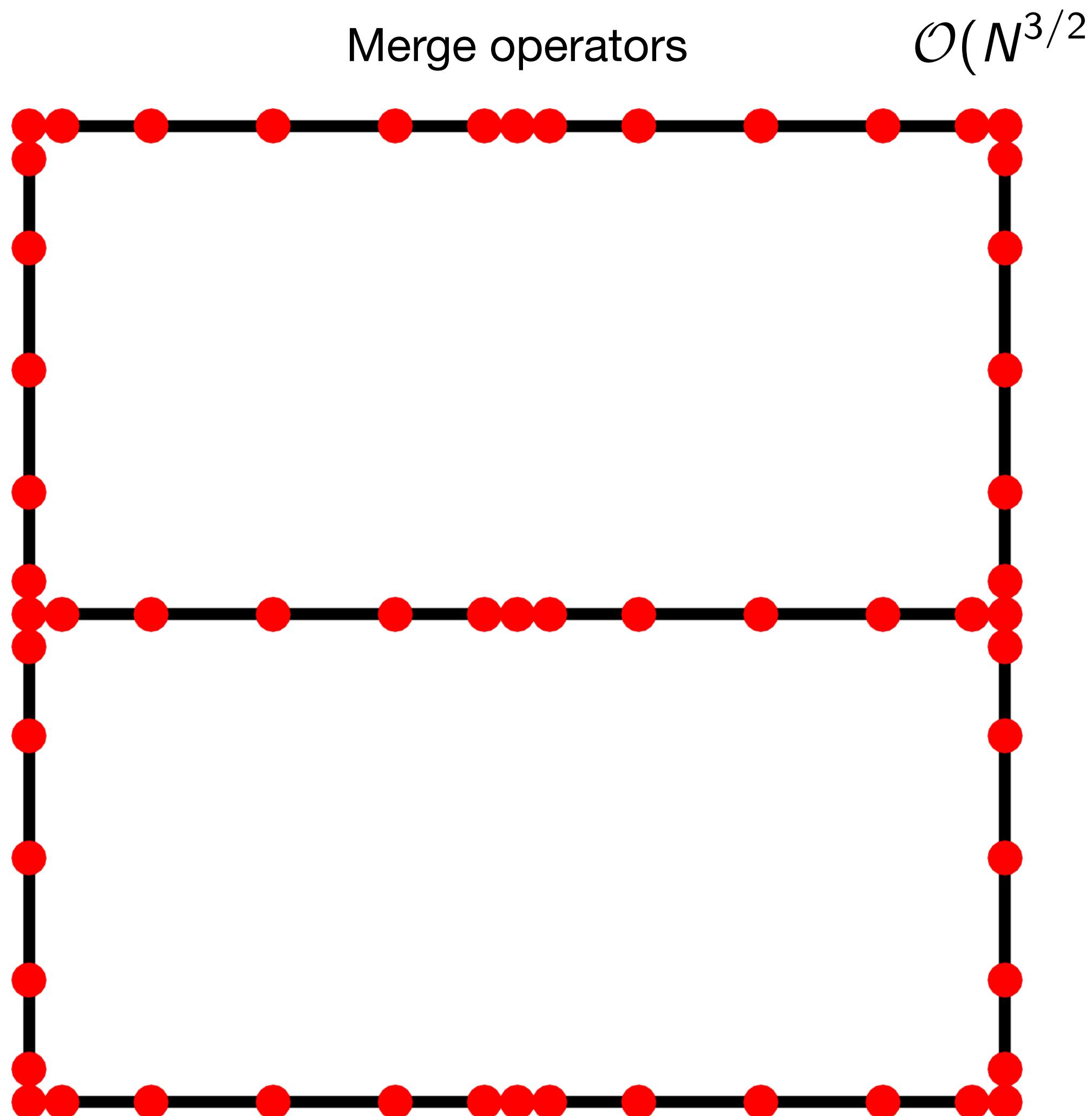
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

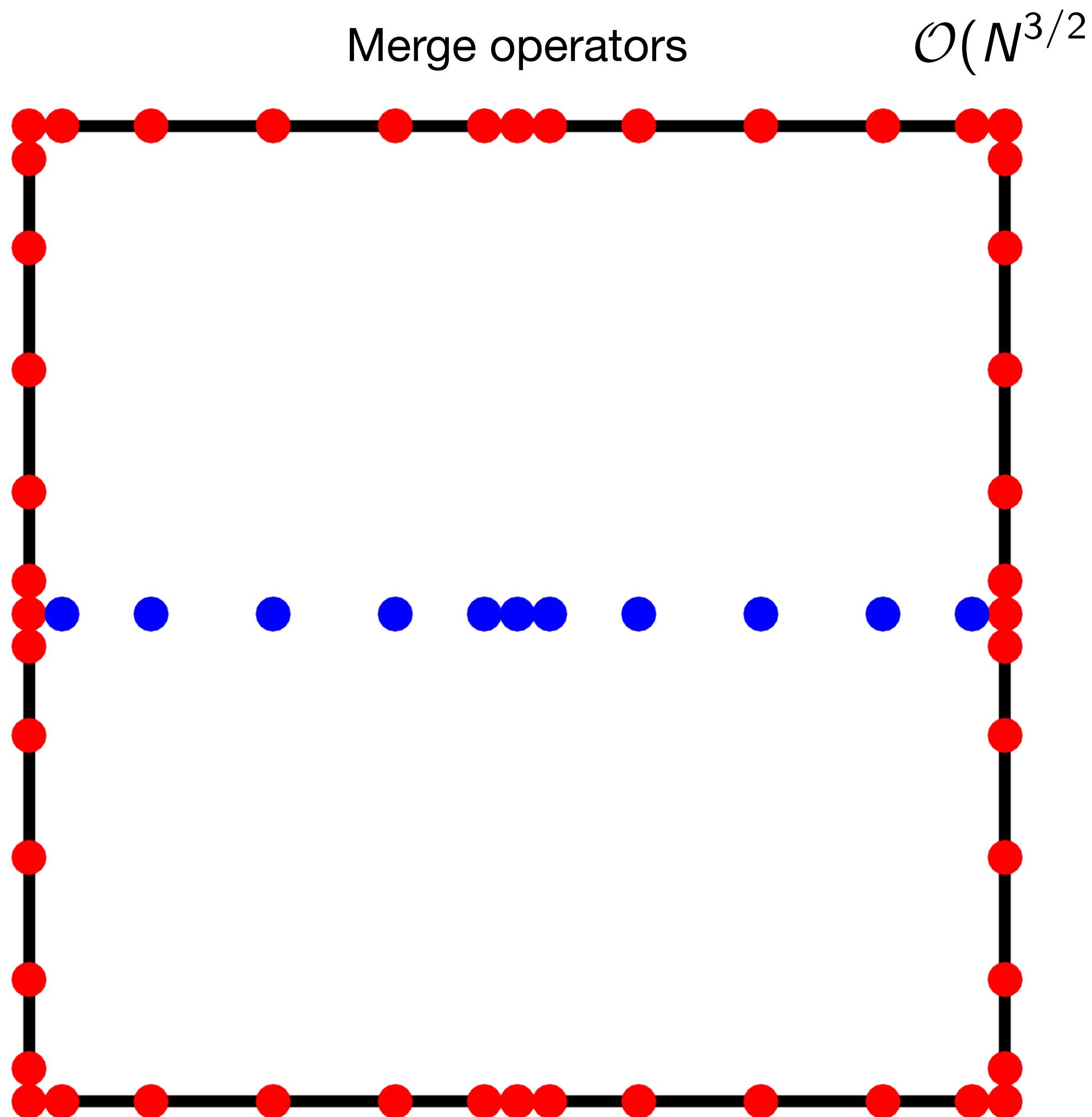
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

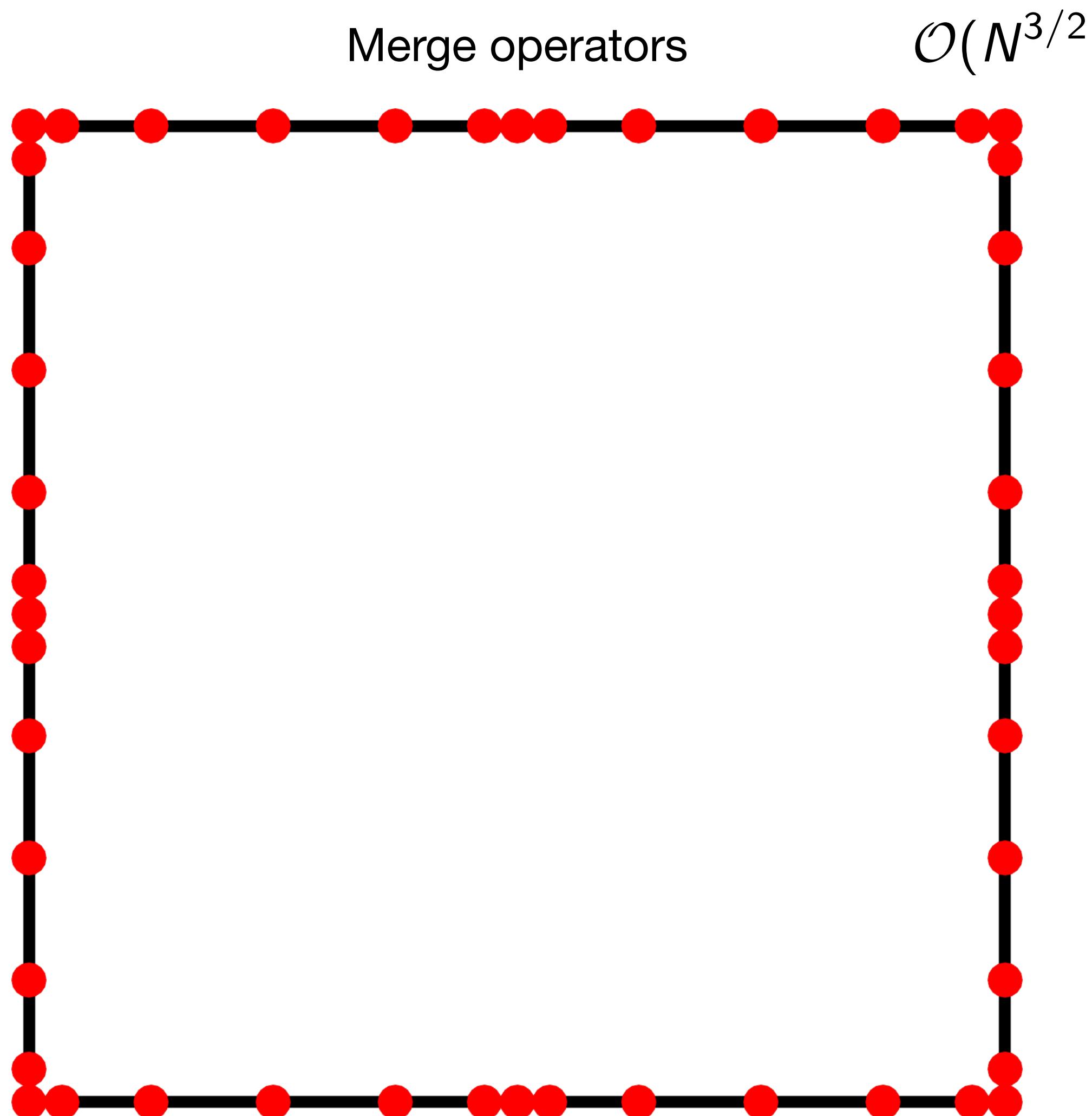
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

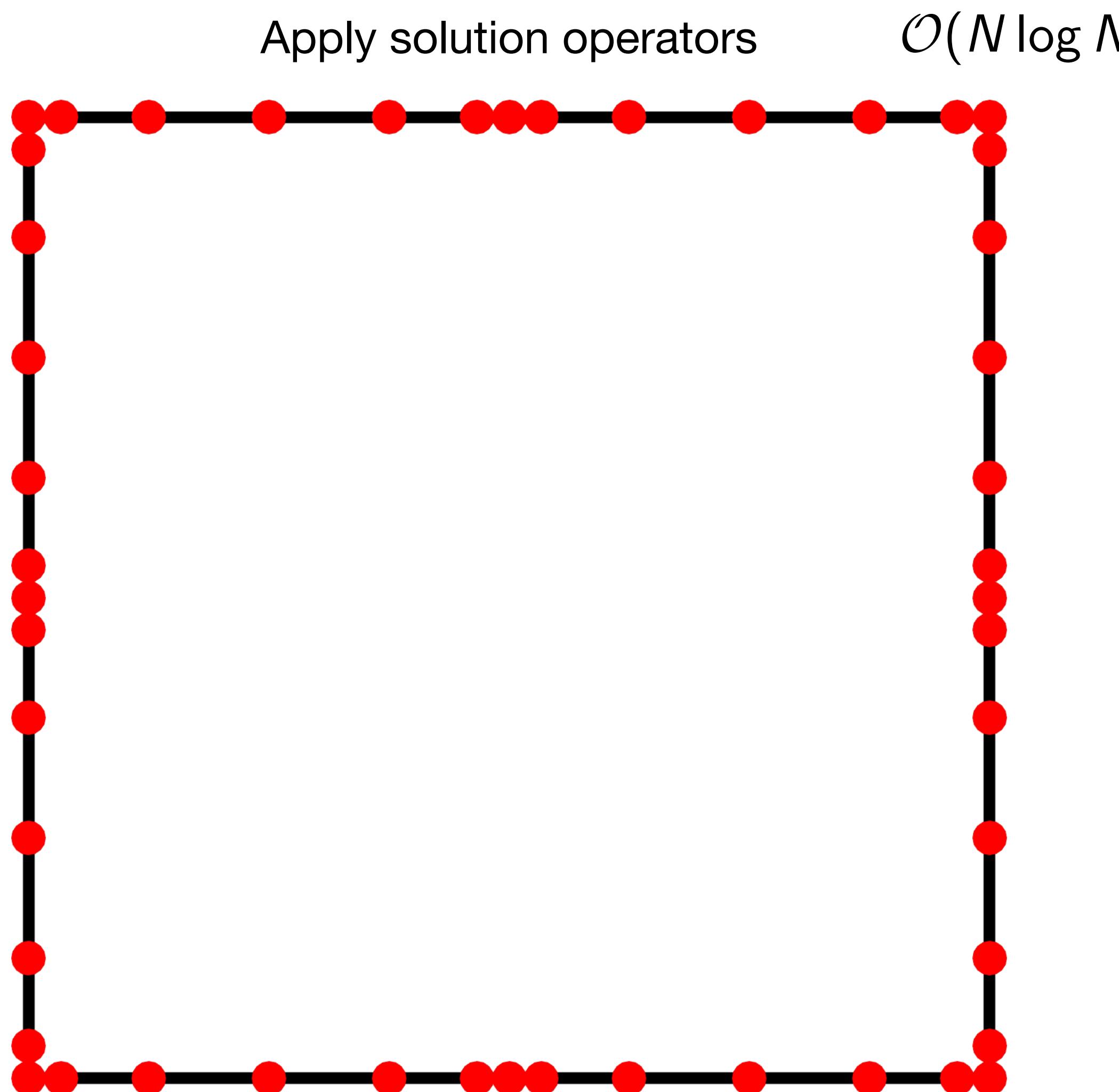
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

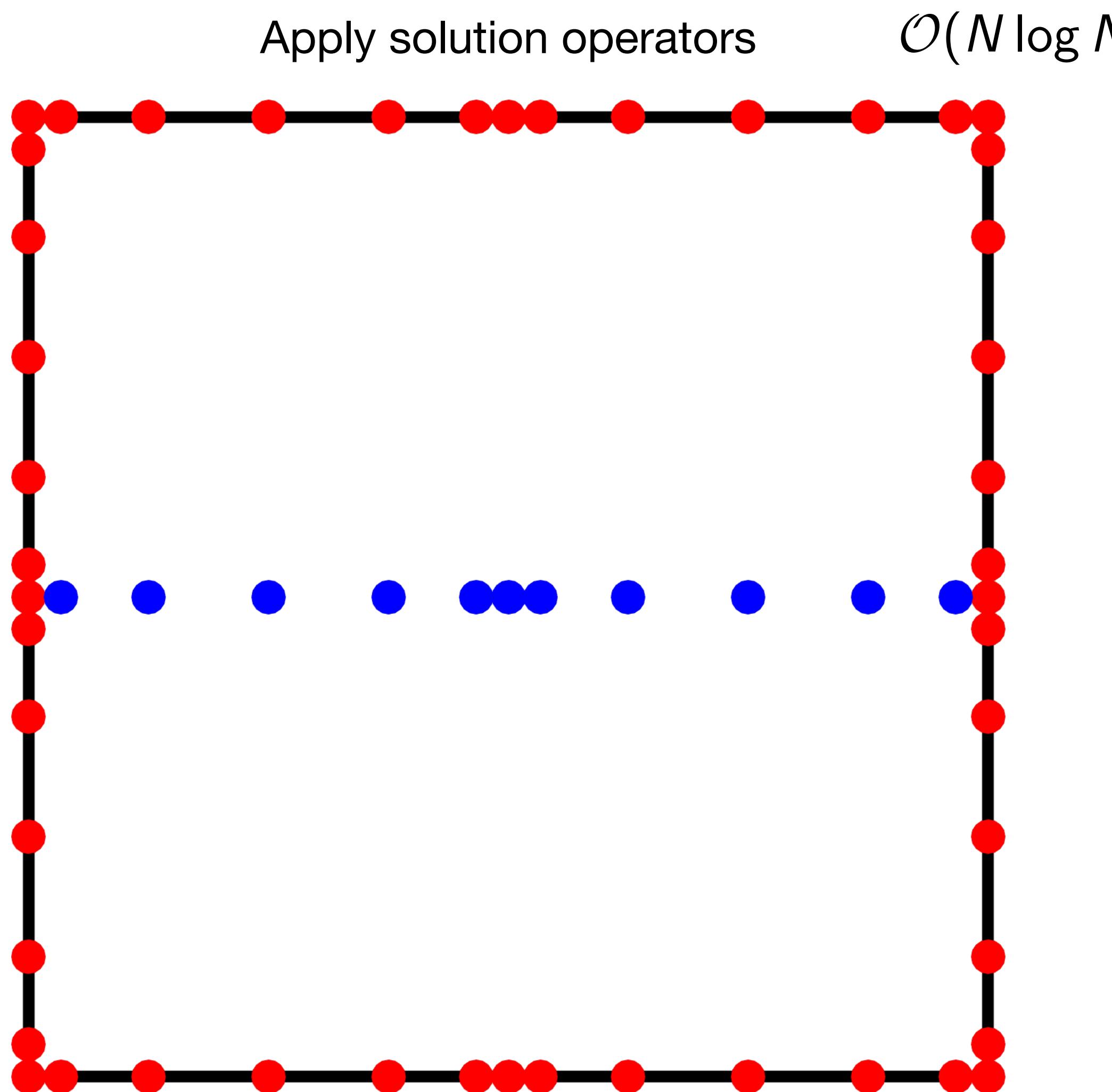
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

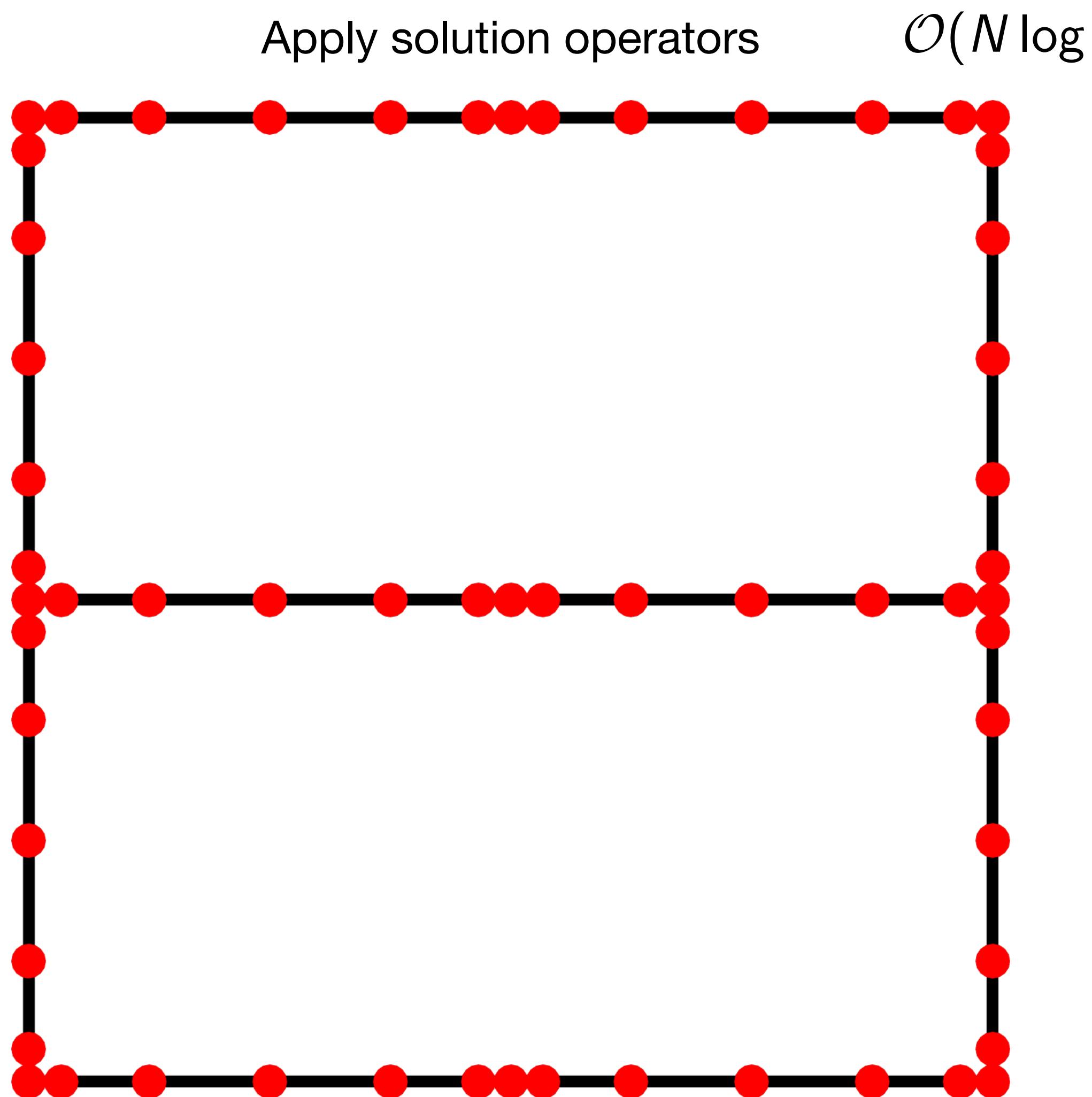
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

