A fully adaptive Poisson solver for smooth two-dimensional domains

Alex Barnett

Daniel Fortunato

Flatiron Institute

David Stein

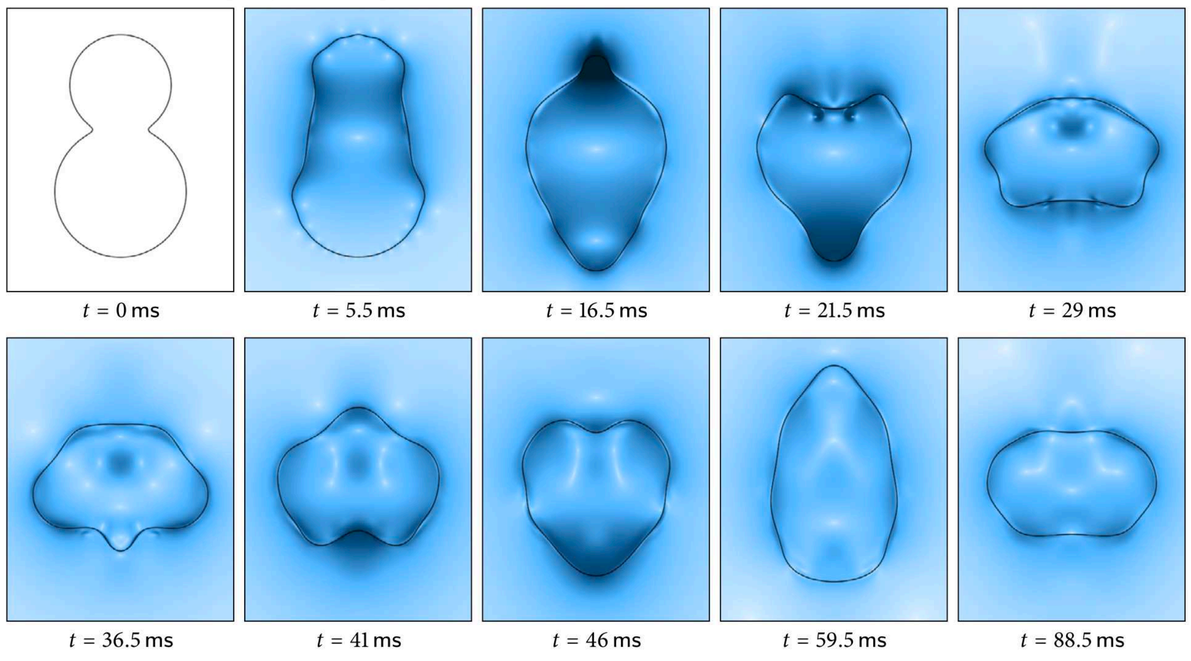
Introduction
Inhomogeneous elliptic BVP

Many applications involve solving an inhomogeneous elliptic BVP.

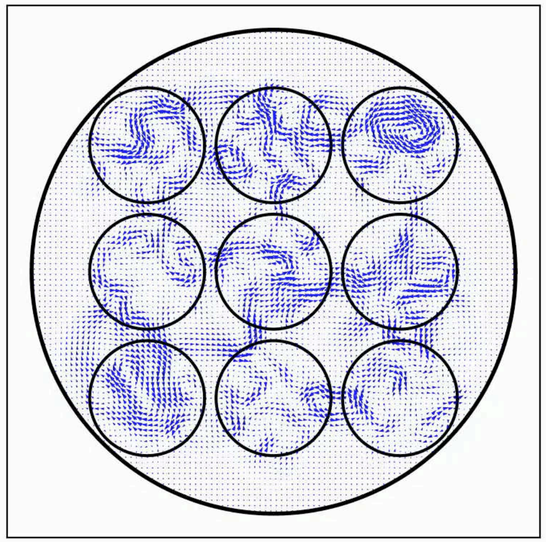
$$Lu = f \quad \text{in } \Omega \qquad \text{(e.g. Poisson, Helmholtz, Stokes, …)}$$
$$Bu = g \quad \text{on } \Gamma \qquad \text{(e.g. Dirichlet, Neumann, Robin, …)}$$

(Today's focus: $L = \Delta$, interior Dirichlet)

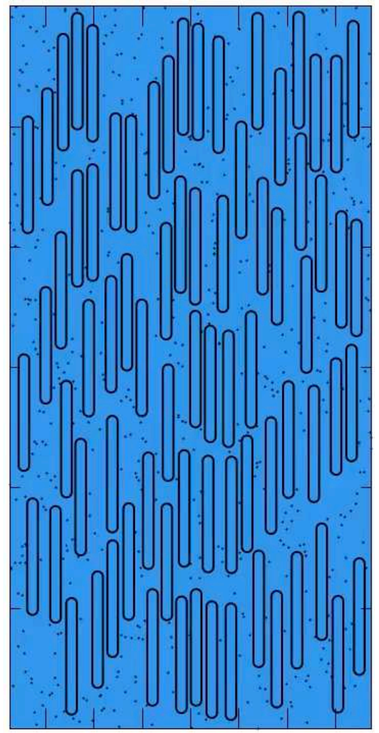
Bubble collision [Saye, 2017]



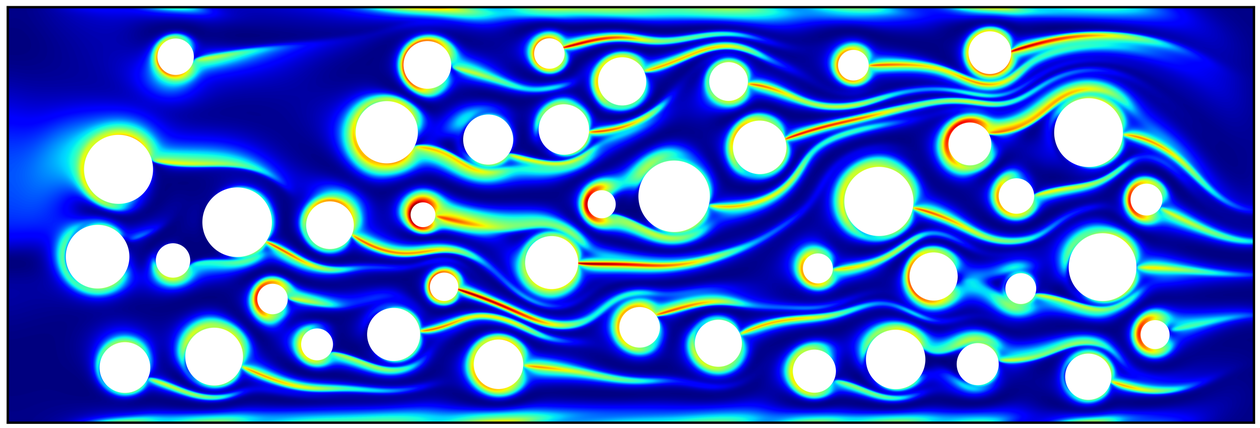
Active droplets [Stein, 2021]



Fluid-structure interaction [Rycroft et al., 2020]



Non-Newtonian fluids [Stein et al., 2019]



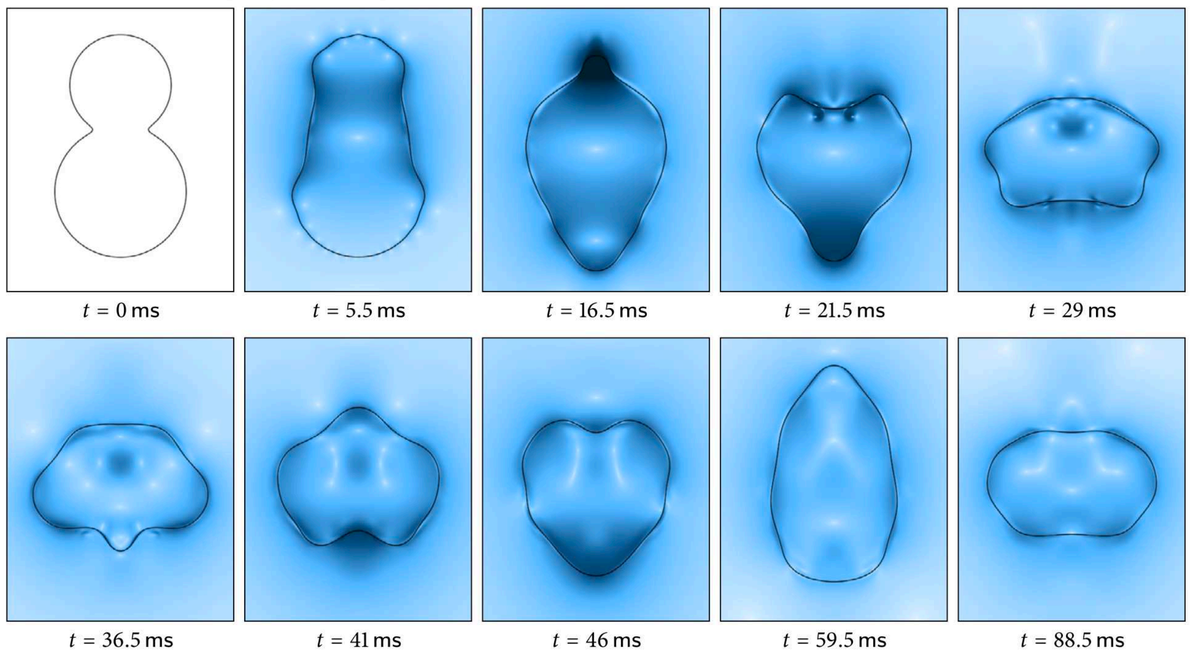
Introduction
Inhomogeneous elliptic BVP

Many applications involve solving an inhomogeneous elliptic BVP.

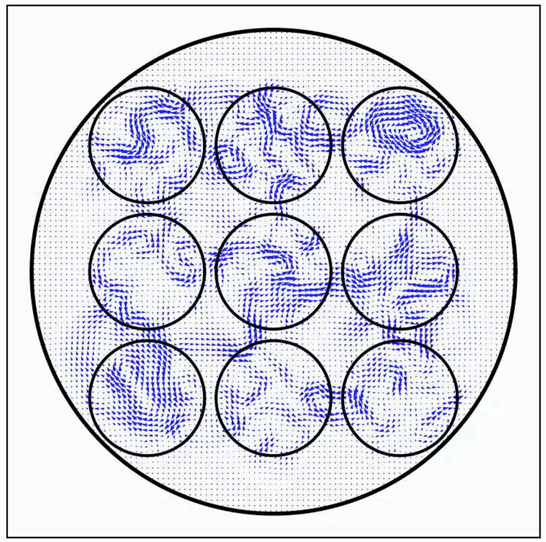
$$Lu = f \quad \text{in } \Omega \qquad \text{(e.g. Poisson, Helmholtz, Stokes, …)}$$
$$Bu = g \quad \text{on } \Gamma \qquad \text{(e.g. Dirichlet, Neumann, Robin, …)}$$

(Today's focus: $L = \Delta$, interior Dirichlet)

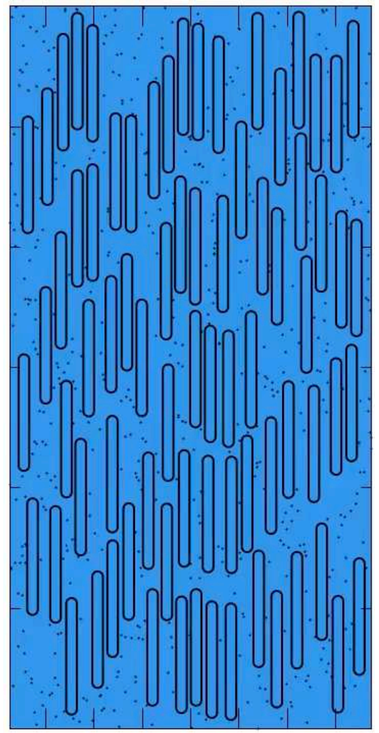
Bubble collision [Saye, 2017]



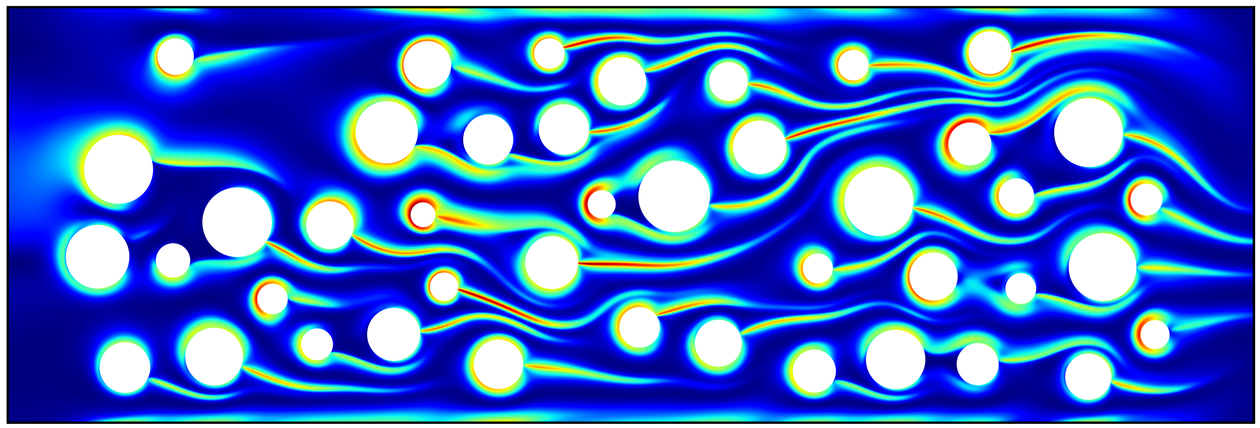
Active droplets [Stein, 2021]



Fluid-structure interaction [Rycroft et al., 2020]

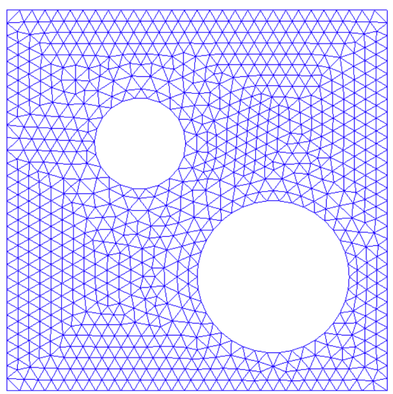


Non-Newtonian fluids [Stein et al., 2019]

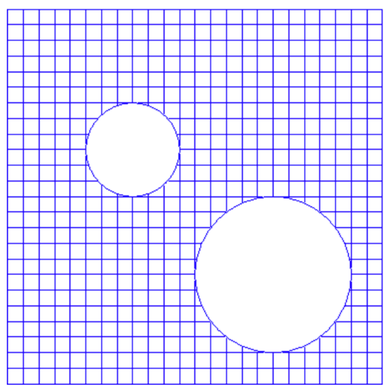


Introduction
Many approaches for inhomogeneous BVP

Traditional



Conforming



Cut-cell

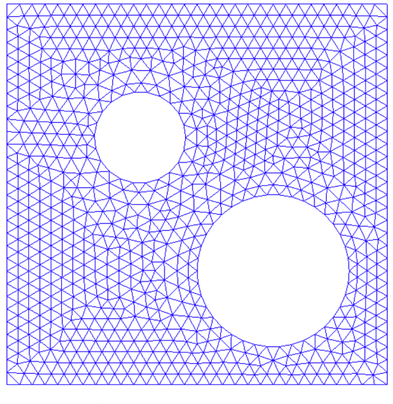
- Mesh generation (or cut-cell generation)
- Directly discretize with FEM, FDM, SEM, …

$$Ax = b$$

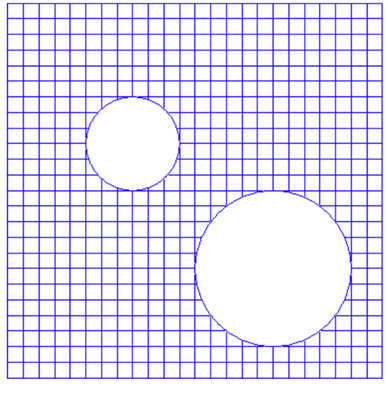
Solve linear system for volume DoFs.
Sparse but preconditioning needed.

Introduction
Many approaches for inhomogeneous BVP

Traditional



Conforming



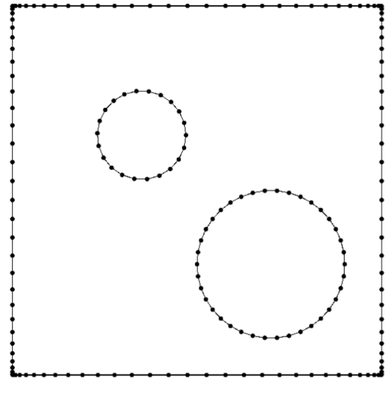
Cut-cell

- Mesh generation (or cut-cell generation)
- Directly discretize with FEM, FDM, SEM, …

$$Ax = b$$

Solve linear system for volume DoFs.
Sparse but preconditioning needed.

Analysis-based



- Write solution as $u = u^{\mathrm{h}} + u^{\mathrm{p}}$

homogeneous solution *particular solution*

- Find **some** (any!) function u^{p} such that

$$Lu^{\mathrm{p}} = f \quad \text{in } \Omega$$

No solve required. Convolve with Green's function.

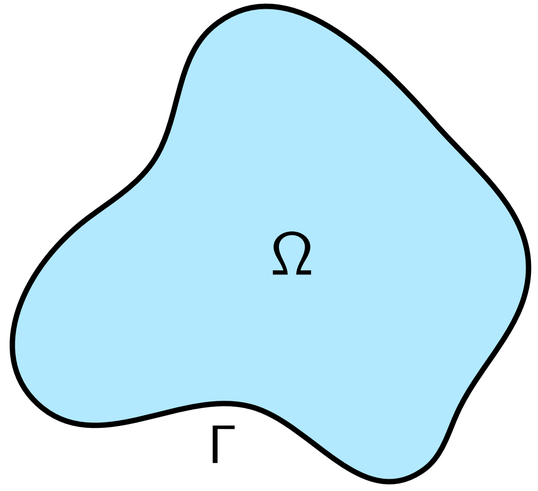
- Compute u^{h} to satisfy boundary conditions:

$$\begin{aligned} Lu^{\mathrm{h}} &= 0 && \text{in } \Omega \\ Bu^{\mathrm{h}} &= g - Bu^{\mathrm{p}} && \text{on } \Gamma \end{aligned}$$

Solve linear system for boundary DoFs using BIE.

