A radial basis function partition of unity method for divergence-free approximation of vector fields on the sphere*

Grady B. Wright
Boise State University

Collaborator: Edward J. Fuselier, High Point University

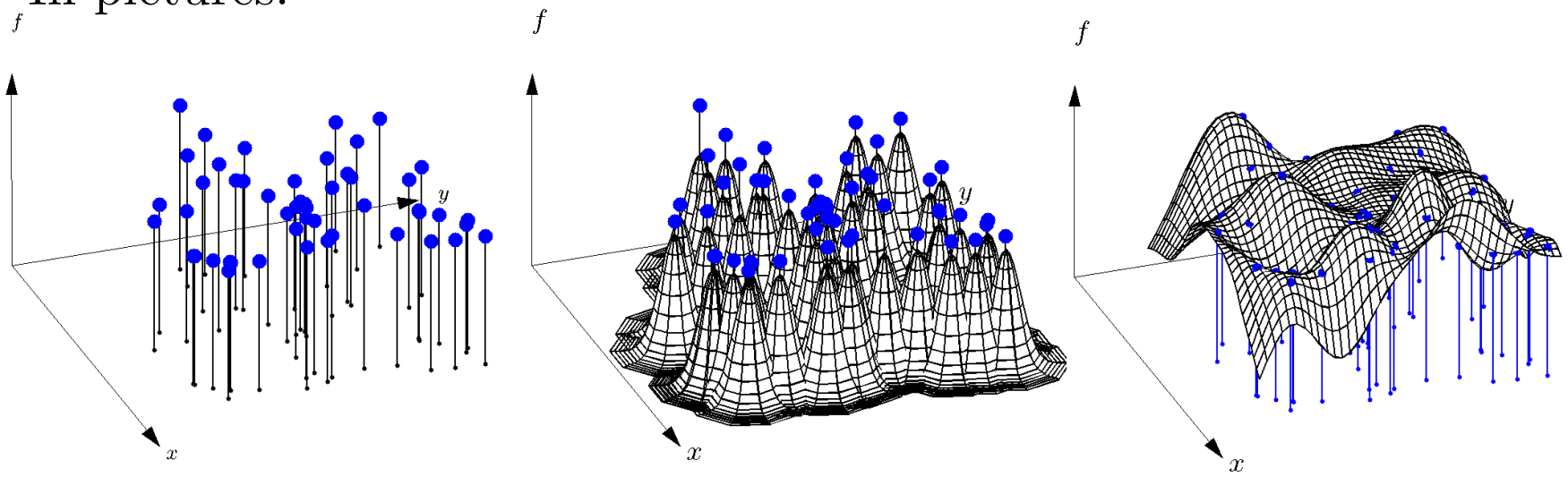
*This work is supported by NSF CMG grant DMS 0934581

Overview

- Brief review of radial basis function (RBF) approximations.
- Approximating surface divergence-free vector fields.
- Combing RBFs and the partition of unity method.
- A divergence-free RBF partition of unity method.
- Numerical results.
- Concluding remarks.

Brief review of (scalar) RBF interpolation

- In pictures:

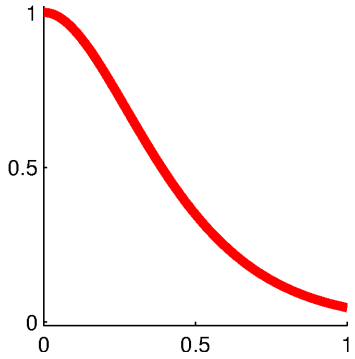


- Examples of radial kernels: $\phi(\|\mathbf{x}\|_2) := \phi(r)$

Finite-smoothness

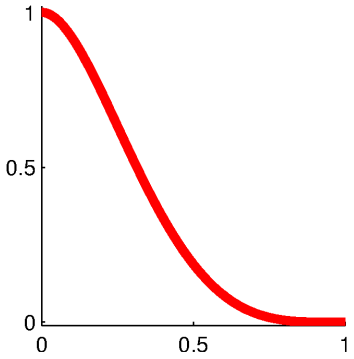
Infinite-smoothness

Matérn



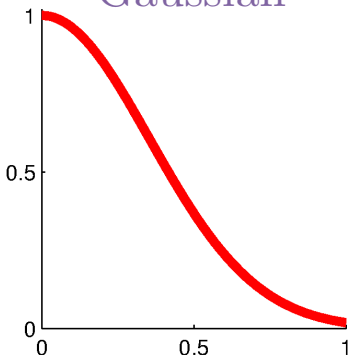
$(\epsilon r)^\nu K_\nu(\epsilon r)$

Wendland



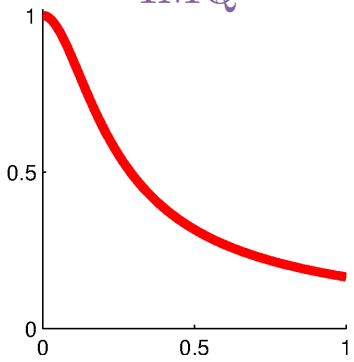
$(1-\epsilon r)_+(3\epsilon r+1)$

Gaussian



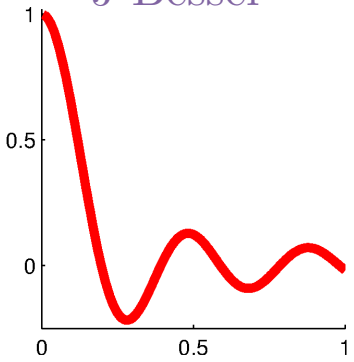
$\exp(-(\epsilon r)^2)$

IMQ



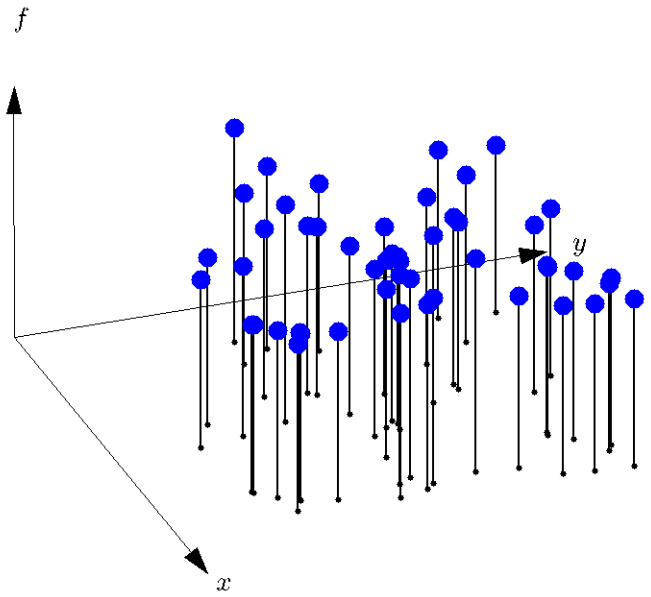
$\frac{1}{\sqrt{1+(\epsilon r)^2}}$

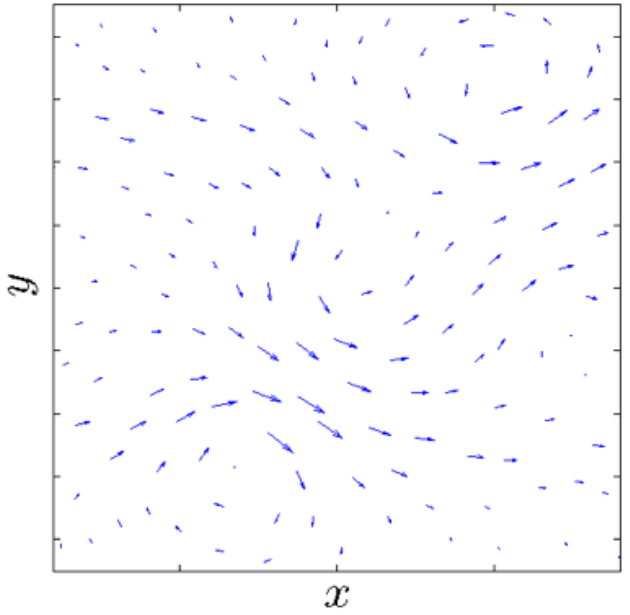
J-Bessel



$\frac{J_\nu(\epsilon r)}{(\epsilon r)^\nu}$

Scalar vs. vector RBF interpolation





Scalar interpolant:

$$s(\mathbf{x}) = \sum_{j=1}^{N} \phi(\|\mathbf{x} - \mathbf{x}_j\|) b_j, \; s(\mathbf{x}_k) = f_k, \; k = 1, \ldots, N$$

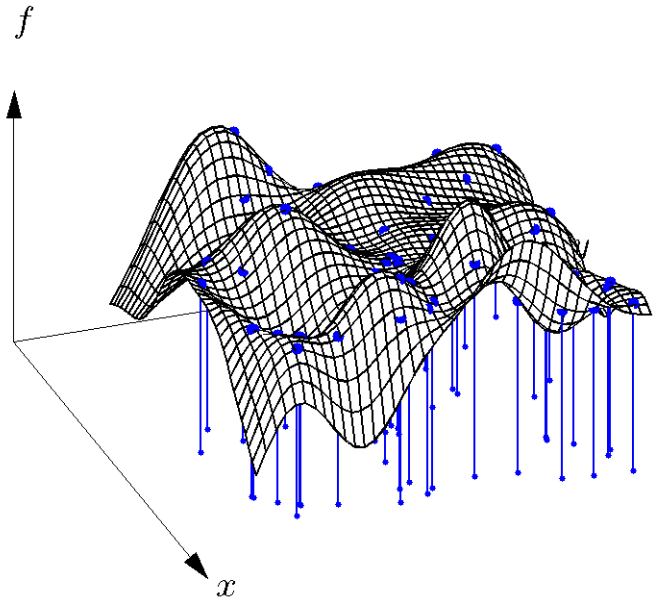
- ϕ is scalar-valued "radial" kernel.
- Nodes can be "scattered".
- Interpolation matrix is positive definite.
- Can impose additional constraints on s.
- Form of the interpolant does not depend on the topology of domain.

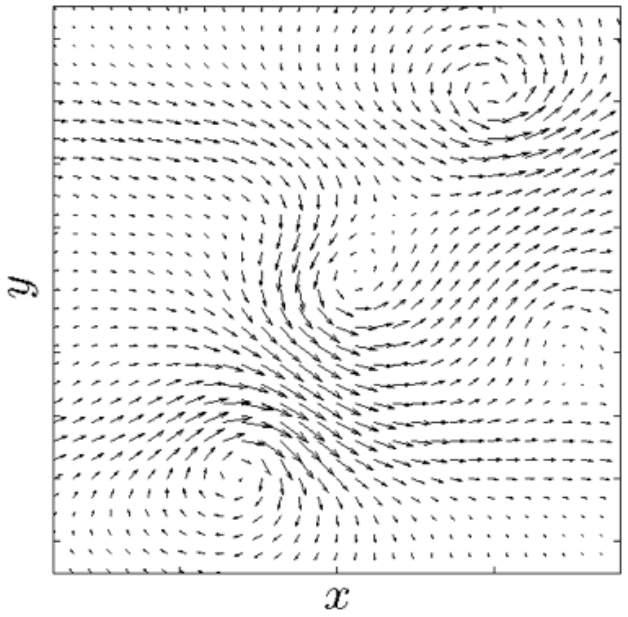
Vector interpolant:

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \Phi(\mathbf{x}, \mathbf{x}_j) \mathbf{c}_j, \; \mathbf{s}(\mathbf{x}_k) = \mathbf{u}_k, \; k = 1, \ldots, N$$

- Φ is *matrix-valued* kernel based on scalar ϕ.
- Nodes can be "scattered".
- Interpolation matrix is positive definite.
- Can impose additional constraints on $\mathbf{s}$.
- Form of the interpolant does depend on the topology of domain.

