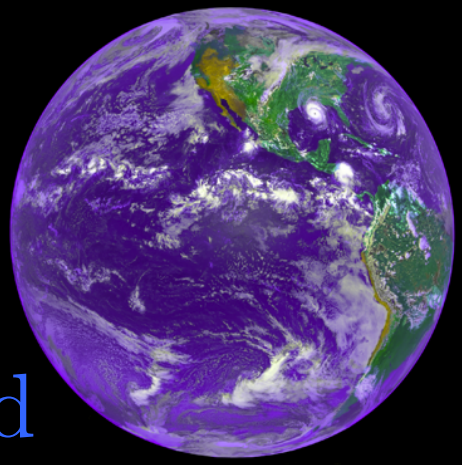


Combining radial basis functions with the partition-of-unity method for numerically solving PDEs on the sphere

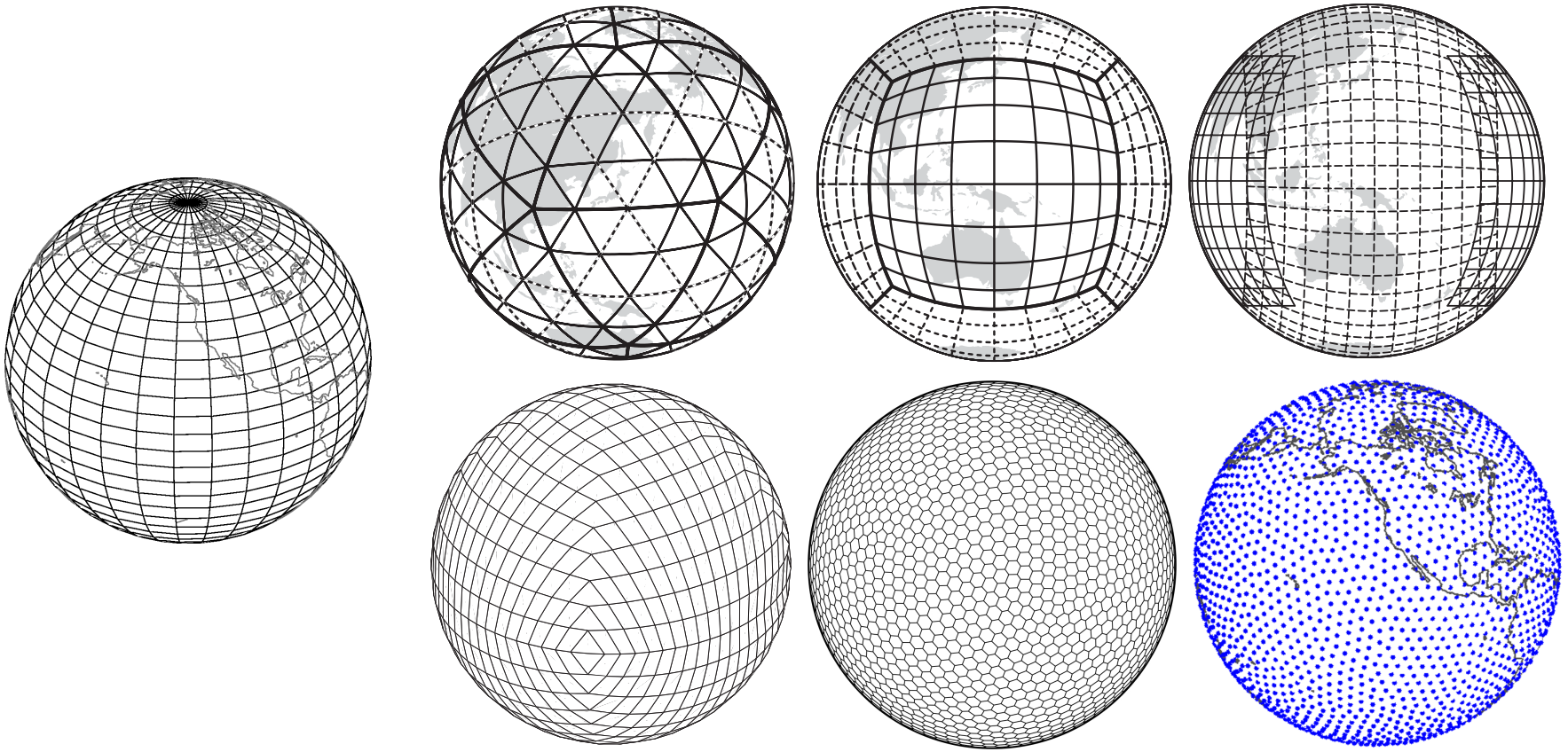
Grady B. Wright
Boise State University

Collaborator: Kevin Aiton, Boise State University

*This work is supported by NSF CMG grant DMS 0934581

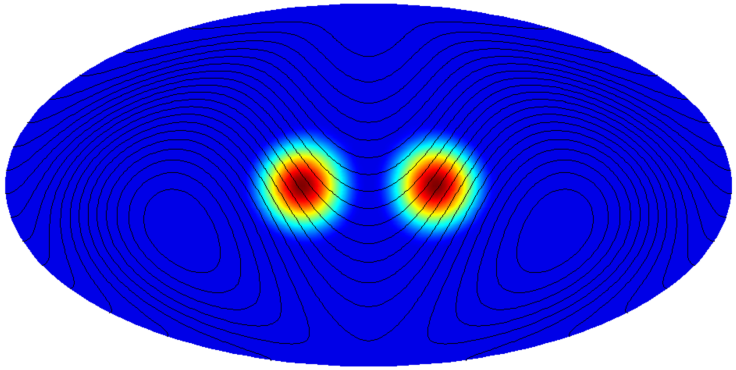
Methods for PDEs on spheres

- Grids/meshes/nodes used in methods for large-scale applications:



- Methods used:
  - Finite-difference, finite-element, finite-volume, semi-Lagrangian
  - Double Fourier, spherical harmonics, spectral elements, discontinuous Galerikin (DG), and radial basis functions (RBF)

Transport equation on the sphere



- We consider the transport of a scalar valued quantity h on the surface of the unit sphere in an incompressible velocity field $\mathbf{u}$.

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

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

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

- Surface gradient operator:

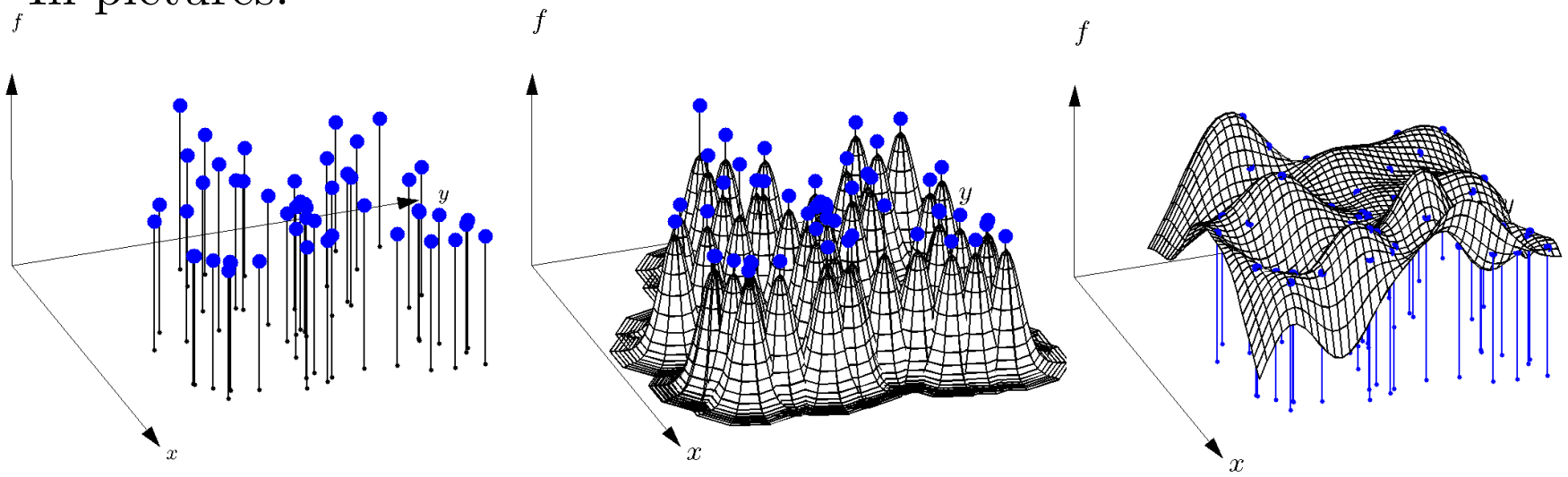
$$\mathbf{P}\nabla = (\mathbf{I} - \mathbf{x}\mathbf{x}^T)\nabla = \begin{bmatrix} (1-x^2) & -xy & -xz \\ -xy & (1-y^2) & -yz \\ -xz & -yz & (1-z^2) \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x \cdot \nabla \\ \mathbf{p}_y \cdot \nabla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

- Goal: Construct good numerical approximations to

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

Intro to RBFs via interpolation

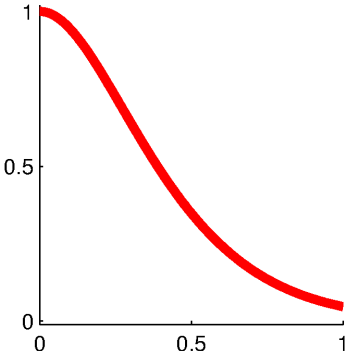
- In pictures:



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

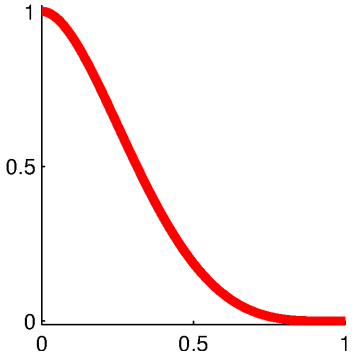
Finite-smoothness

Matérn



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

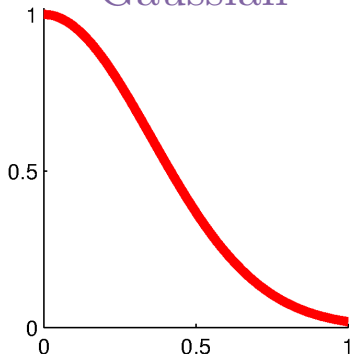
Wendland



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

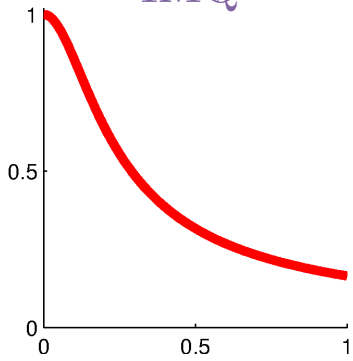
Infinite-smoothness

Gaussian



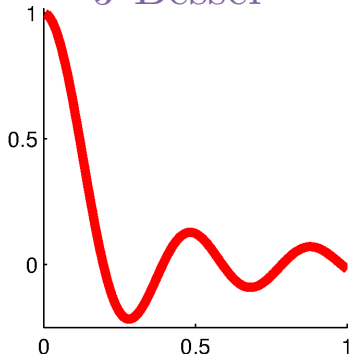
$$\exp(-(\epsilon r)^2)$$

IMQ



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

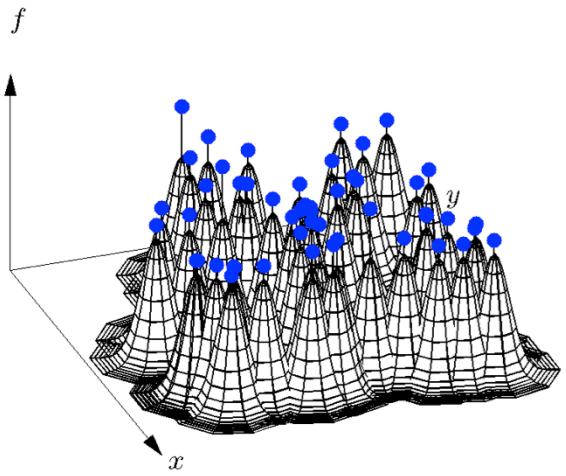
J-Bessel



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

Intro to RBFs via interpolation

- Let $X = \{\mathbf{x}_j\}_{j=1}^N$ be a distinct set of scattered nodes on $\mathbb{R}^d$ and f some target function sampled at X.



- RBF interpolant of $f\big|_X$ is given by

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

  where $\|\cdot\|$ is the standard two-norm.

