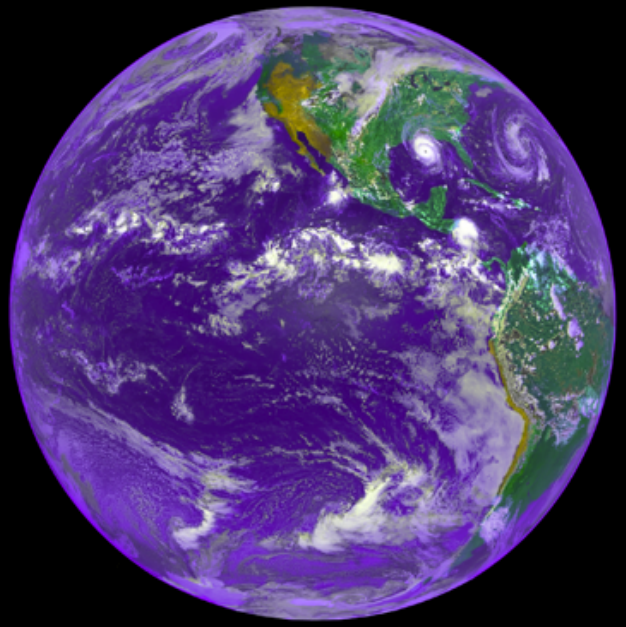
2014 Montestigliano Workshop

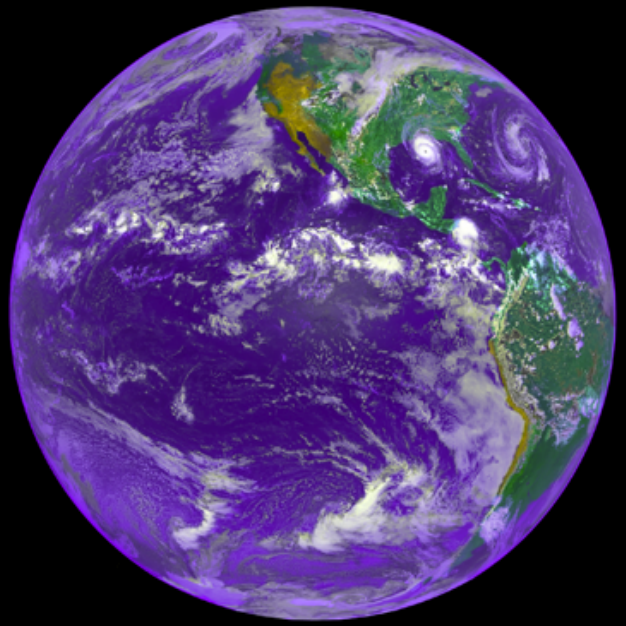


Radial Basis Functions for Scientific Computing

Grady B. Wright
Boise State University

*This work is supported by NSF grants DMS 0934581

2014 Montestigliano Workshop



Part II: Advanced Techniques

Grady B. Wright
Boise State University

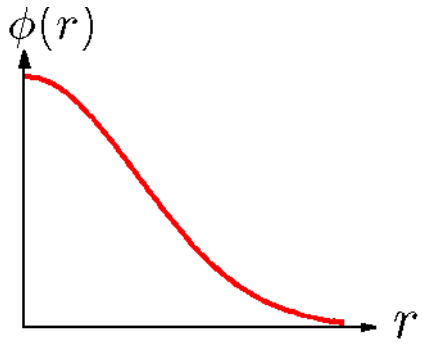
*This work is supported by NSF grants DMS 0934581

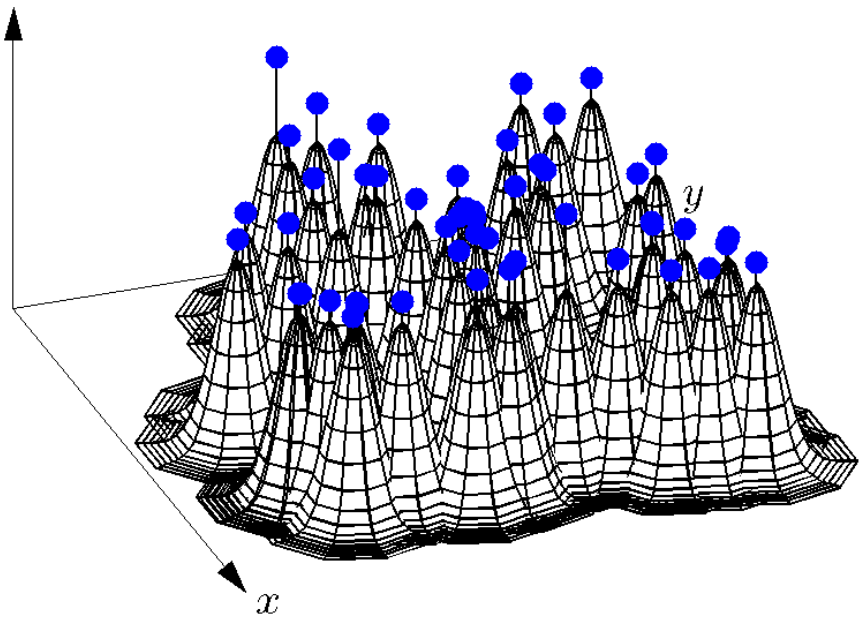
Localized bases for "scale-independent" radial kernels

Review: RBF interpolation

Key idea: linear combination of translates and rotations of a single radial kernel:

$f \qquad X = \{\mathbf{x}_j\}_{j=1}^N \subset \Omega, \quad f\big|_X = \{f_j\}_{j=1}^N$





Basic RBF Interpolant for $\Omega \subseteq \mathbb{R}^2$

$$I_X f(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|)$$

Linear system for determining the interpolation coefficients

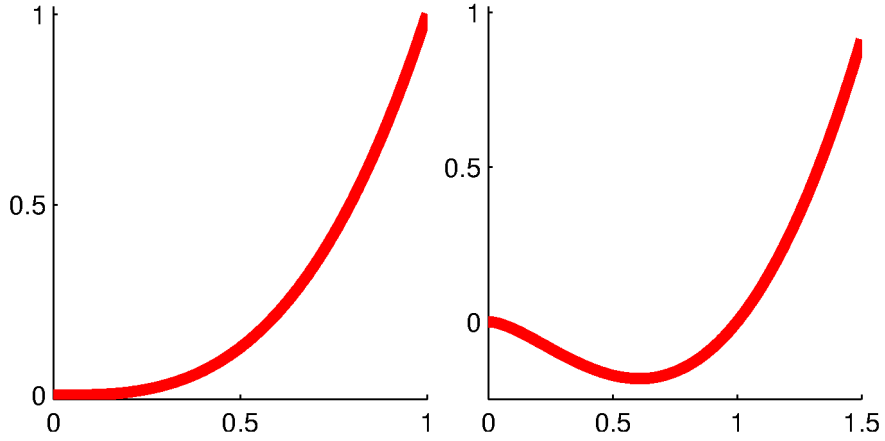
$$\underbrace{\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots \quad \ddots \quad \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_N - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix}}_{A_X} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}}_{\underline{c}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\underline{f}}$$

A_X is guaranteed to be positive definite if ϕ is positive definite.

Localized bases

- "Scale independent" kernels.

Ex: $\phi(r) = r^3$ $\phi(r) = r^2 \log r$



Issues:

- For large N, interpolation matrices are dense.
- Matrices are not nice for iterative methods.

- Ideas for constructing a better basis:
  - Difference functionals: Dyn, Levin & Rippa; Sibson & Stone; Beatson, Levesley, & Mouat.
  - Approximate cardinal functions: Beatson & Powell; Faul & Powell; Beatson, Cherrie, & Mouat.
  - Orthonormal: Schaback & Müller; Schaback & Pazouki; De Marchi & Santin

Lagrange functions on the sphere

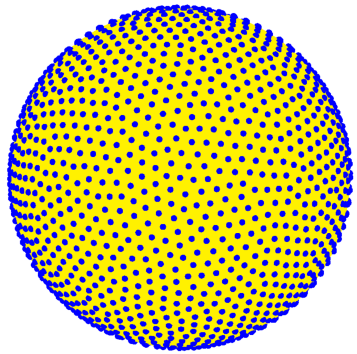
Restrict our attention to $\Omega = \mathbb{S}^2$ and $\phi_\ell(r) = r^{2(\ell-1)}\log(r)$

Standard RBF interpolant:

$$s_X(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi_\ell(\|\mathbf{x}-\mathbf{x}_j\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x}) \Bigg\} \sum_{j=1}^{N} c_j p_k(\mathbf{x}_j) = 0,\ 1 \le k \le \ell^2$$

Lagrange form:

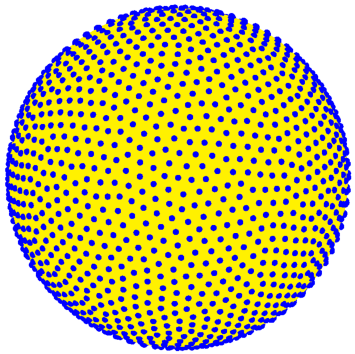
$$s_X(\mathbf{x}) = \sum_{j=1}^{N} L_j(\mathbf{x}) f_j, \quad L_i(\mathbf{x}_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$



Lagrange functions on the sphere

Restrict our attention to $\Omega = \mathbb{S}^2$ and $\phi_\ell(r) = r^{2(\ell-1)}\log(r)$

Standard RBF interpolant:
$$s_X(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi_\ell(\|\mathbf{x}-\mathbf{x}_j\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x}) \Bigg\} \sum_{j=1}^{N} c_j p_k(\mathbf{x}_j) = 0,\ 1 \le k \le \ell^2$$



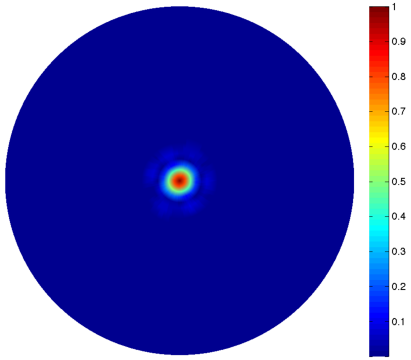
Lagrange form:
$$s_X(\mathbf{x}) = \sum_{j=1}^{N} L_j(\mathbf{x}) f_j, \quad L_i(\mathbf{x}_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Results on the Lagrange functions for quasi-uniform X:
(Hangelbroek, Narcowich, Sun, Ward)

1. Lagrange basis is local (HNW, 2010):
$$|L_j(\mathbf{x})| \le C \exp\left[-\nu \frac{\text{dist}(\mathbf{x}_j,\mathbf{x})}{h_X}\right]$$

2. Lebesgue constant is bounded (HNW, 2010):
$$\mathcal{L}_X := \max_{\mathbf{x}\in\mathbb{S}^2} \sum_{j=1}^{N} |L_j(\mathbf{x})| \le C$$

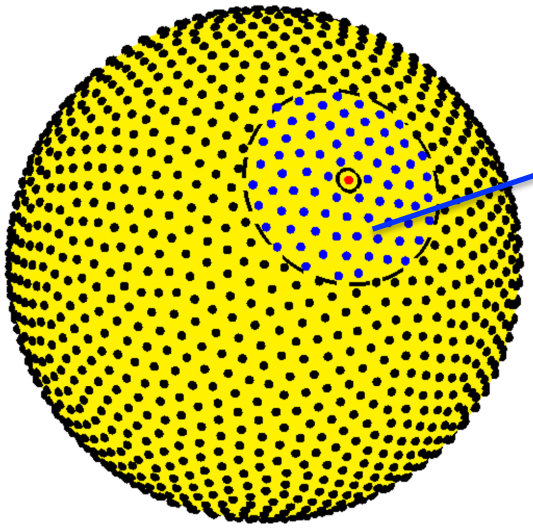
3. Lagrange basis is stable (HNSW, 2011)



$|L_j(\mathbf{x})|$

Local Lagrange functions on the sphere

$X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$



Algorithm: For $i = 1, \ldots, N$

1. Choose $n << N$ nearest neighbors to $\mathbf{x}_i$:

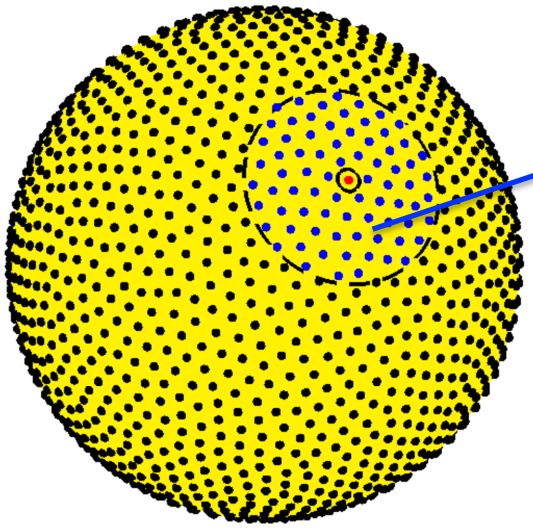
   $X_i = \{\mathbf{x}_j^i\}_{j=1}^n \subset X$

2. Construct the local Lagrange function on X:

$$\widetilde{L}_i(\mathbf{x}) = \sum_{j=1}^{n} c_j^i \phi_\ell(\|\mathbf{x} - \mathbf{x}_j^i\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x})$$

Local Lagrange functions on the sphere

$X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$



Algorithm: For $i = 1, \ldots, N$

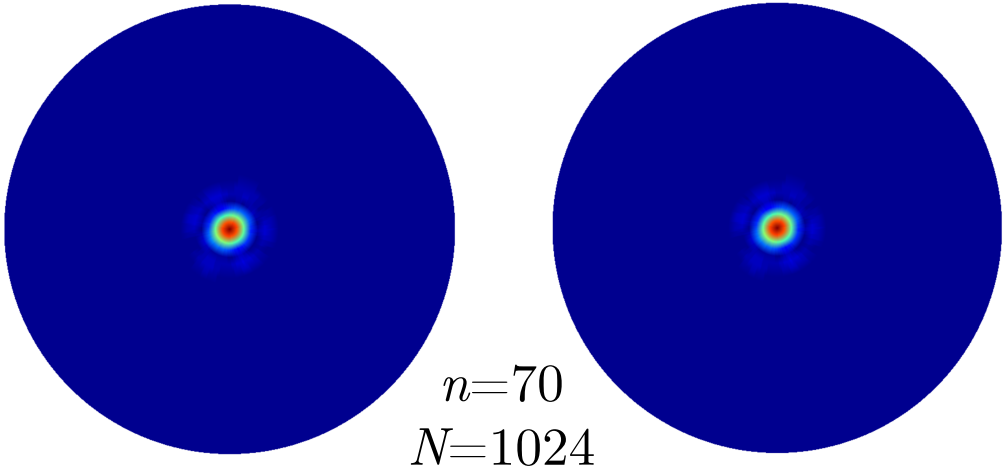
1. Choose $n << N$ nearest neighbors to $\mathbf{x}_i$:

   $X_i = \{\mathbf{x}_j^i\}_{j=1}^n \subset X$

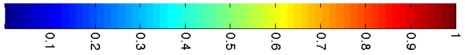
2. Construct the local Lagrange function on X:

$$\widetilde{L}_i(\mathbf{x}) = \sum_{j=1}^{n} c_j^i \phi_\ell(\|\mathbf{x} - \mathbf{x}_j^i\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x})$$

$|L_j(\mathbf{x})|$ $\qquad$ $|\widetilde{L}_j(\mathbf{x})|$

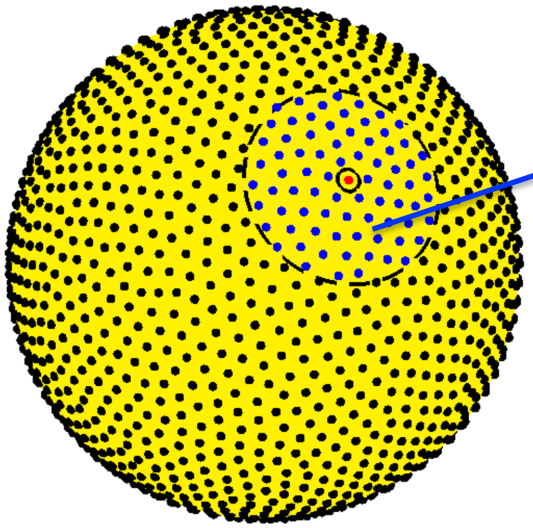


n=70
N=1024



Local Lagrange functions on the sphere

$X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$



Algorithm: For $i = 1, \ldots, N$

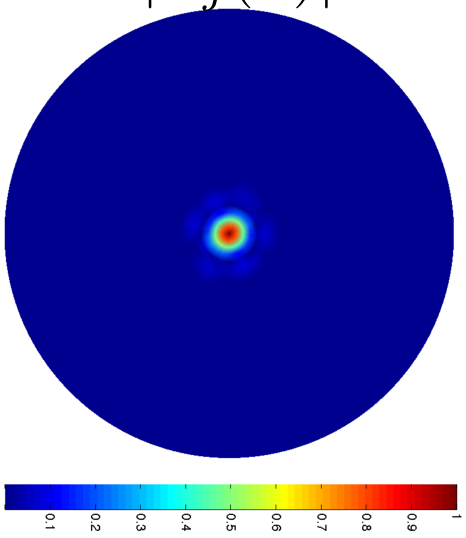
1. Choose $n << N$ nearest neighbors to $\mathbf{x}_i$:

   $X_i = \{\mathbf{x}_j^i\}_{j=1}^n \subset X$

2. Construct the local Lagrange function on X:

$$\widetilde{L}_i(\mathbf{x}) = \sum_{j=1}^{n} c_j^i \phi_\ell(\|\mathbf{x} - \mathbf{x}_j^i\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x})$$

$|\widetilde{L}_j(\mathbf{x})|$



Estimates: (FHNWW, 2013)

If each $\widetilde{L}_j(\mathbf{x})$ is constructed from $n = M(\log N)^2$ neighbors then

$$\|\widetilde{L}_j - L_j\|_\infty \leq C\ h_X^J$$

$$|\widetilde{L}_j(\mathbf{x})| \leq C\big(1 + \mathrm{dist}(\mathbf{x}, \mathbf{x}_j)/h_X\big)^{-J}$$

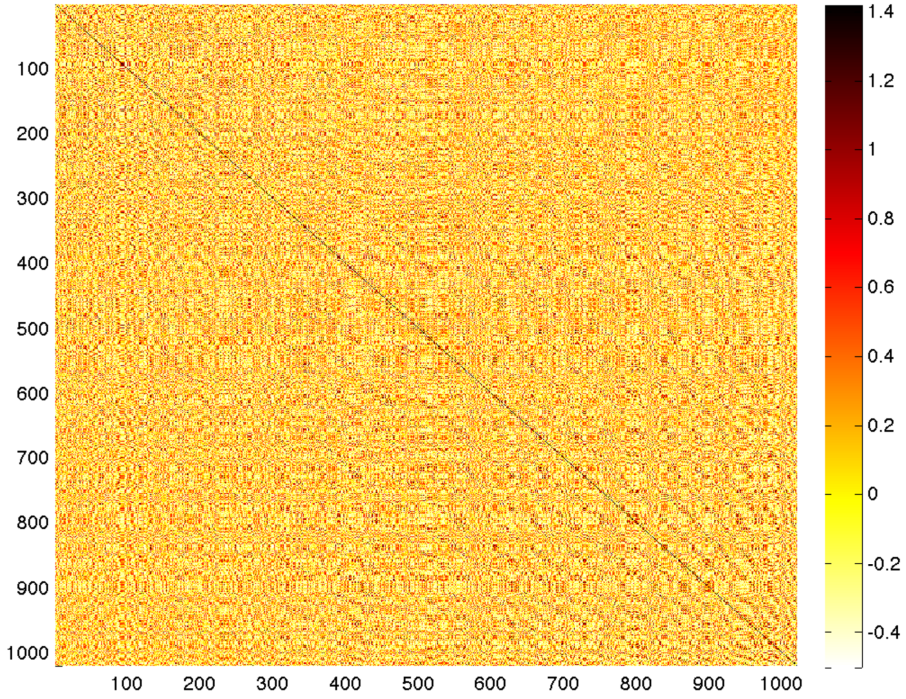
Interpolation matrices

- Example: N=1024, n=70

Standard basis:

$$s_X(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi_2(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{4} b_k p_k(\mathbf{x})$$

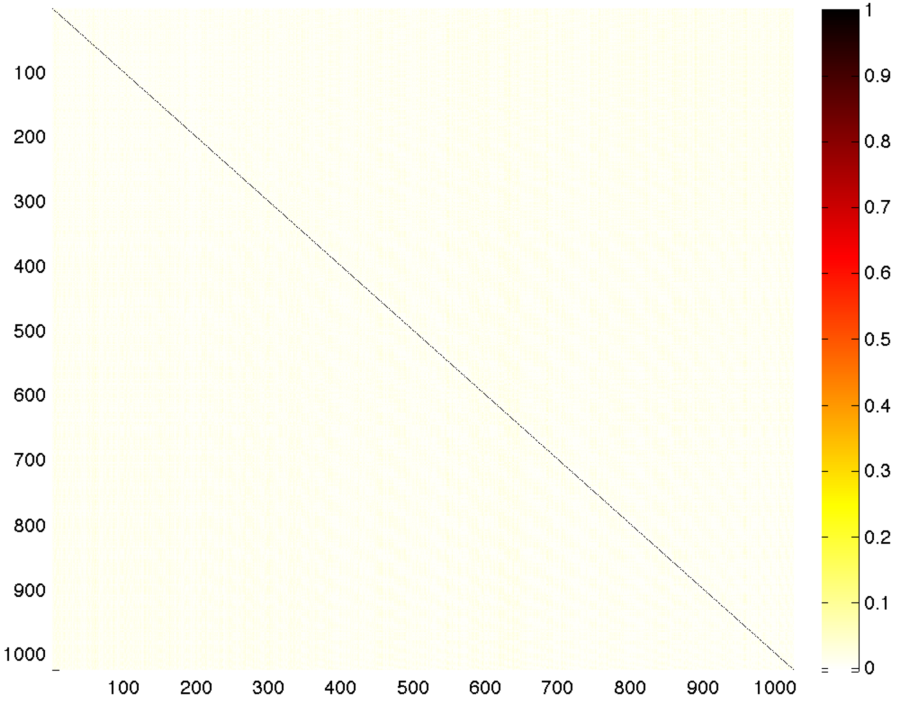
Interpolation matrix



Approximate Lagrange basis:

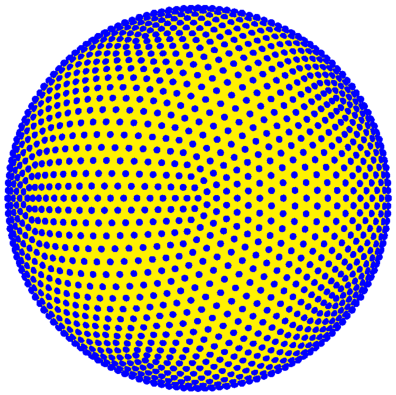
$$s(\mathbf{x}) = \sum_{j=1}^{N} a_j \widetilde{L}_j(\mathbf{x})$$

Interpolation matrix



Solving "preconditioned" systems

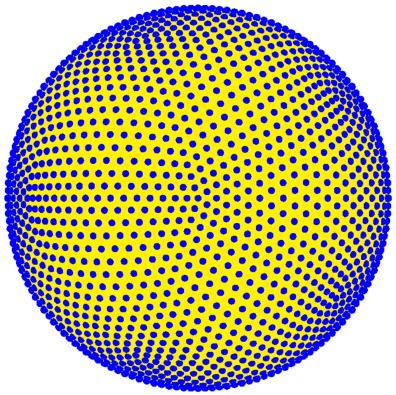
- Numerical experiment: $s(\mathbf{x}) = \sum_{j=1}^{N} a_j \widetilde{L}_j(\mathbf{x})$



  - Target f: random values distributed between $[-1, 1]$.
  - $\widetilde{L}_j$ constructed from $n = 7\lceil(\log_{10} N)^2\rceil$ neighbors
  - Systems solved using GMRES iterative method (Saad & Schultz, 1986)

Solving "preconditioned" systems

- Numerical experiment: $s(\mathbf{x}) = \sum_{j=1}^{N} a_j \widetilde{L}_j(\mathbf{x})$



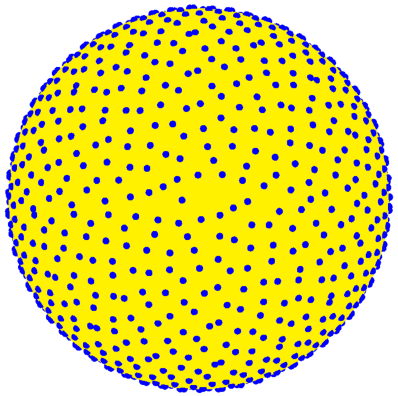
  - Target f: random values distributed between $[-1, 1]$.
  - $\widetilde{L}_j$ constructed from $n = 7\lceil(\log_{10} N)^2\rceil$ neighbors
  - Systems solved using GMRES iterative method (Saad & Schultz, 1986)

			Number GMRES iterations			
			$tol = 10^{-6}$	$tol = 10^{-8}$	$tol = 10^{-10}$	$tol = 10^{-12}$
N	n	ρ_X	Icosahedral nodes			
2562	84	1.650	7	8	9	10
10242	119	1.679	5	7	8	9
23042	140	1.688	6	7	8	9
40962	154	1.693	5	7	7	8
92162	175	1.688	6	8	9	10
163842	196	1.701	5	7	7	8

Note: Each iteration takes $\mathcal{O}(N^2)$ operations, but may be reduced to $\mathcal{O}(N \log N)$ using NFFT (Keiner, Kunis, & Potts, 2006). Project?

Solving "preconditioned" systems

- Numerical experiment: $s(\mathbf{x}) = \sum_{j=1}^{N} a_j \widetilde{L}_j(\mathbf{x})$



  - Target f: random values distributed between $[-1, 1]$.
  - $\widetilde{L}_j$ constructed from $n = 7\lceil(\log_{10} N)^2\rceil$ neighbors
  - Systems solved using GMRES iterative method (Saad & Schultz, 1986)

			Number GMRES iterations			
			$tol = 10^{-6}$	$tol = 10^{-8}$	$tol = 10^{-10}$	$tol = 10^{-12}$
N	n	ρ_X	Hammersley nodes			
4000	91	24.56	8	10	11	12
8000	112	34.74	8	9	11	12
16000	126	49.13	7	9	10	11
32000	147	69.48	7	8	10	11
64000	168	98.26	7	9	10	12

Note: Also appears to work well for less uniform nodes, but no theory (yet!).

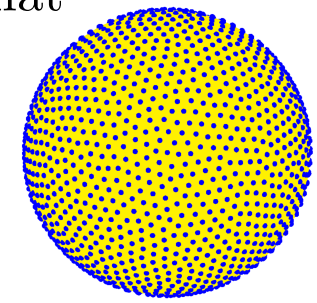
Concluding remarks on local Lagrange basis

- Local Lagrange basis appears to provide a good bases for certain kernel spaces on $\mathbb{S}^2$.
  - Can be computed using $\mathcal{O}(N(\log N)^2)$ nearest neighbors.
  - Computation is embarrassingly parallel.
  - Works very well as a preconditioner for global interpolation problem.
- Research problems:
  - Combine with fast *evaluation* algorithms to compute interpolation coefficients for the global interpolant in total of $\mathcal{O}(N \log(N)^2)$.
  - Use local Lagrange basis as a quasi-interpolant.
  - Use local Lagrange basis as trial functions in a Galerkin formulation of PDEs on the sphere. (Need quadrature formulas for the sphere.)

Using RBF interpolation for developing quadrature formulas on the sphere

Quadrature on the sphere

- Problem: Given $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$, find weights $\{w_j\}_{j=1}^N$ such that



$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \sum_{j=1}^{N} w_j f(\mathbf{x}_j) =: Q(f), \quad f \in C(\mathbb{S}^2)$$

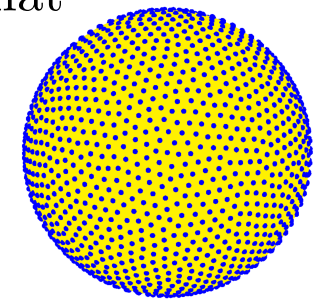
- One solution: Find the weights from the kernel interpolant of f on X:

$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \int_{\mathbb{S}^2} s_X(\mathbf{x}) d\mu(\mathbf{x})$$

- So what are the weights?

