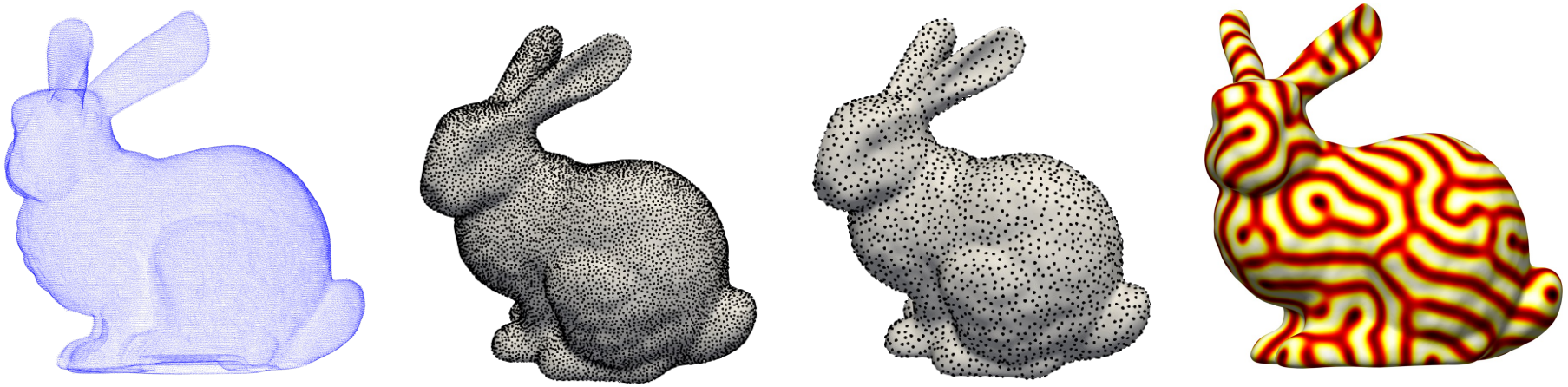
Computing on Point Cloud Surfaces:
PDEs, Multilevel Solvers, Quadrature, & Implicit Reconstruction



Grady B. Wright
Boise State University

Research Supported By NSF grant DMS 2309712

Collaborators

Varun Shankar
University of Utah

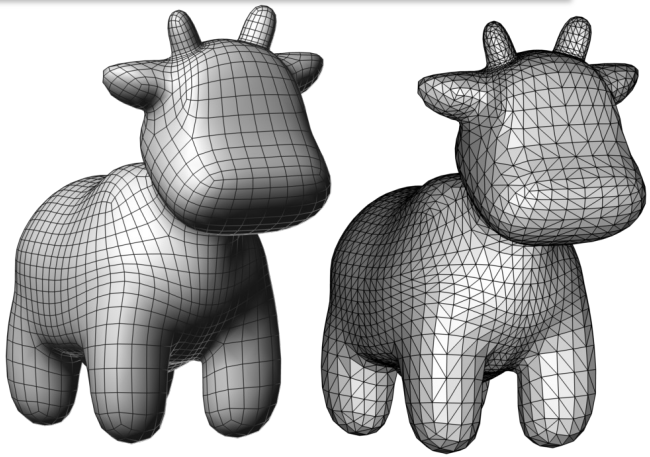
Andrew Jones
Sandia

Kathryn Drake
Apple

Meshes vs Point Cloud Surfaces

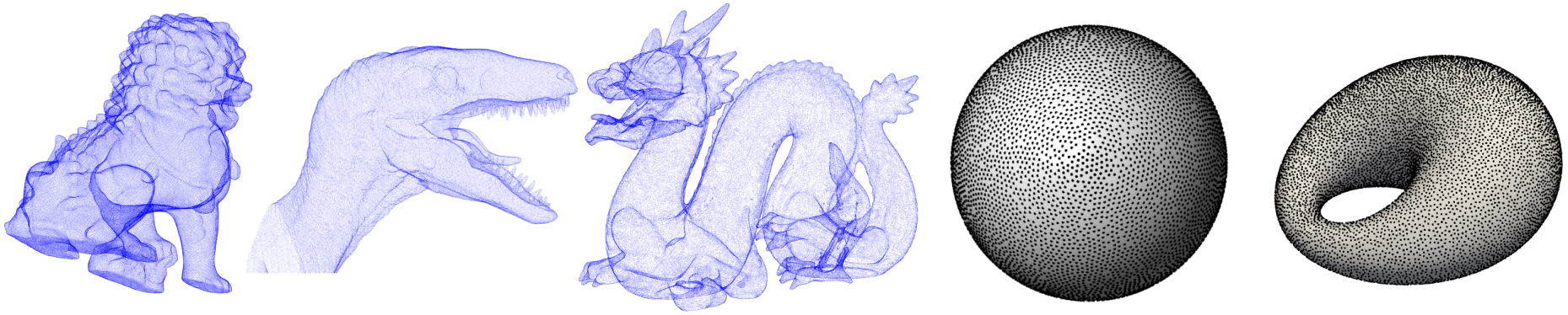
Mesh: a set of points that connects points into faces

(e.g., triangles or quads), forming a continuous surface.



- Great for computing when available:
  Surface finite elements (Dziuk & Elliot, 2013)
  Surface spectral elements (Fortunato 2024)

Point Cloud: a set of unstructured set of points without connectivity

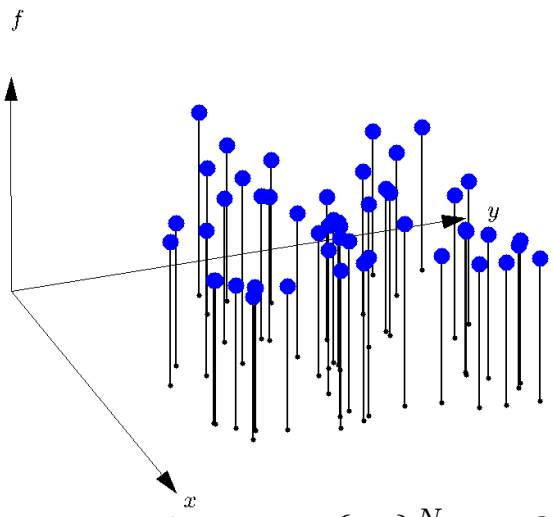


- More flexible representation for 3D geometry, especially for complex or data-defined shapes
- Easy to generate large point point sets
- No issues with mesh imprinting
- More difficult to develop high-order, scalable methods

Aim: Overview of a kernel-based framework for addressing these challenges

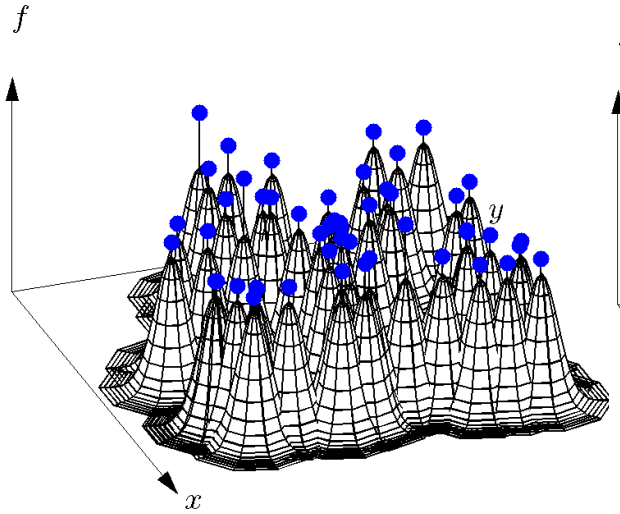
Kernel-based approximation

Interpolation with radial kernels: $\kappa(\mathbf{x}, \mathbf{y}) = \phi(\|\mathbf{x} - \mathbf{y}\|)$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$

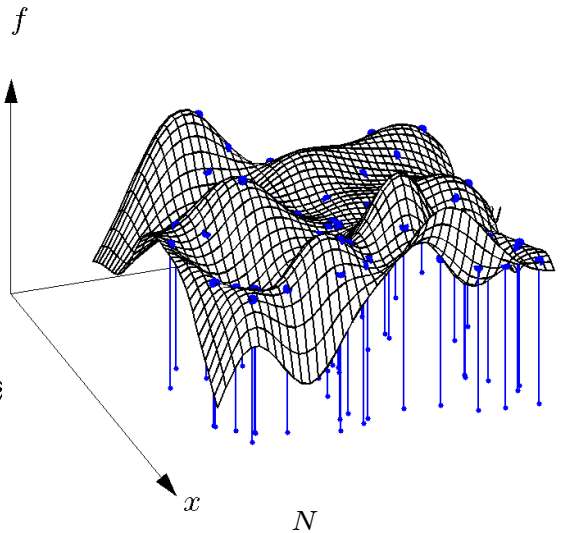


Nodes: $X = \{\mathbf{x}_j\}_{j=1}^N \subset \Omega$

Data: $\{f(\mathbf{x}_j)\}_{j=1}^N$



Collocation



$$s(\mathbf{x}) = \sum_{j=1}^{N} a_j \kappa(\mathbf{x}, \mathbf{x}_j),$$

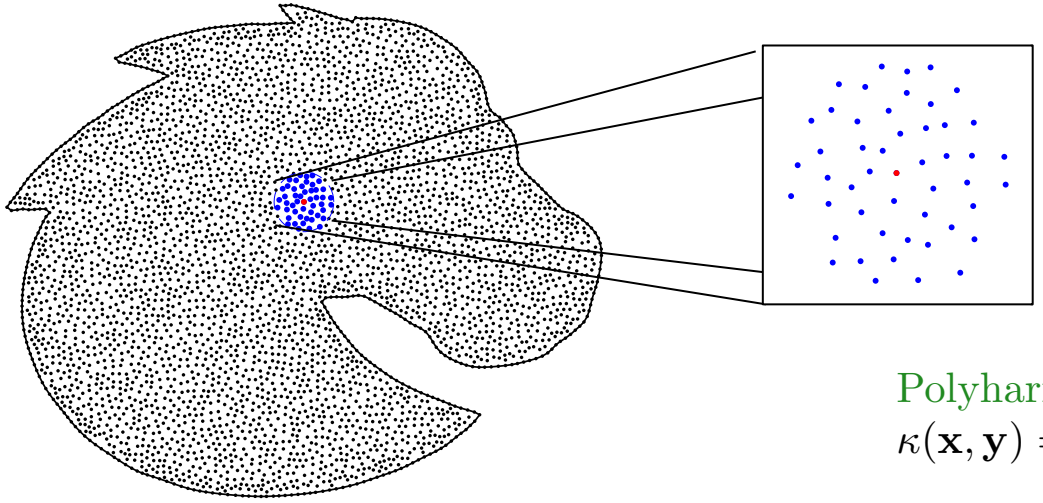
Linear system for expansion coefficients

$$\underbrace{\begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) \cdots \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) \cdots \kappa(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots \quad \ddots \quad \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \kappa(\mathbf{x}_N, \mathbf{x}_2) \cdots \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}}_{K_{XX}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}}_{a_X} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{f_X}$$

Similarities to Gaussian Processes and Kriging (Banff Workshop 26w5635, Fall 2026)

Localized approximation using polyharmonic spline kernels

Construct kernel approximations over local stencils of $X = \{\mathbf{x}_j\}_{j=1}^N \subset \Omega$



Stencil $X_i \subseteq X$ for center $\mathbf{x}_i$
with $n << N$ points

Polyharmonic spline (PHS) kernel:
$\kappa(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^{2\ell+1}, \quad \ell \geq 0$

Augmented PHS kernel expansion with bivariate polynomials $\{p_k\}$ of degree ℓ

$$s(\mathbf{x}) = \sum_{j=1}^{n} a_j \kappa(\mathbf{x}, \mathbf{x}_j) + \sum_{k=1}^{L(\ell)} b_k p_k(\mathbf{x})$$

(Iske 2003; Bayona, Flyer, Fornberg, Barnett 2017)

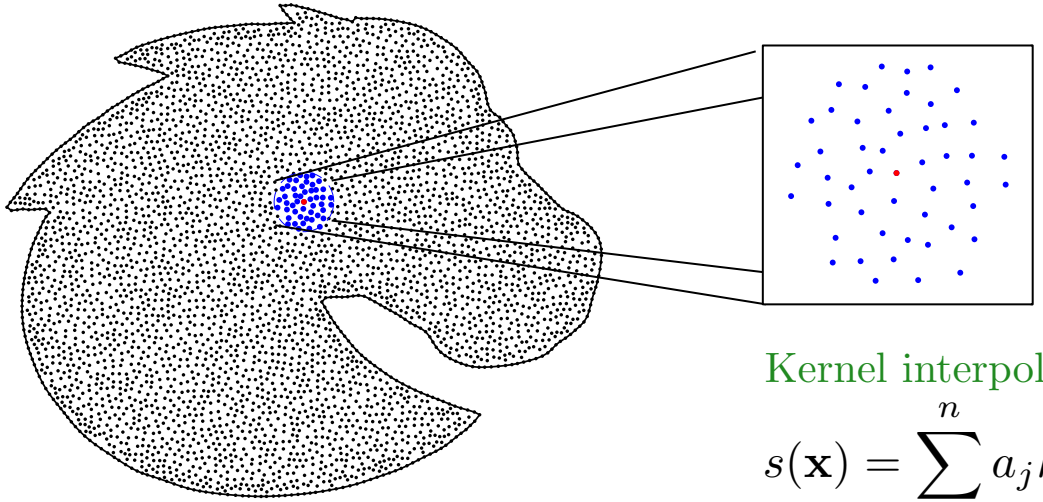
Enforce interpolation $s\big|_{X_i} = f\big|_{X_i}$, and moment conditions $\sum_{j=1}^{n} a_j p_k(\mathbf{x}_j) = 0$

$$\Longrightarrow \begin{bmatrix} K_{X_i X_i} & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} a_{X_i} \\ b_{X_i} \end{bmatrix} = \begin{bmatrix} f_{X_i} \\ 0 \end{bmatrix}$$

System is non-singular under mild restrictions on X_i (Duchon 1977)

Kernel-based Finite Difference (FD) Method

Construct kernel approximations over local stencils of $X = \{\mathbf{x}_j\}_{j=1}^N \subset \Omega$



Stencil $X_i \subseteq X$ for center $\mathbf{x}_i$
with $n << N$ points

Kernel interpolant:

$$s(\mathbf{x}) = \sum_{j=1}^{n} a_j \kappa(\mathbf{x}, \mathbf{x}_j) + \sum_{k=1}^{L(\ell)} b_k p_k(\mathbf{x})$$

Idea: Use $s(\mathbf{x})$ to produce local approximations to differential operators over X_i

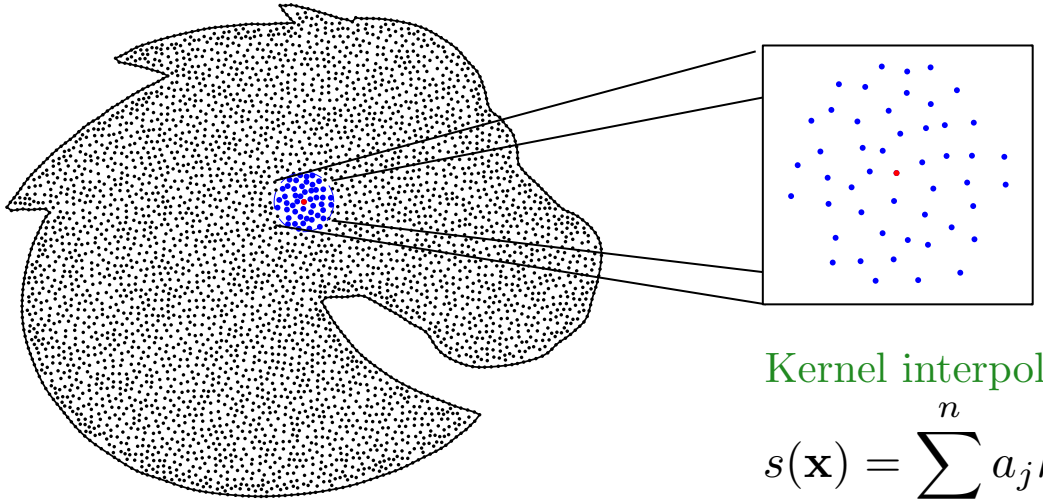
$$\mathcal{L}u|_{\mathbf{x}_i} \approx c_1 u(\mathbf{x}_1) + c_2 u(\mathbf{x}_2) + \cdots c_n u(\mathbf{x}_n)$$

Generalization of finite difference (FD) method to scattered points and kernels

RBF-FD Method: Tolstykch & Shirobokov (2003), Shu, Ding, & Yeo (2003), W (2003), Fornberg & Flyer (2015), Flyer, Fornberg, Barnett (2017)

Kernel-based Finite Difference (FD) Method

Construct kernel approximations over local stencils of $X = \{\mathbf{x}_j\}_{j=1}^N \subset \Omega$



Stencil $X_i \subseteq X$ for center $\mathbf{x}_i$
with $n << N$ points

Kernel interpolant:

$$s(\mathbf{x}) = \sum_{j=1}^{n} a_j \kappa(\mathbf{x}, \mathbf{x}_j) + \sum_{k=1}^{L(\ell)} b_k p_k(\mathbf{x})$$

Idea: Use $s(\mathbf{x})$ to produce local approximations to differential operators over X_i

$$\mathcal{L}u\big|_{\mathbf{x}_i} \approx c_1 u(\mathbf{x}_1) + c_2 u(\mathbf{x}_2) + \cdots c_n u(\mathbf{x}_n)$$

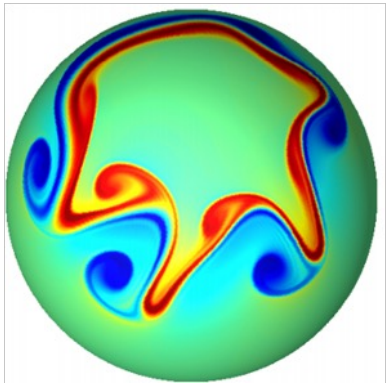
Solve a small system for the weights c_j

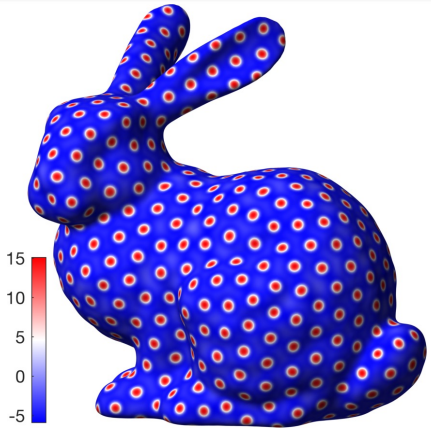
$$\Longrightarrow \begin{bmatrix} K_{X_i X_i} & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} c_{X_i} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathcal{L}\kappa(\mathbf{x}, \mathbf{x}_j)\big|_{\mathbf{x}_i} \\ \mathcal{L}p_k(\mathbf{x})\big|_{\mathbf{x}_i} \end{bmatrix}$$

