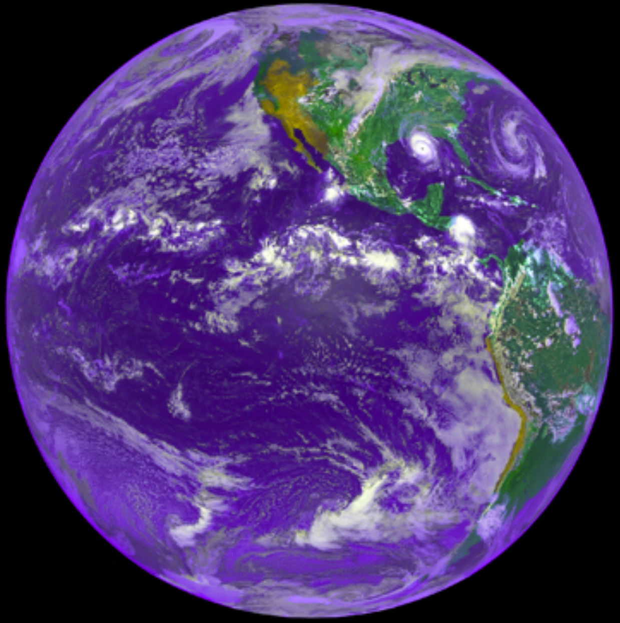
2013 Dolomites Research Week on Approximation

Lecture 4: Global and local kernel methods for approximating derivatives on the sphere



Grady B. Wright
Boise State University

Overview

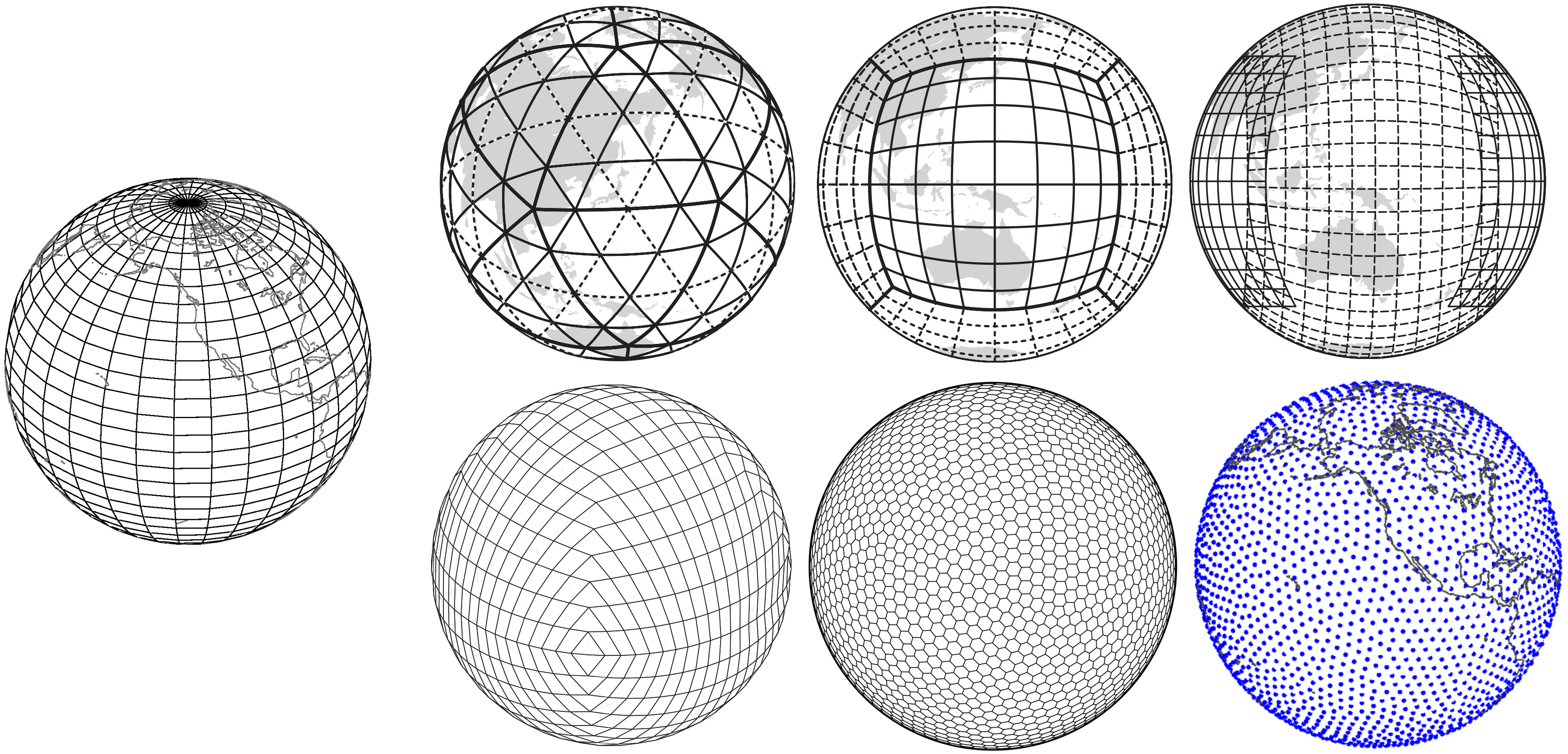
- Methods currently in use for computational geosciences
- Transport equation on the sphere
- Global method RBF collocation method
    - Numerical examples and comparisons to other methods
- RBF finite difference method (RBF-FDM)
    - Numerical examples
- RBF partition of unity method (RBF-PUM)
    - Numerical examples and comparison to RBF-FDM
- Concluding remarks