Introduction
Computing a particular solution

Suppose we only know $f(x)$ inside Ω. We have a few options…



Introduction
Computing a particular solution

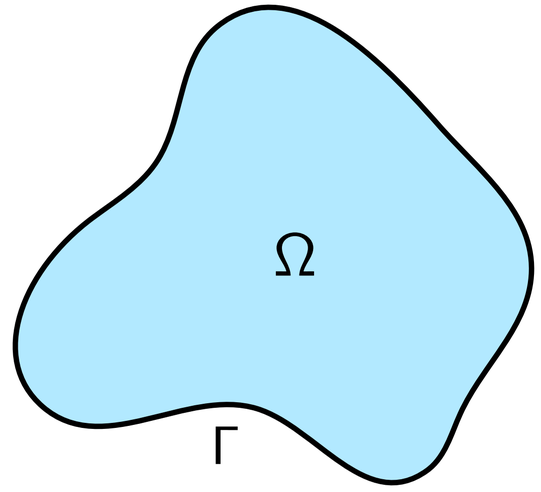
Suppose we only know $f(x)$ inside Ω. We have a few options…

- Build accurate quadrature scheme over Ω (e.g. adaptive boxes with cut cells near boundary) and compute

$$u^{\mathrm{p}}(x) = \int_{\Omega} G(x,y) f(y) dy$$

free space fundamental solution

$$\left(\text{Poisson: } G(x,y) = \frac{1}{2\pi} \log \frac{1}{|x-y|} \right)$$



Introduction
Computing a particular solution

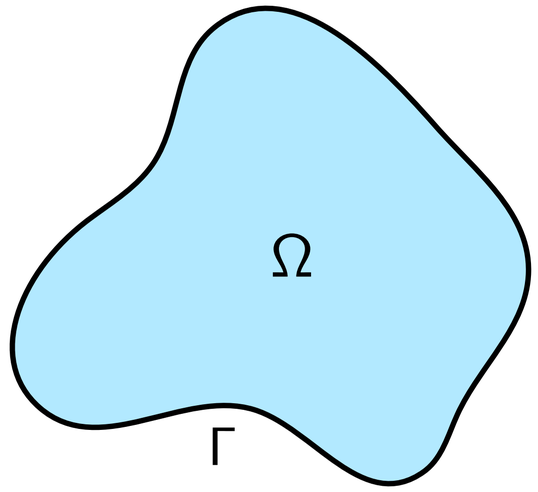
Suppose we only know $f(x)$ inside Ω. We have a few options…

- Build accurate quadrature scheme over Ω (e.g. adaptive boxes with cut cells near boundary) and compute

$$u^{\mathrm{p}}(x) = \int_{\Omega} G(x,y) f(y) dy$$

free space fundamental solution

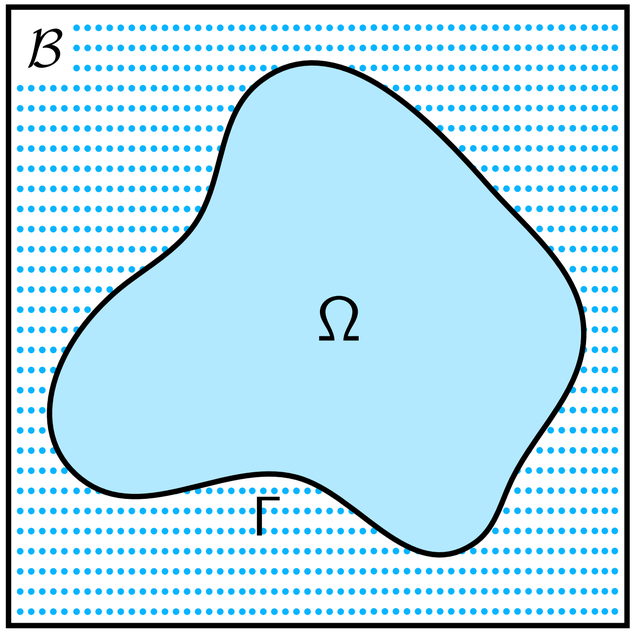
$$\left(\text{Poisson: } G(x,y) = \frac{1}{2\pi} \log \frac{1}{|x-y|} \right)$$



- Extend f to $\tilde{f}$ outside Ω ("function extension"). Adaptively resolve $\tilde{f}$ and compute

$$u^{\mathrm{p}}(x) = \int_{\mathcal{B}} G(x,y) \tilde{f}(y) dy$$

Quadrature tables for boxes can be **precomputed** (FMM "box code")



Introduction
Computing a particular solution

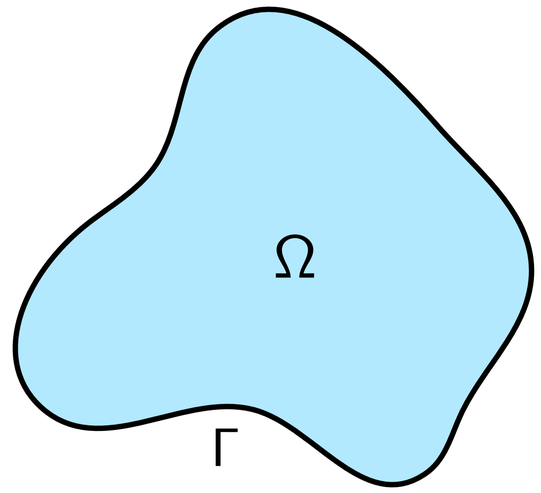
Suppose we only know $f(x)$ inside Ω. We have a few options…

- Build accurate quadrature scheme over Ω (e.g. adaptive boxes with cut cells near boundary) and compute

$$u^{\mathrm{p}}(x) = \int_{\Omega} G(x,y) f(y) dy$$

free space fundamental solution

$$\left(\text{Poisson: } G(x,y) = \frac{1}{2\pi} \log \frac{1}{|x-y|} \right)$$

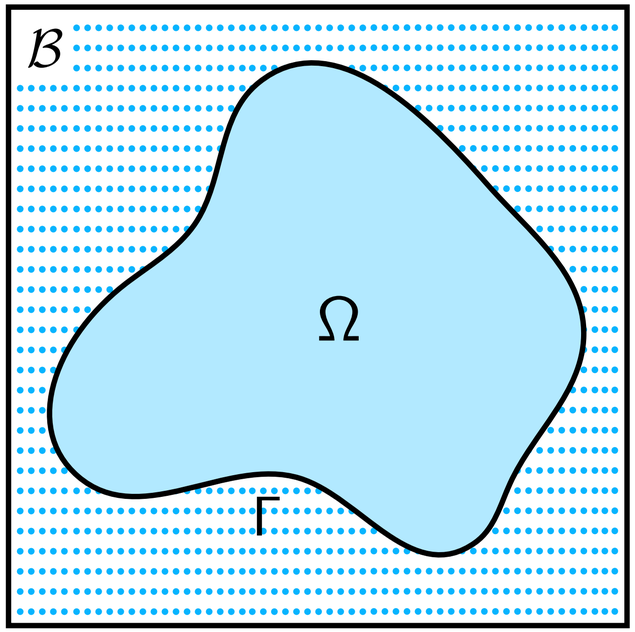


- Extend f to $\tilde{f}$ outside Ω ("function extension"). Adaptively resolve $\tilde{f}$ and compute

$$u^{\mathrm{p}}(x) = \int_{\mathcal{B}} G(x,y) \tilde{f}(y) dy$$

Quadrature tables for boxes can be **precomputed** (FMM "box code")

- Or, sample $\tilde{f}$ on a uniform grid and use an FFT-based solver on $\mathcal{B}$.



Introduction
Computing a particular solution

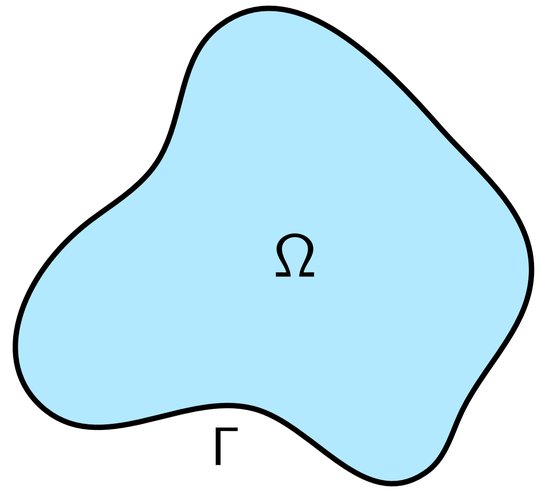
Suppose we only know $f(x)$ inside Ω. We have a few options…

- Build accurate quadrature scheme over Ω (e.g. adaptive boxes with cut cells near boundary) and compute

$$u^{\mathrm{p}}(x) = \int_{\Omega} G(x,y) f(y) dy$$

free space fundamental solution

$$\left(\text{Poisson: } G(x,y) = \frac{1}{2\pi} \log \frac{1}{|x-y|} \right)$$



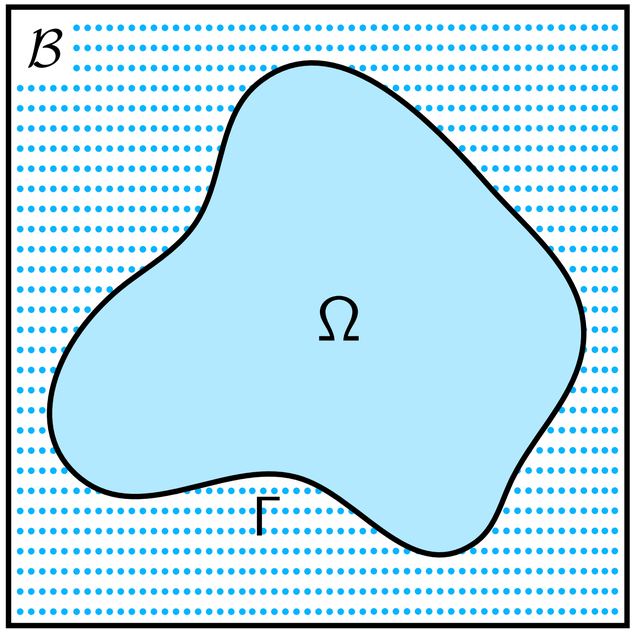
- Extend f to $\tilde{f}$ outside Ω ("function extension"). Adaptively resolve $\tilde{f}$ and compute

$$u^{\mathrm{p}}(x) = \int_{\mathcal{B}} G(x,y) \tilde{f}(y) dy$$

Quadrature tables for boxes can be **precomputed** (FMM "box code")

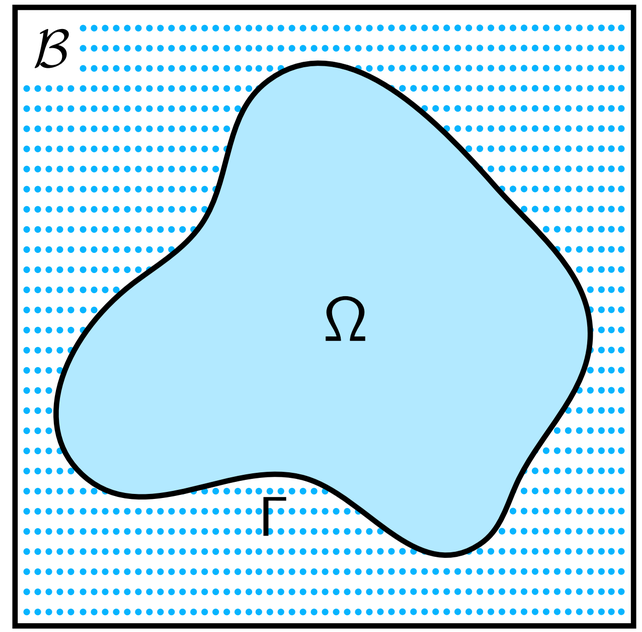
- Or, sample $\tilde{f}$ on a uniform grid and use an FFT-based solver on $\mathcal{B}$.

Want $\tilde{f}$ as smooth as f for fast convergence. How?



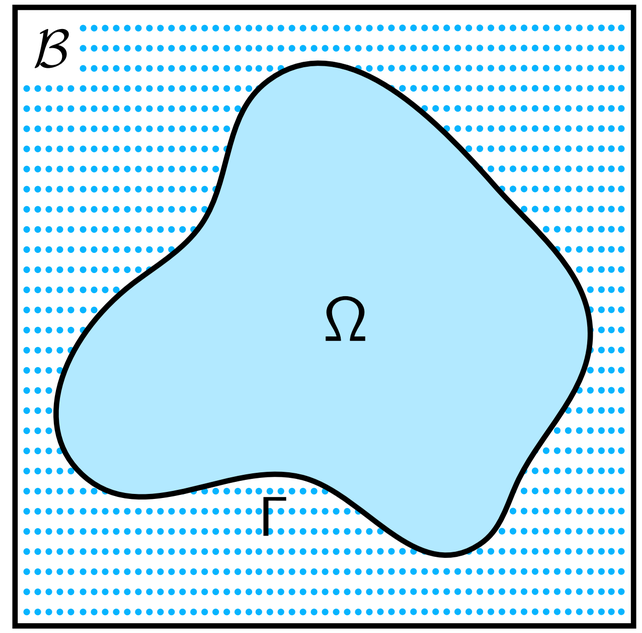
Function extension
Prior work

- Finite difference extension, second-order accurate [Mayo, 1984]
- Fourier continuation [Bruno & Lyon, 2010], [Bruno & Paul, 2020]
- Immersed boundary smooth extension [Stein, Guy, & Thomases, 2015]
- C^k polyharmonic extension + box code, fourth-order accurate [Askham & Cerfon, 2017]
- Partition of unity extension [Fryklund, Lehto, & Tornberg, 2018]

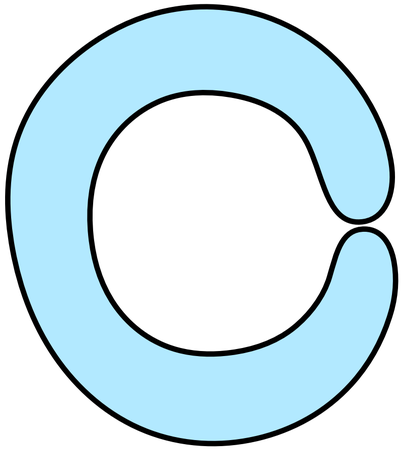


Function extension
Prior work

- Finite difference extension, second-order accurate [Mayo, 1984]
- Fourier continuation [Bruno & Lyon, 2010], [Bruno & Paul, 2020]
- Immersed boundary smooth extension [Stein, Guy, & Thomases, 2015]
- C^k polyharmonic extension + box code, fourth-order accurate [Askham & Cerfon, 2017]
- Partition of unity extension [Fryklund, Lehto, & Tornberg, 2018]

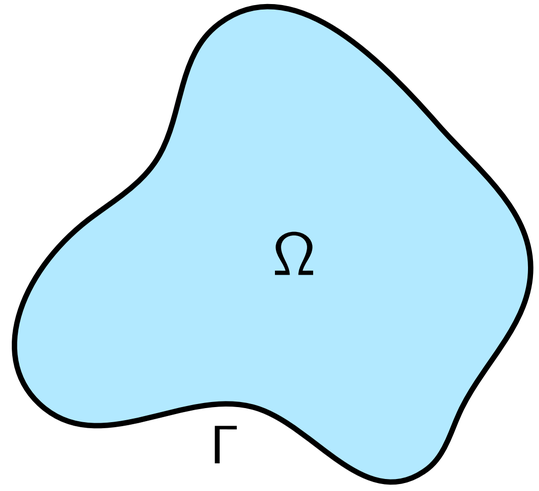


In general, smoothly extending f in a robust way is challenging.
(Especially for multiscale geometry, multiscale f, close-to-touching regions, …)



Function “intension”
Our approach

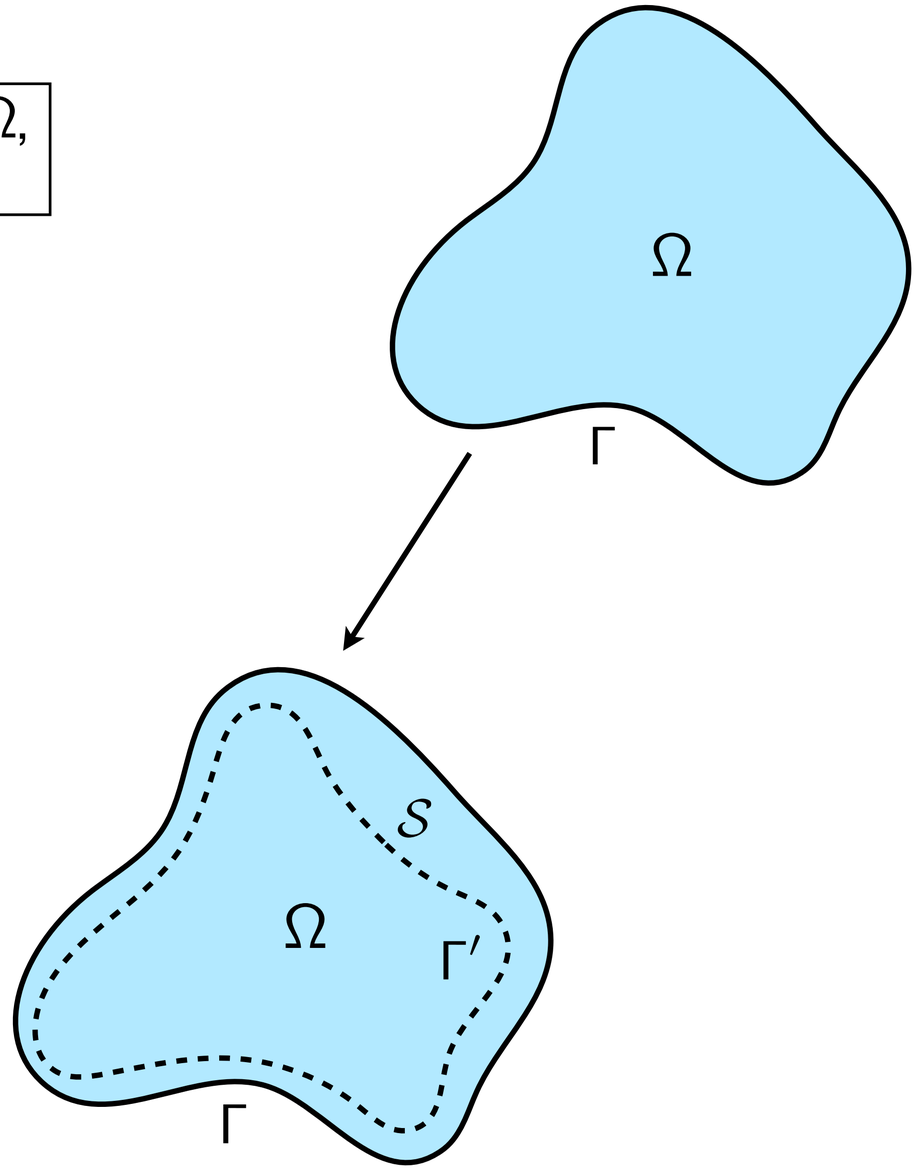
Instead of trying to make $\tilde{f}$ smooth outside Ω,
let’s make it smooth **inside** Ω.