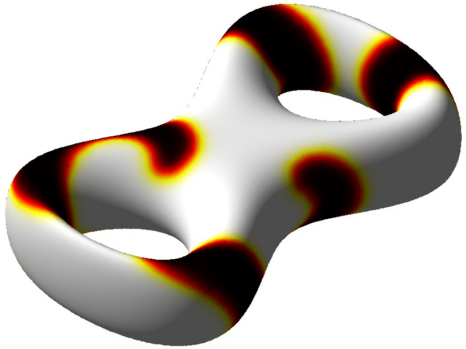
Polynomial degree ℓ controls the accuracy of the formulas (Davydov & Schaback 2019)

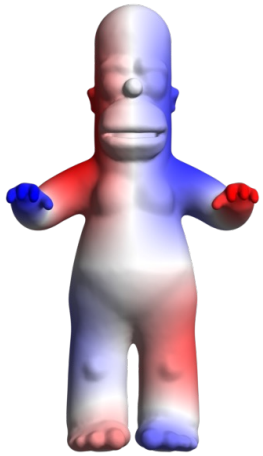
Solving Surface PDEs

Partial Differential Equations (PDEs) on Surfaces









Applications

Geophysics: numerical weather/climate prediction

Biology: advective/diffusive transport on a membrane, morphogenesis, pattern formation

Chemistry: waves in excitable media (cardiac arrhythmia, electrical signals in the brain)

Computer graphics: texture mapping, shape analysis, image processing

Prototypical PDE model:

$$(\mu\mathcal{I} - \Delta_{\mathcal{M}})u = f$$

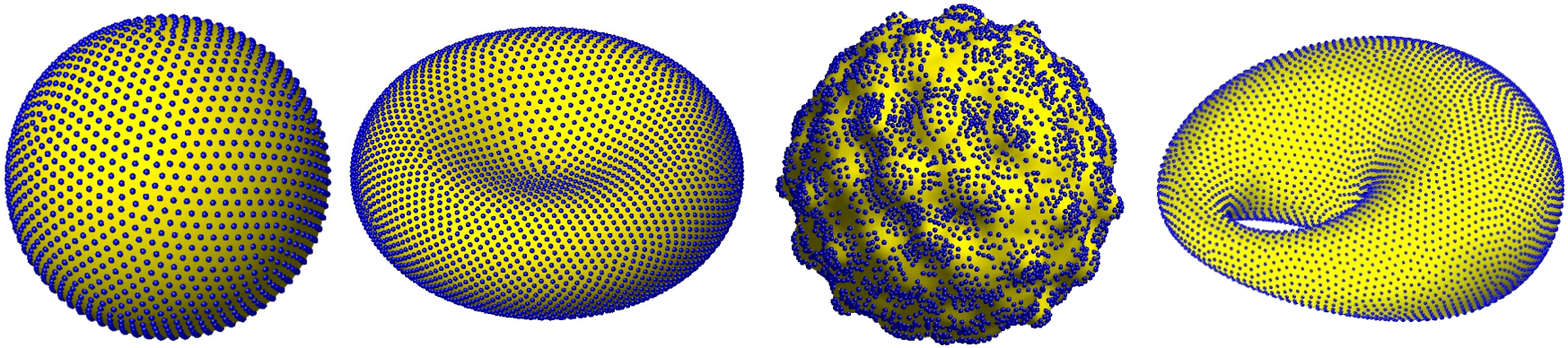
$$\begin{cases} \dfrac{\partial u}{\partial t} + \mathbf{w}\cdot\nabla_{\mathcal{M}}\, u = \delta_1 \Delta_{\mathcal{M}}\, u + f_1(t,u,v) \\ \dfrac{\partial v}{\partial t} + \mathbf{w}\cdot\nabla_{\mathcal{M}}\, v = \delta_2 \Delta_{\mathcal{M}}\, v + f_2(t,u,v) \end{cases}$$

$\mathcal{M}$ is a smooth embedded submanifold of co-dimension one in $\mathbb{R}^3$ with no boundary

$\Delta_{\mathcal{M}}$ is the Laplace-Beltrami operator

$\nabla_{\mathcal{M}}$ is the surface gradient

Kernel-based methods



Highlights:

- No mesh is required, just a point cloud
- Do not need additional points in the embedding space
- Can produce high orders of accuracy for smooth solutions

References

Global kernel methods and local finite difference methods

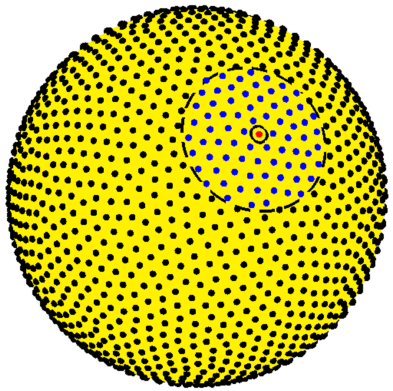
Piret (2012), Fuselier+W (2012, 2013), Shankar+W+Fogelson+Kriby (2014, 2015), Flyer+W+Fornberg (2015), Piret+Dunn (2016), Lehto+Shankar+W (2017), Shankar+Narayan+Kirby (2018), Shankar+W (2018), Petras+Ling+Ruuth (2018), Sokolov et. al. (2019), Petras et. al. (2019), Wendland+Künemund (2020), Shankar+W+Narayan (2020), Álvarez et. al. (2021), W+Jones+Shankar (2023), Jones et. al. (2023)

Generalized finite difference and moving least squares methods

Liang+Zhao (2013), Suchde+Kuhnert (2019), Trask+Kuberry (2020), Gross et. al. (2020), Hangelbroek+Rieger+W (2025)

Localized kernel-based methods for Laplace-Beltrami operator

Let $X_h = \{\mathbf{x}_i\}_{j=1}^{N_h} \subset \mathcal{M}$, where $\mathbf{x}_i = (x_i, y_i, z_i)$



- For each $\mathbf{x}_i \in X_h$, let σ_i^h be the index set for its n nearest neighbors in X_h

- $X_h^i = \{\mathbf{x}_j\}_{j \in \sigma_i^h}$ is the stencil for $\mathbf{x}_i$

- Find weights $\{c_{ij}\}_{j \in \sigma_i^h}$ such that for $u : \mathcal{M} \to \mathbb{R}$:

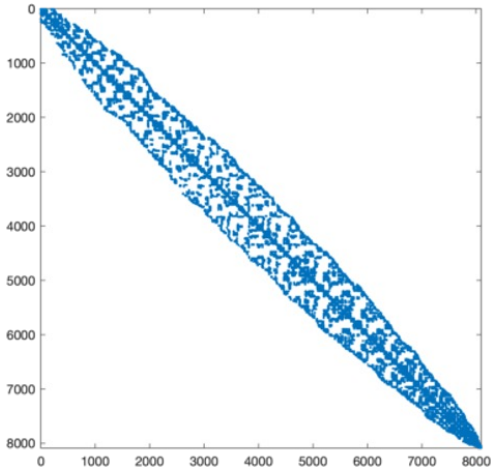
$$\Delta_{\mathcal{M}} u|_{\mathbf{x}_i} \approx \sum_{j \in \sigma_i^h} c_{ij} u(\mathbf{x}_j)$$

- Can assemble these weights into an N_h-by-N_h sparse differentiation matrix D_h to approximate $\mathcal{L}u = f$

  Let u^h, f^h contain samples of u and f at X_h

$$\mathcal{L}u = f \quad \longrightarrow \quad L_h u^h = f^h$$

$$L_h = \begin{cases} D_h & \text{if } \mathcal{L} = \Delta_{\mathcal{M}} \\ \mu I_h - D_h & \text{if } \mathcal{L} = \mu \mathcal{I} - \Delta_{\mathcal{M}} \end{cases}$$



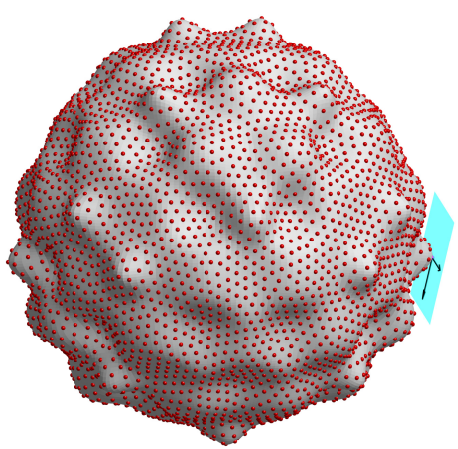
Tangent plane method for Laplace-Beltrami

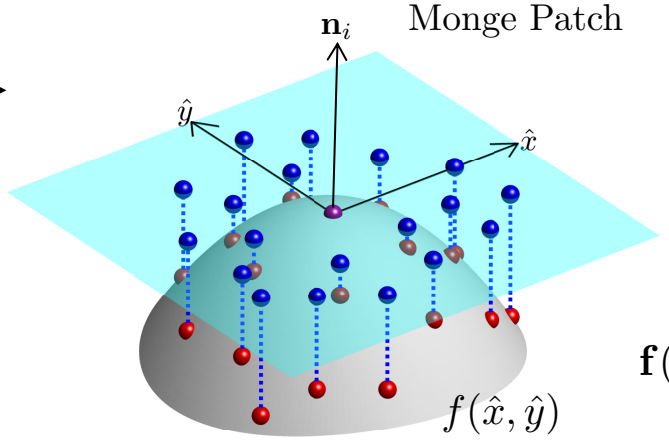
Tangent Plane method

Only requires point cloud $X_h \subset \mathcal{M}$ and normals $\{\mathbf{n}_j\}_{j=1}^{N_h}$

Demanet (2006), Shaw (2019), Suchde & Kuhnert (2020), Gunderman et. al. (2020), Jones et. al. (2023)

Idea: Consider local parameterization of the surface about each $\mathbf{x}_i$





Parameterization

$$\mathbf{f}(\hat{x},\hat{y}) = (\hat{x},\hat{y},f(\hat{x},\hat{y}))$$

Local metric tensor:

$$G = \begin{bmatrix} \partial_{\hat{x}}\mathbf{f}\cdot\partial_{\hat{x}}\mathbf{f} & \partial_{\hat{x}}\mathbf{f}\cdot\partial_{\hat{y}}\mathbf{f} \\ \partial_{\hat{y}}\mathbf{f}\cdot\partial_{\hat{x}}\mathbf{f} & \partial_{\hat{y}}\mathbf{f}\cdot\partial_{\hat{y}}\mathbf{f} \end{bmatrix} = \begin{bmatrix} 1+(\partial_{\hat{x}}f)^2 & (\partial_{\hat{x}}f)(\partial_{\hat{y}}f) \\ (\partial_{\hat{x}}f)(\partial_{\hat{y}}f) & 1+(\partial_{\hat{y}}f)^2 \end{bmatrix}, \qquad G^{-1} = \begin{bmatrix} g^{11} & g^{12} \\ g^{21} & g^{22} \end{bmatrix}, \qquad g = \det(G)$$

Laplace-Beltrami of u at $\mathbf{x}_i$

$$\Delta_{\mathcal{M}} u\big|_{\mathbf{x}_i} = \frac{1}{\sqrt{|g|}}\left(\partial_{\hat{x}}\left(\sqrt{|g|}g^{11}\partial_{\hat{x}}\right) + \partial_{\hat{x}}\left(\sqrt{|g|}g^{12}\partial_{\hat{y}}\right) + \partial_{\hat{y}}\left(\sqrt{|g|}g^{21}\partial_{\hat{x}}\right) + \partial_{\hat{y}}\left(\sqrt{|g|}g^{22}\partial_{\hat{y}}\right)\right)u\big|_{\hat{\mathbf{x}}_i}$$

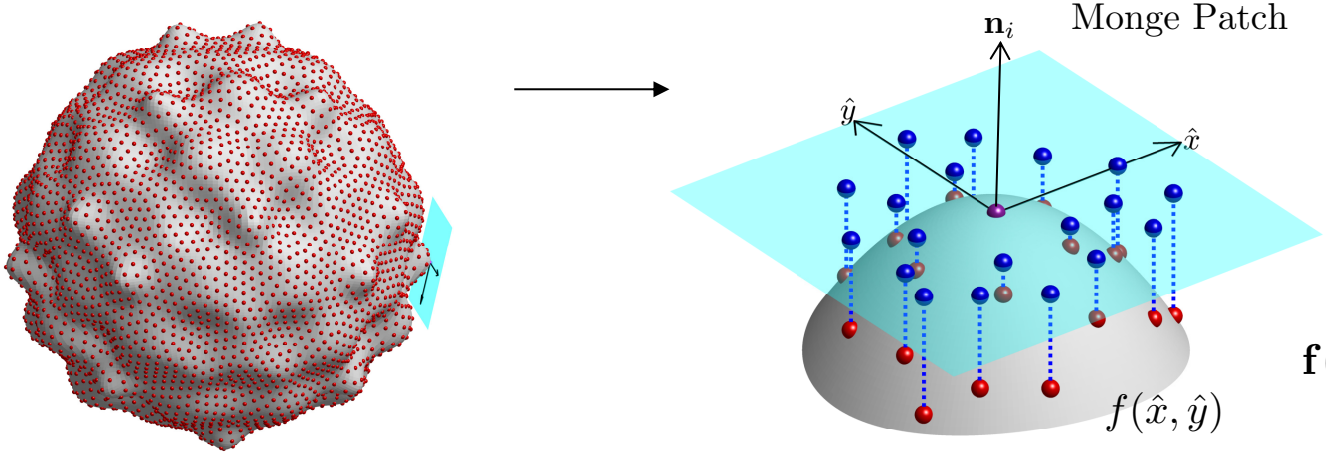
Tangent plane method for Laplace-Beltrami

Tangent Plane method

Only requires point cloud $X_h \subset \mathcal{M}$ and normals $\{\mathbf{n}_j\}_{j=1}^{N_h}$

Demanet (2006), Shaw (2019), Suchde & Kuhnert (2020), Gunderman et. al. (2020), Jones et. al. (2023)

Idea: Consider local parameterization of the surface about each $\mathbf{x}_i$



Parameterization

$$\mathbf{f}(\hat{x},\hat{y}) = (\hat{x},\hat{y},f(\hat{x},\hat{y}))$$

Local metric tensor:

$$G = \begin{bmatrix} \partial_{\hat{x}}\mathbf{f}\cdot\partial_{\hat{x}}\mathbf{f} & \partial_{\hat{x}}\mathbf{f}\cdot\partial_{\hat{y}}\mathbf{f} \\ \partial_{\hat{y}}\mathbf{f}\cdot\partial_{\hat{x}}\mathbf{f} & \partial_{\hat{y}}\mathbf{f}\cdot\partial_{\hat{y}}\mathbf{f} \end{bmatrix} = \begin{bmatrix} 1+(\partial_{\hat{x}}f)^2 & (\partial_{\hat{x}}f)(\partial_{\hat{y}}f) \\ (\partial_{\hat{x}}f)(\partial_{\hat{y}}f) & 1+(\partial_{\hat{y}}f)^2 \end{bmatrix}, \qquad G^{-1} = \begin{bmatrix} g^{11} & g^{12} \\ g^{21} & g^{22} \end{bmatrix}, \qquad g = \det(G)$$

Laplace-Beltrami of u at $\mathbf{x}_i$ simplifies

$$\Delta_{\mathcal{M}} u\big|_{\mathbf{x}_i} = \partial_{\hat{x}\hat{x}} u\big|_{\hat{\mathbf{x}}_i} + \partial_{\hat{y}\hat{y}} u\big|_{\hat{\mathbf{x}}_i} = \hat{\Delta} u\big|_{\hat{\mathbf{x}}_i}$$

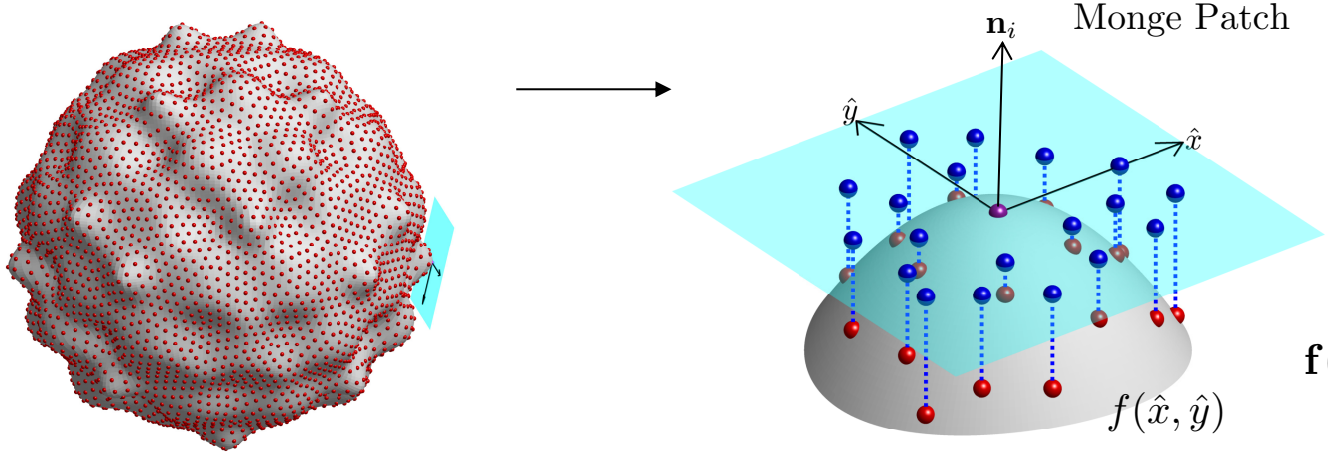
Tangent plane method for Laplace-Beltrami

Tangent Plane method

Only requires point cloud $X_h \subset \mathcal{M}$ and normals $\{\mathbf{n}_j\}_{j=1}^{N_h}$

Demanet (2006), Shaw (2019), Suchde & Kuhnert (2020), Gunderman et. al. (2020), Jones et. al. (2023)

Idea: Consider local parameterization of the surface about each $\mathbf{x}_i$



Parameterization

$$\mathbf{f}(\hat{x},\hat{y}) = (\hat{x},\hat{y},f(\hat{x},\hat{y}))$$

Consider PHS+Poly interpolant for the projected stencil X_h^i in the tangent plane