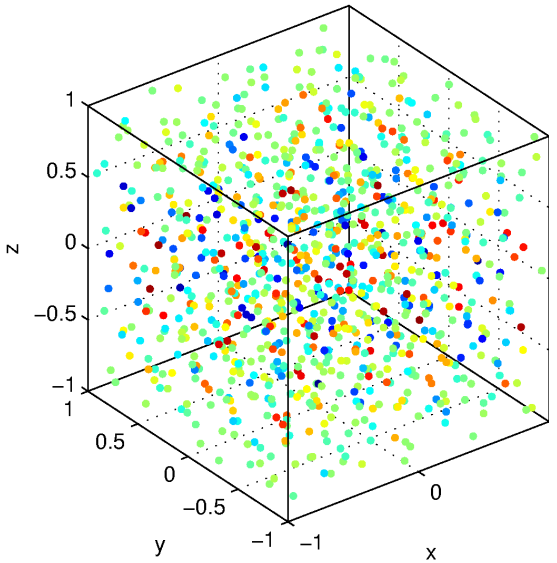
- The coefficients c_j are determined from collocation:

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

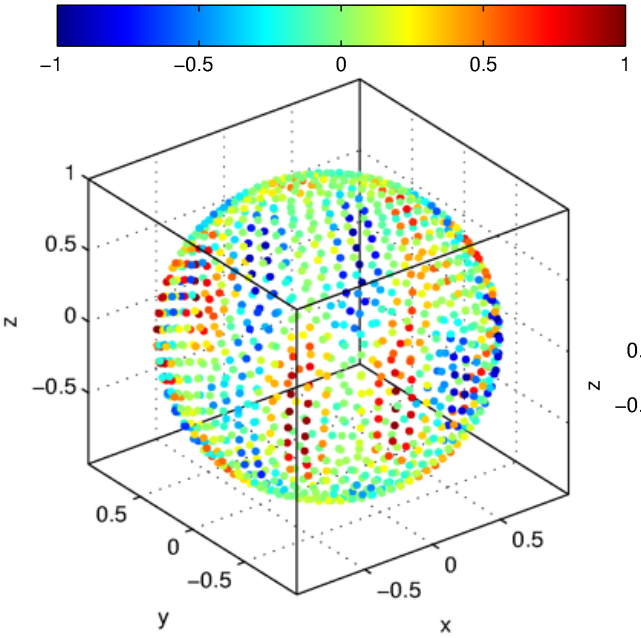
Geometric flexibility of RBFs

- Consider interpolation of f from a set of scattered nodes X on $\Omega \subset \mathbb{R}^d$

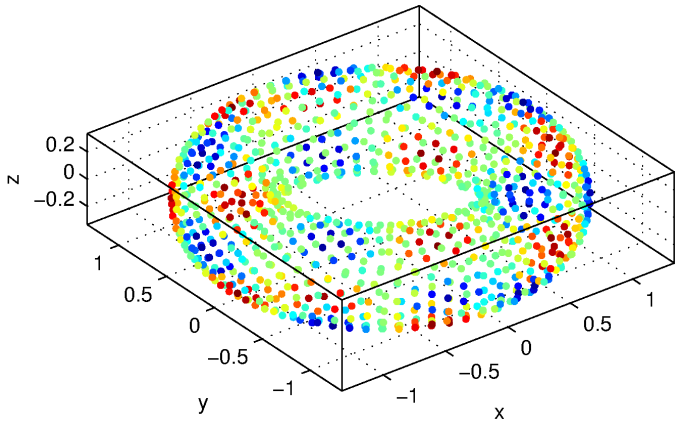
Examples:



$\Omega = [-1, 1]^3$



$\Omega =$ unit sphere



$\Omega =$ a torus

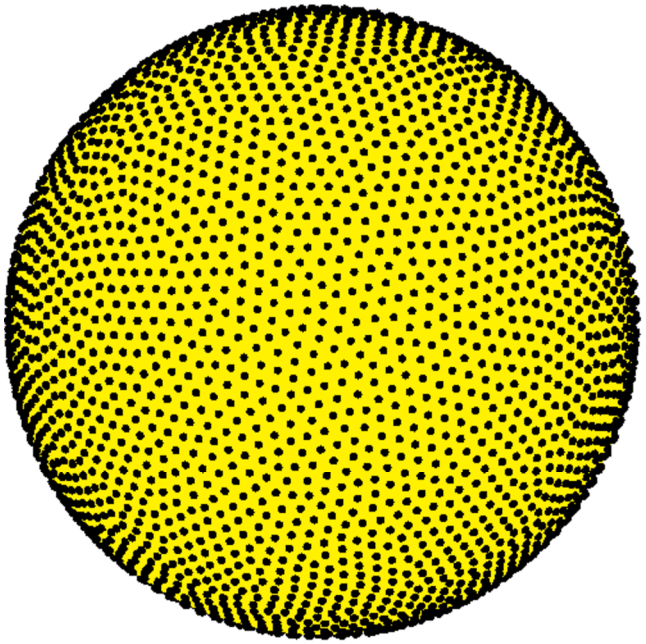
- Form of the RBF interpolant does not change:

$$s_X(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|), \quad \|\mathbf{x} - \mathbf{x}_j\|^2 = (x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2$$

- Can use Cartesian (extrinsic) coordinates!

RBFs and partition-of-unity on the sphere

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



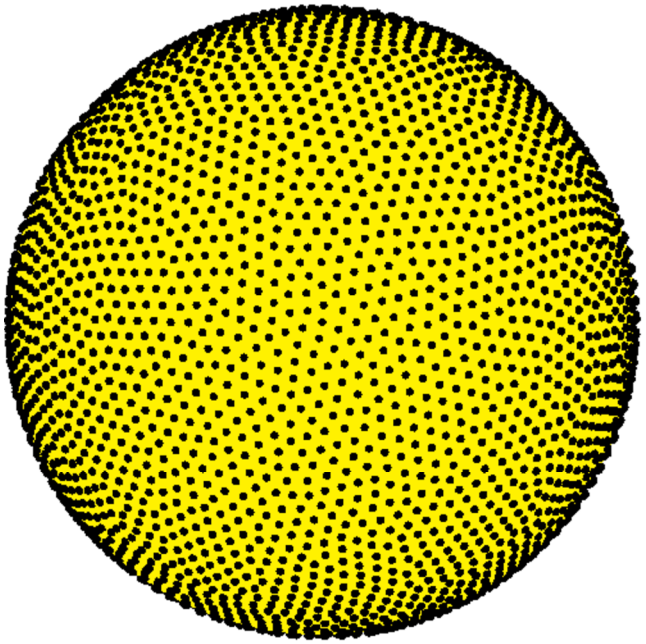
Key references:

- 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:

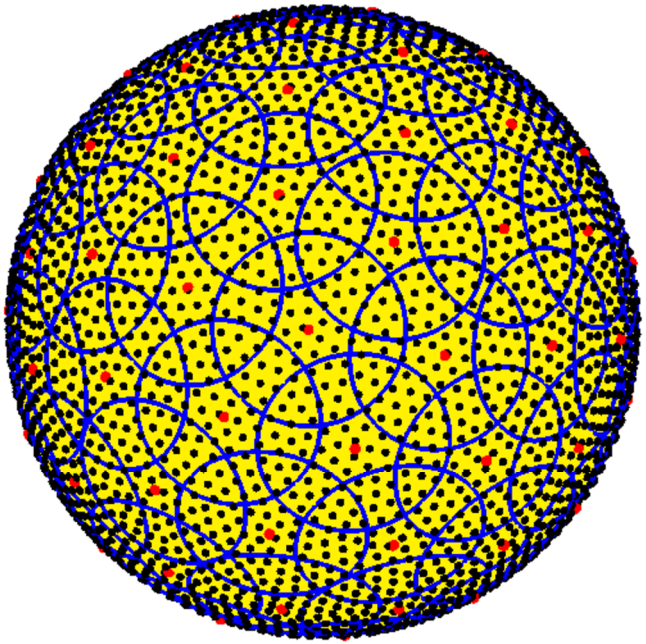
1. Generate a set of overlapping patches (spherical caps) $\Omega = \{\Omega_k\}_{k=1}^M$ with the properties:

   (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:

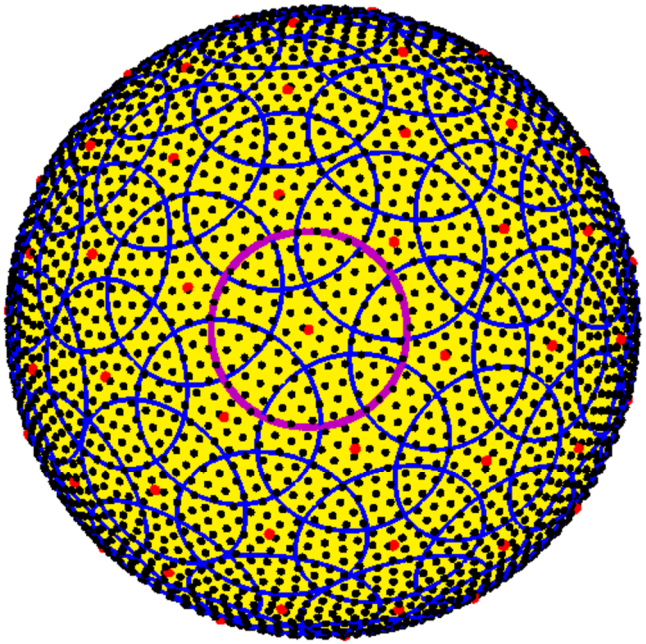
1. Generate a set of overlapping patches (spherical caps) $\Omega = \{\Omega_k\}_{k=1}^M$ with the properties:

   (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

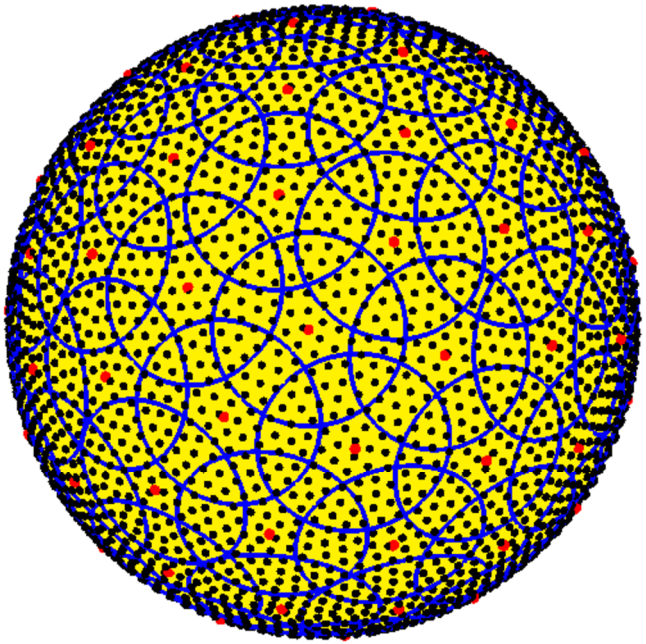
Key Steps:

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

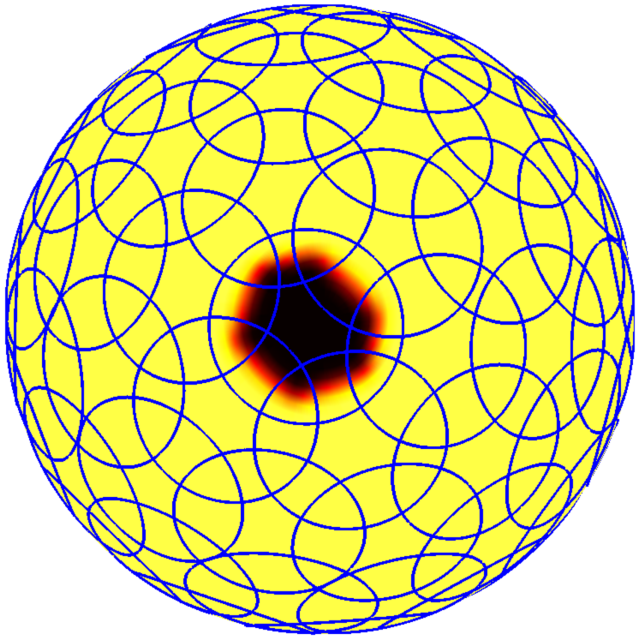
Key Steps:

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

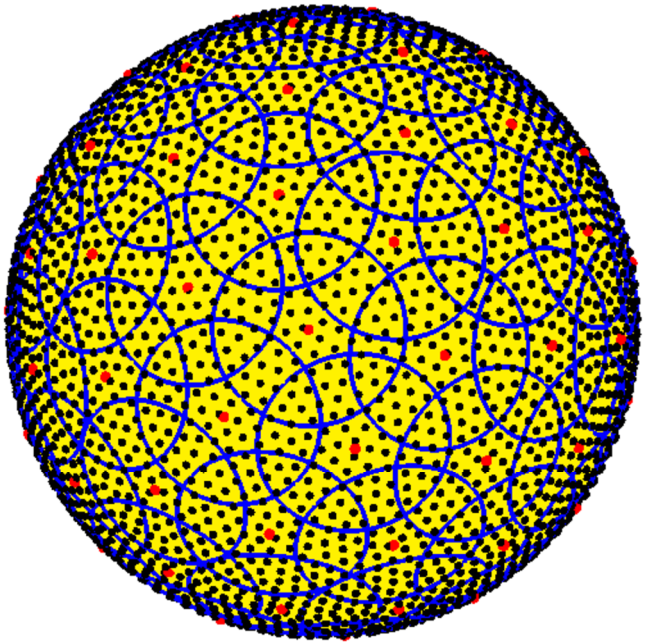
Key Steps:

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:

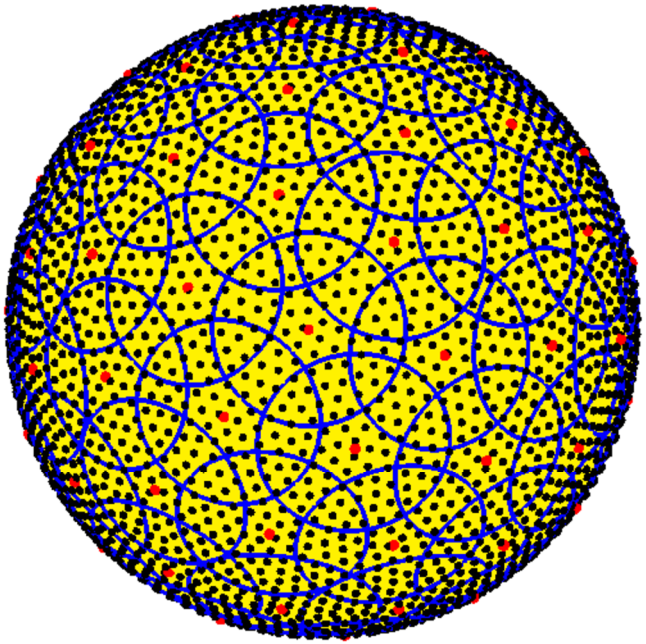
$$w_k(\mathbf{x}) = \frac{\psi_k(\mathbf{x})}{\sum_{i=1}^{M} \psi_i(\mathbf{x})} \longrightarrow \psi_k(\mathbf{x}) = \psi\left(\frac{\|\mathbf{x} - \boldsymbol{\xi}_k\|}{\rho_k}\right)$$