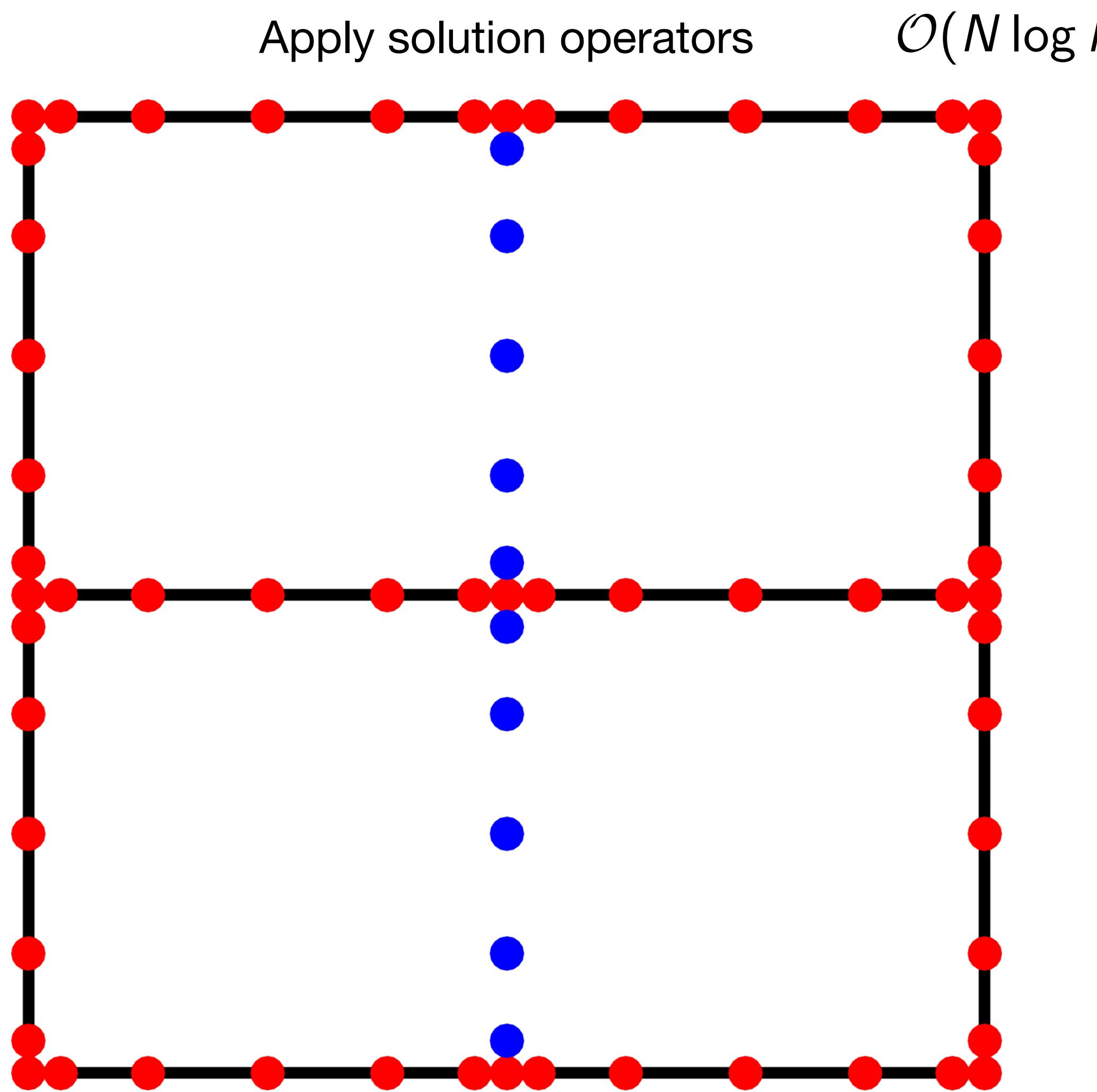
Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)



A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

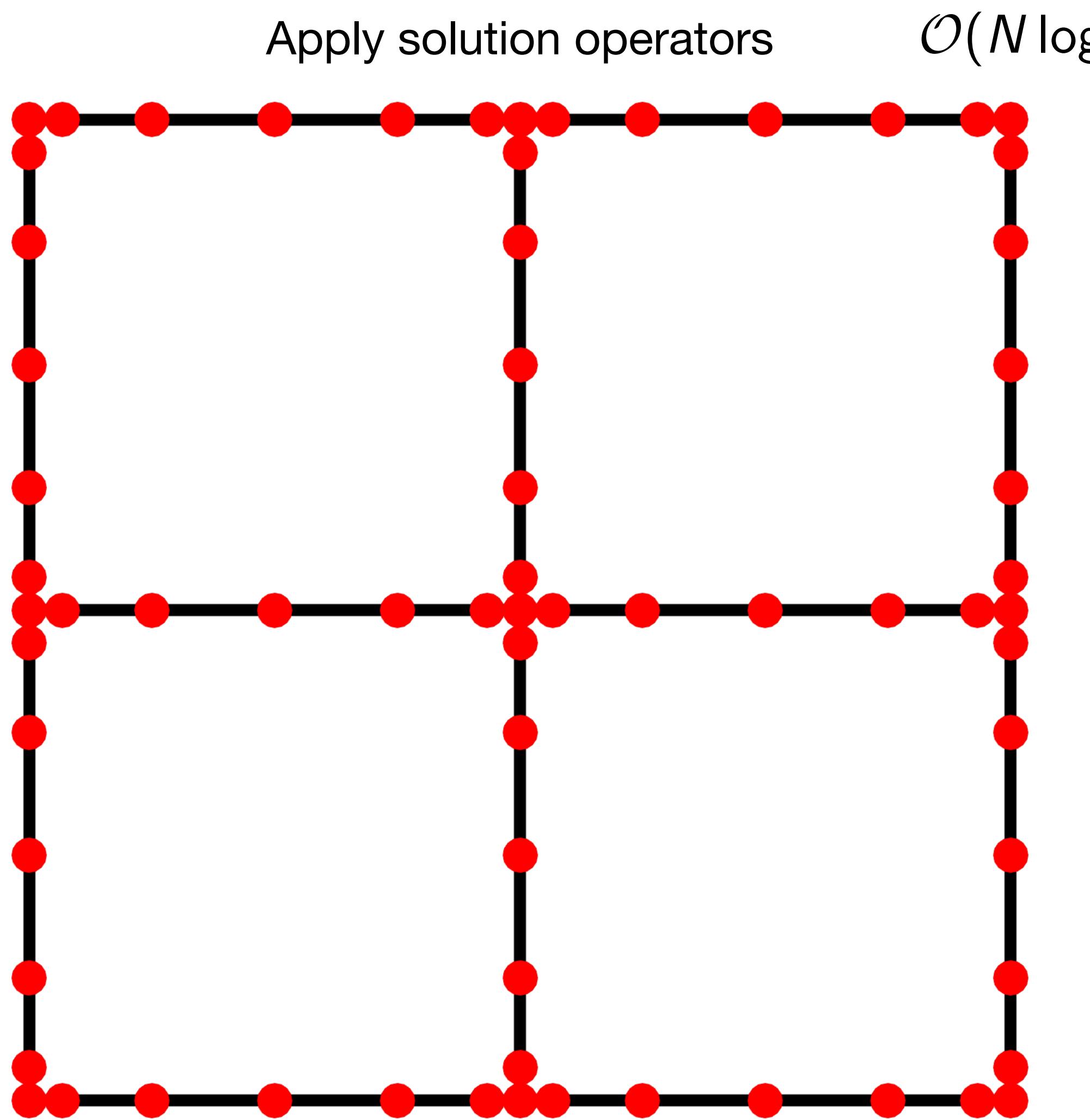
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

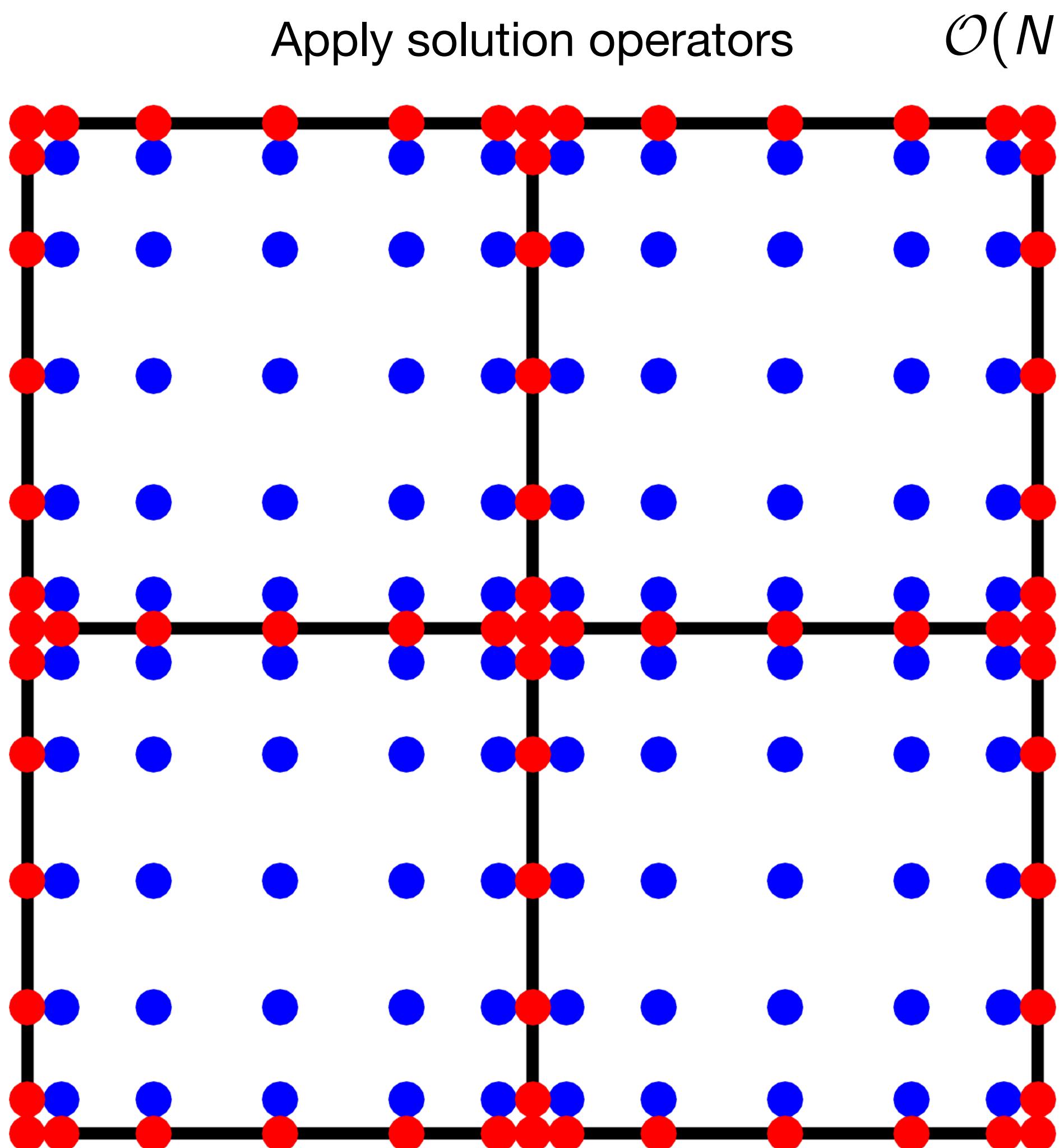
Key idea: Recursively glue elements together in a hierarchy (cf. nested dissection)



A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



[Martinsson, 2013]

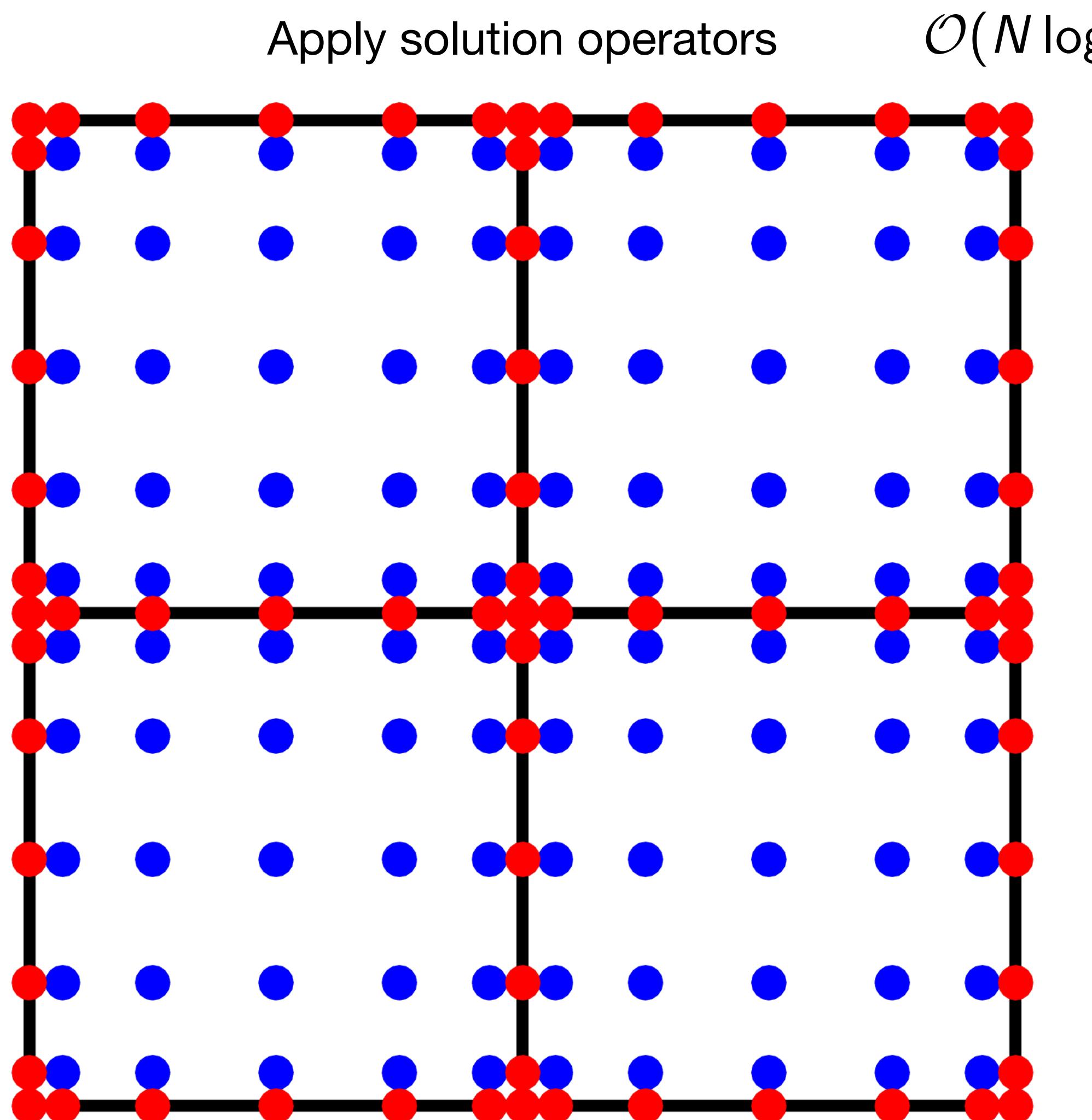
[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

Hierarchical Poincaré–Steklov method

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



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

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

[Martinsson, 2013]

[Gillman & Martinsson, 2014]

[F., 2022]

A fast direct solver on surfaces

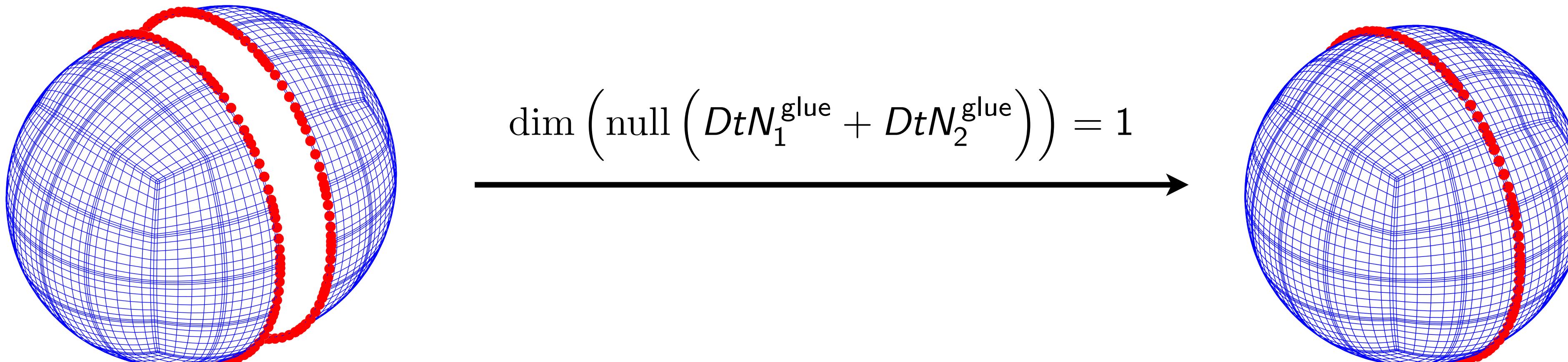
Laplace–Beltrami and rank deficiency

$$\Delta_\Gamma u = f$$

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

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

- This rank deficiency is only seen in the final gluing:



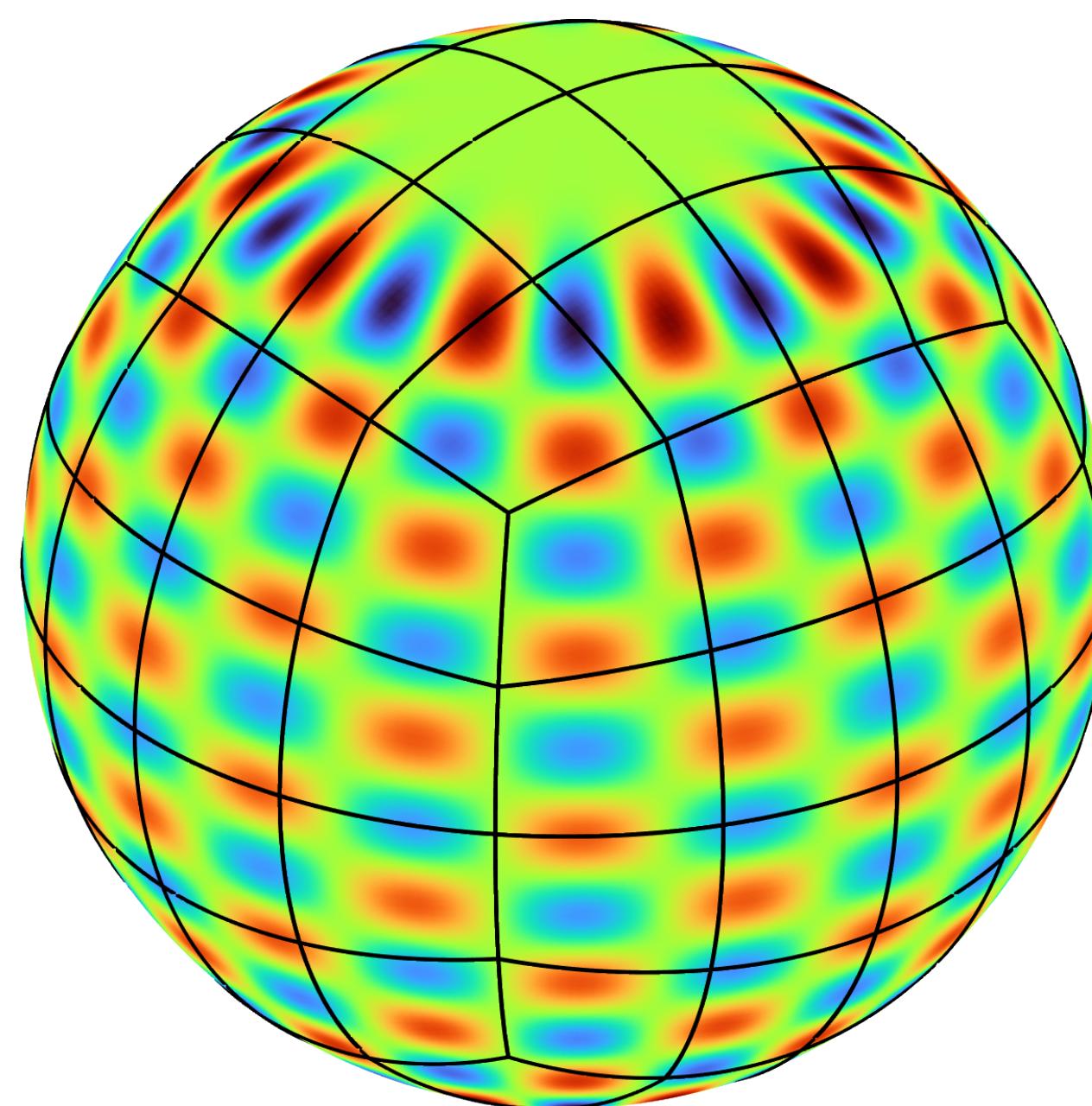
- We add the mean-zero constraint to fix the rank deficiency at the top level:

$$\dim \left(\text{null} \left(D t N_1^{\text{glue}} + D t N_2^{\text{glue}} + \mathbf{q} \mathbf{q}^\top \right) \right) = 0$$

Examples

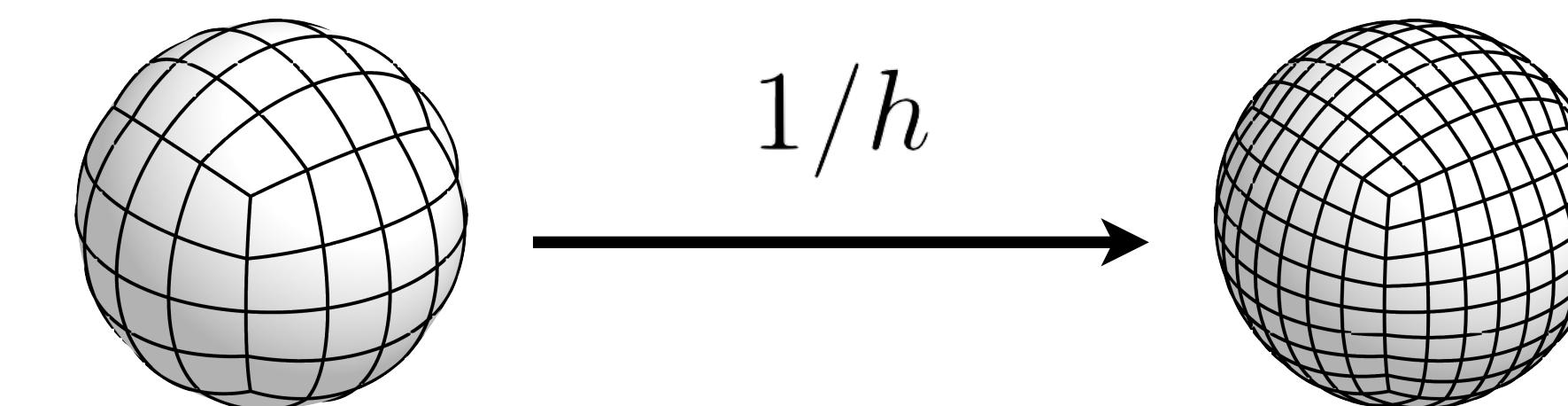
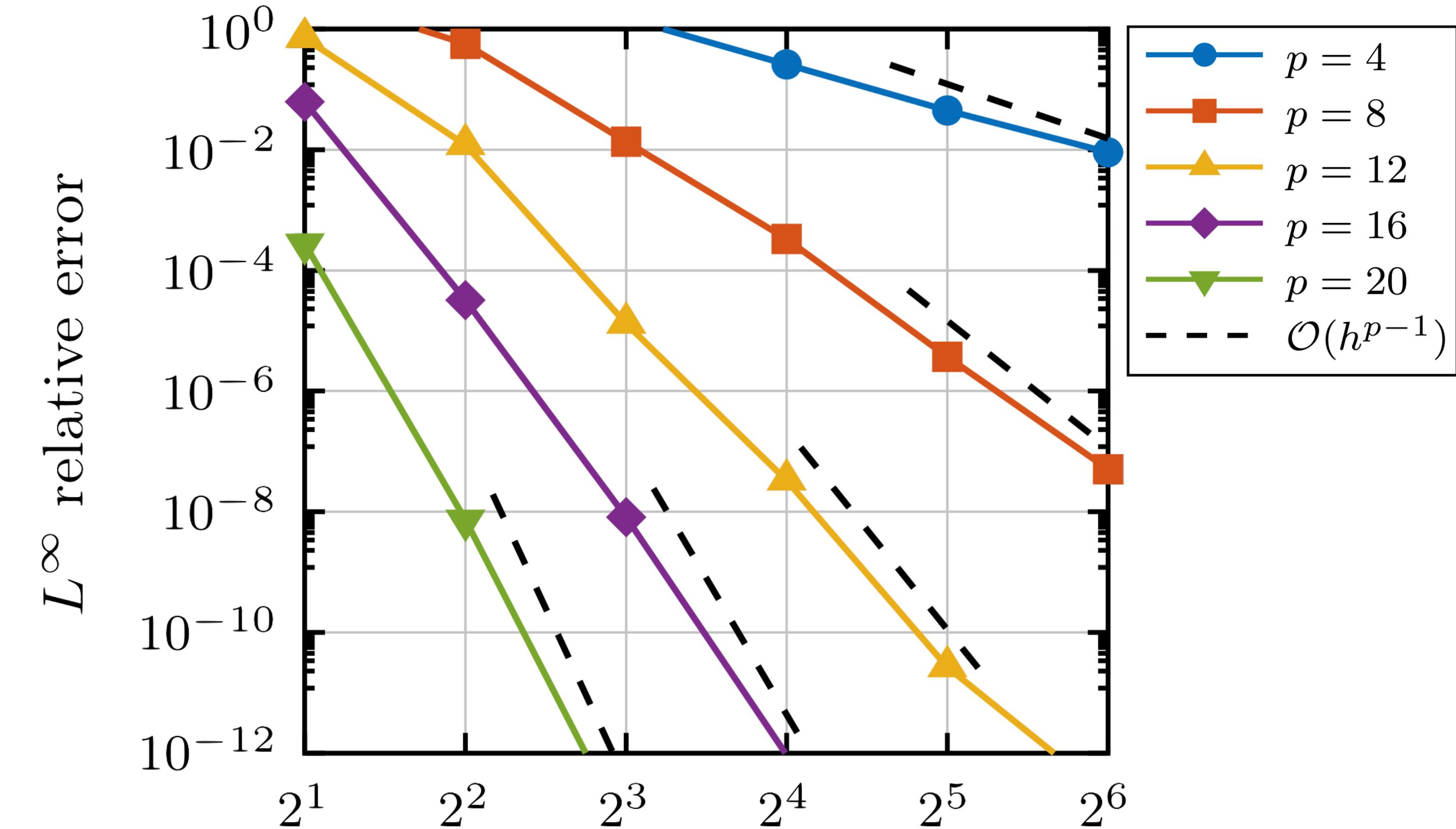
Laplace–Beltrami and convergence

