Unstructured to Structured

Geometric Multigrid on Complex Domains via Mesh Remapping

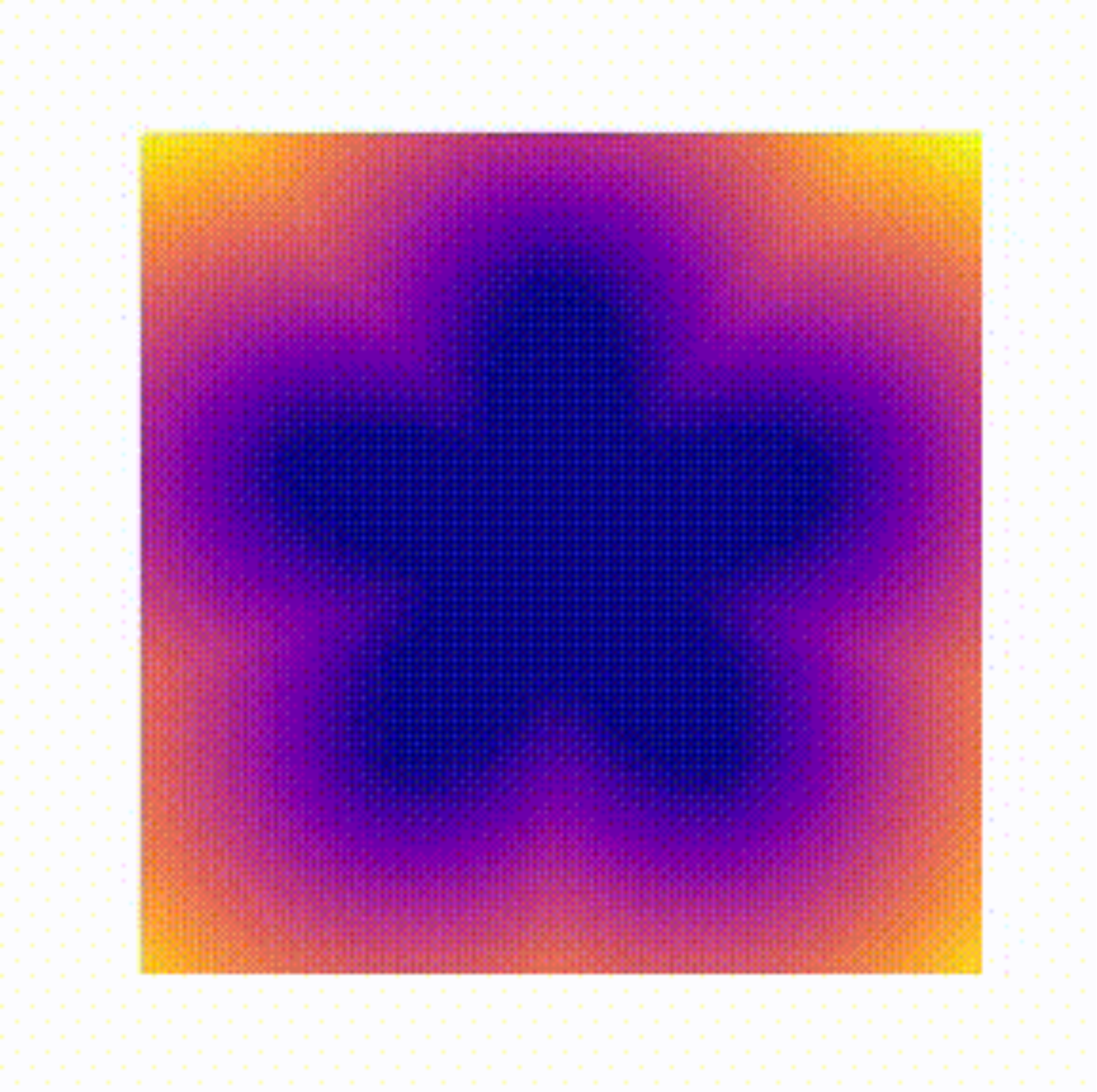
22nd Copper Mountain Conference on Multigrid Methods

Nicolas Nytko[1], Scott MacLachlan[3], J. David Moulton[2], Luke Olson[1], Andrew Reisner[2], Matt West[1]

[1] University of Illinois Urbana-Champaign [2] Memorial University of Newfoundland [3] Los Alamos National Laboratory

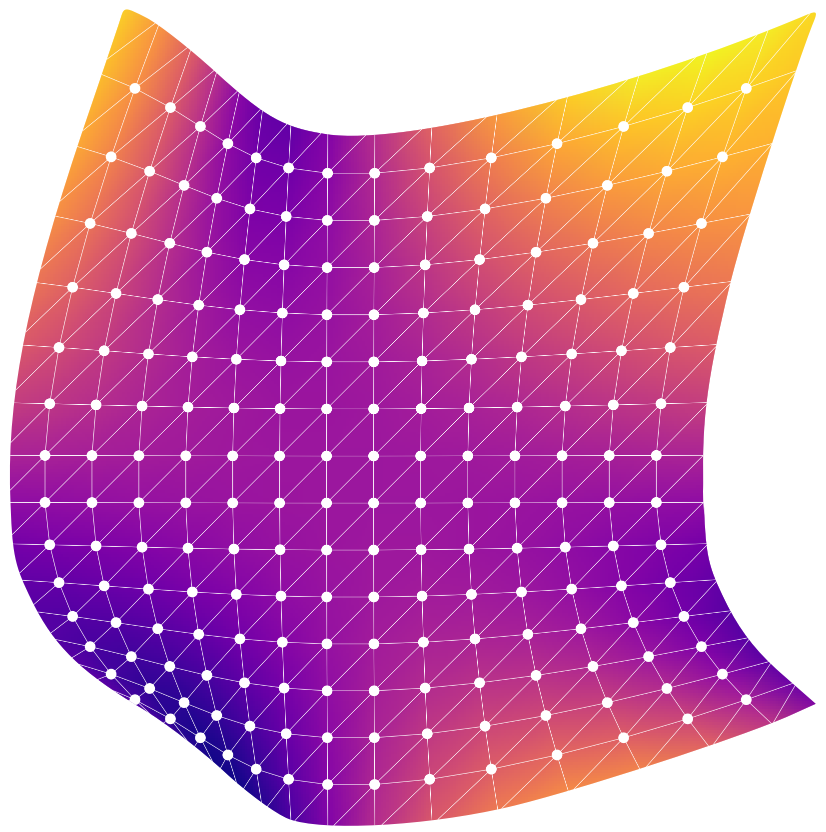
Introduction

- Geometric multigrid methods are fast, but require some sense of structure
- Want:
    - Speed and optimal convergence of geometric multigrid
    - To able to apply it to arbitrary complex meshes
    - Fast solve on GPU



Idea: remap to a simple domain

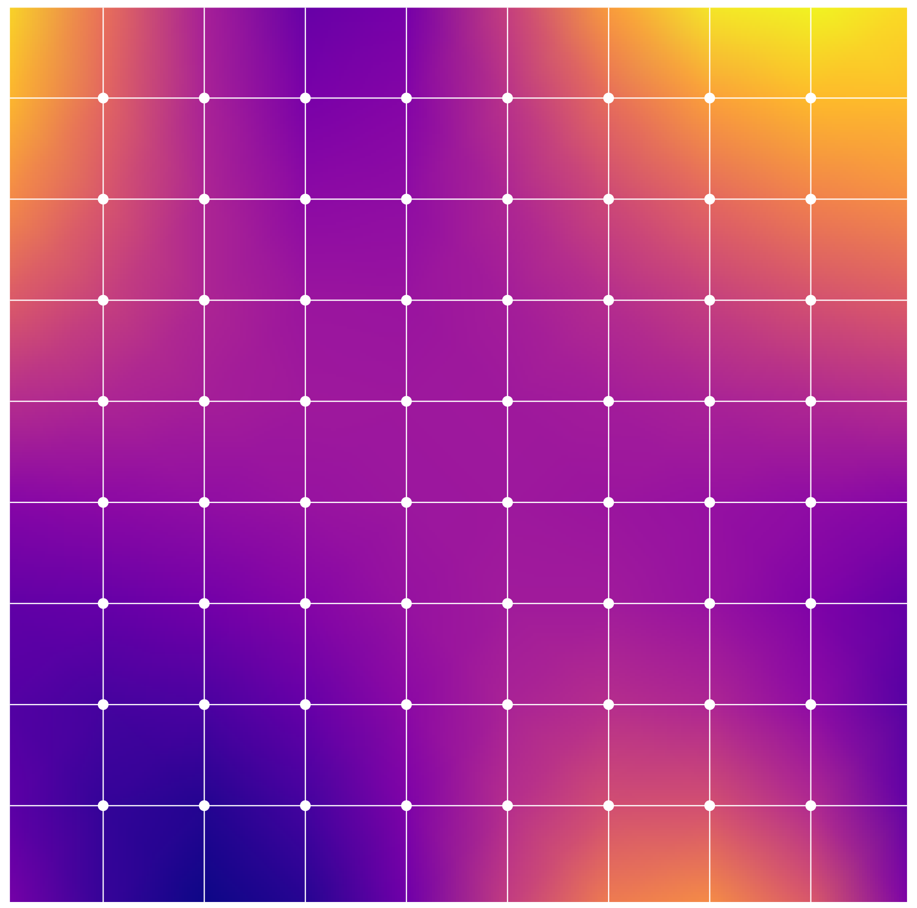
- Define an auxiliary *computational domain* along with a smooth, invertible map T



Ω_p

$$\rightarrow T \rightarrow$$

$$\leftarrow T^{-1} \leftarrow$$

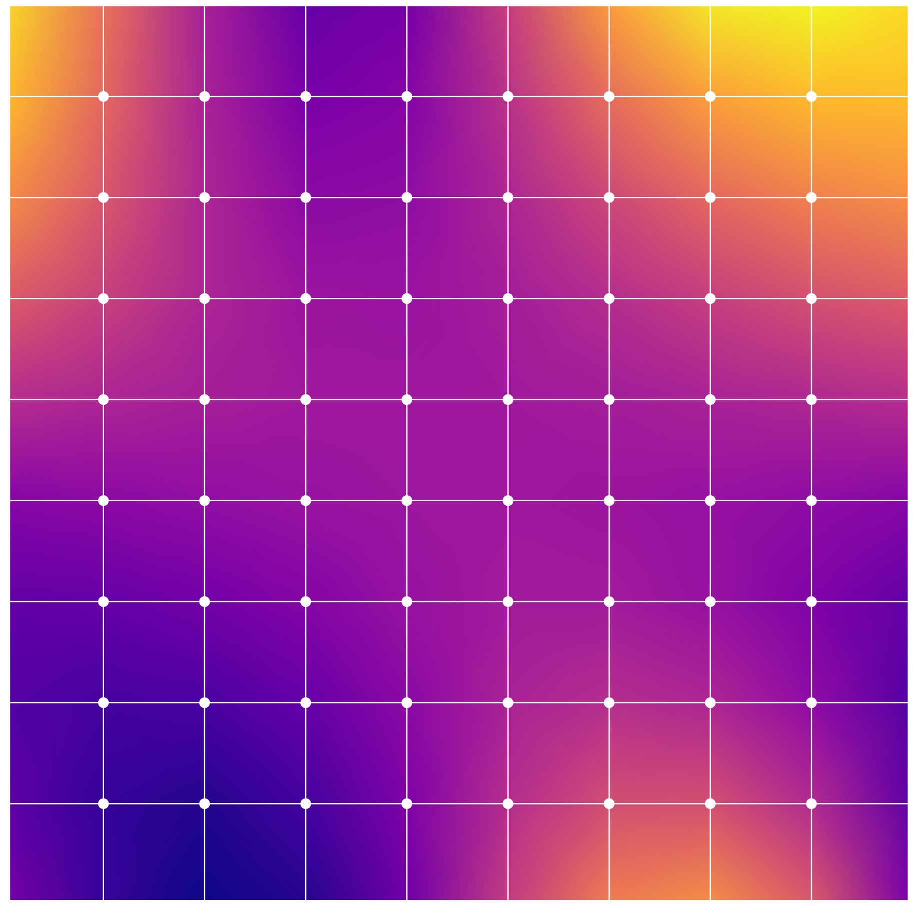


Ω_c

$$T : \Omega_p \rightarrow \Omega_c$$

Remapping to a simple domain

- Transferring the solve to a rectangular domain allows:
  - Using fast geometric solvers
  - Regular memory accesses
  - Better parallelism on SIMD architectures