(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

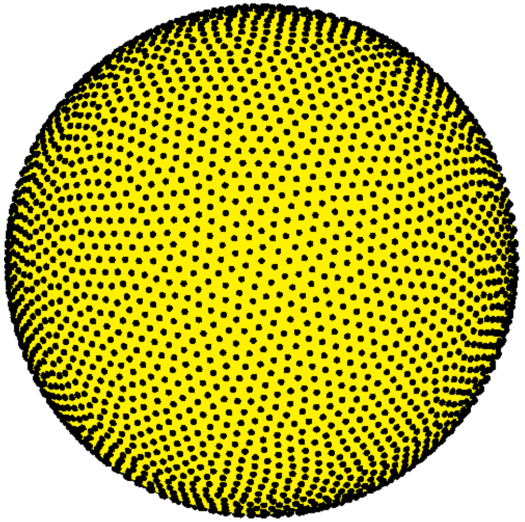
$$s_X(\mathbf{x}) = \sum_{k=1}^{M} w_k(\mathbf{x}) s_k(\mathbf{x})$$

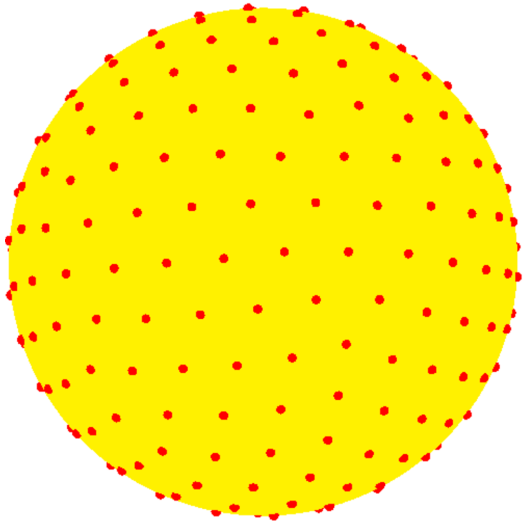
Choosing the nodes and patches

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



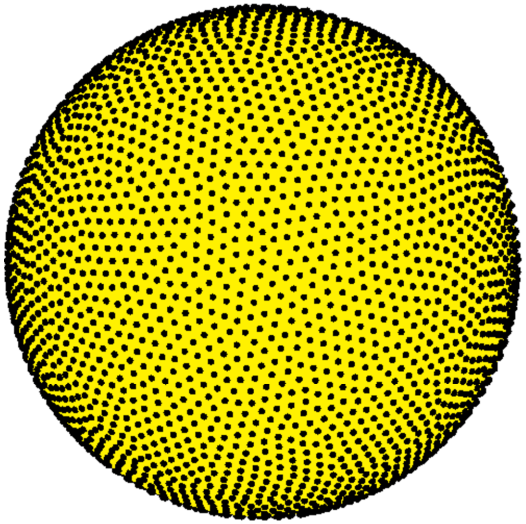


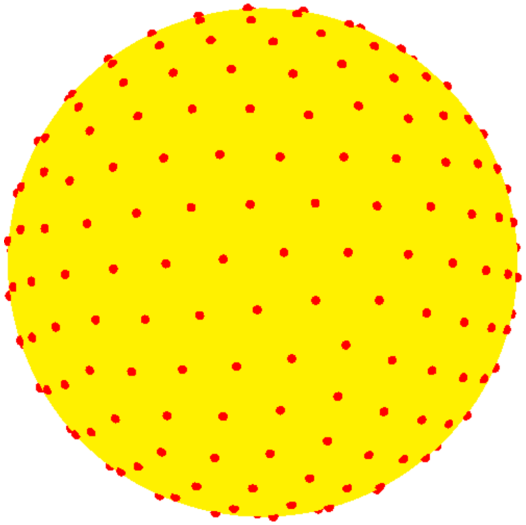
Patch centers

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:

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

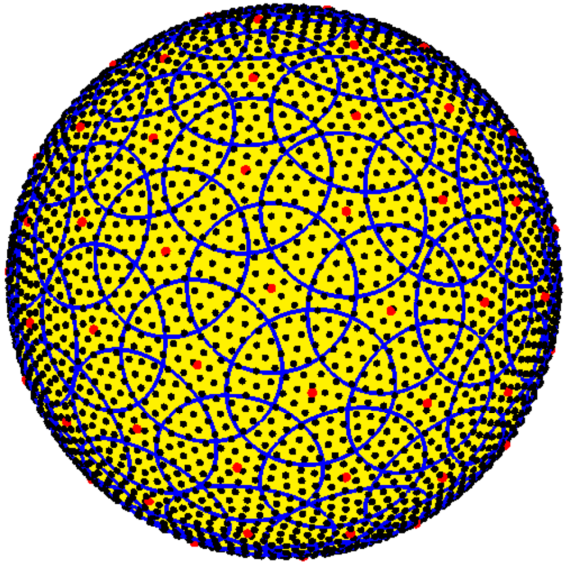
- The overlap parameter q determines the **average number of patches a node belongs to**, and satisfies the relationship:

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

Choosing the nodes and patches

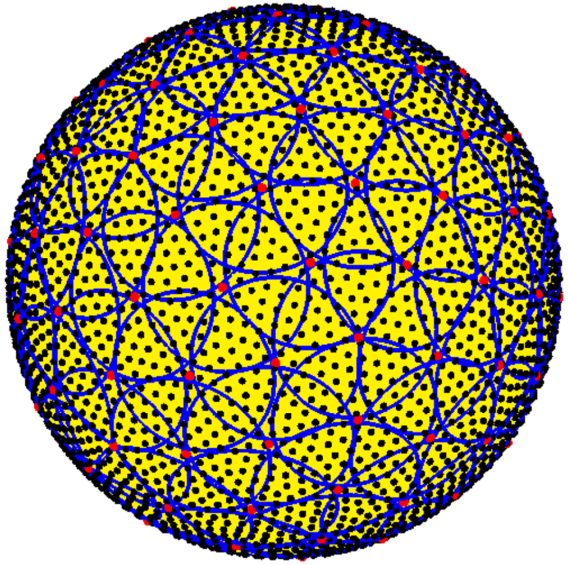
- Illustration of the patches for N=4096, n=100, and different q:

q=2



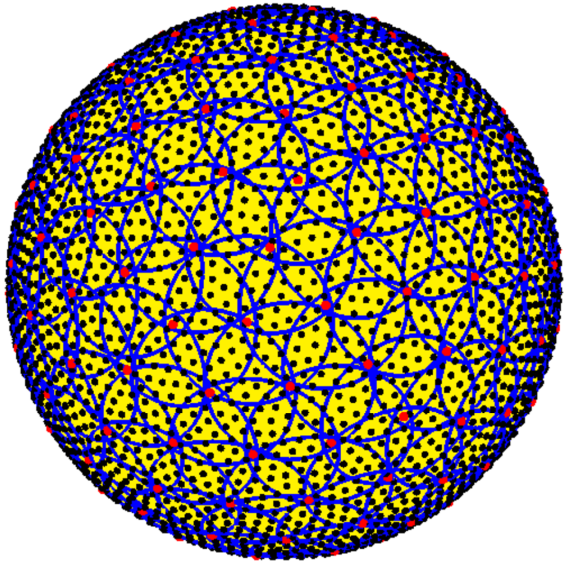
M=82

q=3



M=123

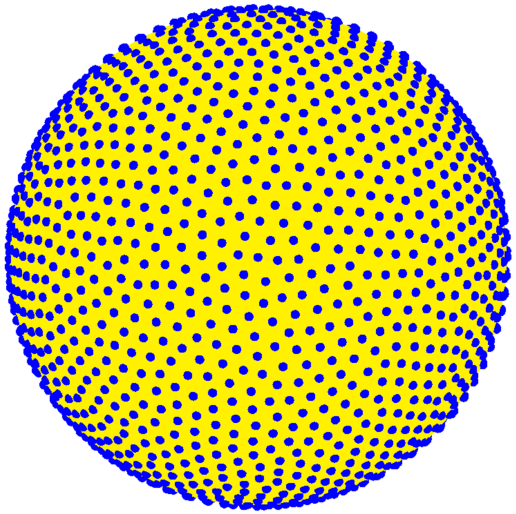
q=4



M=184

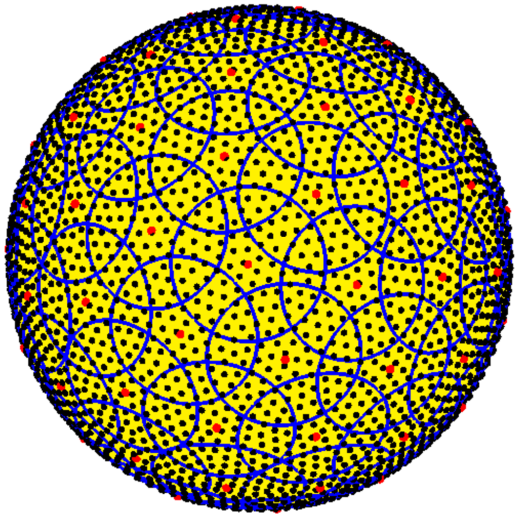
Comparison to global RBF method

Global RBFs



N=total nodes

RBF-PUM



N=total nodes

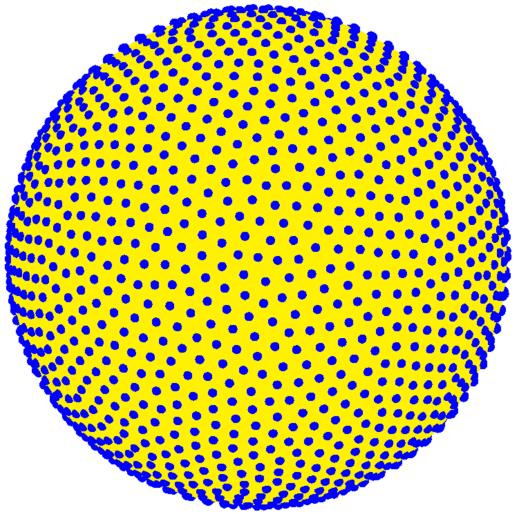
M=total patches
n=nodes per patch
q=avg. # patches
a node belongs to

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:	$O(N^2)$	$O(q\, n\, N)$

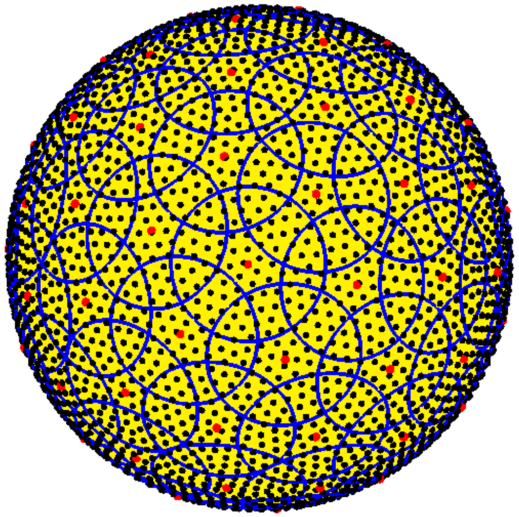
Comparison to global RBF method

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 interpolation with RBF-PUM in $\mathbb{R}^d$ says same convergence orders should be expected as global RBFs.

Spectral orders are therefore theoretically possible.

No theory on for $\mathbb{S}^2$, but should expect similar results.

Collocation of surface gradient operator

- Surface gradient operator:

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

- Let $s_X(\mathbf{x})$ be the RBF-PUM interpolant of f at the nodes, then we can approximate the x-component of the surface gradient of f by:

$$\begin{aligned} [\mathbf{p}_x \cdot \nabla f(\mathbf{x})]\Big|_{\mathbf{x}=\mathbf{x}_j} &\approx [\mathbf{p}_x \cdot \nabla s_X(\mathbf{x})]\Big|_{\mathbf{x}=\mathbf{x}_j}, \ j = 1, \ldots, N \\ &= \sum_{k=1}^{M} [\mathbf{p}_x \cdot \nabla \left(w_k(\mathbf{x}) s_k(\mathbf{x})\right)]\Big|_{\mathbf{x}=\mathbf{x}_j}, \ j = 1, \ldots, N \\ &= D_N^x \underline{f} \end{aligned}$$

- We can similarly compute the differentiation matrices D_N^y and D_N^z for $(\mathbf{p}_y \cdot \nabla)$ and $(\mathbf{p}_z \cdot \nabla)$.