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



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

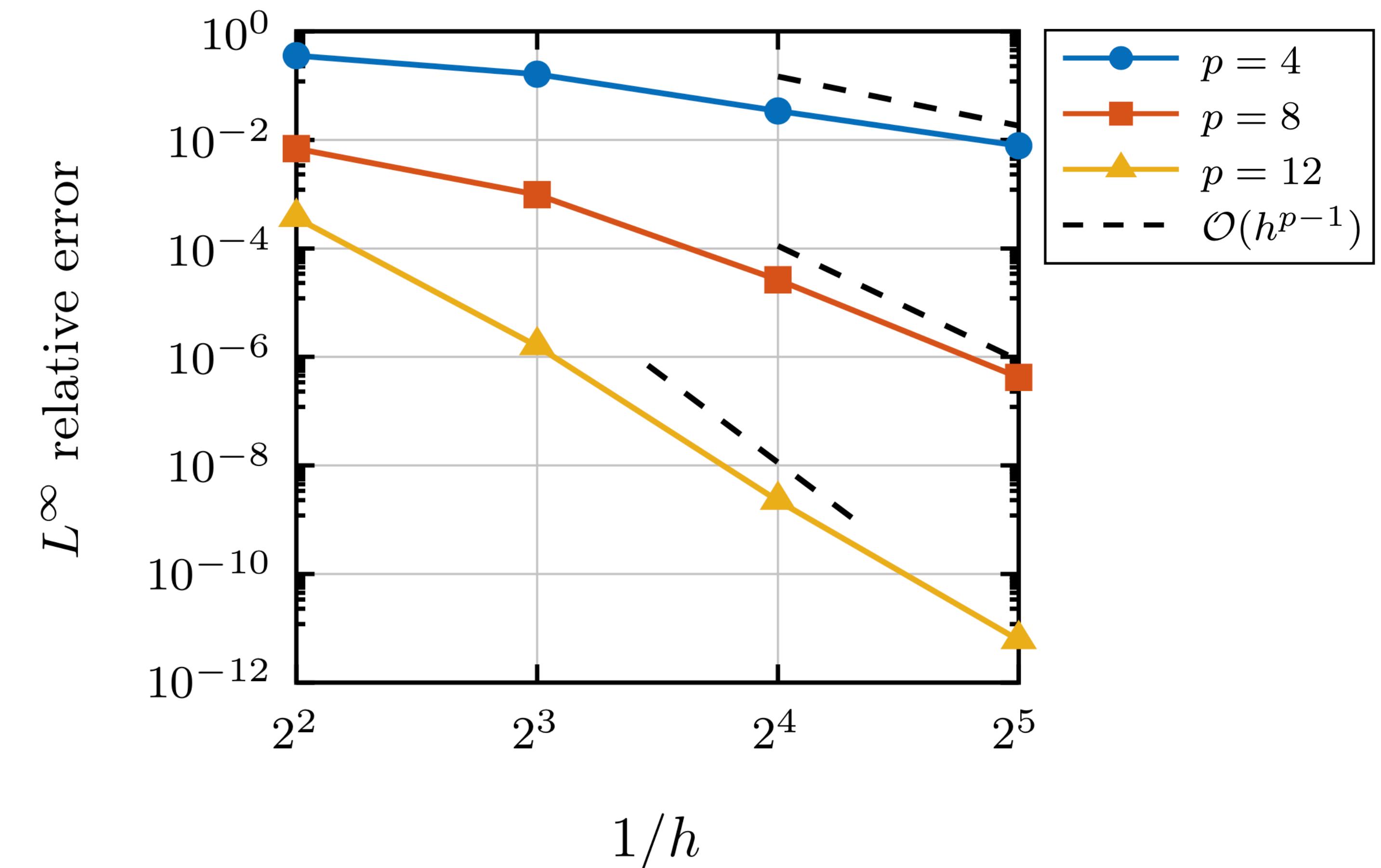
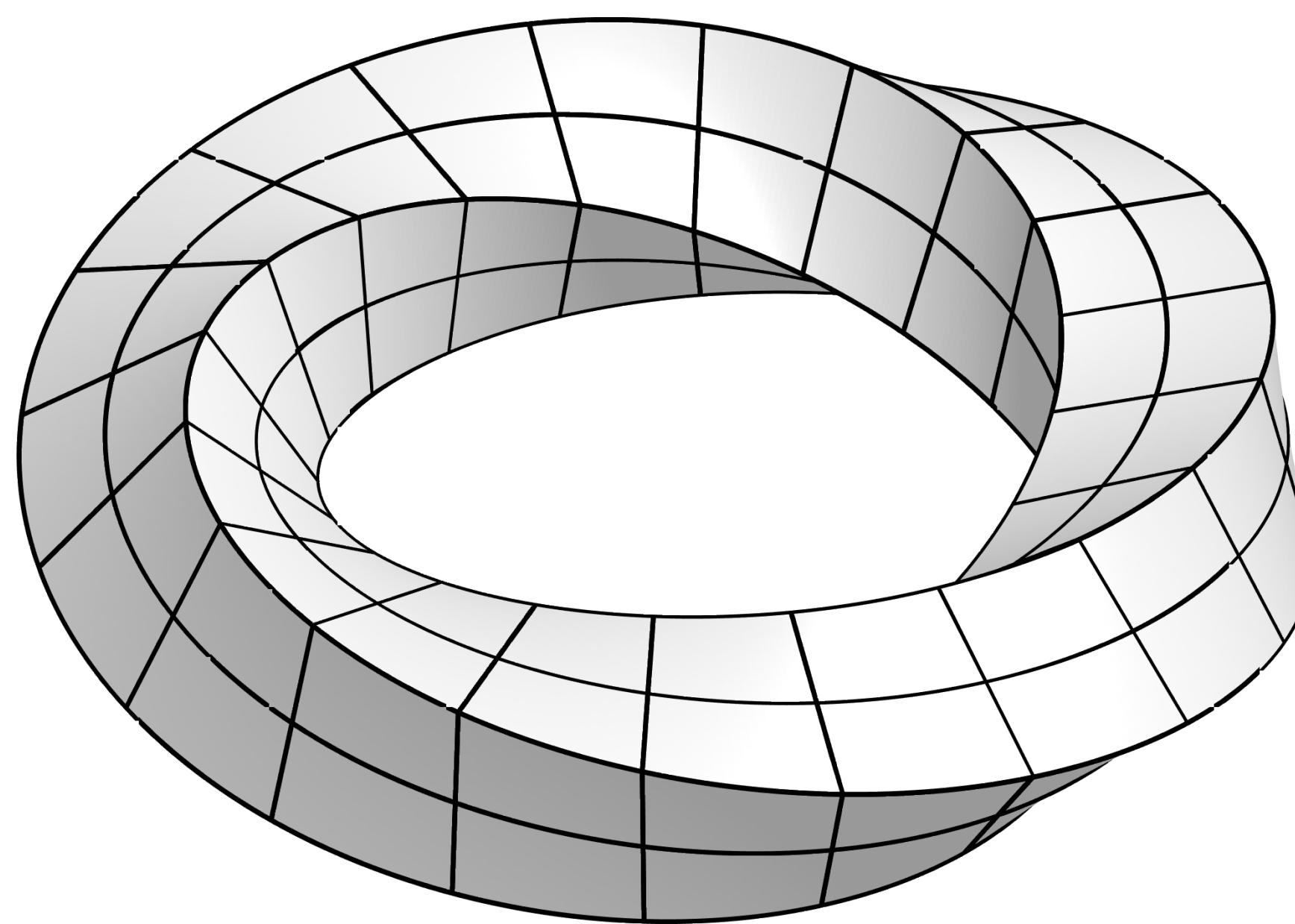
$$f(\mathbf{x}) = -\ell(\ell + 1) Y_{\ell}^m(\mathbf{x})$$



Examples

Laplace–Beltrami with edges

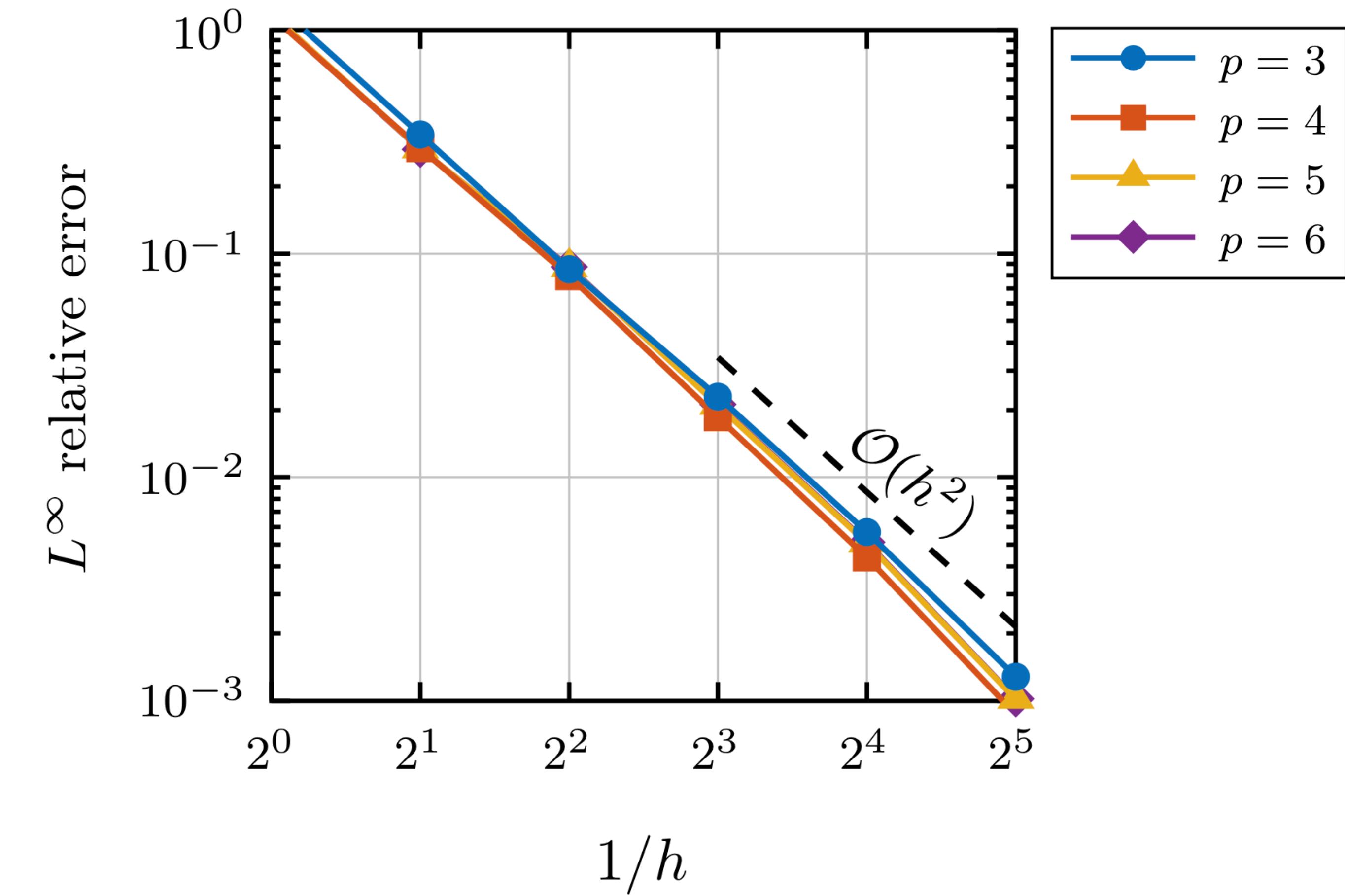
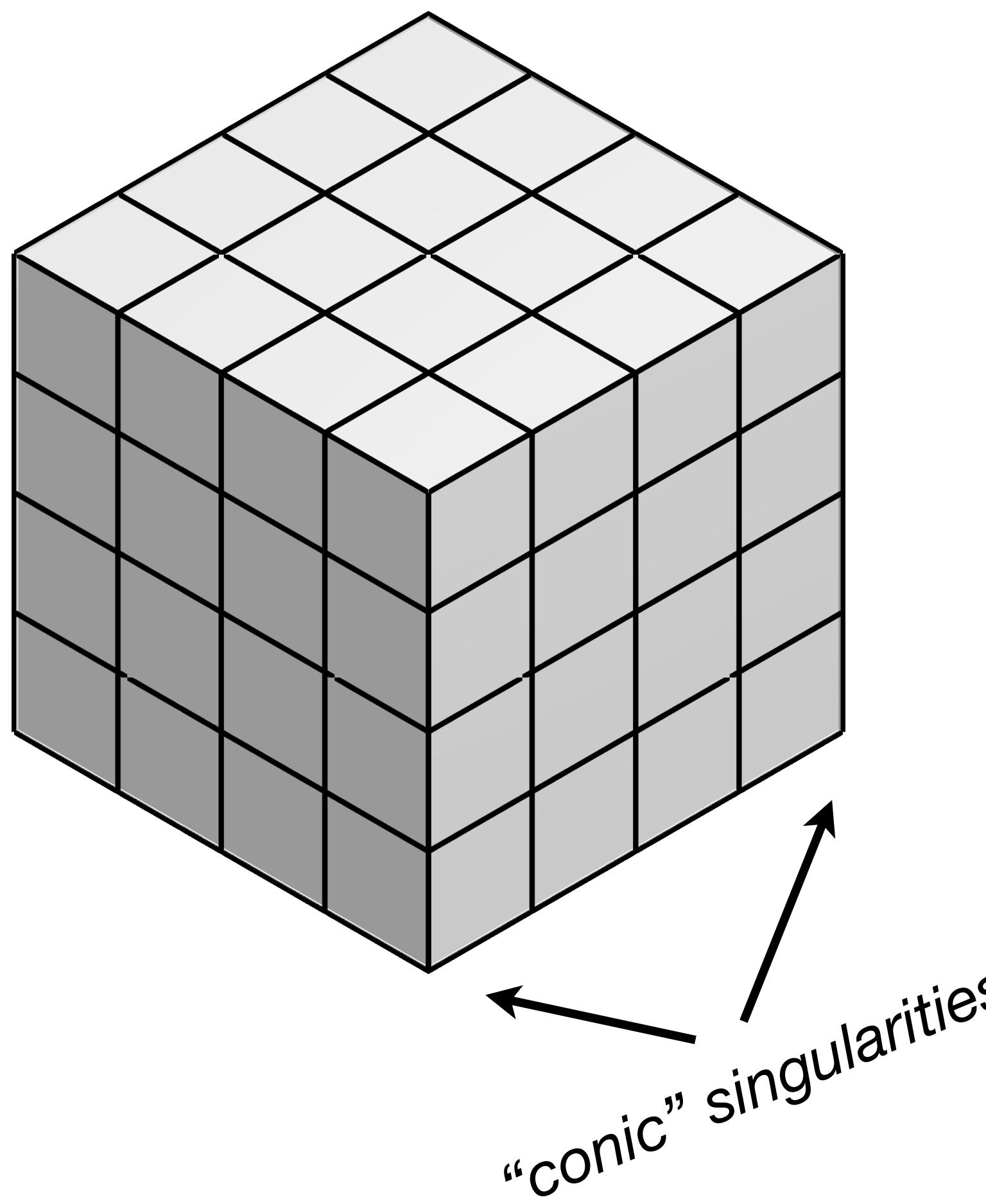
Glue conditions also allow for sharp interfaces and corners.