Function “intension”
Our approach

Instead of trying to make $\tilde{f}$ smooth outside Ω,
let’s make it smooth **inside** Ω.

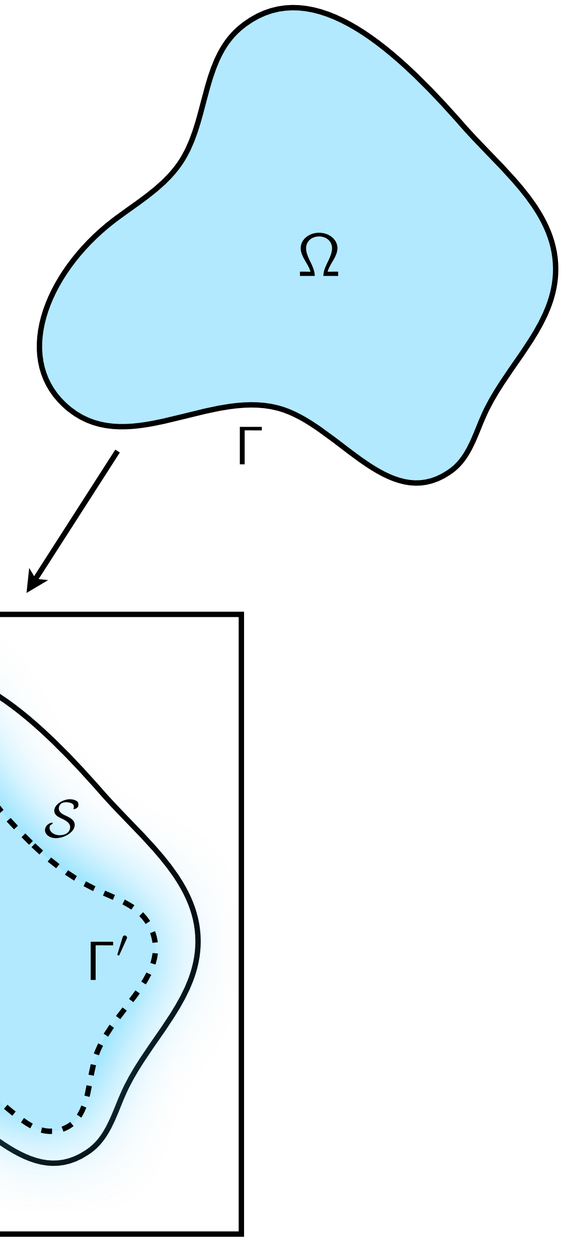
- Define an annular strip $\mathcal{S}$ inside Ω.



Function “intension”
Our approach

Instead of trying to make $\tilde{f}$ smooth outside Ω,
let’s make it smooth **inside** Ω.

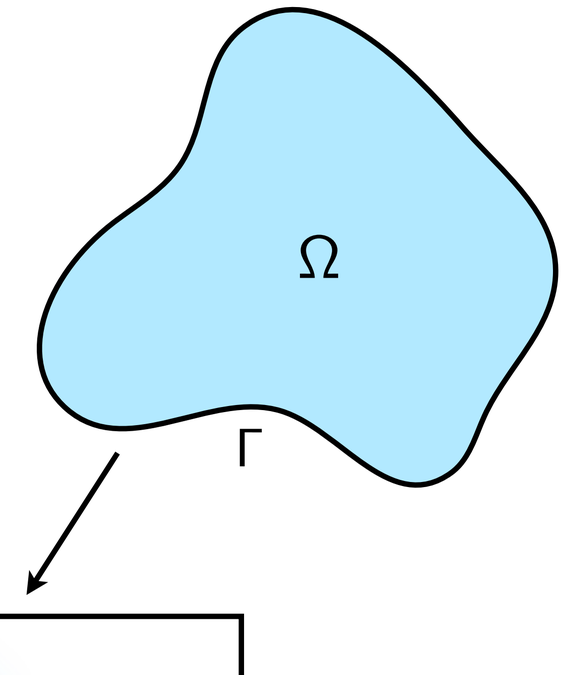
- Define an annular strip $\mathcal{S}$ inside Ω.
- Roll off f to zero smoothly in $\mathcal{S}$. This is $\tilde{f}$.

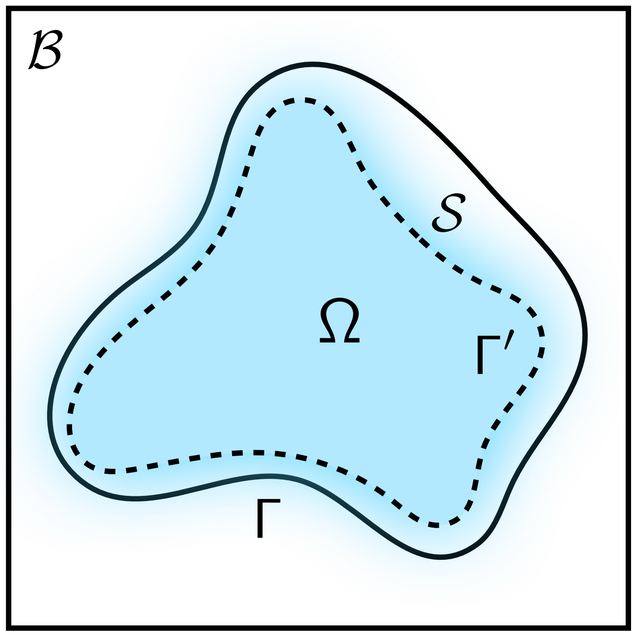


Function “intension”
Our approach

Instead of trying to make $\tilde{f}$ smooth outside Ω,
let’s make it smooth **inside** Ω.

- Define an annular strip $\mathcal{S}$ inside Ω.
- Roll off f to zero smoothly in $\mathcal{S}$. This is $\tilde{f}$.
- Compute a particular solution for $\tilde{f}$ in $\mathcal{B}$.

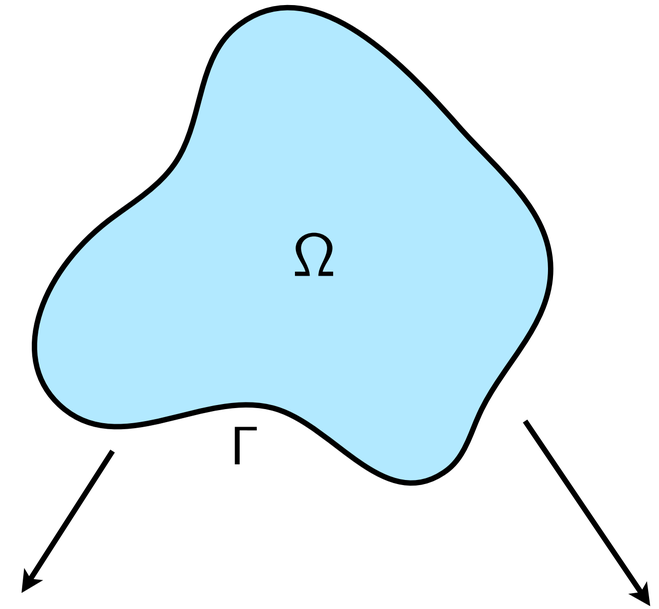


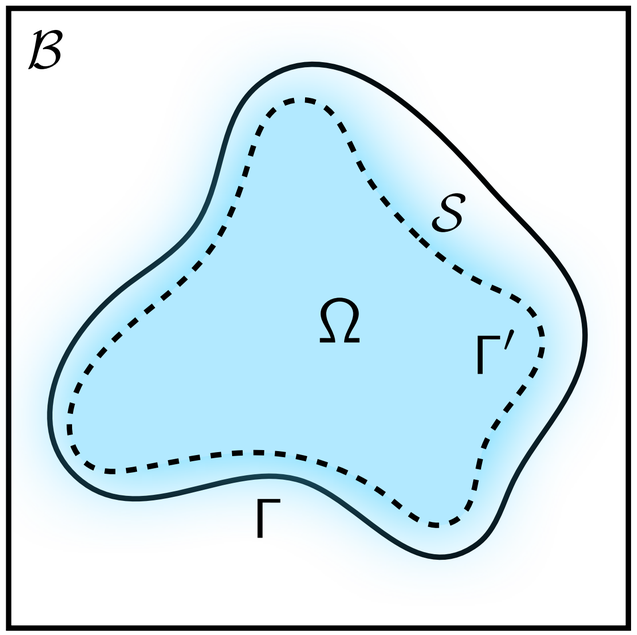


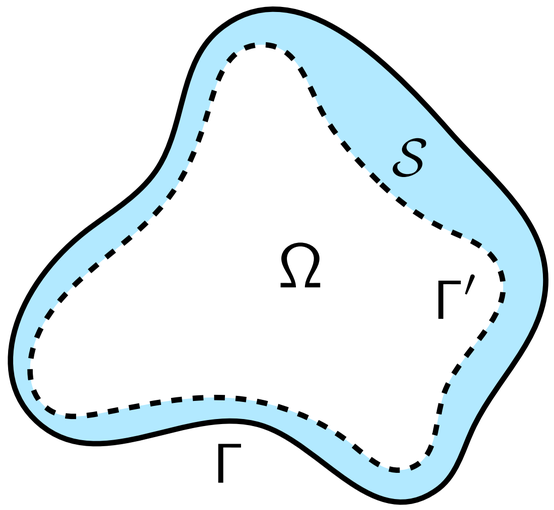
Function “intension”
Our approach

Instead of trying to make $\tilde{f}$ smooth outside Ω,
let’s make it smooth **inside** Ω.

- Define an annular strip $\mathcal{S}$ inside Ω.
- Roll off f to zero smoothly in $\mathcal{S}$. This is $\tilde{f}$.
- Compute a particular solution for $\tilde{f}$ in $\mathcal{B}$.
- Compute a particular solution for f in $\mathcal{S}$.



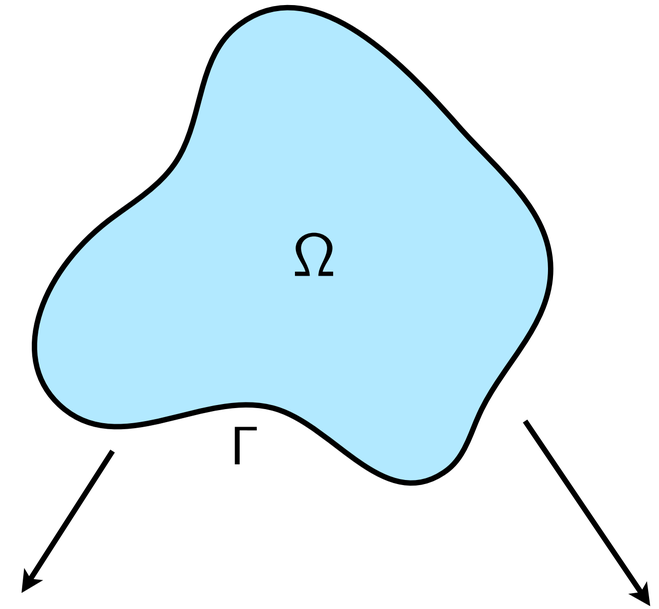


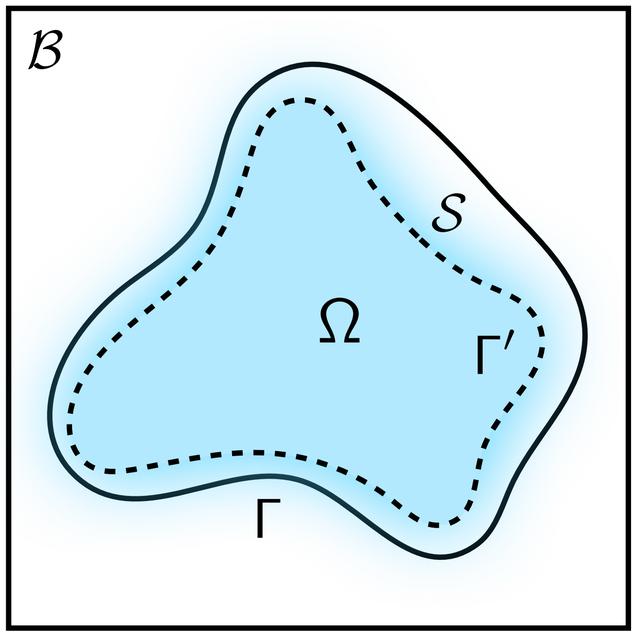


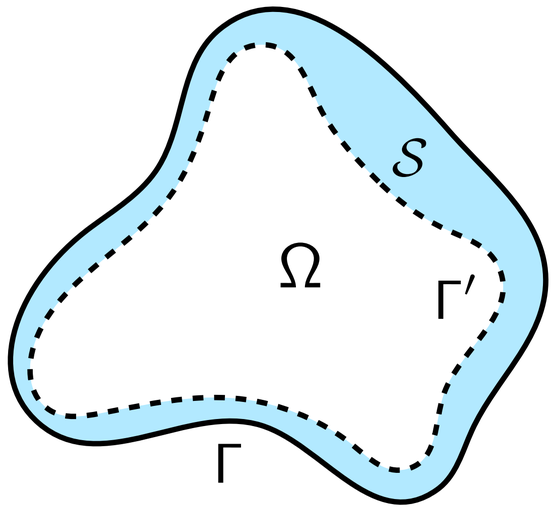
Function “intension”
Our approach

Instead of trying to make $\tilde{f}$ smooth outside Ω,
let’s make it smooth **inside** Ω.

- Define an annular strip $\mathcal{S}$ inside Ω.
- Roll off f to zero smoothly in $\mathcal{S}$. This is $\tilde{f}$.
- Compute a particular solution for $\tilde{f}$ in $\mathcal{B}$.
- Compute a particular solution for f in $\mathcal{S}$.
- Patch solutions together.







Function “intension”

Our approach

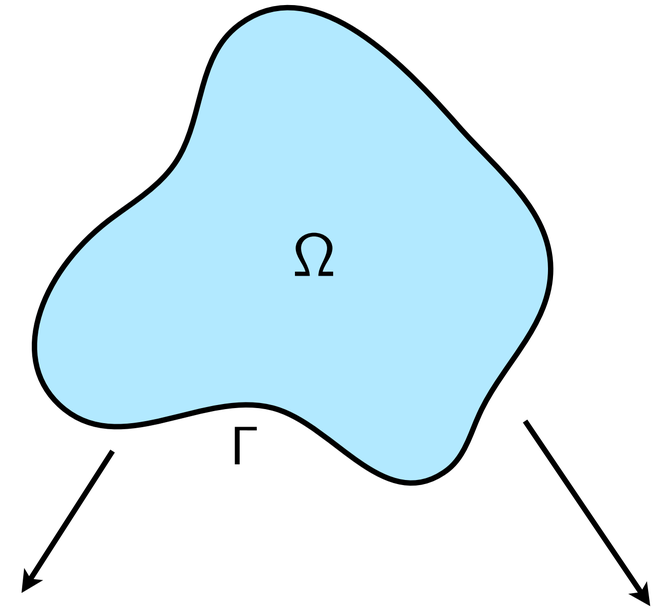
Instead of trying to make $\tilde{f}$ smooth outside Ω, let’s make it smooth **inside** Ω.

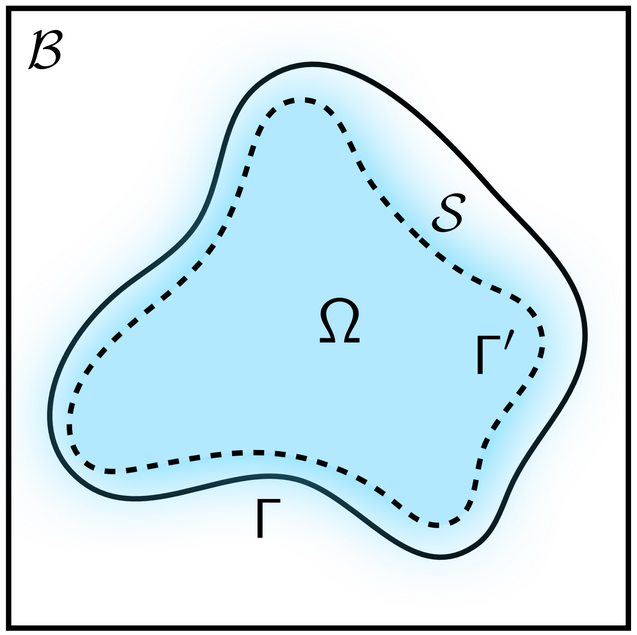
- Define an annular strip $\mathcal{S}$ inside Ω.
- Roll off f to zero smoothly in $\mathcal{S}$. This is $\tilde{f}$.
- Compute a particular solution for $\tilde{f}$ in $\mathcal{B}$.
- Compute a particular solution for f in $\mathcal{S}$.
- Patch solutions together.

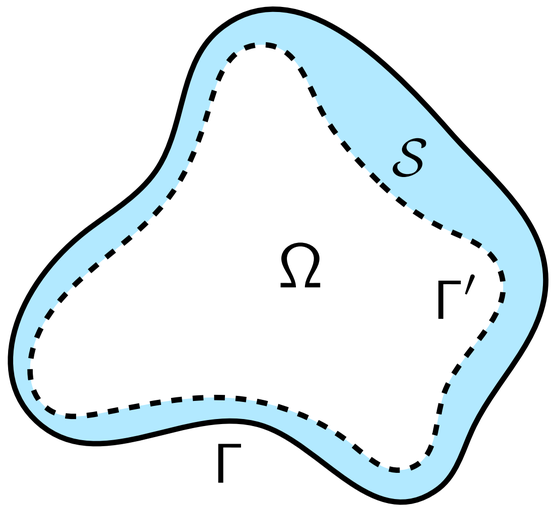
How to define the strip?