Quadrature on the sphere

- Problem: Given $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$, find weights $\{w_j\}_{j=1}^N$ such that



$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \sum_{j=1}^N w_j f(\mathbf{x}_j) =: Q(f), \quad f \in C(\mathbb{S}^2)$$

- One solution: Find the weights from the kernel interpolant of f on X:

$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \int_{\mathbb{S}^2} s_X(\mathbf{x}) d\mu(\mathbf{x})$$

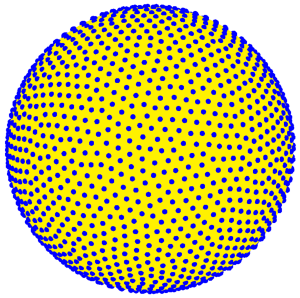
- So what are the weights?

$$\text{Lagrange form}: \int_{\mathbb{S}^2} s_X(\mathbf{x}) d\mu(\mathbf{x}) = \sum_{j=1}^N \underbrace{\left(\int_{\mathbb{S}^2} L_j(\mathbf{x}) d\mu(\mathbf{x}) \right)}_{w_j} f_j$$

- How can this be made computationally tractable for large N?

Quadrature on the sphere

- A neat results for radial (zonal) kernels:



$$\int_{\mathbb{S}^2} s_X(\mathbf{x}) d\mu(\mathbf{x}) = \sum_{j=1}^{N} \left(\int_{\mathbb{S}^2} \phi(\|\mathbf{x} - \mathbf{x}_j\|) d\mu(\mathbf{x}) \right) c_j$$

$$= \left(\int_{\mathbb{S}^2} \phi(\|\mathbf{x} - \mathbf{x}_1\|) d\mu(\mathbf{x}) \right) \sum_{j=1}^{N} c_j = J_0 \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}$$

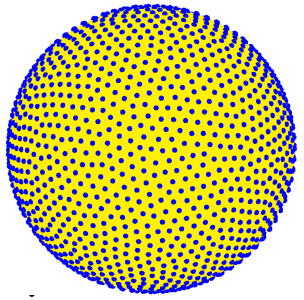
$$= \underbrace{J_0 \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} & & & \\ & \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) & & \\ & & & \end{bmatrix}^{-1}}_{\begin{bmatrix} w_1 & w_2 & \cdots & w_N \end{bmatrix}} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}$$

- So the weights can be computed from solving one linear system.

Quadrature on the sphere

- Problem: Given $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$, find weights $\{w_j\}_{j=1}^N$ such that

$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \sum_{j=1}^N w_j f(\mathbf{x}_j) =: Q(f), \quad f \in C(\mathbb{S}^2)$$



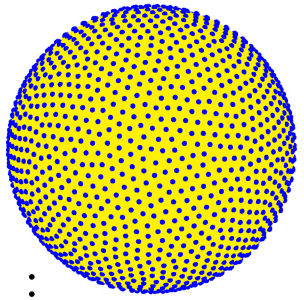
- One solution: Find the weights from the kernel interpolant of $f\big|_X$.

$$\begin{bmatrix} & \\ & \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) \\ & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} J_0 \\ \vdots \\ J_0 \end{bmatrix}$$

Quadrature on the sphere

- Problem: Given $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$, find weights $\{w_j\}_{j=1}^N$ such that

$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \sum_{j=1}^N w_j f(\mathbf{x}_j) =: Q(f), \quad f \in C(\mathbb{S}^2)$$



- One solution: Find the weights from the kernel interpolant of $f\big|_X$:

$$\begin{bmatrix} & \\ & \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) \\ & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} J_0 \\ \vdots \\ J_0 \end{bmatrix}$$

- Note that this idea can be extended to CPD kernels as well (See problem 7):

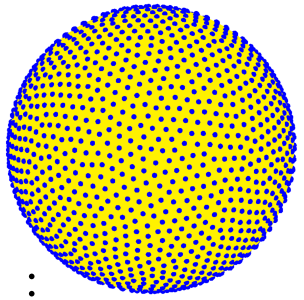
$$s_X(\mathbf{x}) = \sum_{j=1}^N c_j \phi_\ell(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x}), \quad \phi_\ell(r) = r^{2(\ell-1)} \log(r)$$

Error: (FHNWW2014)
$$\left| \int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) - \sum_{j=1}^N w_j f_j \right| \leq \begin{cases} h_X^r \|f\|_{C^r(\mathbb{S}^2)} & 0 < r \leq 2\ell \\ h_X^r \|f\|_{H^r(\mathbb{S}^2)} & 1 < r \leq \ell \end{cases}$$

Quadrature on the sphere

- Problem: Given $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$, find weights $\{w_j\}_{j=1}^N$ such that

$$\int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) \approx \sum_{j=1}^N w_j f(\mathbf{x}_j) =: Q(f), \quad f \in C(\mathbb{S}^2)$$



- One solution: Find the weights from the kernel interpolant of $f\big|_X$:

$$\begin{bmatrix} & \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} J_0 \\ \vdots \\ J_0 \end{bmatrix}$$

- Note that this idea can be extended to CPD kernels as well (See problem 7):

$$s_X(\mathbf{x}) = \sum_{j=1}^N c_j \phi_\ell(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x}), \quad \phi_\ell(r) = r^{2(\ell-1)} \log(r)$$

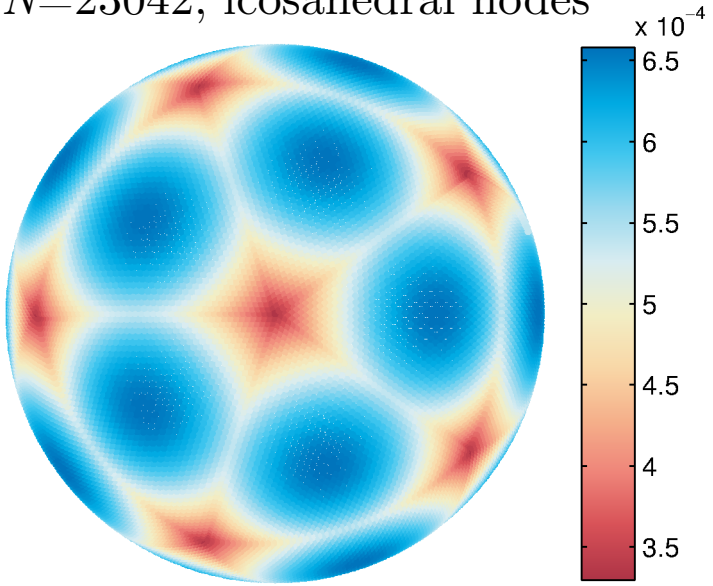
- How ELSE can this be made computationally tractable for large N

Local Lagrange basis!

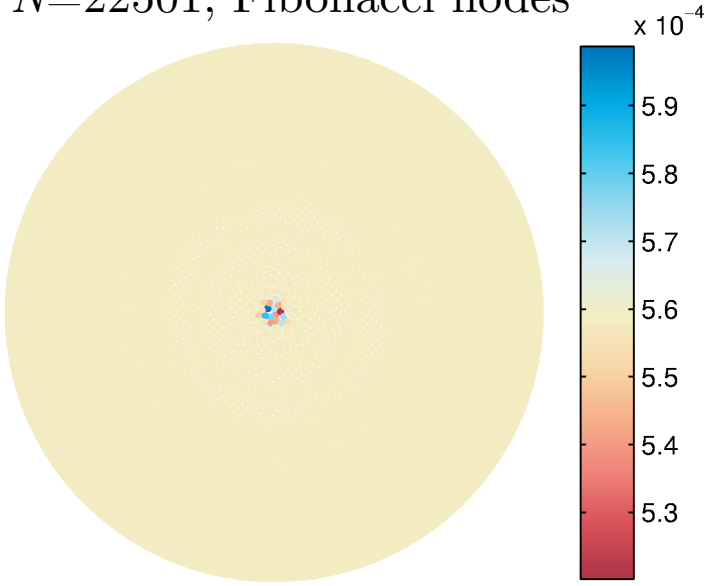
Example of quadrature weights

- Quadrature weights computed using
$$\phi_2(r) = r^2 \log(r)$$
- Computations done using local Lagrange basis as a preconditioner

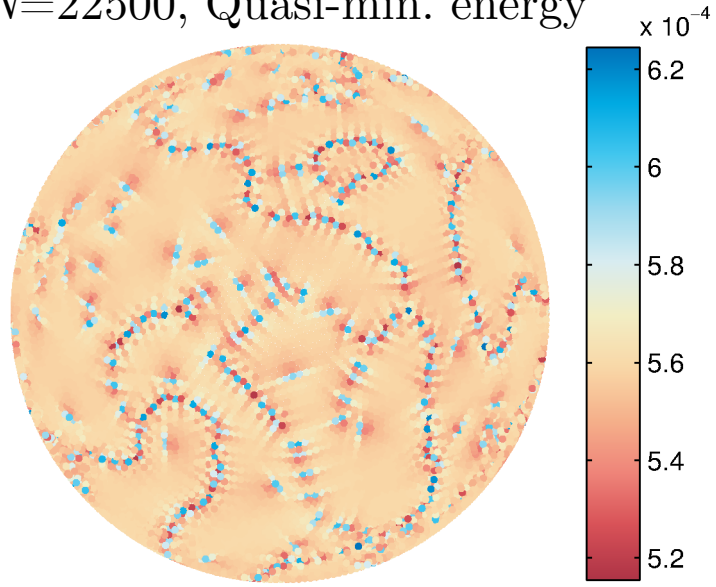
N=23042, icosahedral nodes



N=22501, Fibonacci nodes



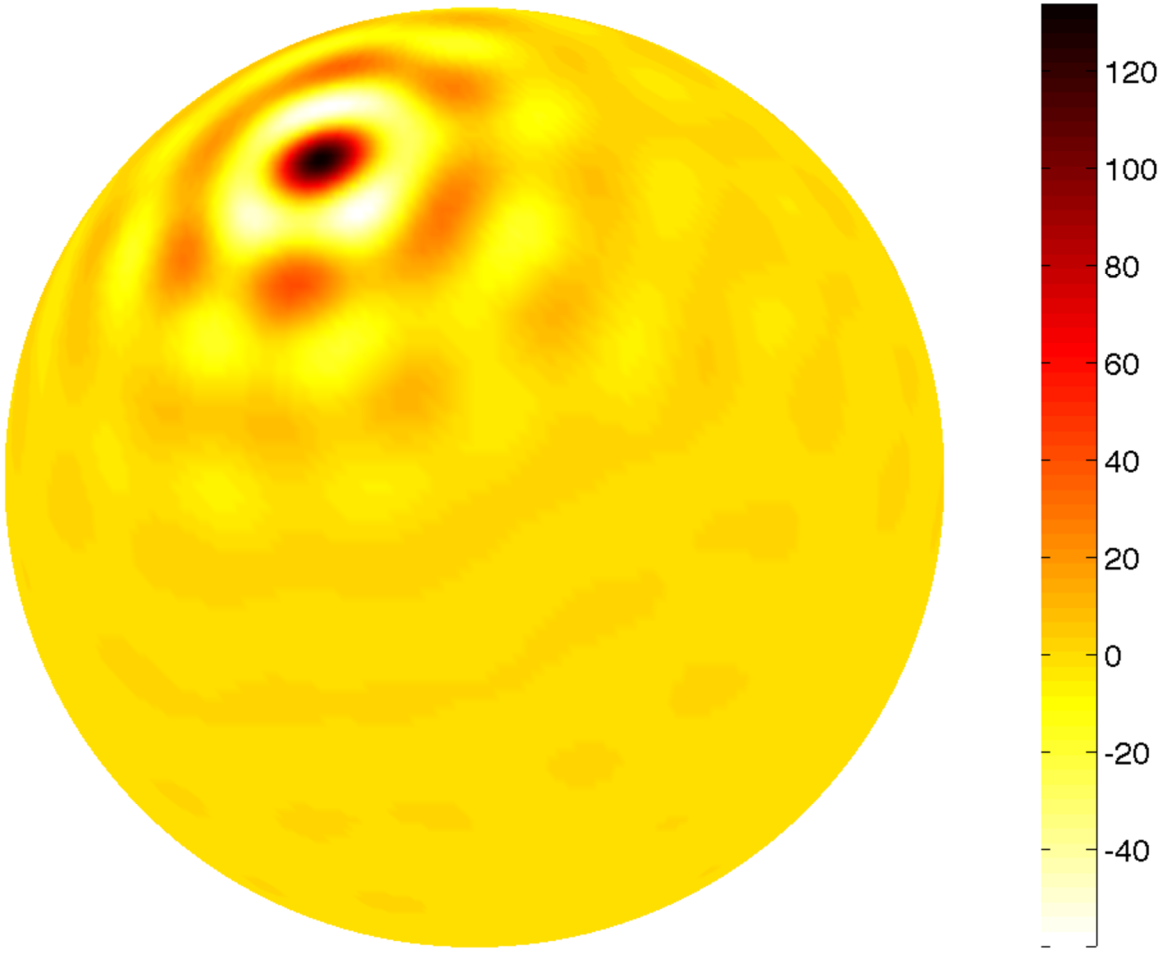
N=22500, Quasi-min. energy



Borodachov, Hardin, Saff (2014)