Examples

Laplace–Beltrami with corners

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

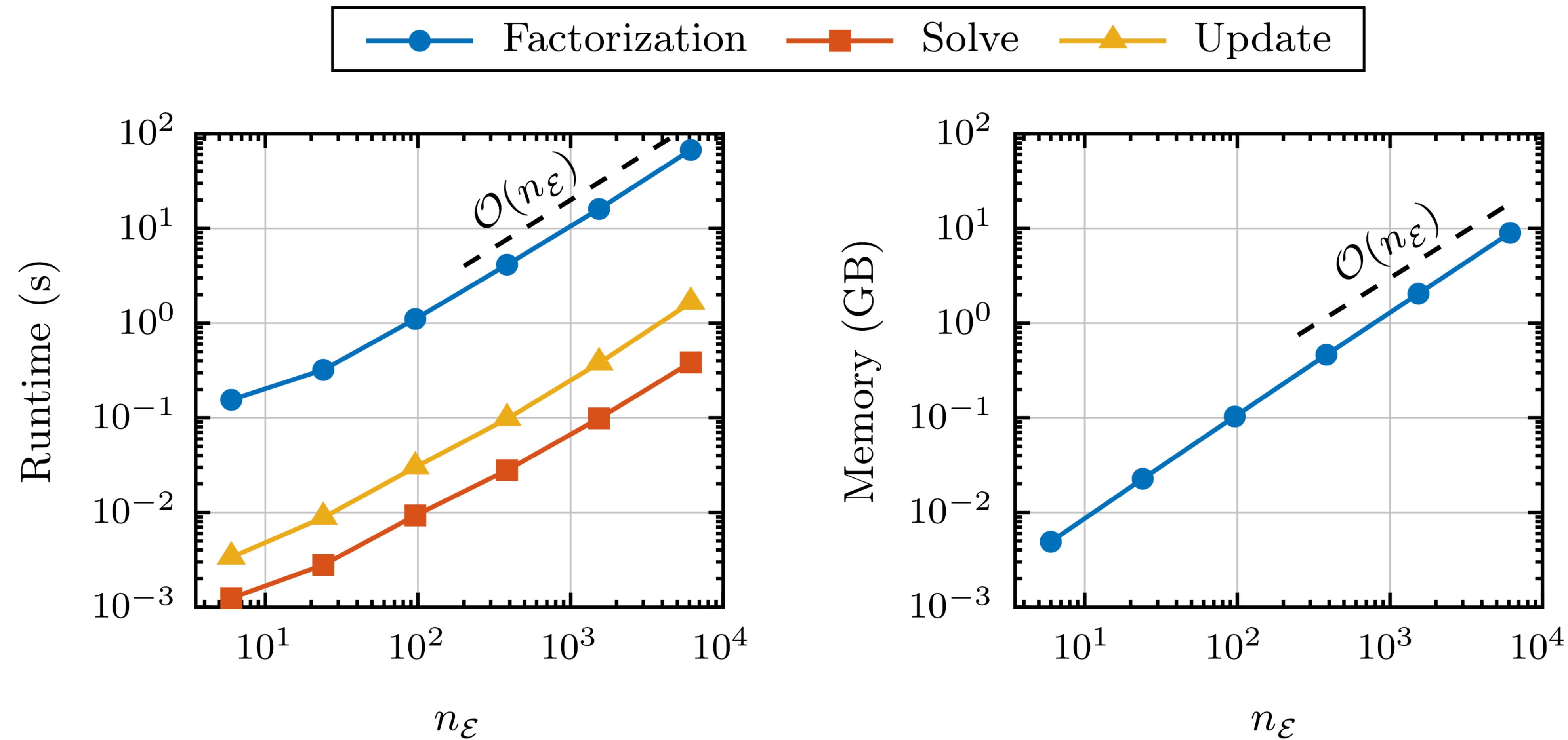


Examples

Performance

surfacefun.readthedocs.io

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

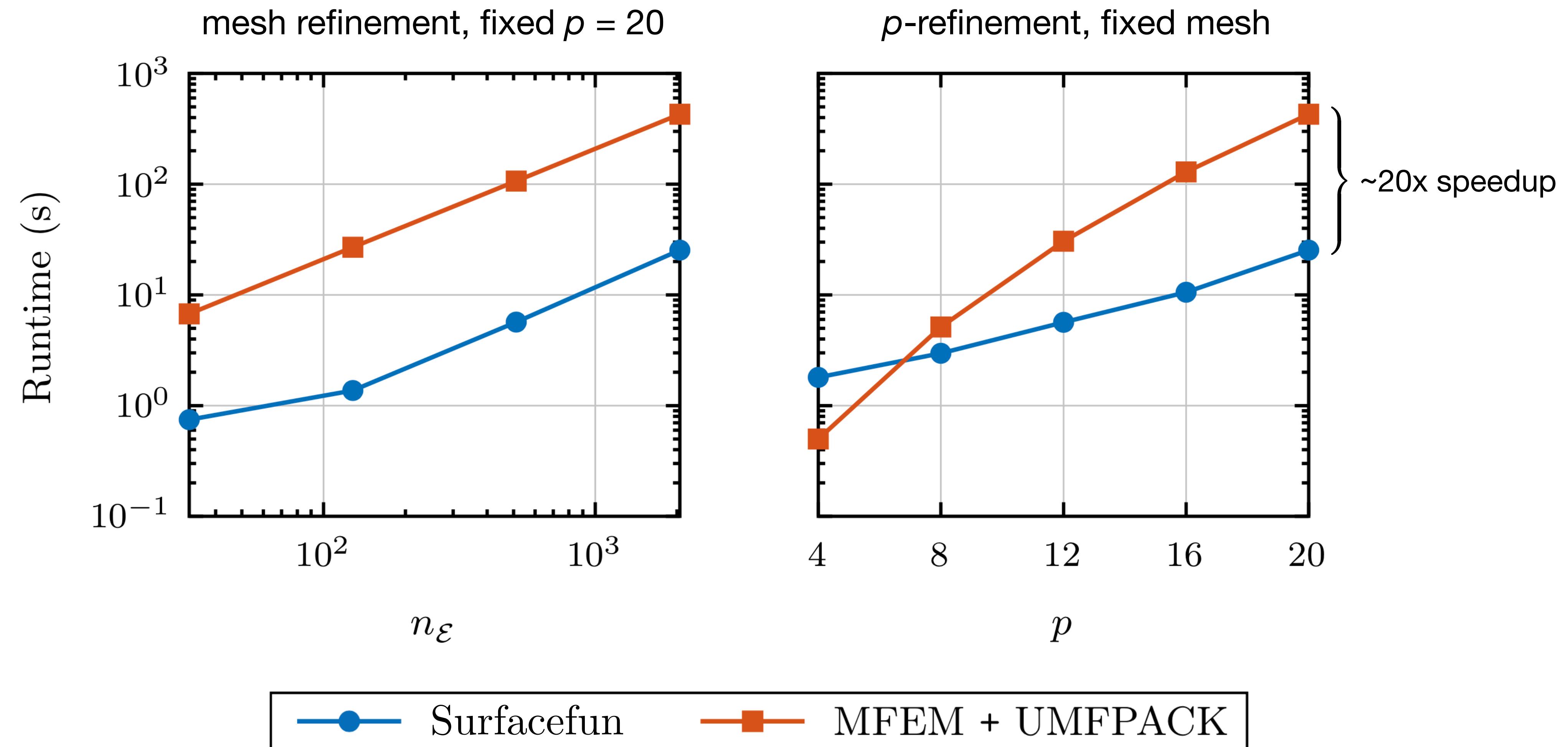


Examples

Performance

surfacefun.readthedocs.io

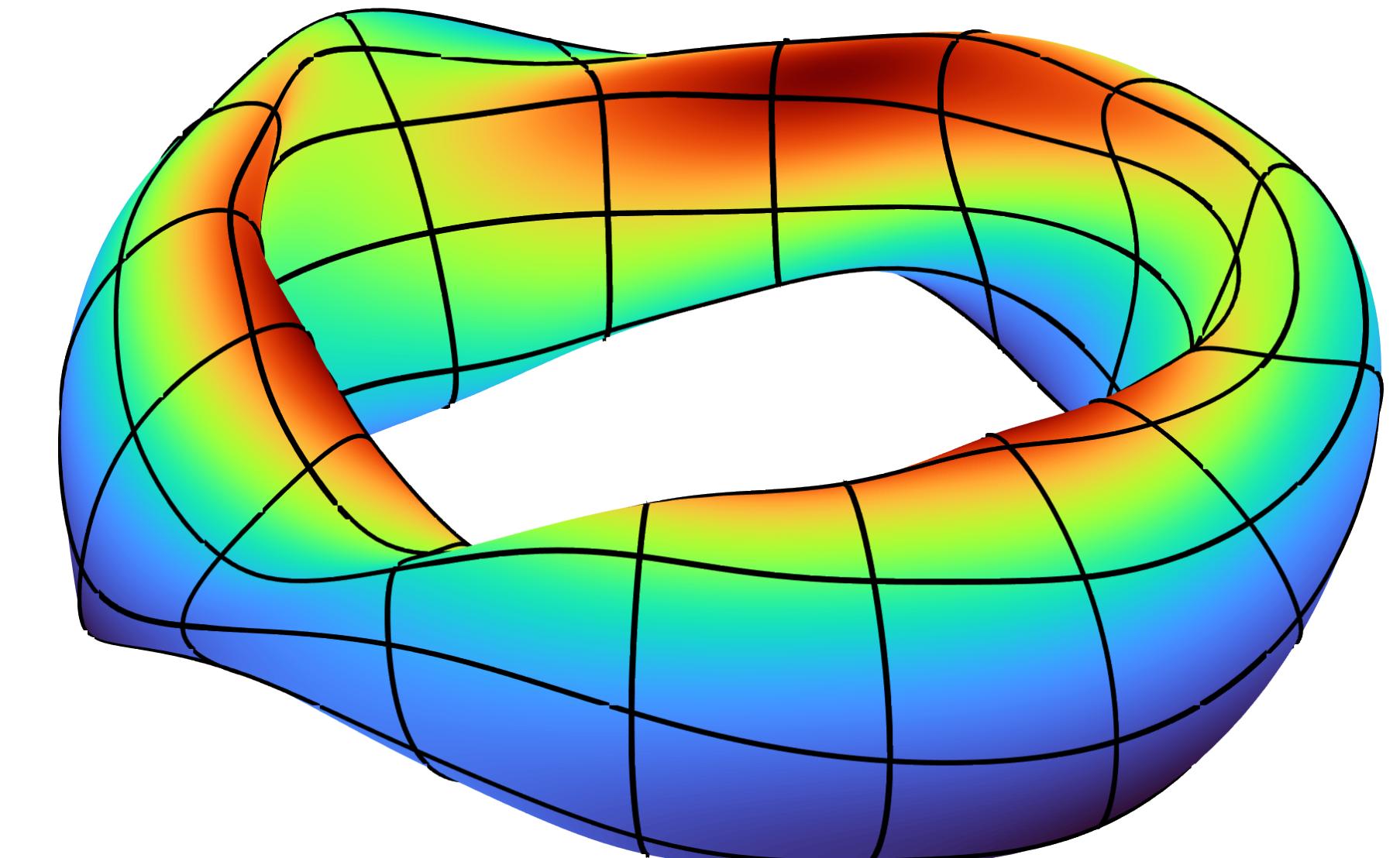
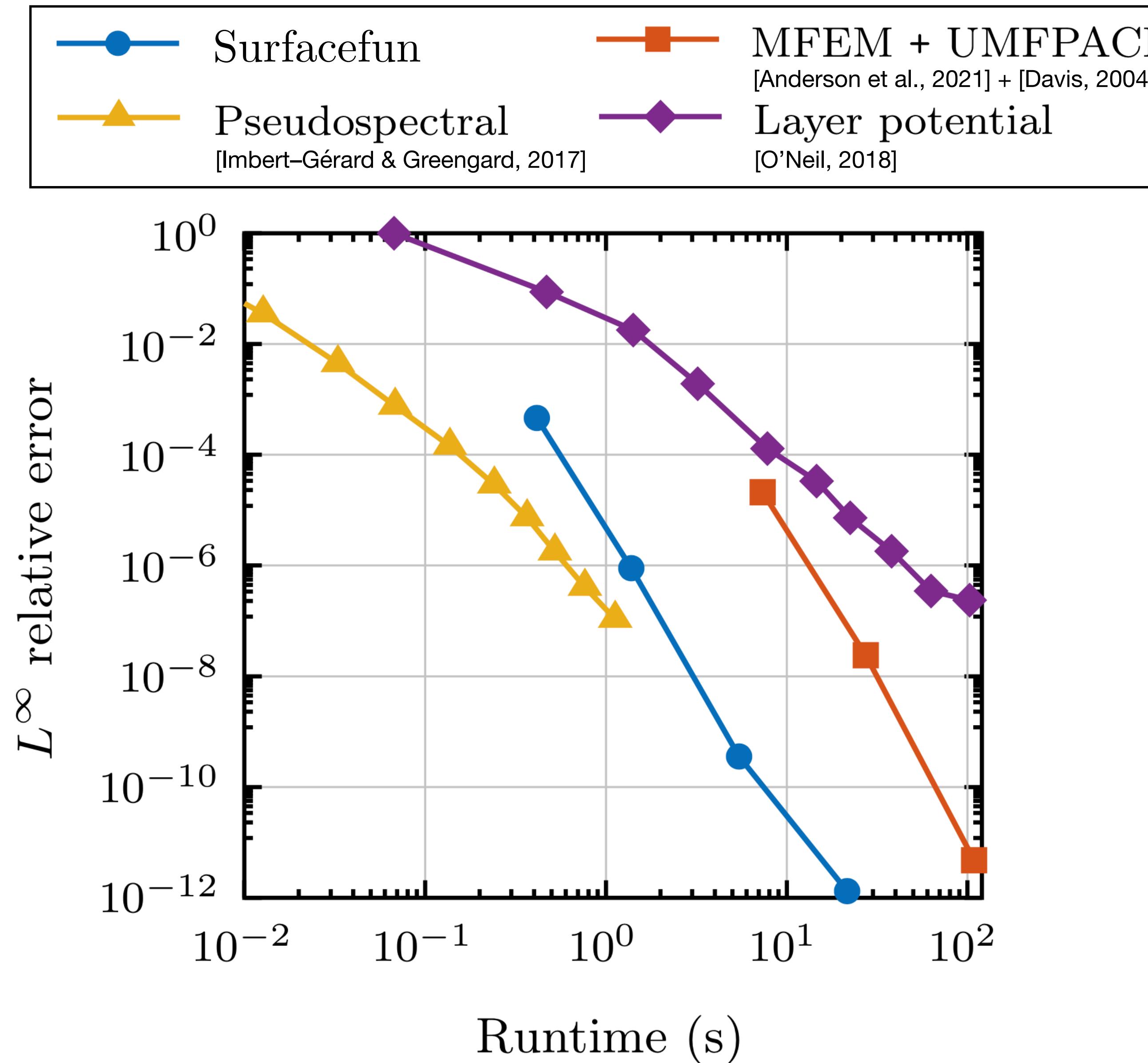
Significant speedups at high order when compared to FEM with a sparse direct solver.



Examples

Laplace–Beltrami: “accuracy vs. effort”

surfacefun.readthedocs.io



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

Surfacefun tutorial

<https://surfacefun.readthedocs.io/>

Example 1

Hodge decomposition

Examples

Hodge decomposition

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

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

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

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

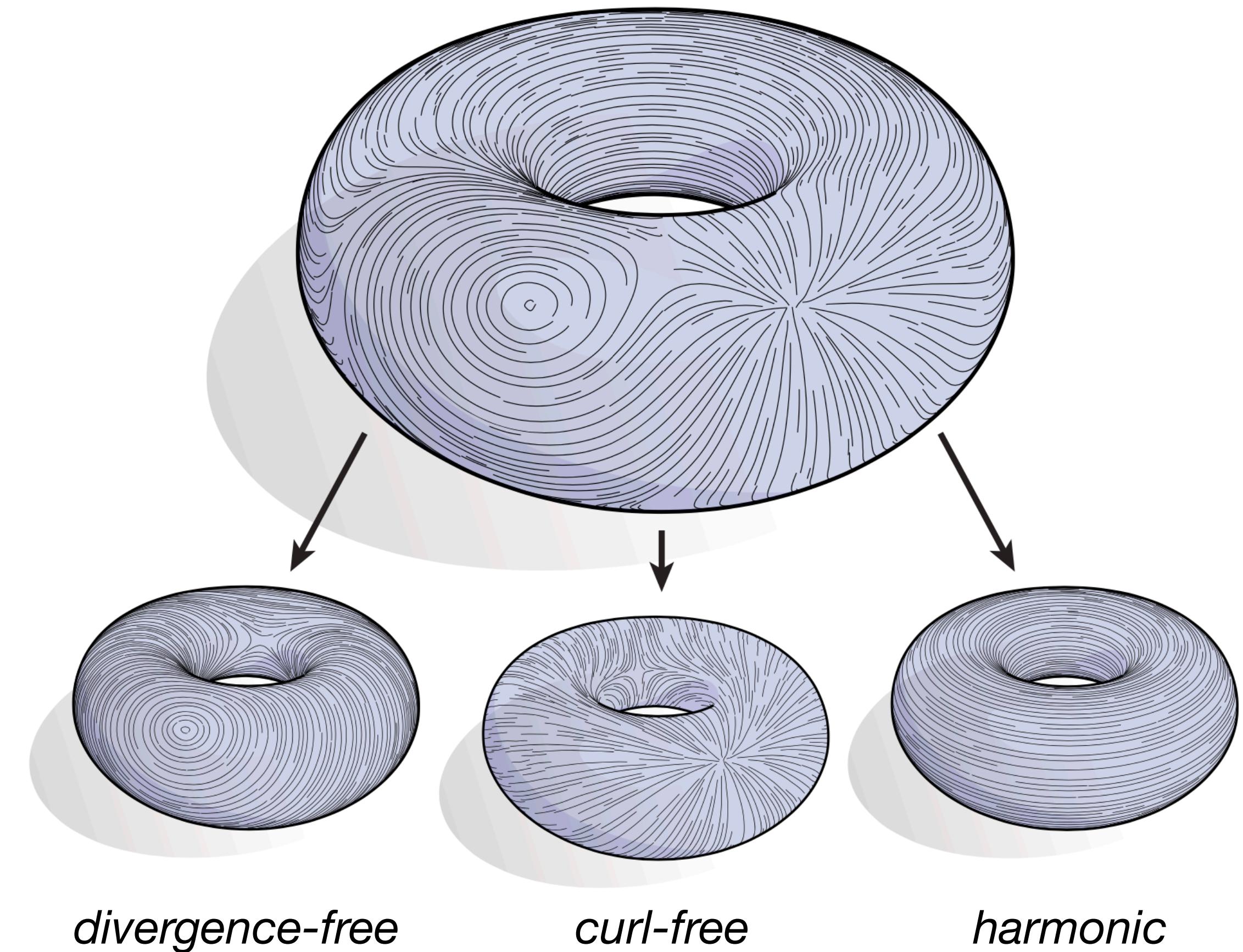


Illustration by Keenan Crane

Examples

Hodge decomposition

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

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

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

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

One may compute this decomposition by solving

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

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

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

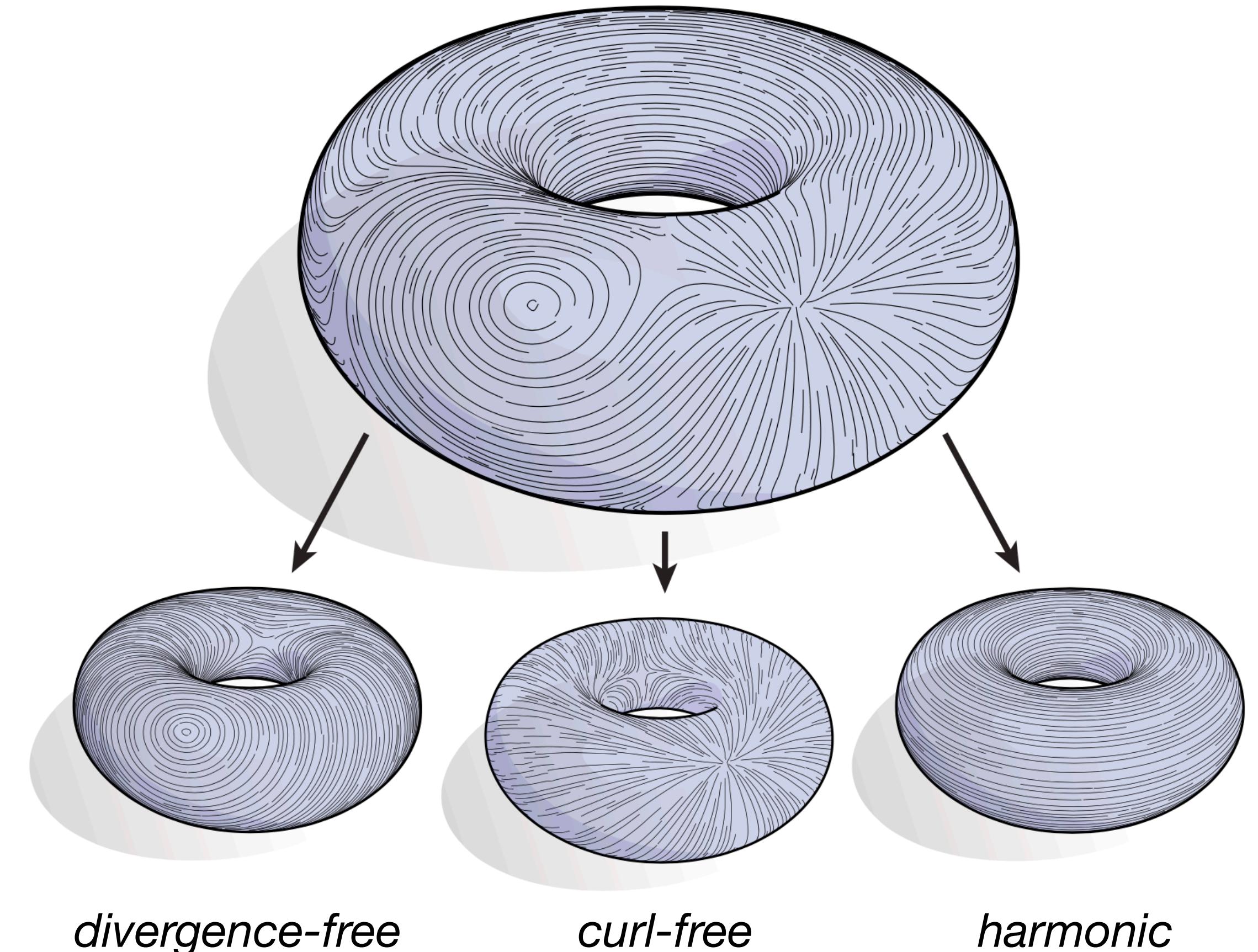
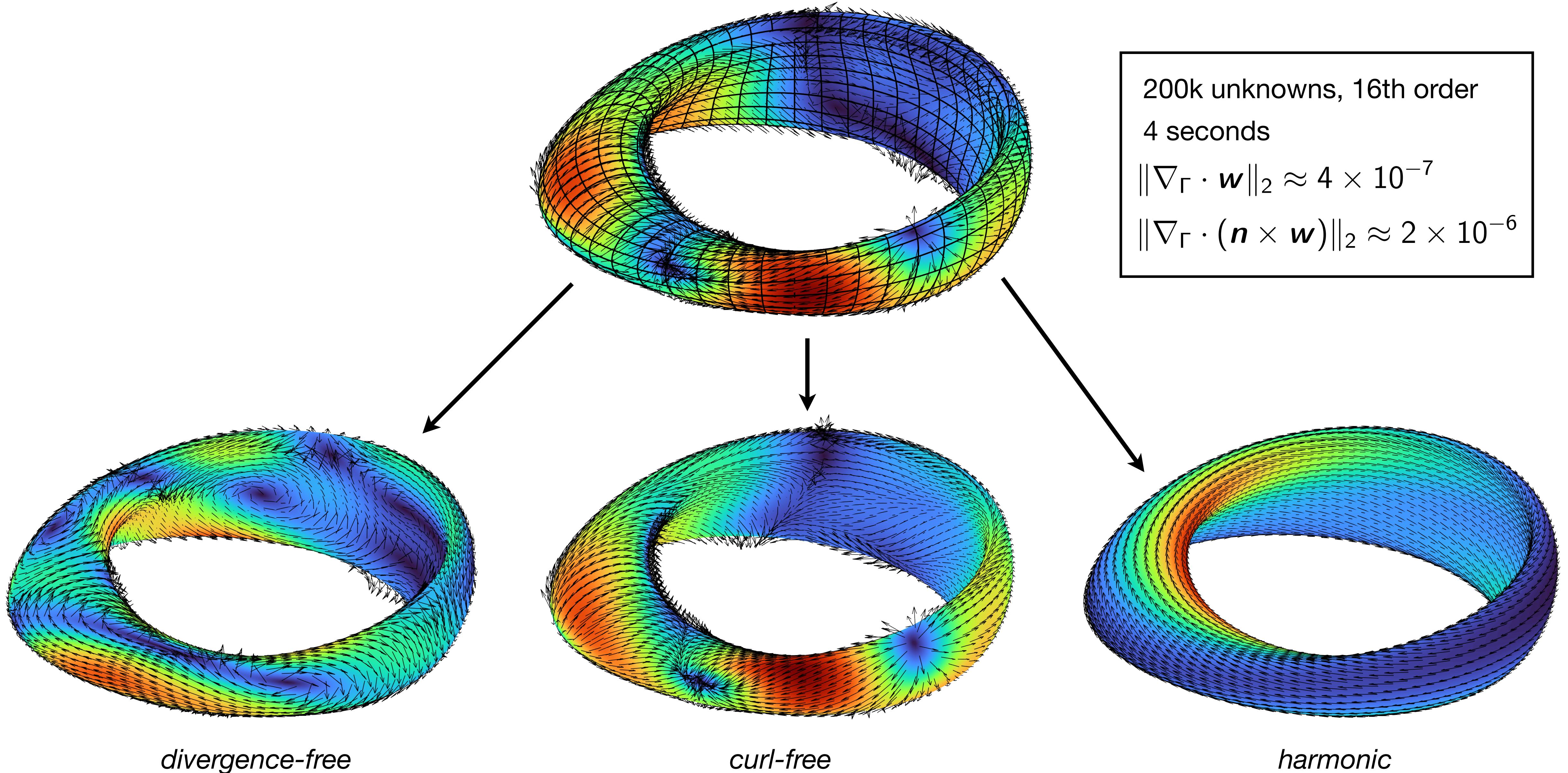


Illustration by Keenan Crane

Examples

Hodge decomposition



Example 2

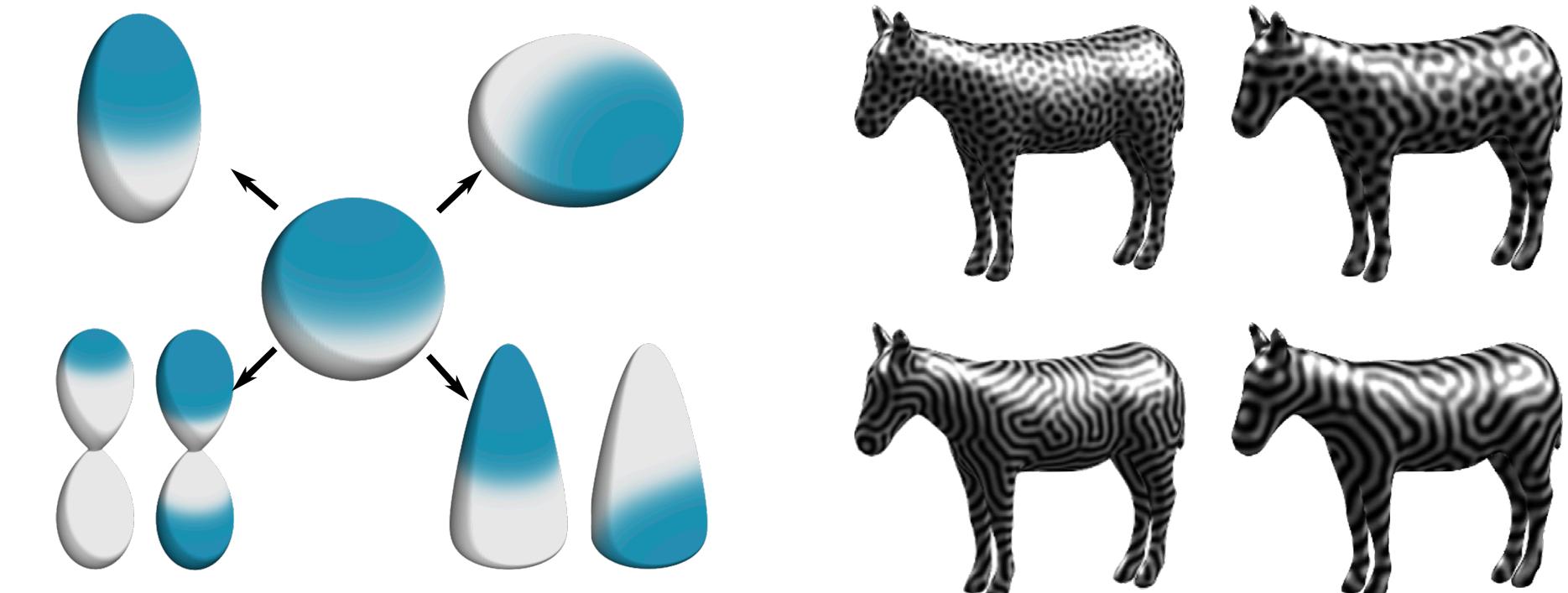
Time-dependent reaction-diffusion

Examples

Reaction–diffusion systems

- Reaction–diffusion processes are ubiquitous in biology.