Scalar vs. vector RBF interpolation





Scalar interpolant:

$$s(\mathbf{x}) = \sum_{j=1}^{N} \phi(\|\mathbf{x} - \mathbf{x}_j\|) b_j, \; s(\mathbf{x}_k) = f_k, \; k = 1, \ldots, N$$

- ϕ is scalar-valued "radial" kernel.
- Nodes can be "scattered".
- Interpolation matrix is positive definite.
- Can impose additional constraints on s.
- Form of the interpolant does not depend on the topology of domain.

Vector interpolant:

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \Phi(\mathbf{x}, \mathbf{x}_j) \mathbf{c}_j, \; \mathbf{s}(\mathbf{x}_k) = \mathbf{u}_k, \; k = 1, \ldots, N$$

- Φ is *matrix-valued* kernel based on scalar ϕ.
- Nodes can be "scattered".
- Interpolation matrix is positive definite.
- Can impose additional constraints on $\mathbf{s}$.
- Form of the interpolant does depend on the topology of domain.

Some background

- Customizing RBF approximations for vector fields in $\mathbb{R}^d$

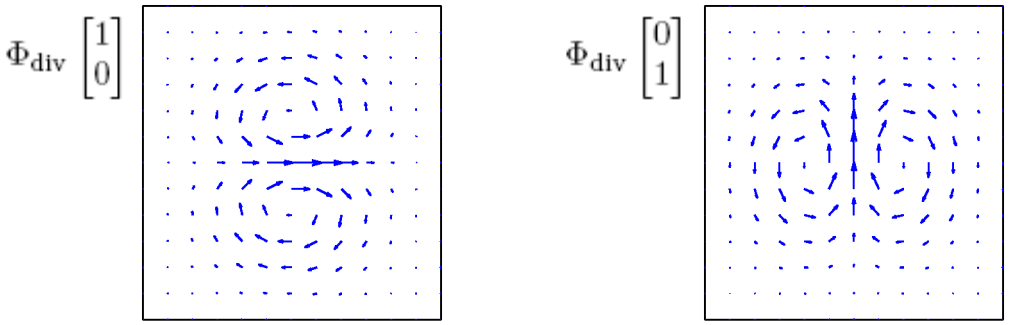
$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \Phi(\mathbf{x}, \mathbf{x}_j)\mathbf{c}_j, \ \mathbf{s}(\mathbf{x}_k) = \mathbf{u}_k, \ k = 1, \ldots, N$$

  - Developed by Narcowich and Ward (1994):

    F. J. Narcowich and J. D. Ward. Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comp.*, 63:661-687, 1994.

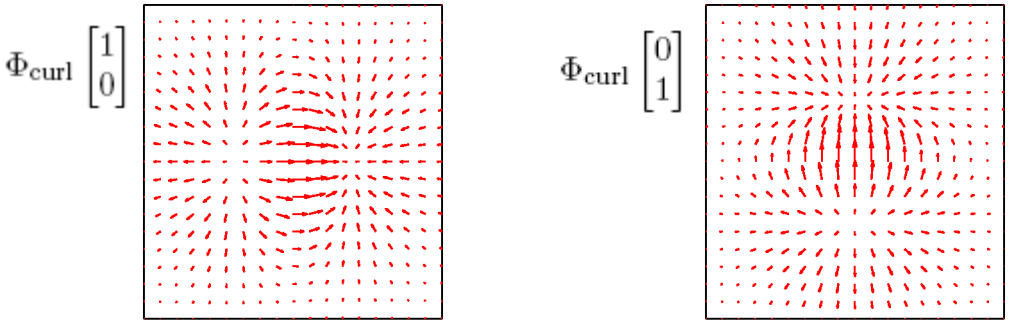
  - Divergence-free vector fields: incompressible fluids, (static) magnetic fields

    Columns of div-free matrix-valued kernel:



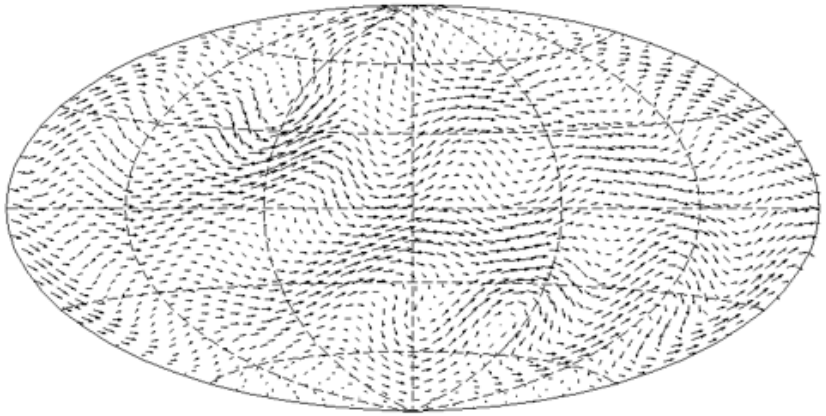
  - Curl-free vector fields: gravity fields, (static) electric fields

    Columns of curl-free matrix-valued kernel:



Some background

- Customizing RBF approximations for tangent vector fields:



  - Developed by Fuselier, Narcowich, Ward, and W:

    - F.J. Narcowich, J.D. Ward, and G.B.W. Divergence-free RBFs on Surfaces. *J. Fourier Anal. Appl.* 13 (2007), 643-663.
    - E.J. Fuselier, F.J. Narcowich, J.D. Ward, and G.B.W. Error and stability estimates for surface-divergence free RBF interpolants on the sphere. *Math. Comp.*, 78 (2009), 2157-2186.
    - E.J. Fuselier and G.B.W. Stability and error estimates for vector field interpolation and decomposition on the sphere with RBFs. *SIAM J. Numer. Anal.*, 47 (2009), 3213-3239.

  - Surface divergence-free approximation
  - Surface curl-free approximations
  - Helmholtz-Hodge decomposition

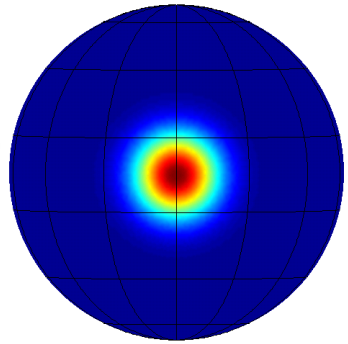
(Surface) Div, Grad, Curl, and all that

SIAM AN2012
July 9, 2012

	Spherical Coords.	**Cartesian Coords.**
Point:	$(\lambda, \theta, 1)$	(x, y, z)
Unit vectors:	$\hat{\mathbf{i}}$ = longitudinal $\hat{\mathbf{j}}$ = latitudinal $\hat{\mathbf{k}}$ = radial	$\hat{\mathbf{i}}$ = x-direction $\hat{\mathbf{j}}$ = y-direction $\hat{\mathbf{k}}$ = z-direction
Unit tangent vectors:	$\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$	$\boldsymbol{\zeta} = \frac{1}{\sqrt{1-z^2}}\begin{bmatrix} -y \\ x \\ 0 \end{bmatrix}$, $\boldsymbol{\mu} = \frac{1}{\sqrt{1-z^2}}\begin{bmatrix} -zx \\ -zy \\ 1-z^2 \end{bmatrix}$
Unit normal vector:	$\hat{\mathbf{k}}$	$\mathbf{x} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}}$
* Gradient of scalar g:	$\mathbf{u}_s = \nabla_s\, g = \frac{1}{\cos\theta}\frac{\partial g}{\partial \lambda}\hat{\mathbf{i}} + \frac{\partial g}{\partial \theta}\hat{\mathbf{j}}$	$\mathbf{u}_c = P_{\mathbf{x}}(\nabla_c\, g) = P_{\mathbf{x}}\left(\frac{\partial g}{\partial x}\hat{\mathbf{i}} + \frac{\partial g}{\partial y}\hat{\mathbf{j}} + \frac{\partial g}{\partial z}\hat{\mathbf{k}}\right)$
Surface divergence of $\mathbf{u}$:	$\nabla_s \cdot \mathbf{u}_s = \frac{1}{\cos\theta}\frac{\partial u_s}{\partial \lambda} + \frac{\partial v_s}{\partial \theta}$	$(P_{\mathbf{x}}\nabla_c) \cdot \mathbf{u}_c = \nabla_c \cdot \mathbf{u}_c - \mathbf{x} \cdot \nabla(\mathbf{u}_c \cdot \mathbf{x}) + \mathbf{x} \cdot \mathbf{u}_c$
* Curl of a scalar f:	$\mathbf{u}_s = \hat{\mathbf{k}} \times (\nabla_s f) = -\frac{\partial f}{\partial \theta}\hat{\mathbf{i}} + \frac{1}{\cos\theta}\frac{\partial f}{\partial \lambda}\hat{\mathbf{j}}$	$\mathbf{u}_c = \mathbf{x} \times (P_{\mathbf{x}}\nabla_c f) = Q_{\mathbf{x}}P_{\mathbf{x}}(\nabla_c f) = Q_{\mathbf{x}}(\nabla_c f)$
Surface curl of a vector $\mathbf{u}$:	$\hat{\mathbf{k}} \cdot (\nabla_s \times \mathbf{u}_s) = -\nabla_s \cdot (\hat{\mathbf{k}} \times \mathbf{u}_s)$	$\mathbf{x} \cdot ((P_{\mathbf{x}}\nabla_c) \times \mathbf{u}_c) = -\nabla_c \cdot (Q_{\mathbf{x}}\mathbf{u}_c)$

$$P_{\mathbf{x}} = I - \mathbf{x}\mathbf{x}^T = \begin{bmatrix} 1-x^2 & -xy & -xz \\ -xy & 1-y^2 & -yz \\ -xz & -yz & 1-z^2 \end{bmatrix} \qquad Q_{\mathbf{x}} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

Surface div-free RBFs on the sphere

- Use extrinsic (Cartesian) coordinates, $\mathbf{x} = (x, y, z) \in \mathbb{S}^2$.
- Start with a radial kernel centered at $\mathbf{y} \in \mathbb{S}^2$: $\phi(\|\mathbf{x} - \mathbf{y}\|)$ 
- Construct 3-by-3 *matrix-valued* function

$$\begin{aligned}\Phi_{\text{div}}(\mathbf{x}, \mathbf{y}) &= (Q_{\mathbf{x}}\nabla_{\mathbf{x}})(Q_{\mathbf{y}}\nabla_{\mathbf{y}})^T \phi(\|\mathbf{x} - \mathbf{y}\|) \\ &= Q_{\mathbf{x}}(\nabla\nabla^T \phi(\|\mathbf{x} - \mathbf{y}\|))Q_{\mathbf{y}}\end{aligned}$$

- If $\mathbf{c} = (c_1, c_2, c_3)^T$ is tangent to $\mathbb{S}^2$ at $\mathbf{y}$ then $\Phi_{\text{div}}(\mathbf{x}, \mathbf{y})\mathbf{c}$ is tangent to $\mathbb{S}^2$ at $\mathbf{x}$.
- Furthermore,

$$\begin{aligned}\Phi_{\text{div}}(\mathbf{x}, \mathbf{y})\mathbf{c} &= \left[Q_{\mathbf{x}}(\nabla\nabla^T \phi(\|\mathbf{x} - \mathbf{y}\|))Q_{\mathbf{y}}\right]\mathbf{c} \\ &= Q_{\mathbf{x}}\nabla\left[\nabla^T\left(\phi(\|\mathbf{x} - \mathbf{y}\|)Q_{\mathbf{y}}\mathbf{c}\right)\right] \\ &= Q_{\mathbf{x}}(\nabla f).\end{aligned}$$

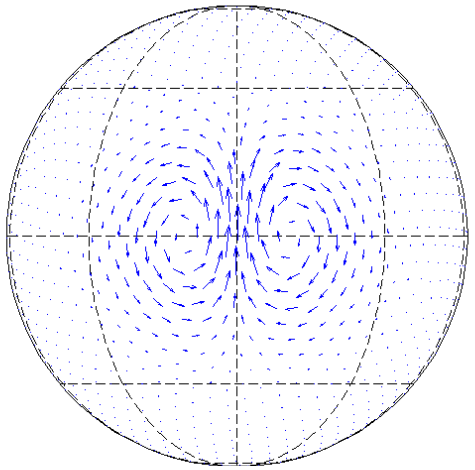
  Thus, $\Phi_{\text{div}}(\mathbf{x}, \mathbf{y})\mathbf{c}$ is surface divergence-free.