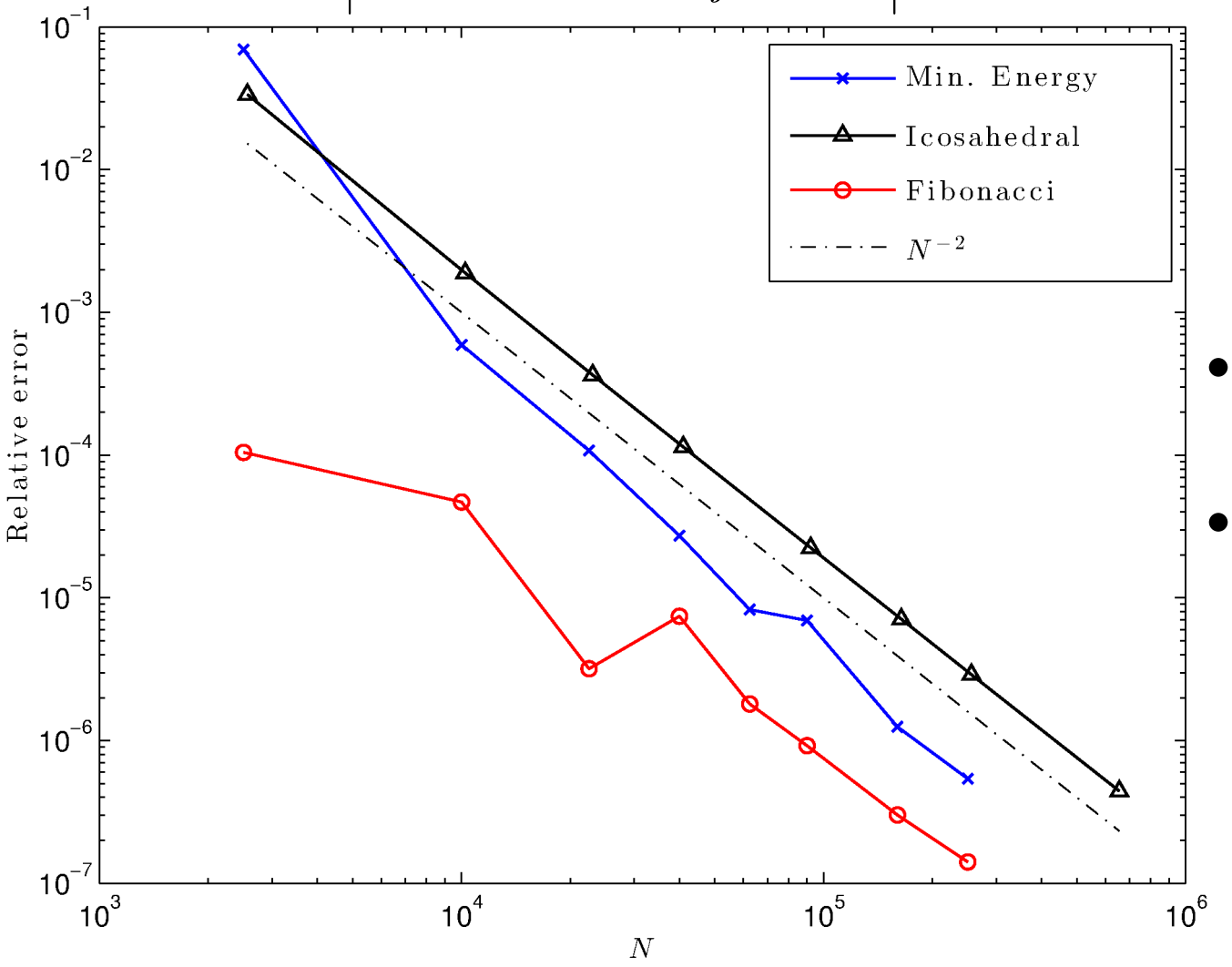
Numerical example

Smooth target function



Numerical example

$$\left| \int_{\mathbb{S}^2} f(\mathbf{x}) d\mu(\mathbf{x}) - \sum_{j=1}^{N} w_j f_j \right|$$



- Weights computed up to $N = 650000$
- Convergence is $\mathcal{O}(h_X^4)$.

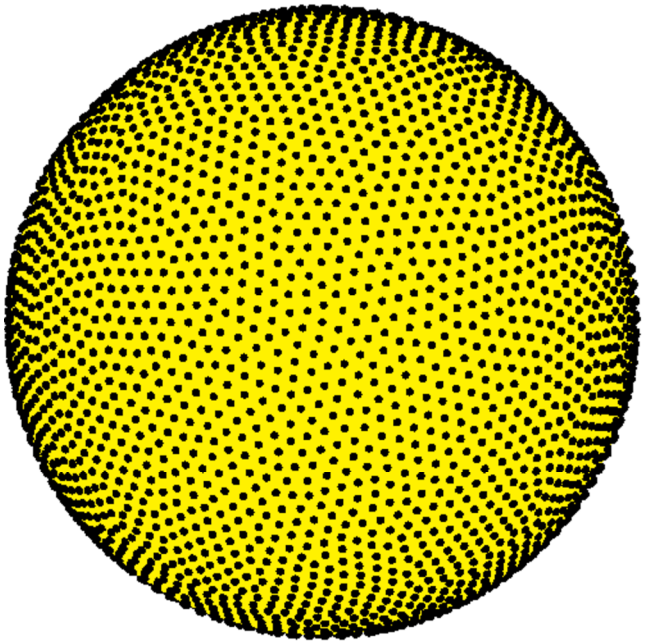
Remarks on RBF-based quadrature

- Quadrature weights can be computed by solving one linear system.
- For large N, these weights can be computed in $O(N^2)$ operations for the thin plate spline using the local Lagrange functions as a preconditioner.
- Research problems:
  - Develop a fast *evaluation* algorithm to compute weights in $\mathcal{O}(N \log(N)^2)$.
  - Simply use the local Lagrange functions to compute the quadrature weights (using a a quasi-interpolant instead of an interpolant).
  - Use the quadrature weights for computing the integrals associated with a Galerkin formulation of some PDEs on the sphere. (Use local Lagrange functions as the trial functions.)
  - Develop local Lagrange functions and quadrature weights for more general ellipsoids.

Combining RBFs and the Partition-of-Unity method for interpolation (RBF-PUM)

RBFs and partition-of-unity on the sphere

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



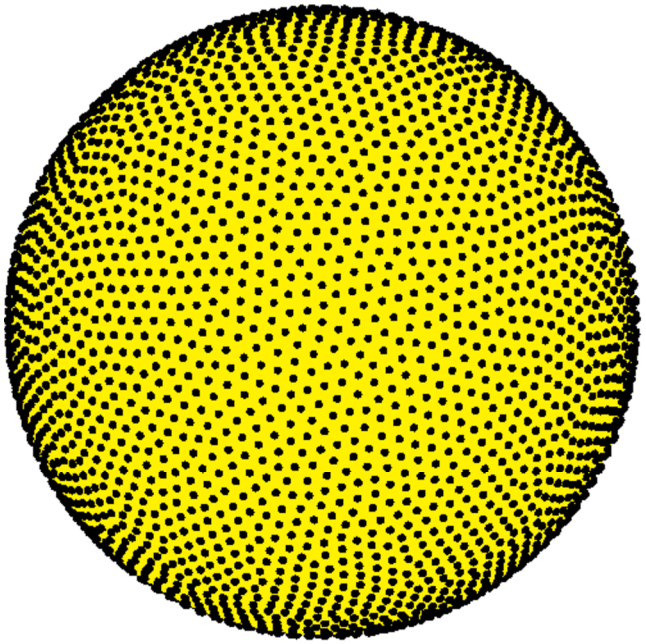
Key references:

- I. Babuška & J.M. Melenk. The partition of unity method. *IJNME* (1998).

- R. Cavoretto & A. DeRossi, Fast and accurate interpolation of large scattered data sets on the sphere. *J. Comput. Appl. Math.* (2010)
  - First application of PUM to RBF interpolation on the sphere

RBFs and partition-of-unity 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:

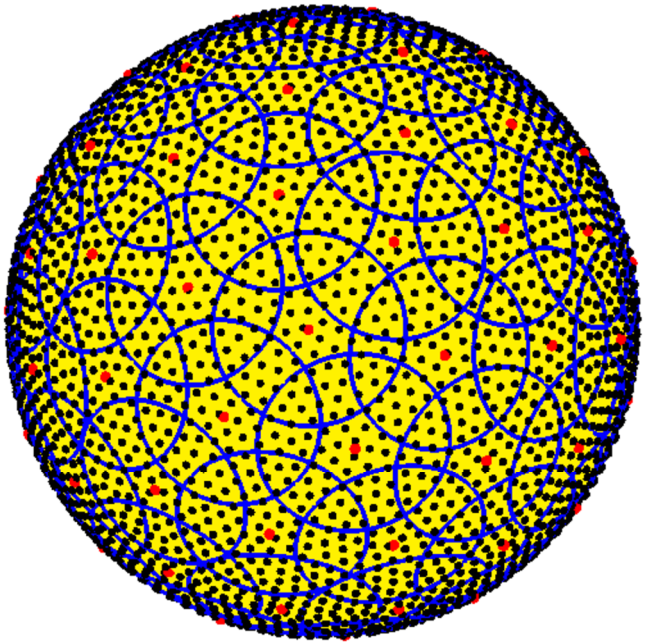
1. Generate a set of overlapping patches (spherical caps) $\Omega = \{\Omega_k\}_{k=1}^{M}$ with the properties:

   (a) Each patch contains roughly n nodes of X.

   (b) $\bigcup_{k=1}^{M} \Omega_k = \mathbb{S}^2$.

RBFs and partition-of-unity 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:

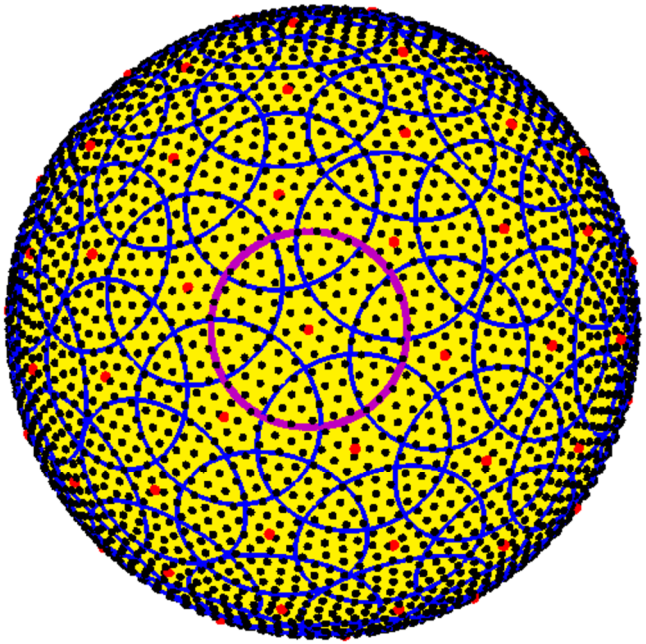
1. Generate a set of overlapping patches (spherical caps) $\Omega = \{\Omega_k\}_{k=1}^{M}$ with the properties:

   (a) Each patch contains roughly n nodes of X.

   (b) $\displaystyle\bigcup_{k=1}^{M} \Omega_k = \mathbb{S}^2$.

RBFs and partition-of-unity 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

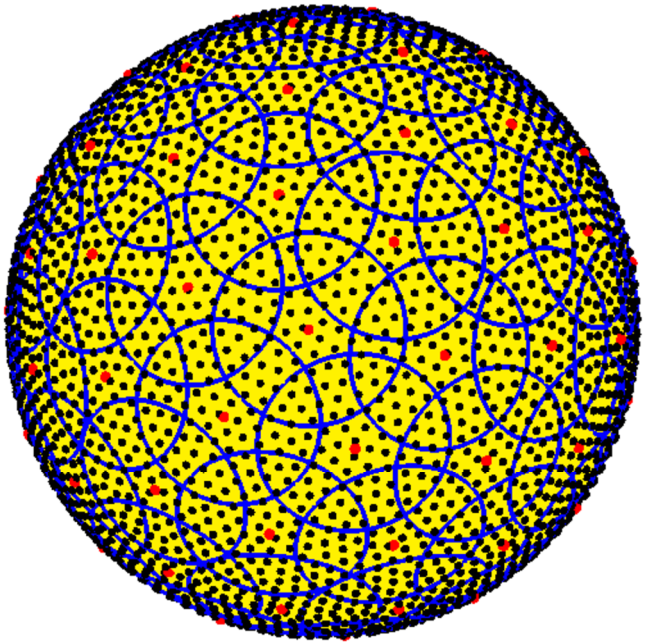
Key Steps:

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

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

RBFs and partition-of-unity 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

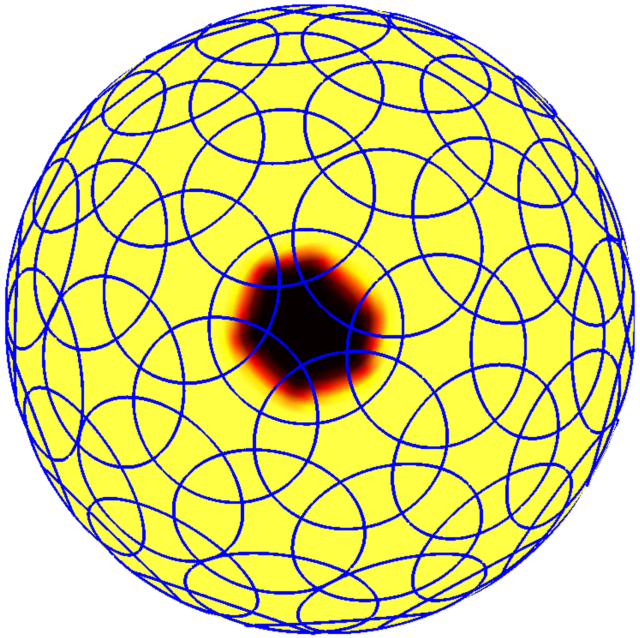
Key Steps:

3. Define weight functions $w_k : \mathbb{S}^2 \to \mathbb{R}$, $k = 1, \ldots, M$, such that:
   (a) Each w_k is compactly supported over Ω_k.
   (b) The set of all w_k form a partition-of-unity over Ω:

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

RBFs and partition-of-unity 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

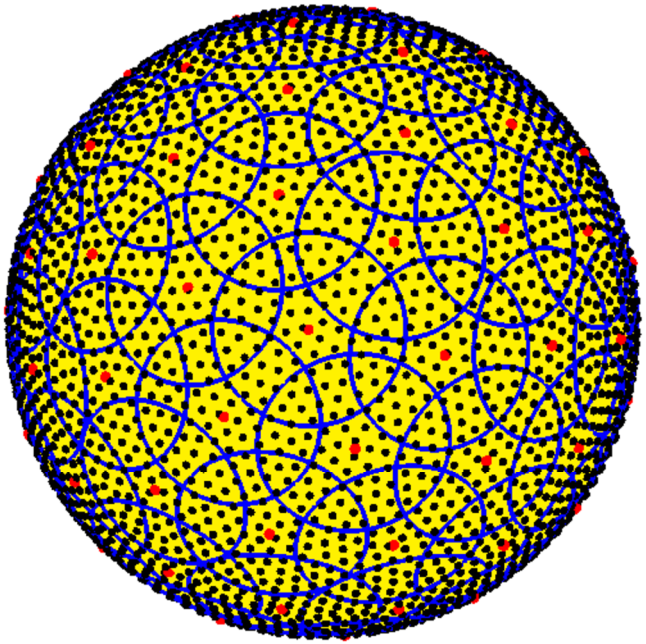
Key Steps:

3. Define weight functions $w_k : \mathbb{S}^2 \to \mathbb{R}$, $k = 1, \ldots, M$, such that:
   (a) Each w_k is compactly supported over Ω_k.
   (b) The set of all w_k form a partition-of-unity over Ω:

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

RBFs and partition-of-unity 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:

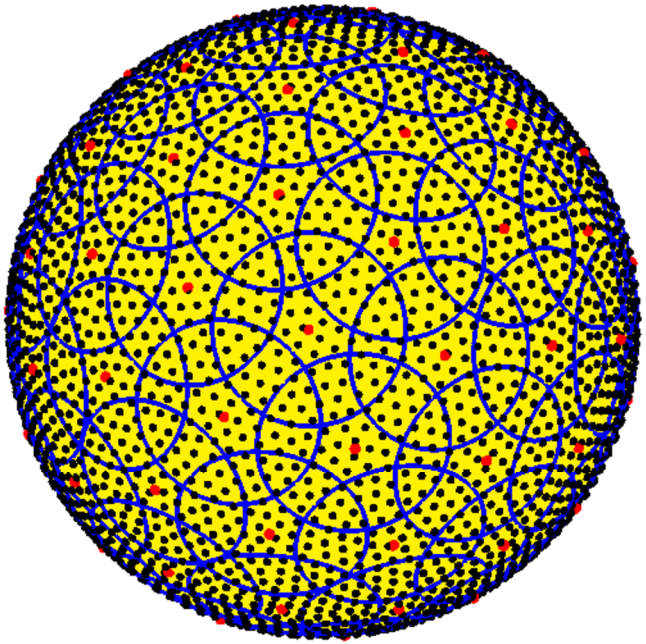
$$w_k(\mathbf{x}) = \frac{\psi_k(\mathbf{x})}{\sum_{i=1}^{M} \psi_i(\mathbf{x})} \longrightarrow \psi_k(\mathbf{x}) = \psi\left(\frac{\|\mathbf{x} - \boldsymbol{\xi}_k\|}{\rho_k}\right)$$

(Shepard weight function)

ρ_k = radius of patch Ω_k
ψ has compact support over $[0, 1]$
Ex: ψ = cubic B-spline

RBFs and partition-of-unity 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

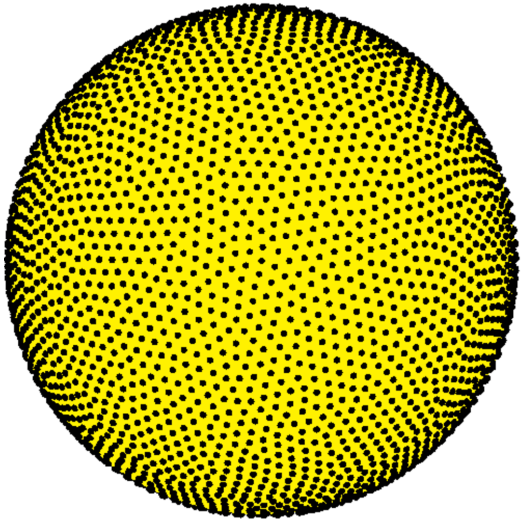
$$s_X(\mathbf{x}) = \sum_{k=1}^{M} w_k(\mathbf{x}) s_k(\mathbf{x})$$

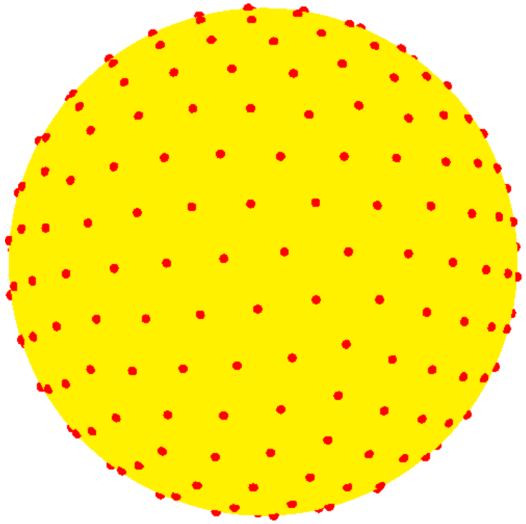
Example for choosing the nodes and patches

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



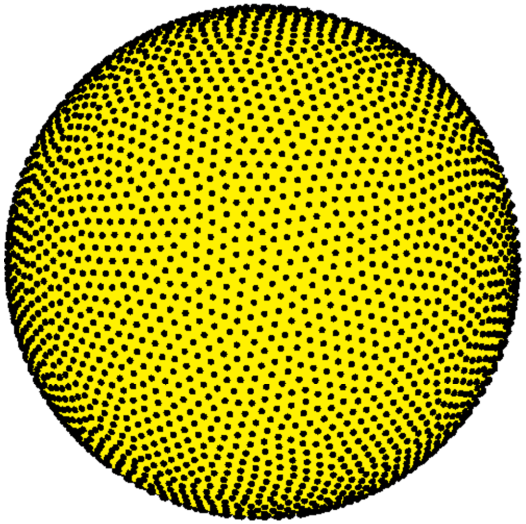


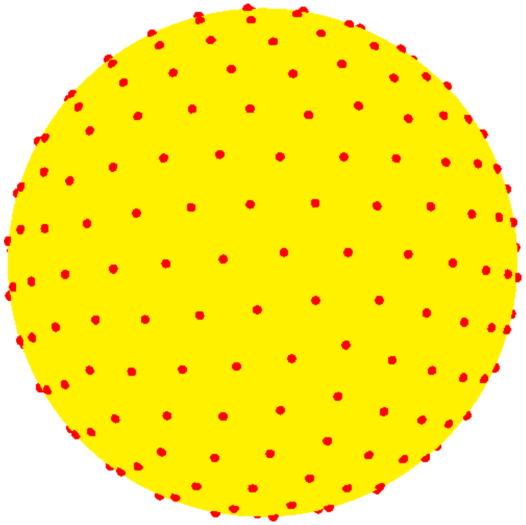
Patch centers

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:

$$\frac{4\pi}{N} \approx \frac{\pi\rho^2}{n} \quad \Longrightarrow \quad \rho \approx 2\sqrt{\frac{n}{N}}$$

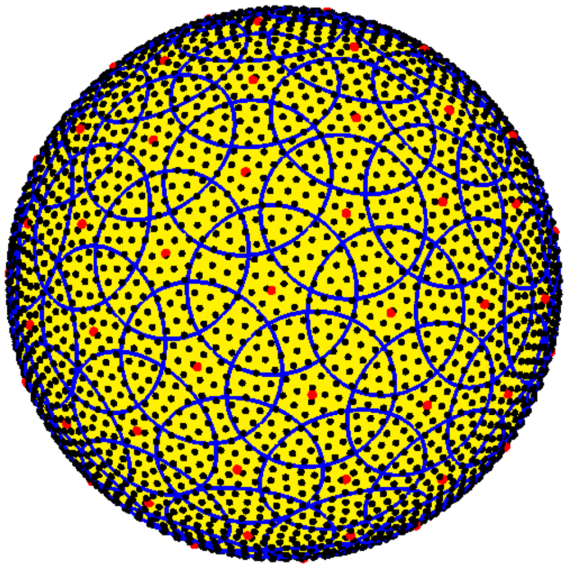
- The overlap parameter q determines the **average number of patches a node belongs to**, and satisfies the relationship:

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

Choosing the nodes and patches

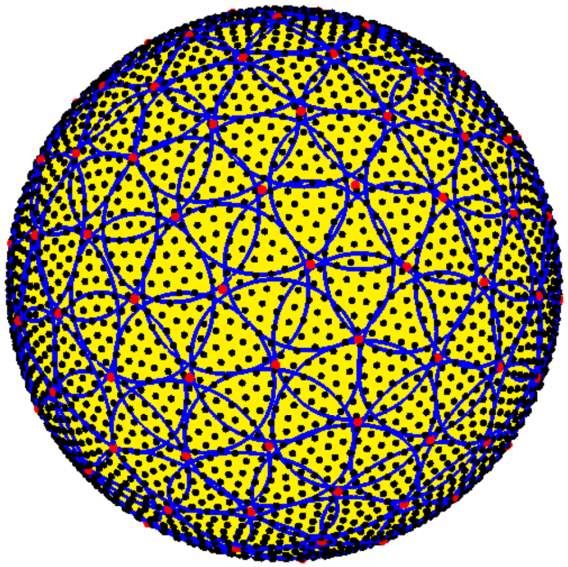
- Illustration of the patches for N=4096, n=100, and different q:

q=2



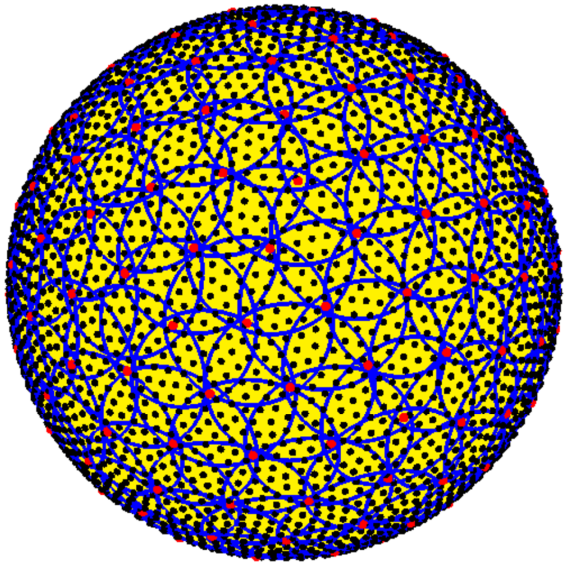
M=82

q=3



M=123

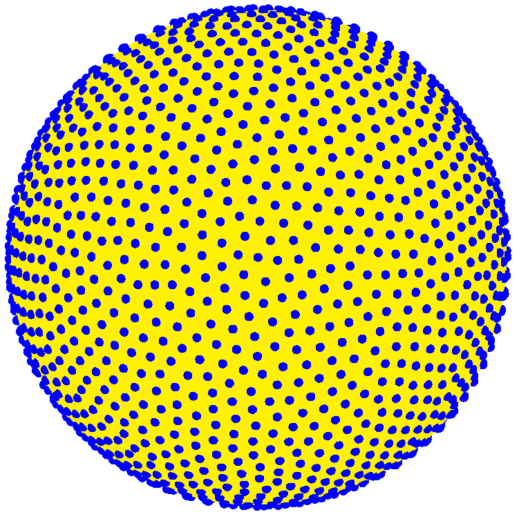
q=4



M=184

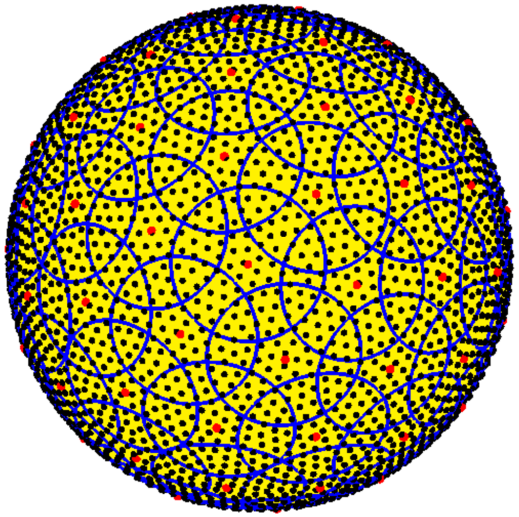
Comparison to global RBF method

Global RBFs



N=total nodes

RBF-PUM



N=total nodes

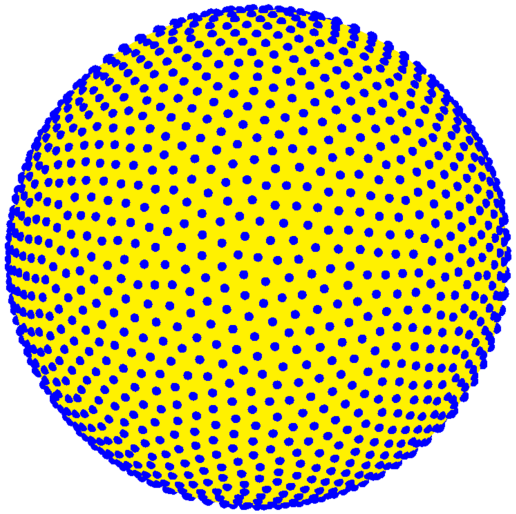
M=total patches
n=nodes per patch
q=avg. # patches
a node belongs to

Computational cost

Collocation	**Global RBF**	**RBF-PUM***
Construction:	$O(N^3)$	$O(n^3 M) + O(N \log N) =$ $O(n^2 q N) + O(N \log N)$
Evaluation at K nodes:	$O(KN)$	$O(q\,n\,K)$

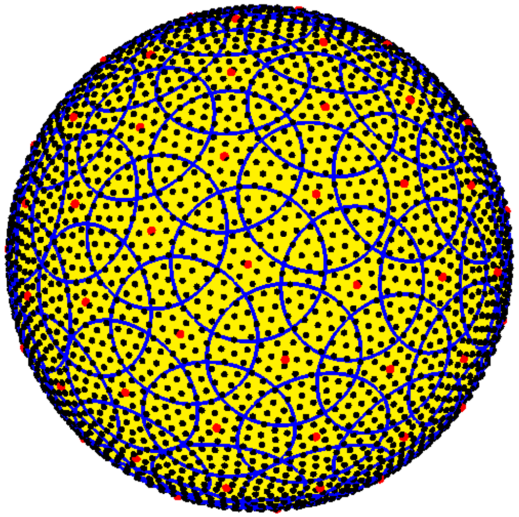
Comparison to global RBF method

Global RBFs



N=total nodes

RBF-PUM



N=total nodes

M=total patches
n=nodes per patch
q=avg. # patches
a node belongs to

Accuracy Comparison:

- Theory for RBF-PUM interpolation in $\mathbb{R}^d$ says same convergence orders should be expected as global RBFs (although constants involved are larger and depend strongly on the partitions).
- No theory on for $\mathbb{S}^2$, but should expect similar results.

Remarks on RBF-PUM

- RBF-PUM interpolant has near-optimal computational scaling.
- Error estimates for the sphere still need to be worked out.
- Research problems:
  - Develop RBF-PUM for non-uniform nodes on the sphere.
  - Develop a multilevel (or multiscale) framework for fitting/filtering data using RBF-PUM.
  - Develop a PBF-PUM to solve PDEs, such as advection-reaction-diffusion on the sphere. Use collocation or Galerkin method.
  - Use RBF-PUM for geometric modeling (see problems 3 and 9).