Bottom line for numerical methods:

Need numerical methods that provide high-resolution and accuracy at low computational costs to resolve the multi-scale features.

Grids, meshes, nodes, used in large scale models

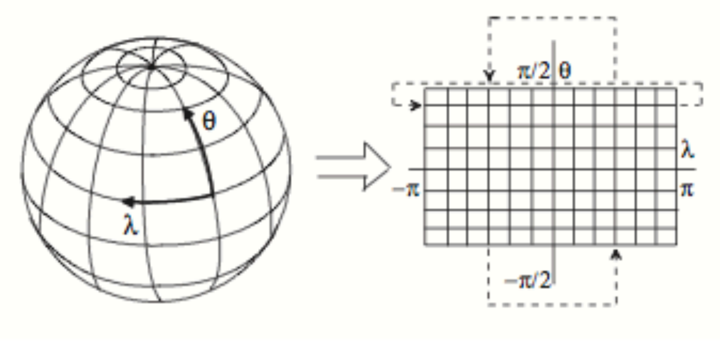
- Grids/meshes/nodes used used in methods for large-scale applications:



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

Highlights of some high-order methods in large-scale models

Double Fourier series:



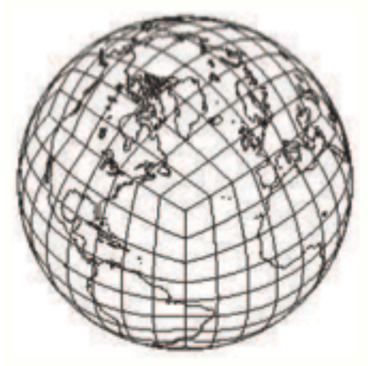
Strength: Exponential accuracy
Computationally fast because of FFTs

Weakness: No practical option for local mesh refinement

Spherical harmonics:

Strength: Exponential accuracy

Weakness: No practical option for local mesh refinement
Relatively high computational cost
Poor scalability on parallel computer architectures



Spectral elements:

Strength: Accuracy approaching exponential
Local refinement is feasible but complex

Weakness: Loss of efficiency due to unphysical element boundaries
(Runge phenomenon - oscillations near boundaries $\rightarrow$ restrictive time-step)
High algorithmic complexity
High pre-processing cost

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

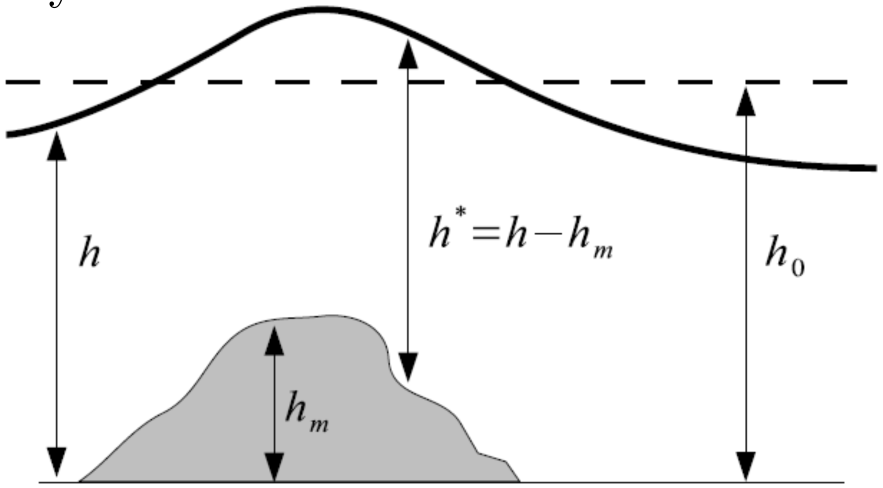
	Spherical Coords.	**Cartesian Coords.**
Point:	$(\lambda, \theta, 1)$	(x, y, z)
Unit vectors:	$\hat{\mathbf{i}} =$ longitudinal $\hat{\mathbf{j}} =$ latitudinal $\hat{\mathbf{k}} =$ radial	$\hat{\mathbf{i}} = x$-direction $\hat{\mathbf{j}} = y$-direction $\hat{\mathbf{k}} = z$-direction
Unit tangent 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:

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

Shallow water wave equations on a rotating sphere

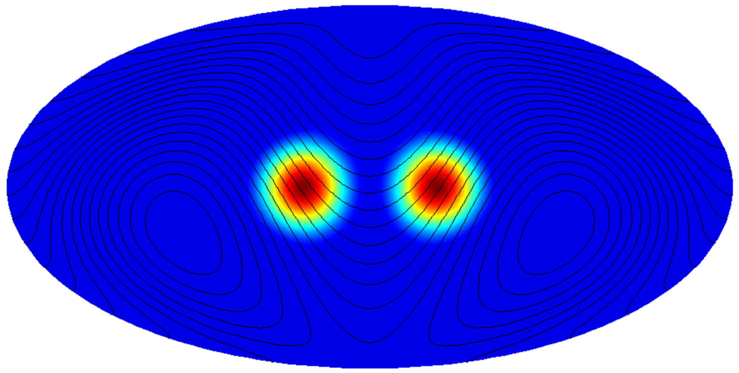
- Model for the nonlinear dynamics of a shallow, hydrostatic, homogeneous, and inviscid fluid layer.



- 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!	

Simpler problem: transport equation on the sphere



- For this tutorial we focus on the transport equation for 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:

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

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

- Surface gradient operator:

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

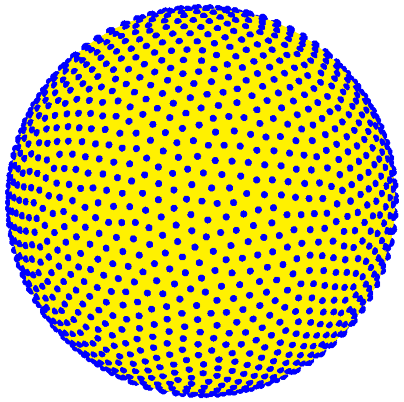
- Goal: Show how to construct good numerical approximations to

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

Surface gradient approximation: Global RBF method

- 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:

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

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

Surface gradient approximation: Global RBF method

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

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

- 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 fo $(\mathbf{p}_y \cdot \nabla)$ and $(\mathbf{p}_z \cdot \nabla)$.

Global RBF collocation for transport equation

- Continuous transport equation for some $\mathbf{u} = (u, v, w) \in T_X\mathbb{S}^2$:

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

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

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

- 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

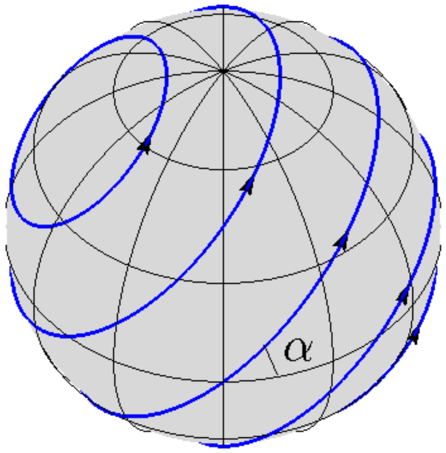
- Solid body rotation of a non-smooth cosine bell
  (Williamson et. al. JCP (1992))

  Stream Function for flow

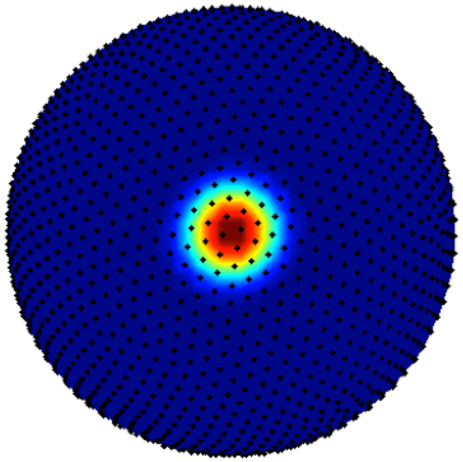
$$\psi(\mathbf{x}) = \cos(\alpha)z + \sin(\alpha)y \qquad \alpha = \pi/2 \text{ (flow over the poles)}$$