- Idea can be extended to other smooth orientable manifolds.

Surface div-free RBF interpolation

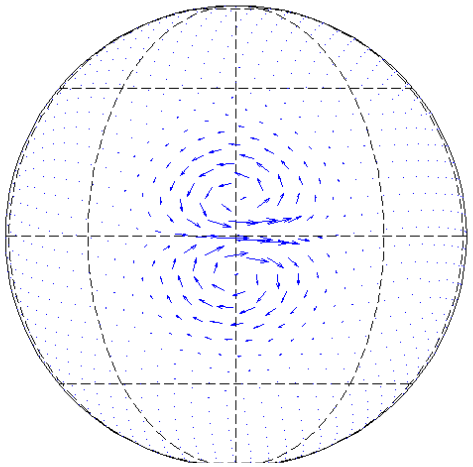
Illustration of the div-free basis:

Meridional basis



$\Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j)\boldsymbol{\mu}_j$

Zonal basis



$\Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j)\boldsymbol{\zeta}_j$

Interpolation procedure:

1. For each distinct node $\mathbf{x}_j$, center a $\Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j)\boldsymbol{\mu}_j$ and $\Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j)\boldsymbol{\zeta}_j$.

2. Linearly combine these *vector-valued* functions to satisfy the interpolation conditions.

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j) \underbrace{[a_j\boldsymbol{\mu}_j + b_j\boldsymbol{\zeta}_j]}_{\mathbf{c}_j}$$

3. Solve $2N$-by-$2N$ linear system for unknown coefficients.

Surface div-free RBF interpolation

Surface div-free interpolant: $\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j) \underbrace{[a_j \boldsymbol{\mu}_j + b_j \boldsymbol{\zeta}_j]}_{\mathbf{c}_j}$

Nice properties:

- Existence and uniqueness (linear system for determining $\mathbf{c}_j$ is positive definite).
- Interpolated field will be (analytically) divergence-free.
- Similar error estimates to scalar RBF interpolation on the sphere.
- Can obtain a stream function for the interpolated field "for free":

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{N} \underbrace{Q_{\mathbf{x}} \nabla \nabla^T \phi(\|\mathbf{x} - \mathbf{x}_j\|) Q_{\mathbf{x}_j}^T}_{\Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j)} \mathbf{c}_j = Q_{\mathbf{x}} \nabla \underbrace{\left[\sum_{j=1}^{N} \nabla^T \phi(\|\mathbf{x} - \mathbf{x}_j\|) Q_{\mathbf{x}_j}^T \mathbf{c}_j \right]}_{\psi(\mathbf{x})}$$

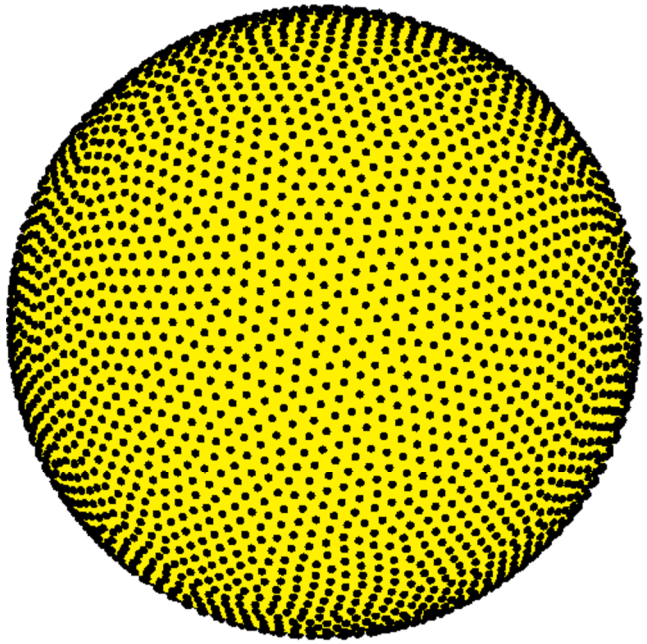
Not-so-nice properties:

- Shape parameter ill-conditioning issues when solving directly for $\mathbf{c}_j$
- Poor computational scaling: $O(N^3)$ to find $\mathbf{c}_j$ and $O(N)$ to evaluate $\mathbf{s}(\mathbf{x})$.

A partition of unity method (PUM) for approximations with *scalar* RBFs on the sphere

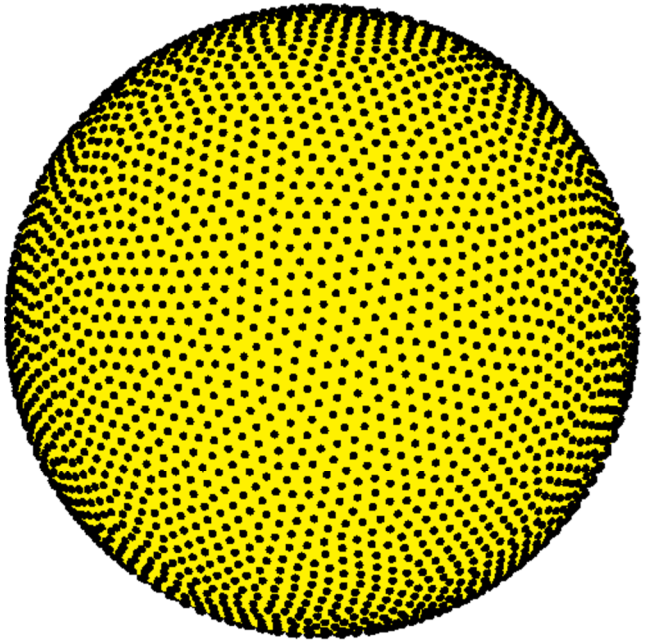
Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^{N} \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:

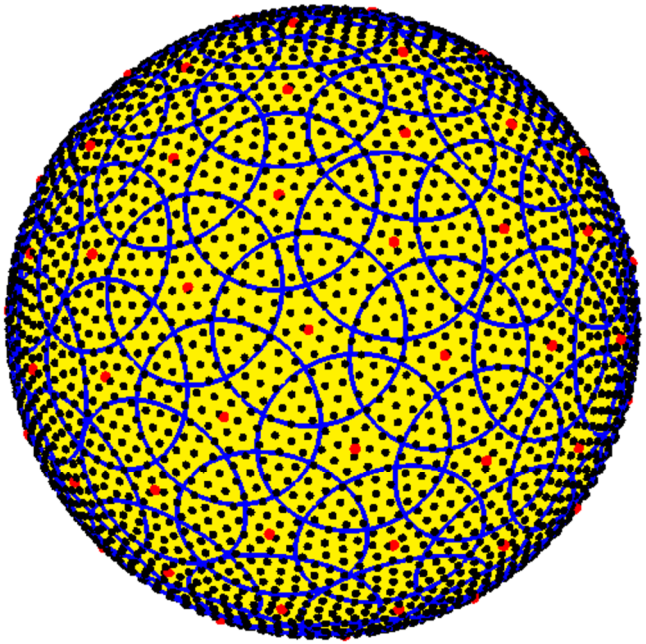


Key Steps:

1. Generate a set of overlapping patches (spherical caps) $\Omega = \{\Omega_k\}_{k=1}^M$ that creates a uniform partition of the sphere with respect to the density of the nodes in X, and each patch contains roughly n nodes of X.

Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



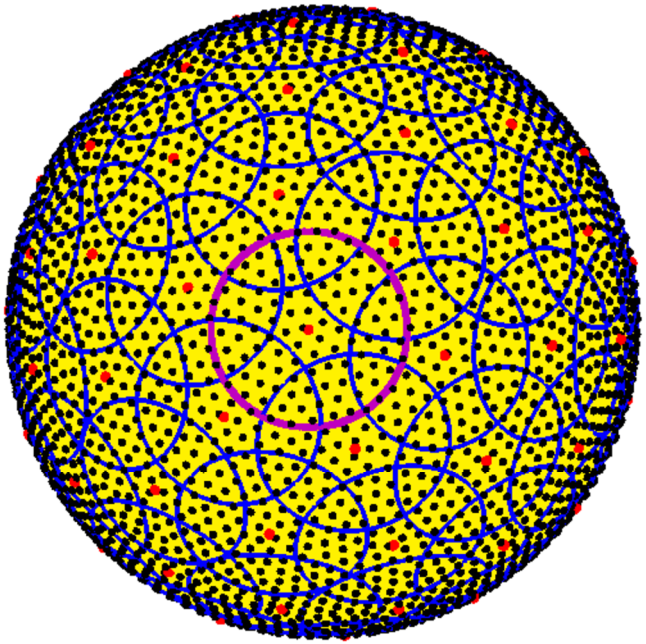
M total patches
n nodes per patch
ξ_k = center of patch Ω_k

Key Steps:

1. Generate a set of overlapping patches (spherical caps) $\Omega = \{\Omega_k\}_{k=1}^M$ that creates a uniform partition of the sphere with respect to the density of the nodes in X, and each patch contains roughly n nodes of X.

Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



M total patches
n nodes per patch
ξ_k = center of patches Ω_k

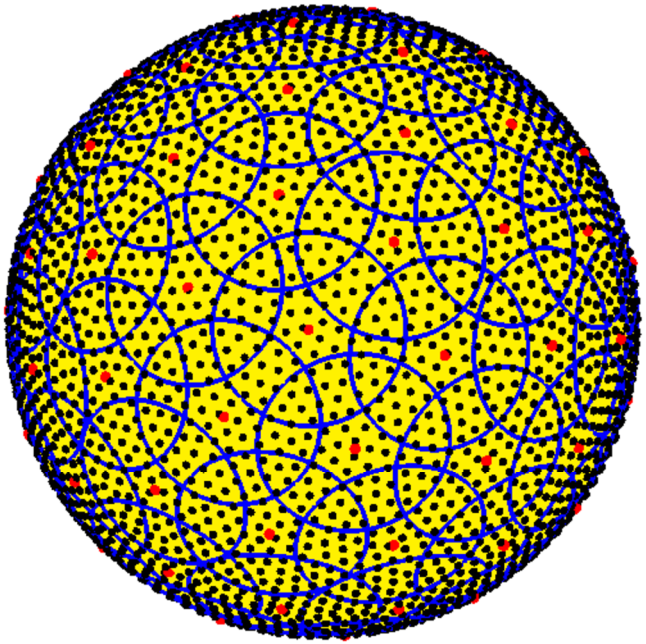
Key Steps:

2. Letting X_k denote the set of nodes in patch Ω_k, construct RBF interpolants s_k, for $k = 1, \ldots, M$:

$$s_k(\mathbf{x}) = \sum_{j=1}^{n} c_j^k \phi(\|\mathbf{x} - \mathbf{x}_j^k\|)$$

Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^{N} \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



M total patches
n nodes per patch
ξ_k = center of patches Ω_k

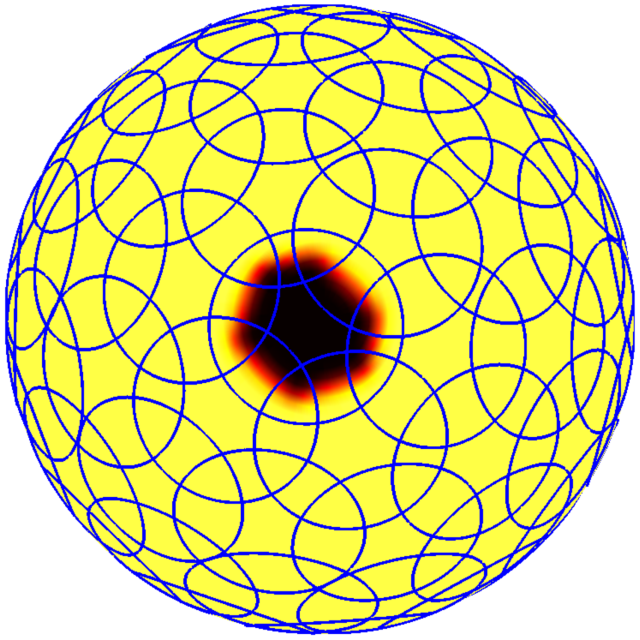
Key Steps:

3. Define weight functions $w_k : \mathbb{S}^2 \to \mathbb{R}$, $k = 1 \ldots, M$, with the properties that each w_k is compactly supported over Ω_k and the set of all w_k form a partition-of-unity over Ω:

$$\sum_{k=1}^{M} w_k(\mathbf{x}) \equiv 1, \, \mathbf{x} \in \Omega$$

Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^{N} \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



M total patches
n nodes per patch
ξ_k = center of patches Ω_k

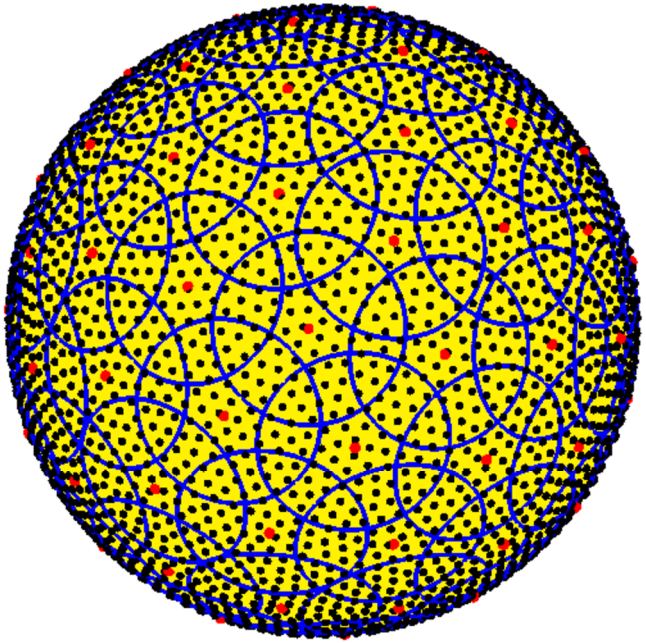
Key Steps:

3. Define weight functions $w_k : \mathbb{S}^2 \to \mathbb{R}$, $k = 1 \ldots, M$, with the properties that each w_k is compactly supported over Ω_k and the set of all w_k form a partition-of-unity over Ω:

$$\sum_{k=1}^{M} w_k(\mathbf{x}) \equiv 1, \, \mathbf{x} \in \Omega$$

Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^{N} \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



M total patches
n nodes per patch
ξ_k = center of patches Ω_k

Key Steps:

Weight function details:

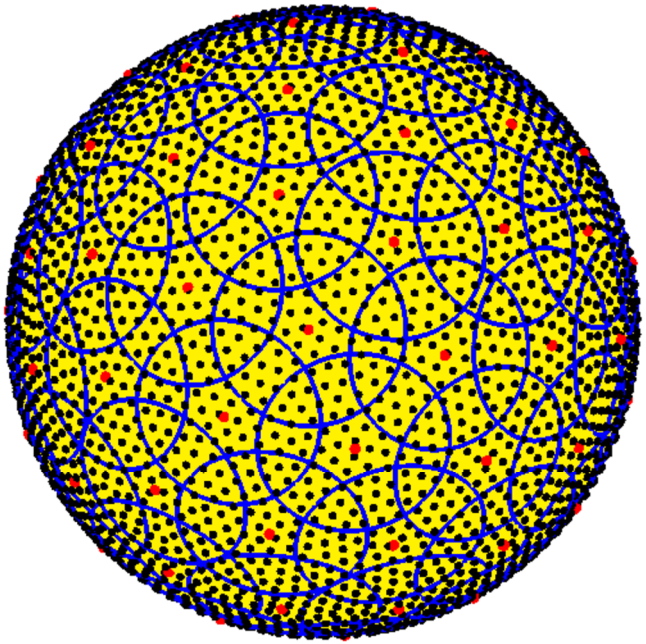
$$\chi_k(\mathbf{x}) = \chi\left(\frac{\|\mathbf{x} - \boldsymbol{\xi}_k\|}{\rho_k}\right)$$

$$w_k(\mathbf{x}) = \frac{\chi_k(\mathbf{x})}{\sum_{i=1}^{M} \chi_i(\mathbf{x})}$$

ρ_k = radius of patch Ω_k
χ has compact support over $[0, 1]$

Scalar RBFs and PUM on the sphere

- Consider $X = \{\mathbf{x}_j\}_{j=1}^{N} \subset \mathbb{S}^2$, where $\mathbf{x}_j = (x_j, y_j, z_j)$:



M total patches
n nodes per patch
ξ_k = center of patches Ω_k

Key Steps:

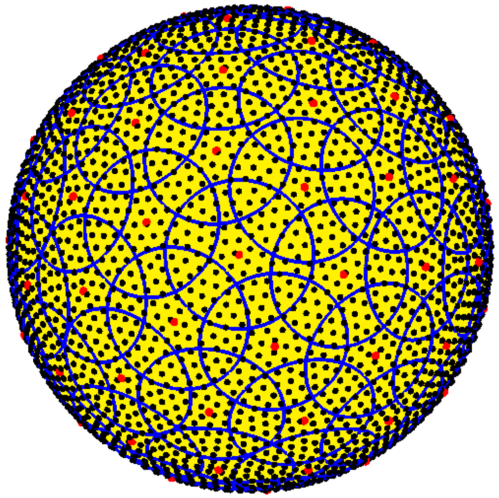
4. Create a global interpolant for X as

$$s(\mathbf{x}) = \sum_{k=1}^{M} w_k(\mathbf{x}) s_k(\mathbf{x})$$

R, Cavoretto and A. De Rossi. Spherical interpolation using the partition of unity method: An efficient and flexible algorithm. *Appl. Math. Lett.*, In press 2011.

Back to surface divergence-free approximations

Surface div-free RBFs and PUM



- How can we adapt this method for surface div-free RBFs?

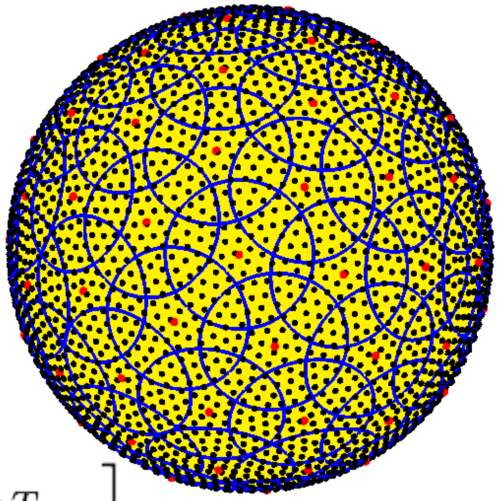
Bad idea:

- Construct global vector approximant like the scalar case:

$$\mathbf{s}(\mathbf{x}) = \sum_{k=1}^{M} w_k(\mathbf{x})\mathbf{s}_k(\mathbf{x})$$

$\mathbf{s}(\mathbf{x})$ interpolates but is not divergence-free!

Surface div-free RBFs and PUM



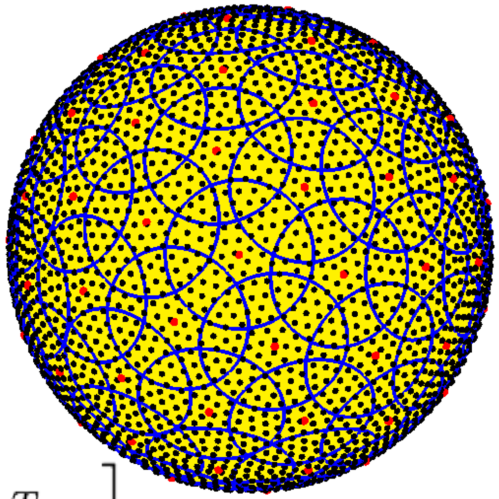
- How can we adapt this method for surface div-free RBFs?

Better idea:

- For each patch Ω_k, construct the div-free RBF interpolant $\mathbf{s}_k(\mathbf{x})$ and extract a *stream function*:

$$\mathbf{s}_k(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{\Phi}_{\text{div}}(\mathbf{x}, \mathbf{x}_j^k)\mathbf{c}_j^k = Q_{\mathbf{x}} \nabla \underbrace{\left[\sum_{j=1}^{n} \nabla^T \phi(\|\mathbf{x} - \mathbf{x}_j^k\|) Q_{\mathbf{x}_j}^T \mathbf{c}_j \right]}_{\psi_k(\mathbf{x})}$$

Surface div-free RBFs and PUM



- How can we adapt this method for surface div-free RBFs?