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



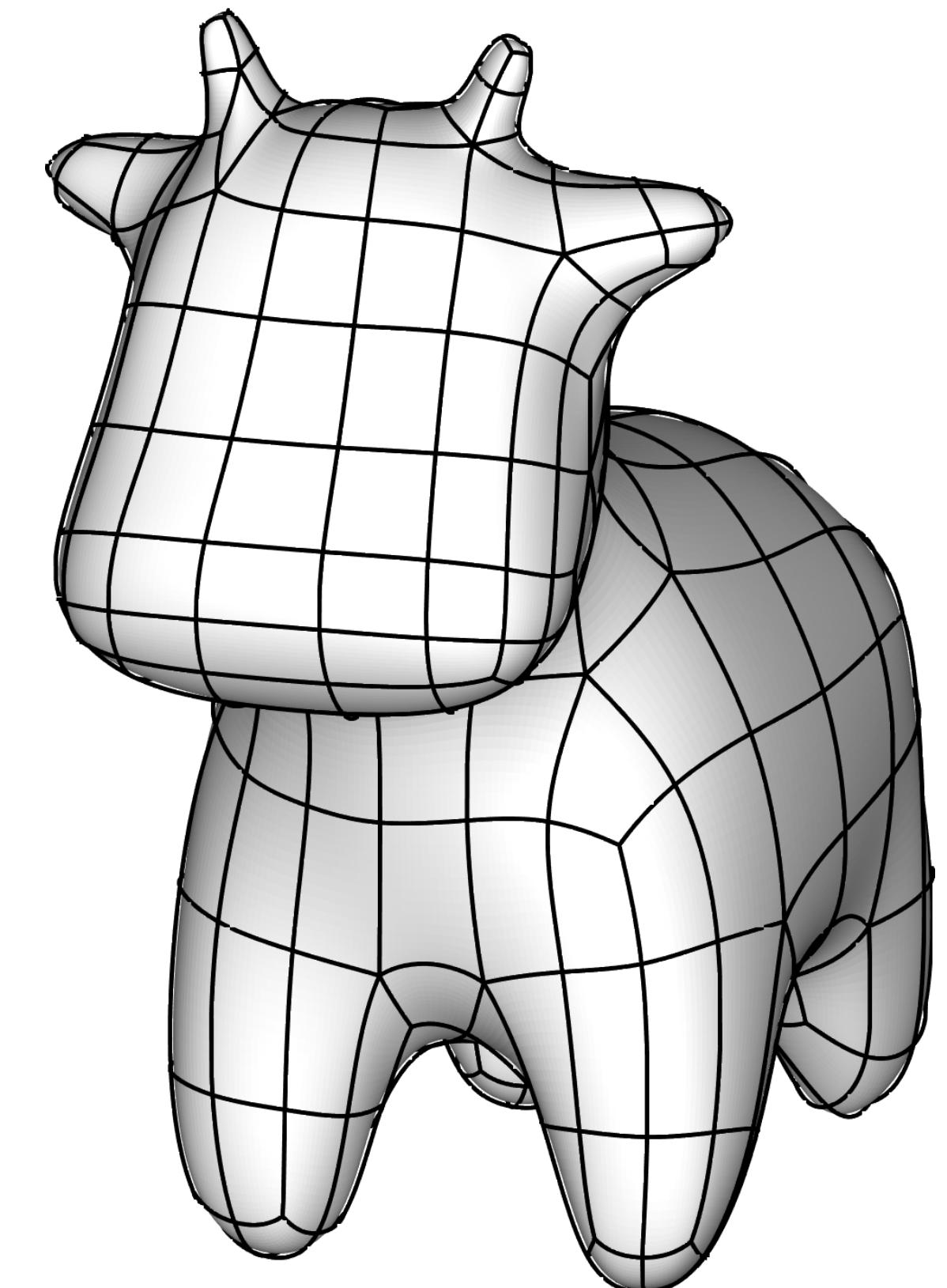
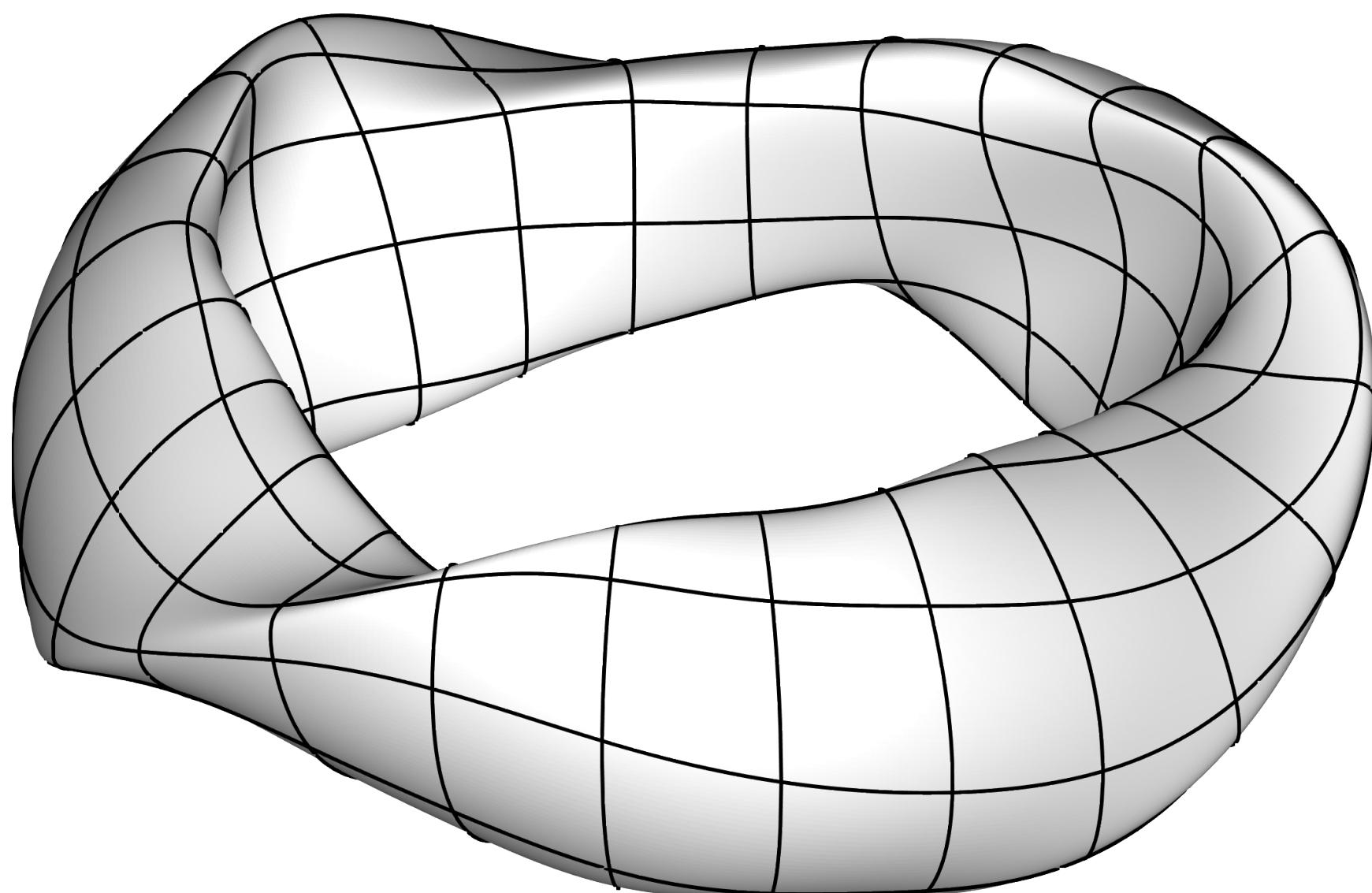
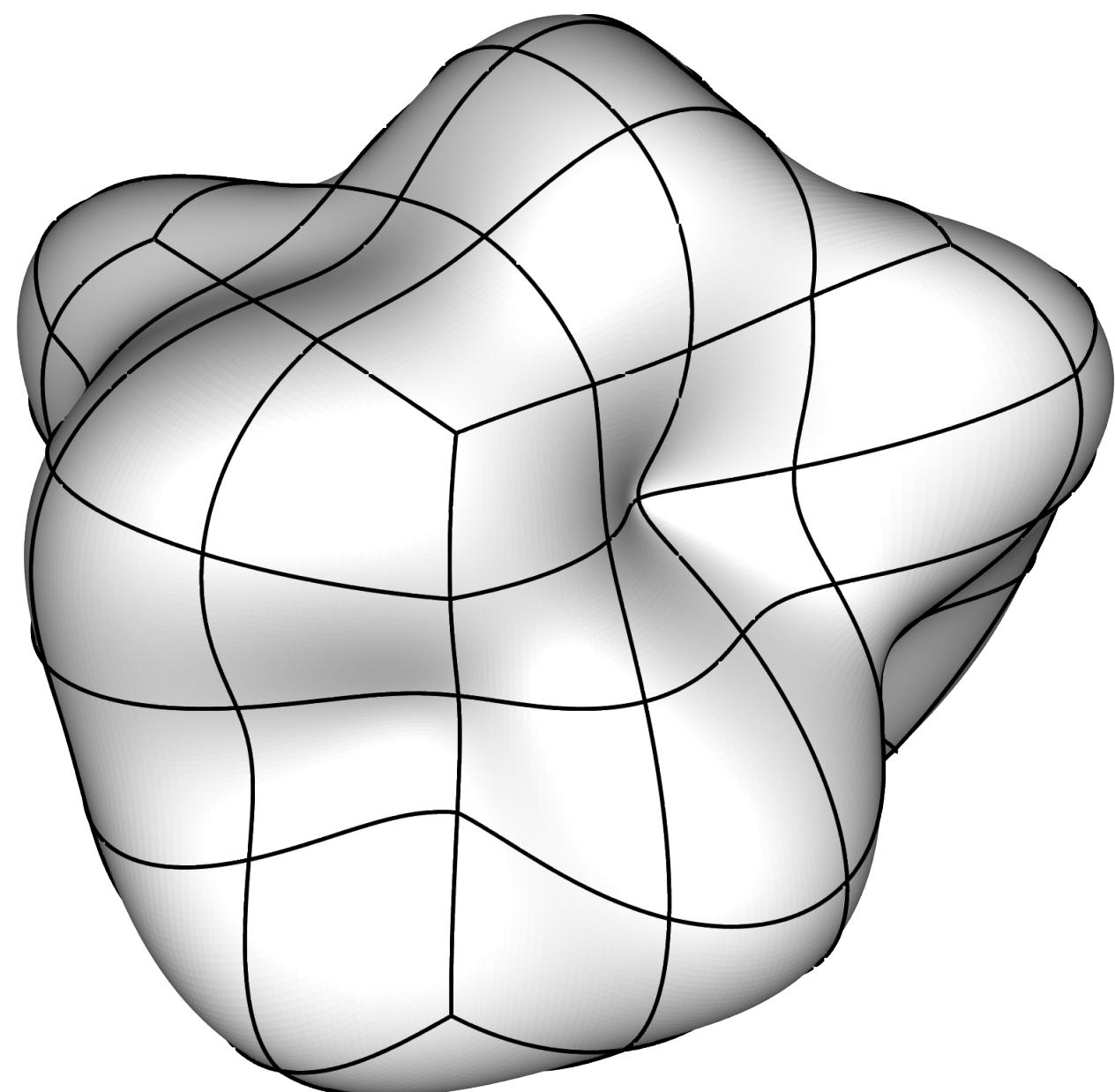
- Reaction and diffusion timescales are often orders of magnitude different.
- Implicit time-stepping can alleviate stability issues (e.g., backward Euler or IMEX-BDF4)

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

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

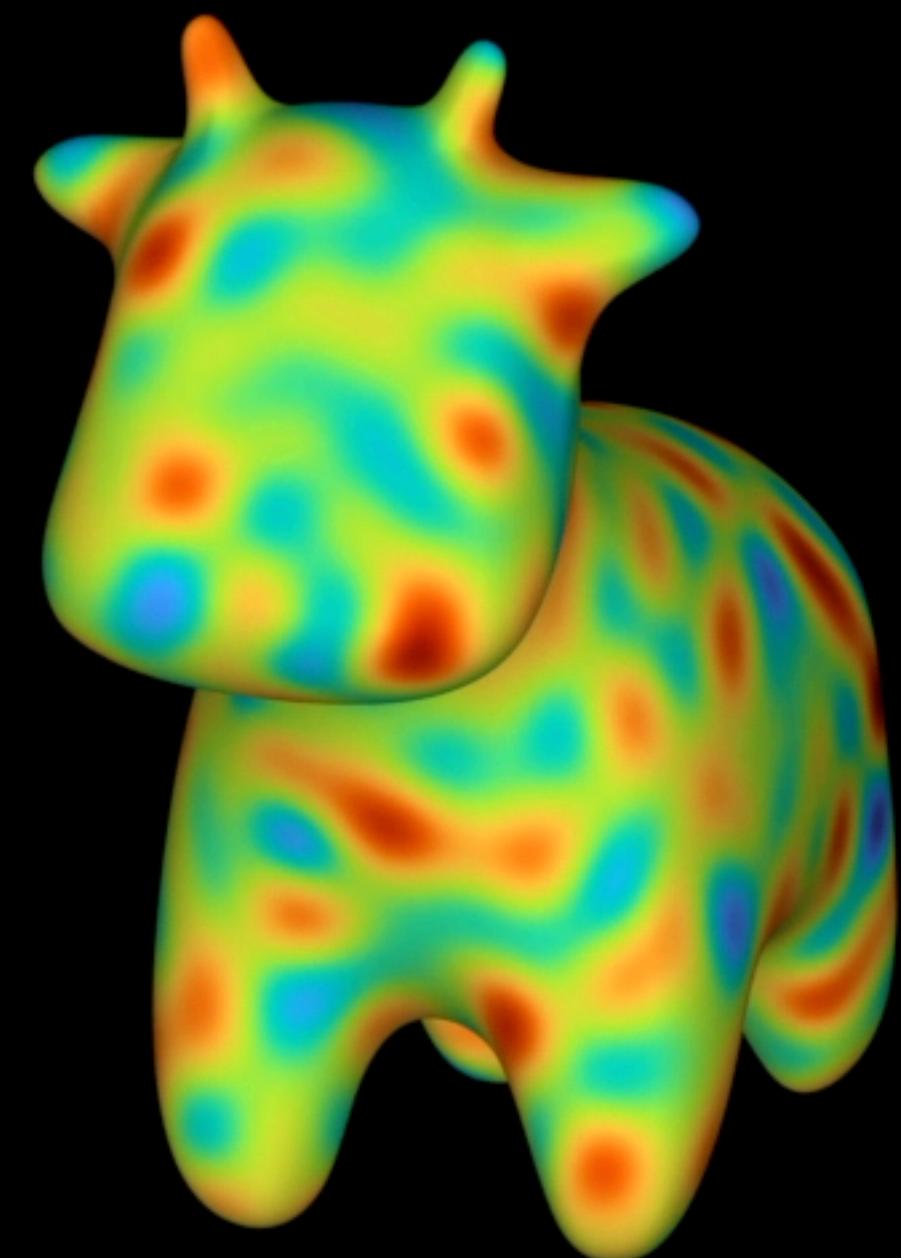
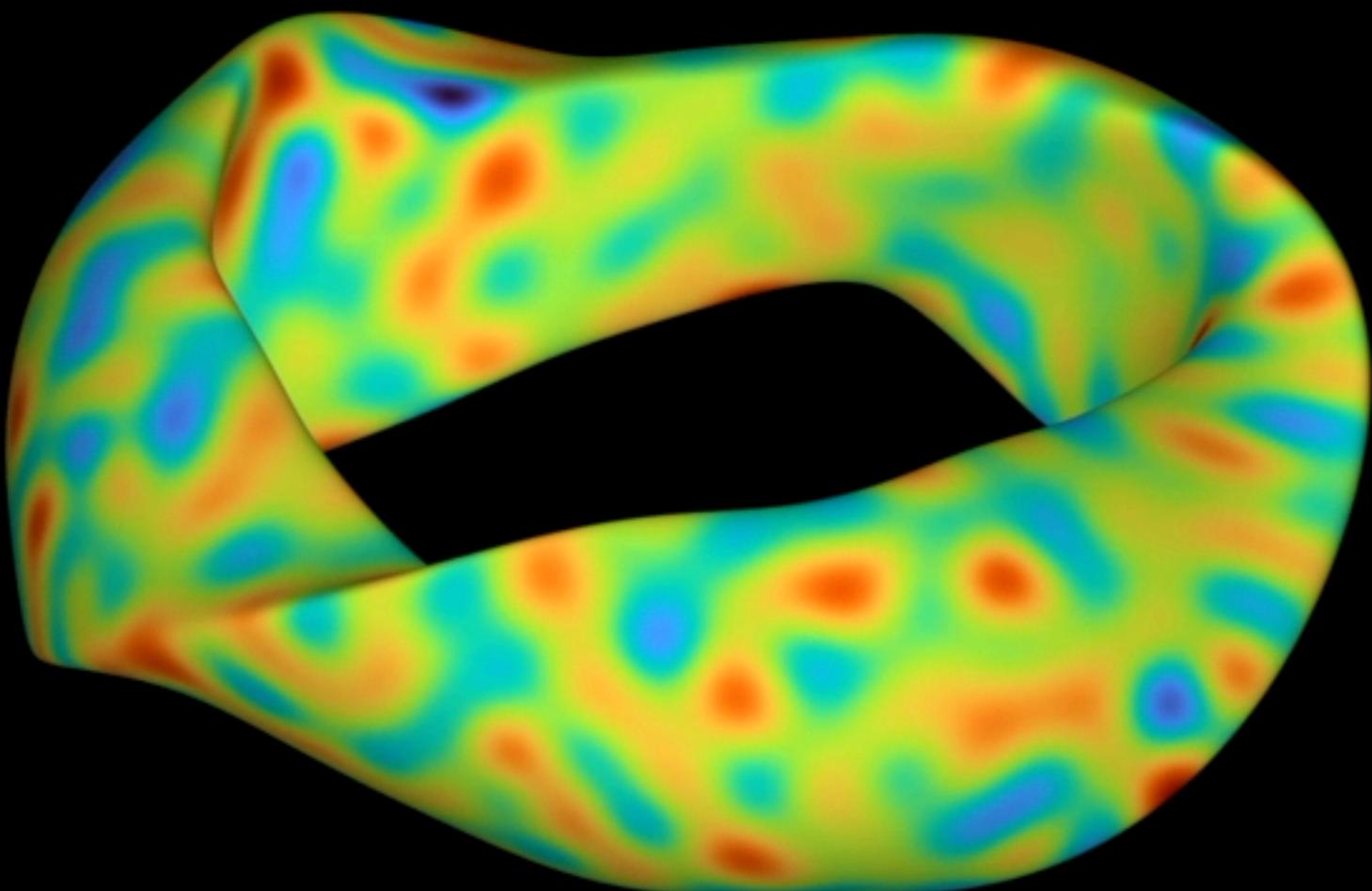
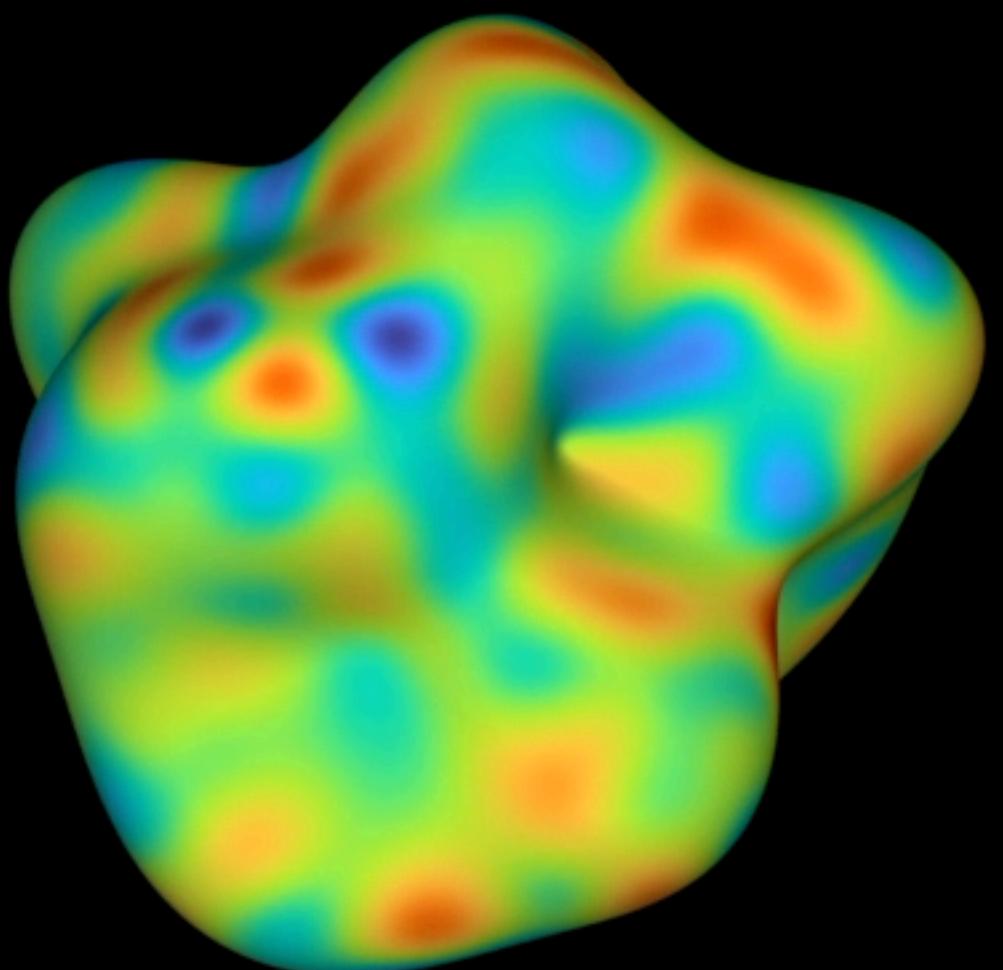
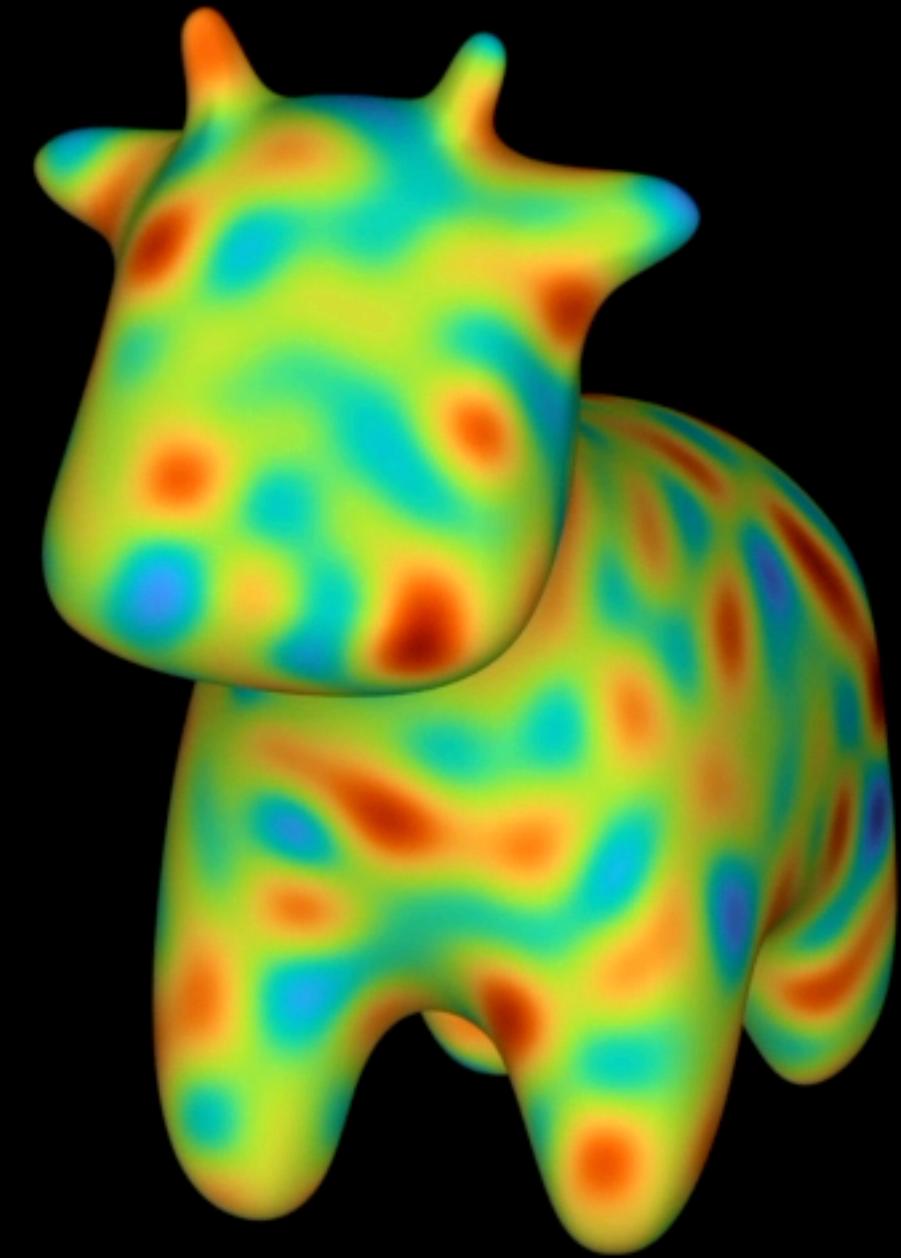
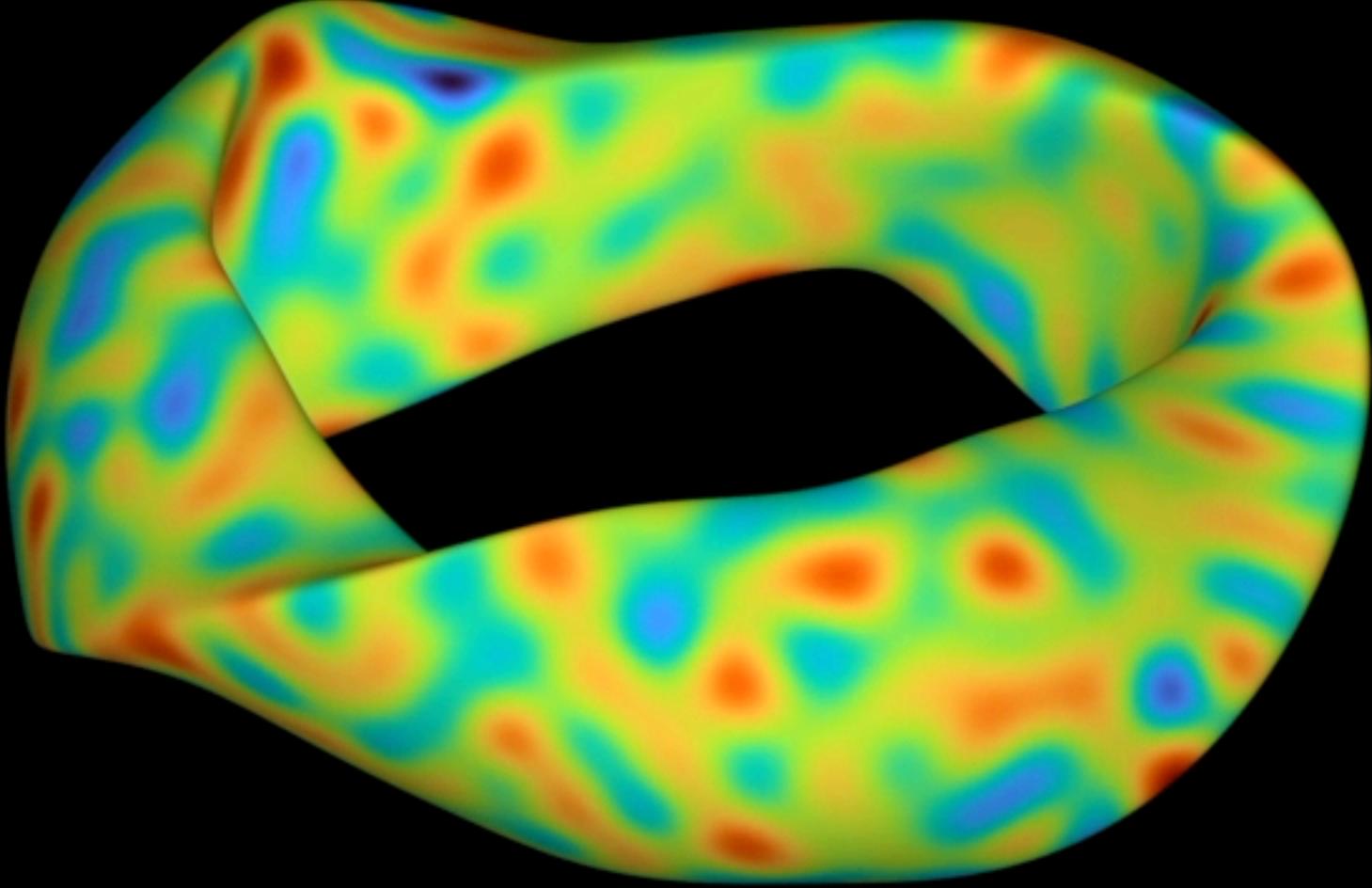
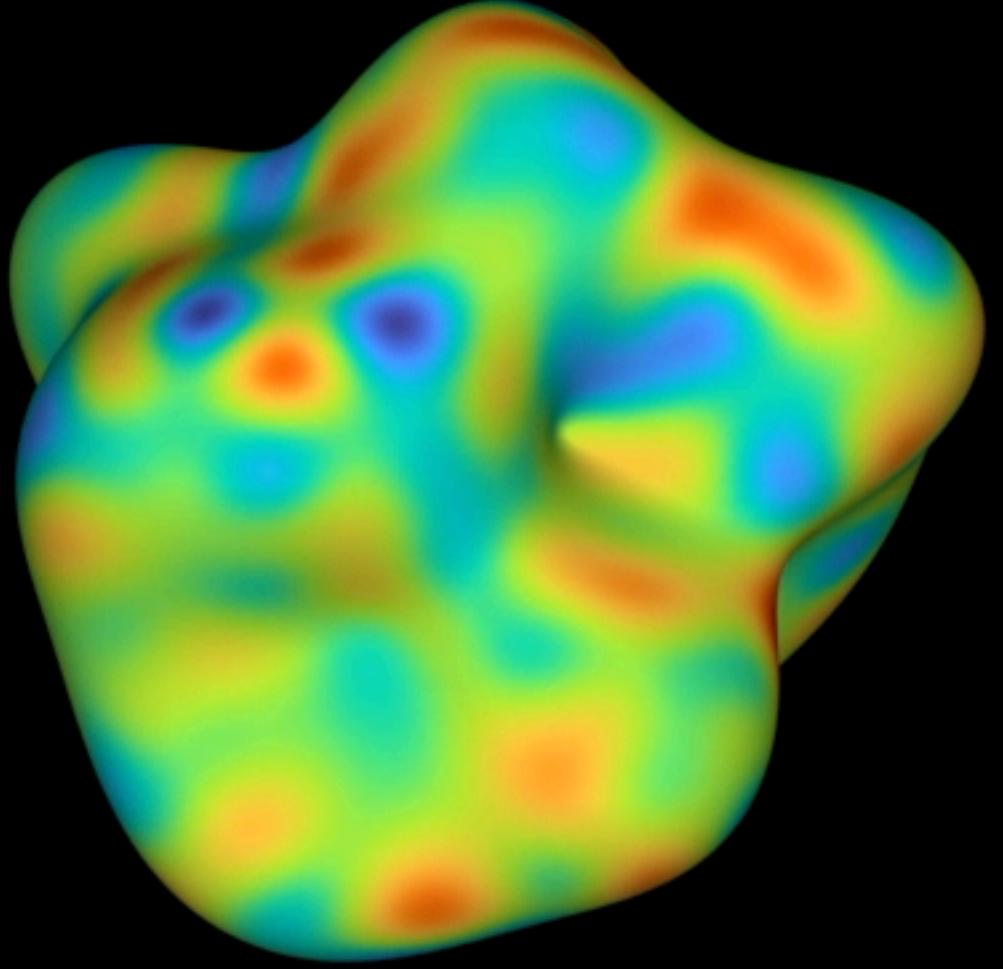
Examples