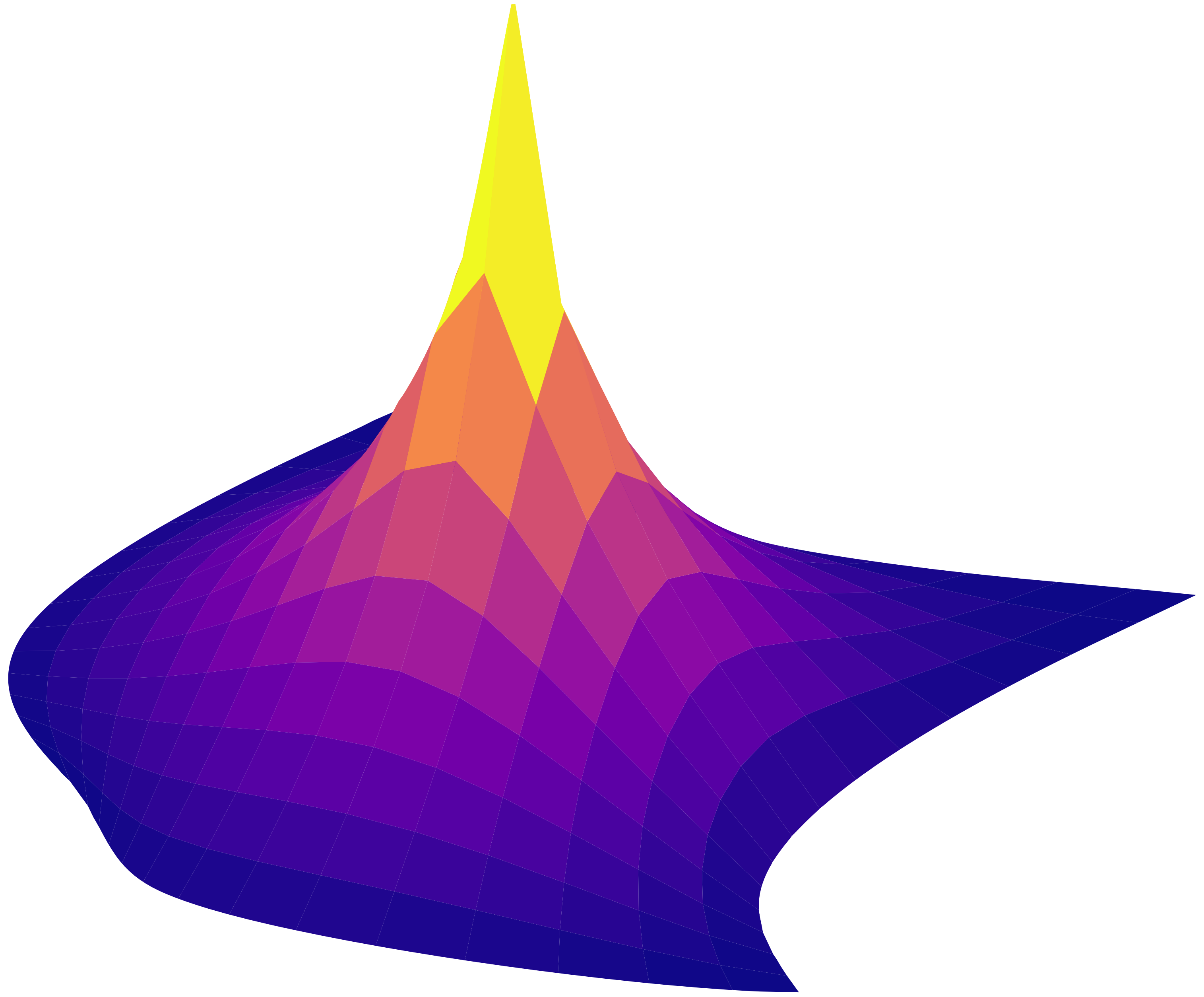
Ω_c

$$A = \begin{pmatrix} -1 & -1 & 4 & -1 & -1 \\ -2 & -1 & 6 & -1 & -2 \\ & & \vdots & & \\ -3 & -3 & 8 & -3 & -3 \end{pmatrix}$$

Problem setup

- Solving steady diffusion problem on some physical domain Ω_p

$$-\nabla \cdot (\mathcal{D}\nabla u) = f \quad \text{in } \Omega_p$$
$$u = 0 \quad \text{on } \partial\Omega_p$$



- Look at this through finite element lens: want to find u satisfying

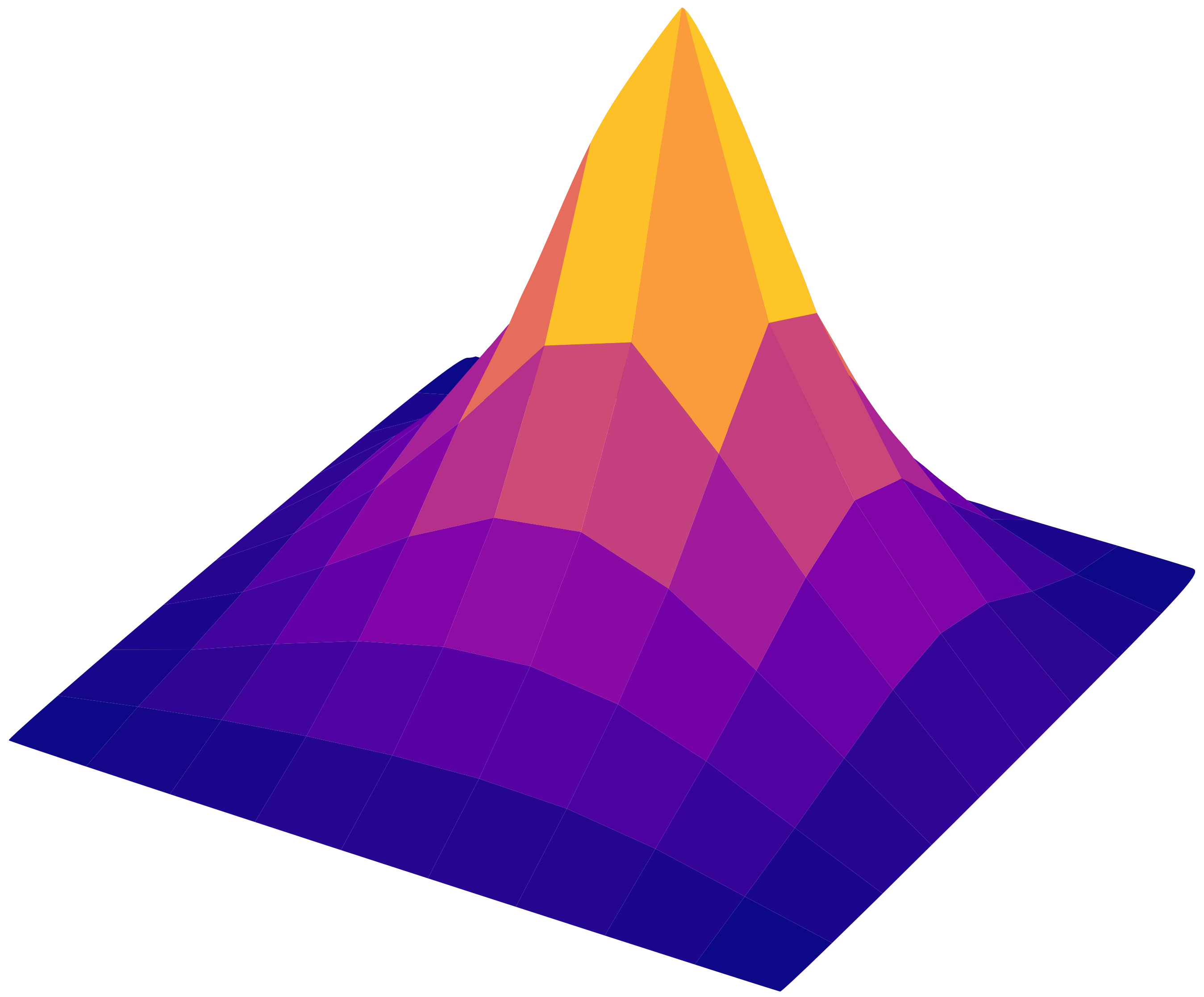
$$\int_{\Omega_p} \mathcal{D}\nabla u \cdot \nabla v \; dx = \int_{\Omega_p} f v \; dx \quad \forall v \in H_0^1(\Omega_p)$$

Transferring the weak form

- Take weak form and apply change of coordinates to computational domain

$$\int_{\Omega_c} \mathcal{D}(J_T^T \nabla u) \cdot (J_T^T \nabla v) \; |J_T| \; dx = \int_{\Omega_c} f v \; |J_T| \; dx \quad \forall v \in H_0^1(\Omega_c)$$

- (for Jacobian J_T of map T)
- Obtain weak form over computational domain that we can discretize