  Better idea:

  - For each patch Ω_k, construct the div-free RBF interpolant $\mathbf{s}_k(\mathbf{x})$ and extract a *stream function*:

$$\mathbf{s}_k(\mathbf{x}) = \sum_{j=1}^{n} \Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j^k)\mathbf{c}_j^k = Q_{\mathbf{x}}\nabla \underbrace{\left[\sum_{j=1}^{n} \nabla^T \phi(\|\mathbf{x} - \mathbf{x}_j^k\|) Q_{\mathbf{x}_j}^T \mathbf{c}_j\right]}_{\psi_k(\mathbf{x})}$$

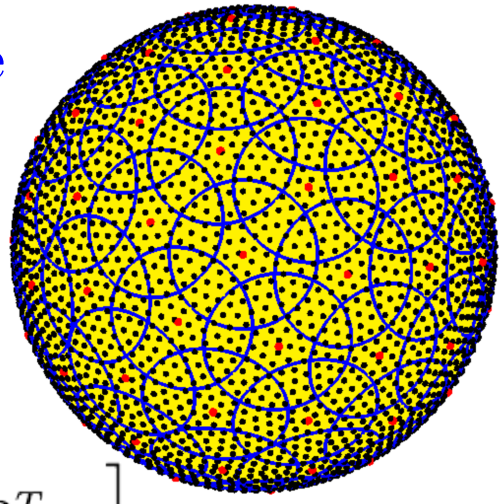
  - Construct a global div-free approximant by combining stream functions and partition of unity weight functions:

$$\mathbf{s}(\mathbf{x}) = Q_{\mathbf{x}}\nabla \sum_{k=1}^{M} w_k(\mathbf{x})\psi_k(\mathbf{x}) = \sum_{k=1}^{M} Q_{\mathbf{x}}\nabla(w_k(\mathbf{x})\psi_k(\mathbf{x}))$$

$$= \sum_{k=1}^{M} w_k(\mathbf{x})\mathbf{s}_k(\mathbf{x}) + \sum_{k=1}^{M} \psi_k(\mathbf{x}) Q_{\mathbf{x}}\nabla(w_k(\mathbf{x}))$$

Still an issue with this construction.

Surface div-free RBFs and PUM



- How can we adapt this method for surface div-free RBFs?

  Best idea:

  - For each patch Ω_k, construct the div-free RBF interpolant $\mathbf{s}_k(\mathbf{x})$ and extract a *stream function*:

$$\mathbf{s}_k(\mathbf{x}) = \sum_{j=1}^{n} \Phi_{\text{div}}(\mathbf{x}, \mathbf{x}_j^k)\mathbf{c}_j^k = Q_{\mathbf{x}}\nabla \underbrace{\left[\sum_{j=1}^{n} \nabla^T \phi(\|\mathbf{x} - \mathbf{x}_j^k\|) Q_{\mathbf{x}_j}^T \mathbf{c}_j\right]}_{\psi_k(\mathbf{x})}$$

  - "Adjust" stream functions to make them consistent across patches.

  - Construct a global div-free approximant by combining *adjusted* stream functions and partition of unity weight functions:

$$\mathbf{s}(\mathbf{x}) = Q_{\mathbf{x}}\nabla \sum_{k=1}^{M} w_k(\mathbf{x})\widetilde{\psi}_k(\mathbf{x}) = \sum_{k=1}^{M} Q_{\mathbf{x}}\nabla(w_k(\mathbf{x})\widetilde{\psi}_k(\mathbf{x}))$$

$$= \sum_{k=1}^{M} w_k(\mathbf{x})\mathbf{s}_k(\mathbf{x}) + \sum_{k=1}^{M} \widetilde{\psi}_k(\mathbf{x}) Q_{\mathbf{x}}\nabla(w_k(\mathbf{x}))$$

(Does not interpolate the field)

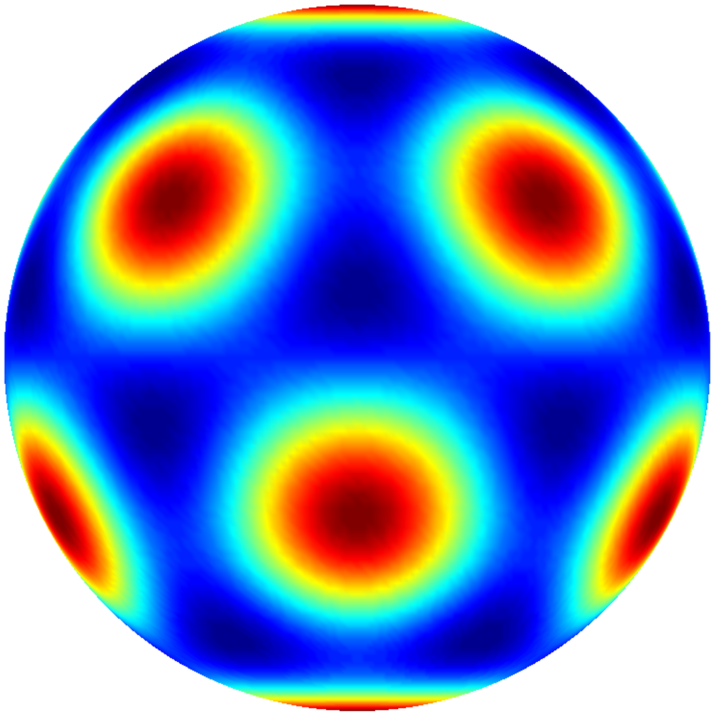
Example

- Generate field from stream function:

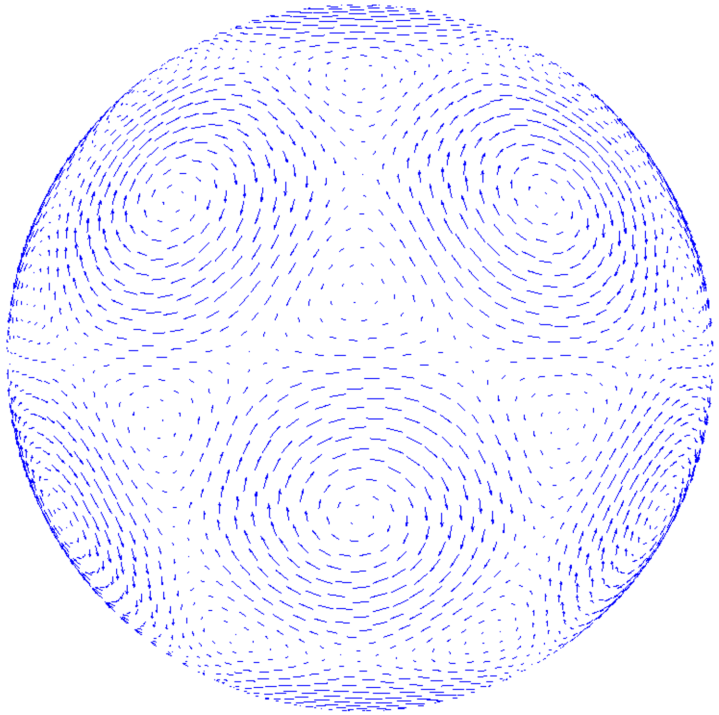
$$u(\mathbf{x}) = Q_{\mathbf{x}} \nabla[\psi(\mathbf{x})] = Q_{\mathbf{x}} \nabla \left[Y_6^0(\mathbf{x}) + \sqrt{\frac{14}{11}} Y_6^5(\mathbf{x}) \right]$$

- Field sampled at N=4096 nodes, number of patches M=82, nodes-per-patch $n = 100$, multiquadric RBF used for constructing Φ_{div}.

Target stream function



Target vector field



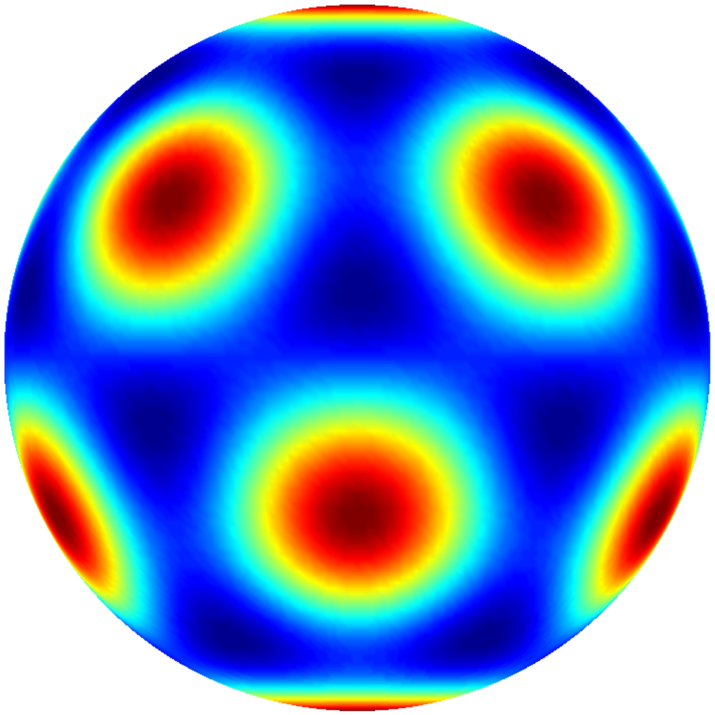
Example

- Generate field from stream function:

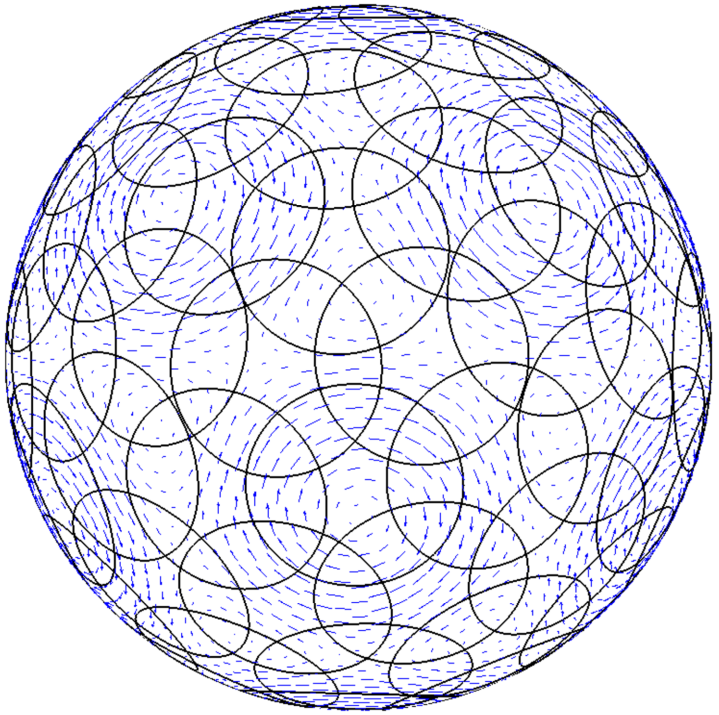
$$u(\mathbf{x}) = Q_{\mathbf{x}} \nabla[\psi(\mathbf{x})] = Q_{\mathbf{x}} \nabla \left[Y_6^0(\mathbf{x}) + \sqrt{\frac{14}{11}} Y_6^5(\mathbf{x}) \right]$$

- Field sampled at N=4096 nodes, number of patches M=82, nodes-per-patch $n = 100$, multiquadric RBF used for constructing Φ_{div}.