RBF-PUM for transport equation

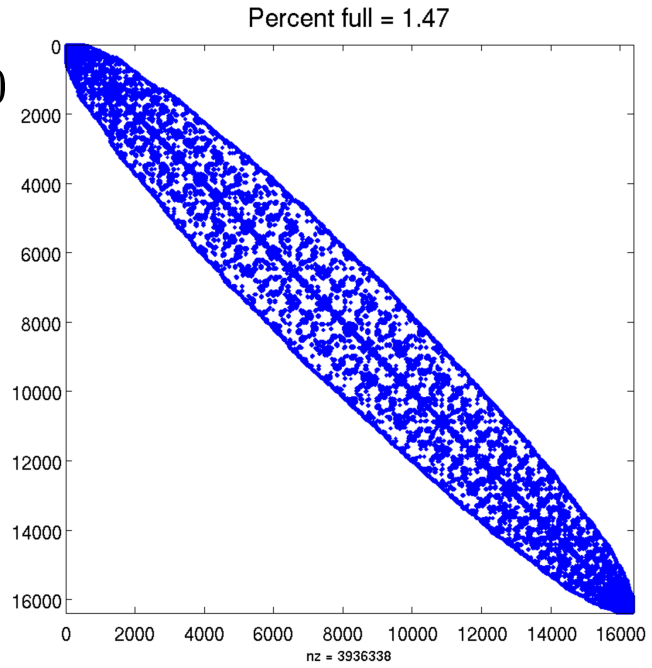
- Continuous transport equation: $h_t + \mathbf{u} \cdot (\mathbf{P}\nabla)\, h = 0$
- Semi-discrete formulation:

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

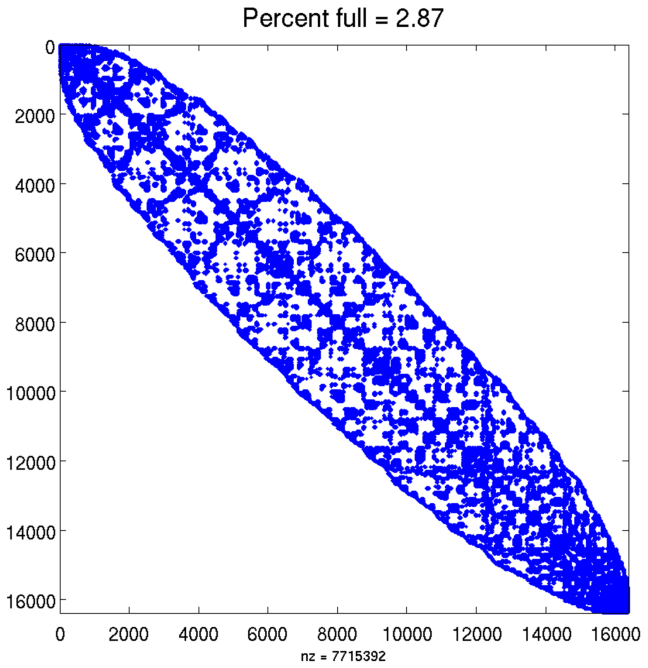
- Unlike global RBF-type methods the matrix D_N above is *sparse*!

Ex: Sparsity patterns of D_N for N=16384 and q=4

n=100



n=196

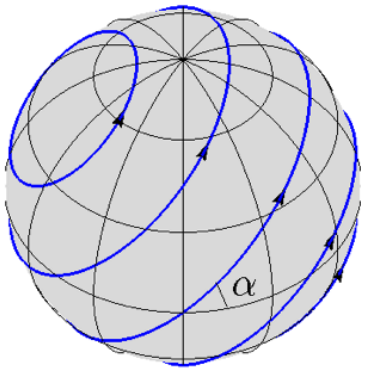


What about stability?

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

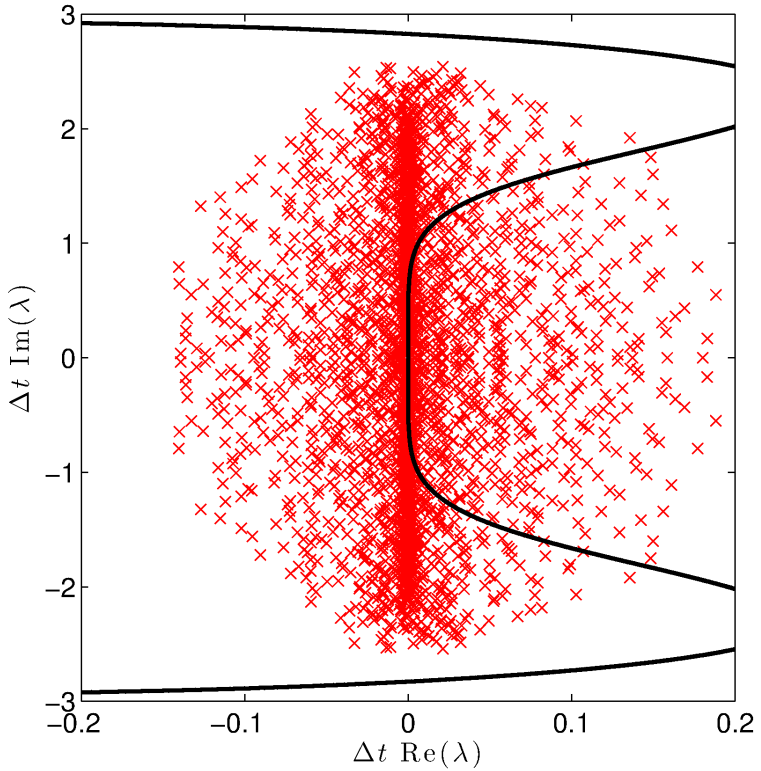
- Consider the case of where

$$\mathbf{u} = \mathbf{x} \times \nabla(\cos(\alpha)z + \sin(\alpha)y)$$

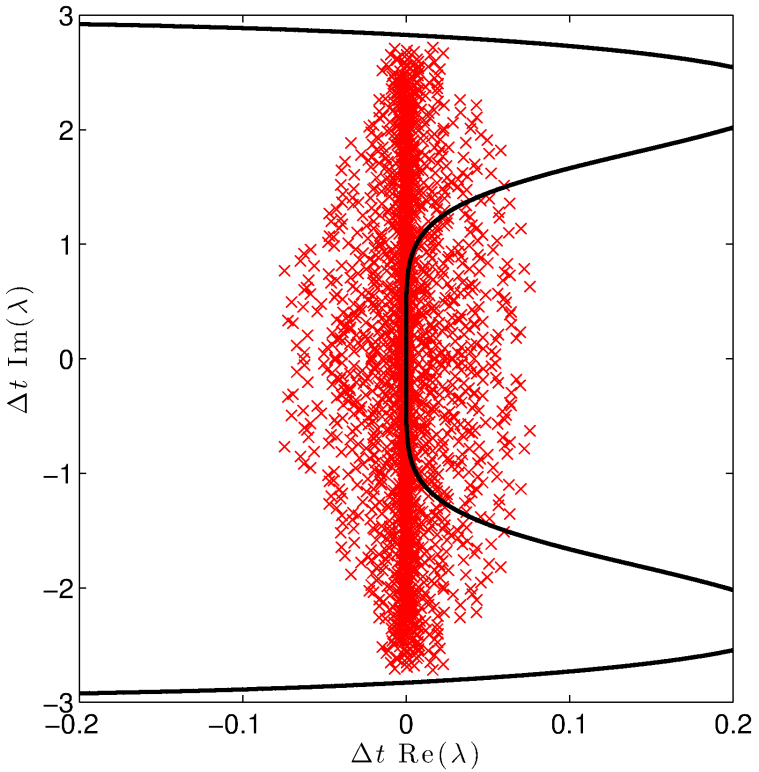


Illustrations of eigenvalues of $-D_N$ for N=4096 and q=4

n=100



n=256

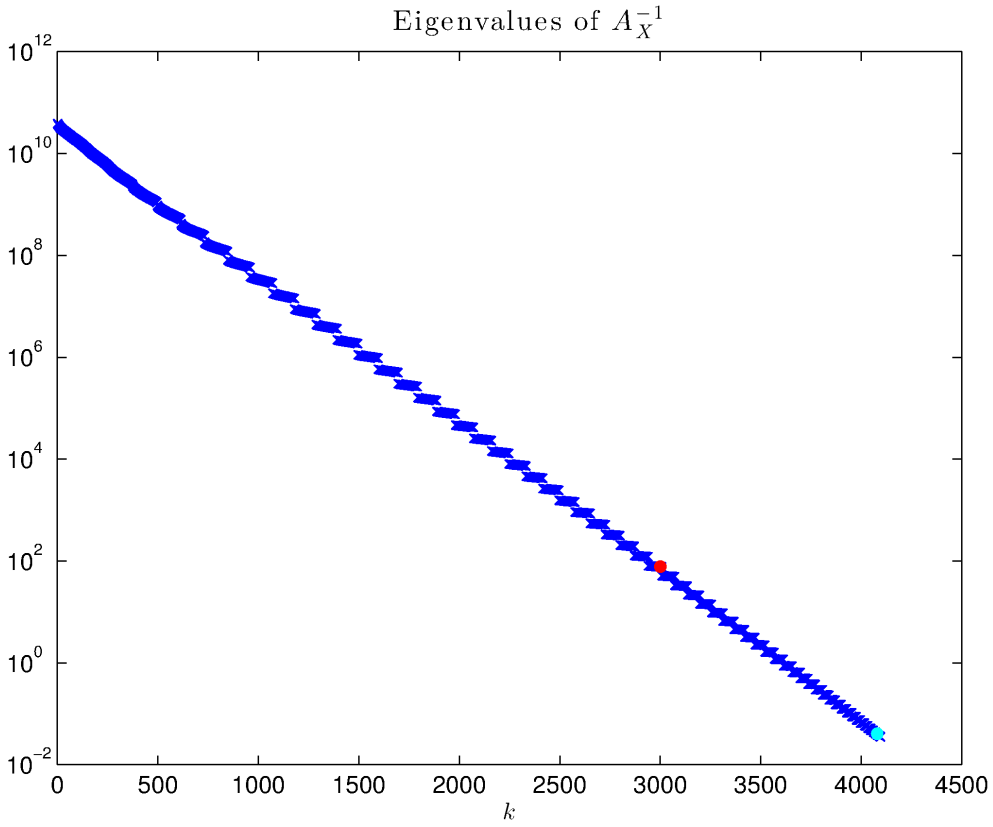


Stabilization via "global" hyperviscosity

- We need a filter to damp all these spurious eigenmodes while leaving the physically relevant ones essentially intact.

- Idea for stabilizing global RBF methods (Fornberg & Lehto, JCP (2011))

$$\frac{d}{dt}\underline{h} = -\mu(A_X)^{-1}\underline{h}, \quad \mu > 0$$



Stabilization via "global" hyperviscosity

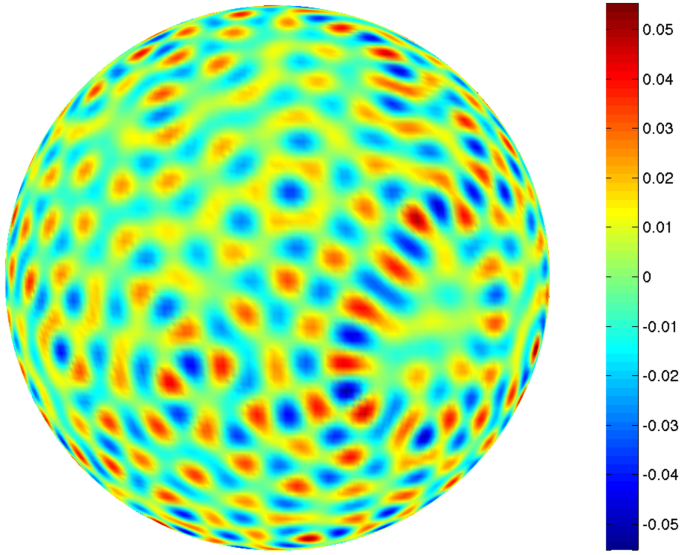
- We need a filter to damp all these spurious eigenmodes while leaving the physically relevant ones essentially intact.

- Idea for stabilizing global RBF methods (Fornberg & Lehto, JCP (2011)

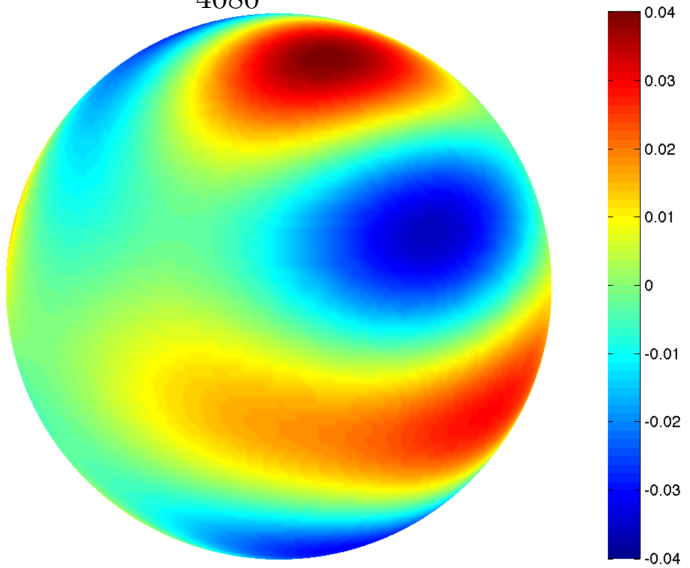
$$\frac{d}{dt}\underline{h} = -\mu(A_X)^{-1}\underline{h}, \quad \mu > 0$$

Eigenvectors:

λ_{3000}=77.2



λ_{4080}=0.0404



(A_X^{-1}) damps high-frequency modes extremely fast and low ones slowly.

Stabilization via "global" hyperviscosity

- We need a filter to damp all these spurious eigenmodes while leaving the physically relevant ones essentially intact.