How to solve in the strip?

How to patch the solutions?







Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

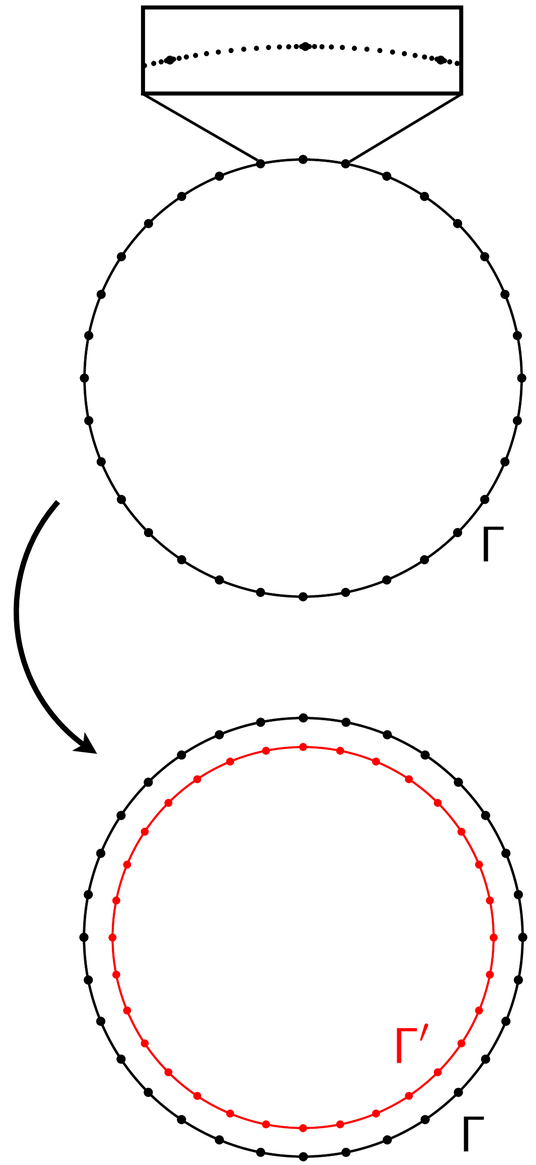
- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

- What about a uniform perturbation in the normal direction?

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

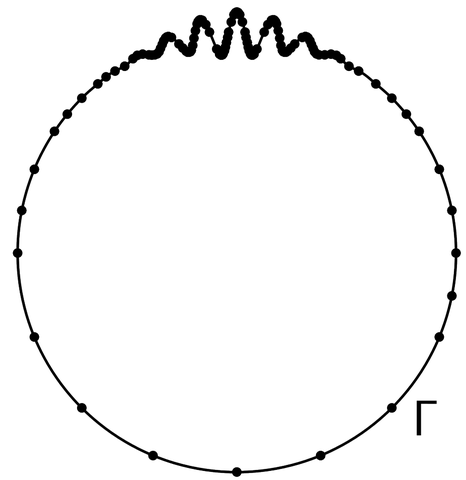


- What about a uniform perturbation in the normal direction?
  - When all panels are roughly the same size, works well.

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

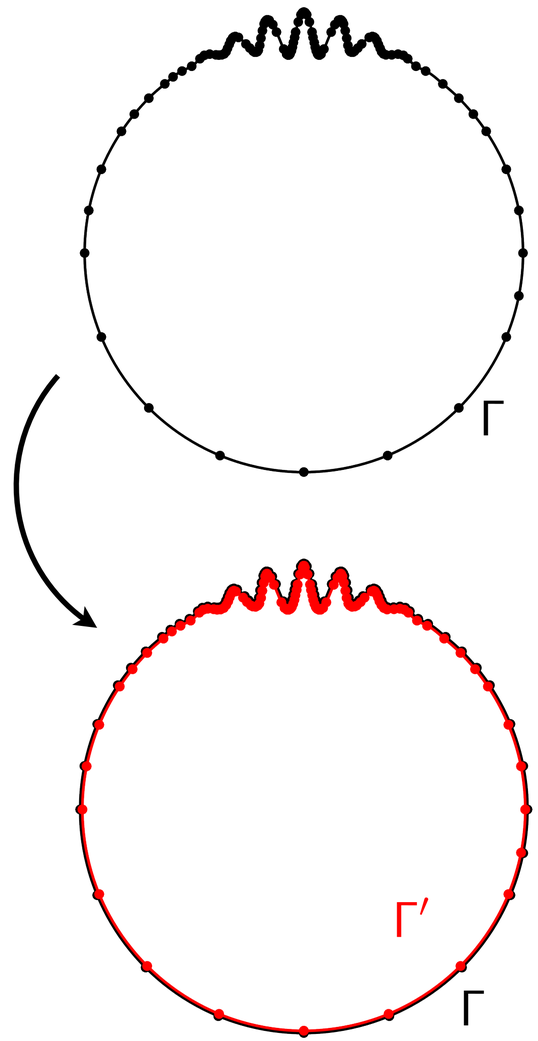


- What about a uniform perturbation in the normal direction?
  - When all panels are roughly the same size, works well.
  - But when panels span many length scales…

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

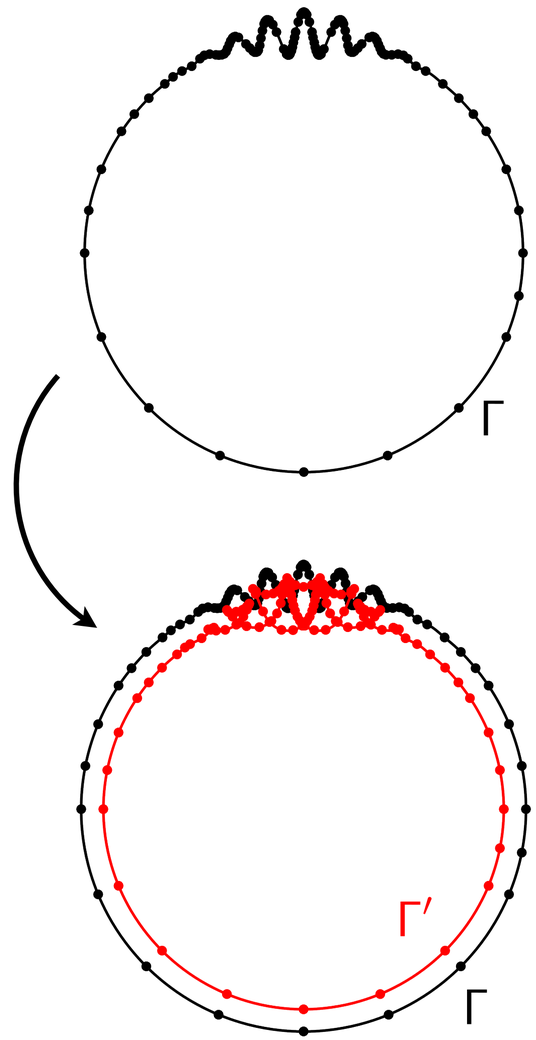


- What about a uniform perturbation in the normal direction?
  - When all panels are roughly the same size, works well.
  - But when panels span many length scales…
    - can over-resolve the largest length scales.

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

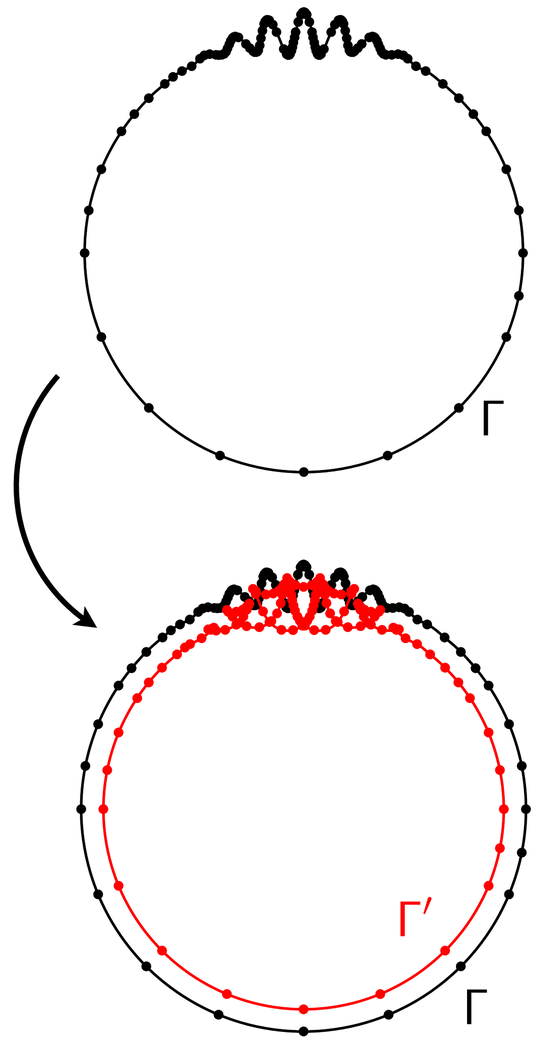


- What about a uniform perturbation in the normal direction?
  - When all panels are roughly the same size, works well.
  - But when panels span many length scales…
    - can over-resolve the largest length scales.
    - can self-intersect.

Defining the strip
Wish list

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.



- What about a uniform perturbation in the normal direction?
  - When all panels are roughly the same size, works well.
  - But when panels span many length scales…
    - can over-resolve the largest length scales.
    - can self-intersect.

Γ' should adapt to local panel size

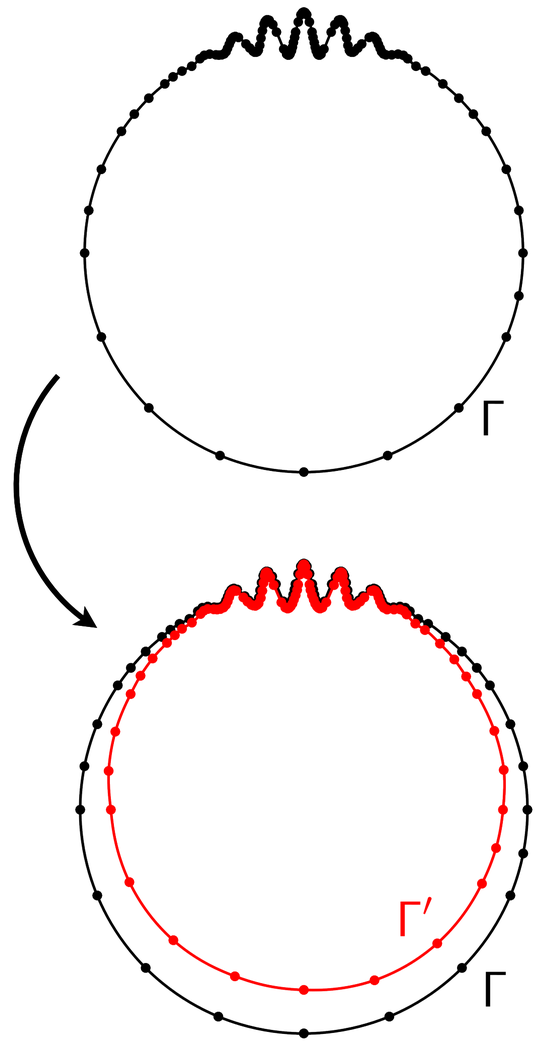
Defining the strip
Adapting to local panel size

Task: Given a panelized curve Γ, compute another panelized curve Γ' that is:

- inside Γ.
- as smooth as Γ.
- not too close to Γ (or the roll off will be sharp).
- not too far from Γ (or the strip will be large → extra work).
- not self intersecting.

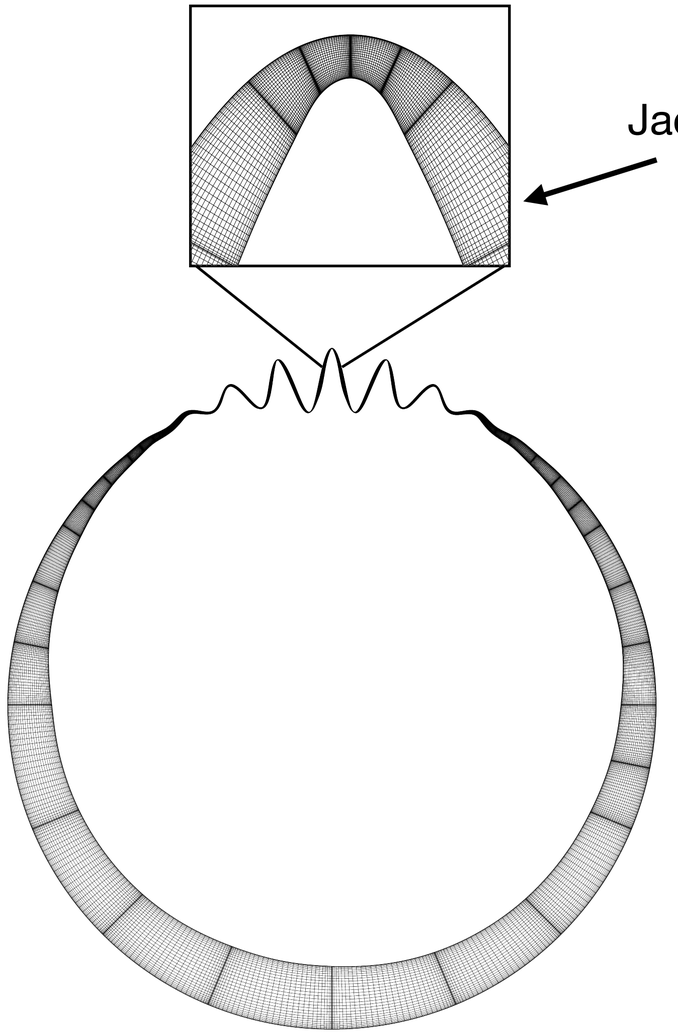
Solution:

- Define piecewise linear width function based on average local panel size
- Approximate each junction by smoothed abs(x)
- Blend together using matched asymptotics
- Perturb in the normal direction

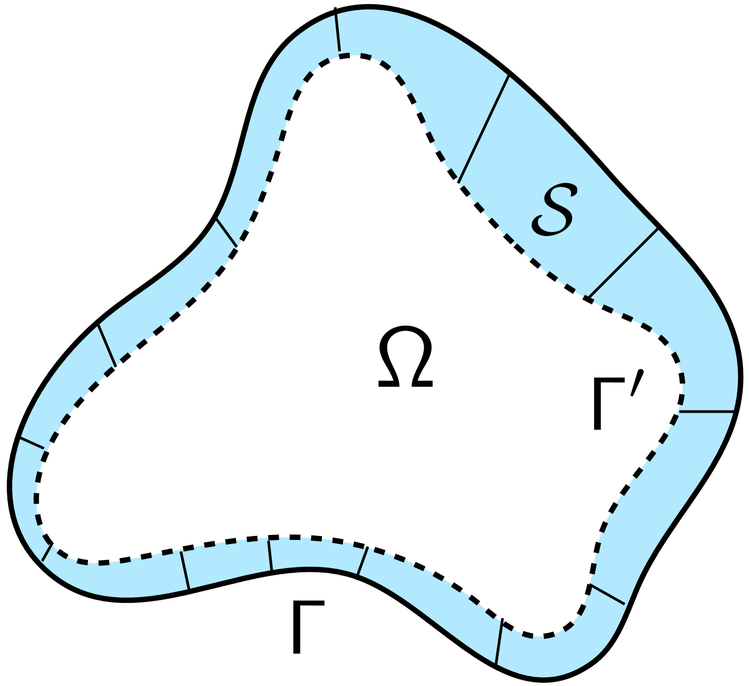


Solving the strip problem
Spectral element discretization

We use a spectral element method in $\mathcal{S}$, with spectral collocation at tensor-product Chebyshev nodes on each element.



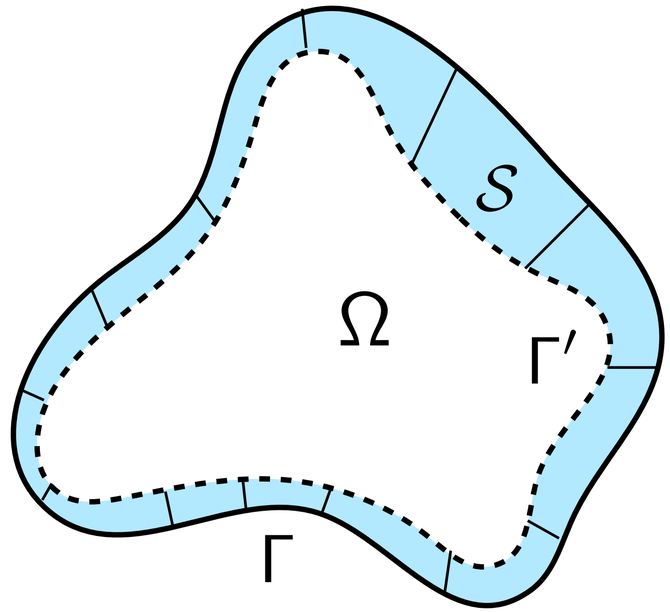
Jacobian is numerically computed at each point to form differentiation matrices.



Solving the strip problem

A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.