Target stream function



Target vector field



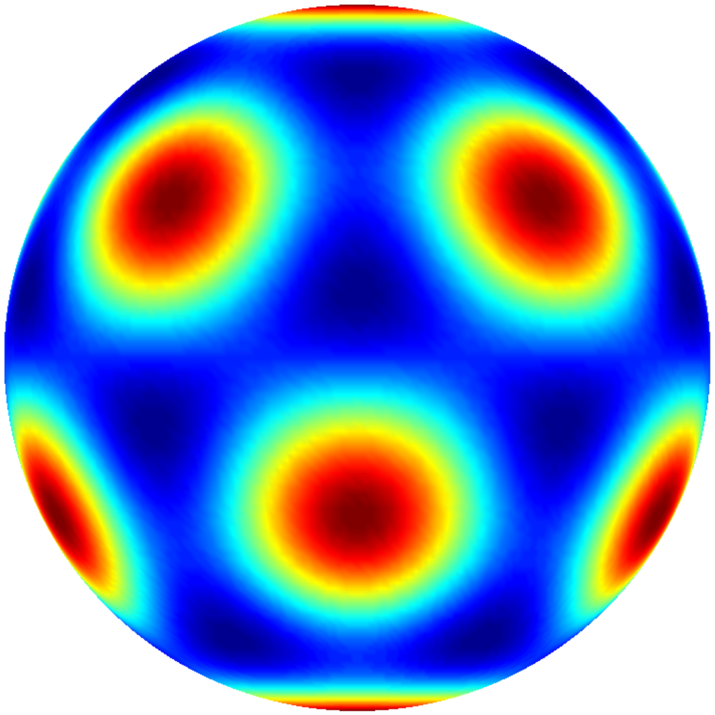
Example

- Generate field from stream function:

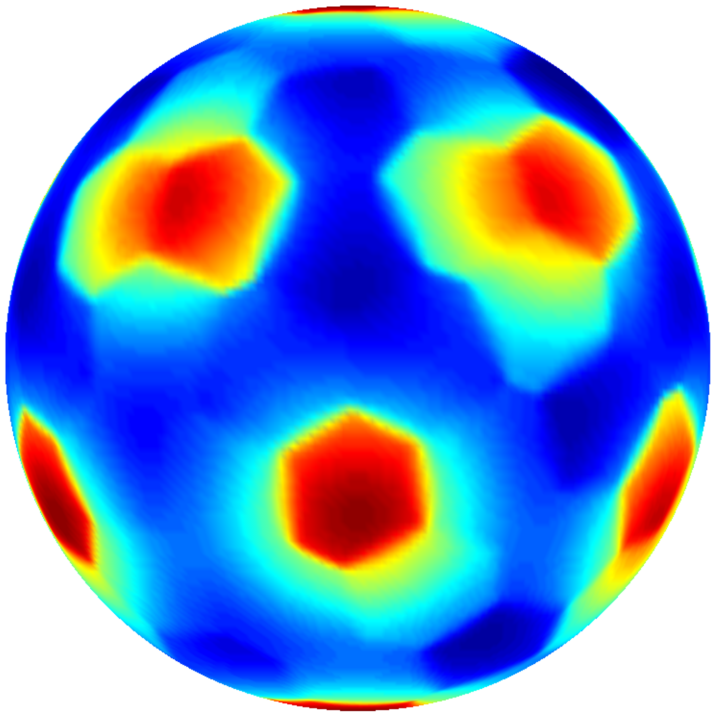
$$u(\mathbf{x}) = Q_{\mathbf{x}} \nabla[\psi(\mathbf{x})] = Q_{\mathbf{x}} \nabla \left[Y_6^0(\mathbf{x}) + \sqrt{\frac{14}{11}} Y_6^5(\mathbf{x}) \right]$$

- Field sampled at N=4096 nodes, number of patches M=82, nodes-per-patch $n = 100$, multiquadric RBF used for constructing Φ_{div}.

Target stream function



Reconstructed stream function
without adjusted ψ_k



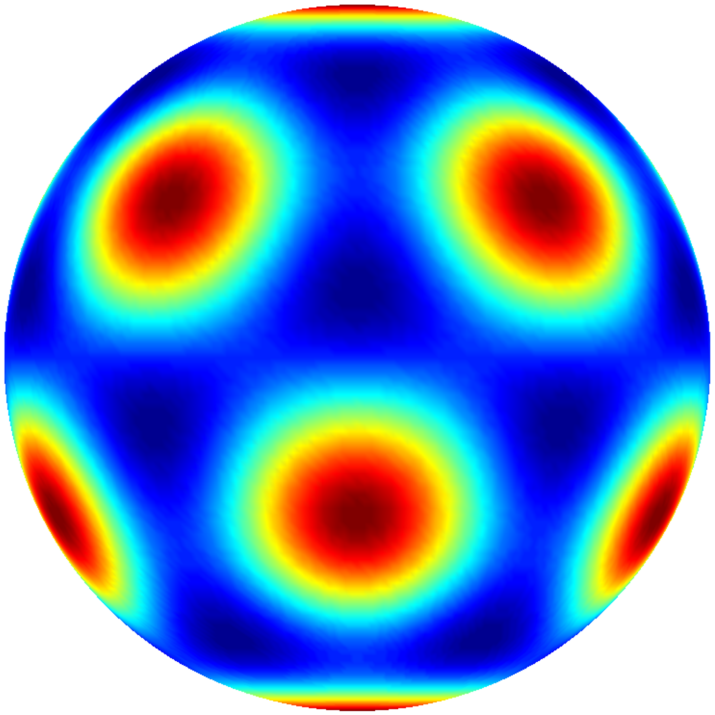
Example

- Generate field from stream function:

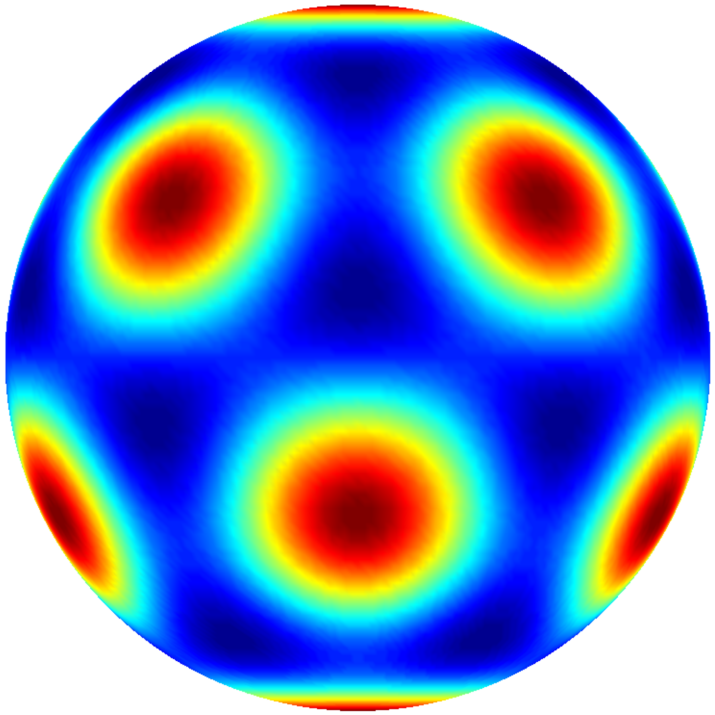
$$u(\mathbf{x}) = Q_{\mathbf{x}} \nabla[\psi(\mathbf{x})] = Q_{\mathbf{x}} \nabla \left[Y_6^0(\mathbf{x}) + \sqrt{\frac{14}{11}} Y_6^5(\mathbf{x}) \right]$$

- Field sampled at N=4096 nodes, number of patches M=82, nodes-per-patch $n = 100$, multiquadric RBF used for constructing Φ_{div}.

Target stream function



Reconstructed stream function with adjusted ψ_k

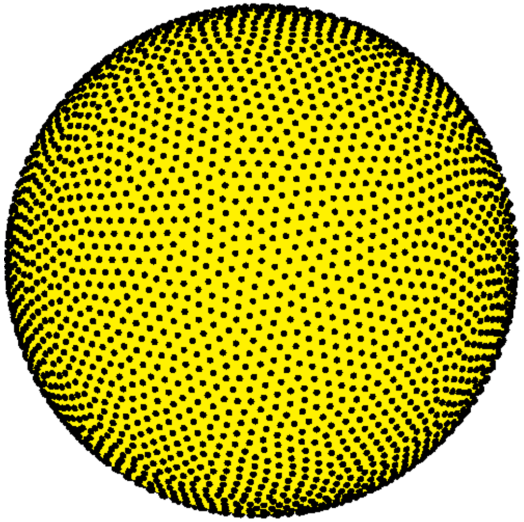


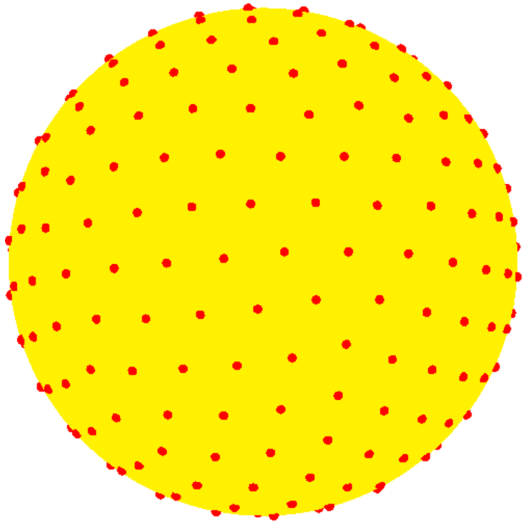
Example for nodes and patches

Nodes: We use the *maximal determinant* (MD) node sets, which are quasi-uniformly distributed over the sphere. R.S. Womersley, I. Sloan (2001)

Patches: We use *minimum energy* (ME) points, which are also quasi-uniformly distributed over the sphere. D.P. Hardin, E.B. Saff (2004)

Nodes



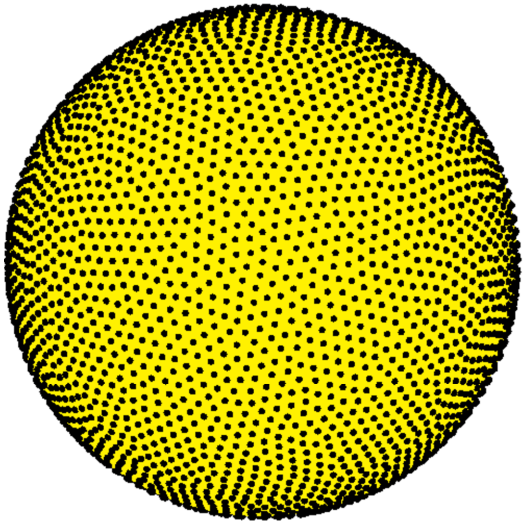


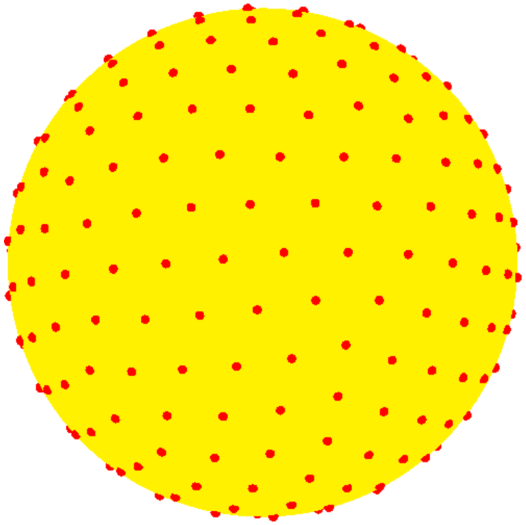
Patch centers

Parameters: Given N nodes, there are 2 parameters to choose for determining the total number of patches M:

- n = approx. number of nodes in each patch;
- q = measure of the amount the patches overlap.