- Idea for stabilizing global RBF methods (Fornberg & Lehto, JCP (2011)

$$\frac{d}{dt}\underline{h} = -\mu(A_X)^{-1}\underline{h}, \quad \mu > 0$$

- Modified PDE to solve:

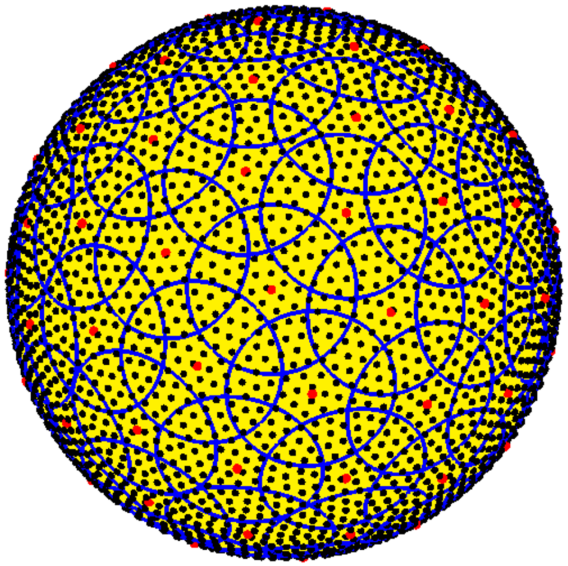
$$\frac{d}{dt}\underline{h} = -\left(D_X + \mu(A_X)^{-1}\right)\underline{h}, \quad \mu > 0$$

- Need some heuristic for choosing μ.

Stabilization via "local" hyperviscosity

- We need a filter to damp all these spurious eigenmodes while leaving the physically relevant ones essentially intact.

- Idea for stabilizing RBF-PUM:



1. On each patch Ω_k, apply A_k^{-1} to $\underline{h}_k$ (solution at time t on Ω_k).
2. Use the PUM weight functions w_k to sum up all the $A_k^{-1}\underline{h}_k(t)$.

- Modified PDE to solve:

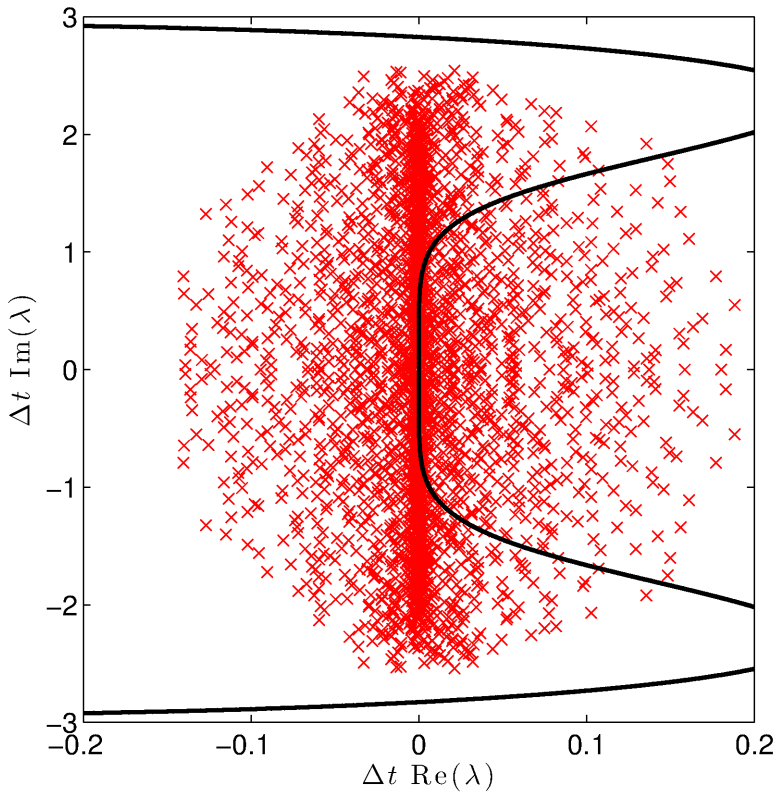
$$\frac{d}{dt}\underline{h} = -\left(D_N + \mu L_N\right)\underline{h}, \quad \mu > 0$$

Stabilization via "local" hyperviscosity

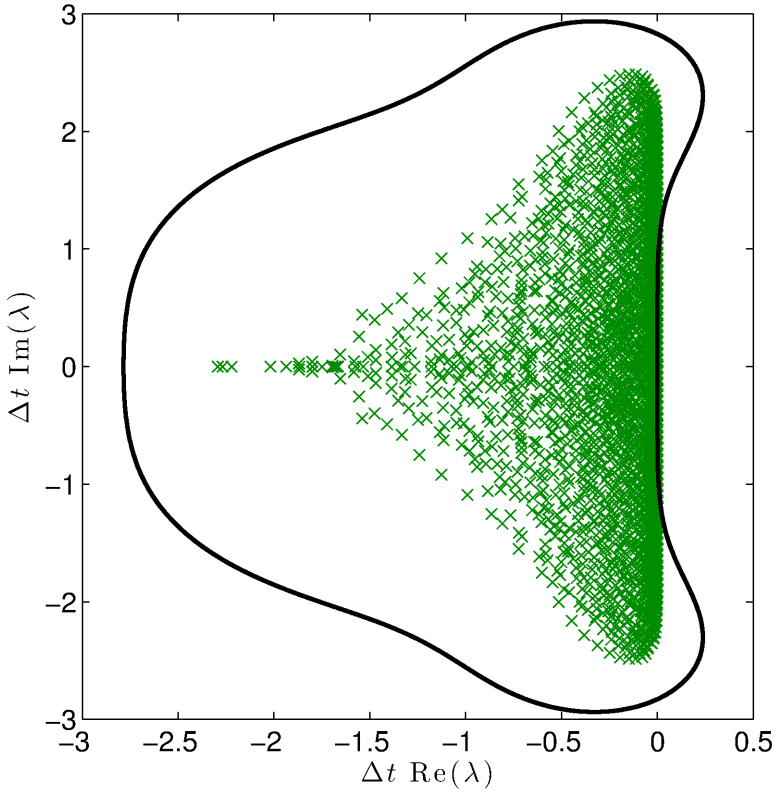
Illustrations of the eigenvalues for N=4096 and q=4:

n=100

$-D_N$



$-(D_N+\mu L_N)$



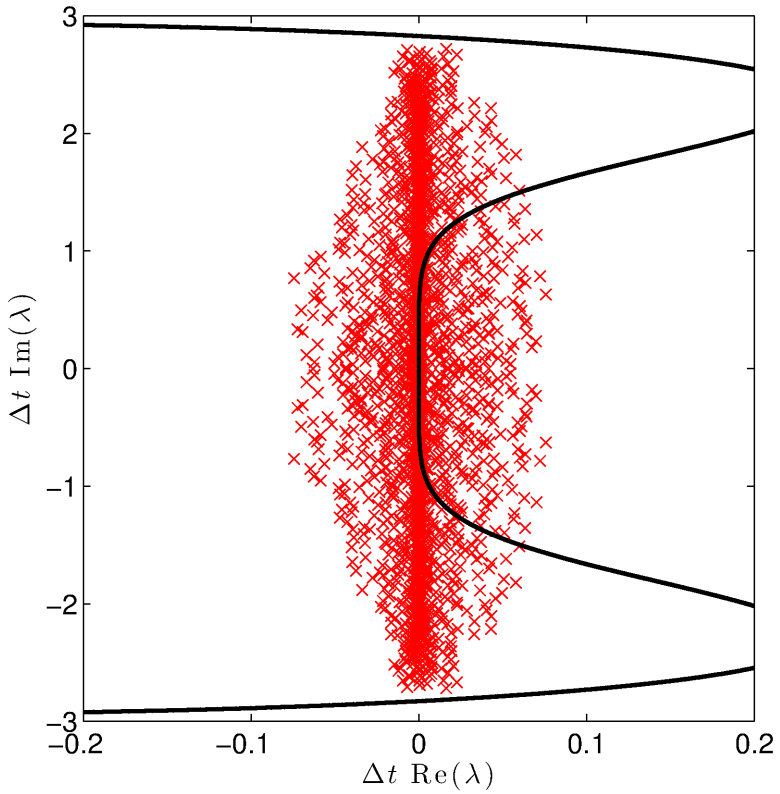
- Eigenvalues have been scaled by Δt=2π/125, μ=8×10^{-9}
- Black solid line is the stability domain of RK4

Stabilization via "local" hyperviscosity

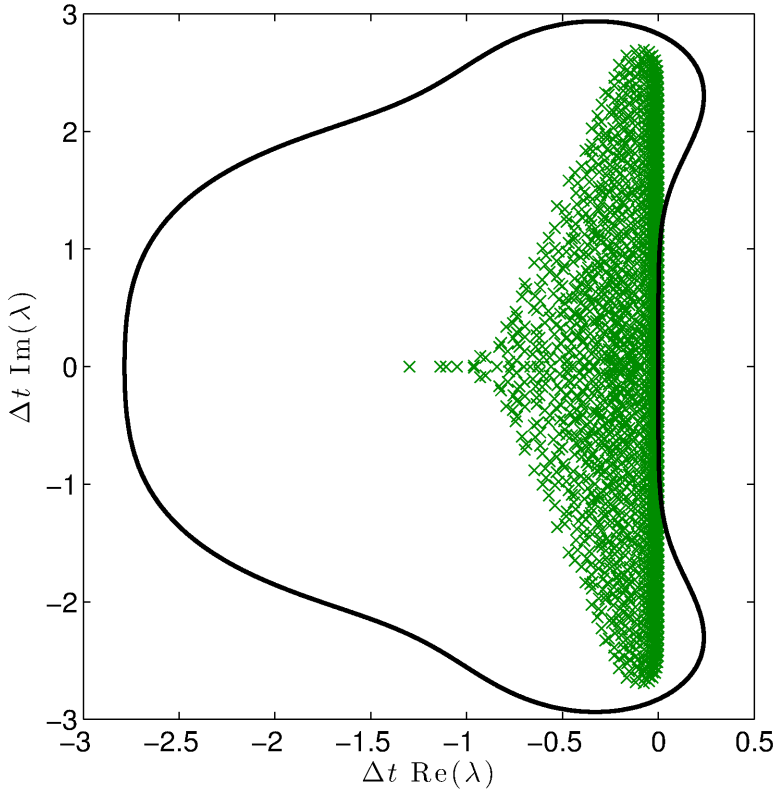
Illustrations of the eigenvalues for N=4096 and q=4:

n=256

$-D_N$



$-(D_N+\mu L_N)$



- Eigenvalues have been scaled by Δt=2π/125, μ=8×10^{-9}
- Black solid line is the stability domain of RK4

Numerical results: solid body rotation

Details for all numerical results:

- We use the Gaussian RBF
- Time-step Δt is not optimized
- Overlap is set to q=4
- We set $\epsilon = a_n\sqrt{N} + b_n$

- 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)

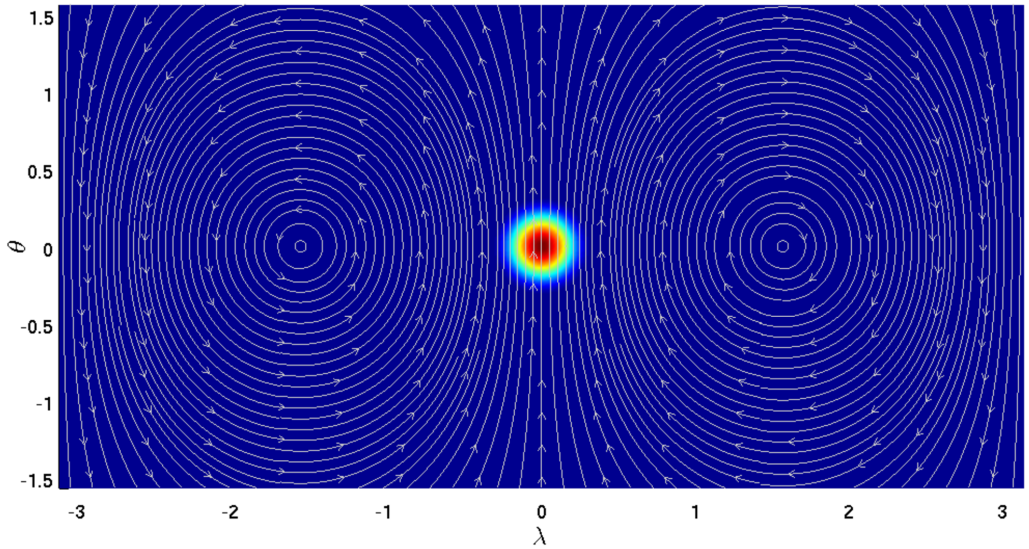
$$h(\mathbf{x}) = \begin{cases} \frac{1}{2}\left(1 + \cos(3\pi r(\mathbf{x}))\right) & r(\mathbf{x}) < 1/3 \\ 0 & r(\mathbf{x}) \geq 1/3 \end{cases}$$

$$r(\mathbf{x}) = \arccos(x)$$

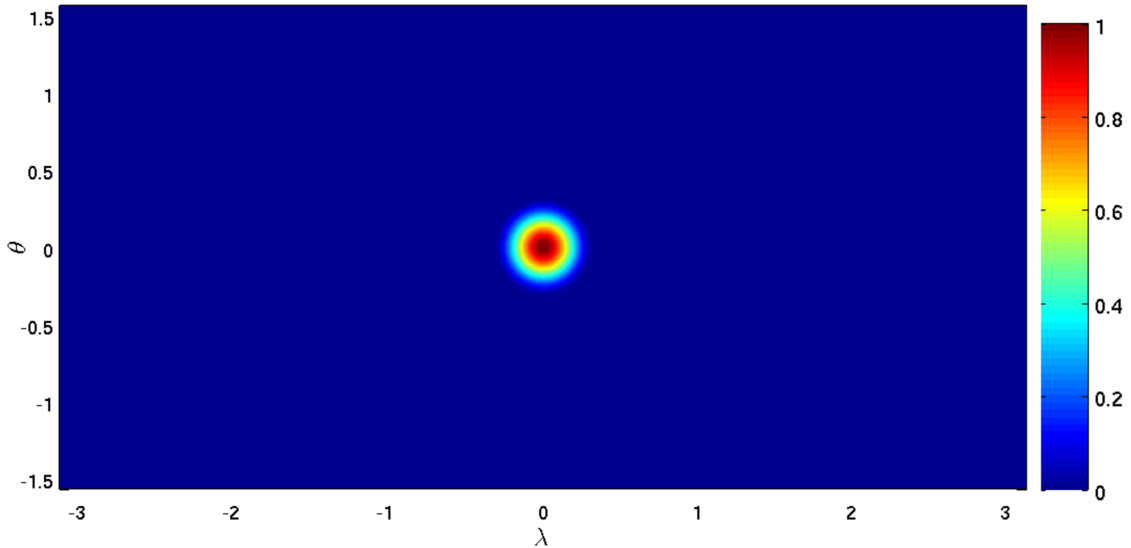
Numerical results: solid body rotation

Plots of the solution for N=12544, n=144, Δt=$2\pi/1600$, μ=8×10^{-9}:

Initial condition, t=0, with streamlines



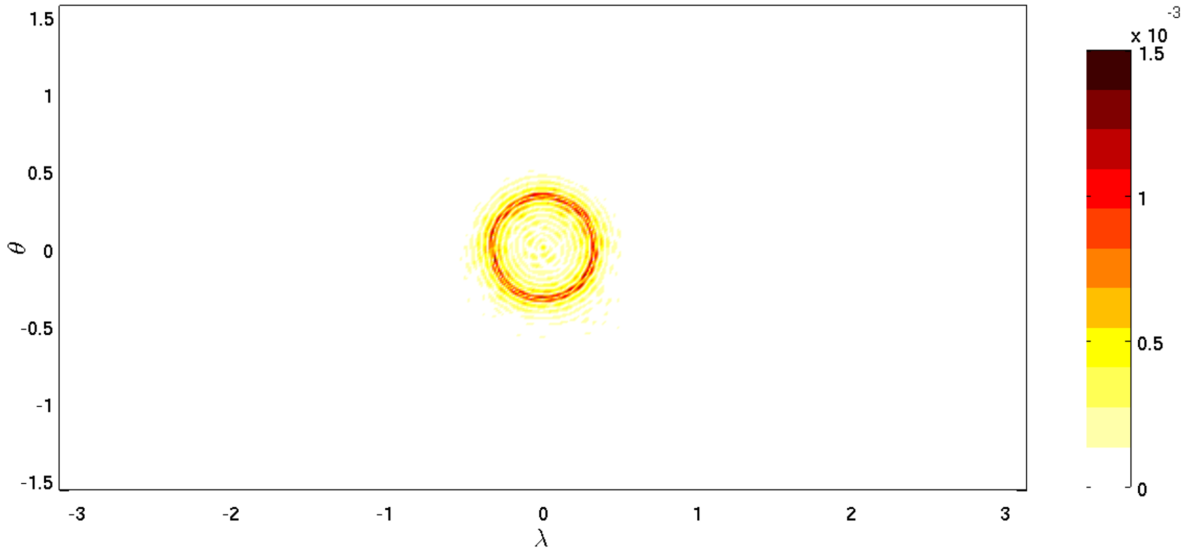
Solution after one revolution, t=2π



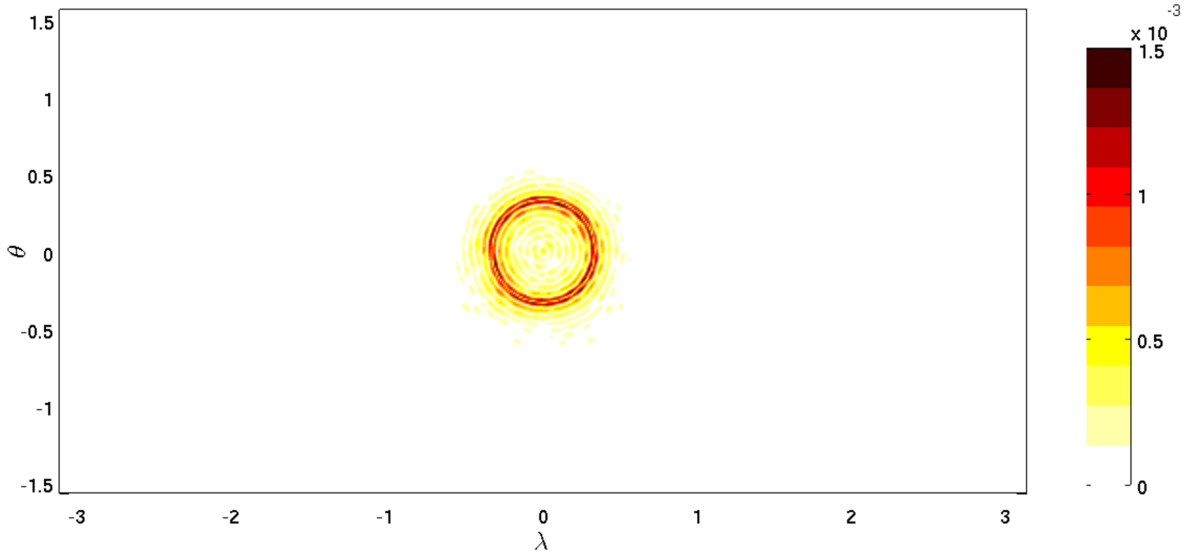
Numerical results: solid body rotation

Plots of the error for N=12544, n=144, $\Delta t = 2\pi/1600$, $\mu = 8\times10^{-9}$:

Magnitude of the error after one revolution



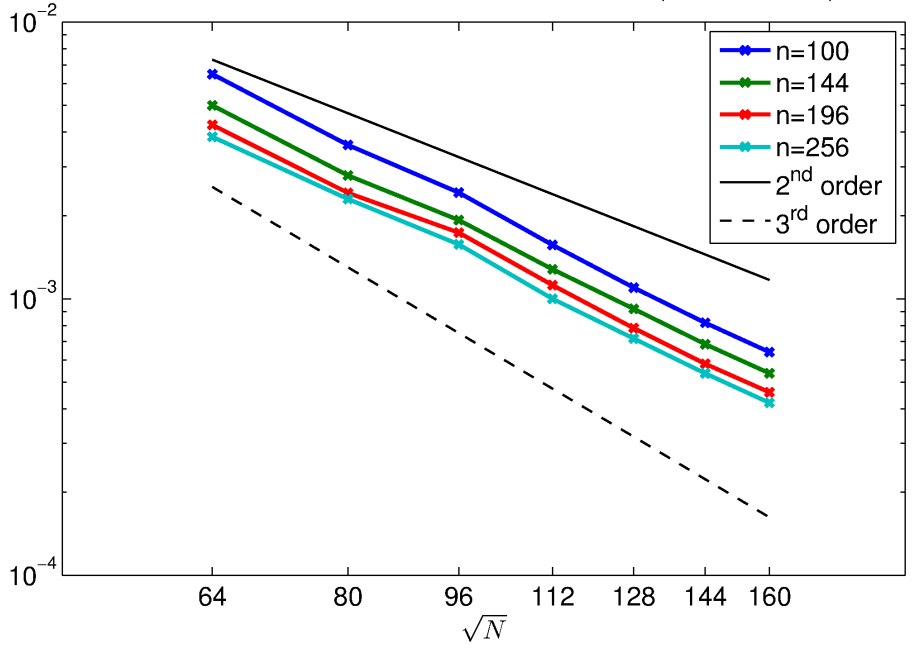
Magnitude of the error after ten revolutions



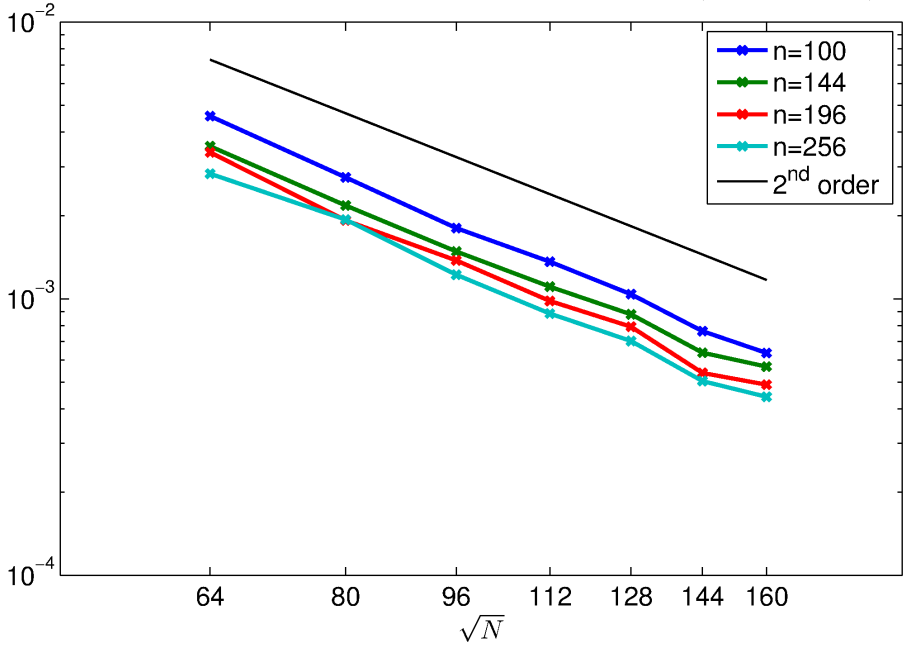
Numerical results: solid body rotation

Convergence plots for increasing N and n, $\Delta t = 2\pi/1600$, $\mu = 8\times10^{-9}$:

Relative ℓ_2 error vs. $\sqrt{N}$ (logscale)



Relative ℓ_∞ error vs. $\sqrt{N}$ (logscale)

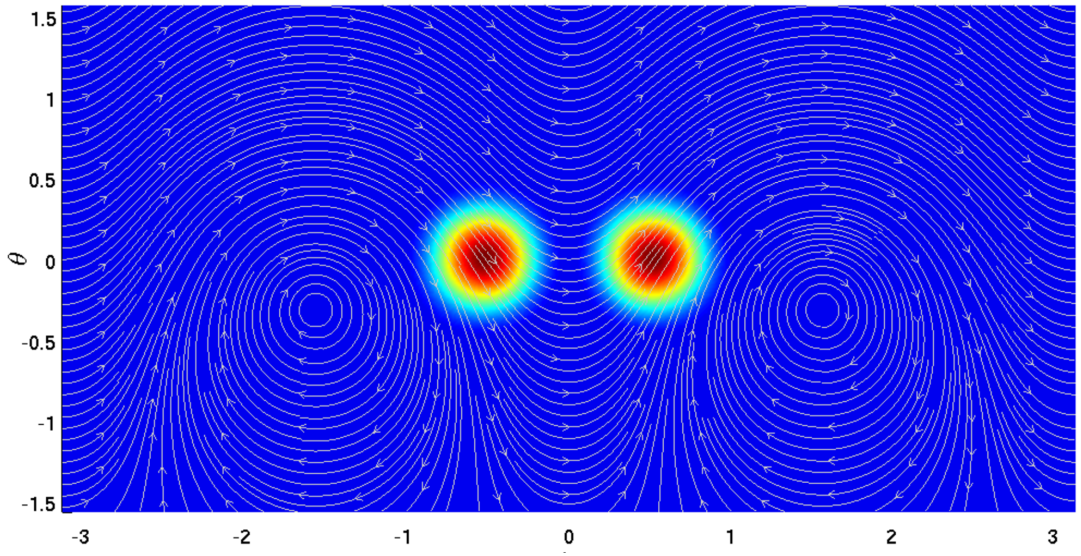


Convergence rates are as expected given smoothness of the initial condition.