Recap thus far

- We have bilinear forms on both domains

$$a : V_p \times V_p \to \mathbb{R}$$
$$a_c : V_c \times V_c \to \mathbb{R}$$

- However, in a iterative solver, we have error corrections $e_c \in V_c$ and residual $r_p \in V_p'$ that we also want to transfer

$$Pe_c = e_p \in V_p$$
$$Rr_p = r_c \in V_c'$$

Interpolating between spaces

- Function interpolation: given $u_c \in V_c$ want to find nearest $u_p \in V_p$ …

$$\arg\min_{u_p} \frac{1}{2}\|u_p - u_c \circ T\|^2_{L^2(\Omega_p)}$$

$$\implies \sum_i^{N_c} u_c^i \int_{\Omega_p} (\phi_c^i \circ T)\phi_p^k \; dx = \sum_i^{N_p} u_p^i \int_{\Omega_p} \phi_p^i \phi_p^k \; dx \quad \forall \phi_p^k \in V_p$$

$$\implies Cu_c = M_p u_p \implies \boxed{P = M_p^{-1}C}$$

$$[C]_{ij} = \int_{\Omega_p} \phi_p^i (\phi_c^j \circ T) \; dx$$

$$[M_p]_{ij} = \int_{\Omega_p} \phi_p^i \phi_p^j \; dx$$

Restriction between spaces

- Function interpolation: given $u'_p \in V'_p$ want to find nearest $u'_c \in V'_c$

$$M_p u_p = u'_p \quad M_c u_c = u'_c$$

$$\arg\min_{u_c} \frac{1}{2}\|u_p \circ T^{-1} - u_c\|^2_{L^2(\Omega_c)}$$

$$\implies \sum_i^{N_c} u_c^i \int_{\Omega_c} \phi_c^i \phi_c^k \; dx = \sum_i^{N_c} u_c^i \int_{\Omega_p} (\phi_p^i \circ T^{-1}) \phi_c^k \; dx \quad \forall \phi_c^k \in V_c$$

$$\implies C^T u_p = M_c u_c \implies C^T M_p^{-1} u'_p = M_c M_c^{-1} u'_c$$

$$\implies \boxed{R = C^T M_p^{-1} = P^T}$$

Constructing a 2-grid cycle

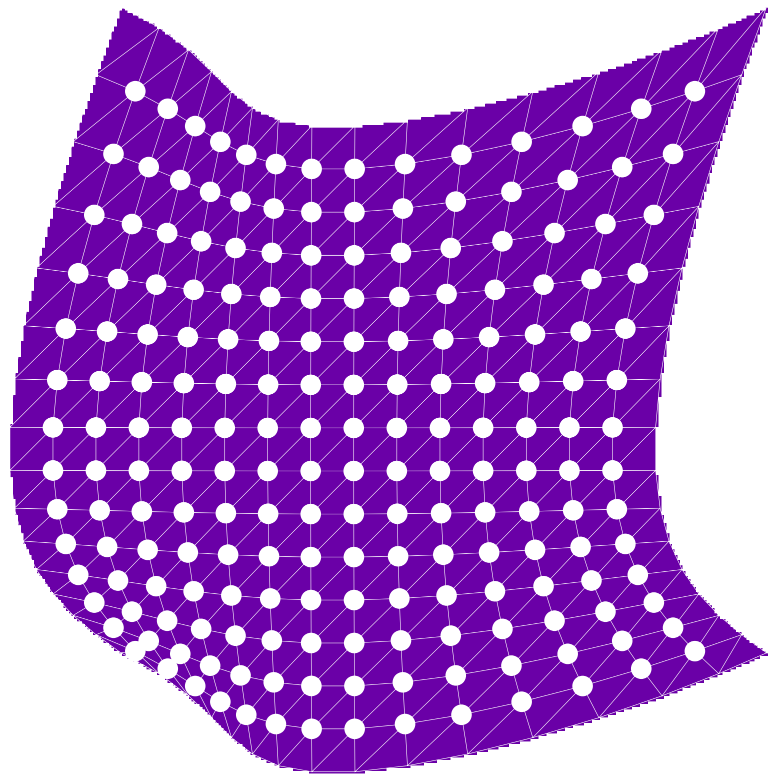
- We have:
  - Physical operator A
  - Computational operator A_c by discretizing a_c
  - Interpolation P
- Everything we need for a two grid cycle!

$$u \leftarrow \text{relax}(A, u, f)$$

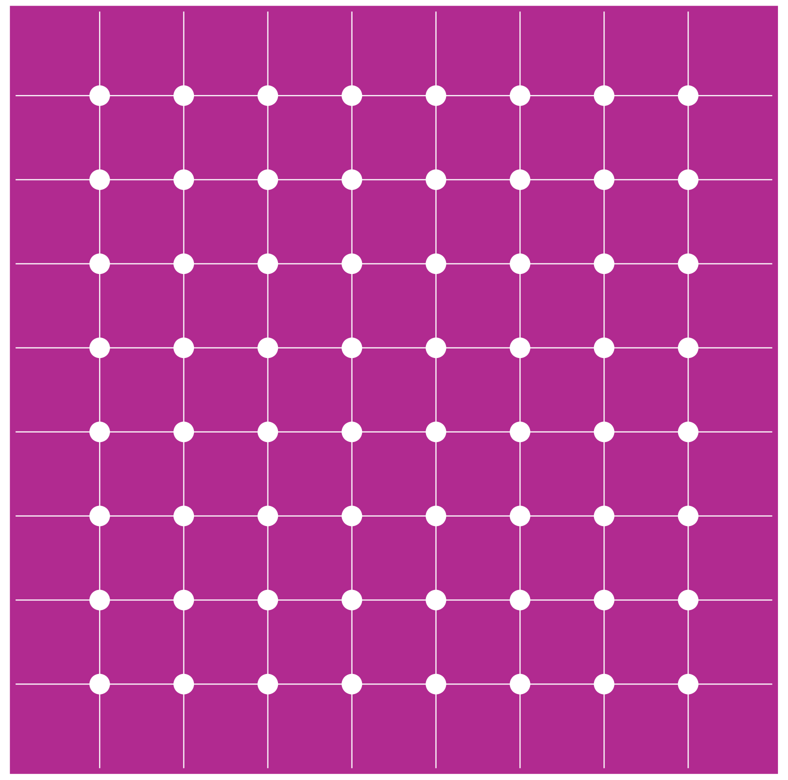
$$r = f - Au$$

$$u \leftarrow P A_c^{-1} P^T r$$

$$u \leftarrow \text{relax}(A, u, f)$$



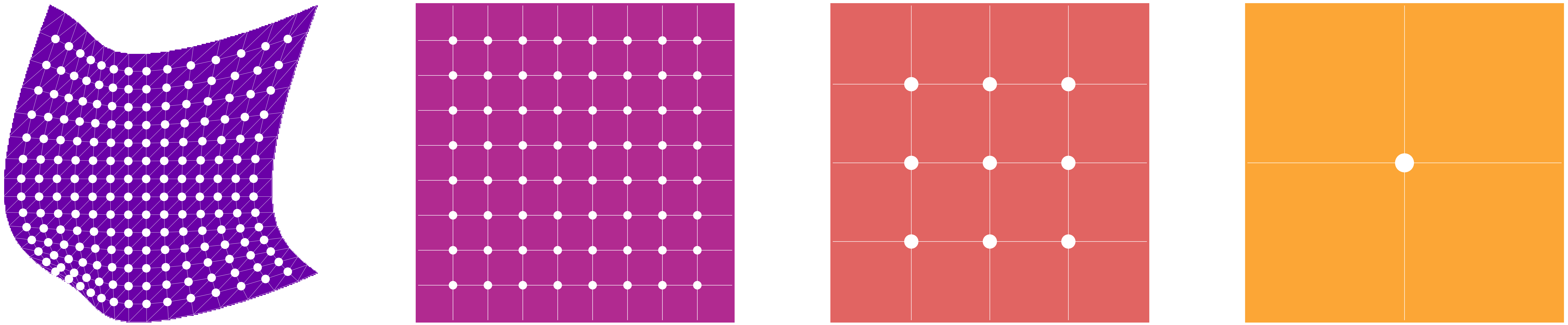
Ω_p



Ω_c

Putting it together into a V-cycle

- Recursively coarsen the computational grid
  - Use Black-Box MG for structured hierarchy[1]
- Obtain multigrid hierarchy with physical mesh as fine grid, structured hierarchy as coarse grids

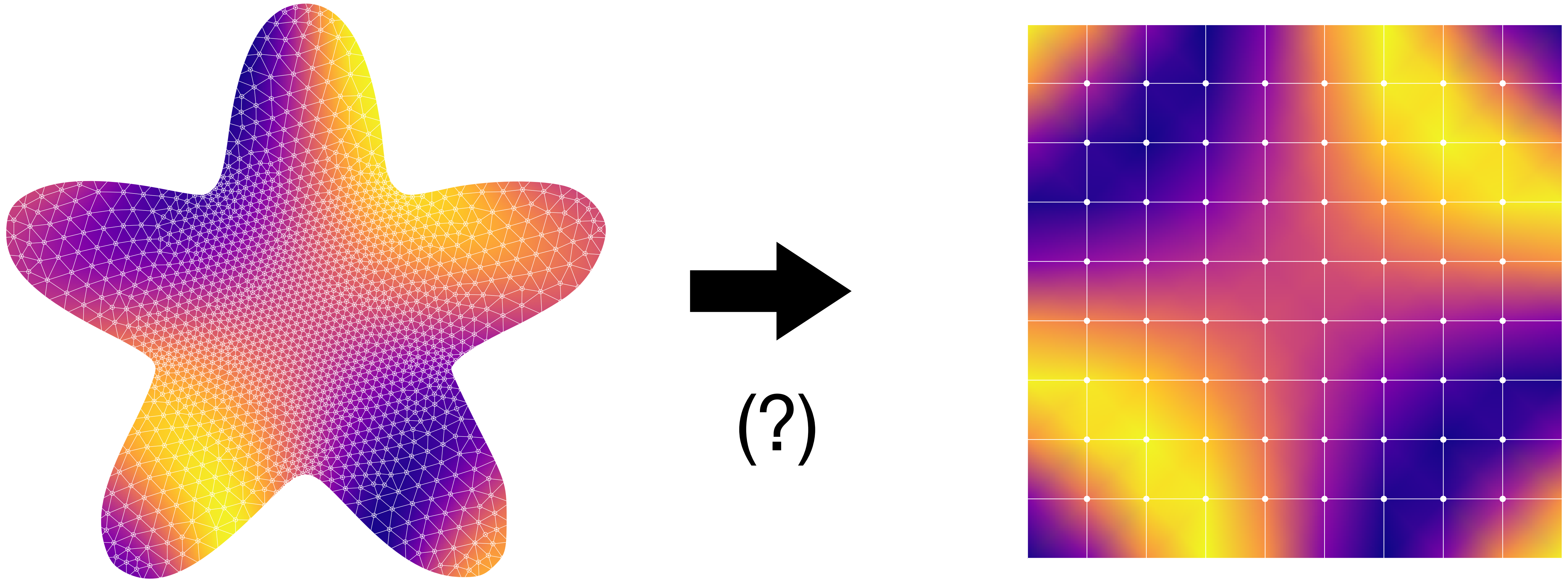


Ω_p Ω_c^1 Ω_c^2 Ω_c^3

[1]Reisner, A., Olson, L. N., & Moulton, J. D. (2018). Scaling Structured Multigrid to 500K+ Cores through Coarse-Grid Redistribution

Learning the map

- What about domains where we don’t know the mapping analytically?