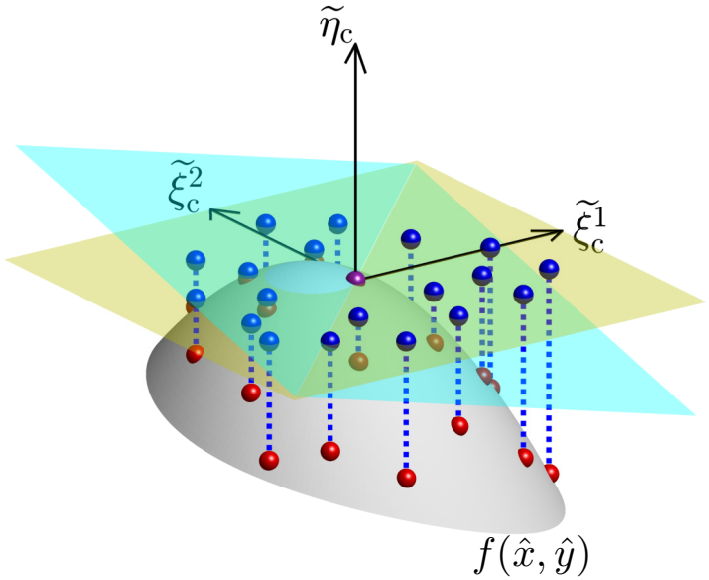
$$s(\hat{\mathbf{x}}) = \sum_{i=1}^{n} a_i \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2\ell+1} + \sum_{k=1}^{L(\ell)} b_k p_k(\hat{\mathbf{x}})$$

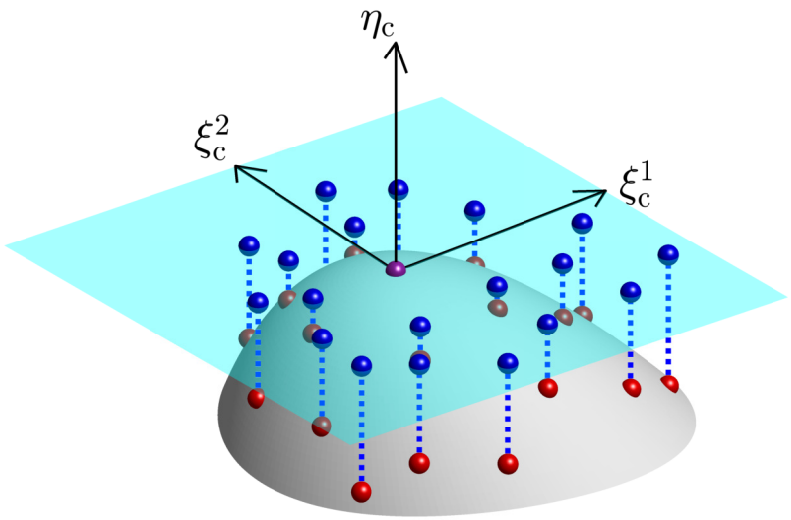
Use approximation $\Delta_{\mathcal{M}} u\big|_{\mathbf{x}_i} \approx \hat{\Delta} s\big|_{\hat{\mathbf{x}}_i}$ to find the stencil weights

Tangent plane method for Laplace-Beltrami

Approximating normals

Idea can also be used to approximate the normals $\{\mathbf{n}_j\}_{j=1}^{N_h}$ for $X_h \subset \mathcal{M}$





Start with a low-order approximation to the tangent plane about $\mathbf{x}_i$ using PCA (yellow plane)

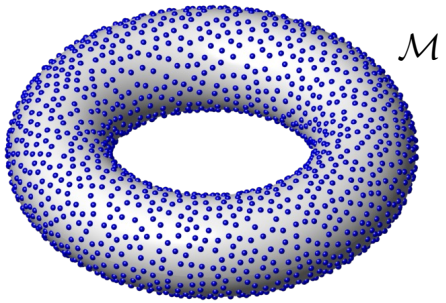
Fit local samples of f over the approximate tangent plane

$$s(\hat{\mathbf{x}}) = \sum_{i=1}^{n} a_i \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2\ell+1} + \sum_{k=1}^{L(\ell)} b_k p_k(\hat{\mathbf{x}})$$

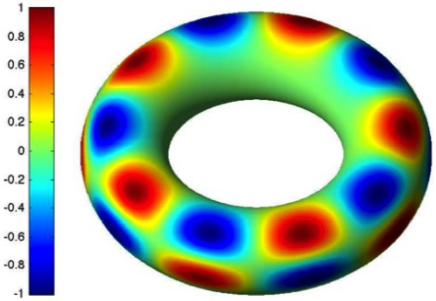
Use normal to this surface at $\mathbf{x}_i$ to improve the low-order approximation

Example: convergence of discrete surface Laplacian

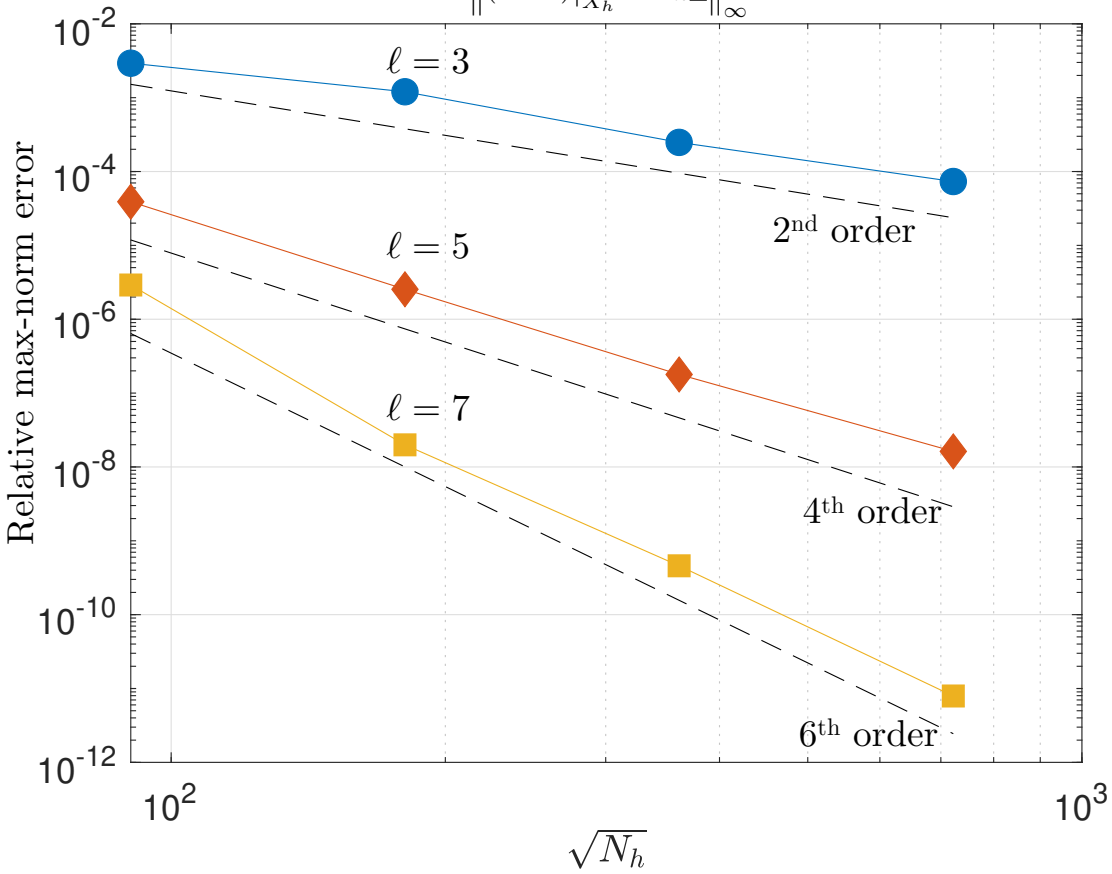
Problem: Approximate $\Delta_{\mathcal{M}} u$



Smooth target u



Error: $\left\| (\Delta_{\mathbb{M}} u)\big|_{X_h} - L_h \underline{u} \right\|_\infty$



Different poly. degrees ℓ in kernel approx.

Stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

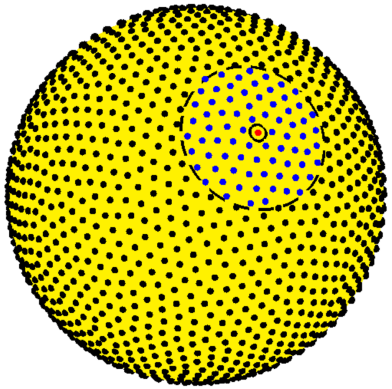
Point clouds with N_h unstructured nodes

$$h_{X_h} \sim 1/\sqrt{N_h}$$

Localized kernel methods for the Laplace-Beltrami

Let $X_h = \{\mathbf{x}_i\}_{j=1}^{N_h} \subset \mathcal{M}$, where $\mathbf{x}_i = (x_i, y_i, z_i)$

- For each $\mathbf{x}_i \in X_h$, let σ_i^h be the index set for its n nearest neighbors in X_h



- $X_h^i = \{\mathbf{x}_j\}_{j \in \sigma_i^h}$ is the stencil for $\mathbf{x}_i$
- Find weights $\{c_{ij}\}_{j \in \sigma_i^h}$ such that for $u : \mathcal{M} \to \mathbb{R}$:

$$\Delta_{\mathcal{M}} u|_{\mathbf{x}_i} \approx \sum_{j \in \sigma_i^h} c_{ij} u(\mathbf{x}_j)$$

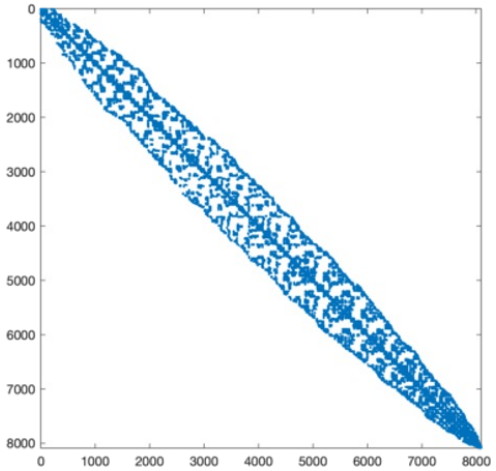
- Can assemble these weights into an N_h-by-N_h sparse differentiation matrix D_h to approximate $\mathcal{L}u = f$

Let u^h, f^h contain samples of u and f at X_h

$$\mathcal{L}u = f \quad \longrightarrow \quad L_h u^h = f^h$$

$$L_h = \begin{cases} D_h & \text{if } \mathcal{L} = \Delta_{\mathcal{M}} \\ \mu I_h - D_h & \text{if } \mathcal{L} = \mu \mathcal{I} - \Delta_{\mathcal{M}} \end{cases}$$

Large (sparse) non-symmetric linear systems

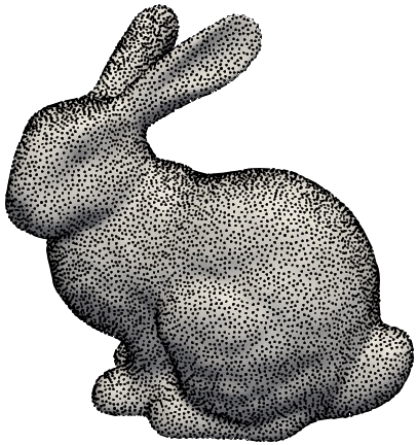


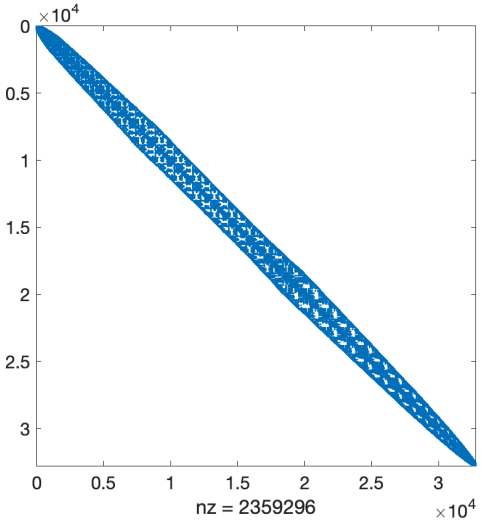
Geometric Multilevel Solver

Solving unstructured sparse linear systems

$$\mathcal{L}u = f \quad \longrightarrow \quad L_h u^h = f^h$$

$$L_h = \begin{cases} D_h & \text{if } \mathcal{L} = \Delta_{\mathcal{M}} \\ \mu I_h - D_h & \text{if } \mathcal{L} = \mu\mathcal{I} - \Delta_{\mathcal{M}} \end{cases}$$





Large (sparse) non-symmetric linear systems

Common general approaches:

- Direct sparse solver
- Preconditioned Krylov method
- Preconditioner:
  - Incomplete LU or Cholesky
  - Algebraic multigrid (AMG)

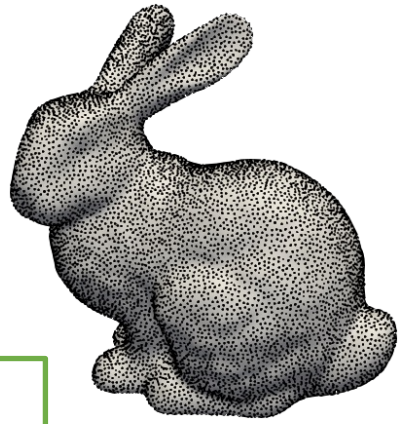
What about a general geometric multilevel solver/preconditioner?

W, Jones, Shankar (SISC 2023)
Meshfree geometric multilevel method for surface PDE on point clouds

Meshfree geometric multilevel (MGM) method: two-level cycle

Solve: $L_h u^h = f^h$

Level h nodes: X_h



1. Smooth initial guess:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

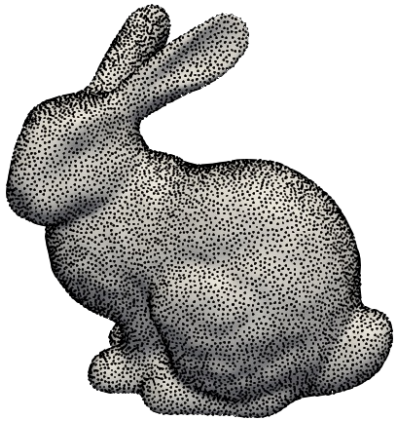
2. Compute residual:
   $r^h = f^h - L_h \hat{u}^h$

Defect correction:
- Solve for defect: $L_h e^h = r^h$
- Correct approximation: $\hat{u}^h \leftarrow \hat{u}^h + e^h$

MGM method: two-level cycle

Solve: $L_h u^h = f^h$

Level h nodes: X_h

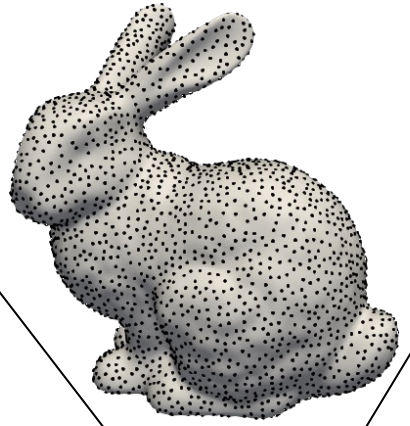


1. Smooth initial guess:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

2. Compute residual:
   $r^h = f^h - L_h \hat{u}^h$

3. Restrict residual:
   $r^H = I_h^H r^h$

Level H nodes: X_H



4. Solve for coarse level defect:
   $L_H e^H = r^H$

5. Interpolate defect:
   $e^h = I_H^h e^H$

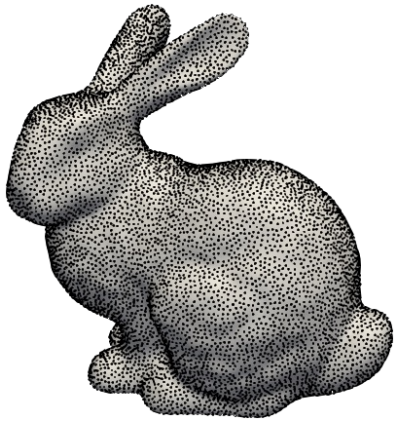
6. Correct approximation:
   $\hat{u}^h \leftarrow \hat{u}^h + e^h$

7. Smooth approximation:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

MGM method: two-level cycle

Solve: $L_h u^h = f^h$

Level h nodes: X_h

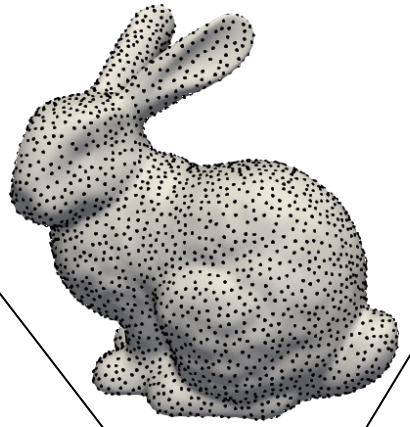


1. Smooth initial guess:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

2. Compute residual:
   $r^h = f^h - L_h \hat{u}^h$

3. Restrict residual:
   $r^H = I_h^H r^h$

Level H nodes: X_H



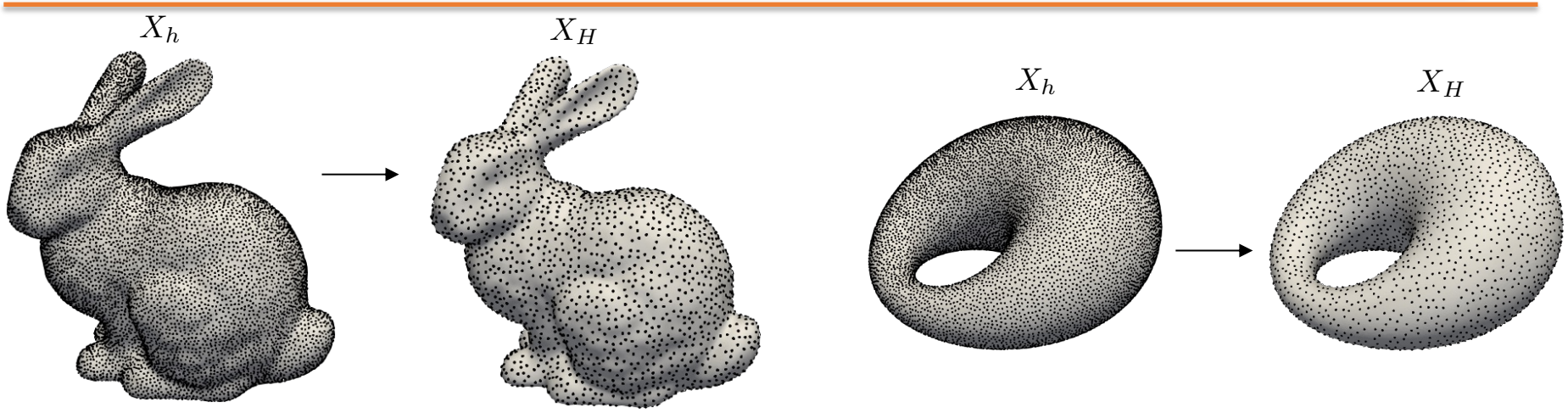
4. Solve for coarse level defect:
   $L_H e^H = r^H$

5. Interpolate defect:
   $e^h = I_H^h e^H$

6. Correct approximation:
   $\hat{u}^h \leftarrow \hat{u}^h + e^h$

7. Smooth approximation:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

Node coarsening



Idea: Given a point cloud X_h with N_h samples, determine a subset X_H with N_H samples that has maximal Poisson disk (or separation) radius.