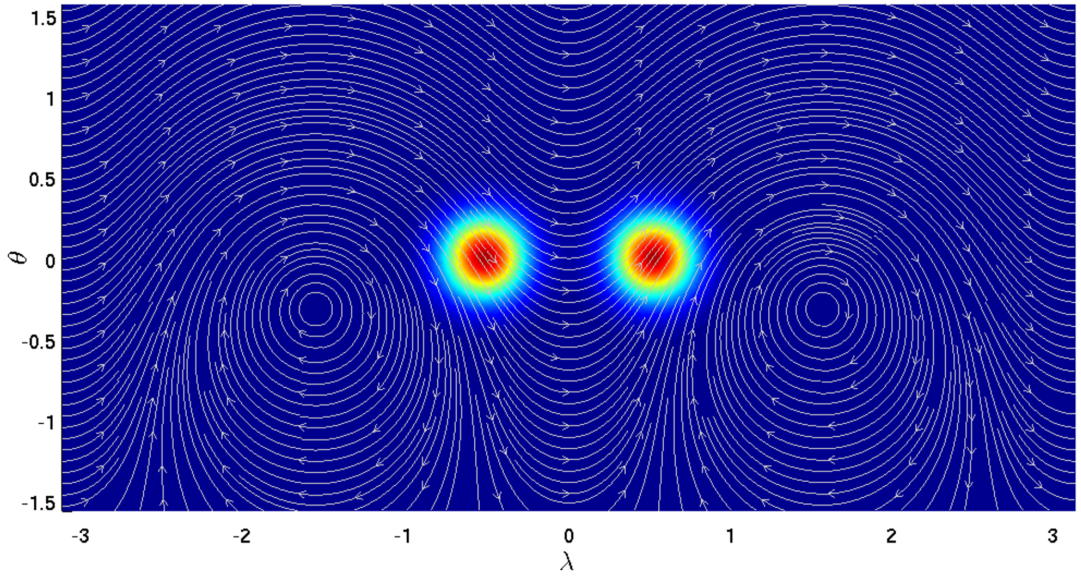
Numerical results: deformational flow

- Deformational/rotational flow (R.D. Nair and P.H. Lauritzen, JCP (2010))

Non-smooth initial condition:

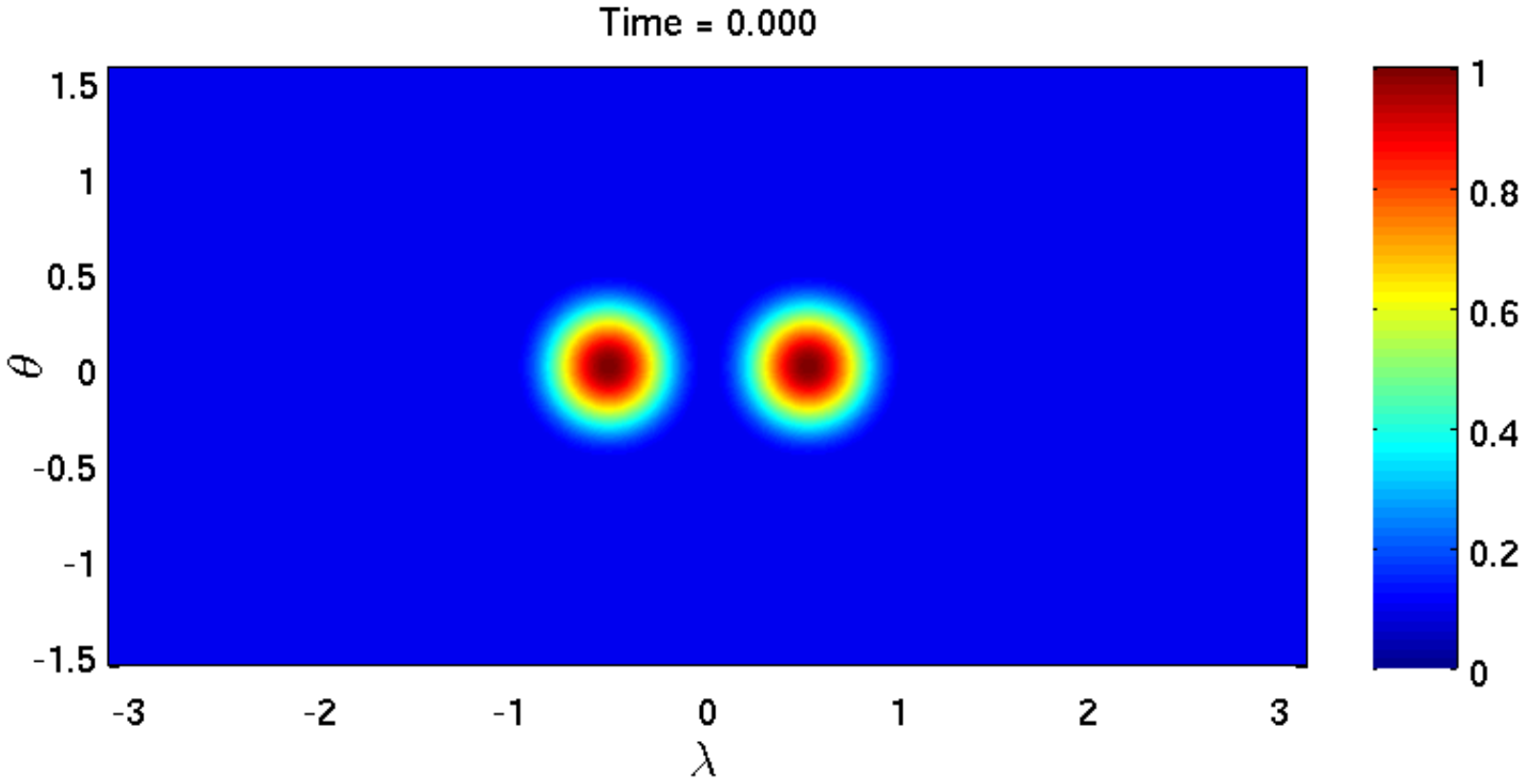


Smooth initial condition:



Numerical results: deformational flow

Simulation for non-smooth IC, N=20736, n=100, Δt=5/2400, μ=10^{-8}

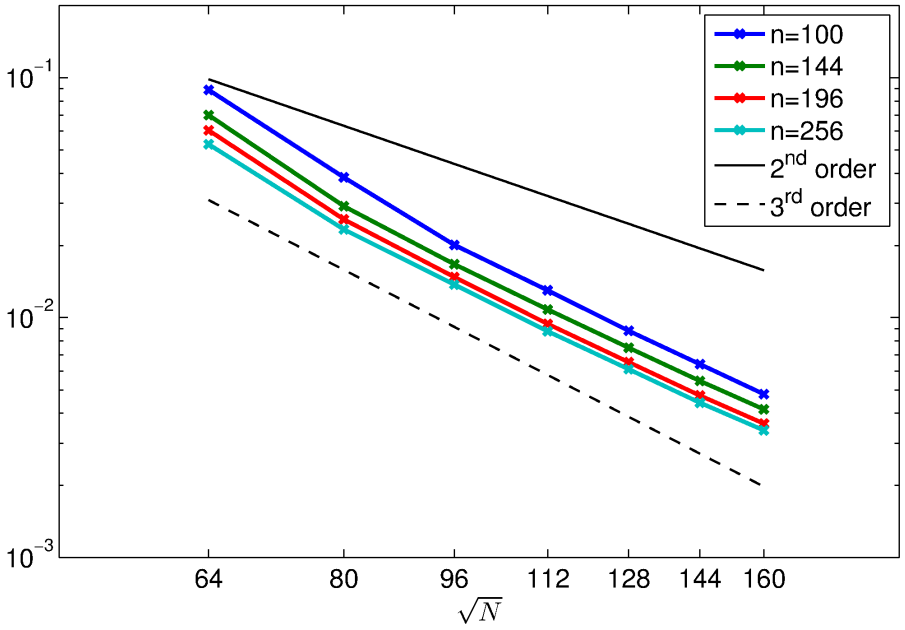


Numerical results: deformational flow

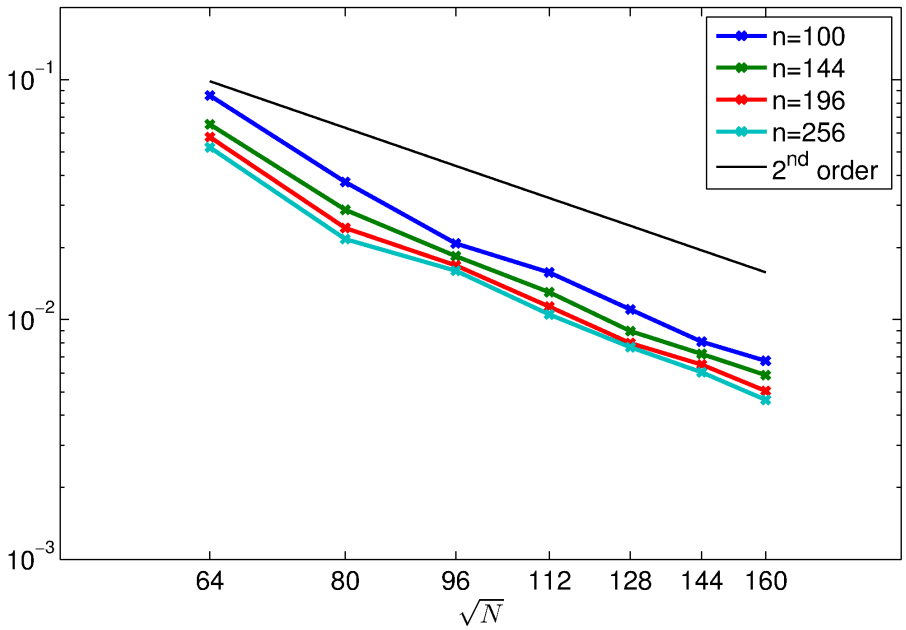
Convergence plots for increasing N and n, Δt=5/2400, μ=10^{-8}

- Non-smooth initial condition:

Relative ℓ_2 error vs. $\sqrt{N}$ (logscale)



Relative ℓ_∞ error vs. $\sqrt{N}$ (logscale)

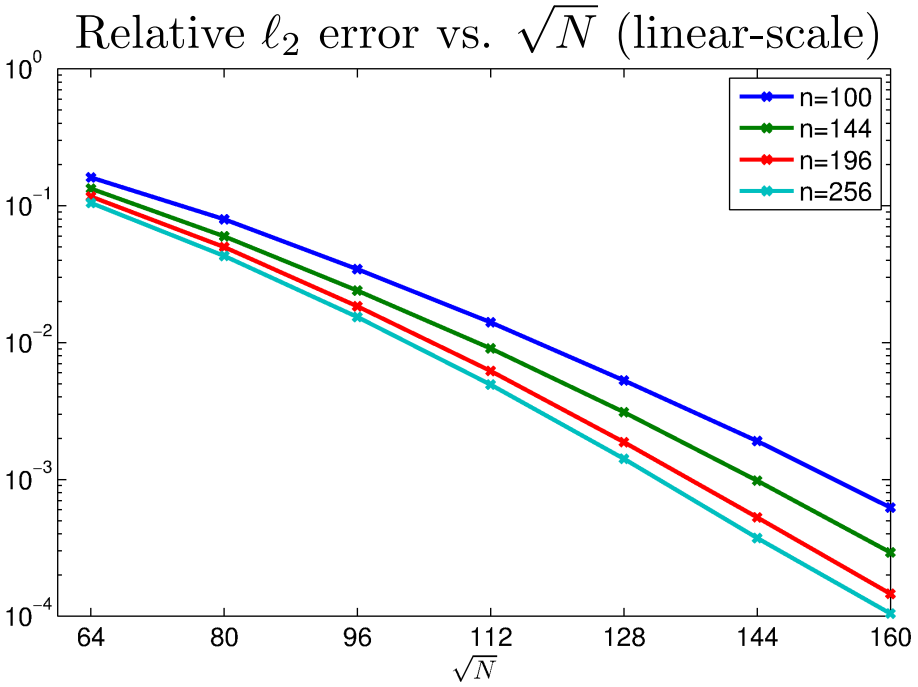


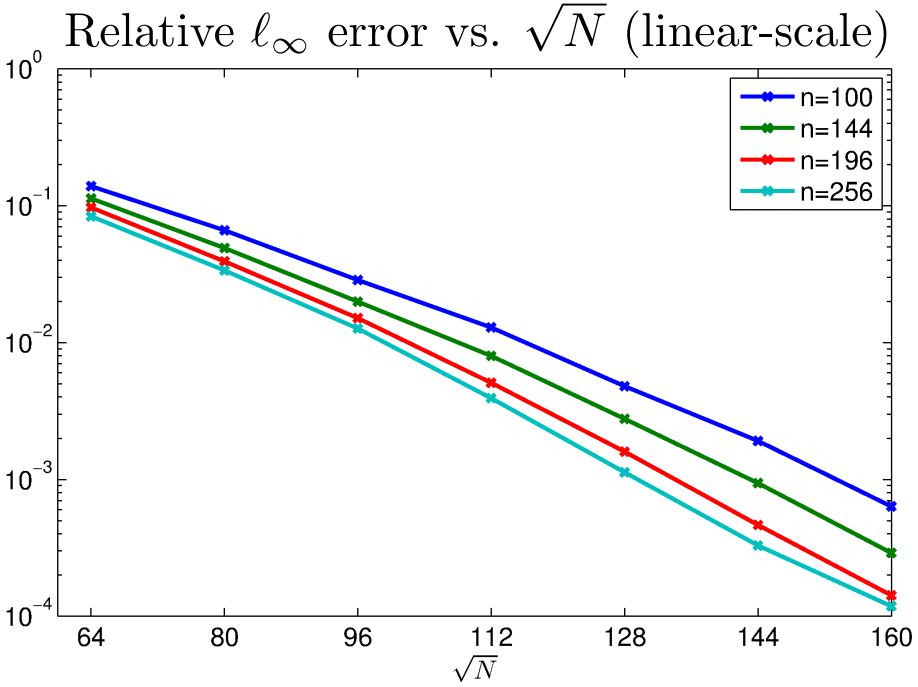
Convergence rates are as expected given smoothness of the initial condition.