[Gillman & Martinsson, 2015], [Martinsson, 2015]

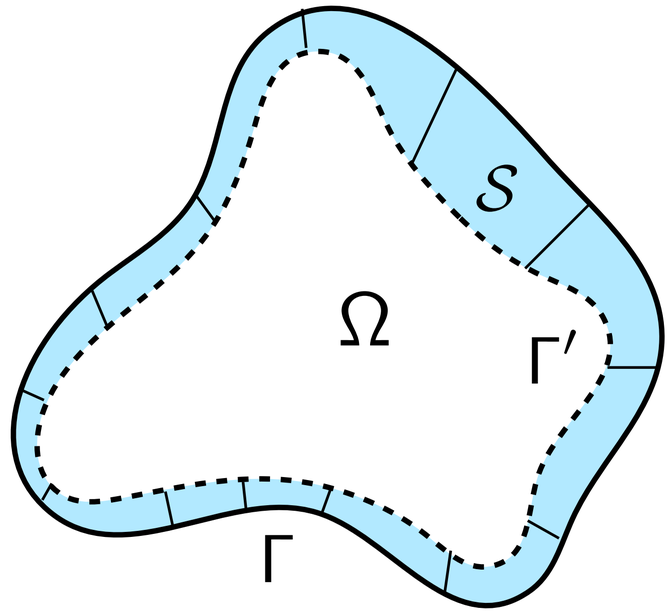
Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem

A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

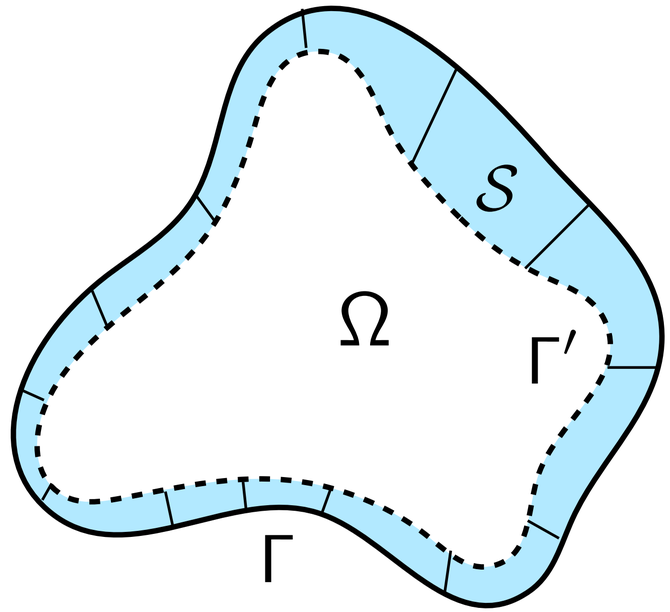
Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

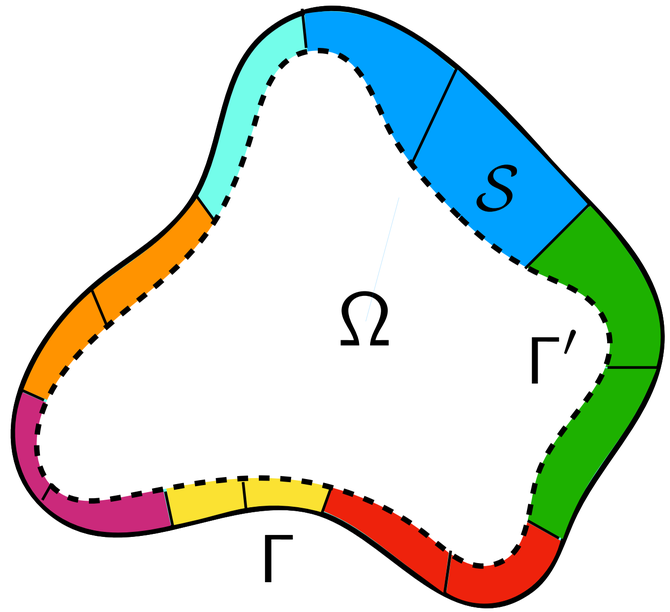
Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

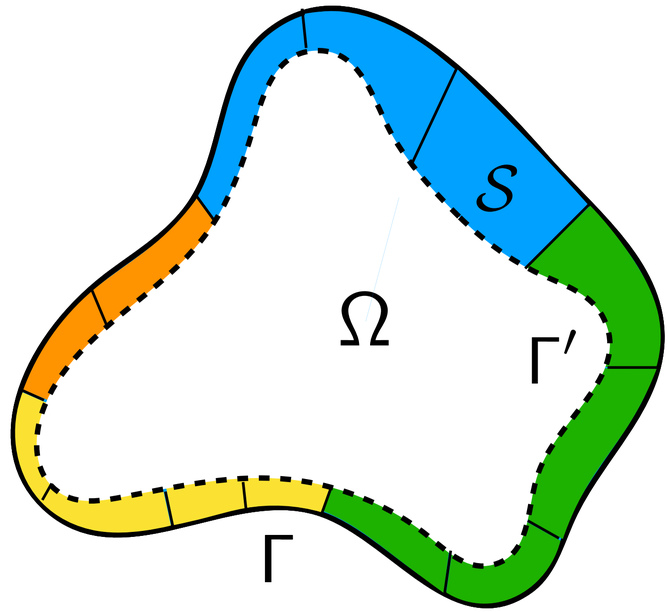
Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

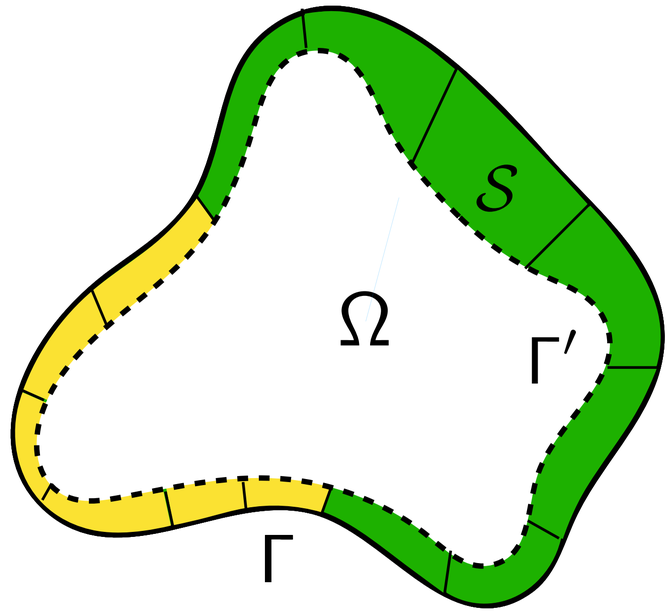
Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

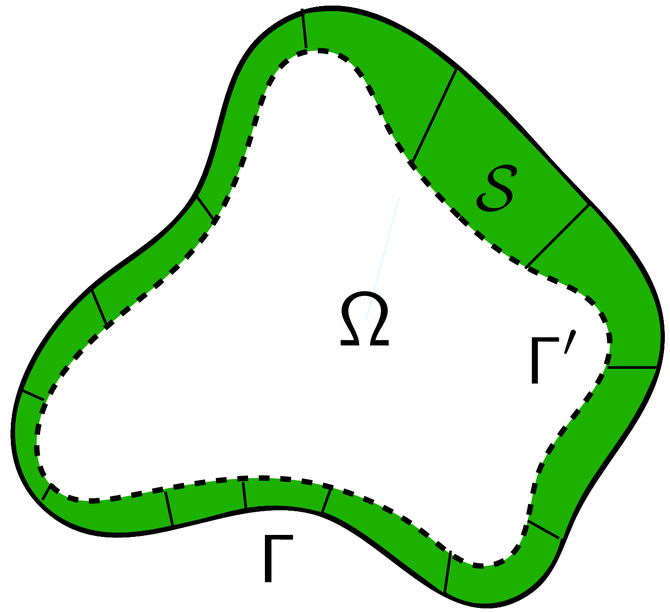
Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

Given an inhomogeneity f:

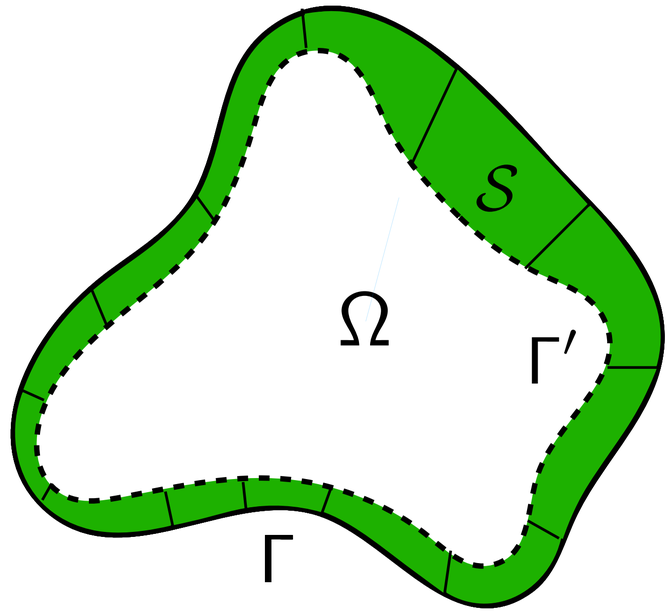
① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement

③ Recursively apply S, starting from known boundary conditions at the top level.



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem
A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

Given an inhomogeneity f:

① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

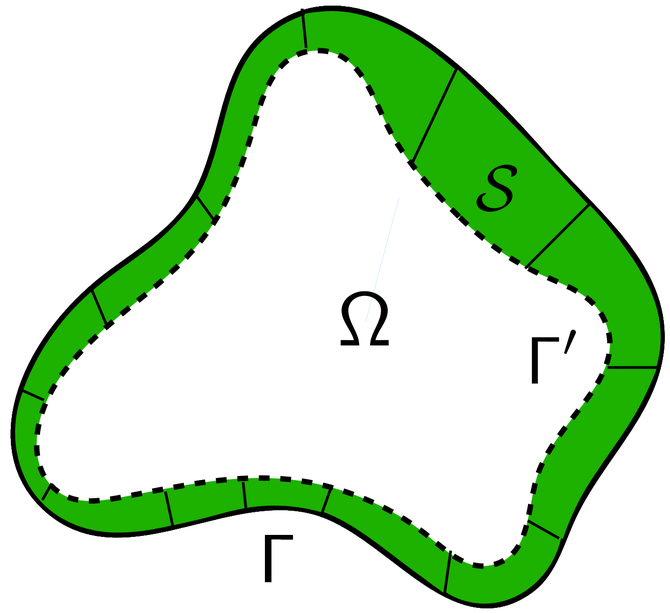
② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement

③ Recursively apply S, starting from known boundary conditions at the top level.

Cost: $\underbrace{\mathcal{O}(p^6 n_{\text{panel}})}_{①} + \underbrace{\mathcal{O}(p^3 n_{\text{panel}})}_{②} + \underbrace{\mathcal{O}(p^2 n_{\text{panel}})}_{③} = \mathcal{O}(n_{\text{panel}})$

We typically use p = 16 on each panel and upsample the SEM grid to 2p = 32.



[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the strip problem

A fast direct solver for the strip

We use the hierarchical Poincare-Steklov scheme to build a **fast direct solver** in $\mathcal{S}$.

Given an inhomogeneity f:

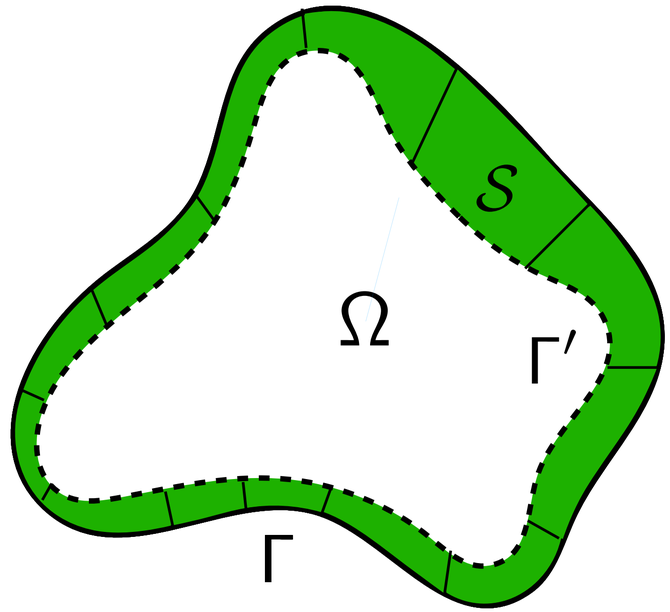
① On each element, compute:

- Solution operator: $S \in \mathbb{R}^{n^2 \times 4n}$
- Dirchlet-to-Neumann map: $DtN \in \mathbb{R}^{4n \times 4n}$

② Merge adjacent elements pairwise

- Compute S and DtN on parent via Schur complement

③ Recursively apply S, starting from known boundary conditions at the top level.



Cost: $\underbrace{\mathcal{O}(p^6 n_{\text{panel}})}_{①} + \underbrace{\mathcal{O}(p^3 n_{\text{panel}})}_{②} + \underbrace{\mathcal{O}(p^2 n_{\text{panel}})}_{③} = \mathcal{O}(n_{\text{panel}})$

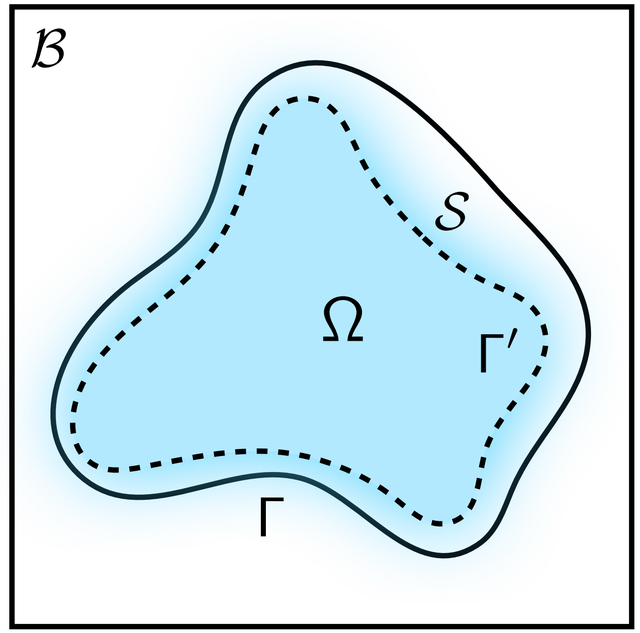
Takeaway: 1D HPS is fast out of the box.

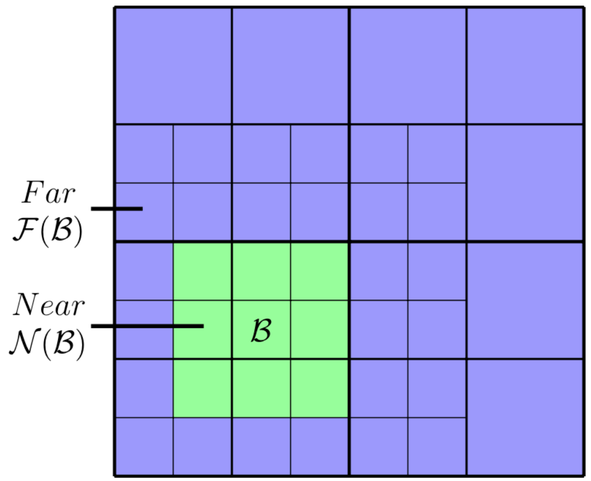
We typically use p = 16 on each panel and upsample the SEM grid to 2p = 32.

[Gillman & Martinsson, 2015], [Martinsson, 2015]

Solving the bulk problem
Evaluating the roll off function

- **Goal**: Approximate $\tilde{f}$ by quad-tree of tensor product Chebyshev nodes.

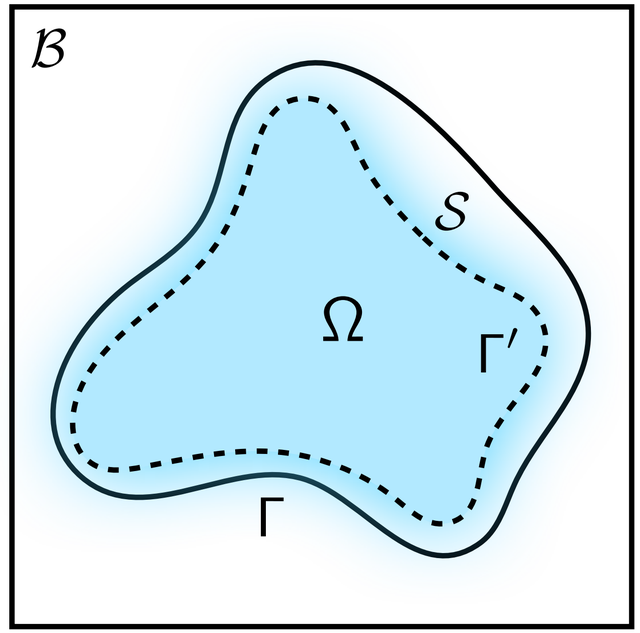


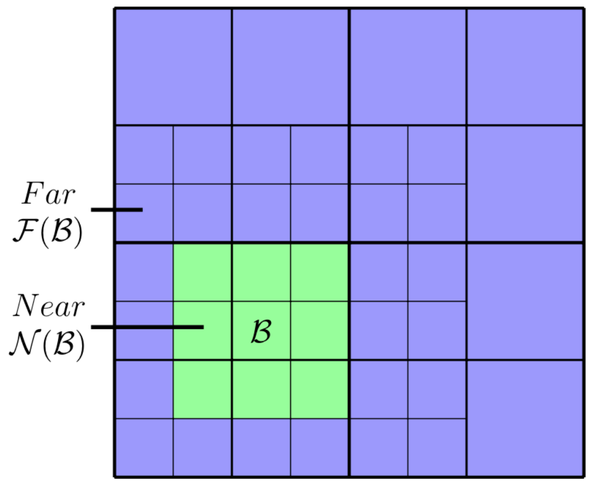


Solving the bulk problem
Evaluating the roll off function

- **Goal**: Approximate $\tilde{f}$ by quad-tree of tensor product Chebyshev nodes.

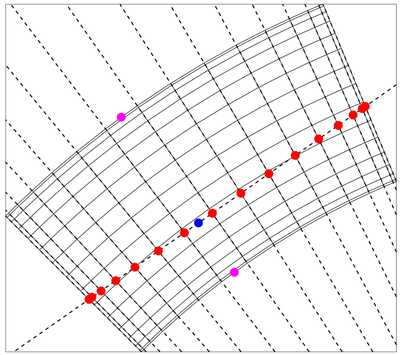
- **Problem**: To evaluate the roll off function at a point, we need to know where that point falls in the strip.