Many algorithms developed in computer graphics to approximate the solution

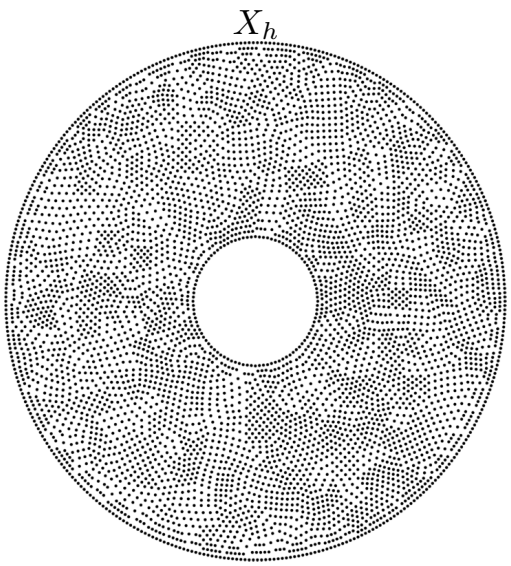
- Dart throwing – Cook (1986), Bridson (2007)
- Importance sampling - Ostromoukhov *et al.* (2004)
- Weighted Sample Elimination (WSE) – Yuksel (2015)

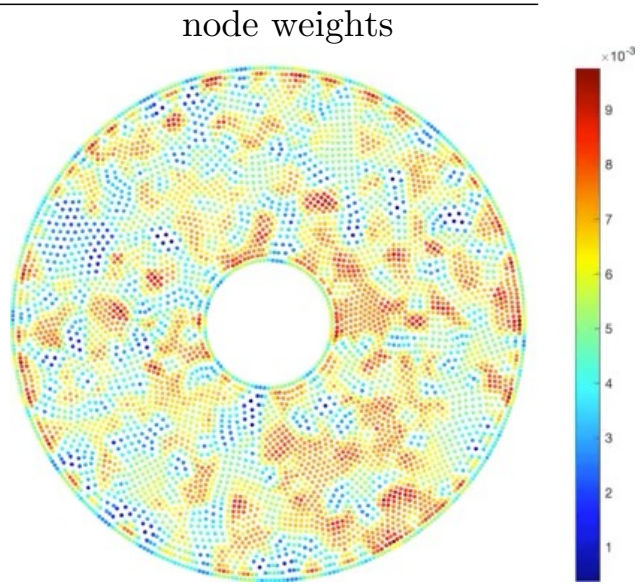
  Complexity: $\mathcal{O}(N_h - N_H) + \mathcal{O}(N_h \log N_h)$

We set $N_H = \lfloor N_h/4 \rfloor$, so that $H \approx 2h$

Node coarsening: Weighted Sample Elimination (WSE)

Input: X_h, N_h and N_H
Build a kd-tree for $\mathbf{X}_h$
for $i = N_h$ **do**
    Find all neighbors $\mathbf{x}_j$ contained in a ball of radius ρ centered at $\mathbf{x}_i$
    Assign weight w_{ij} for each $\mathbf{x}_j$ based on its distance to $\mathbf{x}_i$
    Compute weight $\overline{w}_i = \sum_j w_{ij}$
end for
Build a heap for X_h using weights $\overline{w}_i$
for $i = N_h - N_H$ **do**
    $\mathbf{x}_j \leftarrow$ pull sample from heap with highest priority
    for each sample $\mathbf{x}_i$ contained in a ball of radius ρ centered at $\mathbf{x}_j$ **do**
        Remove weight w_{ij} from $\overline{w}_i$: $\overline{w}_i \leftarrow \overline{w}_i - w_{ij}$
        Update the priority of $\mathbf{x}_i$ in the heap
    end for
end for





Meshfree interpolation/restriction operators

Interpolation operator: Stencil based approach

For each $\mathbf{x}_i \in X_h$, let σ_i^H be the index set for the m nearest neighbors in X_H to $\mathbf{x}_i$

Find weights $\{c_{ij}\}_{j \in \sigma_i^h}$ that interpolate $\{e_j^H\}_{j \in \sigma_i^H}$ to e_i^h:

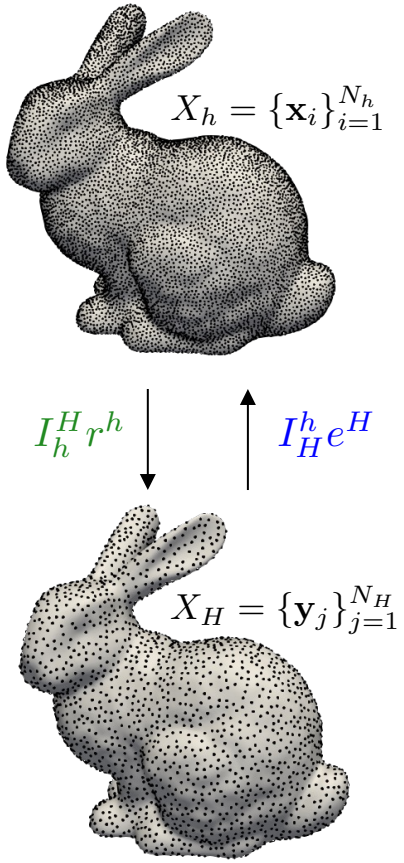
$$e_i^h = \sum_{j \in \sigma_H^i} c_{ij} e_j^H$$

Assemble these weights into an N_h-by-N_H sparse interpolation matrix I_H^h

We use a linear PHS interpolant to compute each set of weights $\{c_{i,j}\}_{j \in \sigma_H^i}$:

$$s_i(\mathbf{x}) = \sum_{j \in \sigma_H^i} a_{ij} \|\mathbf{x} - \mathbf{y}_j\| + b_i$$

Restriction operator: We use $I_h^H = (I_H^h)^T$



Coarse level operator

Two main ideas for constructing the coarse level operator

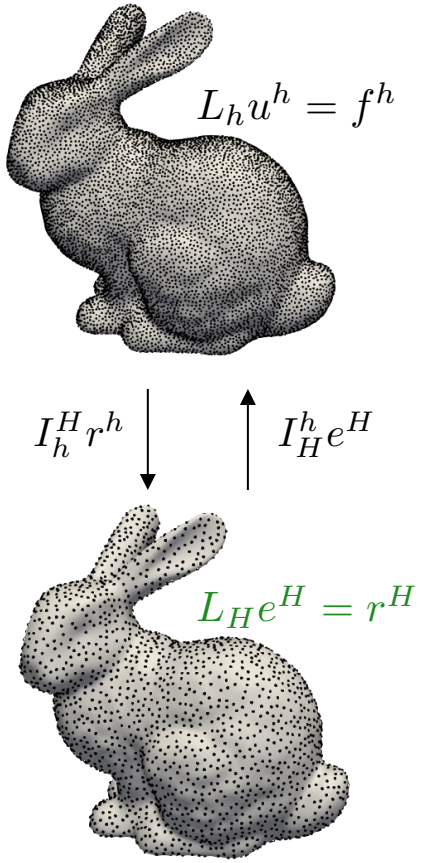
Direct discretization: Apply Kernel-FD method for $\Delta_{\mathcal{M}}$ on X_H

Galerkin projection: Use interpolation and restriction operators

$$L_H := I_h^H L_h I_H^h = (I_H^h)^T L_h I_H^h$$

We use this more general Galerkin approach

Leads to a variational principle for the coarse-level correction operator when L_h satisfies certain properties (Trottenberg et. al. 2000)



Other considerations

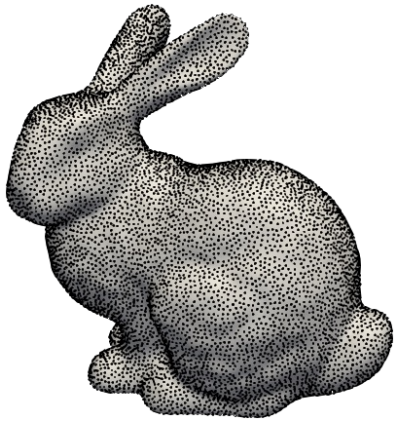
Smoother: Gauss-Seidel

Coarse level solver: Direct (sparse) solver

MGM method: two-level cycle

Solve: $L_h u^h = f^h$

Level h nodes: X_h

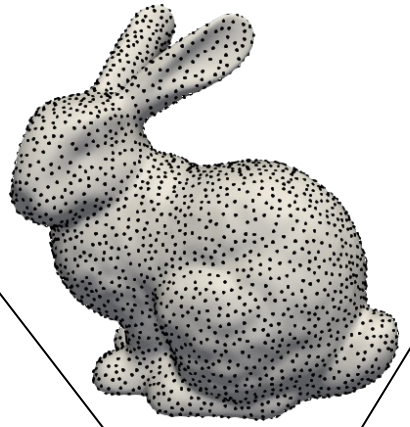


1. Smooth initial guess:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

2. Compute residual:
   $r^h = f^h - L_h \hat{u}^h$

3. Restrict residual:
   $r^H = I_h^H r^h$

Level H nodes: X_H



4. Solve for coarse level defect:
   $L_H e^H = r^H$

5. Interpolate defect:
   $e^h = I_H^h e^H$

6. Correct approximation:
   $\hat{u}^h \leftarrow \hat{u}^h + e^h$

7. Smooth approximation:
   $\hat{u}^h \leftarrow \text{smooth}(\hat{u}^h, f^h, L_h)$

Recursively apply two-level cycle

MGM: two-level cycle to multilevel

Algorithm MGM preprocessing phase

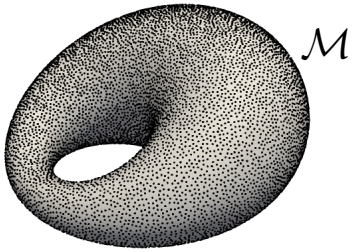
Input: Fine level nodes X_1 and operator L_1; min number of coarse level nodes $N_{\min}$
Compute number of levels: $p = \lfloor \log(N_1/N_{\min})/\log(4) \rfloor + 1$
for $j = 1 \dots p-1$ **do**
    Generate coarse point cloud X_{j+1} with $N_{j+1} = \lfloor N_1/4^j \rfloor$ points
    Generate interpolation operator I_{j+1}^j from X_{j+1} to X_j
    Set the restriction operator to $I_j^{j+1} = (I_{j+1}^j)^T$
    Generate Galerkin coarse level operator $L_{j+1} = I_{j+1}^j L_j I_j^{j+1}$
end for
Compute sparse LU decomposition of L_p

Algorithm MGM V(ν_1, ν_2)-cycle

Input: Right hand side f^1; Initial guess u^1; Number levels p; $\{L_j\}_{j=1}^{p-1}$; $\{I_{j+1}^j\}_{j=1}^{p-1}$;
    $\{I_j^{j+1}\}_{j=1}^{p-1}$; Sparse LU factorization of L_p;
Presmooth initial guess: $u^1 \leftarrow \text{presmooth}(L_1, u^1, f^1, \nu_1)$
Compute/restrict residual: $r^1 = I_1^2(f^1 - L_1 u^h)$
for $j = 2 \dots p-1$ **do**
    Presmooth defect: $e^j = \text{presmooth}(L_j, 0, r^j, \nu_1)$
    Compute/restrict residual: $r^{j+1} = I_j^{j+1}(r^j - L_j e^j)$
end for
Compute defect: Solve $L_p e^p = r^p$ using sparse LU decomposition of L_p
for $j = p-1, \dots, 2$ **do**
    Interpolate/correct defect: $e^j \leftarrow e^j + I_{j+1}^j e^{j+1}$
    Post smooth defect: $e^j \leftarrow \text{postsmooth}(L_j, e^j, r^j, \nu_2)$
end for
Interpolate defect/correct approximation: $u^1 \leftarrow u^1 + I_2^1 e^2$
Post smooth approximation: $u^1 \leftarrow \text{postsmooth}(L_1, u^1, f^1, \nu_2)$

Numerical results: convergence rates

Problem: $(\mathcal{I} - \Delta_{\mathcal{M}})u = f$



Different poly. degrees ℓ in kernel approx.

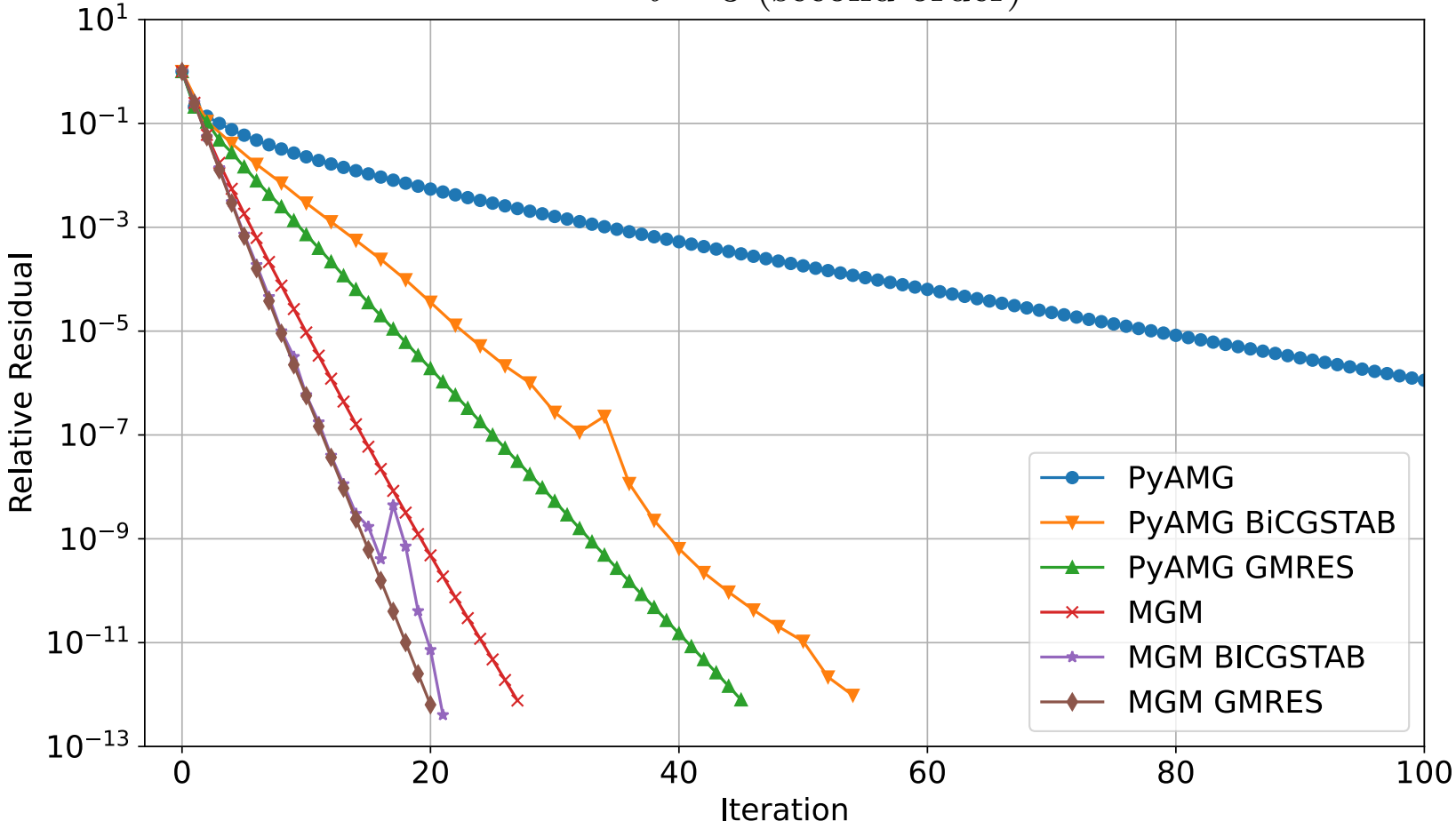
Stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

Point cloud with N_h=2,097,152 unstructured nodes

Use MGM as a solver or preconditioner

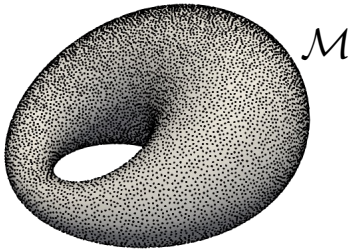
Compare with PyAMG (Olson & Schroder)

$\ell = 3$ (second order)



Numerical results: convergence rates

Problem: $(\mathcal{I} - \Delta_{\mathcal{M}})u = f$



Different poly. degrees ℓ in kernel approx.