Numerical results: deformational flow

Convergence plots for increasing N and n, Δt=5/2400, μ=10^{-8}

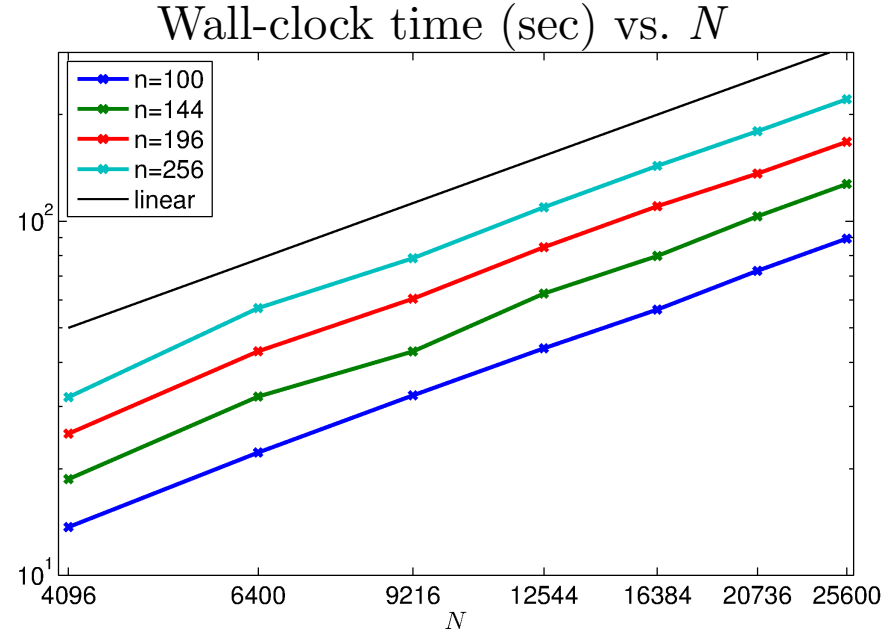
- Smooth initial condition:

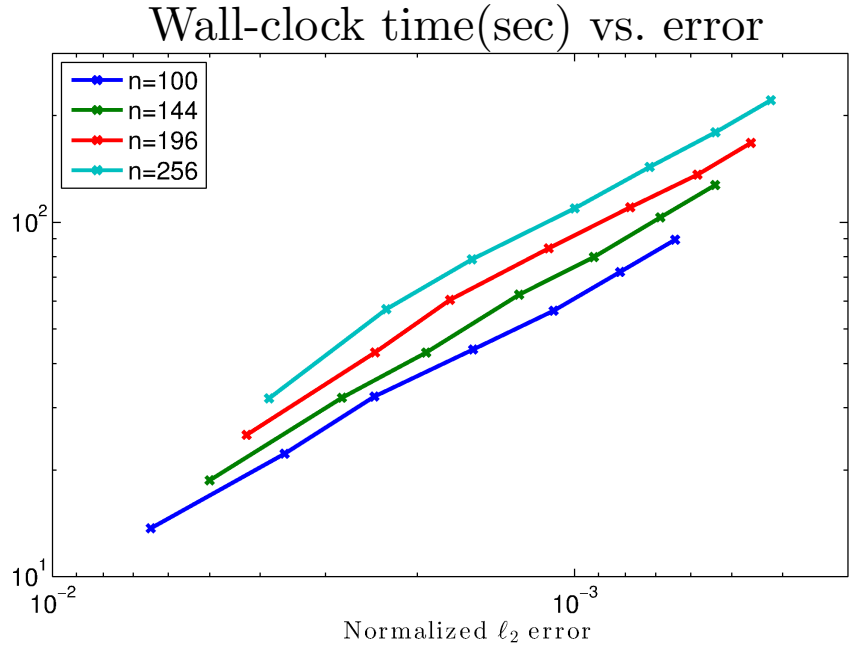


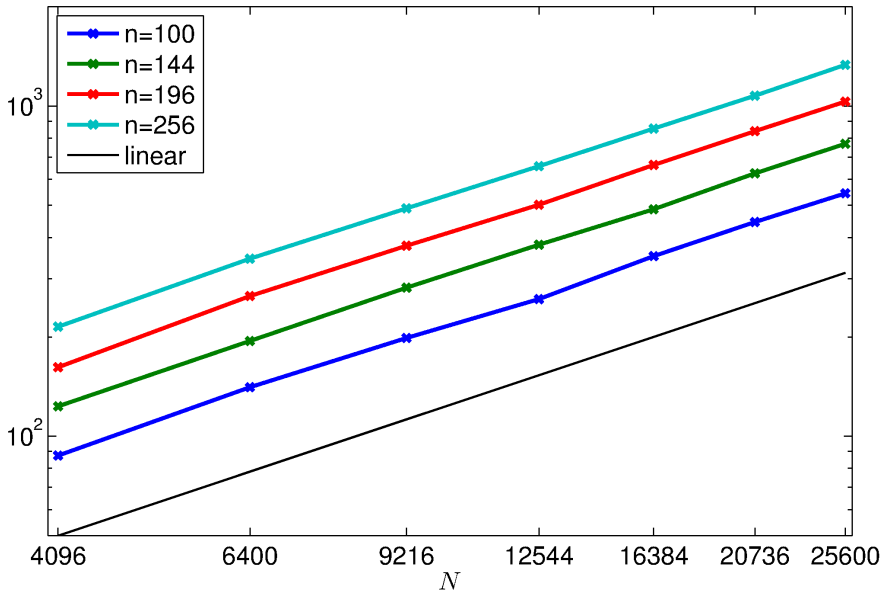


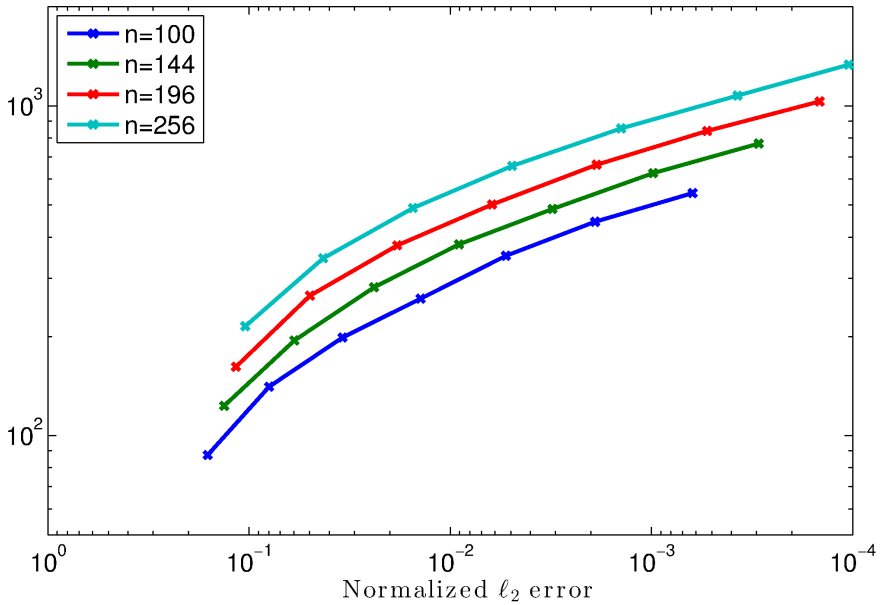
Observed convergence rate is spectral for the smooth initial condition.

Computational performance: both tests



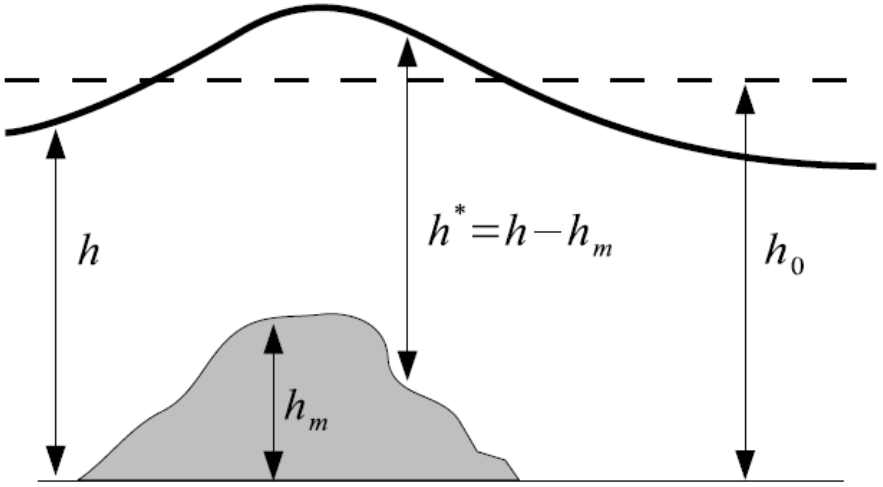






Full shallow water wave equations

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



$$\mathbf{u} = [u\ v\ w]^T$$

- Governing equations (in Cartesian form):

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{P}\begin{bmatrix} (\mathbf{u}\cdot\mathbf{P}\nabla)u + f(\mathbf{x}\times\mathbf{u})\cdot\hat{\mathbf{i}} + g(\mathbf{p}_x\cdot\nabla)h \\ (\mathbf{u}\cdot\mathbf{P}\nabla)v + f(\mathbf{x}\times\mathbf{u})\cdot\hat{\mathbf{j}} + g(\mathbf{p}_y\cdot\nabla)h \\ (\mathbf{u}\cdot\mathbf{P}\nabla)w + f(\mathbf{x}\times\mathbf{u})\cdot\hat{\mathbf{k}} + g(\mathbf{p}_z\cdot\nabla)h \end{bmatrix} = 0$$

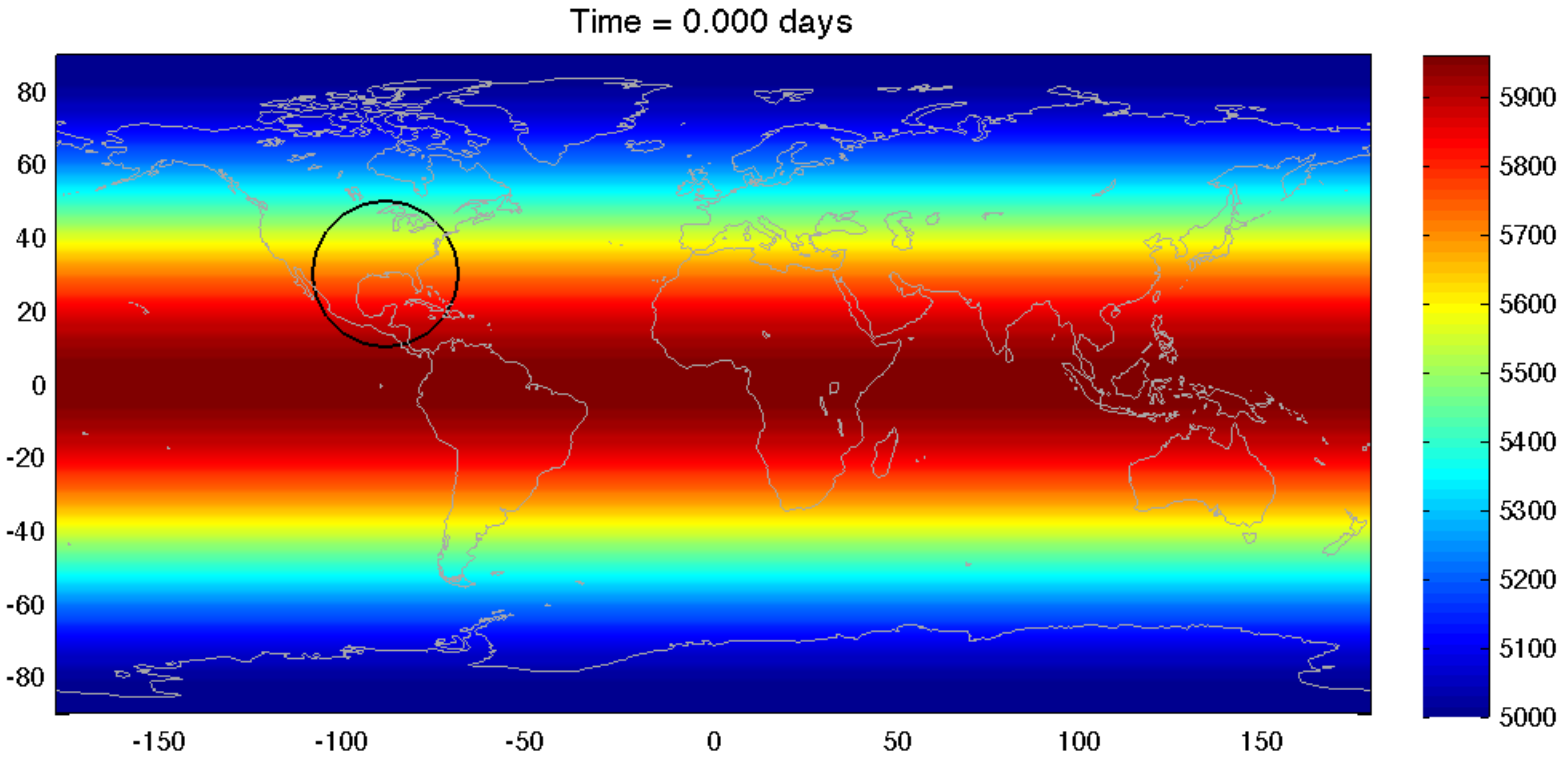
$$\frac{\partial h^*}{\partial t} + (\mathbf{P}\nabla)\cdot(h^*\mathbf{u}) = 0$$

Global RBF Model: Flyer & W, PRS-A (2009)

RBF-PUM for shallow water wave eqs.

- Numerical simulation: Flow over an isolated mountain

  (Test Case 5 from Williamson et. al., JCP (1992))



Concluding remarks

- Computational cost of the RBF-PUM scales linearly with N.
- Spectral-type convergence appears to be possible for smooth solutions.
- Hyperviscosity procedure seems to be effective for stabilizing the method.

Plans for the future:

- Comparison with the RBF finite difference (RBF-FD) method.
- Develop technique to handle non-uniform node sets.
- Explore ways to reduce overlap (q) and still maintain stability.
- Parallelization of the method.
- Implementation of stable *flat* RBF algorithms.
- Further comparisons for shallow water wave equations.