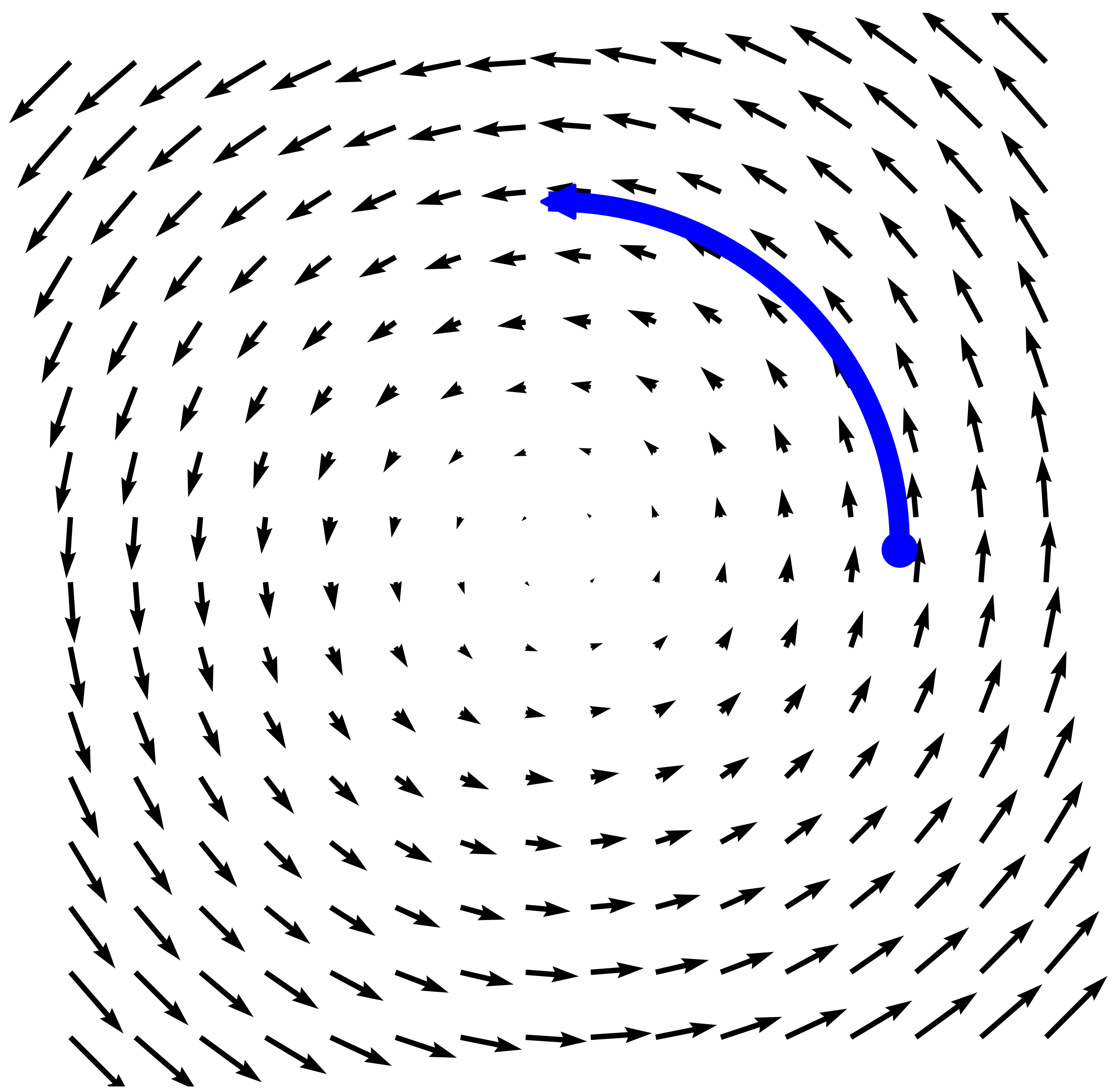


Motivation

- What do we want?
  - Continuous mapping between domains that is mesh invariant
  - Some sense of regularity (no ill conditioning, etc.)
  - Smooth and invertible (diffeomorphism)

Learning a diffeomorphism

- Observation: integrating over vector field gives diffeomorphism*

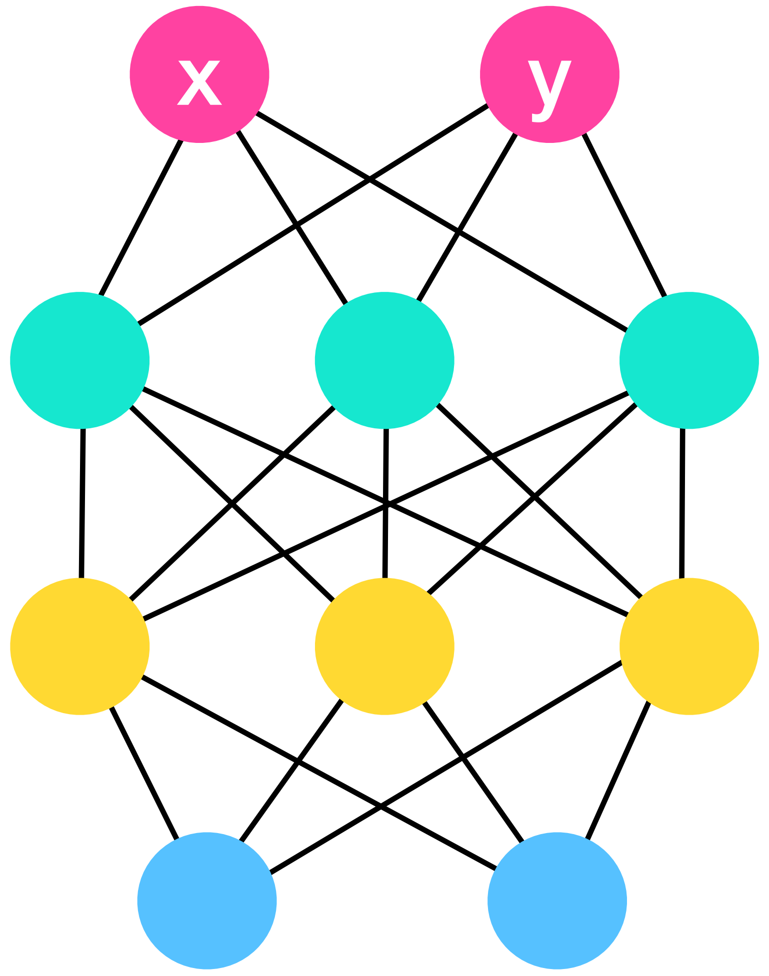


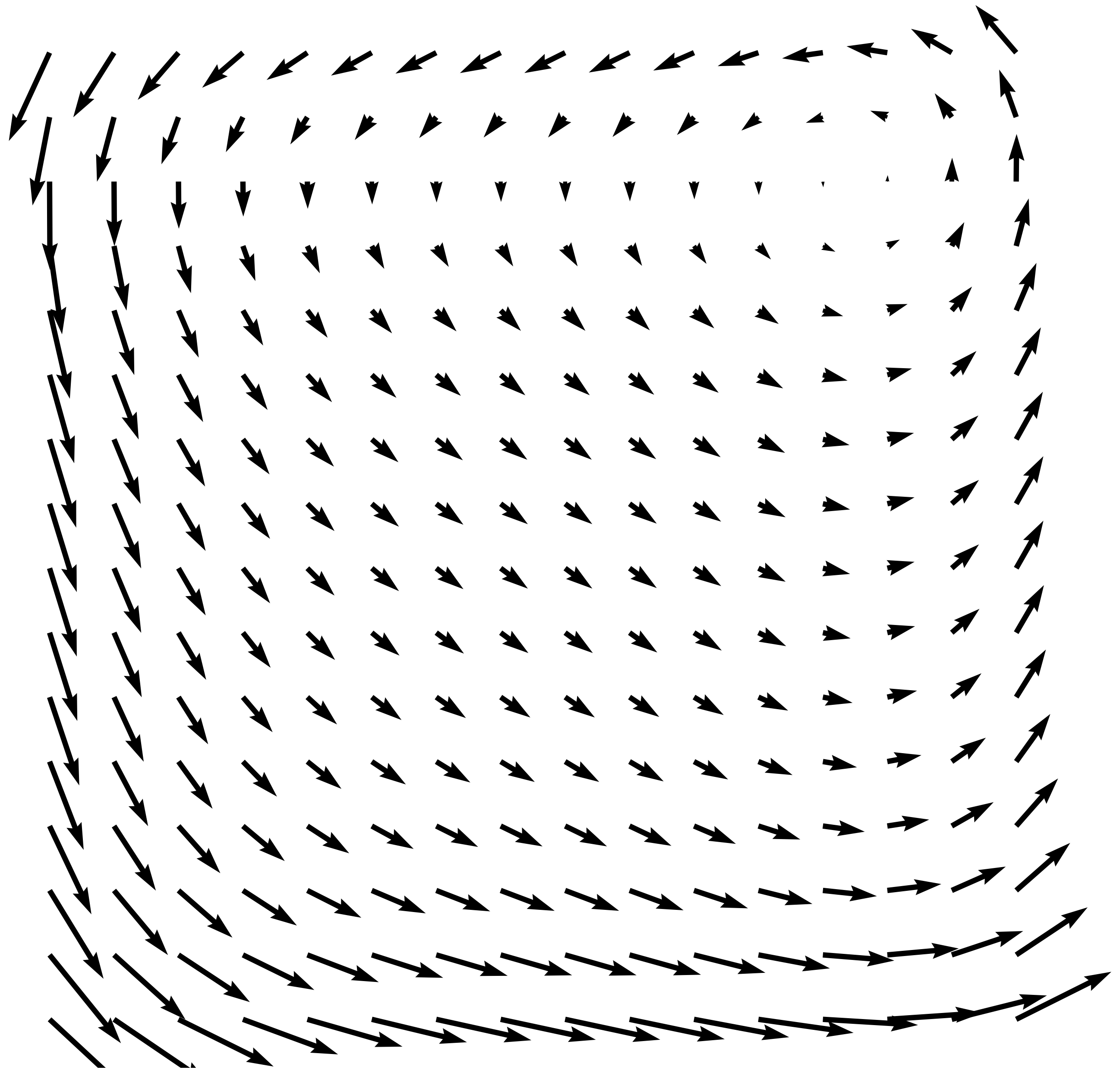
- * assuming Lipschitz, smooth, etc.

Neural ODEs

- Construct the vector field as the output of a neural network — this is a *neural ODE*[3] parameterized by some values θ

$$f_\theta : \mathbb{R}^2 \to \mathbb{R}^2$$





$$T_\theta(x) = z(1) = \int_0^1 f_\theta(z(t), t)\ dt$$

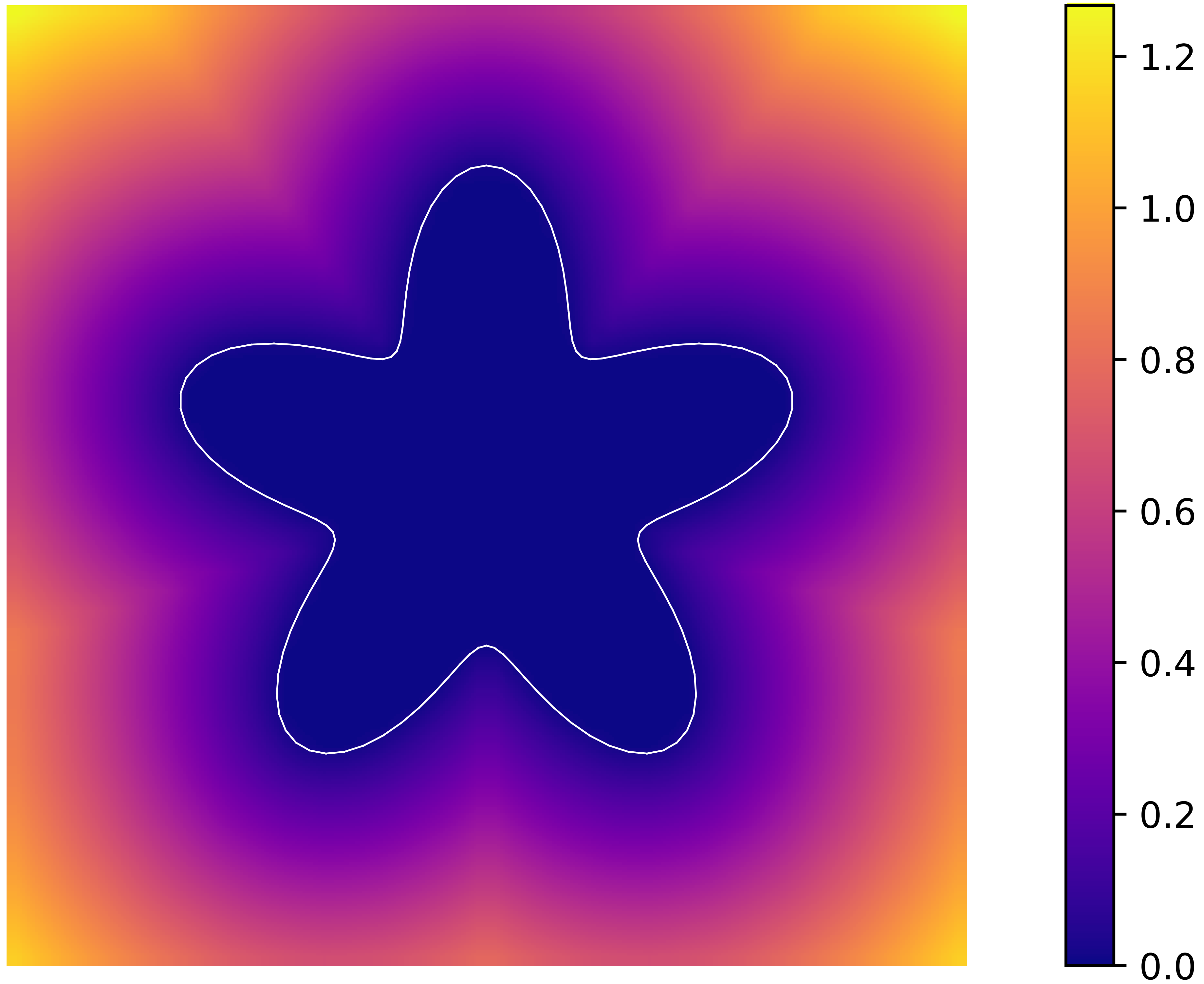
$$T_\theta^{-1}(y) = z(0) = \int_1^0 f_\theta(z(t), t)\ dt$$