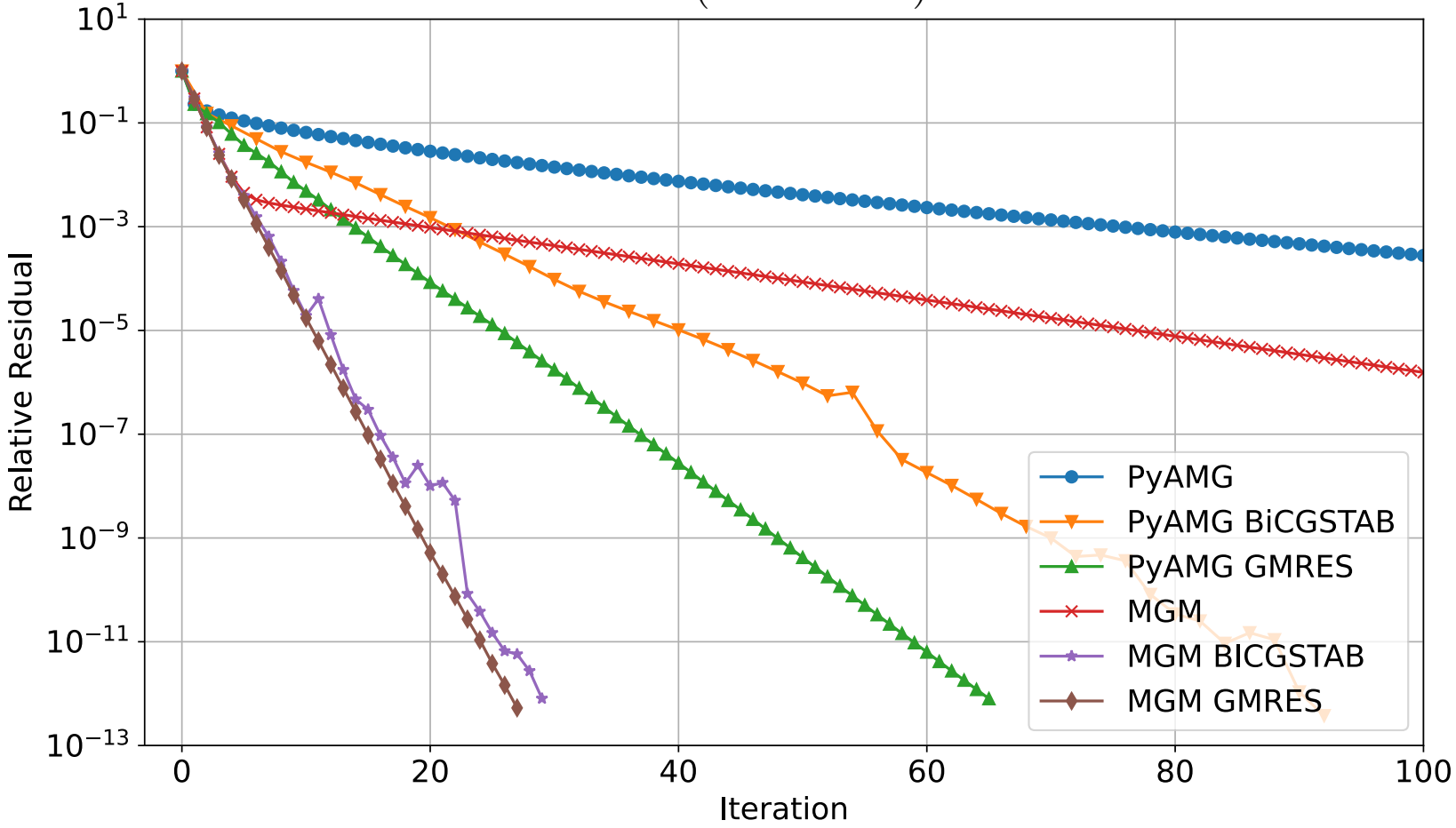
Stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

Point cloud with N_h=2,097,152 unstructured nodes

Use MGM as a solver or preconditioner

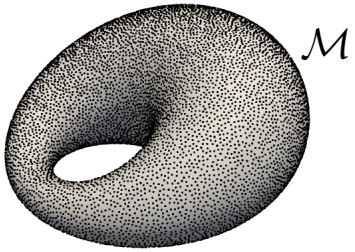
Compare with PyAMG (Olson & Schroder)

$\ell = 7$ (sixth order)



Numerical results: scaling

Problem: $(\mathcal{I} - \Delta_{\mathcal{M}})u = f$



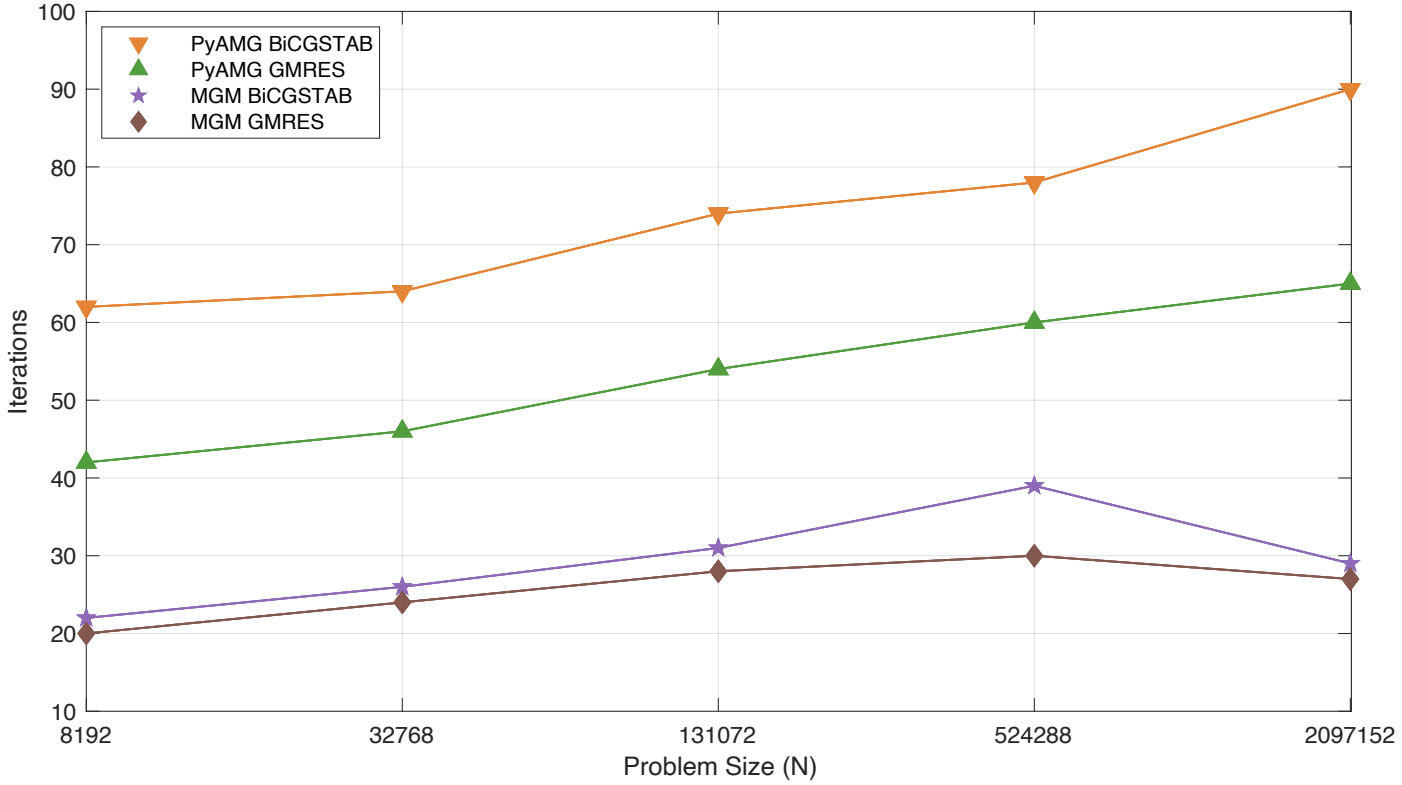
Different poly. degrees ℓ in kernel approx.

Stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

Use MGM as a solver or preconditioner

Compare with PyAMG (Olson & Schroder)

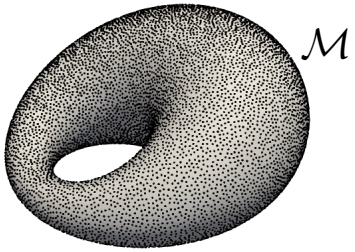
Scaling with N_h $(\ell = 7)$



Number of iterations to satisfy $\|f^h - L_h u^h\|_2 \leq 10^{-12} \|f^h\|_2$

Numerical results: timing

Problem: $(\mathcal{I} - \Delta_{\mathcal{M}})u = f$



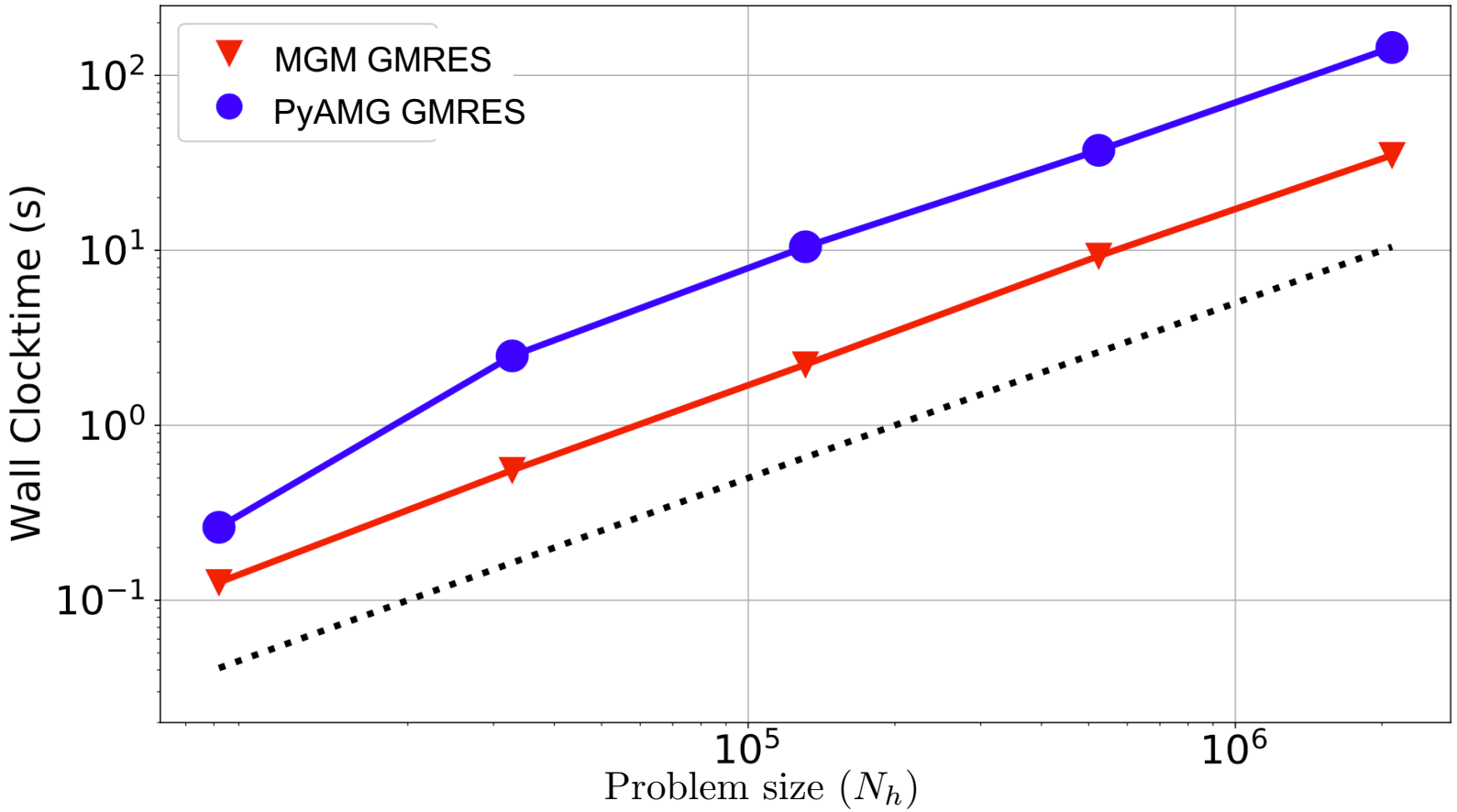
Different poly. degrees ℓ in kernel approx.

Stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

Use MGM as a solver or preconditioner

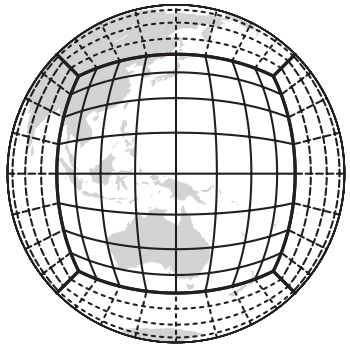
Compare with PyAMG (Olson & Schroder)

$\ell = 7$ (sixth order)



Also works well for mesh-based discretization!

Problem: $(\mathcal{I} - \Delta_{\mathcal{M}})u = f$



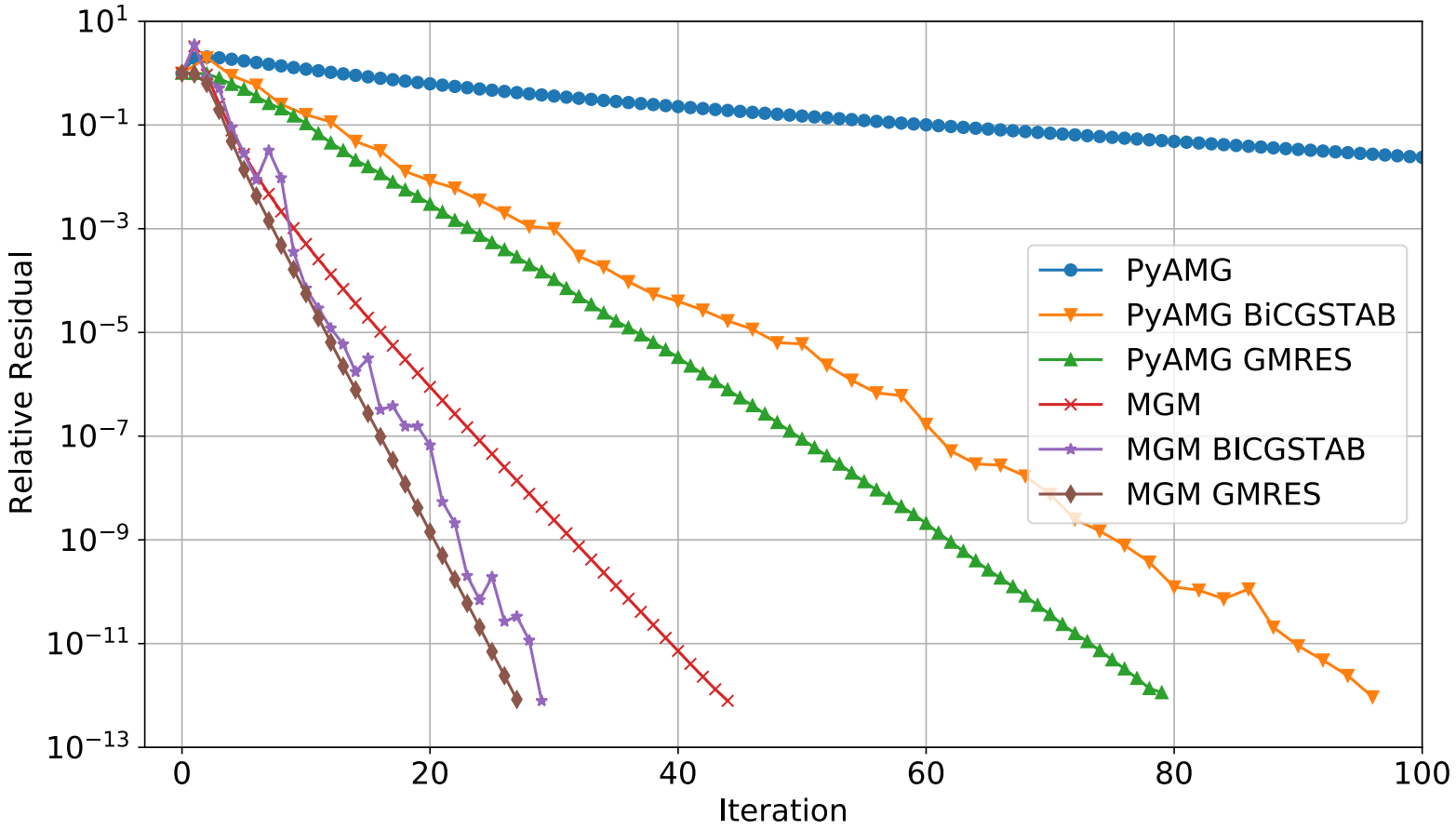
$\mathcal{M}$

Cubed sphere mesh with N_h=393216 (Grid: 64×64)

Discretization: Spectral elements with $p = 4$

Use MGM as a solver or preconditioner

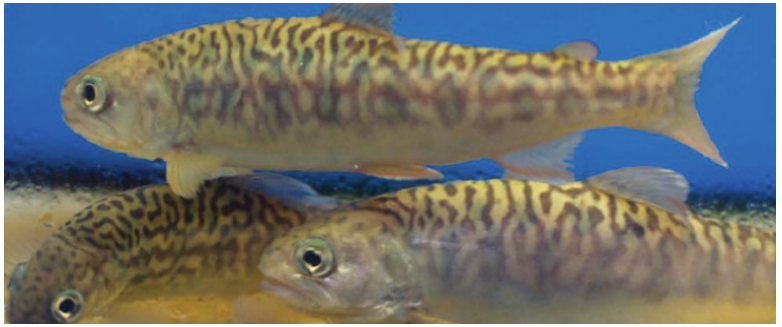
Compare with PyAMG (Olson & Schroder)



Application: Stationary Turing patterns

Pattern formation via non-linear reaction-diffusion systems; Turing (1952)

Possible mechanism for animal coat formation (and other morphogenesis phenomena)



Miyazawa, Okamoto, & Kondo (2010)

Example: Gierer–Meinhardt (1972)