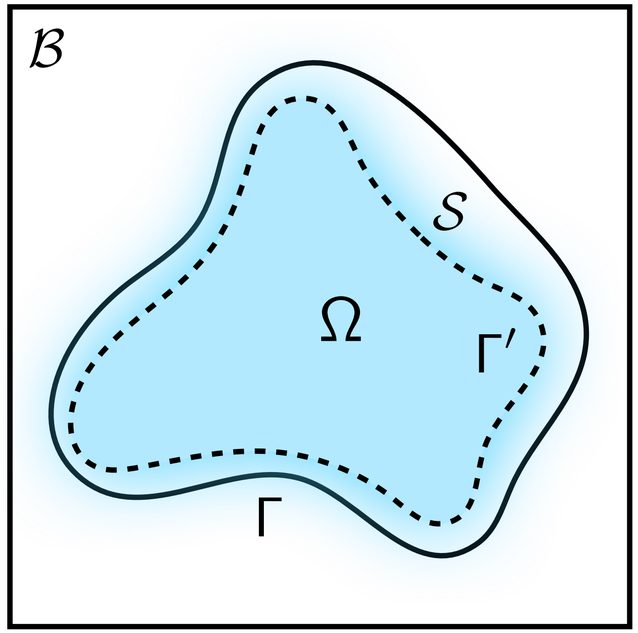


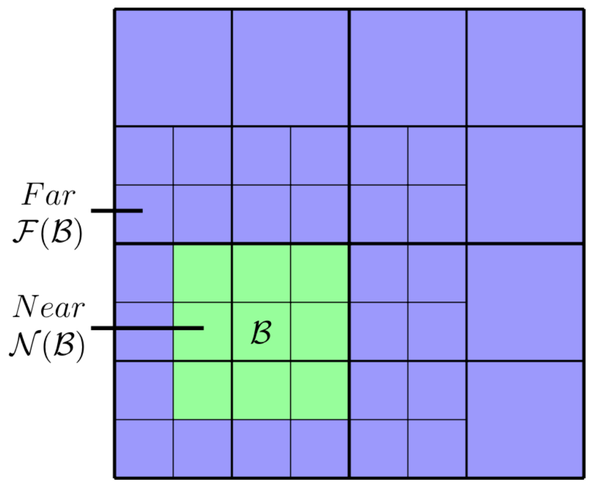


Solving the bulk problem
Evaluating the roll off function

- **Goal**: Approximate $\tilde{f}$ by quad-tree of tensor product Chebyshev nodes.
- **Problem**: To evaluate the roll off function at a point, we need to know where that point falls in the strip.
- **Solution**: Compute local coordinates via 1D interpolation through normal vectors [Bruno & Paul, 2020]

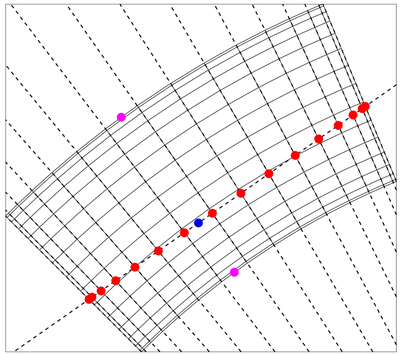


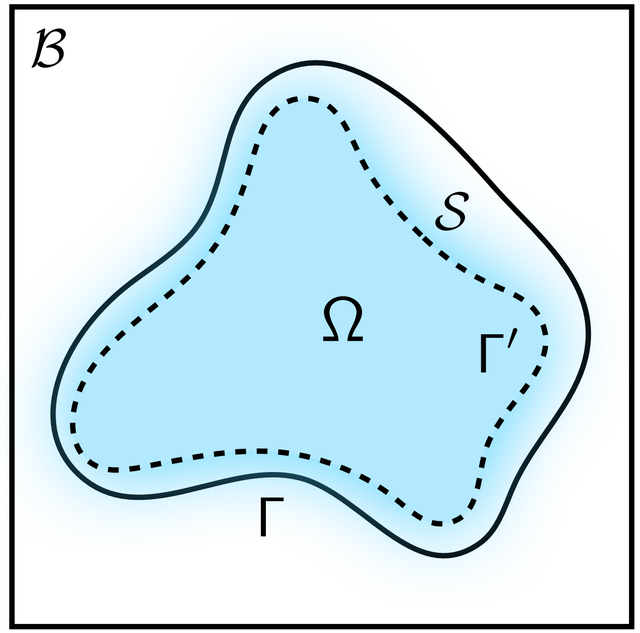




Solving the bulk problem
Evaluating the roll off function

- **Goal**: Approximate $\tilde{f}$ by quad-tree of tensor product Chebyshev nodes.
- **Problem**: To evaluate the roll off function at a point, we need to know where that point falls in the strip.
- **Solution**: Compute local coordinates via 1D interpolation through normal vectors [Bruno & Paul, 2020]

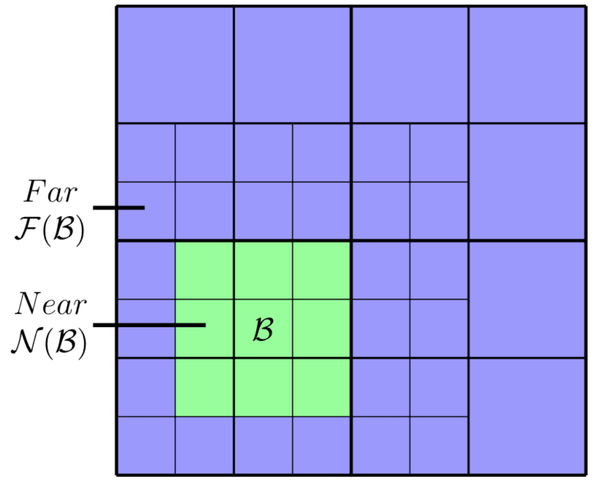




Use a box code to obtain a particular solution:

$$\Delta u_{\text{bulk}} = \tilde{f} \text{ in } \mathcal{B} \qquad \mathcal{O}(N) = \mathcal{O}(p^2 n_{\text{boxes}})$$

Then, $\Delta u_{\text{bulk}} = f$ inside Γ'.



Patching the solutions

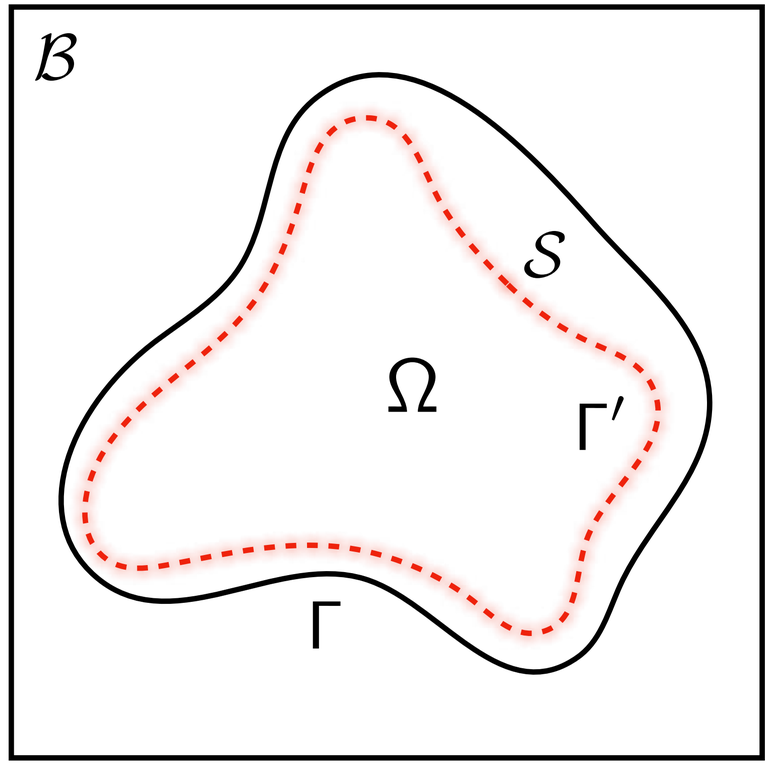
Using the value of u_{bulk} as the inner boundary condition in the SEM, the solutions will match along Γ'. However, their normal derivatives may not.

The "single-layer potential"

$$v(x) = (S\sigma)(x) = \int_{\Gamma'} G(x,y)\sigma(y)dt_y$$

satisfies $\Delta v = 0$ and the jump relation

$$\left.\frac{\partial v}{\partial n}^{+}\right|_{\Gamma'} - \left.\frac{\partial v}{\partial n}^{-}\right|_{\Gamma'} = -\sigma$$



Patching the solutions

Using the value of u_{bulk} as the inner boundary condition in the SEM, the solutions will match along Γ'. However, their normal derivatives may not.

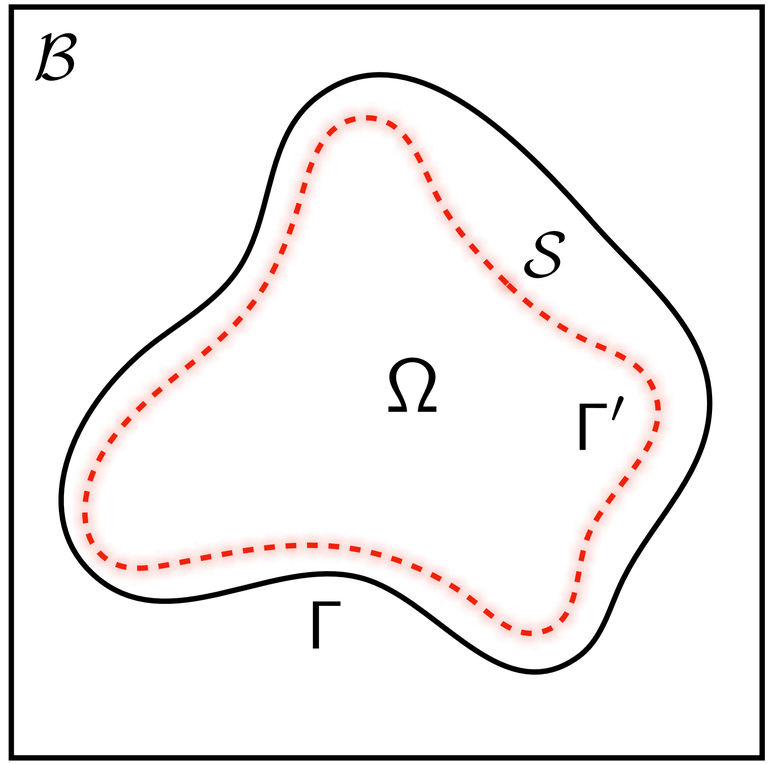
The "single-layer potential"

$$v(x) = (S\sigma)(x) = \int_{\Gamma'} G(x,y)\sigma(y)dt_y$$

satisfies $\Delta v = 0$ and the jump relation

$$\left.\frac{\partial v}{\partial n}^{+}\right|_{\Gamma'} - \left.\frac{\partial v}{\partial n}^{-}\right|_{\Gamma'} = -\sigma$$

Apply SLP: $$u_{\text{glue}} = S\left(\left.\frac{\partial u_{\text{strip}}}{\partial n}\right|_{\Gamma'} - \left.\frac{\partial u_{\text{bulk}}}{\partial n}\right|_{\Gamma'}\right)$$



Patching the solutions

Using the value of u_{bulk} as the inner boundary condition in the SEM, the solutions will match along Γ'. However, their normal derivatives may not.

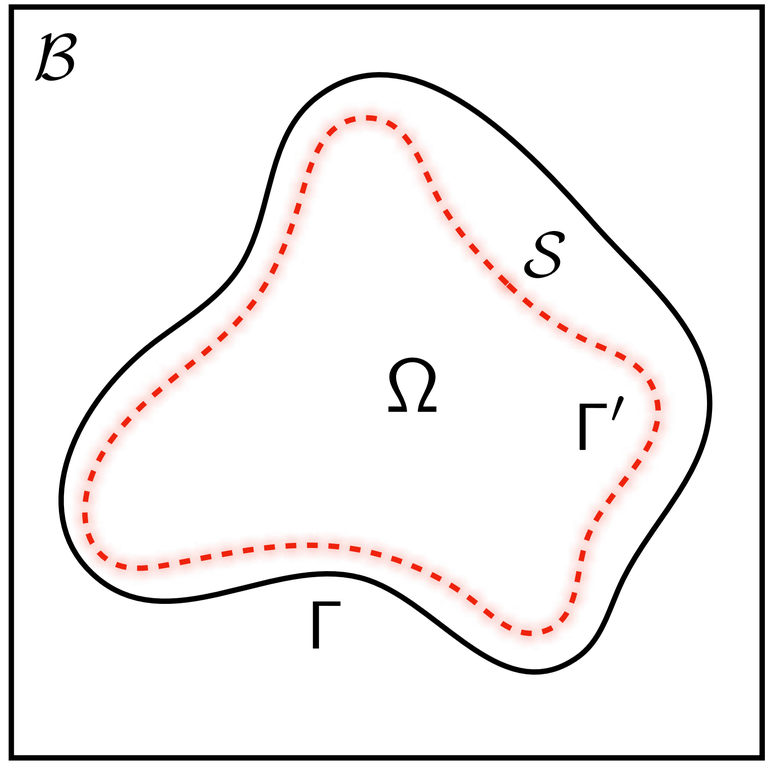
The "single-layer potential"

$$v(x) = (S\sigma)(x) = \int_{\Gamma'} G(x,y)\sigma(y)dt_y$$

satisfies $\Delta v = 0$ and the jump relation

$$\left.\frac{\partial v}{\partial n}^{+}\right|_{\Gamma'} - \left.\frac{\partial v}{\partial n}^{-}\right|_{\Gamma'} = -\sigma$$

Apply SLP: $u_{\text{glue}} = S\left(\left.\frac{\partial u_{\text{strip}}}{\partial n}\right|_{\Gamma'} - \left.\frac{\partial u_{\text{bulk}}}{\partial n}\right|_{\Gamma'}\right)$



Then, $\Delta(u_{\text{bulk}} + u_{\text{glue}}) = f$ in $\Omega \setminus \mathcal{S}$

$\Delta(u_{\text{strip}} + u_{\text{glue}}) = f$ in $\mathcal{S}$

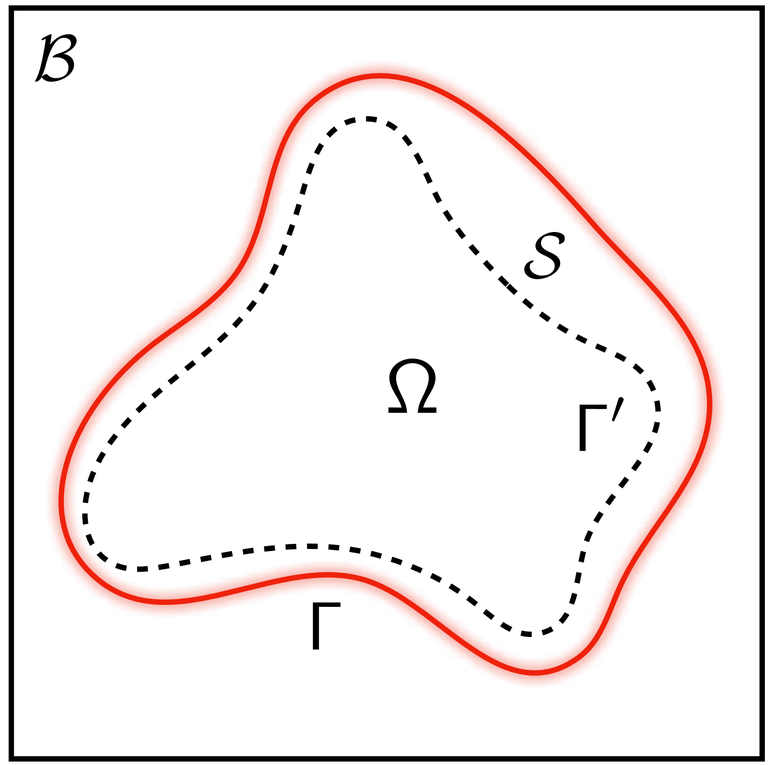
$u_{\text{strip}} = u_{\text{bulk}}$, $\frac{\partial u_{\text{strip}}}{\partial n} = \frac{\partial u_{\text{bulk}}}{\partial n}$ on Γ'

Correcting the boundary conditions

Finally, the boundary conditions may not be satisfied.

Standard BIE solve using double-layer potential:

$$\begin{aligned} \Delta u_{\text{bc}} &= 0 && \text{in } \Omega \\ u_{\text{bc}} &= g - (u_{\text{strip}}|_\Gamma + u_{\text{glue}}|_\Gamma) && \text{on } \Gamma \end{aligned}$$



Correcting the boundary conditions

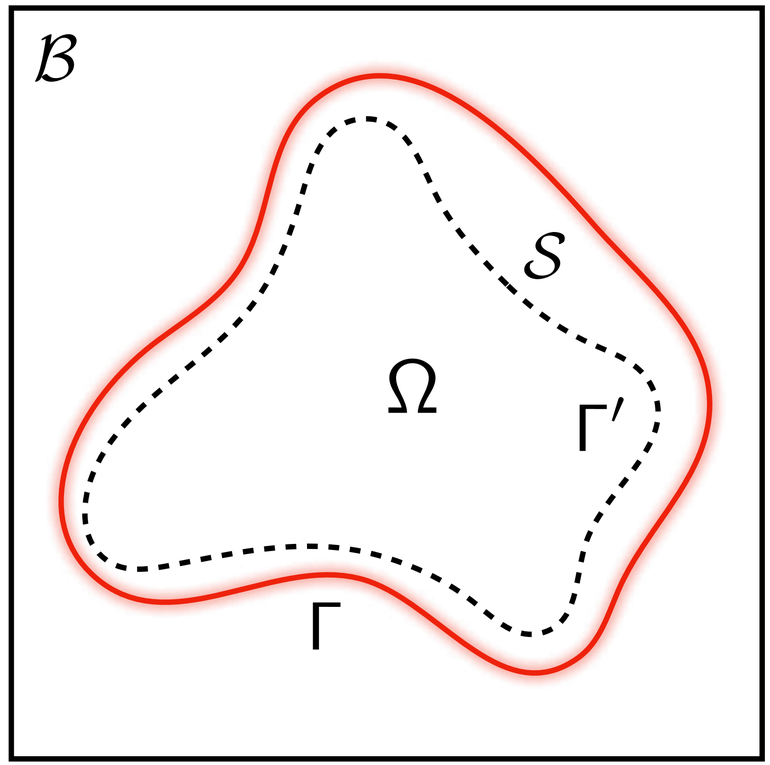
Finally, the boundary conditions may not be satisfied.