[3]Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations. NIPs, 109(NeurIPS), 31–60. https://doi.org/10.48550/arxiv.1806.07366

Constructing a loss

- For both domains, define exterior distance function

$$I(x;\ \Omega) = \begin{cases} 0 & x \in \Omega \\ \inf_{y \in \partial\Omega} \|x - y\| & x \notin \Omega \end{cases}$$



- Want to preserve distance to boundary after mapping
- Removes the need for finding pairs of points on the two domains

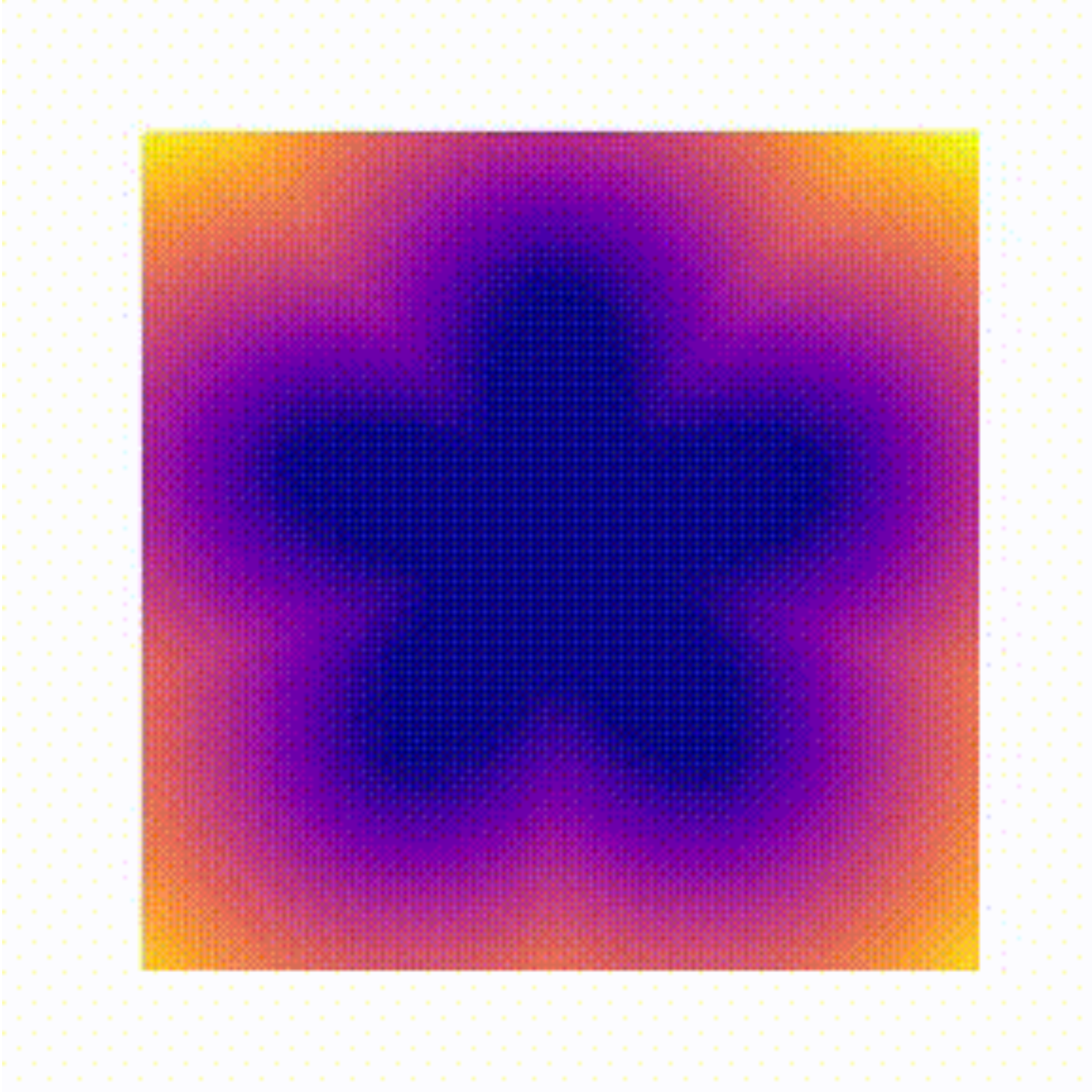
Loss function

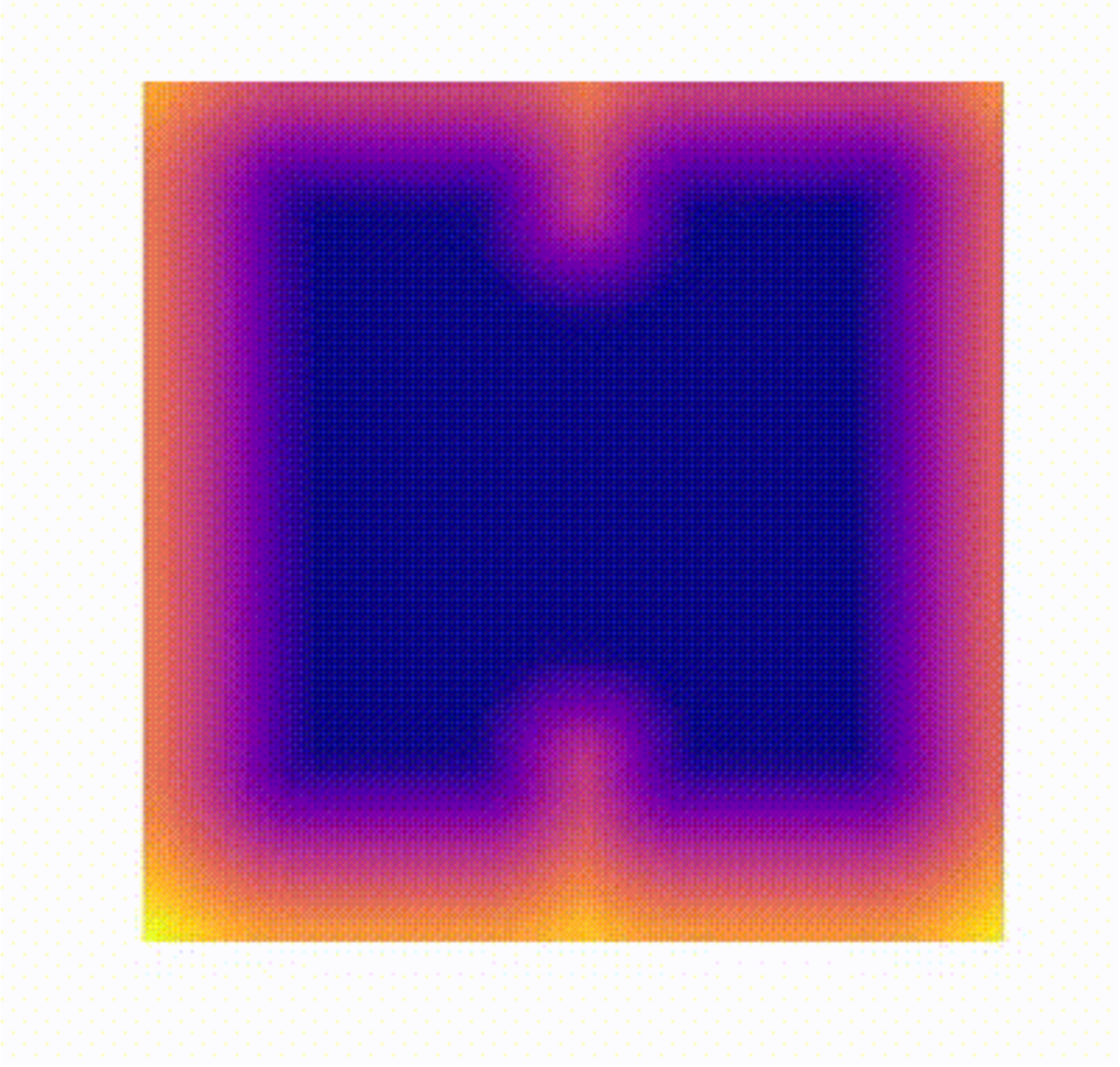
- Want to preserve distance to boundary after mapping

$$\ell(\theta) := \left\| I(T_\theta^{-1}(x); \Omega_p) - I(x, \Omega_c) \right\|^2 + \alpha \int_0^1 \left\| \frac{\partial f}{\partial x} \right\|^2 dt$$