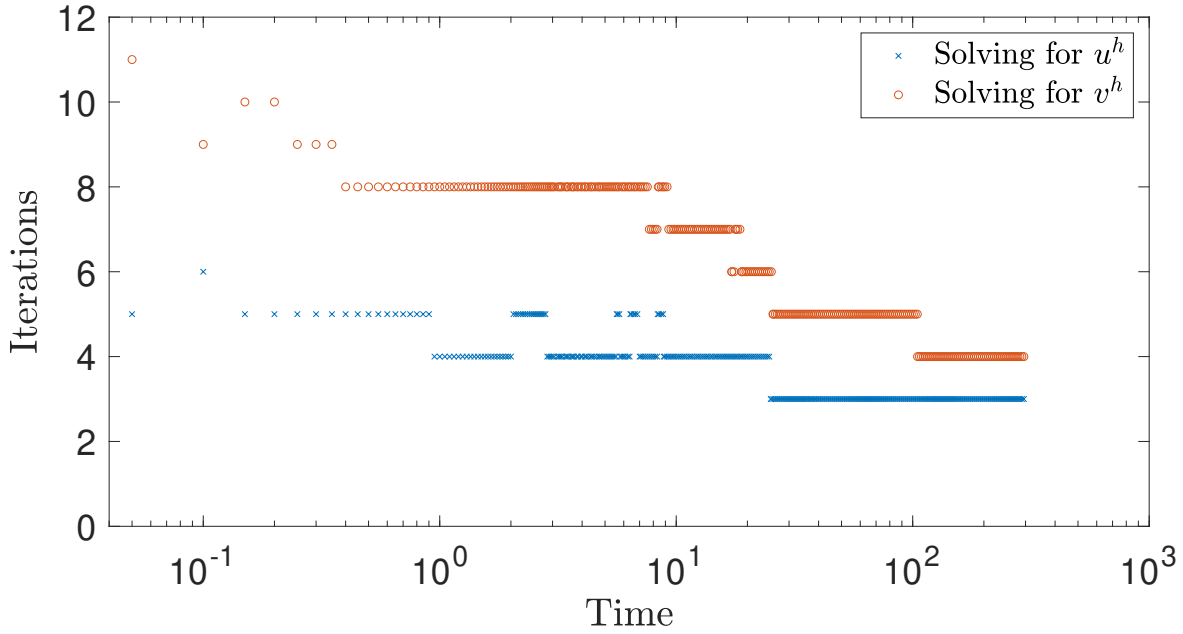
$$\frac{\partial u}{\partial t} = \delta_u \Delta_{\mathcal{M}} u + A - Bu + \left(\frac{u^2}{v(1 + Cu^2)} \right)$$

$$\frac{\partial v}{\partial t} = \delta_v \Delta_{\mathcal{M}} v + u^2 - v$$

Solution on bunny, $N_h = 291804$

Visualization of u at "steady-state"

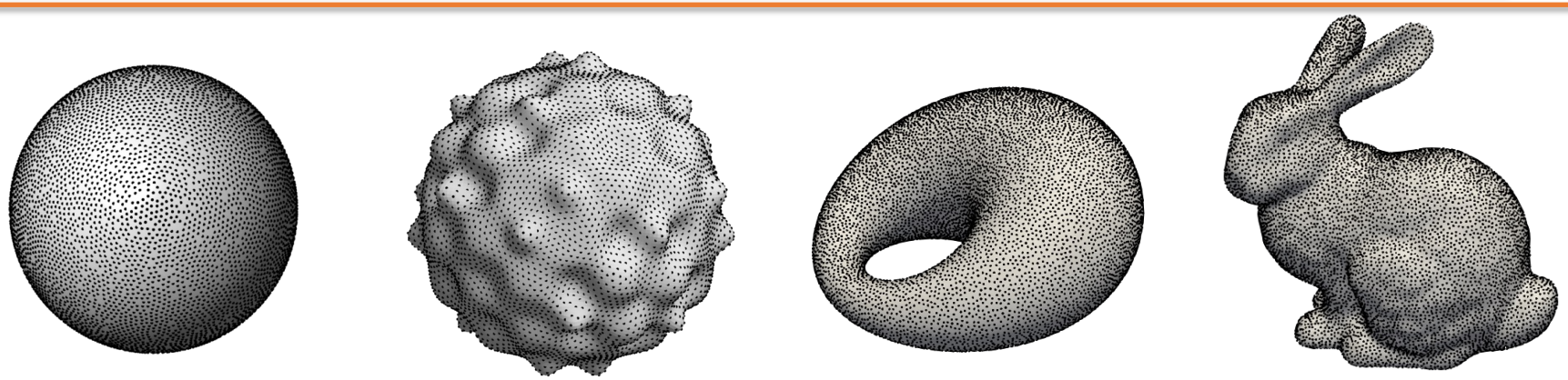
MGM GMRES Iterations, tol=10^{-8}



Quadrature

(cubature)

Quadrature (or cubature) for point clouds



$$I(f) := \int_{\mathcal{M}} f(\mathbf{x})\, dA \approx \sum_{j=1}^{N_h} w_j f(\mathbf{x}_j) =: I_{X_h}(f)$$

Nodes: $X_h = \{\mathbf{x}_j\} \subset \mathcal{M}$

Weights: $w^h = \{w_j\}$

Determine w^h so the error $E_{X_h}(f) = I(f) - I_{X_h}(f)$ is small for some space of functions and decreases rapidly to zero as $N_h \to \infty$

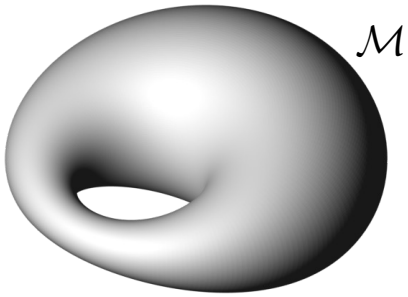
Dominant strategy for quadrature: Exactness

Pick a "good" vector space of functions $\mathcal{V}$ and enforce $E_{X_h}(f) = 0$ for all $f \in \mathcal{V}$

- What vector space of functions $\mathcal{V}$ should we use?
- How do we pick a bases for $\mathcal{V}$ and/or how do we integrate it over $\mathcal{M}$?

Introduce a new framework for quadrature that follows a different strategy

Motivation: Poisson problem on compact, connected manifold



Solve: $\Delta_{\mathcal{M}} u = g$

Issue: $\Delta_{\mathcal{M}}$ has a non-trivial null-space consisting of the constant function

Theorem (Fredholm alternative for surface Poisson problem)**.** *The solution of* $\Delta_{\mathcal{M}} u = g$*, with* $g \in L_2(\mathcal{M})$*, exists if and only if*

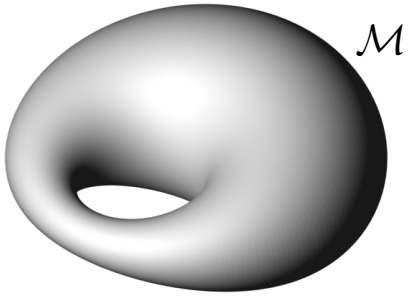
$$< g, w >= 0$$

for all w *in the nullspace of the adjoint of* $\Delta_{\mathcal{M}}$*.*

Poisson problem is self-adjoint and nullspace consists only of the constant function

$$\Longrightarrow \text{Solution exists if and only if } \int_{\mathcal{M}} g(\mathbf{x}) dA = 0$$

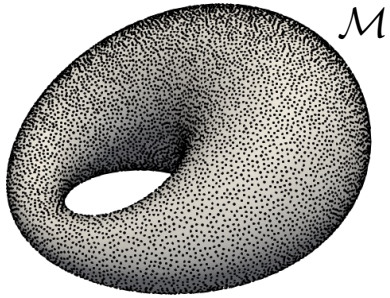
Motivation: Poisson problem on compact, connected manifold



New problem

Solve: $\Delta_{\mathcal{M}} u = g$ Constraint: $\dfrac{1}{|\mathcal{M}|}\displaystyle\int_{\mathcal{M}} u(\mathbf{x})\, dA = \mu$

Compatibility condition: $\int_{\mathcal{M}} g(\mathbf{x})\, dA = 0$



$X_h = \{\mathbf{x}_j\}_{j=1}^{N_h}$

Discretize

Solve: $L_h u^h = g^h$ Constraint: $\displaystyle\sum_{j=1}^{N_h} w_j u_j = (w^h)^T u^h = |\mathcal{M}|\mu$

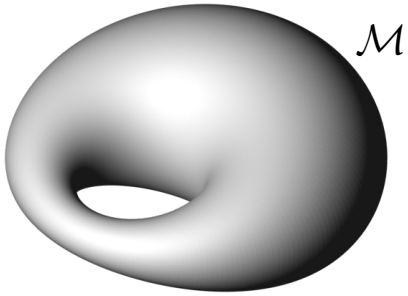
Compatibility condition: $(w^h)^T g^h = 0$

$$\Longrightarrow \begin{bmatrix} L_h & w^h \\ (w^h)^T & 0 \end{bmatrix} \begin{bmatrix} u^h \\ \lambda \end{bmatrix} = \begin{bmatrix} g^h \\ |\mathcal{M}|\mu \end{bmatrix}.$$

This problem has a unique solution provided $(w^h)^T e^h \neq 0$, where $e^h = [1\ 1\ \cdots\ 1]^T$

But, this solution satisfies $L_h u^h + \lambda w^h = g^h$, not the original problem $L_h u^h = g^h$ unless $\lambda = 0$

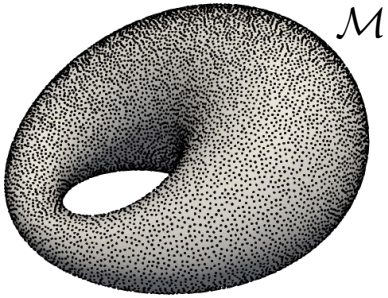
Motivation: Poisson problem on compact, connected manifold



New problem

Solve: $\Delta_{\mathcal{M}} u = g$ Constraint: $\dfrac{1}{|\mathcal{M}|}\displaystyle\int_{\mathcal{M}} u(\mathbf{x})\, dA = \mu$

Compatibility condition: $\int_{\mathcal{M}} g(\mathbf{x})\, dA = 0$



$X_h = \{\mathbf{x}_j\}_{j=1}^{N_h}$

Discretize

Solve: $L_h u^h = g^h$ Constraint: $\displaystyle\sum_{j=1}^{N_h} w_j u_j = (w^h)^T u^h = |\mathcal{M}|\mu$

Compatibility condition: $(w^h)^T g^h = 0$

$$\Longrightarrow \begin{bmatrix} L_h & w^h \\ (w^h)^T & 0 \end{bmatrix} \begin{bmatrix} u^h \\ \lambda \end{bmatrix} = \begin{bmatrix} g^h \\ |\mathcal{M}|\mu \end{bmatrix}.$$

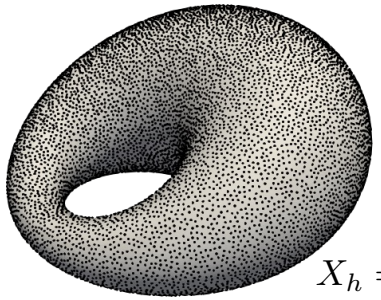
When does $\lambda = 0$?

$\lambda = 0$ if and only if $L_h^T w^h = 0$ and $(w^h)^T g^h = 0$

Fredholm alternative (1903) for linear systems

Main idea

Solve: $L_h u^h = g^h$



$X_h = \{\mathbf{x}_j\}_{j=1}^{N_h}$

Theorem (Fredholm alternative)**.**
A solution to $L_h u^h = g^h$ *exists if and only if* $(w^h)^T g^h = 0$ *for all* w^h *satisfying* $L_h^T w^h = 0$.

Suppose L_h only annihilates constant vectors so $\text{rank}(L_h) = N_h - 1$

The left null vector w^h of L_h will contain quadrature weights (after scaling), corresponding to the nodes X_h, for approximating the integral over $\mathcal{M}$

$$I(f) := \int_{\mathcal{M}} f(\mathbf{x})\, dA \approx \sum_{j=1}^{N_h} w_j f(\mathbf{x}_j) =: I_{X_h}(f)$$

We call $\{w^h, X_h\}$ a Fredholm alternative quadrature (FAQ) formula

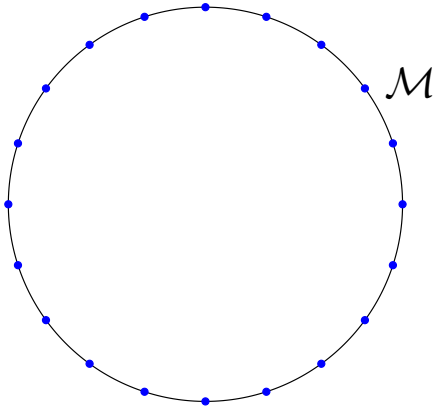
The quadrature formulas only depends on L_h and not on integrals of basis functions

Need to know or be able to approximate the surface area $|\mathcal{M}|$

What are FAQ formulas exact for?

Example: FAQ for the unit circle

Solve: $\Delta_{\mathcal{M}} u = g, \quad \mathcal{M} =$ unit disk



Discretize the domain using a grid of N equally spaced points $X_h = \{\mathbf{x}_j\}_{j=1}^N$, $\mathbf{x}_j = (\cos\theta_j, \sin\theta_j)$

$$u''(\theta) = g(\theta), \quad -\pi \le \theta \le \pi, \text{ periodic boundary conditions}$$

Use 2nd order finite differences

$$\underbrace{\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & & 1 \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & -2 & 1 \\ 1 & & & & & 1 & -2 \end{bmatrix}}_{L_h} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix}}_{u^h} = \underbrace{\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ \vdots \\ g_{N-1} \\ g_N \end{bmatrix}}_{g^h}$$

What is the FAQ formula in this case?

Trapezoidal rule: $w^h = \frac{2\pi}{N}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$

What is this exact for?

Examples: FAQ for a 2D surface

Experiment:

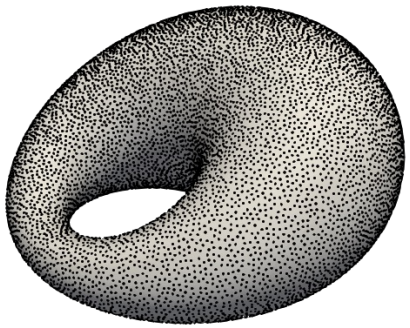
Scattered node set X_h with N_h points

Test convergence of the FAQ formulas as N increases

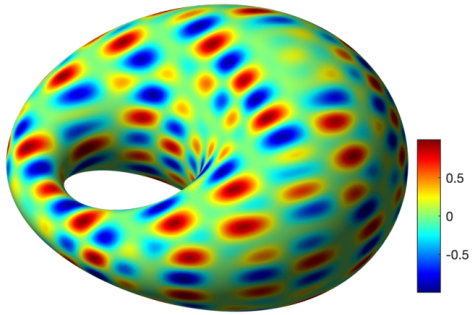
Different poly. degrees ℓ in kernel approx.

Stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

Example node set

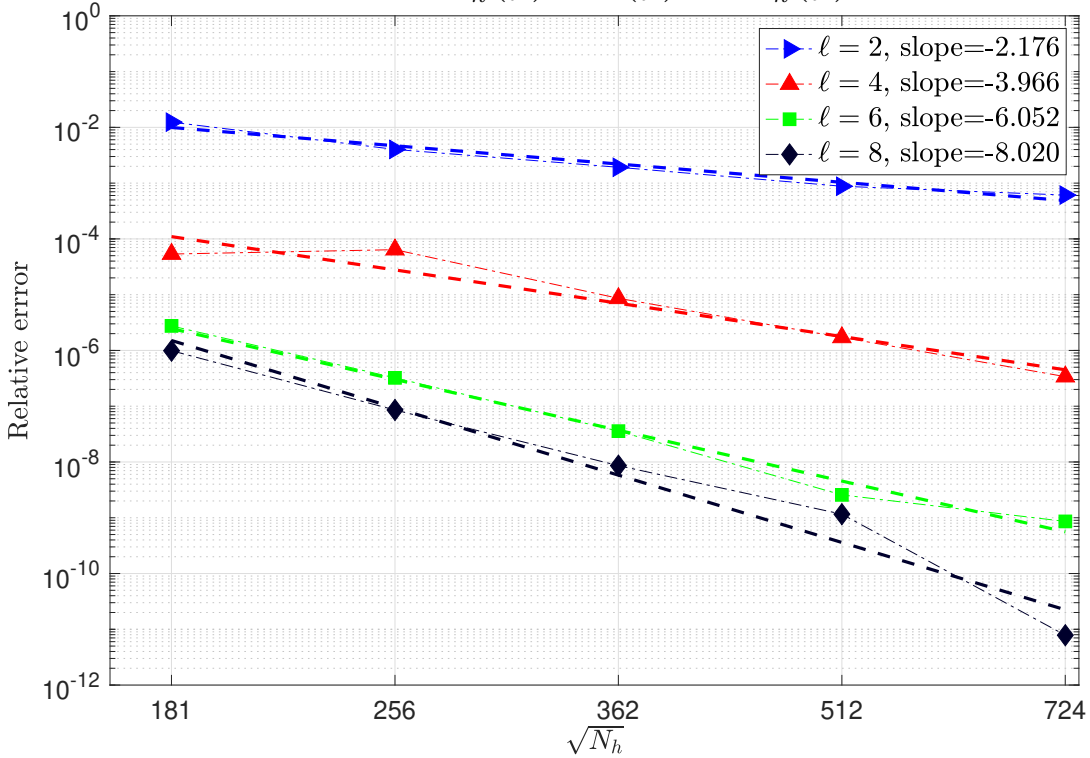


Target function f



$$\int_{\mathcal{M}} f(\mathbf{x})\, dA = 0.524555695704092\ldots$$

Error $E_{X_h}(f) = I(f) - I_{X_h}(f)$



Examples: FAQ for a 2D surface

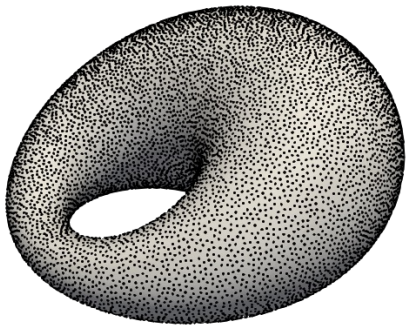
Experiment:

Scattered node set X_h with N_h points

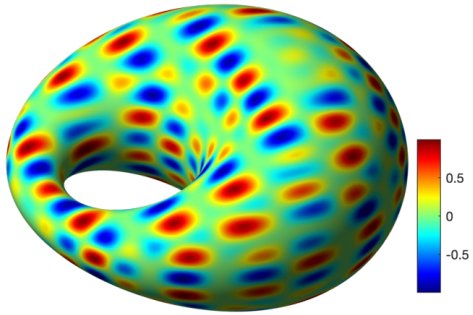
Test convergence of the FAQ formulas as N increases

Can obtain very high accuracy by using a global approximation of $\Delta_{\mathcal{M}}$ (Fuselier & W (2013))