Reaction–diffusion systems



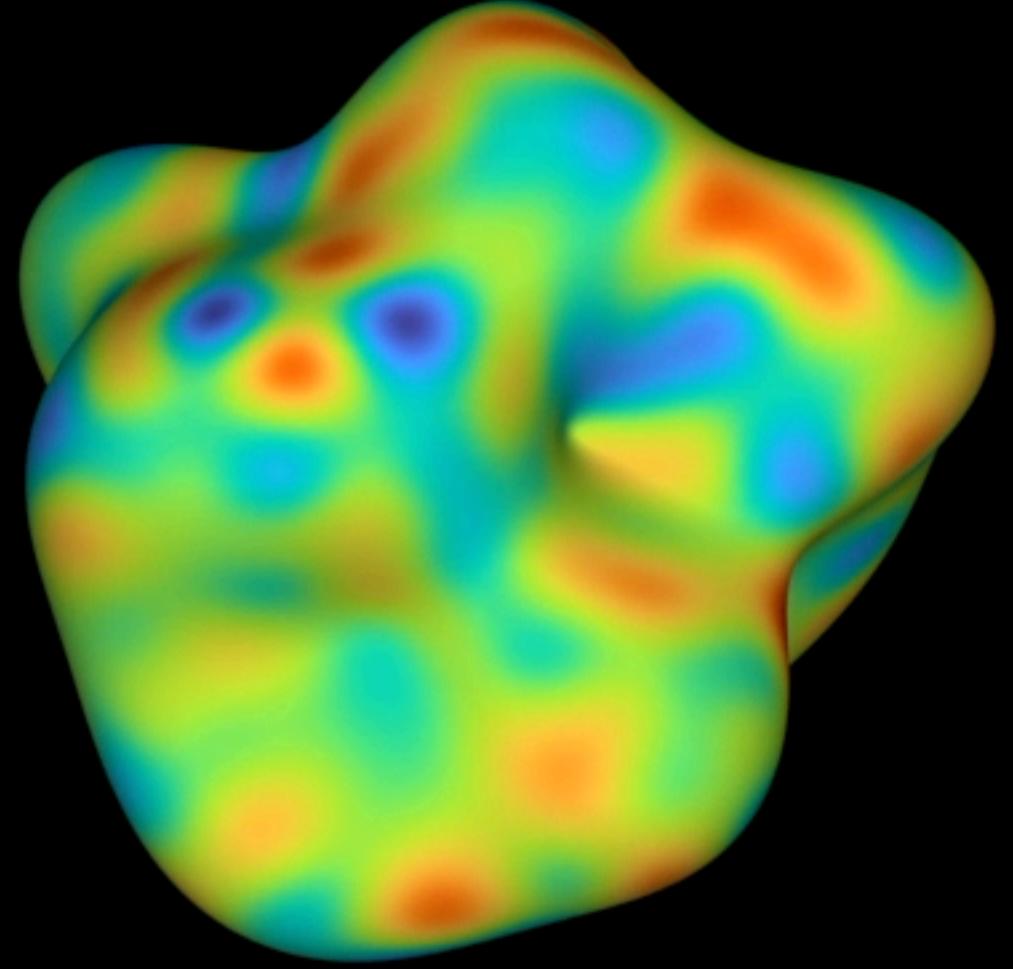
16th order
8-core Intel i9 Macbook Pro
64GB RAM

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

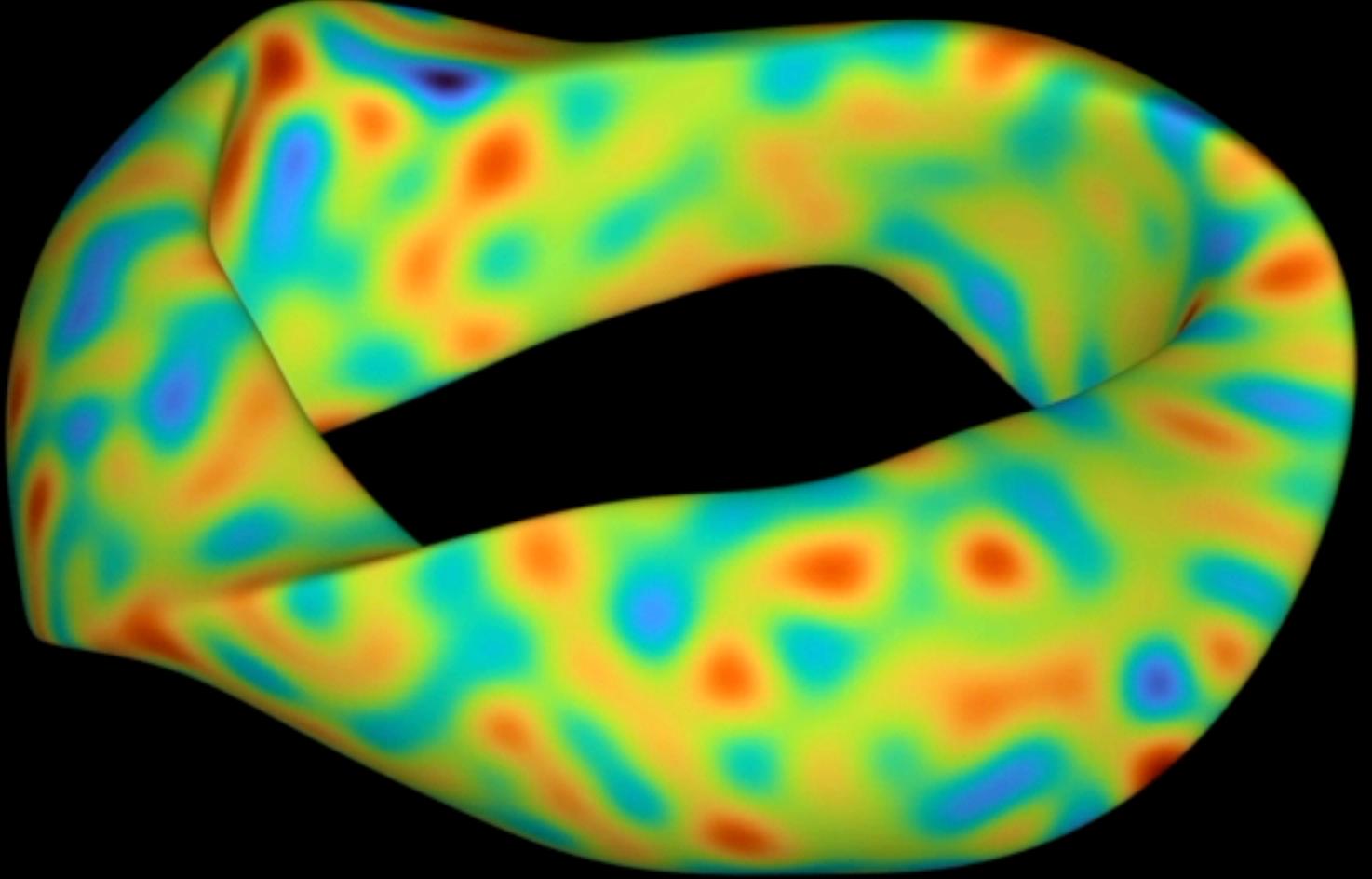


$$\frac{\partial u}{\partial t} = \delta^2 \Delta_\Gamma u + u - (1 + ci)u|u|^2$$

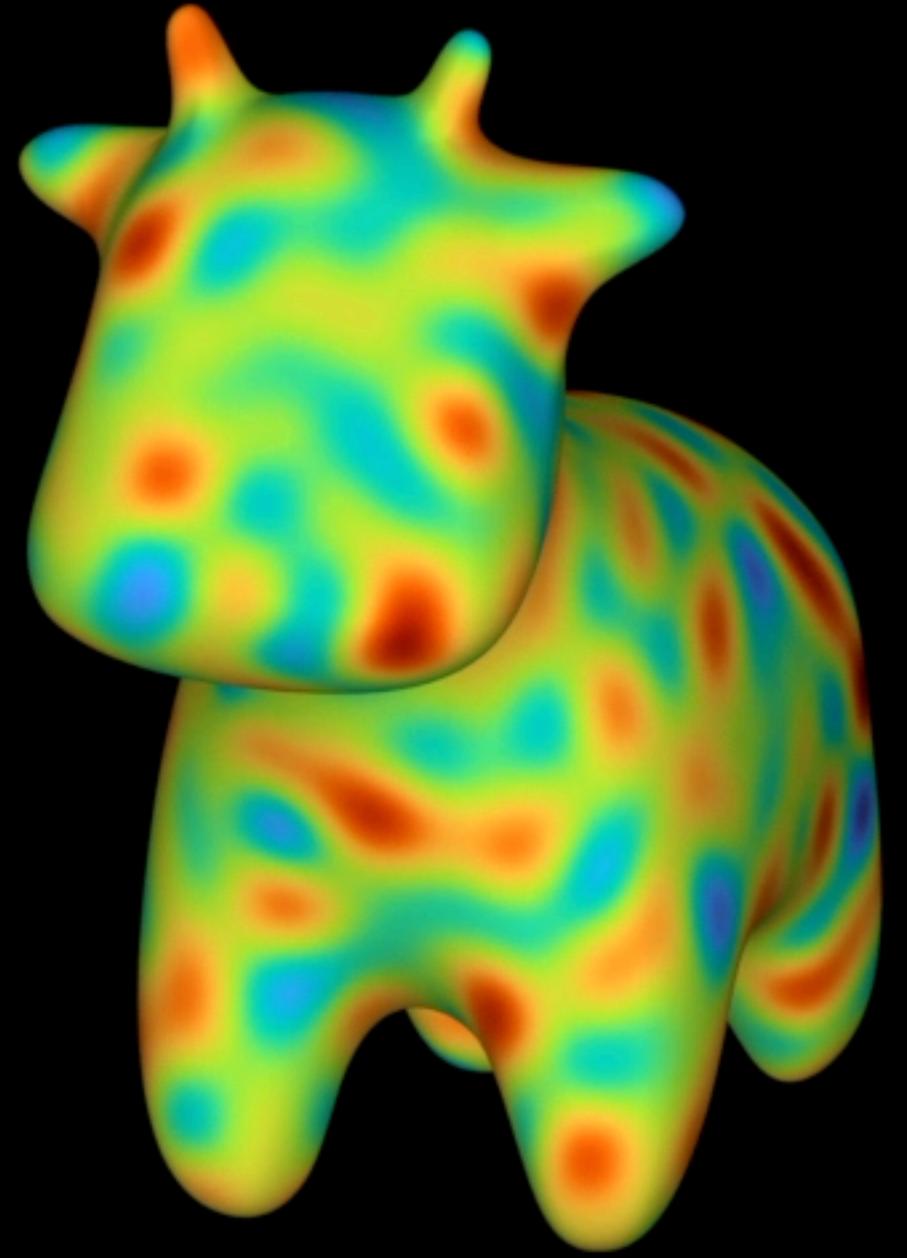
$$\begin{aligned}\frac{\partial u}{\partial t} &= \delta_u^2 \Delta_\Gamma u + \alpha u(1 - \tau_1 v^2) + v(1 - \tau_2 u) \\ \frac{\partial v}{\partial t} &= \delta_v^2 \Delta_\Gamma v + \beta v(1 + \alpha \tau_1 u v / \beta) + u(\gamma + \tau_2 v)\end{aligned}$$



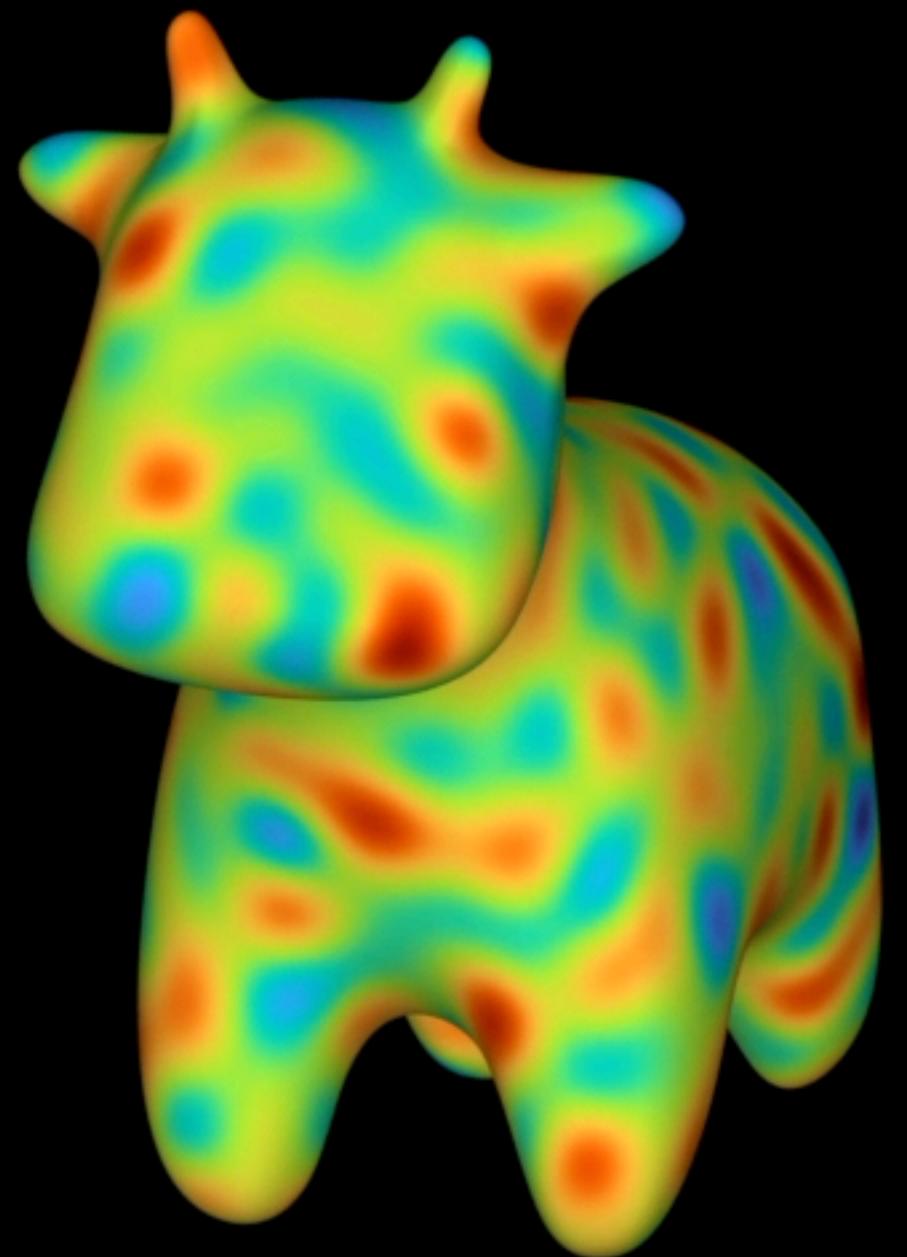
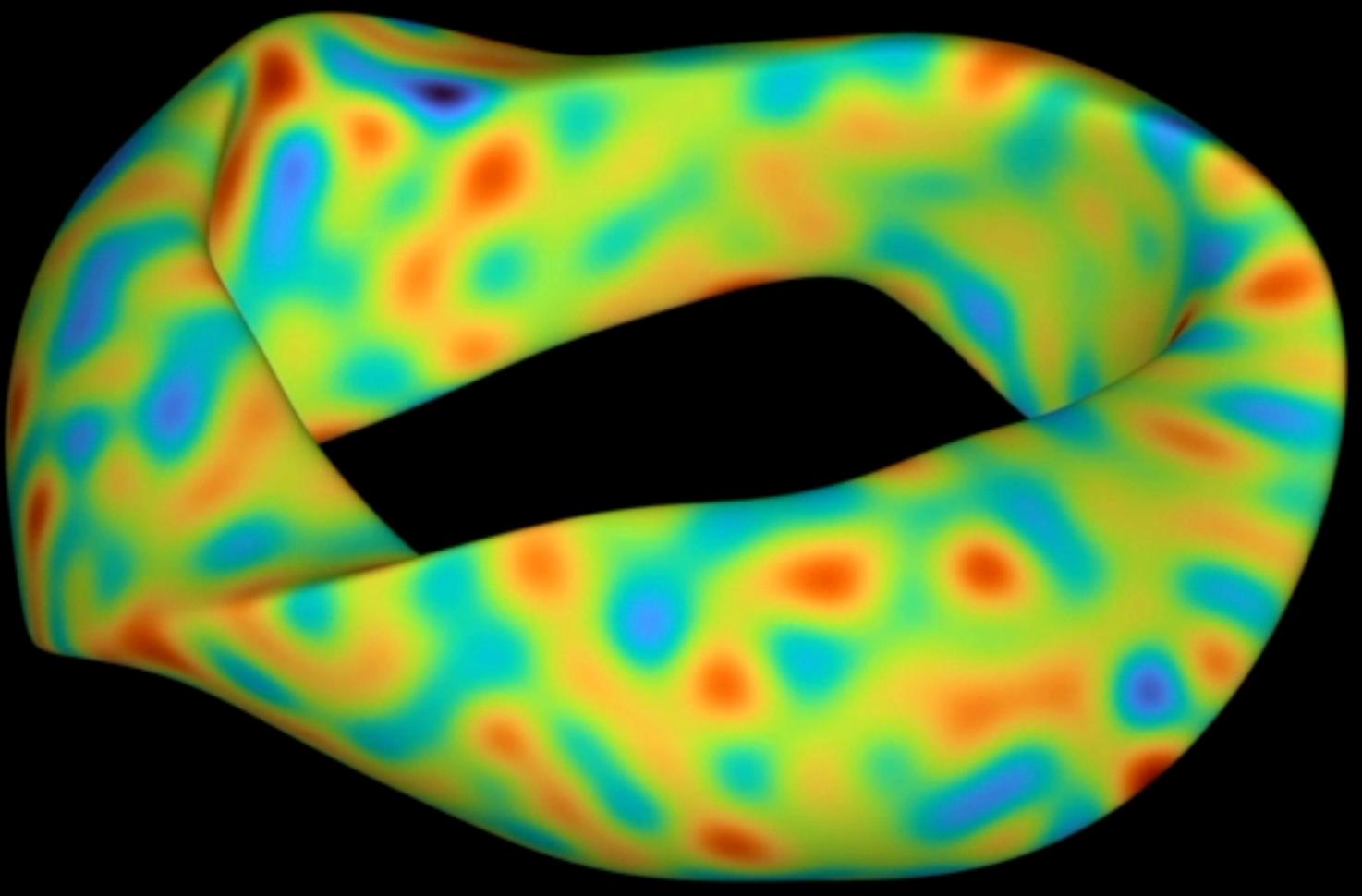
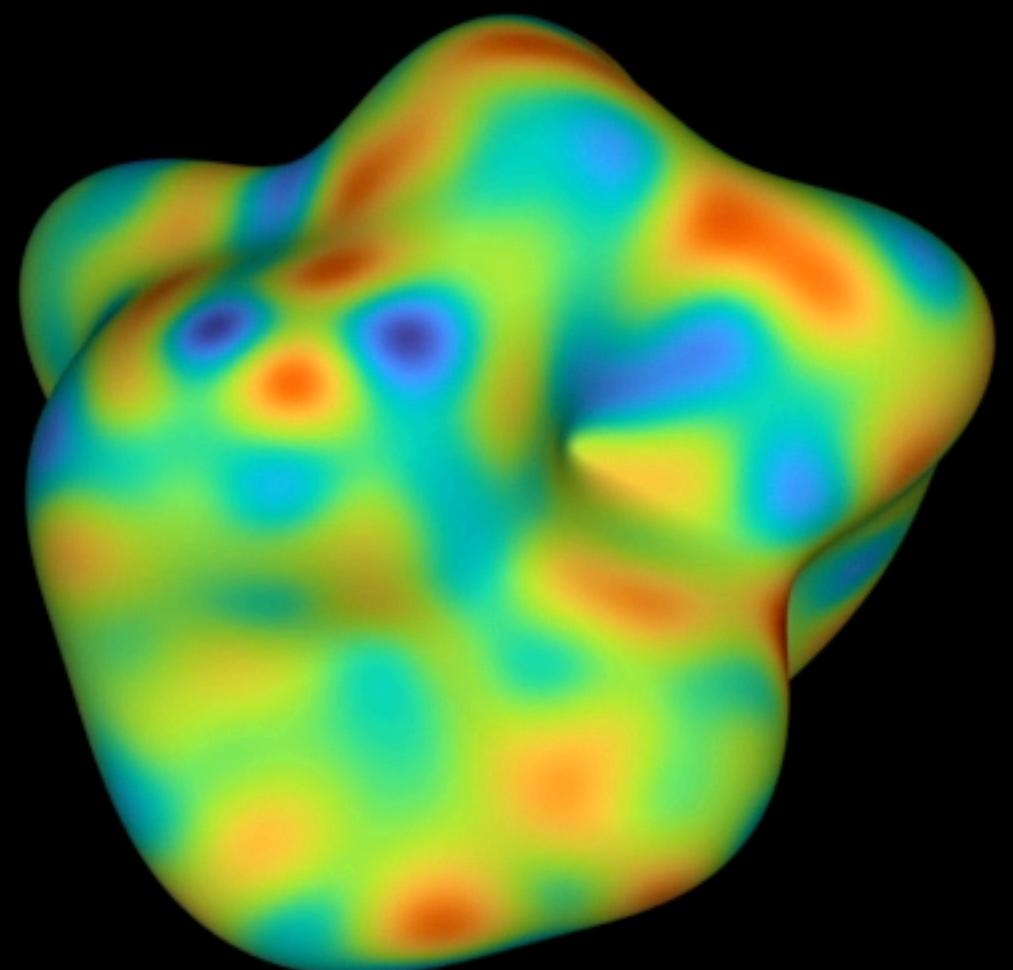
25k unknowns
~20 fps