Reconstruction loss

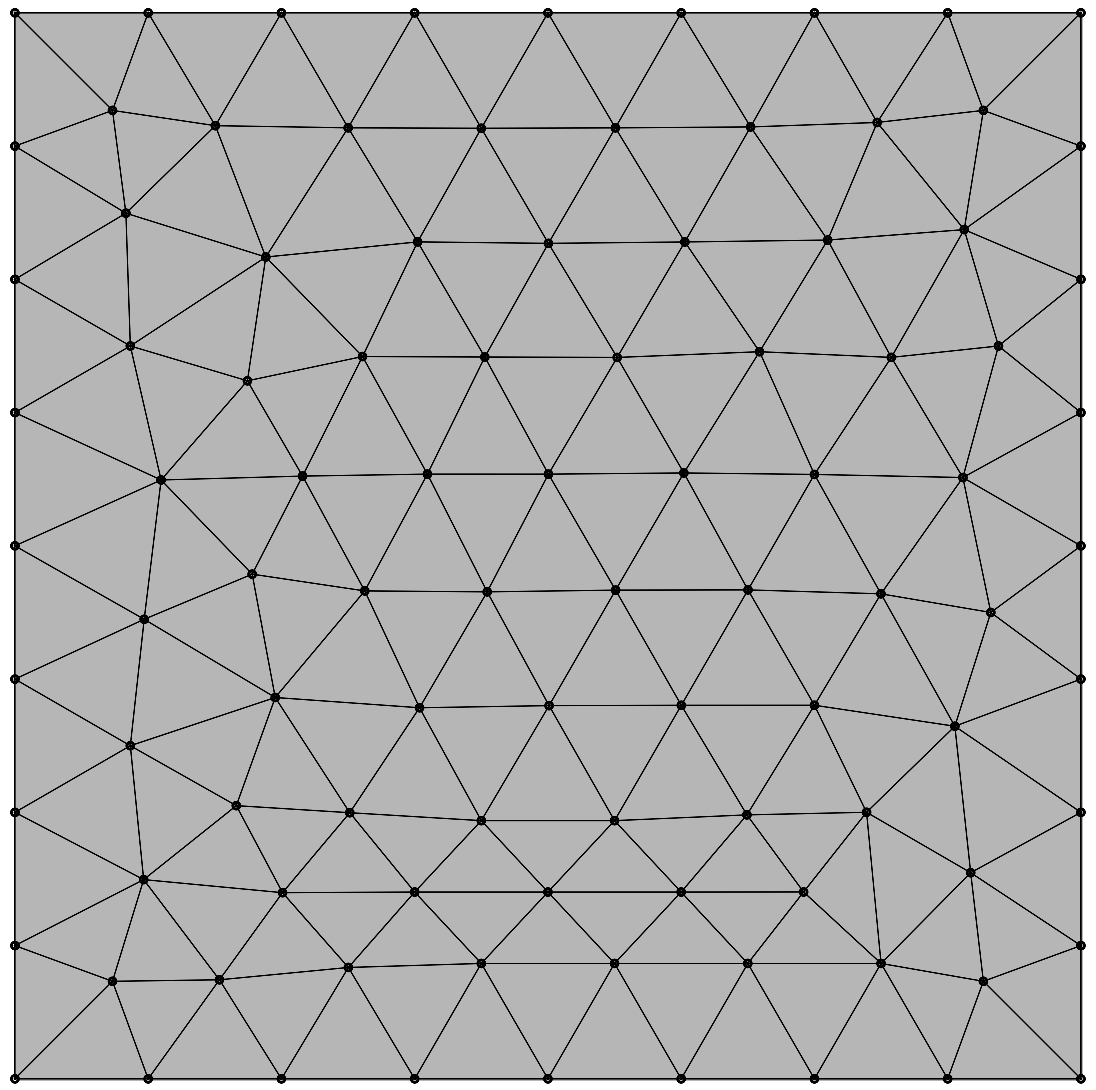
Jacobian regularization

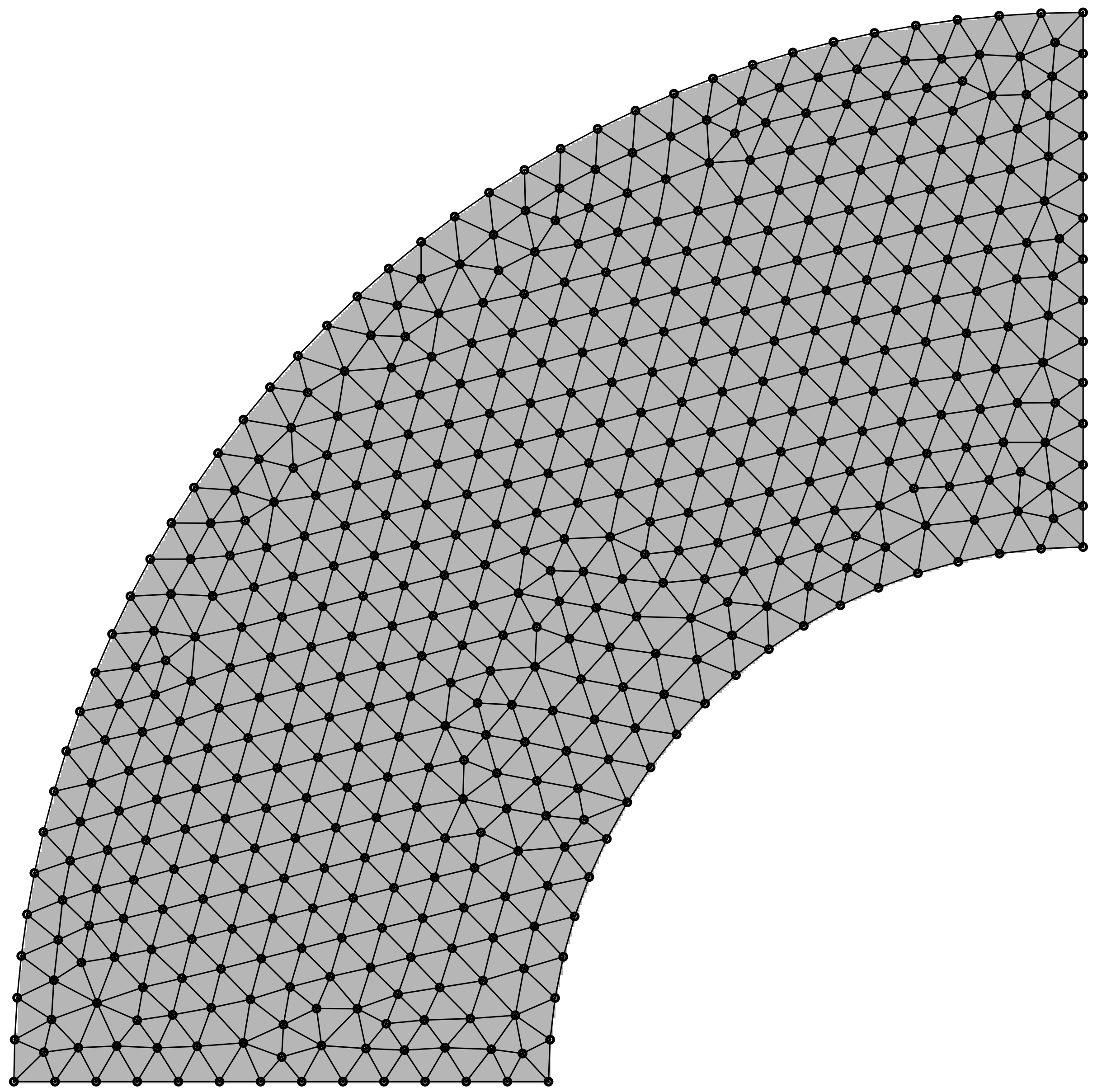




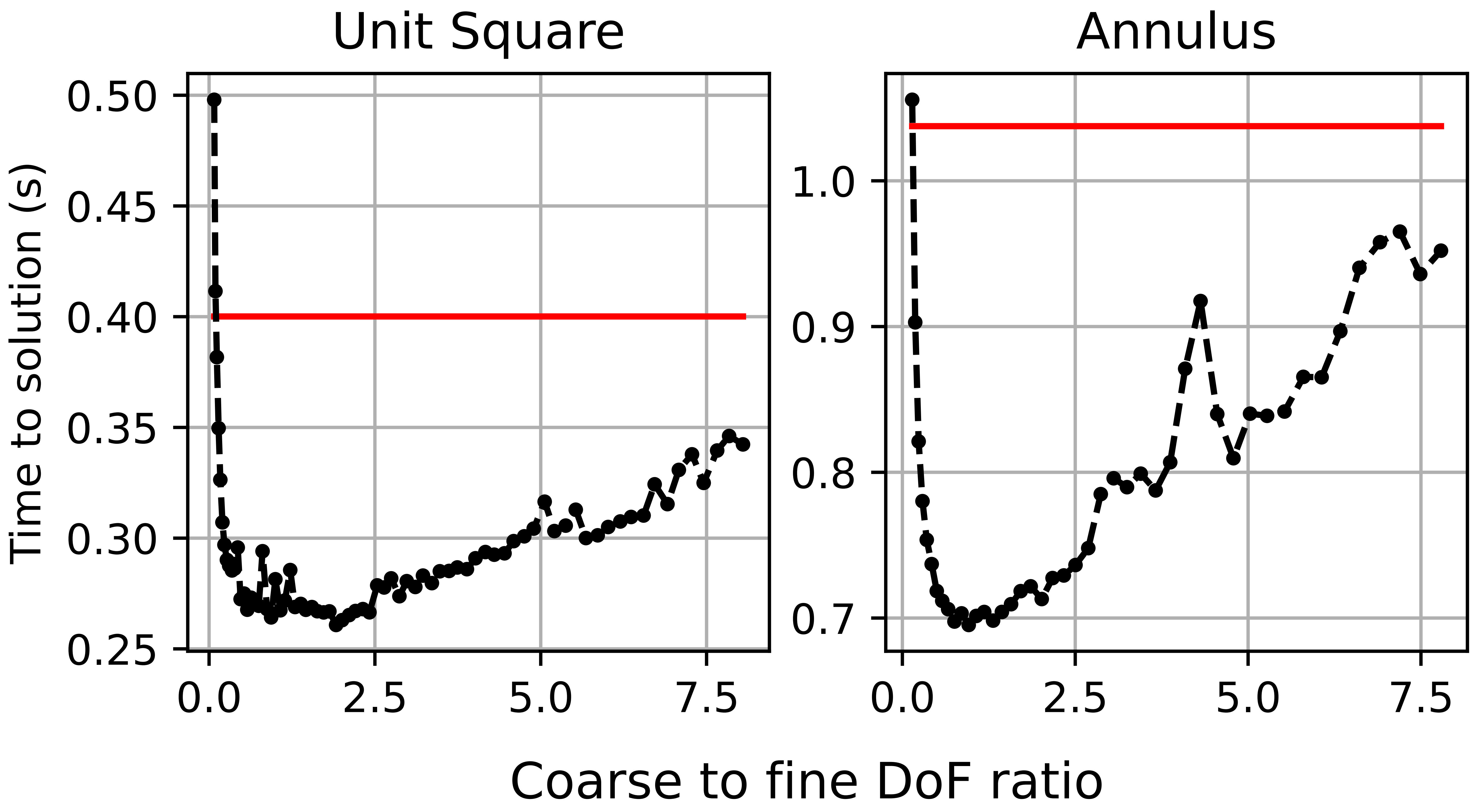
Analytic Results

- Set up solver on a few test domains where the mapping is known analytically
- Compared against serial BoomerAMG with default parameters
- Same relaxation on fine grid for both solvers