Example node set

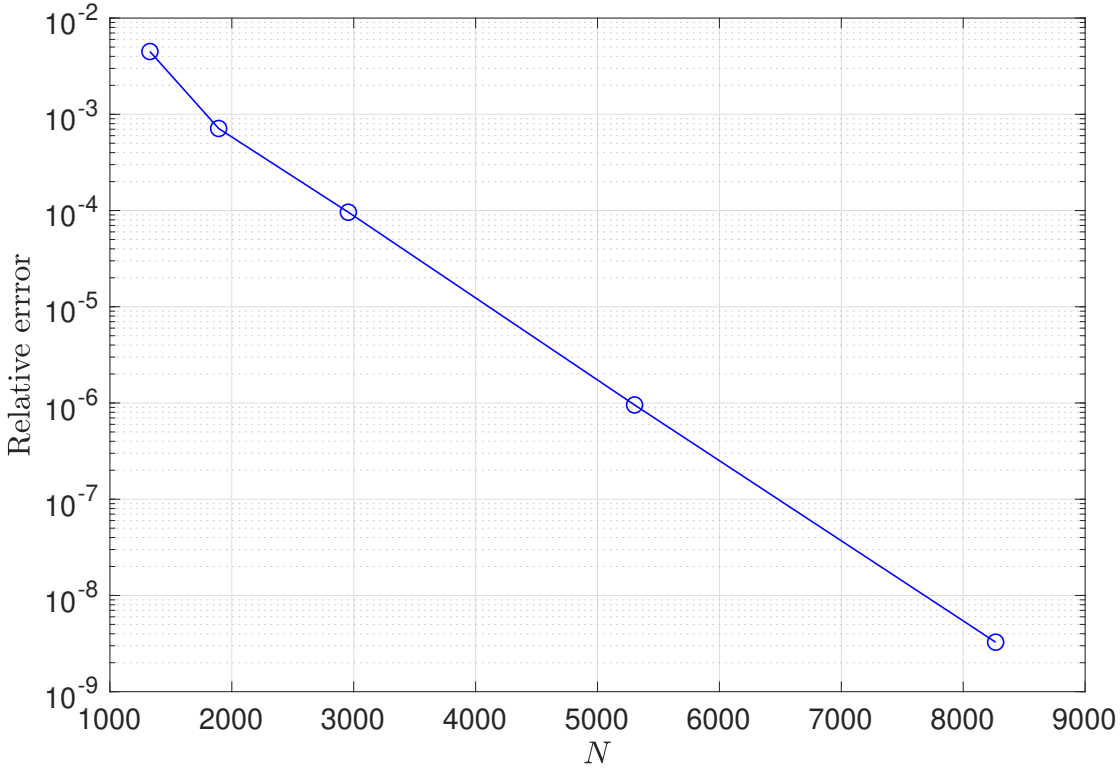


Target function f

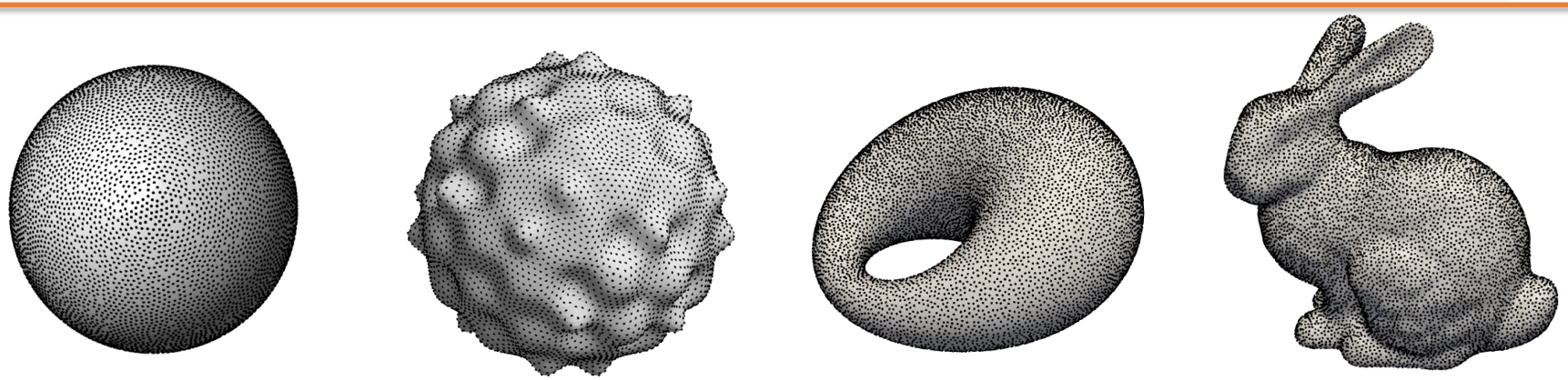


$$\int_{\mathcal{M}} f(\mathbf{x})\, dA = 0.524555695704092\ldots$$

Error $E_{X_h}(f) = I(f) - I_{X_h}(f)$



How to compute the FAQ weights



L_h is some N-by-N matrix representing a consistent discrete approximation to $\Delta_{\mathcal{M}}$ over some set of N nodes $X_h = \{\mathbf{x}_j\}_{j=1}^N \subset \mathcal{M}$.

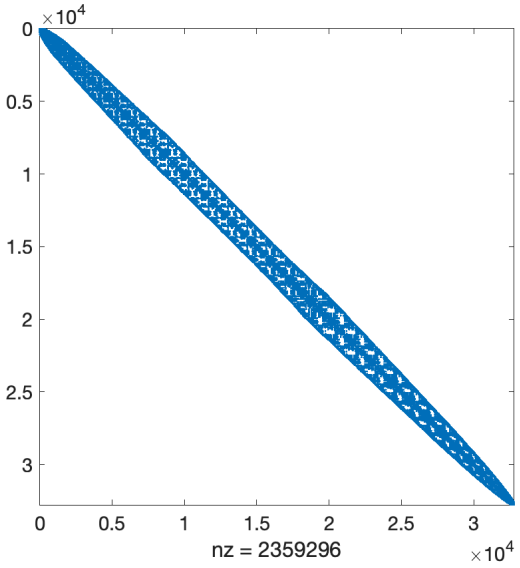
Determine FAQ weights by finding w^h that satisfies

$$L_h^T u^h = 0, \text{ subject to } \sum_{j=1}^{N} w_j = |\mathcal{M}|$$

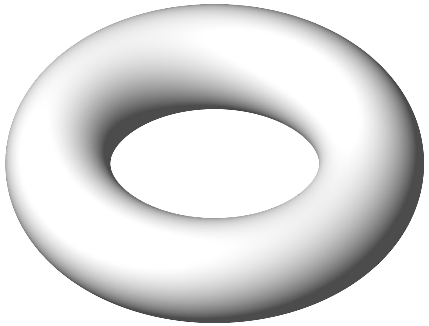
$$\implies \text{Solve } \begin{bmatrix} L_h^T & 1_h \\ 1_h^T & 0 \end{bmatrix} \begin{bmatrix} w^h \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ |\mathcal{M}| \end{bmatrix}$$

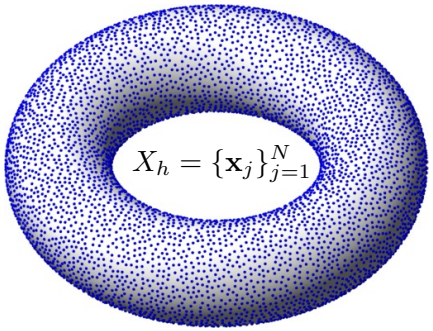
MGM can be used to solve this system efficiently

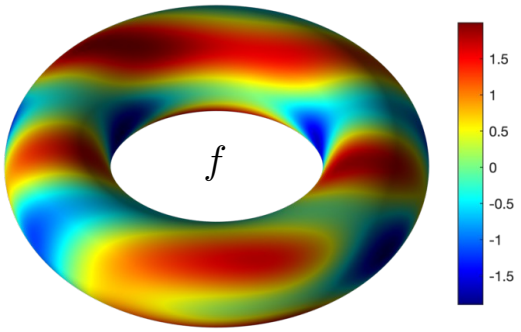
Pattern of non-zero entries of L_h



FAQ Summary







Continuous problem on Ω

$\Delta_{\mathcal{M}} u = g$

$\longrightarrow$

Discrete problem on X_h

$L_h u^h = g^h$

Solve: $L_h^T w^h = 0$, $\sum_j w_j = |\mathcal{M}|$

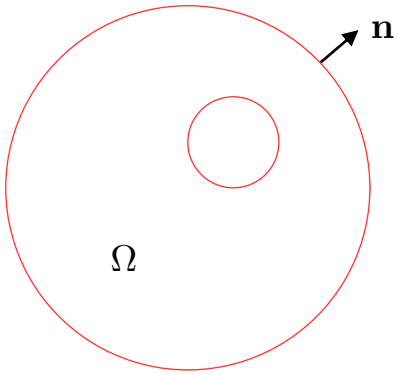
$\Longrightarrow$

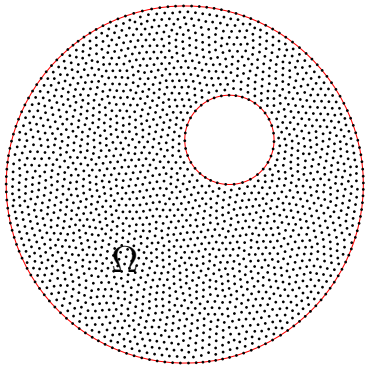
FAQ formula:

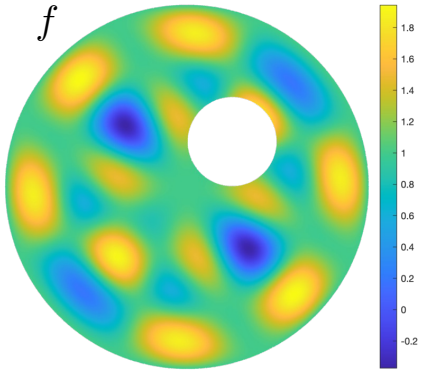
$\int_{\mathcal{M}} f(\mathbf{x}) dA \approx \sum_j w_j f(\mathbf{x}_j)$

Relative err: 6.17×10^{-7}

Can be extended to Euclidean domains via the Neumann-Poisson problem







Continuous problem on Ω

$\Delta u = g$ in Ω

$\mathbf{n} \cdot \nabla u = 0$ on $\partial\Omega$

$\longrightarrow$

Discrete problem on $X_h = \{\mathbf{x}_j\}_{j=1}^N$

$L_h u^h = g^h$

Solve: $L_h^T w^h = 0$, $\sum_j w_j = |\Omega|$

$\Longrightarrow$

Quadrature formula:

$\int_{\Omega} f(\mathbf{x}) d\mathbf{x} \approx \sum_j w_j f(\mathbf{x}_j)$

Relative err: 2.277×10^{-6}

Implicit Surface Reconstruction

Motivation

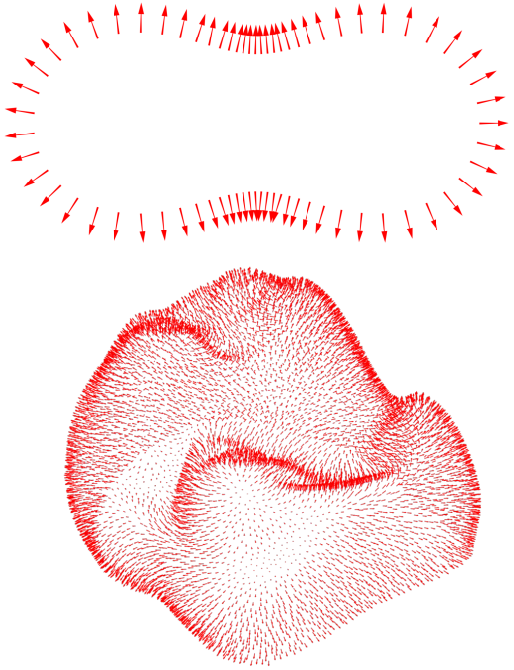
Reconstructing a surface from an (oriented) point cloud has many applications

computer graphics, computer-aided design, medical imaging, manufacturing, remote sensing

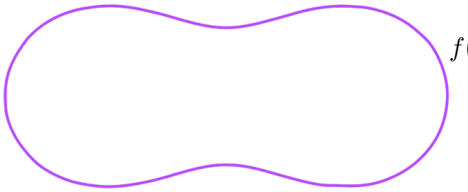
Common approach: implicit surface (zero-level set) methods

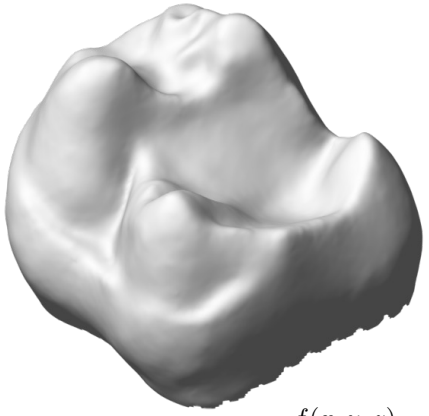
Problem: Given an oriented point cloud $\{(\mathbf{x}_j, \mathbf{n}_j)\}_{j=1}^N$, find a function f such that the level-set $S = \{\mathbf{x} \in \mathbb{R}^d \big| f(\mathbf{x}) = 0\}$ fits the point cloud. S is called the zero-level implicit surface.

Oriented point cloud



Implicit surface





Curl-free Partition of Unity (CFPU) method (Drake, Fuselier, & W, 2022)

Overview of CFPU method

Uses the following fundamental result from vector calculus

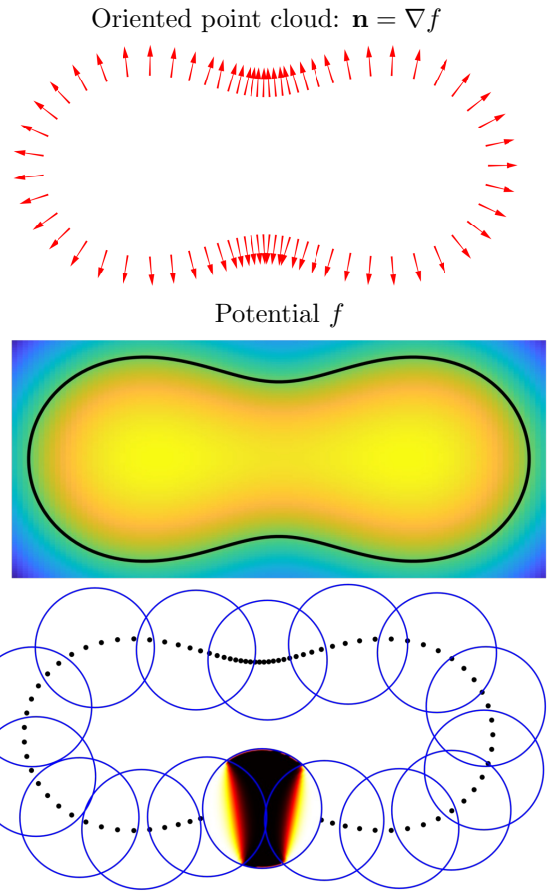
Proposition 1 *A vector field* $\mathbf{n}$ *is curl-free in* $\mathbb{R}^d$ *if and only if locally there exists a scalar potential* $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ *such that* $\mathbf{n} = \nabla f$*. Furthermore,* f *is unique up to a constant.*

Overview of our approach:

- Fit a curl-free Kernel interpolant to the normal vectors

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \Phi(\mathbf{x}, \mathbf{x}_j)\mathbf{c}_j = \sum_{j=1}^{N} \left[\nabla\nabla^T \phi\left(\|\mathbf{x} - \mathbf{x}_j\|\right)\right] \mathbf{c}_j$$

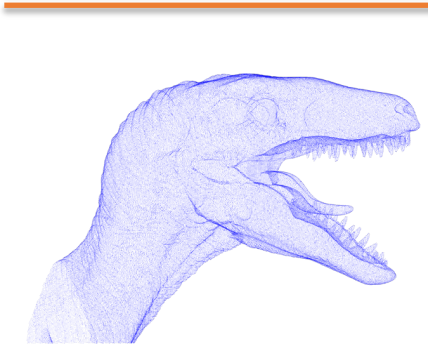
- Extract a scalar potential $\tilde{f}$ for the interpolant
- Shift the potential to be zero at the point cloud
- Include regularization if the data is noisy
- Combine the above in a partition of unity (PU) framework
- Use isosurface extractor to obtain the zero-level set



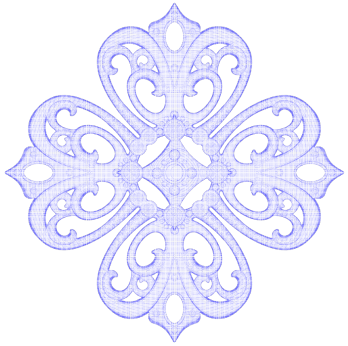
Relationship to previous work

Narrow band: Carr et. al. (1997, 2001), Turk & O'Brien (1999), Morse et. al. (2001), Fasshauer (2007)
Indicator function: Kazhdan et. al. (2006, 2013, 2019), Calakli & Taubin (2011),
Hermite: Walder et. al. (2006), Süßmuth et. al. (2010), Macêdo et. al. (2011), Liu et. al. (2016),

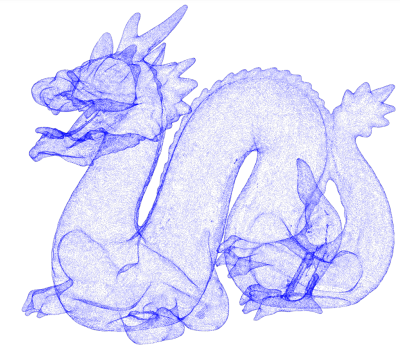
Computational performance



Raptor



Filigree



Dragon

Buddha

Armadillo

Surface	N	M	Grid size	Time in seconds			
				Number of cores			
				1	2	4	8
Raptor	135740	10337	384×143×258	18.2 (16.5)	6.67 (8.84)	4.14 (7.22)	3.35 (5.69)
Filigree	514300	35130	383×63×384	30.4 (7.70)	10.9 (4.10)	6.63 (3.39)	4.99 (3.13)
Dragon	434856	14400	384×175×272	42.73 (11.83)	17.24 (6.46)	10.7 (5.46)	9.20 (4.82)
Buddha	583079	42861	161×162×384	37.34 (7.71)	14.0 (4.20)	8.32 (3.43)	5.99 (3.02)
Armadillo	172974	14349	323×294×384	9.42 (10.0)	3.62 (5.54)	2.25 (4.85)	1.61 (4.55)