Standard BIE solve using double-layer potential:

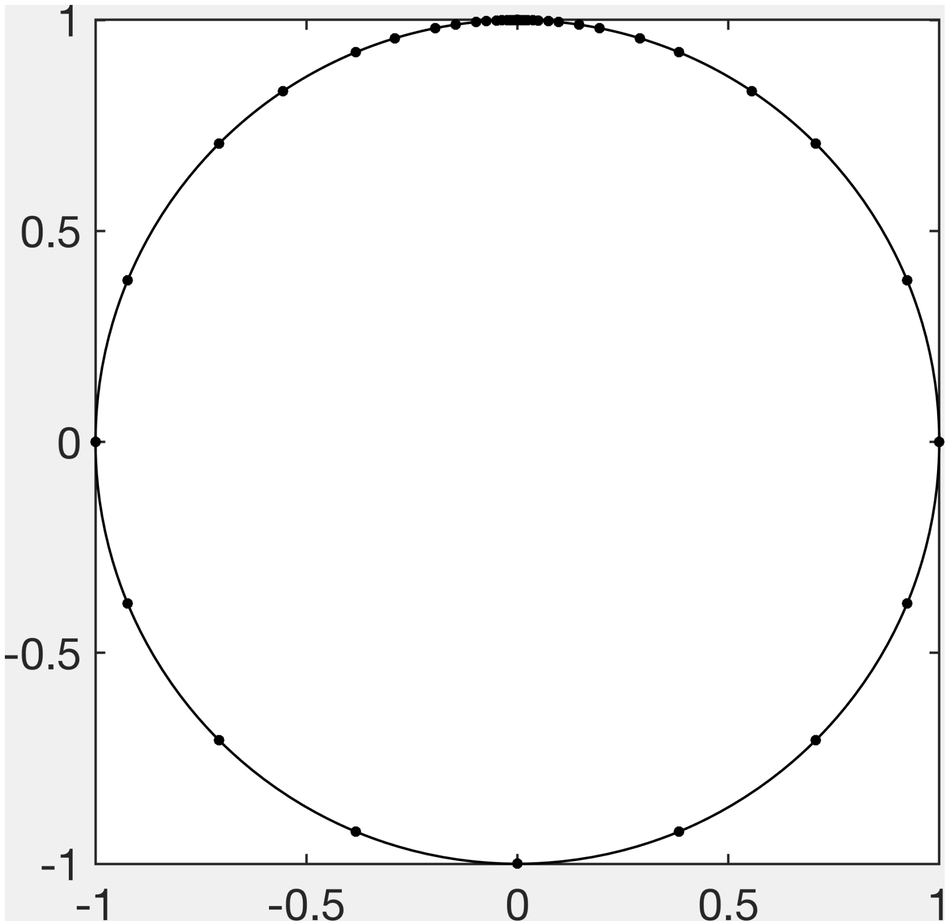
$$\begin{aligned} \Delta u_{\text{bc}} &= 0 && \text{in } \Omega \\ u_{\text{bc}} &= g - (u_{\text{strip}}|_\Gamma + u_{\text{glue}}|_\Gamma) && \text{on } \Gamma \end{aligned}$$

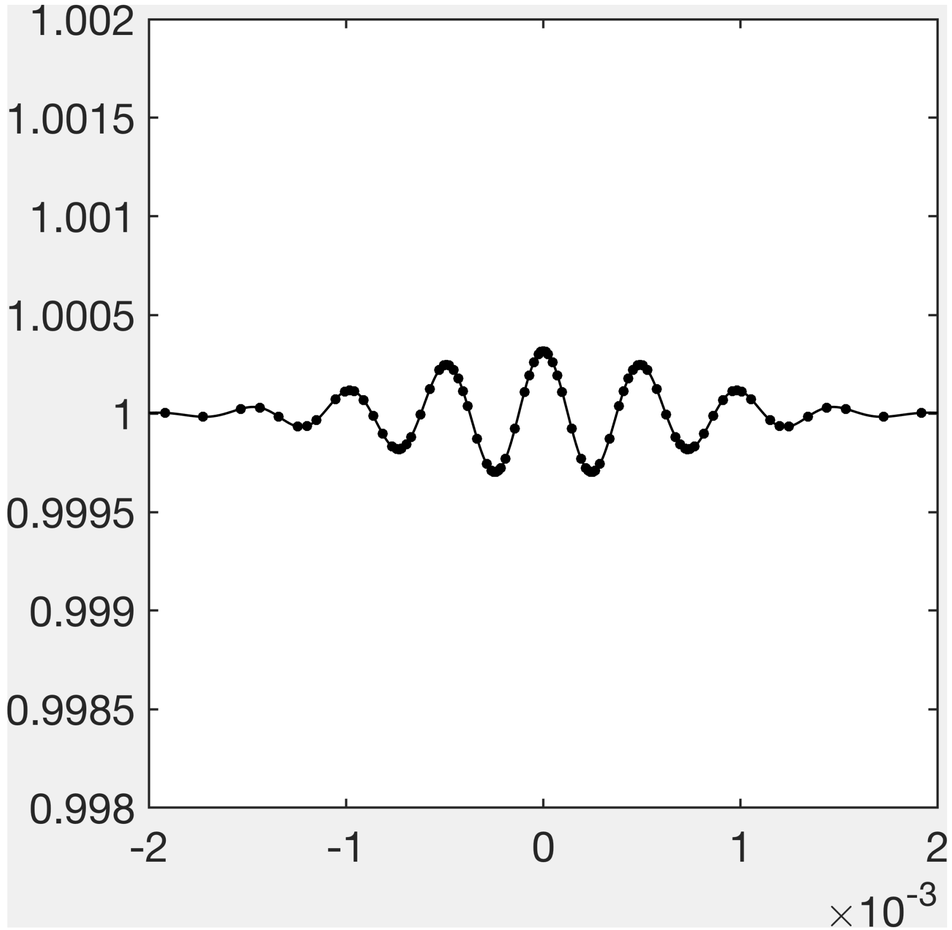
Then $u = \begin{cases} u_{\text{bulk}} + u_{\text{glue}} + u_{\text{bc}} \text{ in } \Omega \setminus \mathcal{S} \\ u_{\text{strip}} + u_{\text{glue}} + u_{\text{bc}} \text{ in } \mathcal{S} \end{cases}$ satisfies:

$$\begin{aligned} \Delta u &= f && \text{in } \Omega \\ u &= g && \text{on } \Gamma \end{aligned}$$