Choosing the nodes and patches





- Using the quasi-uniformity of the nodes and patches, we compute the **radii of the patches** using the approximation:

$$\rho \approx 2\sqrt{\frac{n}{N}}$$

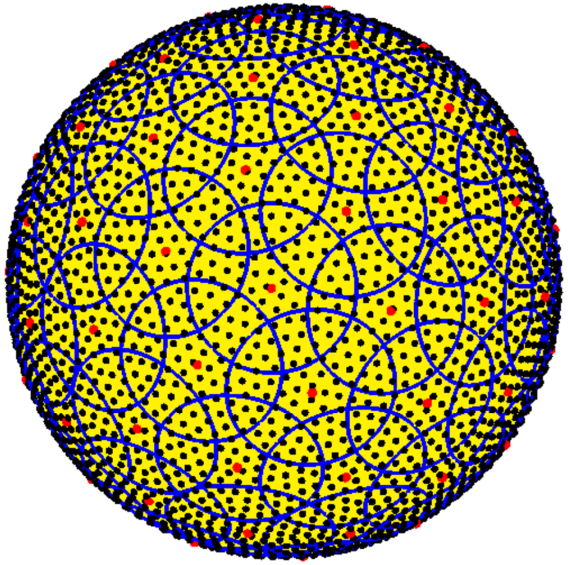
- The overlap parameter q determines the **average number of patches a node belongs to**, and satisfies the relationship:

$$\frac{4\pi}{M} \approx \frac{\pi\rho^2}{q} \qquad \Longrightarrow M = \left\lceil q\frac{N}{n} \right\rceil$$

Choosing the nodes and patches

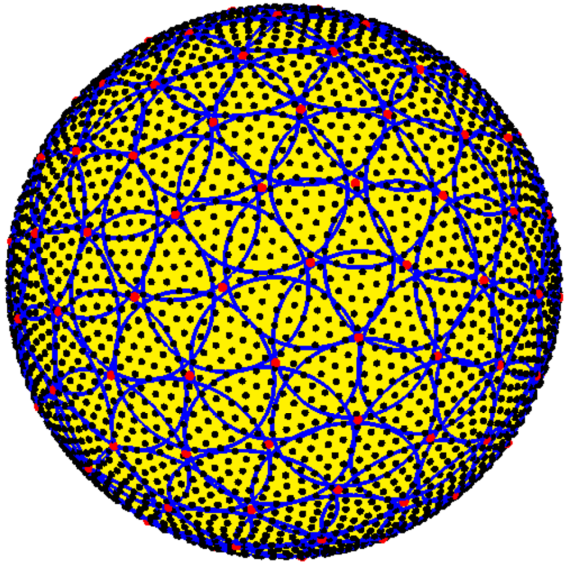
- Illustration of the patches for N=4096, n=100, and different q:

q=2



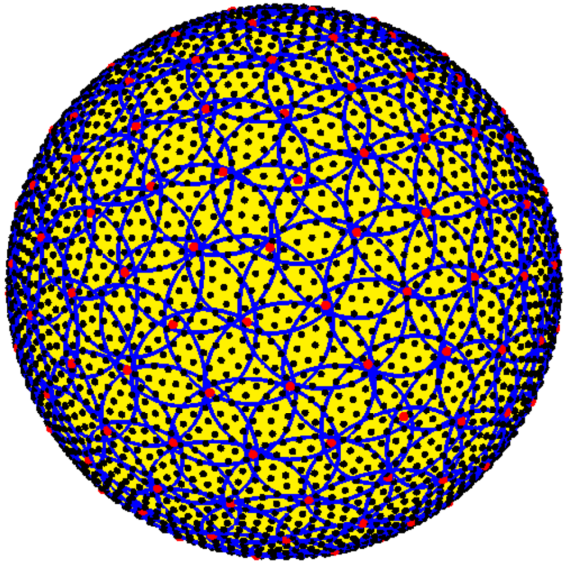
M=82

q=3



M=123

q=4



M=184

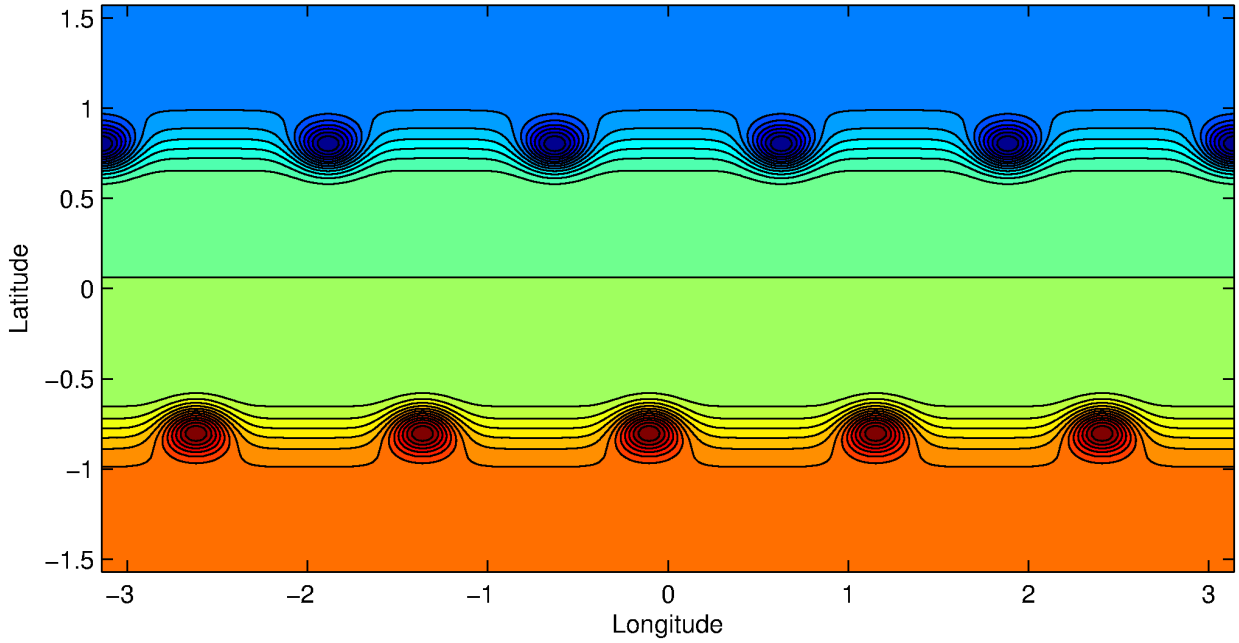
Cost comparison using direct methods

Collocation	**Global RBF**	**RBF-PUM**
Construction:	$O(N^3)$	$O(n^3M)$
Cost per evaluation:	$O(N)$	$O(n)$

Numerical example: convergence test

- Details for the numerical test:
  - Smooth div-free target field.
  - Multiquadric radial kernel to generate Φ_{div}
  - Shape parameter is fixed for all N.
  - Overlap parameter q=3.

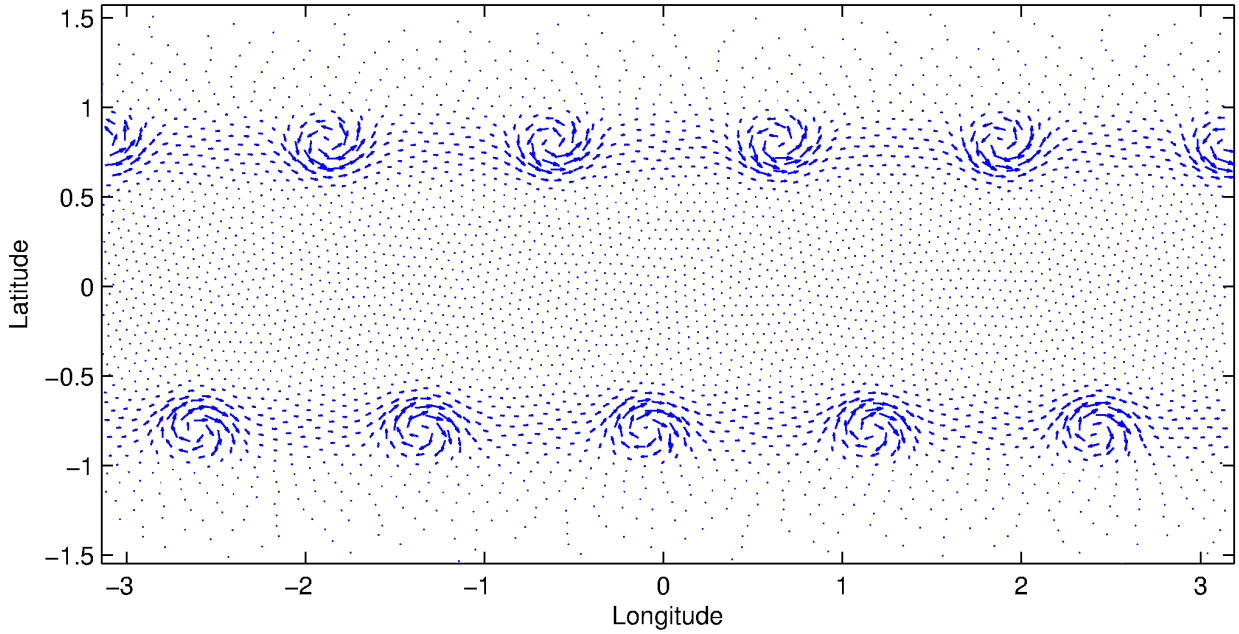
- Stream function for the numerical test:



Numerical example: convergence test

- Details for the numerical test:
  - Smooth div-free target field.
  - Multiquadric radial kernel to generate Φ_{div}
  - Shape parameter is fixed for all N.
  - Overlap parameter q=3.