Computing derivatives and solving PDEs with RBFs

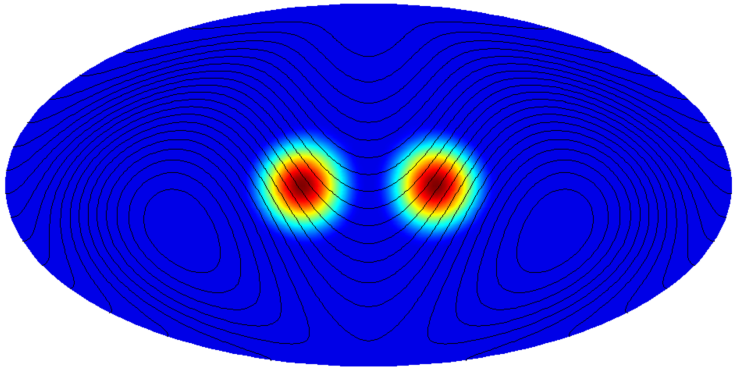
(Surface) Div, Grad, Curl, and all that

Part 2

	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:	$\mathbf{u}_s = \nabla_s\, g = \frac{1}{\cos\theta}\frac{\partial g}{\partial \lambda}\hat{\mathbf{i}} + \frac{\partial g}{\partial \theta}\hat{\mathbf{j}}$	$\mathbf{u}_c = P(\nabla_c\, g) = P\left(\frac{\partial g}{\partial x}\hat{\mathbf{i}} + \frac{\partial g}{\partial y}\hat{\mathbf{j}} + \frac{\partial g}{\partial z}\hat{\mathbf{k}}\right)$
Surface divergence of $\mathbf{u}$:	$\nabla_s \cdot \mathbf{u}_s = \frac{1}{\cos\theta}\frac{\partial u_s}{\partial \lambda} + \frac{\partial v_s}{\partial \theta}$	$(\nabla_c) \cdot \mathbf{u}_c = \nabla_c \cdot \mathbf{u}_c - \mathbf{x} \cdot \nabla(\mathbf{u}_c \cdot \mathbf{x})$
Curl of a scalar f:	$\mathbf{u}_s = \hat{\mathbf{k}} \times (\nabla_s f) = -\frac{\partial f}{\partial \theta}\hat{\mathbf{i}} + \frac{1}{\cos\theta}\frac{\partial f}{\partial \lambda}\hat{\mathbf{j}}$	$\mathbf{u}_c = \mathbf{x} \times (P\nabla_c f) = QP(\nabla_c f) = Q(\nabla_c f)$
Surface curl of a vector $\mathbf{u}$:	$\hat{\mathbf{k}} \cdot (\nabla_s \times \mathbf{u}_s) = -\nabla_s \cdot (\hat{\mathbf{k}} \times \mathbf{u}_s)$	$\mathbf{x} \cdot ((P\nabla_c) \times \mathbf{u}_c) = -\nabla_c \cdot (Q\mathbf{u}_c)$

Here: $P = I - \mathbf{x}\mathbf{x}^T = \begin{bmatrix} 1-x^2 & -xy & -xz \\ -xy & 1-y^2 & -yz \\ -xz & -yz & 1-z^2 \end{bmatrix}$ $Q = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$

Ex: Transport equation on the sphere



- Transport equation for a scalar valued quantity h on the surface of the unit sphere in an incompressible velocity field $\mathbf{u}_c$.
- The governing PDE can be written in Cartesian coordinates as:

$$h_t + \mathbf{u}_c \cdot (P\nabla_c h) = 0$$

P projects arbitrary three-dimensional vectors onto a plane tangent to the unit sphere at $\mathbf{x}$.

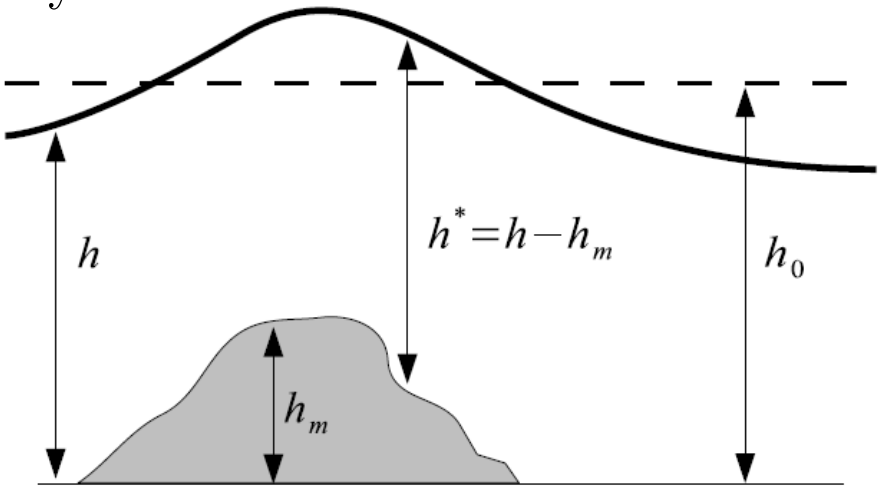
- Surface gradient operator:

$$P\nabla_c = (\mathbf{I} - \mathbf{x}\mathbf{x}^T)\nabla_c = \begin{bmatrix} (1-x^2) & -xy & -xz \\ -xy & (1-y^2) & -yz \\ -xz & -yz & (1-z^2) \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x \cdot \nabla \\ \mathbf{p}_y \cdot \nabla \\ \mathbf{p}_z \cdot \nabla \end{bmatrix}$$

No coordinate singularities!

Ex: shallow water equations on a rotating sphere

- Model for the nonlinear dynamics of a shallow, hydrostatic, homogeneous, and inviscid fluid layer.



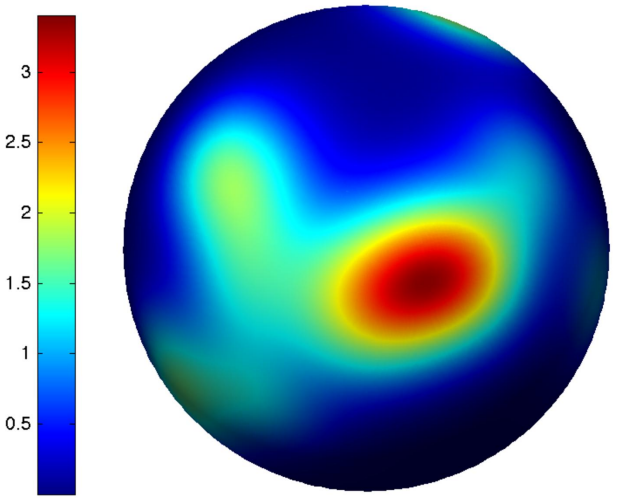
- Idealized test-bed for horizontal dynamics of all 3-D global climate models.

Equations	Momentum	Transport
Spherical coordinates	$\frac{\partial \mathbf{u}_s}{\partial t} + \mathbf{u}_s \cdot \nabla_s \mathbf{u}_s + f\hat{\mathbf{k}} \times \mathbf{u}_s + g\nabla_s h = 0$	$\frac{\partial h^*}{\partial t} + \nabla_s \cdot (h^* \mathbf{u}_s) = 0$
	Singularity at poles!	
Cartesian coordinates	$\frac{\partial \mathbf{u}_c}{\partial t} + P \begin{bmatrix} (\mathbf{u}_c \cdot P\nabla_c)u_c + f(\mathbf{x} \times \mathbf{u}_c) \cdot \hat{\mathbf{i}} + g(P\hat{\mathbf{i}} \cdot \nabla_c)h \\ (\mathbf{u}_c \cdot P\nabla_c)v_c + f(\mathbf{x} \times \mathbf{u}_c) \cdot \hat{\mathbf{j}} + g(P\hat{\mathbf{j}} \cdot \nabla_c)h \\ (\mathbf{u}_c \cdot P\nabla_c)w_c + f(\mathbf{x} \times \mathbf{u}_c) \cdot \hat{\mathbf{k}} + g(P\hat{\mathbf{k}} \cdot \nabla_c)h \end{bmatrix} = 0$	$\frac{\partial h^*}{\partial t} + (P\nabla_c) \cdot (h^* \mathbf{u}_c) = 0$
	Smooth over entire sphere!	

Ex: Diffusion equation on the sphere

- Diffusion of a scalar valued quantity u on the surface of the unit sphere

- The governing PDE can be written in Cartesian coordinates as:



$$u_t = \Delta_s u + f(t,u)$$
$$= (\Delta_c - \mathbf{x}^T(2 + \mathbf{x}^T\nabla_c)\nabla_c)u + f(t,u)$$

No coordinate singularities!

Approximating the surface gradient

- We will illustrate how to approximate the surface gradient operator using RBFs:

$$\mathbf{P}\nabla = (\mathbf{I} - \mathbf{x}\mathbf{x}^T)\nabla = \begin{bmatrix} (1-x^2) & -xy & -xz \\ -xy & (1-y^2) & -yz \\ -xz & -yz & (1-z^2) \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x \cdot \nabla \\ \mathbf{p}_y \cdot \nabla \\ \mathbf{p}_z \cdot \nabla \end{bmatrix}$$

$\mathbf{P}$ *projects* arbitrary three-dimensional vectors onto a plane tangent to the unit sphere at $\mathbf{x}$.

- Goal: Construct good numerical approximations to

$$\mathcal{D}_x = \mathbf{p}_x \cdot \nabla, \quad \mathcal{D}_y = \mathbf{p}_y \cdot \nabla, \quad \mathcal{D}_z = \mathbf{p}_z \cdot \nabla$$

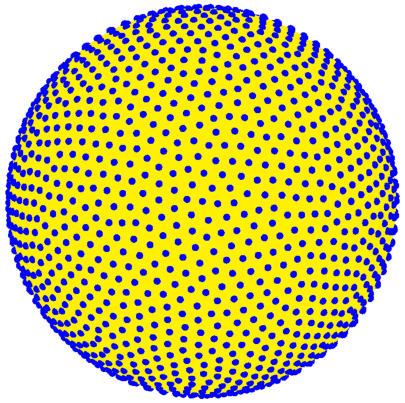
1) Global RBF method
2) Local RBF method: RBF-generated finite differences

- See problems 4-6 for more details on the global method.

Surface gradient approximation: Global RBF method

Part 2

- Setup: $X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$ and $f\Big|_X$ samples of a target function.



- ϕ is some differentiable PD or CPD(1) kernel on $\mathbb{R}^3$.
- RBF interpolant of $f\big|_X$ is given by

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|)$$

- The coefficients c_j are determined from:

$$\underbrace{\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots \quad \ddots \quad \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_N - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}}_{\underline{c}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\underline{f}}$$

- Discretization of the projected gradient closely follows Flyer & W (2007,2009).

Surface gradient approximation: Global RBF method

Part 2

- Approximate the x-component of the surface gradient using collocation:

$$\begin{aligned}
\mathbf{p}_x \cdot \nabla s(\mathbf{x})]\Big|_{\mathbf{x}=\mathbf{x}_j} &= \sum_{k=1}^{N} c_k [\mathbf{p}_x \cdot \nabla \phi_k(\|\mathbf{x}-\mathbf{x}_k\|)]\Big|_{\mathbf{x}=\mathbf{x}_j}, \qquad j = 1,\ldots,N \\
&= \sum_{k=1}^{N} c_k \underbrace{\left[x_j\, \mathbf{x}_j^T \mathbf{x}_k - x_k\right] \left(\frac{\phi_k'(\|\mathbf{x}-\mathbf{x}_k\|)}{\|\mathbf{x}-\mathbf{x}_k\|}\right)\Big|_{\mathbf{x}_j}}_{B^x_{j,k}}, \quad j = 1,\ldots,N \\
&= B^x \underline{c} \\
&= \left(B^x A^{-1}\right) \underline{f} \\
&= D_N^x \underline{f}
\end{aligned}$$

- D_N^x is an N-by-N differentiation matrix (DM) .
- It represents the discrete RBF approximation to $\mathbf{p}_x \cdot \nabla$ at nodes X.
- DMs D_N^y and D_N^z can similarly be constructed for $(\mathbf{p}_y \cdot \nabla)$ and $(\mathbf{p}_z \cdot \nabla)$.

Global RBF collocation for transport equation

- Transport equation in Cartesian coordinates

$$h_t + \mathbf{u} \cdot (P \nabla h) = 0$$

- Let h and $\mathbf{u} = (u, v, w)$ be sampled at X.
- Semi-discrete formulation (method-of-lines) of transport equation (see Problem 5):

$$\underline{h}_t = -\left(\mathrm{diag}(\underline{u}) D_N^x + \mathrm{diag}(\underline{v}) D_N^y + \mathrm{diag}(\underline{w}) D_N^z\right) \underline{h} = -D_N \underline{h}.$$

- Advance the system in time using some standard ODE solver.
- This is a purely hyperbolic problem and temporal stability can be an issue.
  - We stabilize the method by including some high-order diffusion operator L_N (hyperviscosity):

$$\underline{h}_t = -D_N \underline{h} + \mu L_N \underline{h}$$

  - L_N is a discrete approximation to a high power of the Laplacian: Δ^{2k}.