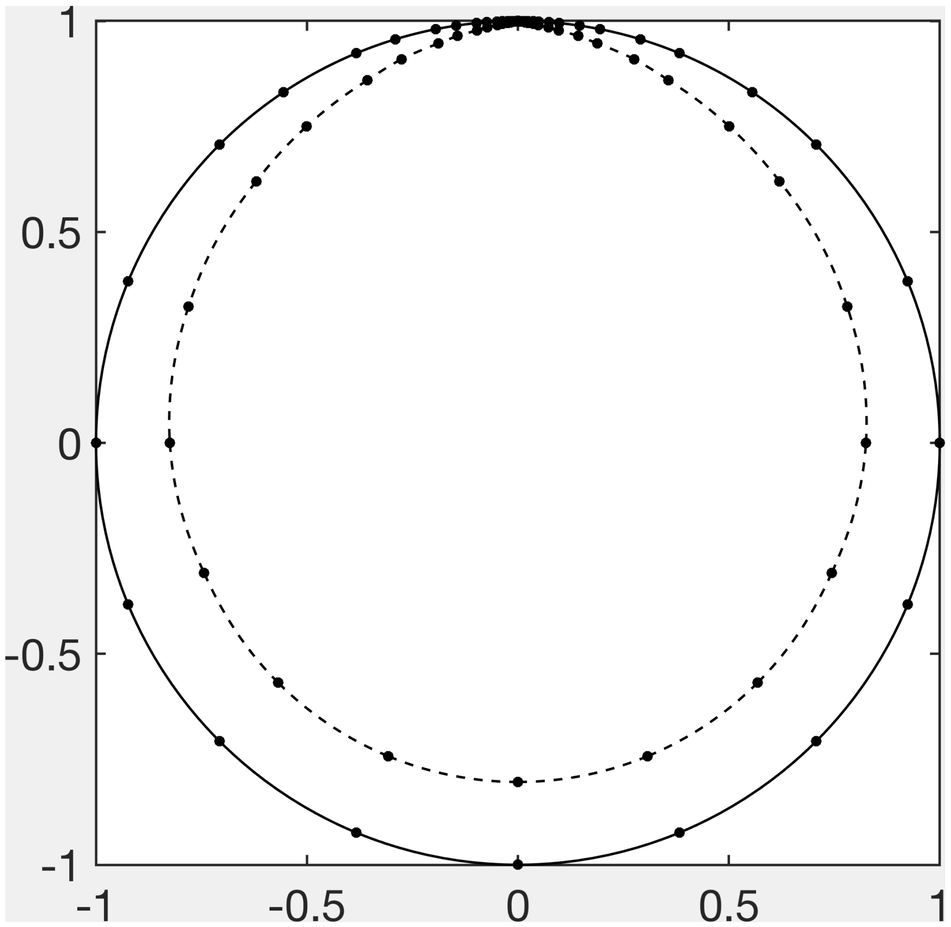


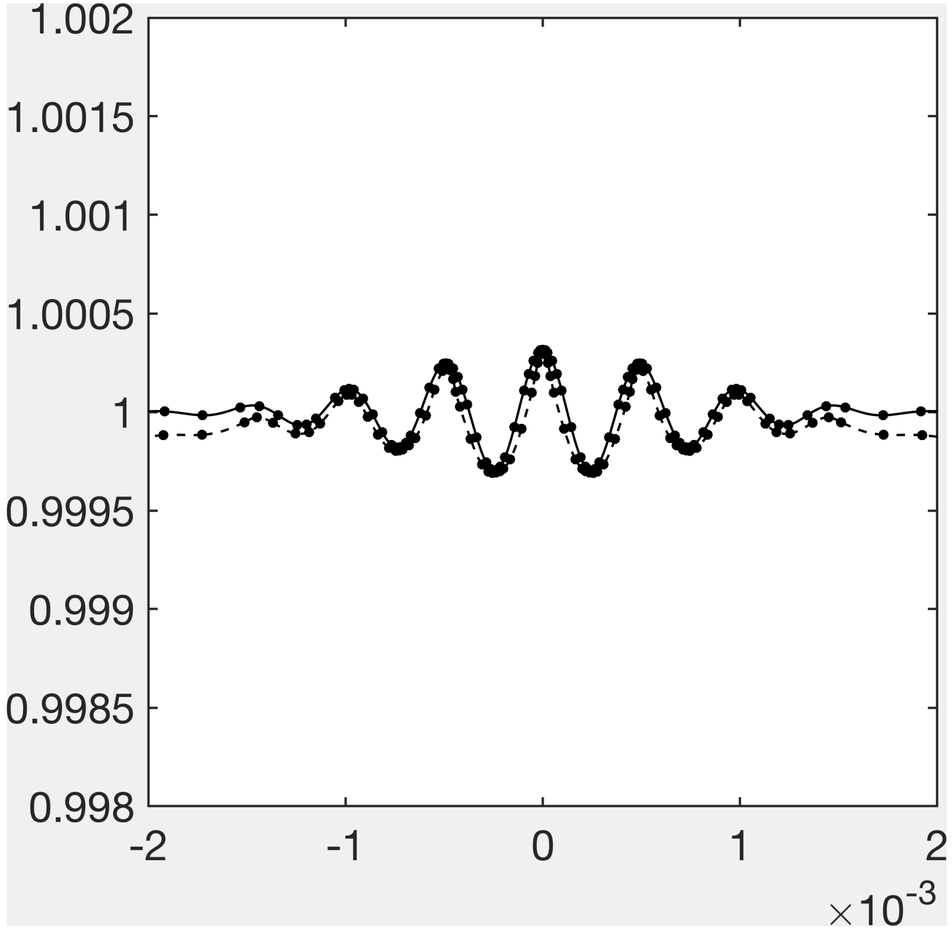
Numerical results
Example



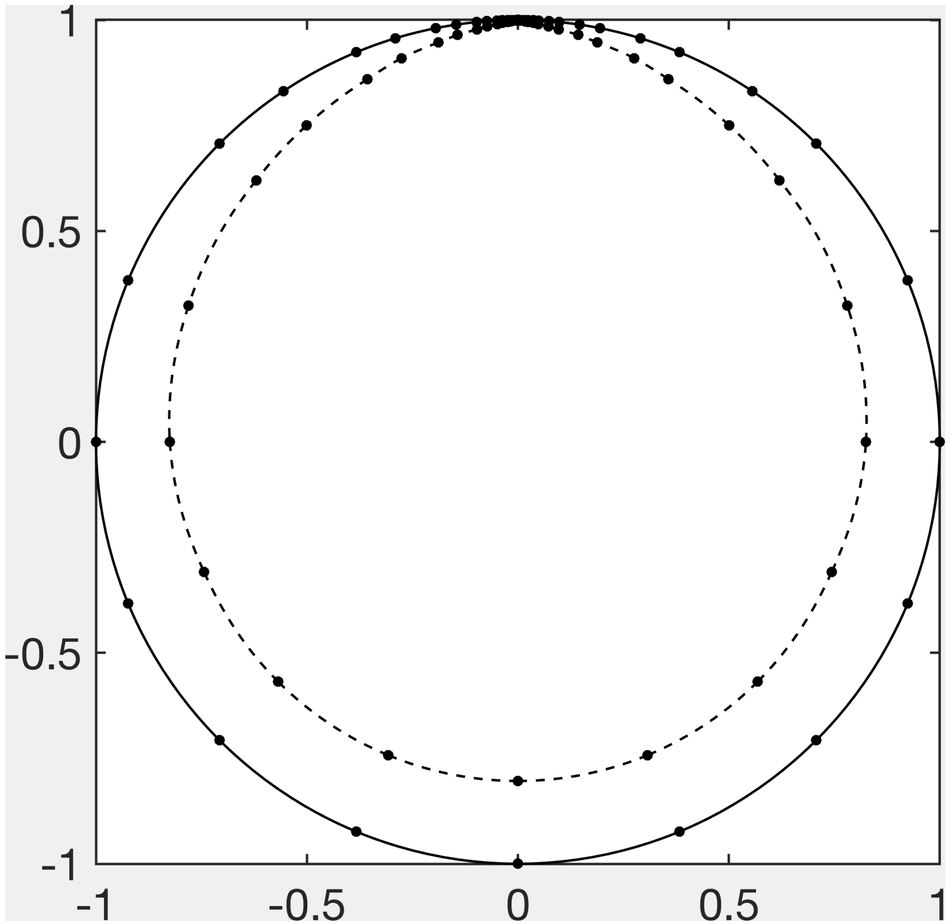


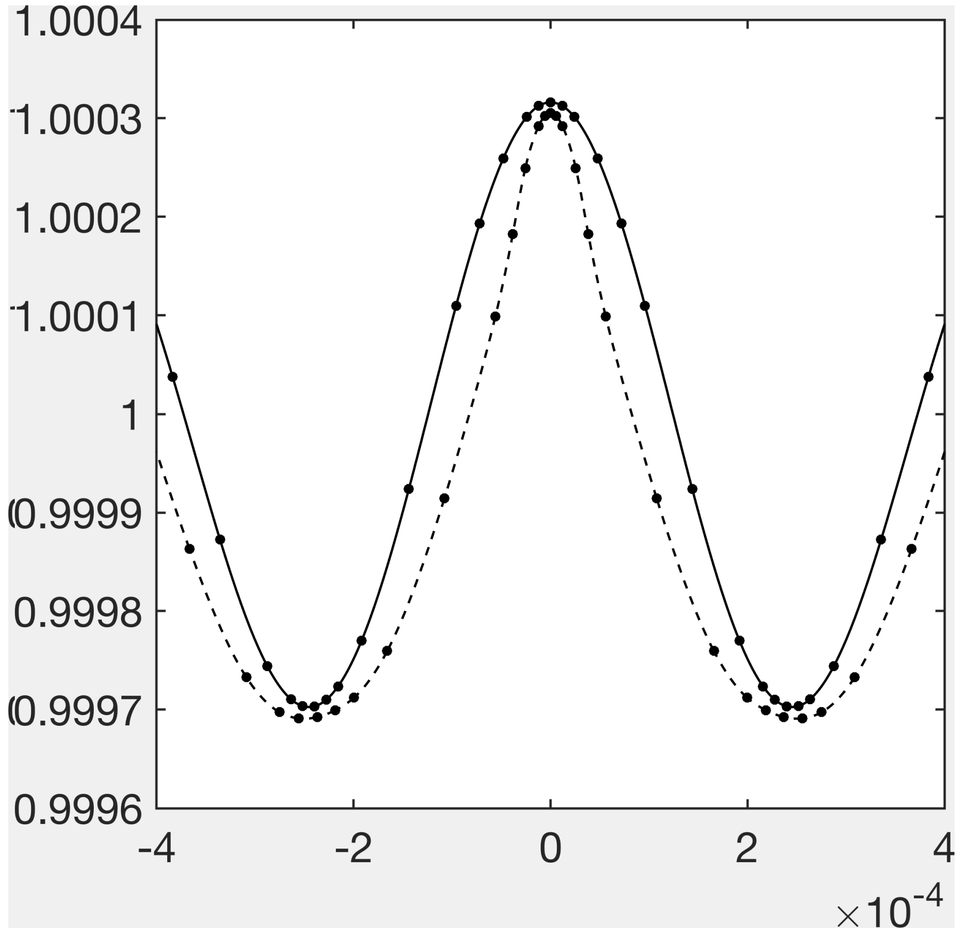
Numerical results
Example



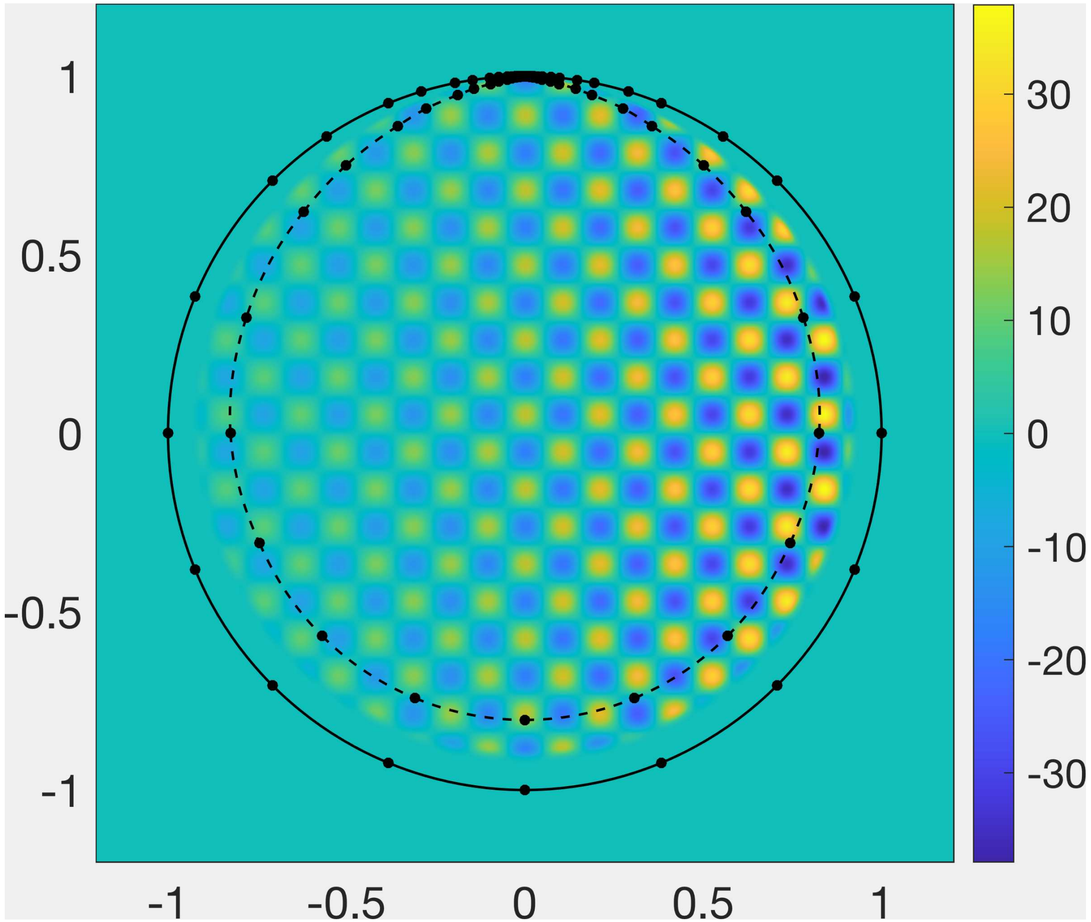


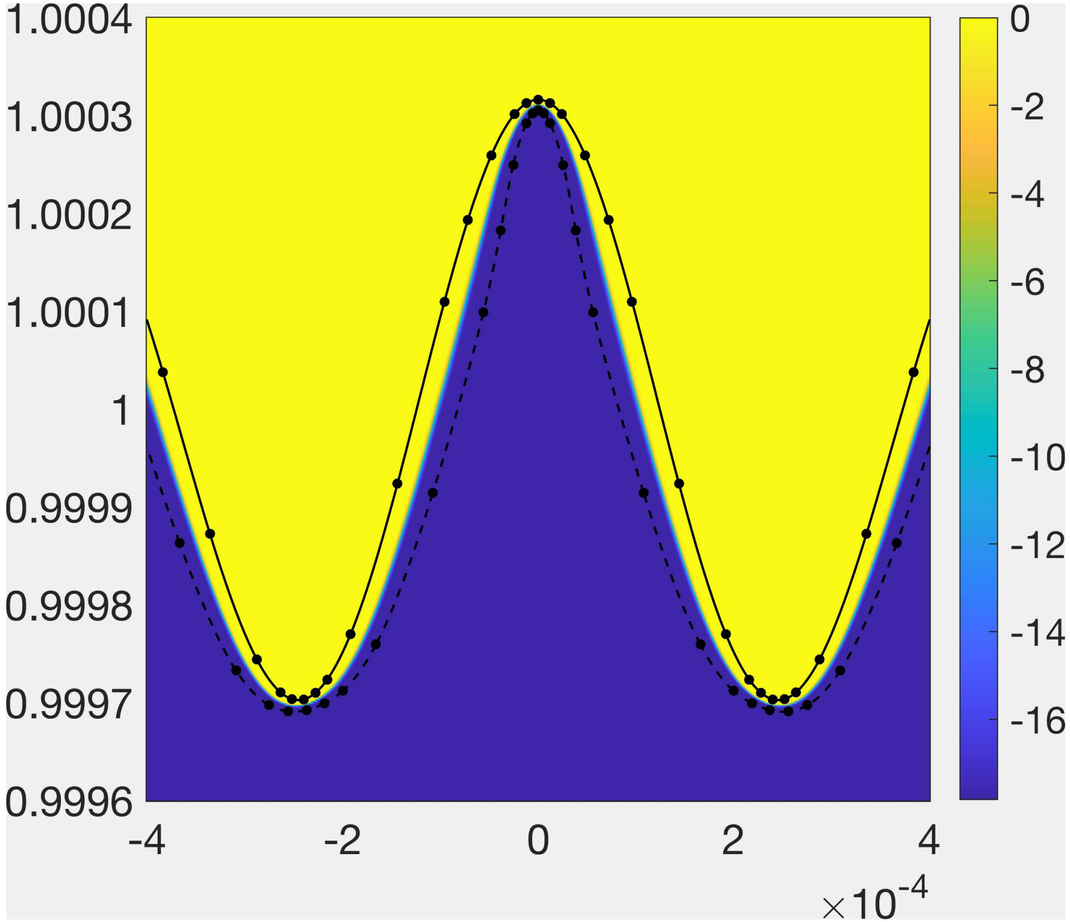
Numerical results
Example



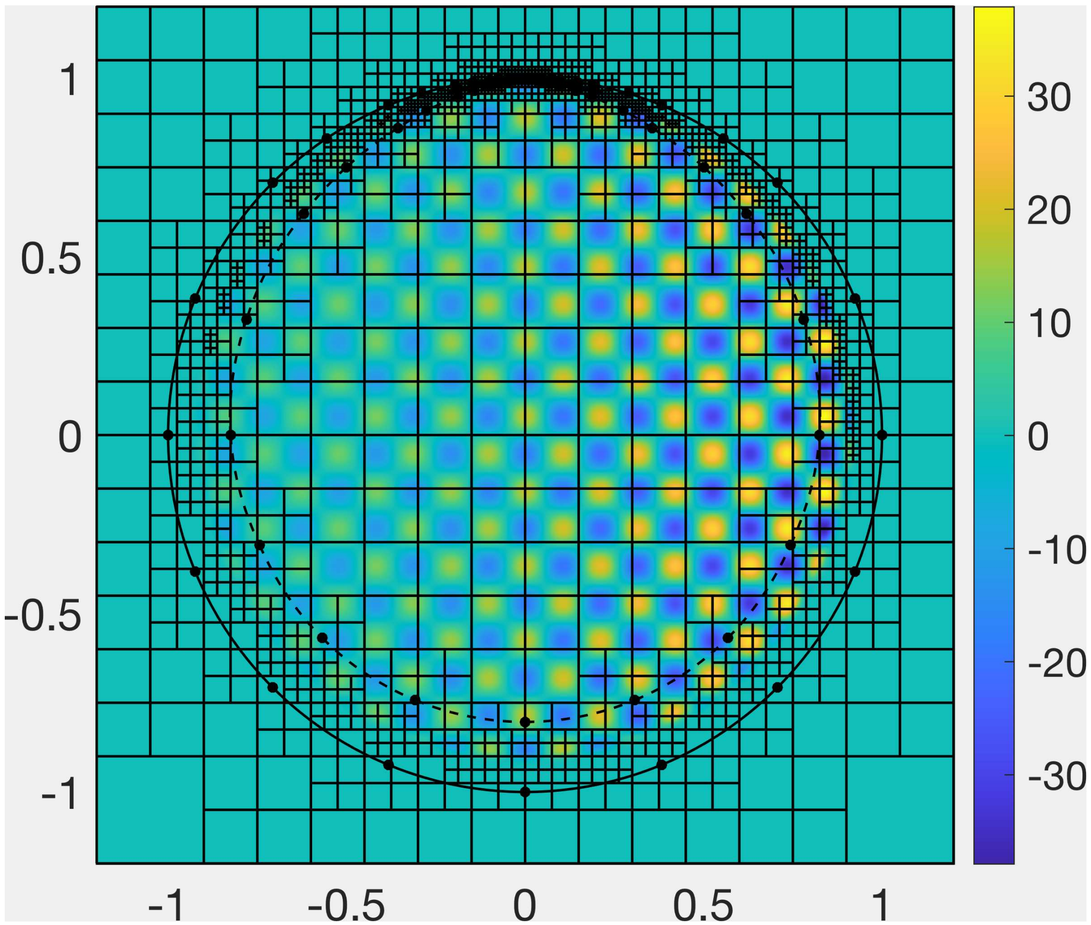


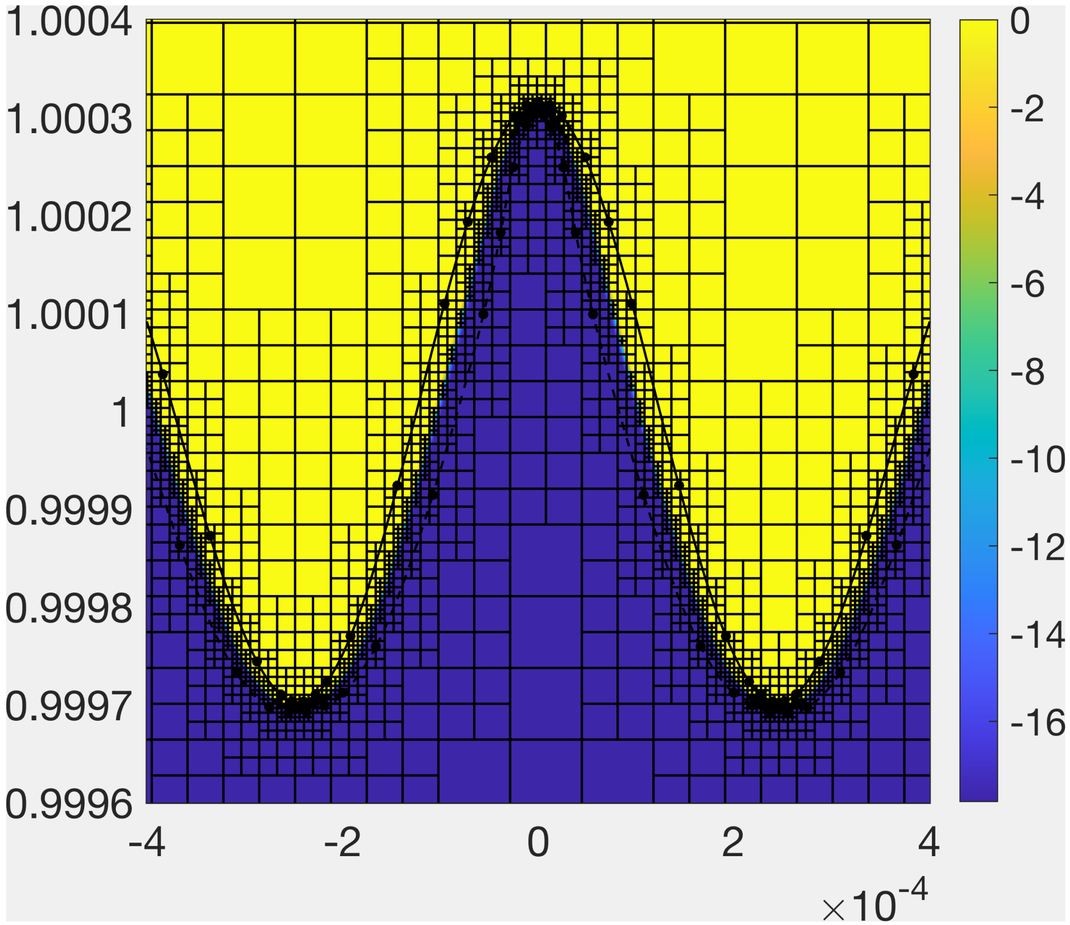
Numerical results
Example



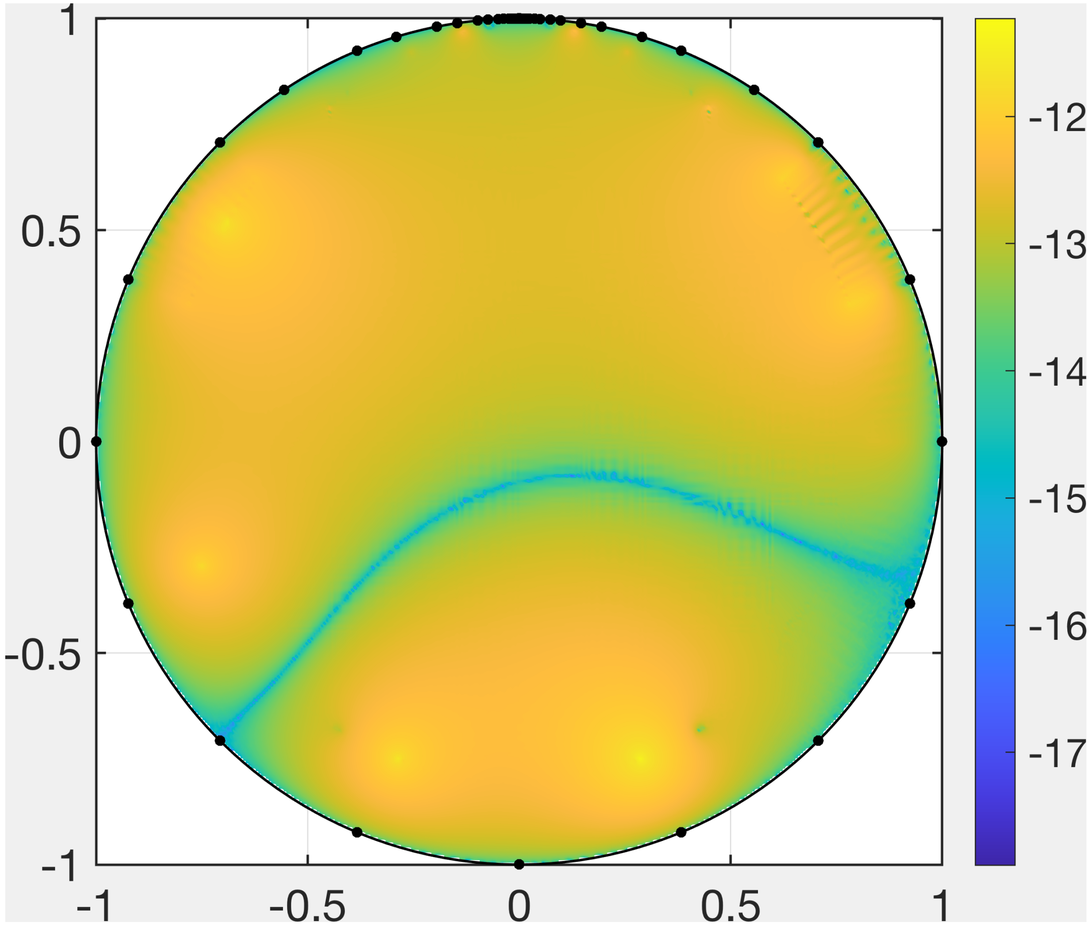


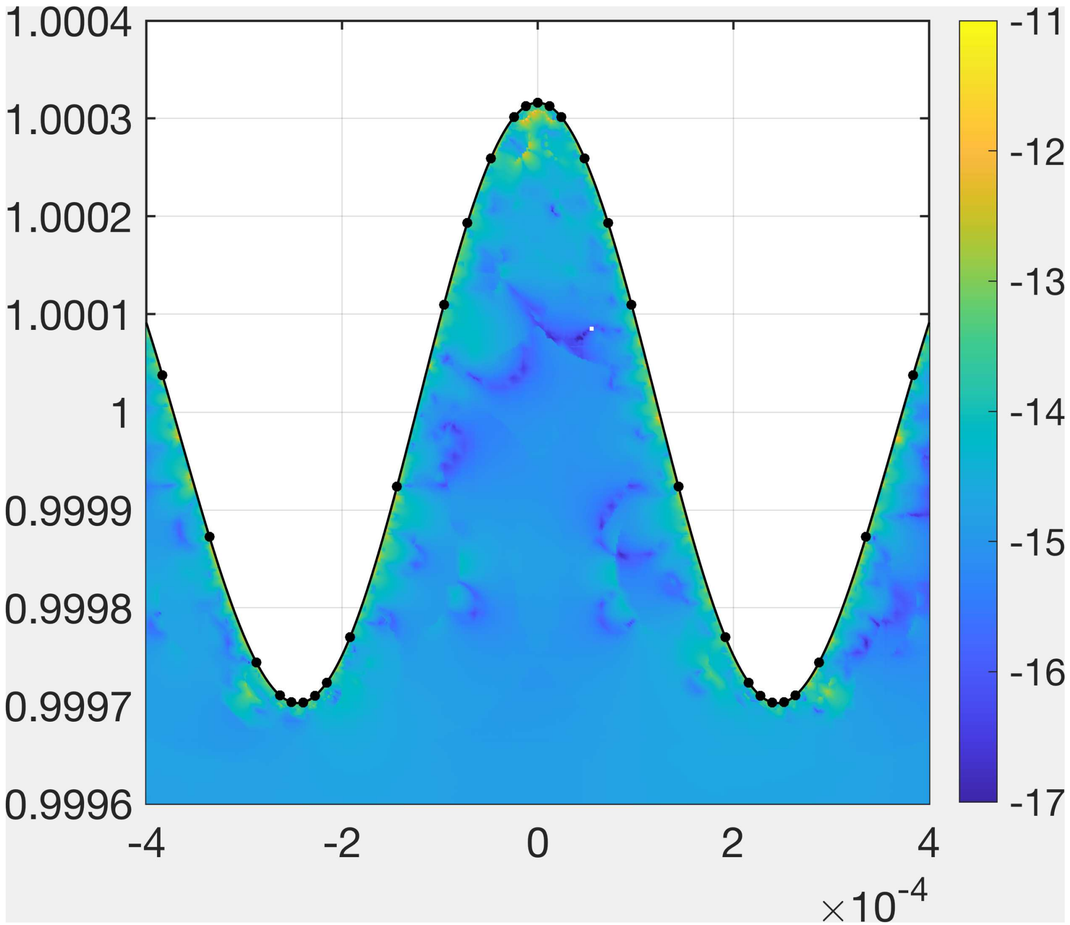
Numerical results
Example





Numerical results
Example





Thank you

- Still ongoing work
- Adaptivity can be performed on boundary (via panelization) and in volume (box code)
- Function “intension” can avoid pitfalls of function extension

Alex Barnett

FLATIRON INSTITUTE
Center for Computational Mathematics

David Stein