50k unknowns
~13 fps



135k unknowns
~4 fps

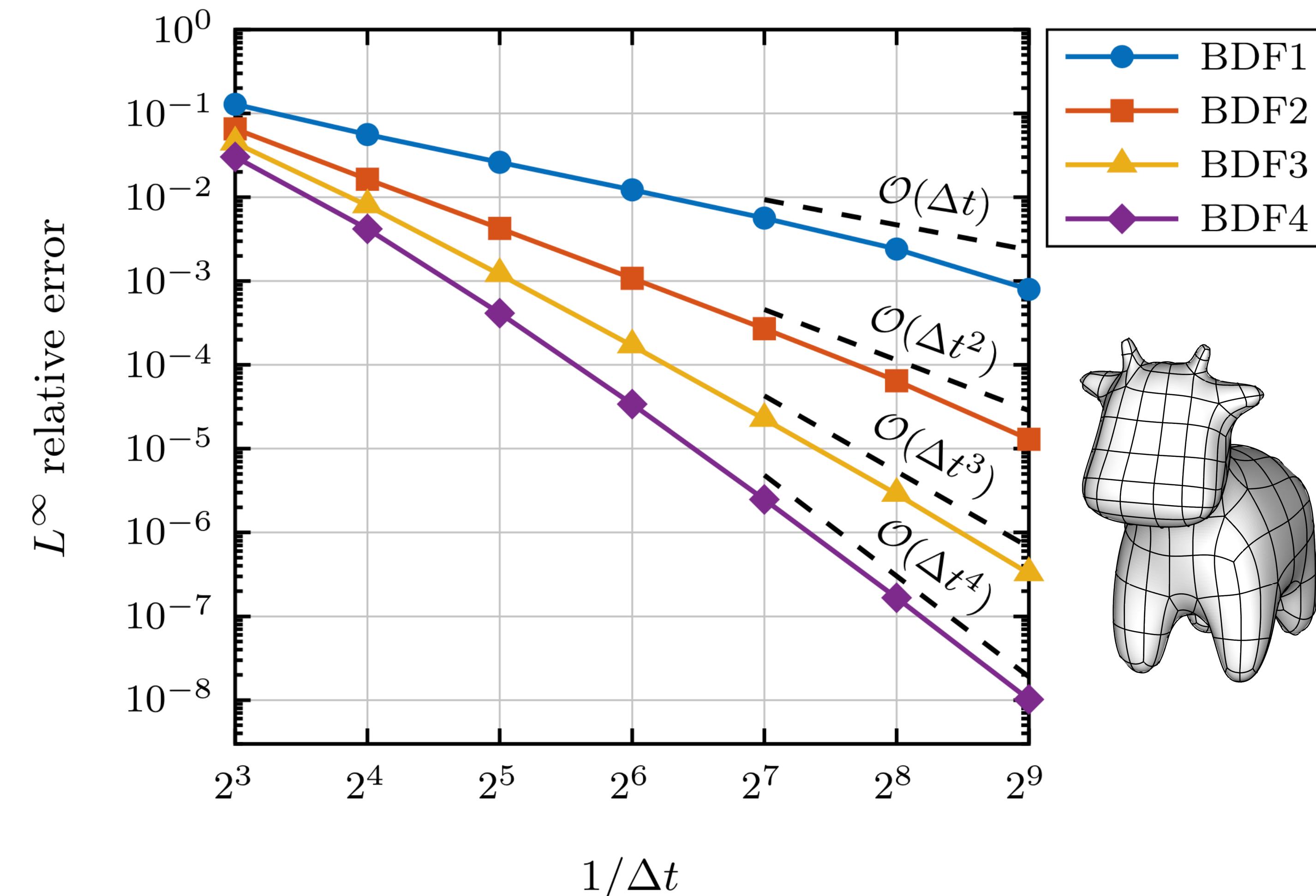


$$\frac{\partial u}{\partial t} = \delta^2 \Delta_\Gamma u + u - (1 + ci)u|u|^2$$

Examples

Reaction-diffusion systems

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



Example 3

Eigenvalue problems

Examples

Eigenvalue problems

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

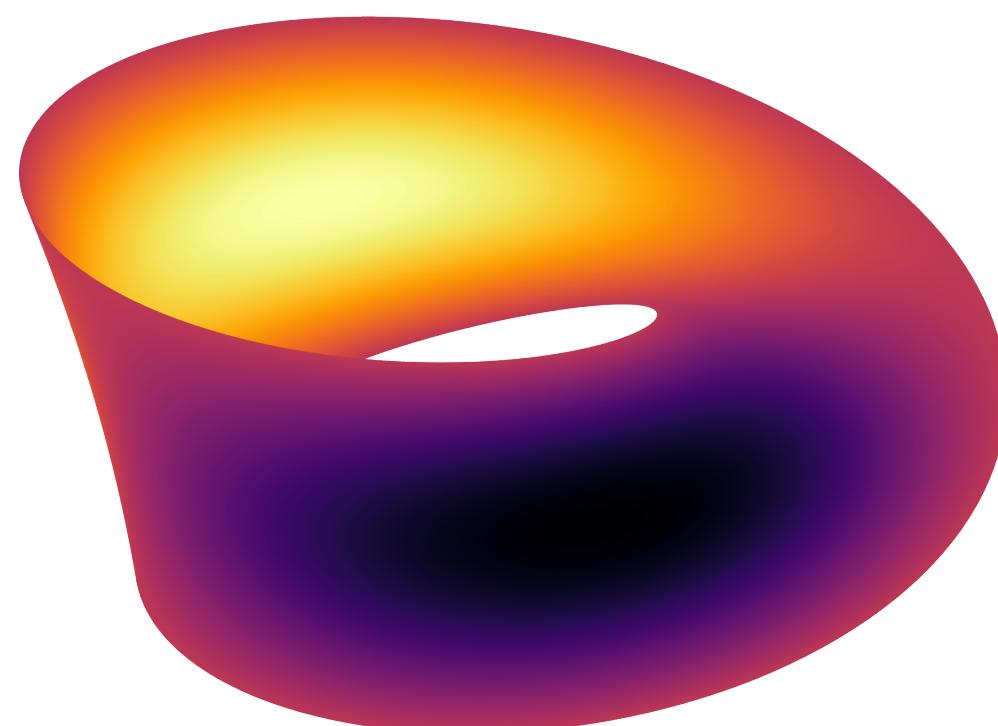
Simultaneous inverse iteration:

$Q^{(0)} = \text{rand}(N, m)$
for $k = 1, 2, \dots$

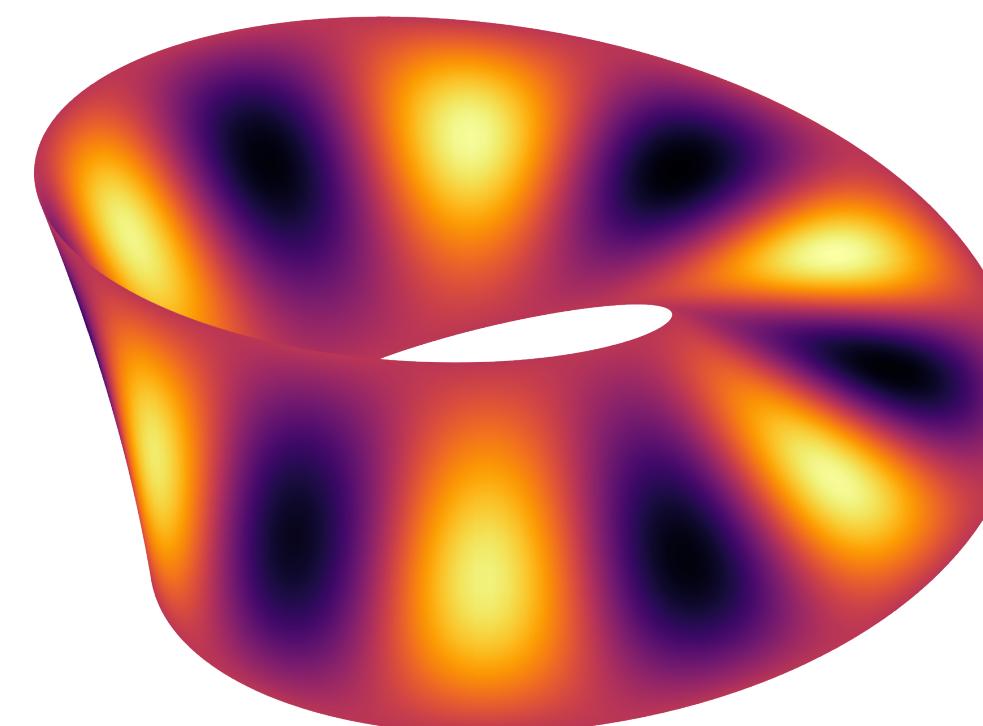
Stored in RAM,
very fast apply

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

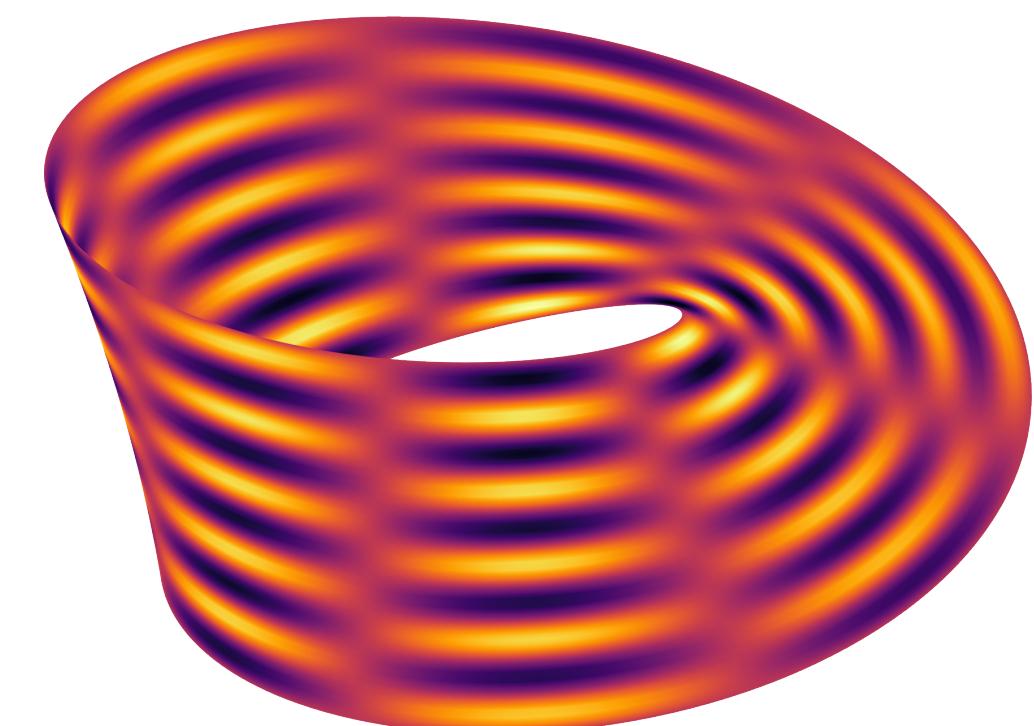
$$Q^{(k)} R^{(k)} = Z^{(k-1)}$$



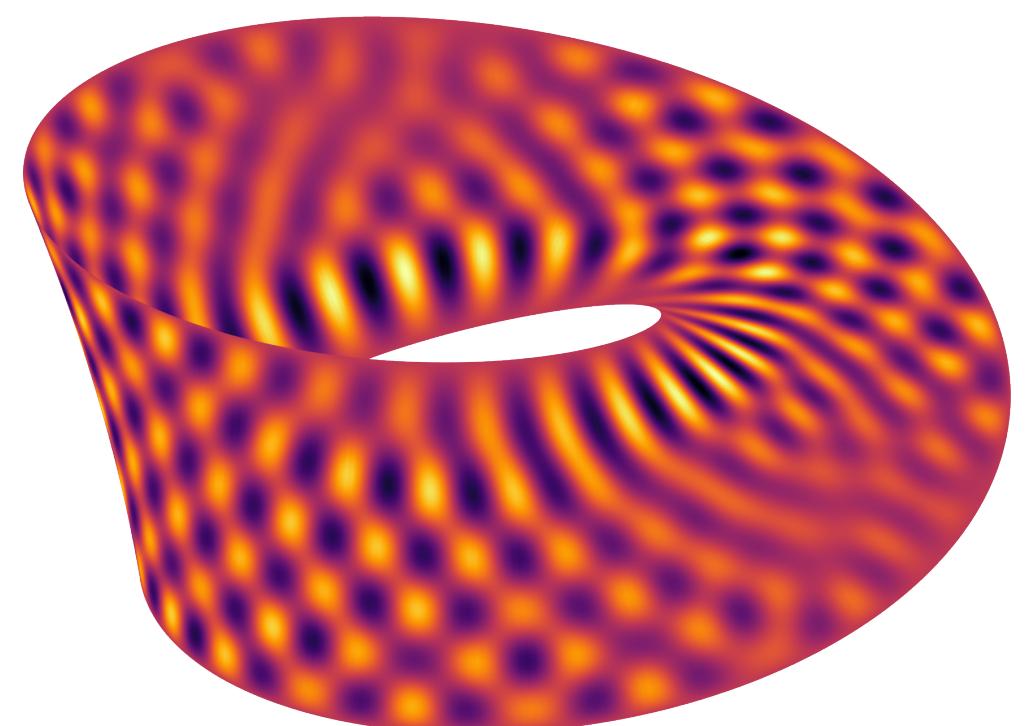
$$\lambda = -10.93\dots$$



$$\lambda = -44.66\dots$$



$$\lambda = -1000.56\dots$$



$$\lambda = -998.11\dots$$

Example 4

Surface scattering

Examples

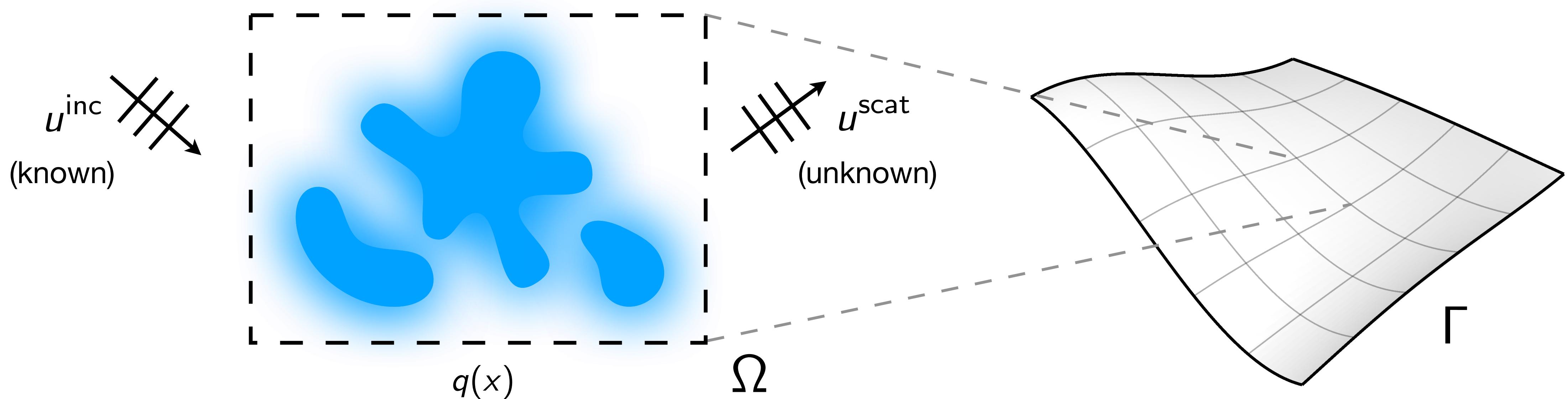
Scattering on a surface

Consider wave scattering in a variable medium on a manifold Γ , modeled by the surface Helmholtz equation

$$\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) = 0$$

with the total field $u = u^{\text{inc}} + u^{\text{scat}}$ and the Sommerfeld radiation condition

$$\lim_{r \rightarrow \infty} \sqrt{r} \left(\frac{\partial u^{\text{scat}}}{\partial r} - iku^{\text{scat}} \right) = 0$$



[Gillman, Barnett, Martinsson, 2015]

[Borges, Gillman, Greengard, 2017]

Examples

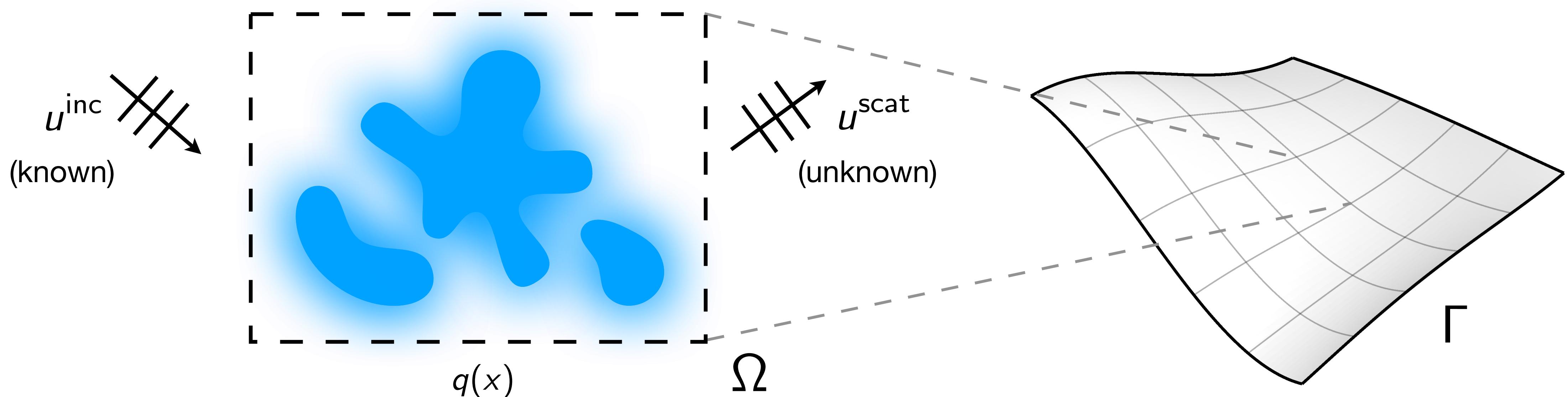
Scattering on a surface

Consider wave scattering in a variable medium on a manifold Γ , modeled by the surface Helmholtz equation

$$\Delta_{\Gamma} u(x) + k^2(1 - q(x))u(x) = 0$$

with the total field $u = u^{\text{inc}} + u^{\text{scat}}$ and the Sommerfeld radiation condition

$$\lim_{r \rightarrow \infty} \sqrt{r} \left(\frac{\partial u^{\text{scat}}}{\partial r} - iku^{\text{scat}} \right) = 0$$



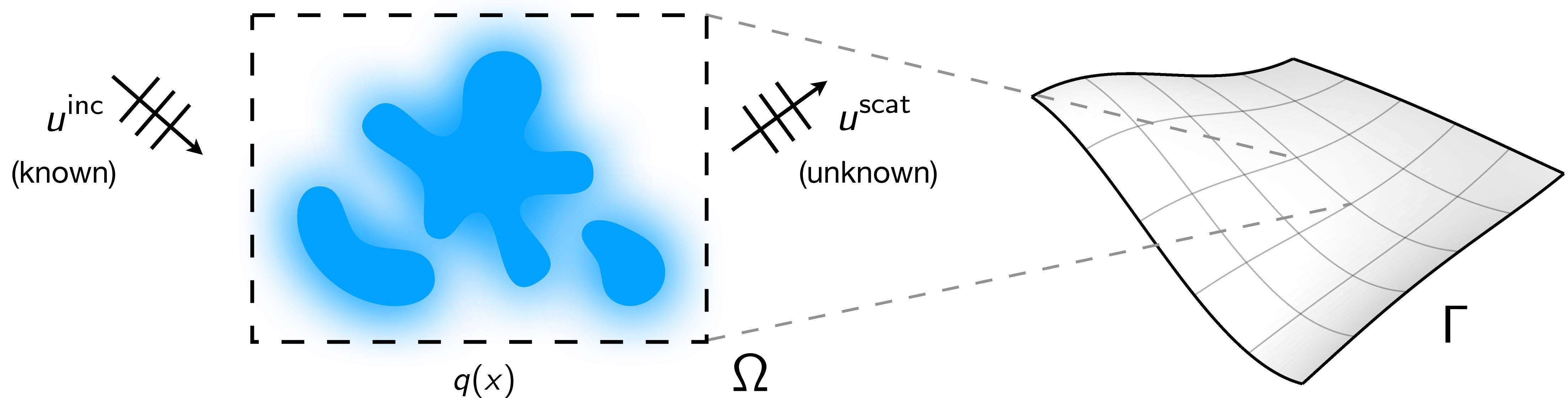
[Gillman, Barnett, Martinsson, 2015]