Numerical results: solid body rotation

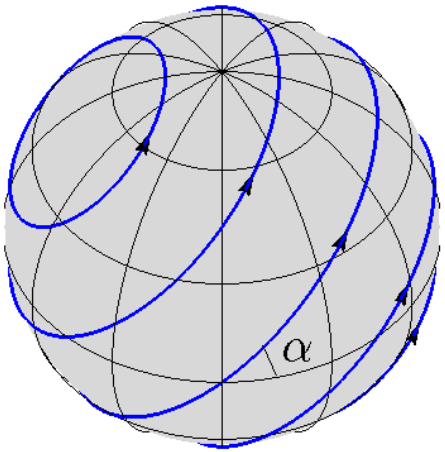
- Solid body rotation of a non-smooth cosine bell
  (Williamson et. al. JCP (1992))

  Stream Function for flow

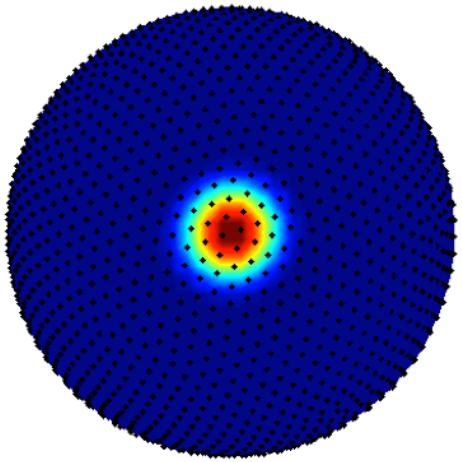
$$\psi(\mathbf{x}) = \cos(\alpha)z + \sin(\alpha)y \qquad \alpha = \pi/2 \text{ (flow over the poles)}$$

  Initial condition (non-smooth: jump in second derivative)

$$h(\mathbf{x}) = \begin{cases} \frac{1}{2}\left(1 + \cos(3\pi r(\mathbf{x}))\right) & r(\mathbf{x}) < 1/3 \\ 0 & r(\mathbf{x}) \geq 1/3 \end{cases} \qquad r(\mathbf{x}) = \arccos(x)$$



Flow direction



Initial condition

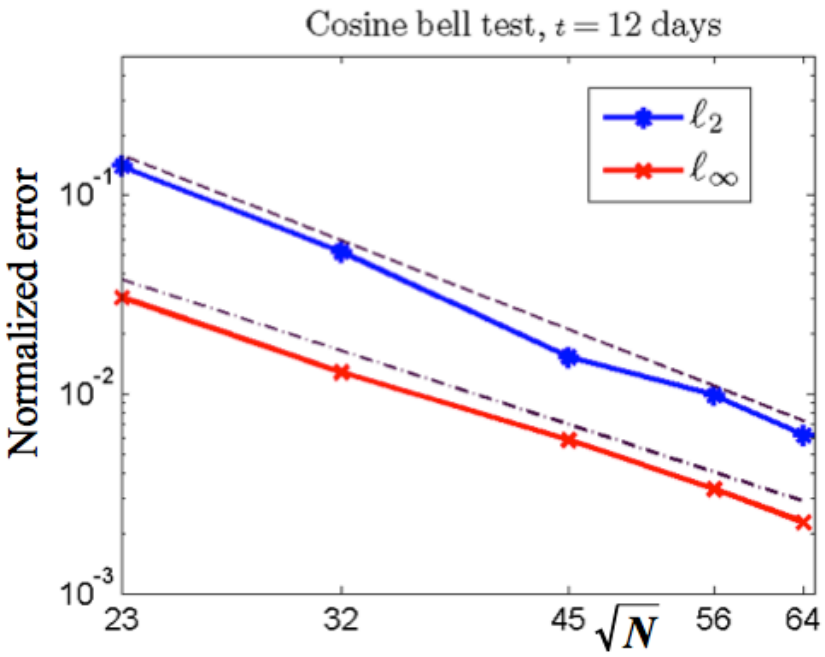
Details:

- Gaussian RBF
- $\Delta t = 30$ minutes
- No stabilization required.
- Minimum energy node sets used.

Numerical results: solid body rotation

- Convergence results as number of nodes N increases (Flyer & W, 2007)
- Error results are for one complete revolution of the cosine bell.

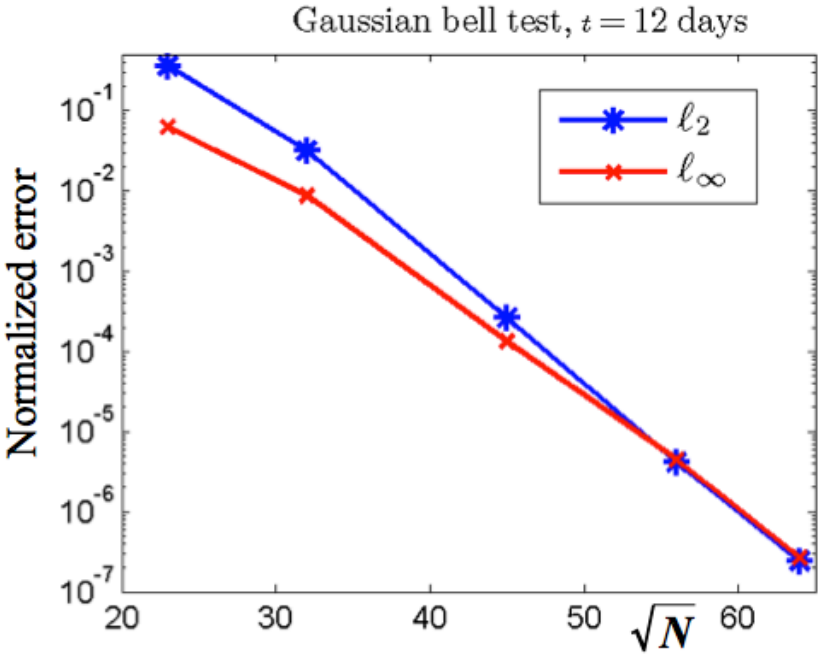
Cosine bell IC,
Discontinuous 2nd derivative



log-log scale

Straight line indicates **algebraic accuracy**

Gaussian bell IC,
Infinitely smooth



log-linear scale

Straight line indicates **spectral accuracy**

Numerical results: solid body rotation

- Comparison to other high order methods (Flyer & W, 2007)

Method	Cost per time-step	ℓ_2 error	Time-step	Number of grid points	Code length (# of lines)	Local mesh refinement
RBF	$O(N^2)$	0.006	1/2 hour	4096	< 40	yes
SH	$O(M^{3/2})$	0.005	90 seconds	32768	> 500	no
DF	$O(N \log N)$	0.005	90 seconds	32768	> 100	no
DG	$O(kN_e)$	0.005	6 minutes	7776	> 1000	yes

RBF=radial basis functions, SH=spherical harmonics, DF=double Fourier, DG=discontinuous Galerkin spectral elements

Comments:

- For RBF and DF $N =$ the number of grid points.
- For SH $M =$ total number of spherical harmonics: $(85+1)^2 = 7396$.
- For DG $N_e =$ total number of nodes per element, and k=number of elements.

Numerical results: solid body rotation

- Comparison to other high order methods (Flyer & W, 2007)

Method	Cost per time-step	ℓ_2 error	Time-step	Number of grid points	Code length (# of lines)	Local mesh refinement
RBF	$O(N^2)$	0.006	1/2 hour	4096	< 40	yes
SH	$O(M^{3/2})$	0.005	90 seconds	32768	> 500	no
DF	$O(N \log N)$	0.005	90 seconds	32768	> 100	no
DG	$O(kN_e)$	0.005	6 minutes	7776	> 1000	yes

RBF=radial basis functions, SH=spherical harmonics, DF=double Fourier, DG=discontinuous Galerkin spectral elements

Comments:

- For RBF and DF $N =$ the number of grid points.
- For SH $M =$ total number of spherical harmonics: $(85+1)^2 = 7396$.
- For DG $N_e =$ total number of nodes per element, and k=number of elements.

Numerical results: solid body rotation

- Comparison to other high order methods (Flyer & W, 2007)

Method	Cost per time-step	ℓ_2 error	Time-step	Number of grid points	Code length (# of lines)	Local mesh refinement
RBF	$O(N^2)$	0.006	1/2 hour	4096	< 40	yes
SH	$O(M^{3/2})$	0.005	90 seconds	32768	> 500	no
DF	$O(N \log N)$	0.005	90 seconds	32768	> 100	no
DG	$O(kN_e)$	0.005	6 minutes	7776	> 1000	yes

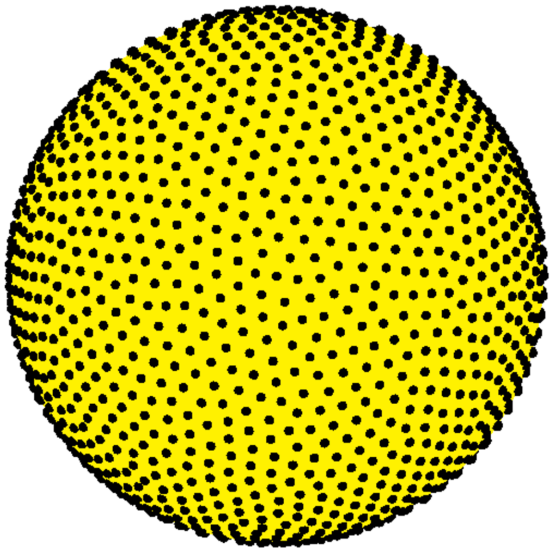
RBF=radial basis functions, SH=spherical harmonics, DF=double Fourier, DG=discontinuous Galerkin spectral elements

- Need ways to reduce this cost.

- Next method we discuss is focused on this.

RBF generated finite differences (RBF-FD)

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

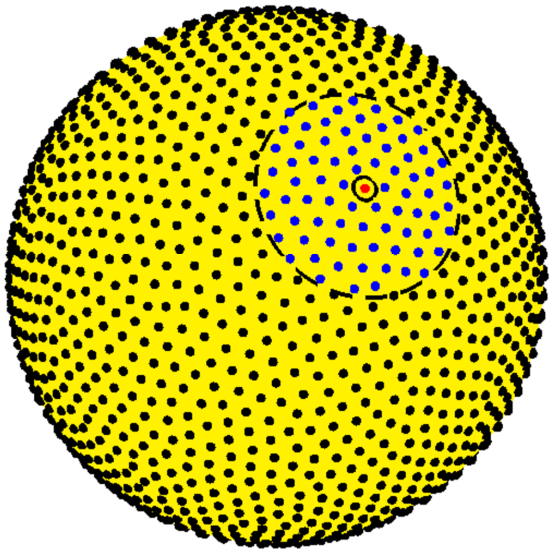


- Generalization of finite-difference (FD) method to scattered nodes using RBFs to compute the FD weights.

- References:
  - W & Fornberg (2006)
  - Fornberg & Lehto (2011)
  - Flyer, Lehto, Blaise, W & St-Cyr (2012)
  - Bollig, Flyer & Erlebacher (2012)

RBF generated finite differences

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



Key Steps:

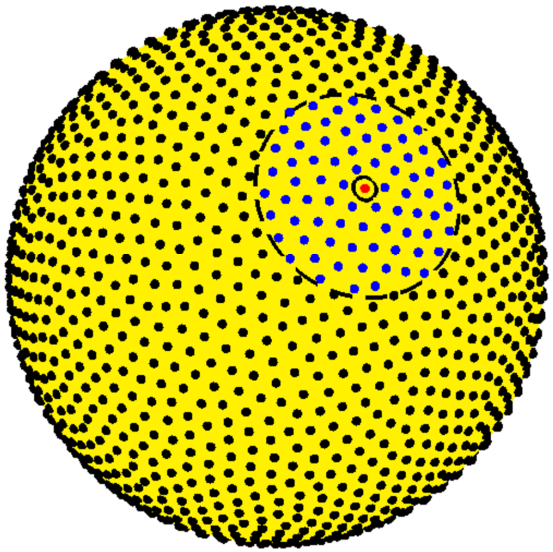
1. For each node $\mathbf{x}_j$, choose $n-1$ of it's nearest neighbors: $X_j = \{\mathbf{x}_i\}_{i=1}^{n}$, with $\mathbf{x}_1 = \mathbf{x}_j$.

2. Approximate $\mathbf{p}_x \cdot \nabla f\big|_{\mathbf{x}_j}$ using linear a combination of the values of f sampled at X_j:

$$\mathbf{p}_x \cdot \nabla f\Big|_{\mathbf{x}_j} = \sum_{i=1}^{n} c_i f(\mathbf{x}_i)$$

RBF generated finite differences

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



Key Steps:

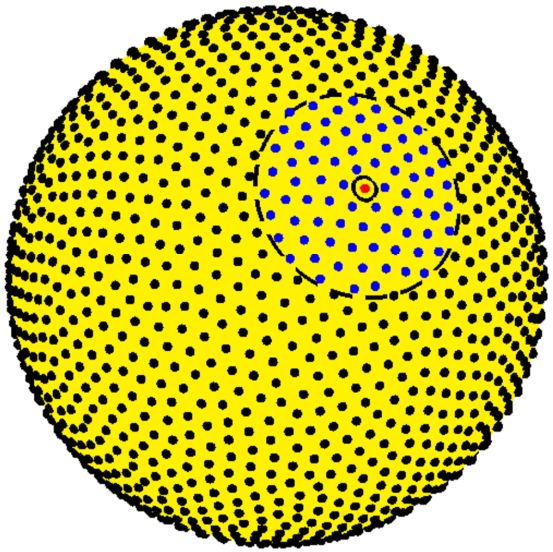
3. Choose the weights c_i such the approximation is exact for $\{\phi(\|\mathbf{x}_1 - \mathbf{x}_k\|)\}_{k=1}^n$:

$$\underbrace{[\mathbf{p}_x \cdot \nabla}_{\mathcal{D}_x} \phi(\|\mathbf{x} - \mathbf{x}_k\|)]\Big|_{\mathbf{x}=\mathbf{x}_1} \equiv \sum_{i=1}^{n} c_i \phi(\|\mathbf{x}_i - \mathbf{x}_k\|), \ k = 1, \ldots, n$$

Similar to standard FD formulas that use polynomials.