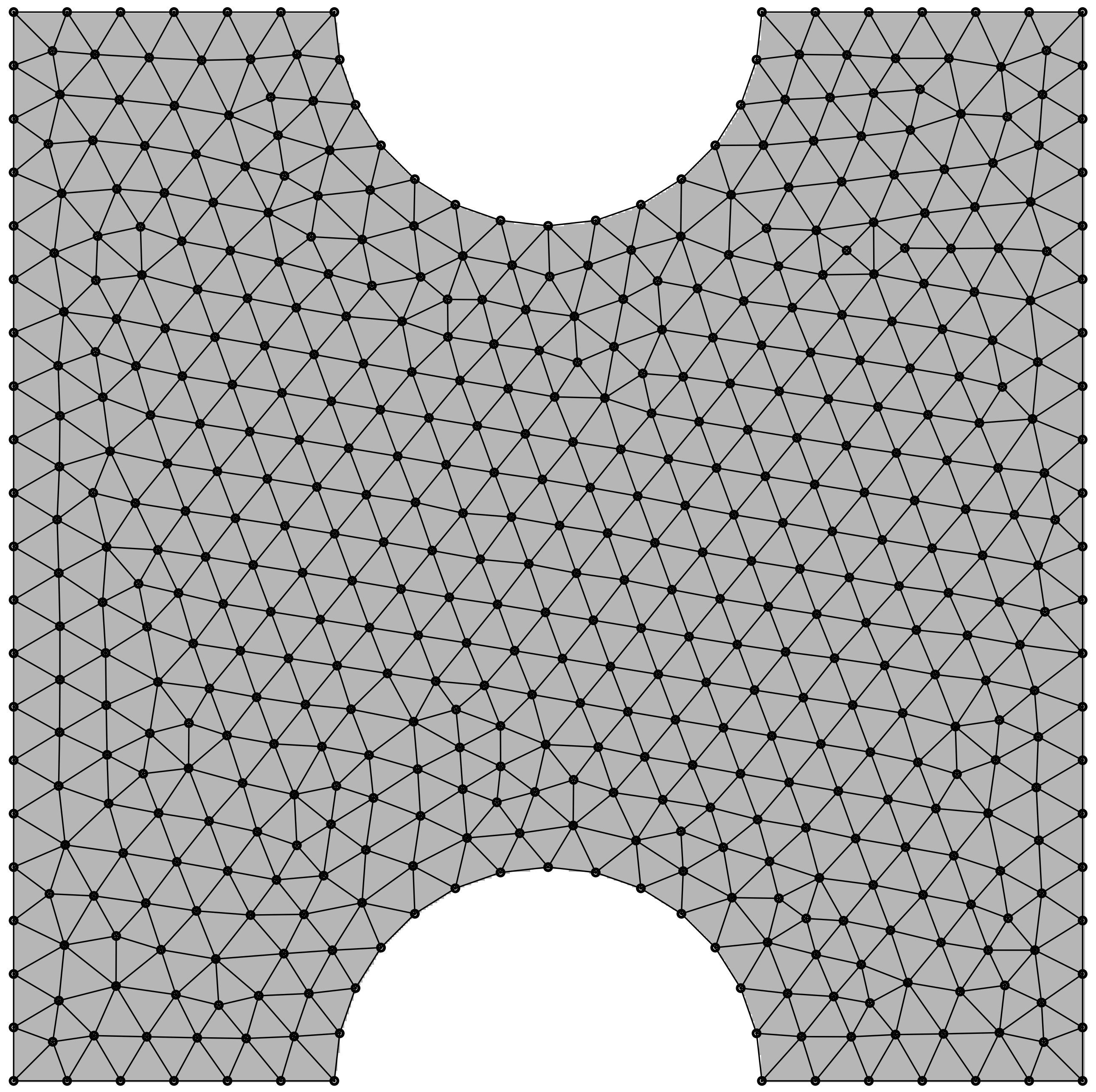
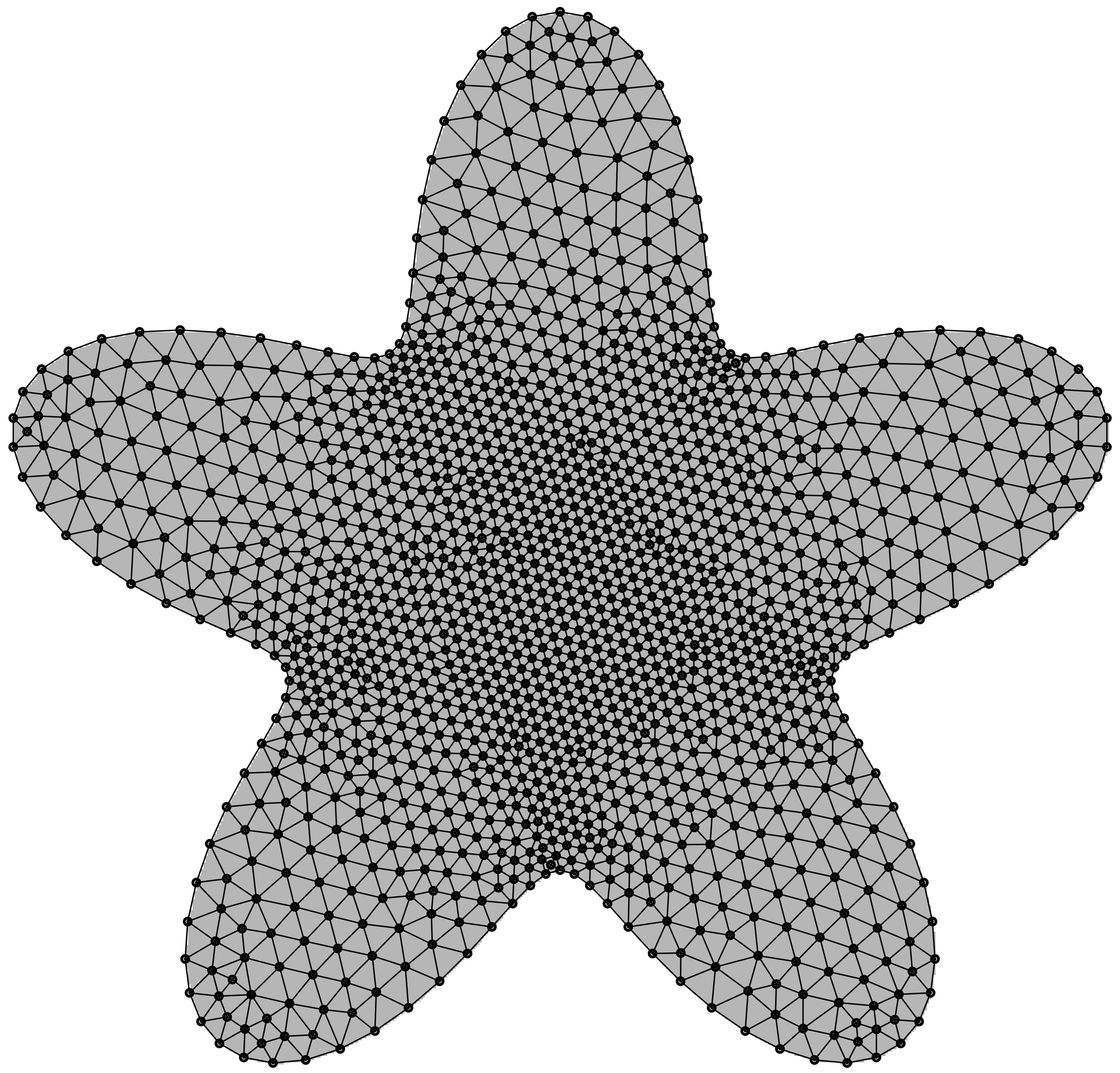