[Borges, Gillman, Greengard, 2017]

Examples

Scattering on a surface

For compactly supported $q(x)$ and Γ , we can decompose this into two problems:



[Gillman, Barnett, Martinsson, 2015]

[Borges, Gillman, Greengard, 2017]

Examples

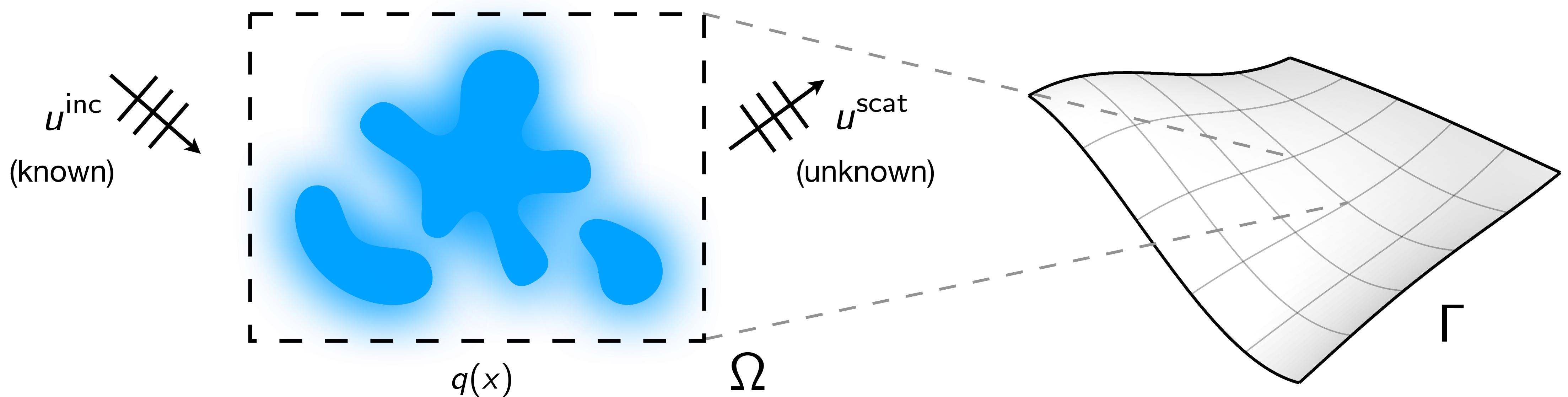
Scattering on a surface

For compactly supported $q(x)$ and Γ , we can decompose this into two problems:

Interior problem

$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= f(x) \quad \text{in } \Omega \\ u(x) &= g(x) \quad \text{on } \partial\Omega\end{aligned}$$

curved, variable coefficient



[Gillman, Barnett, Martinsson, 2015]

[Borges, Gillman, Greengard, 2017]

Examples

Scattering on a surface

For compactly supported $q(x)$ and Γ , we can decompose this into two problems:

Interior problem

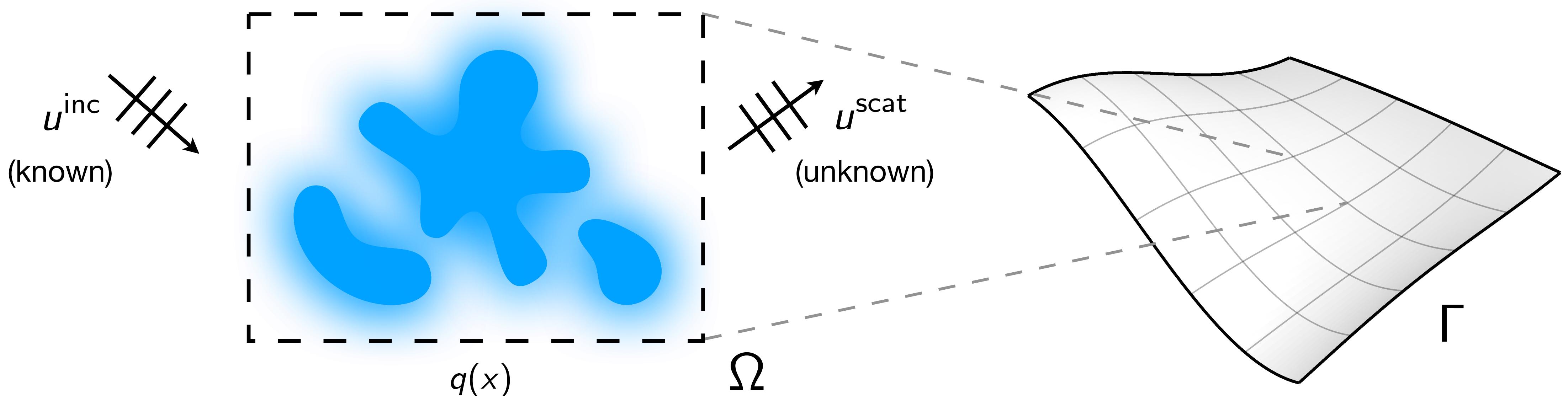
$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= f(x) \quad \text{in } \Omega \\ u(x) &= g(x) \quad \text{on } \partial\Omega\end{aligned}$$

Exterior problem

$$\begin{aligned}\Delta u^{\text{scat}}(x) + k^2 u^{\text{scat}}(x) &= 0 && \text{in } \mathbb{R}^2 \setminus \Omega \\ u^{\text{scat}}(x) &= h(x) && \text{on } \partial\Omega \\ \frac{\partial u^{\text{scat}}}{\partial r} - iku^{\text{scat}} &= o(r^{-1/2}) && \text{as } r = \|x\| \rightarrow \infty\end{aligned}$$

curved, variable coefficient

flat, constant coefficient



[Gillman, Barnett, Martinsson, 2015]

[Borges, Gillman, Greengard, 2017]

Examples

Scattering on a surface

For compactly supported $q(x)$ and Γ , we can decompose this into two problems:

Interior problem

$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= f(x) \quad \text{in } \Omega \\ u(x) &= g(x) \quad \text{on } \partial\Omega\end{aligned}$$

curved, variable coefficient

Exterior problem

$$\begin{aligned}\Delta u^{\text{scat}}(x) + k^2 u^{\text{scat}}(x) &= 0 && \text{in } \mathbb{R}^2 \setminus \Omega \\ u^{\text{scat}}(x) &= h(x) && \text{on } \partial\Omega \\ \frac{\partial u^{\text{scat}}}{\partial r} - iku^{\text{scat}} &= o(r^{-1/2}) && \text{as } r = \|x\| \rightarrow \infty\end{aligned}$$

flat, constant coefficient

1. *What data $g(x), h(x)$ will give us the solution to the original problem?*
2. *How to solve the interior problem? (Hint: PDE)*
3. *How to solve the exterior problem? (Hint: BIE)*

[Gillman, Barnett, Martinsson, 2015]

[Borges, Gillman, Greengard, 2017]

Coupling the interior and exterior Dirichlet-to-Neumann operators

For a second-order elliptic PDE, imposing continuity of $u(x)$ and its normal derivative across Ω will recover the original solution.

[Gillman, Barnett, Martinsson, 2015]
[Borges, Gillman, Greengard, 2017]

Coupling the interior and exterior Dirichlet-to-Neumann operators

For a second-order elliptic PDE, imposing continuity of $u(x)$ and its normal derivative across Ω will recover the original solution.

Define the ***Dirichlet-to-Neumann operators*** as the unique operators satisfying

$$T_{\text{int}} g = \frac{\partial u}{\partial n} \Big|_{\partial\Omega}, \quad T_{\text{ext}} h = \frac{\partial u^{\text{scat}}}{\partial n} \Big|_{\partial\Omega} \quad \forall g, h \in H^1(\partial\Omega)$$

[Gillman, Barnett, Martinsson, 2015]
[Borges, Gillman, Greengard, 2017]

Coupling the interior and exterior Dirichlet-to-Neumann operators

For a second-order elliptic PDE, imposing continuity of $u(x)$ and its normal derivative across Ω will recover the original solution.

Define the ***Dirichlet-to-Neumann operators*** as the unique operators satisfying

$$T_{\text{int}} g = \frac{\partial u}{\partial n} \Big|_{\partial\Omega}, \quad T_{\text{ext}} h = \frac{\partial u^{\text{scat}}}{\partial n} \Big|_{\partial\Omega} \quad \forall g, h \in H^1(\partial\Omega)$$

Then the continuity conditions imply

$$(T_{\text{int}} - T_{\text{ext}}) u^{\text{scat}}|_{\partial\Omega} = \underbrace{\frac{\partial u^{\text{inc}}}{\partial n} \Big|_{\partial\Omega} - T_{\text{int}} u^{\text{inc}}|_{\partial\Omega}}_{\text{known}}$$

[Gillman, Barnett, Martinsson, 2015]
[Borges, Gillman, Greengard, 2017]

Coupling the interior and exterior Dirichlet-to-Neumann operators

For a second-order elliptic PDE, imposing continuity of $u(x)$ and its normal derivative across Ω will recover the original solution.

Define the ***Dirichlet-to-Neumann operators*** as the unique operators satisfying

$$T_{\text{int}} g = \frac{\partial u}{\partial n} \Big|_{\partial\Omega}, \quad T_{\text{ext}} h = \frac{\partial u^{\text{scat}}}{\partial n} \Big|_{\partial\Omega} \quad \forall g, h \in H^1(\partial\Omega)$$

Then the continuity conditions imply

$$(T_{\text{int}} - T_{\text{ext}}) u^{\text{scat}}|_{\partial\Omega} = \underbrace{\frac{\partial u^{\text{inc}}}{\partial n} \Big|_{\partial\Omega} - T_{\text{int}} u^{\text{inc}}|_{\partial\Omega}}_{\text{known}}$$

Solving this linear system will tell us the scattered field on the boundary.

[Gillman, Barnett, Martinsson, 2015]
[Borges, Gillman, Greengard, 2017]

Exterior problem

Boundary integral formulation and chunkie

$$\begin{aligned}\Delta u^{\text{scat}}(x) + k^2 u^{\text{scat}}(x) &= 0 && \text{in } \mathbb{R}^2 \setminus \Omega \\ u^{\text{scat}}(x) &= h(x) && \text{on } \partial\Omega \\ \frac{\partial u^{\text{scat}}}{\partial r} - ik u^{\text{scat}} &= o(r^{-1/2}) && \text{as } r = \|x\| \rightarrow \infty\end{aligned}$$

Exterior problem

Boundary integral formulation and chunkie

$$\begin{aligned}\Delta u^{\text{scat}}(x) + k^2 u^{\text{scat}}(x) &= 0 && \text{in } \mathbb{R}^2 \setminus \Omega \\ u^{\text{scat}}(x) &= h(x) && \text{on } \partial\Omega \\ \frac{\partial u^{\text{scat}}}{\partial r} - ik u^{\text{scat}} &= o(r^{-1/2}) && \text{as } r = \|x\| \rightarrow \infty\end{aligned}$$

As we have a constant-coefficient problem in flat space, a BIE-based discretization is a good choice.

Exterior problem

Boundary integral formulation and chunkie

$$\begin{aligned}\Delta u^{\text{scat}}(x) + k^2 u^{\text{scat}}(x) &= 0 && \text{in } \mathbb{R}^2 \setminus \Omega \\ u^{\text{scat}}(x) &= h(x) && \text{on } \partial\Omega \\ \frac{\partial u^{\text{scat}}}{\partial r} - ik u^{\text{scat}} &= o(r^{-1/2}) && \text{as } r = \|x\| \rightarrow \infty\end{aligned}$$

As we have a constant-coefficient problem in flat space, a BIE-based discretization is a good choice.

Green's representation formula: $\frac{1}{2} u^{\text{scat}}(x) = \mathcal{D}[u^{\text{scat}}|_{\partial\Omega}](x) - \mathcal{S}\left[\frac{\partial u^{\text{scat}}}{\partial n}|_{\partial\Omega}\right](x) \quad \forall x \in \mathbb{R}^2 \setminus \Omega$

Helmholtz Green's function

$$\mathcal{S}[\sigma](x) = \int_{\partial\Omega} G(x, y) \sigma(y) ds(y) \quad \text{"Single layer potential"}$$
$$\mathcal{D}[\sigma](x) = \int_{\partial\Omega} \frac{\partial G(x, y)}{\partial n_y} \sigma(y) ds(y) \quad \text{"Double layer potential"}$$

Exterior problem

Boundary integral formulation and chunkie

$$\begin{aligned}\Delta u^{\text{scat}}(x) + k^2 u^{\text{scat}}(x) &= 0 && \text{in } \mathbb{R}^2 \setminus \Omega \\ u^{\text{scat}}(x) &= h(x) && \text{on } \partial\Omega \\ \frac{\partial u^{\text{scat}}}{\partial r} - ik u^{\text{scat}} &= o(r^{-1/2}) && \text{as } r = \|x\| \rightarrow \infty\end{aligned}$$

As we have a constant-coefficient problem in flat space, a BIE-based discretization is a good choice.

Green's representation formula: $\frac{1}{2} u^{\text{scat}}(x) = \mathcal{D}[u^{\text{scat}}|_{\partial\Omega}](x) - \mathcal{S}\left[\frac{\partial u^{\text{scat}}}{\partial n}|_{\partial\Omega}\right](x) \quad \forall x \in \mathbb{R}^2 \setminus \Omega$

Helmholtz Green's function

$$\mathcal{S}[\sigma](x) = \int_{\partial\Omega} G(x, y) \sigma(y) ds(y) \quad \text{"Single layer potential"}$$
$$\mathcal{D}[\sigma](x) = \int_{\partial\Omega} \frac{\partial G(x, y)}{\partial n_y} \sigma(y) ds(y) \quad \text{"Double layer potential"}$$

Hence, $\frac{\partial u^{\text{scat}}}{\partial n}|_{\partial\Omega} = \mathcal{S}^{-1}(\mathcal{D} - \frac{1}{2}I) u^{\text{scat}}|_{\partial\Omega}$ and so: $T_{\text{ext}} = \mathcal{S}^{-1}(\mathcal{D} - \frac{1}{2}I)$

Exterior problem

Boundary integral formulation and chunkie

- Define the Helmholtz single- and double-layer kernels:

```
Skern = kernel('helmholtz', 's', k);  
Dkern = kernel('helmholtz', 'd', k);
```

Exterior problem

Boundary integral formulation and chunkie

- Define the Helmholtz single- and double-layer kernels:

```
Skern = kernel('helmholtz', 's', k);  
Dkern = kernel('helmholtz', 'd', k);
```

- Construct matrices which discretize the layer potentials on the boundary:

```
S = chunkermat(0mega, Skern);  
D = chunkermat(0mega, Dkern);  
I = eye(0mega.npt);
```

Exterior problem

Boundary integral formulation and chunkie

- Define the Helmholtz single- and double-layer kernels:

```
Skern = kernel('helmholtz', 's', k);  
Dkern = kernel('helmholtz', 'd', k);
```

- Construct matrices which discretize the layer potentials on the boundary:

```
S = chunkermat(0mega, Skern);  
D = chunkermat(0mega, Dkern);  
I = eye(0mega.npt);
```

- The exterior DtN map is then given by linear algebra:

```
DtN_ext = S \ (D - I/2);
```

Interior problem

Domain decomposition and surfacefun

$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= 0 && \text{in } \Omega \\ u(x) &= g(x) && \text{on } \partial\Omega\end{aligned}$$

Interior problem

Domain decomposition and surfacefun

$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= 0 && \text{in } \Omega \\ u(x) &= g(x) && \text{on } \partial\Omega\end{aligned}$$

As we have a variable-coefficient problem on a surface, a PDE-based discretization is a good choice.

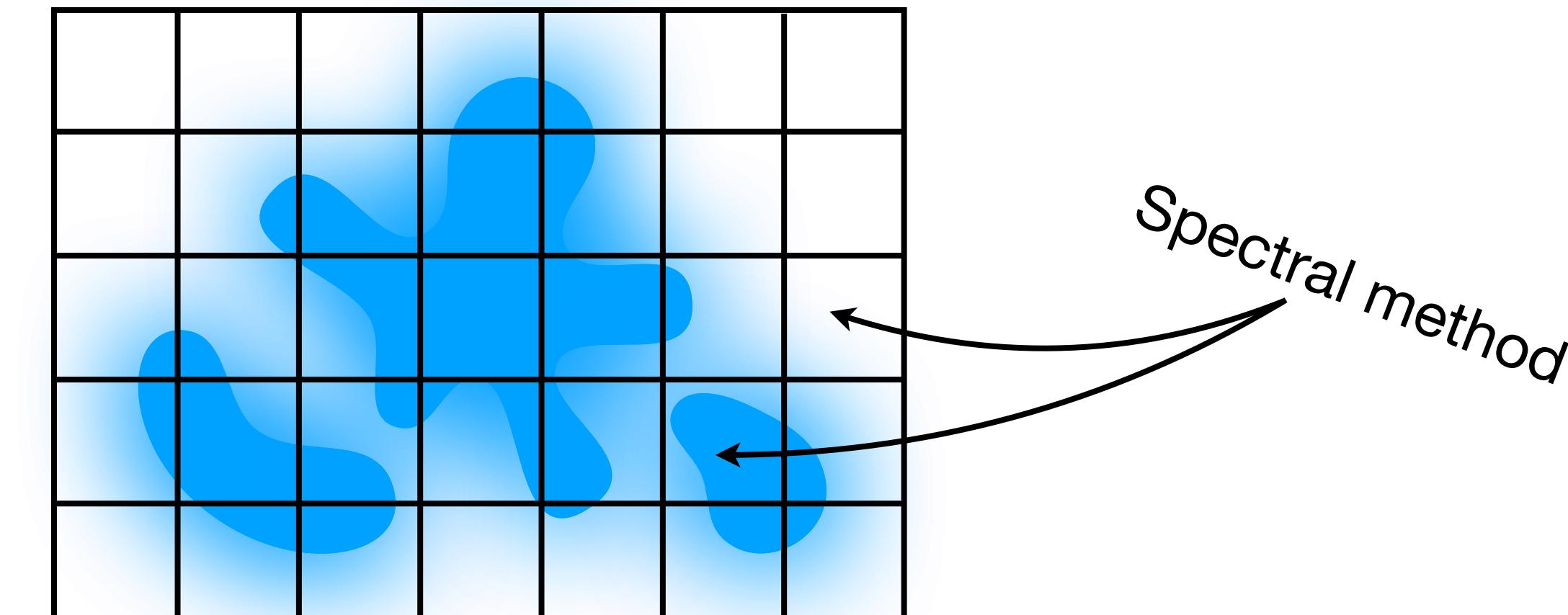
Interior problem

Domain decomposition and surfacefun

$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= 0 && \text{in } \Omega \\ u(x) &= g(x) && \text{on } \partial\Omega\end{aligned}$$

As we have a variable-coefficient problem on a surface, a PDE-based discretization is a good choice.

We will use a multidomain spectral method for discretization.



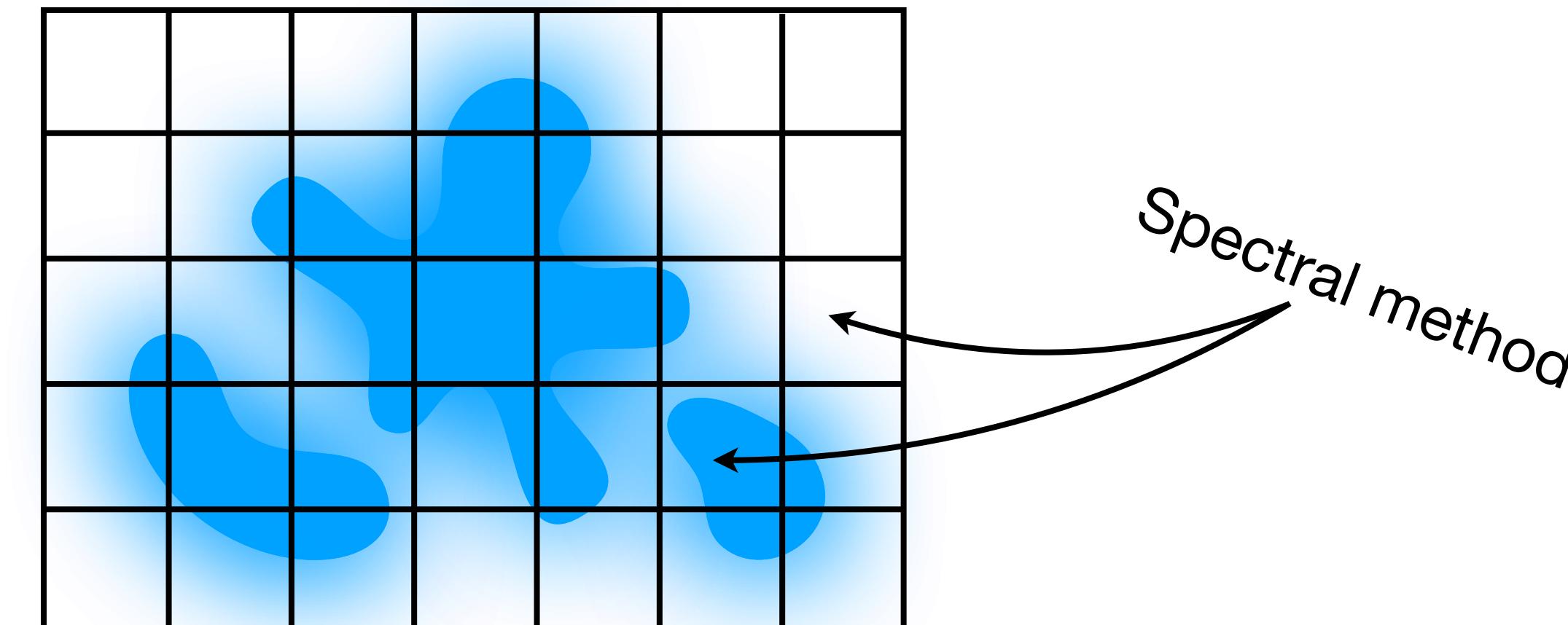
Interior problem

Domain decomposition and `surfacefun`

$$\begin{aligned}\Delta_\Gamma u(x) + k^2(1 - q(x))u(x) &= 0 && \text{in } \Omega \\ u(x) &= g(x) && \text{on } \partial\Omega\end{aligned}$$

As we have a variable-coefficient problem on a surface, a PDE-based discretization is a good choice.

We will use a multidomain spectral method for discretization.



`surfacefun` provides abstractions for computing with functions on surfaces, along with a fast direct solver for surface PDEs using domain decomposition.

Interior problem

Domain decomposition and surfacefun

Interior problem

Domain decomposition and surfacefun

- Define the PDE and geometry

```
pdo = [];  
pdo.lap = 1;  
pdo.c = @(x,y,z) k^2*(1-q(x,y,z));  
dom = surfacemesh(x, y, z);
```

Interior problem

Domain decomposition and surfacefun

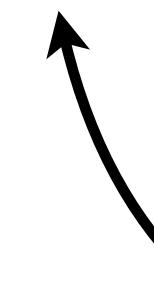
- Define the PDE and geometry

```
pdo = []; pdo.lap = 1; pdo.c = @(x,y,z) k^2*(1-q(x,y,z));  
dom = surfacemesh(x, y, z);
```

- Construct the fast direct solver:

```
L = surfaceop(dom, pdo, method='ItI');
```

In practice, we use “impedance-to-impedance” maps instead of Dirichlet-to-Neumann maps to avoid artificial resonances.



Interior problem

Domain decomposition and surfacefun

- Define the PDE and geometry

```
pdo = []; pdo.lap = 1; pdo.c = @(x,y,z) k^2*(1-q(x,y,z));  
dom = surfacemesh(x, y, z);
```

- Construct the fast direct solver:

```
L = surfaceop(dom, pdo, method='ItI');
```

- The interior DtN map is then computed:

```
DtN_int = L.DtN();
```

In practice, we use “impedance-to-impedance” maps instead of Dirichlet-to-Neumann maps to avoid artificial resonances.



Putting it all together

chunkie and surfacefun

Now that we know Dirichlet-to-Neumann operators for the interior and exterior, we can solve the coupled problem.

Demo