RBF generated finite differences

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



Key Steps:

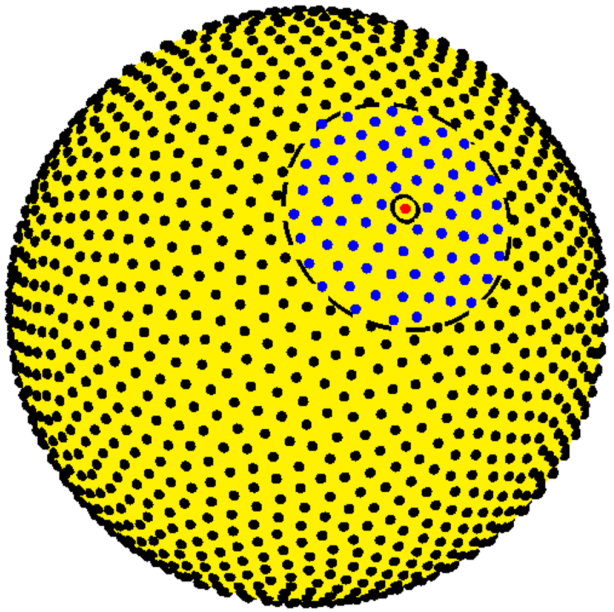
3. The weights $\{c_i\}_{i=1}^n$ can be computed by solving:

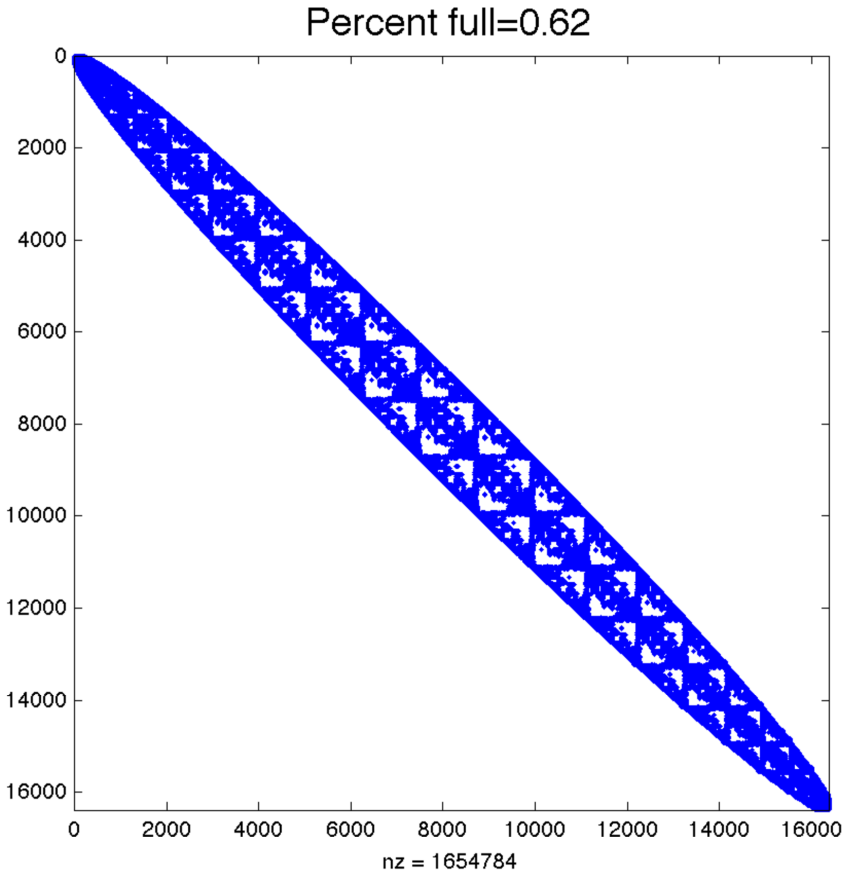
$$\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_1 - \mathbf{x}_n\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_2 - \mathbf{x}_n\|) \\ \vdots & \vdots \quad \ddots \quad \vdots \\ \phi(\|\mathbf{x}_n - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_n - \mathbf{x}_2\|) \cdots \phi(\|\mathbf{x}_n - \mathbf{x}_n\|) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \mathcal{D}_x\phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) \\ \mathcal{D}_x\phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) \\ \vdots \\ \mathcal{D}_x\phi(\|\mathbf{x}_1 - \mathbf{x}_n\|) \end{bmatrix}$$

4. Combine all the weights into a differentiation matrix.

RBF generated finite differences

- Example differentiation matrix (DM) for N=16384, n=101:





- Compare to the global RBF method, which results in a dense differentiation matrix.

RBF-FD method for transport equation

- Transport equation in Cartesian coordinates

$$h_t + \mathbf{u} \cdot (P \nabla h) = 0$$

- Let h and $\mathbf{u} = (u, v, w)$ be sampled at X.
- Semi-discrete formulation (method-of-lines) of transport equation (see Problem 5):

$$\underline{h}_t = -\left(\mathrm{diag}(\underline{u}) D_N^x + \mathrm{diag}(\underline{v}) D_N^y + \mathrm{diag}(\underline{w}) D_N^z\right) \underline{h} = -D_N \underline{h}.$$

- Advance the system in time using some standard ODE solver.
- This is a purely hyperbolic problem and temporal stability can be an issue.
  - We stabilize the method by including some high-order diffusion operator L_N (hyperviscosity):

$$\underline{h}_t = -D_N \underline{h} + \mu L_N \underline{h}$$

  - L_N is a discrete approximation to a high power of the Laplacian: Δ^{2k}.

Numerical results: solid body rotation

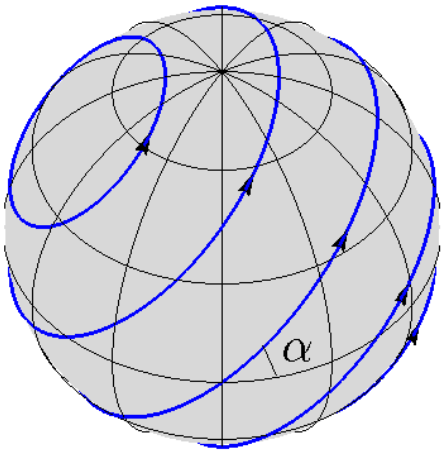
- Solid body rotation of a non-smooth cosine bell
  (Williamson et. al. JCP (1992))

  Stream Function for flow

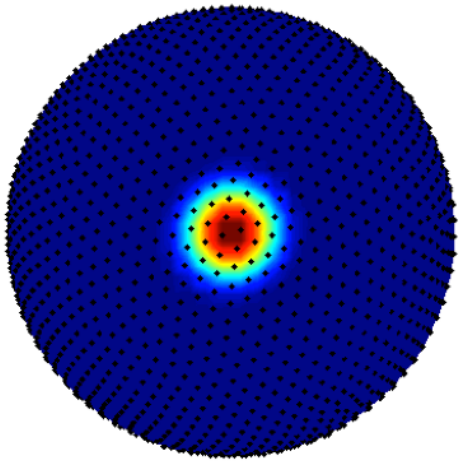
$$\psi(\mathbf{x}) = \cos(\alpha)z + \sin(\alpha)y \qquad \alpha = \pi/2 \text{ (flow over the poles)}$$

  Initial condition (non-smooth: jump in second derivative)

$$h(\mathbf{x}) = \begin{cases} \frac{1}{2}\left(1 + \cos(3\pi r(\mathbf{x}))\right) & r(\mathbf{x}) < 1/3 \\ 0 & r(\mathbf{x}) \geq 1/3 \end{cases} \qquad r(\mathbf{x}) = \arccos(x)$$



Flow direction



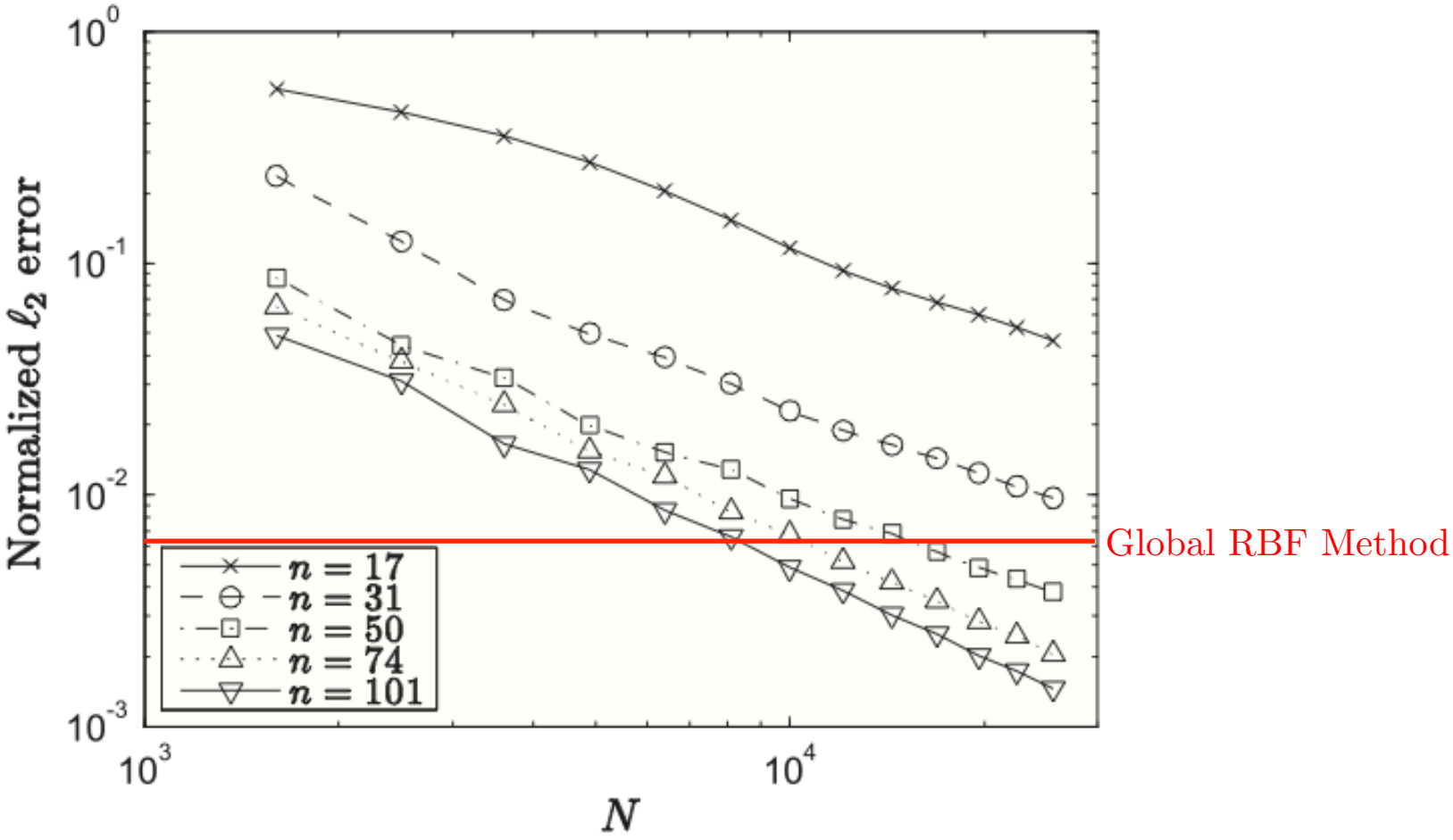
Initial condition

Details:

- Gaussian RBF
- Stabilization required.
- Minimum energy node sets used.

Numerical results: solid body rotation

- Convergence results as number of nodes N increases (Fornberg & Lehto, 2011)
- Error results are for 10 complete revolution of the cosine bell.

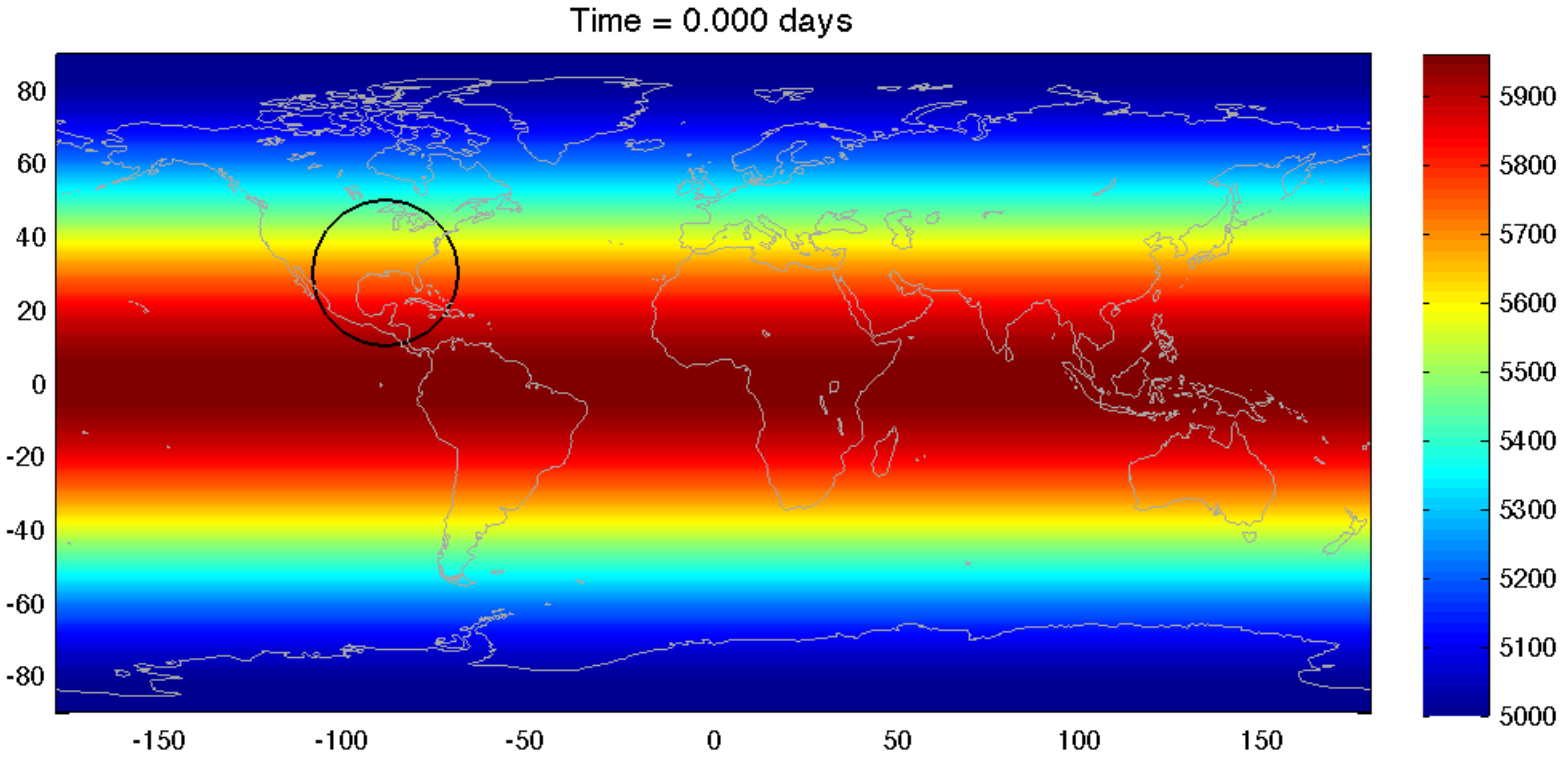


- Errors compare favorably with the global RBF method.
- RBF-FD method much more computationally efficient than global method.

RBF-FD for shallow water wave eqs.

- Numerical simulation: Flow over an isolated mountain

  (Test Case 5 from Williamson *et. al.*, JCP (1992))
- See Flyer *et. al.*, JCP (2012)



Remarks regarding solving PDEs

- The Global RBF collocation method is competitive in terms of accuracy per degree of freedom.
- It are not competitive in terms of computational complexity.
- The RBF generated finite difference (RBF-FD) method shows great promise in terms of accuracy and computational cost.
  - Comparisons with other state-of-the art methods have been done (Flyer *et. al.* 2012) and show the RBF-FD is competitive in terms of accuracy and computational complexity.
  - Parallelization on multi-GPU has already been implemented (Bollig, Flyer, & Erlebacher, 2012).
- Research Ideas for RBF-FD method
  - Extend method to advection-reaction-diffusion on the sphere.
  - Extend method to non-uniform nodes (static/adaptive refinement)
  - Extend method to more general surfaces
  - Develop method for the hemisphere.
  - Develop method for 3D spherical shell
  - Incorporate into immersed boundary type setting for 2D or 3D problems.
  - Couple surface PDEs to PDEs in the bulk medium.