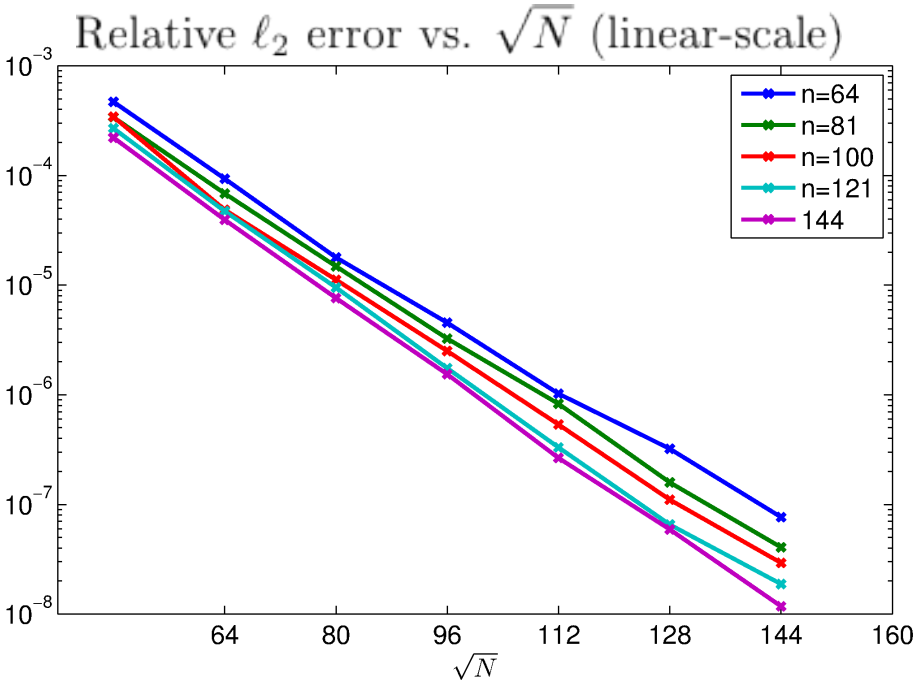
- Div-free vector field for the numerical test:

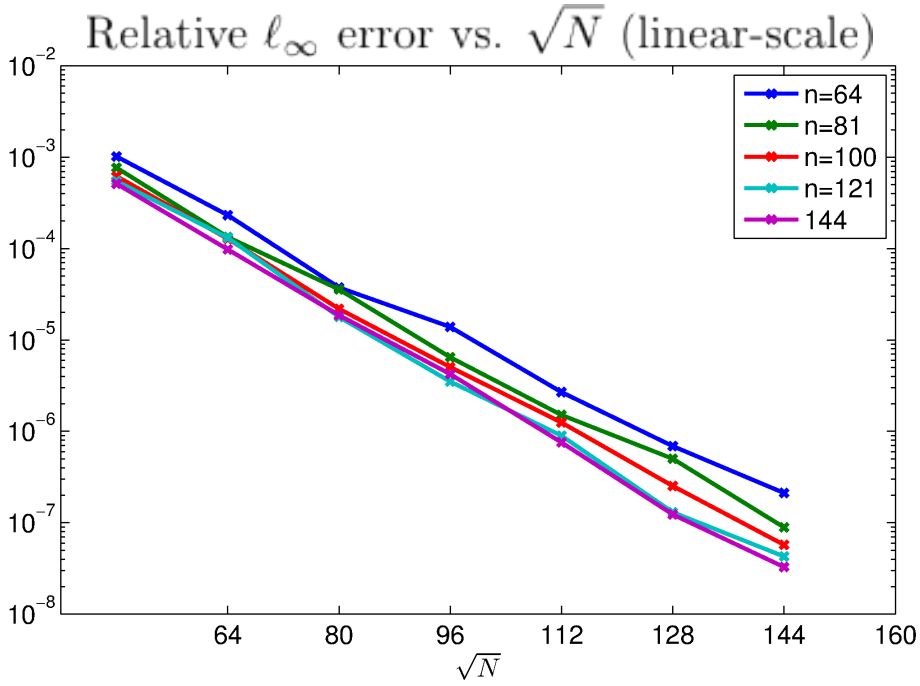


Numerical example: convergence test

- Convergence plots for increasing N and n.

Stream function results:

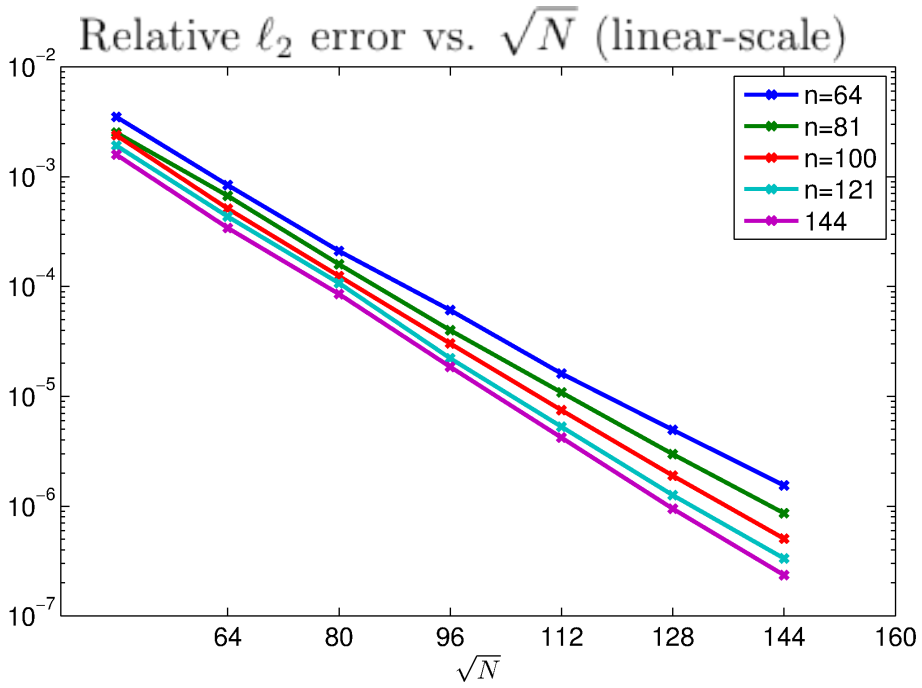


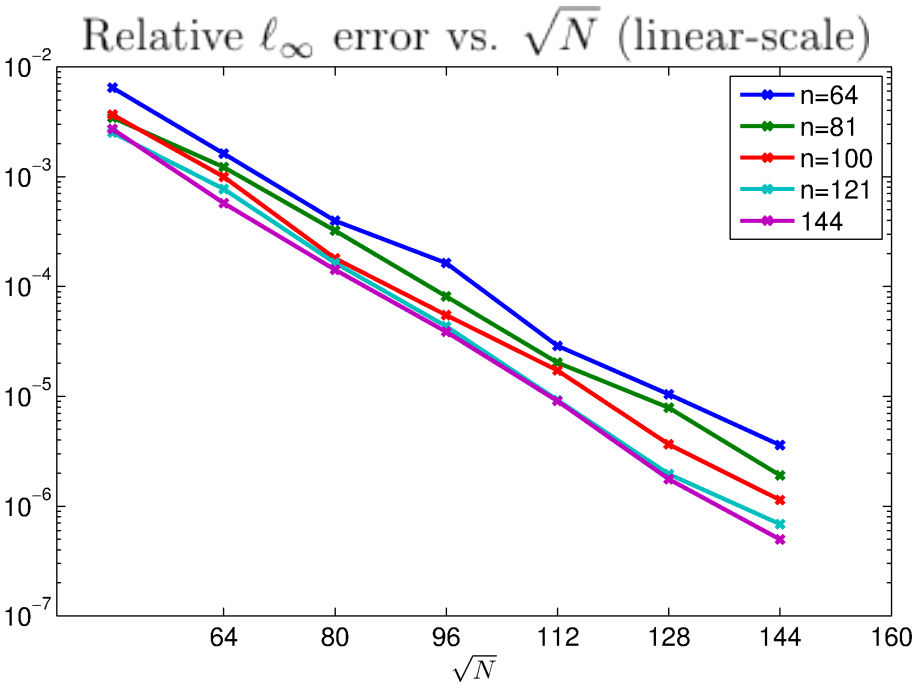


Numerical example: convergence test

- Convergence plots for increasing N and n.

Div-free field results:

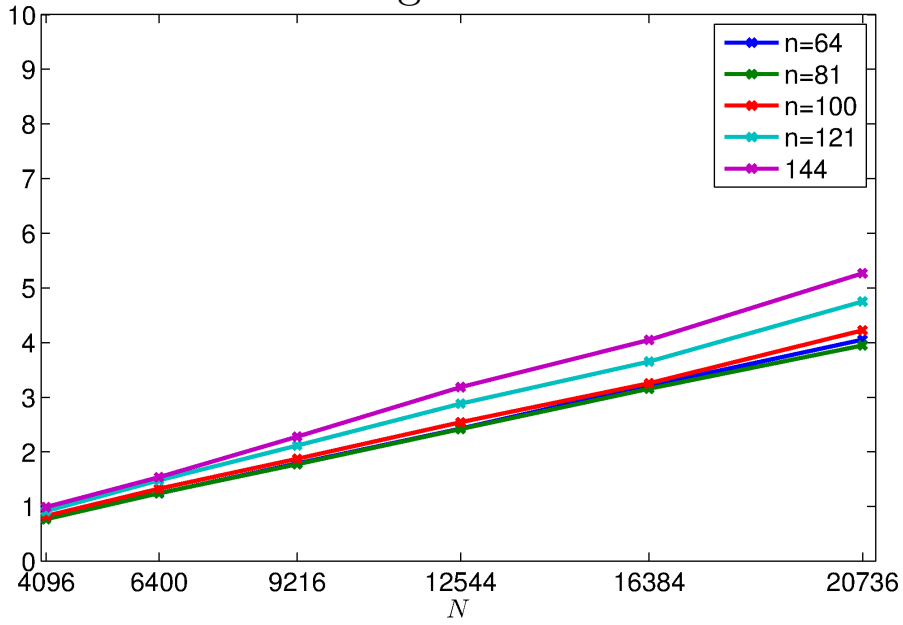




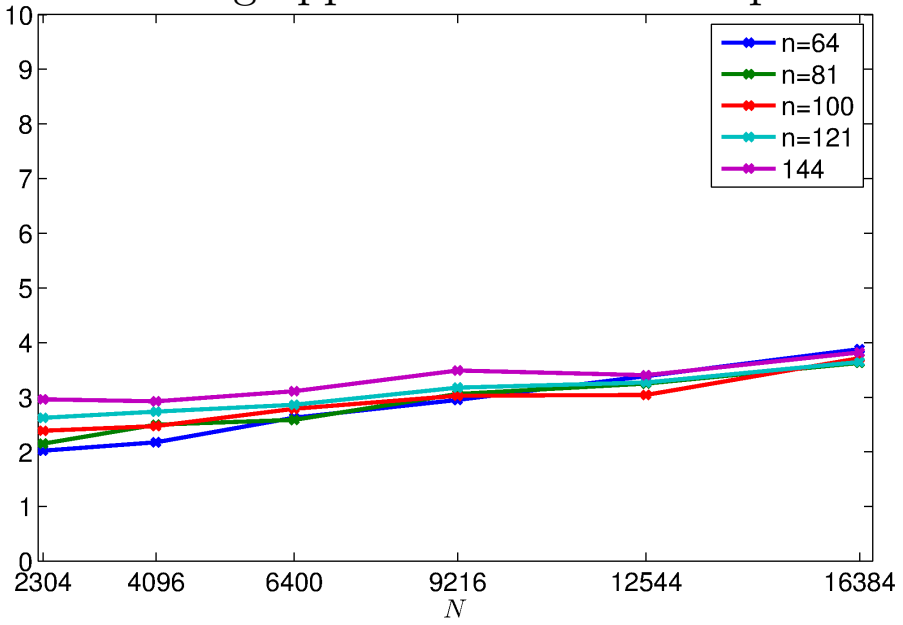
Computational performance

- Wall-clock time (in seconds) vs. N

"Fitting" the data



Evaluating approximant at 27556 points



- Code implemented in MATALB.

Concluding remarks

- Geometric convergence rates appear possible for smooth solutions.
- Computational cost of the method scales favorably.
- Can reconstruct a global stream function directly from samples of the field.

Plans for the future:

- Explore algorithms to generate patches for non-uniform nodes.
- Parallelization of the method.
- Implementation of stable *flat* RBF algorithms for avoiding saturation errors.
- Provide theory on convergence rates.
- Comparison with a "fast algorithm" (i.e. tree-code) version of global divergence-free method. (Any help here would be greatly appreciated).