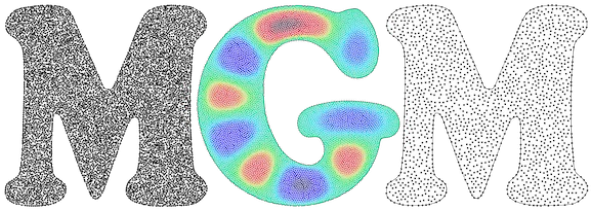
fit time & eval. time

Concluding remarks

Surface PDEs

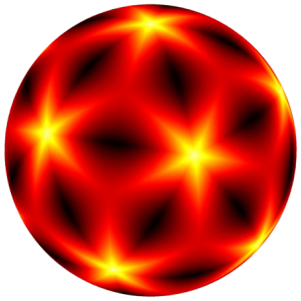
- Incorporation of advection using semi-Lagrangian method
- Anisotropic diffusion
- PDEs on moving surfaces
- KernelDMSuite (https://github.com/gradywright/kerneldmsuite)

Meshfree geometric multilevel (MGM) method



- Generalize to other discretization methods, both meshfree and mesh-restrained
- Euclidean domains
- Going beyond Poisson equation
- Parallelization
- MGM (https://github.com/gradywright/mgm)

Fredholm Alternative Quadrature (FAQ)

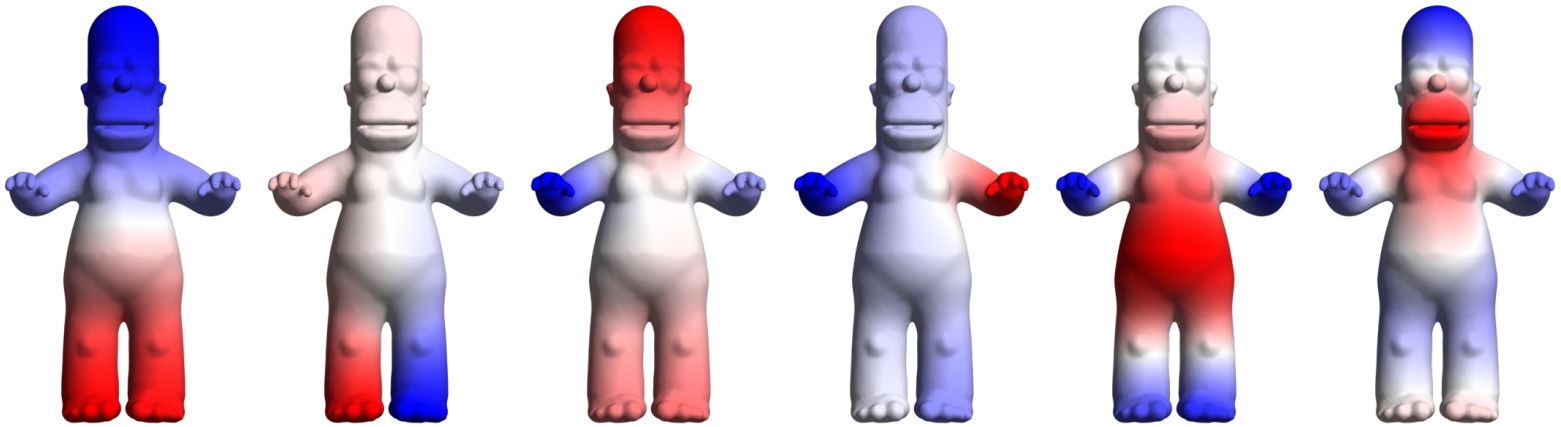


- Positive weight quadrature
- Error estimates
- Domain decomposition
- Non-smooth integrands

Implicit surface reconstruction

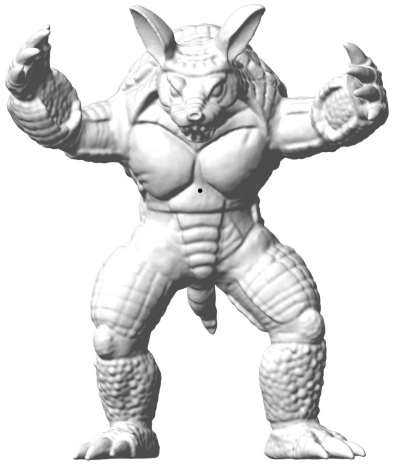
- Curl-free Partition-of-Unity Method (https://github.com/gradywright/cfpu)

Questions?



Application: Geodesic distance

Problem: Compute the geodesic distance from a single point on a surface $\mathcal{M}$



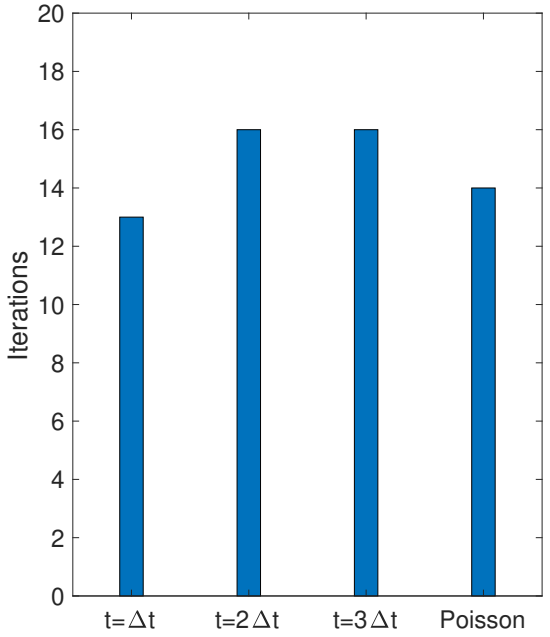
Heat flow method: Crane, Weischedel, Wardetzky (2013)

1. Solve $u_t = \Delta_{\mathcal{M}} u$, with $u_0 = \delta(\mathbf{x}^*)$, for $0 < t \leq m\Delta t$
2. Compute the vector field $\boldsymbol{\eta} = -\nabla_{\mathcal{M}} u / |\nabla_{\mathcal{M}} u|$
3. Solve the Poisson problem $\Delta_{\mathcal{M}} \phi = \nabla_{\mathcal{M}} \cdot \boldsymbol{\eta}$

$\Longrightarrow$ Potential ϕ approximates the geodesic distance from $\mathbf{x}^*$ on $\mathcal{M}$

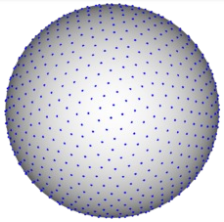
Solution on armadillo, $N_h = 872773$

MGM GMRES Iterations, tol=10^{-8}



Numerical results: accuracy vs. iteration

Problem: $\Delta_{\mathcal{M}} u = f$

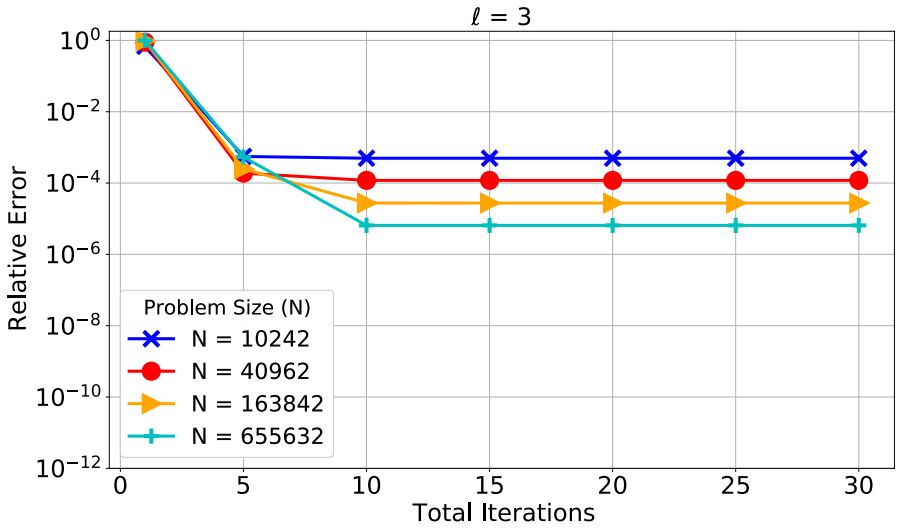


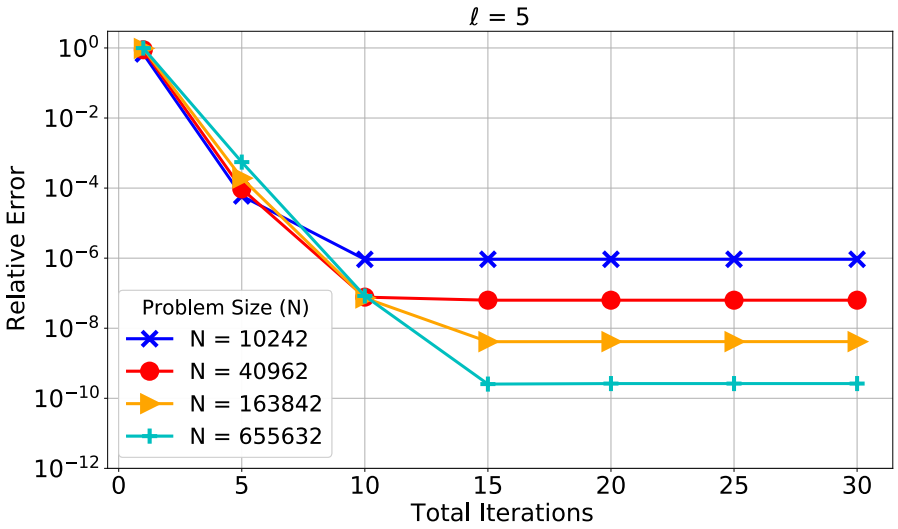
$\mathcal{M}$

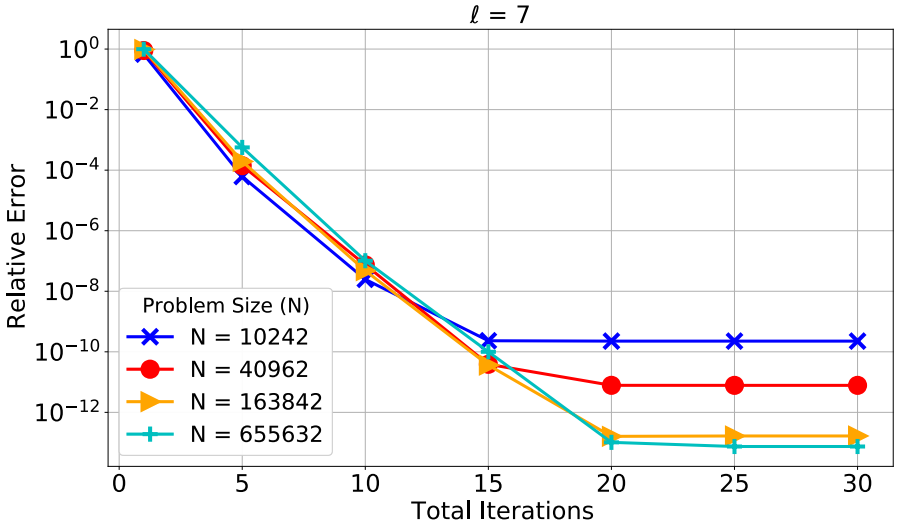
Test different poly. degrees ℓ in RBF-FD

RBF-FD stencil size $n = \lceil (\ell+1)(\ell+2) \rceil$

Exact solution is Y_5^4 spherical harmonic

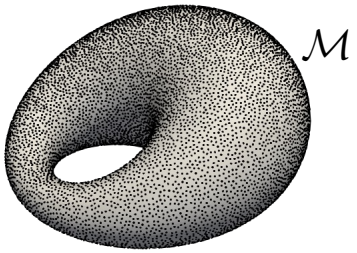




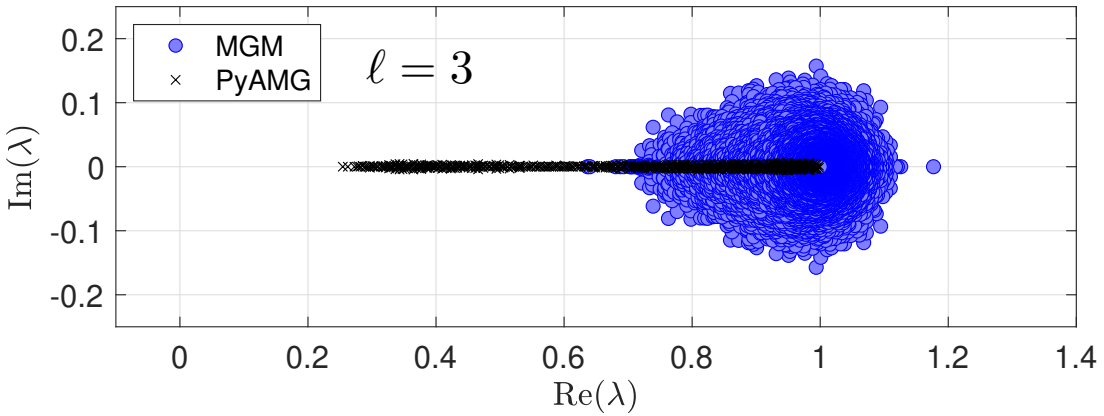


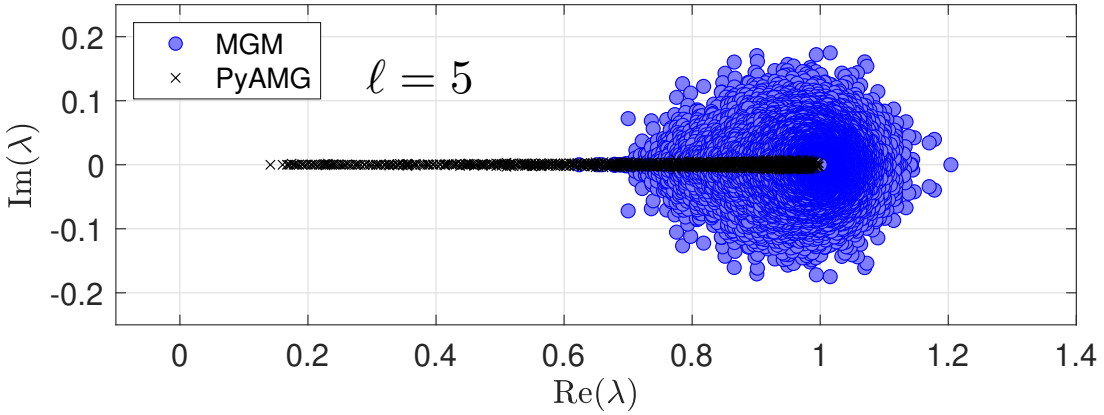
Numerical results: spectrum analysis

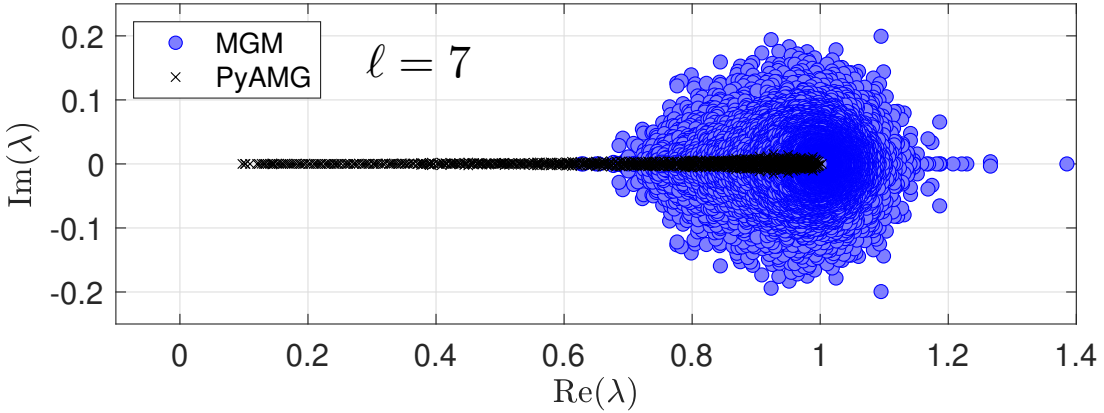
Problem: $(\mathcal{I} - \Delta_{\mathcal{M}})u = f$



Eigenvalues of the preconditioned matrix $L_h M_h$







Modifications required for surface Poisson problem

Suppose $\mathcal{L} = \Delta_{\mathcal{M}}$, so we want to solve

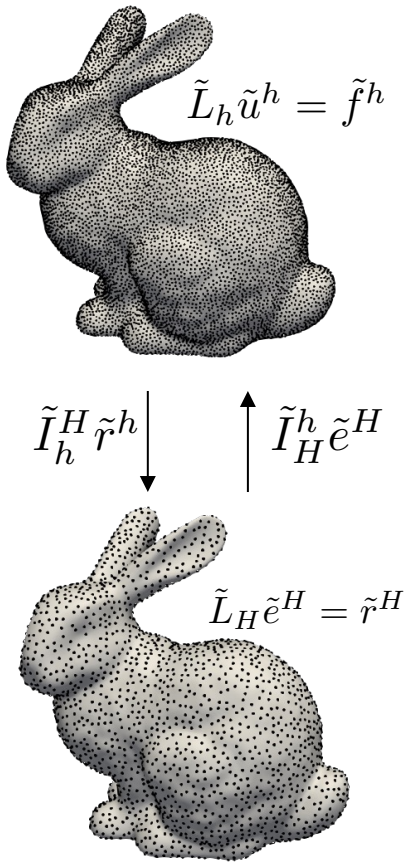
$$\Delta_{\mathcal{M}} u = f$$

Provided f satisfies the compatibility condition $\int_{\mathcal{M}} f dA = 0$, the solution is unique up to a constant

$L_h u^h = f^h$ is singular since L_h annihilates constants

Need to add another condition:

$$\frac{1}{N_h} \sum_{j=1}^{N_h} u_j = 0 \quad \longrightarrow \quad \underbrace{\begin{bmatrix} L_h & b_h^T \\ b_h & 0 \end{bmatrix}}_{\tilde{L}_h} \underbrace{\begin{bmatrix} u^h \\ \lambda^h \end{bmatrix}}_{\tilde{u}^h} = \underbrace{\begin{bmatrix} f^h \\ 0 \end{bmatrix}}_{\tilde{f}^h}$$



Requires modifications to the transfer and coarse level operators (Adams 2004)

$$\tilde{L}_H = \underbrace{\begin{bmatrix} I_h^H & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{I}_h^H} \underbrace{\begin{bmatrix} L_h & b_h^T \\ b_h & 0 \end{bmatrix}}_{\tilde{L}_h} \underbrace{\begin{bmatrix} I_H^h & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{I}_H^h} = \begin{bmatrix} I_h^H L_h I_H^h & I_h^H b_h^T \\ b_h I_H^h & 0 \end{bmatrix}$$