Results on analytic maps

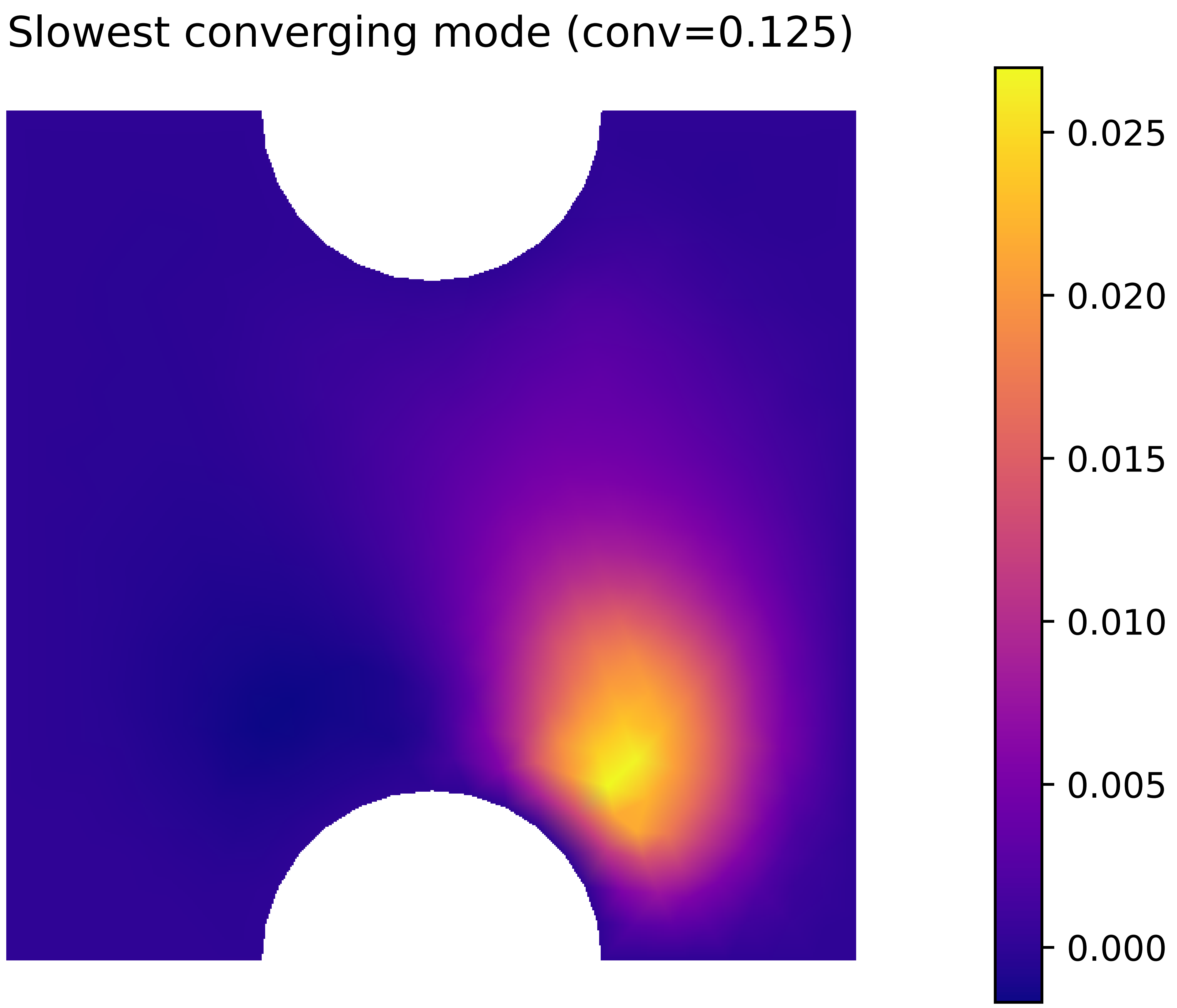


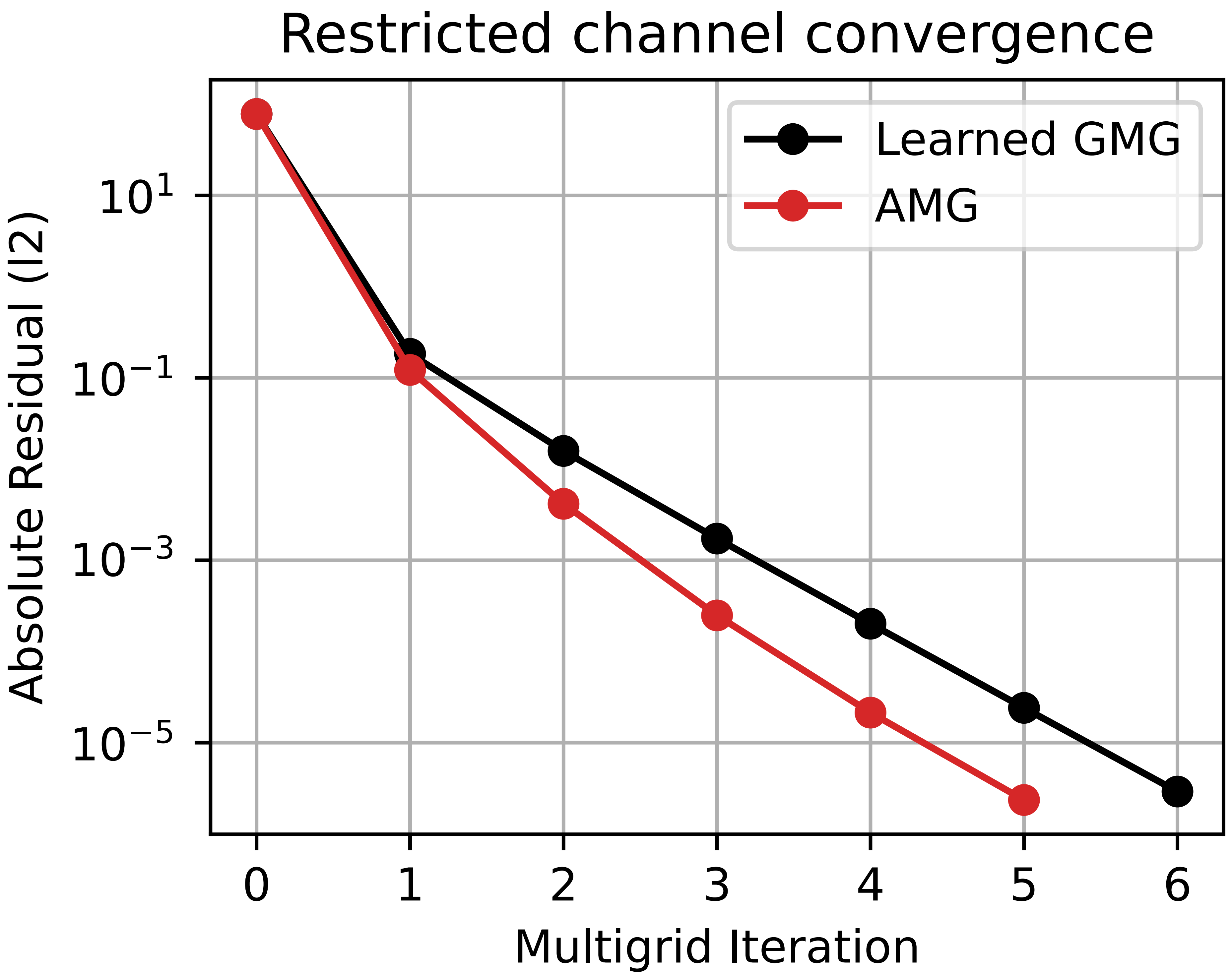
Learned Results
(Preliminary)

- Learned the mapping for a restricted channel and a star-shaped domain
- Looking at convergence rates for roughly coarsening by two

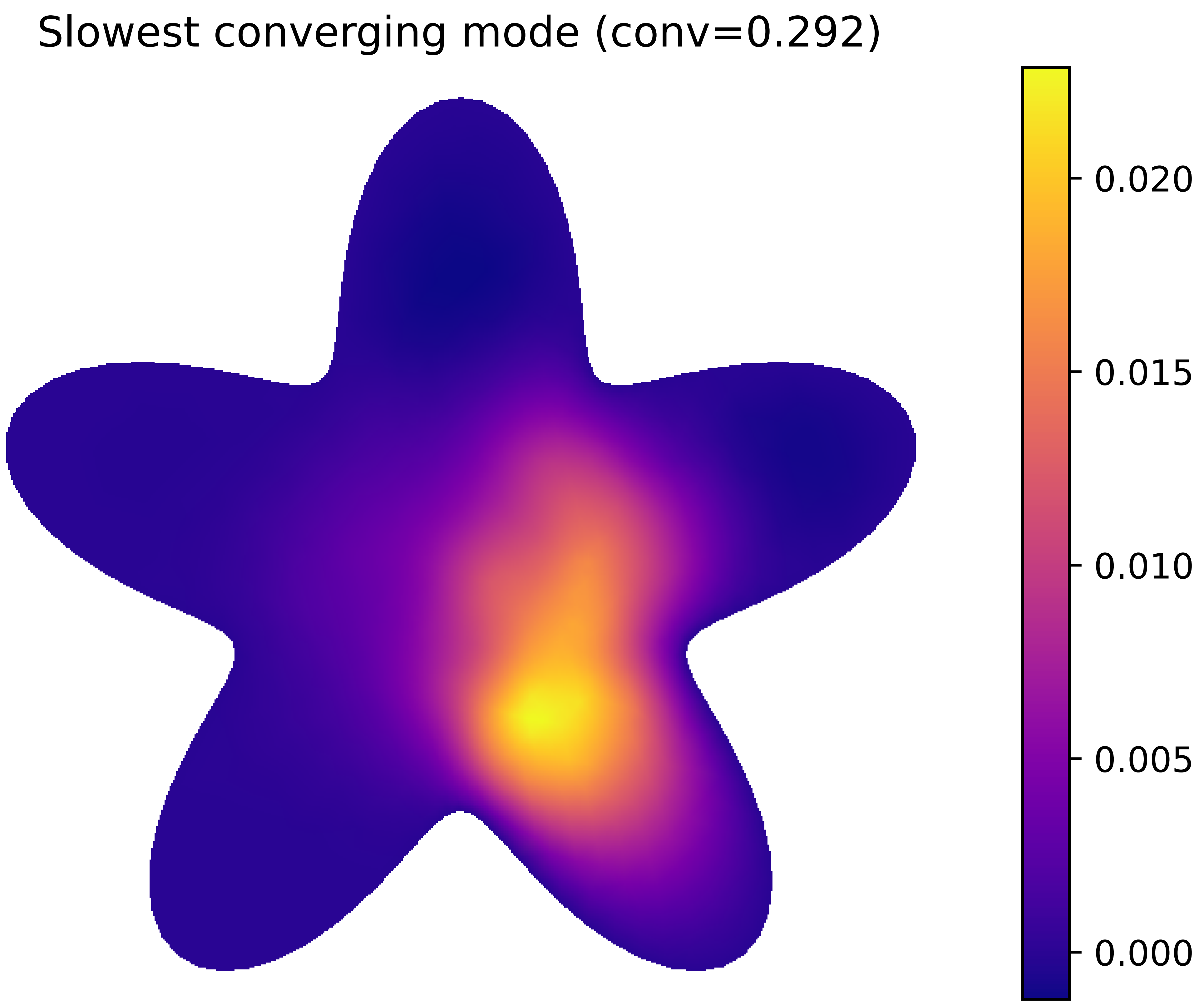



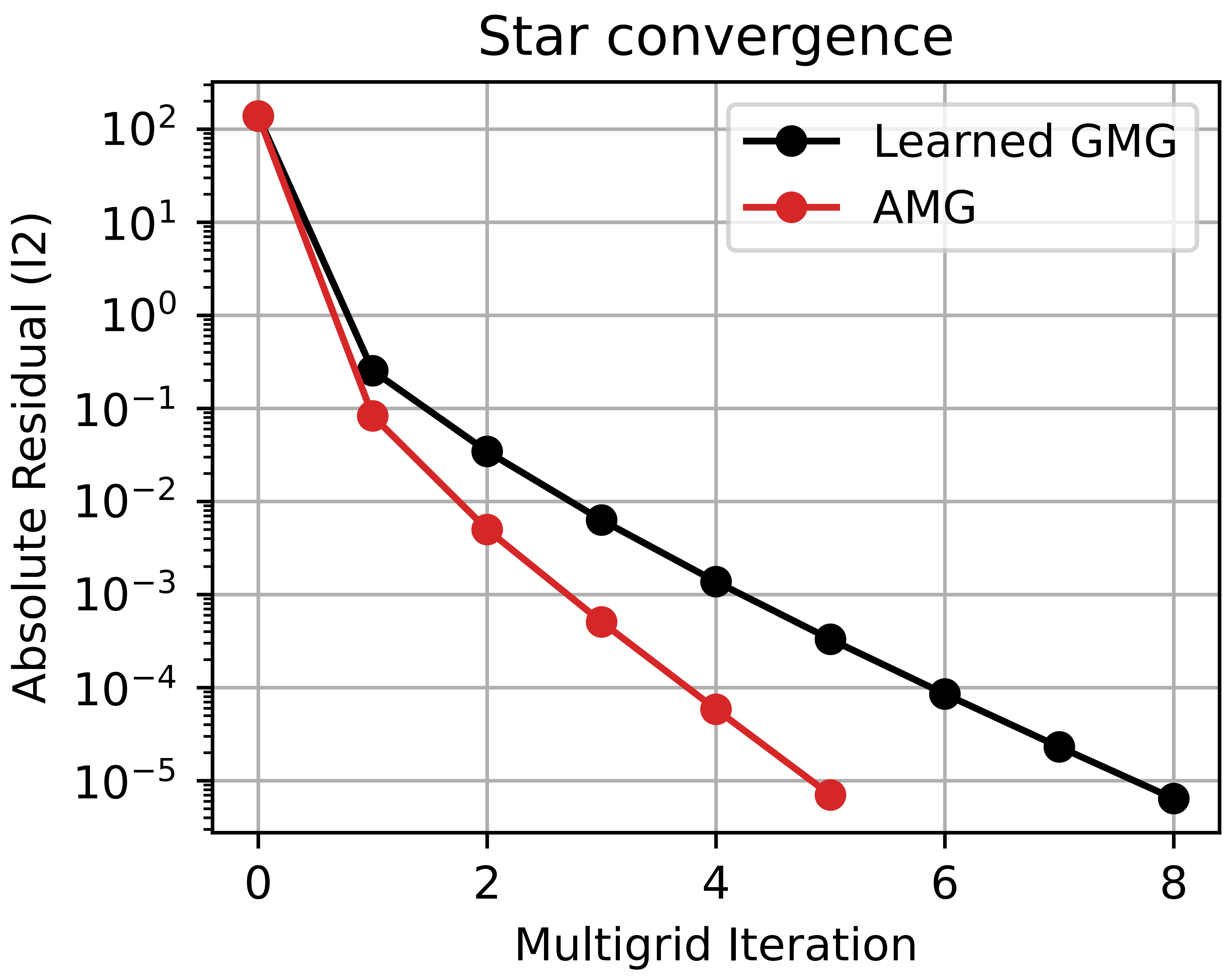
Restricted Channel





Smooth star


Slowest converging mode (conv=0.292)
0.020
0.015
0.010
0.005
0.000


Star convergence
Learned GMG
AMG
Absolute Residual (l2)
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
Multigrid Iteration

Conclusions

- Framework for geometric multigrid on complex geometries
- Wall-clock speedup against AMG on problems with analytic mappings
- Have proposed methodology for learning the mapping that acts on domains
- Future directions:
    - More work on learned mappings
    - Training and solve on GPU
    - Domain decomposition?
    - Matrix or mesh free on physical domain

- nnytko2@illinois.edu
- nicknytko.github.io

Network Architecture

- Vector field is time dependent
- Network architecture:
  - (3, 256, 256, 256, 2)
  - ReLU activation
- Training time: 10-20 minutes on my laptop (CPU)