  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

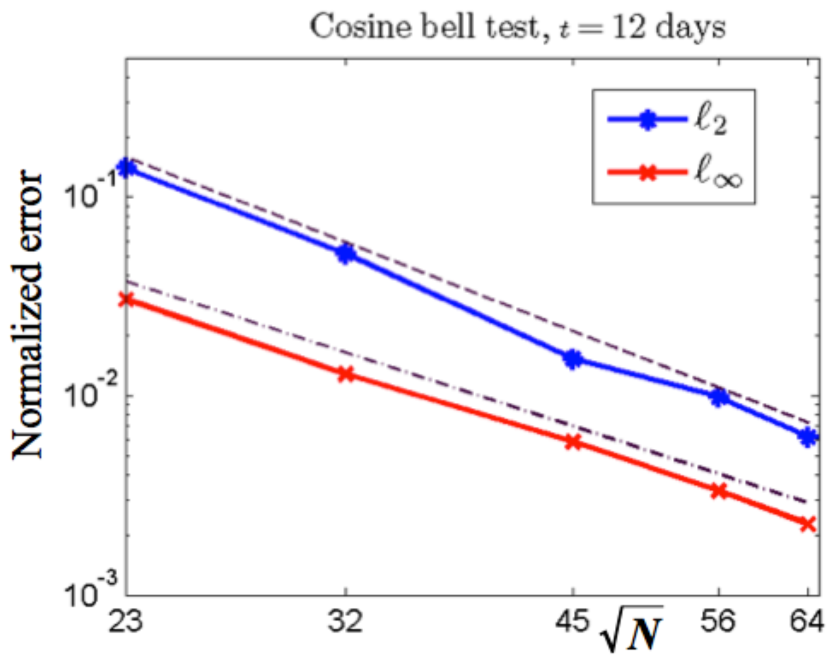
Details:

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

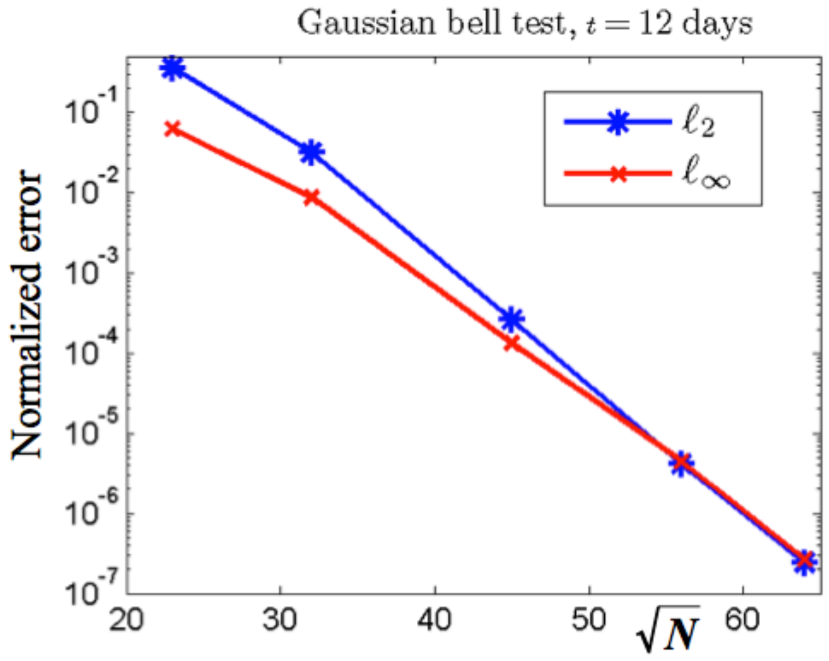
Numerical results: solid body rotation

- 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



Gaussian bell IC,
Infinitely smooth



log-log scale

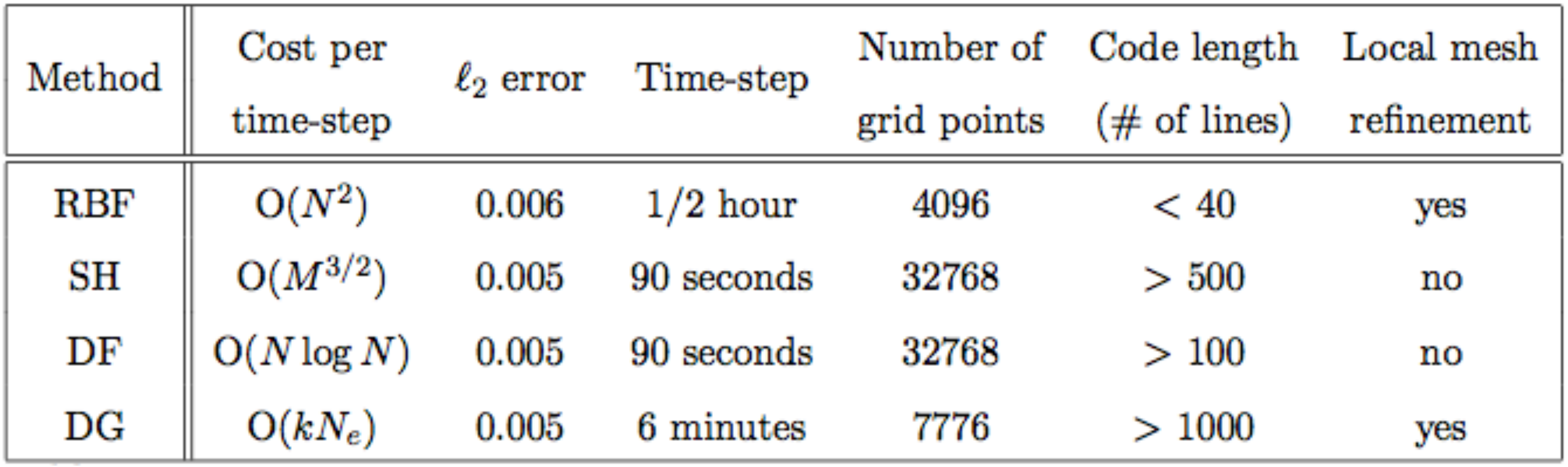
Straight line indicates algebraic accuracy

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

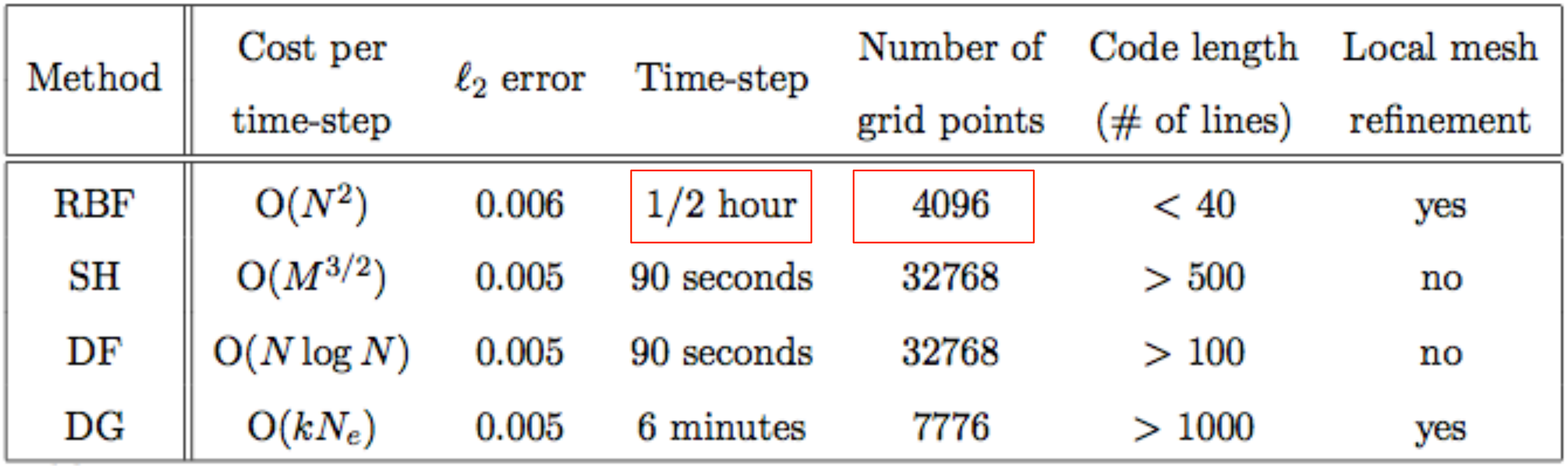
RBF=radial basis functions, SH=spherical harmonics, DF=double Fourier, DG=discontinuous Galerkin spectral elements

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	$\boxed{\text{1/2 hour}}$	$\boxed{4096}$	< 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

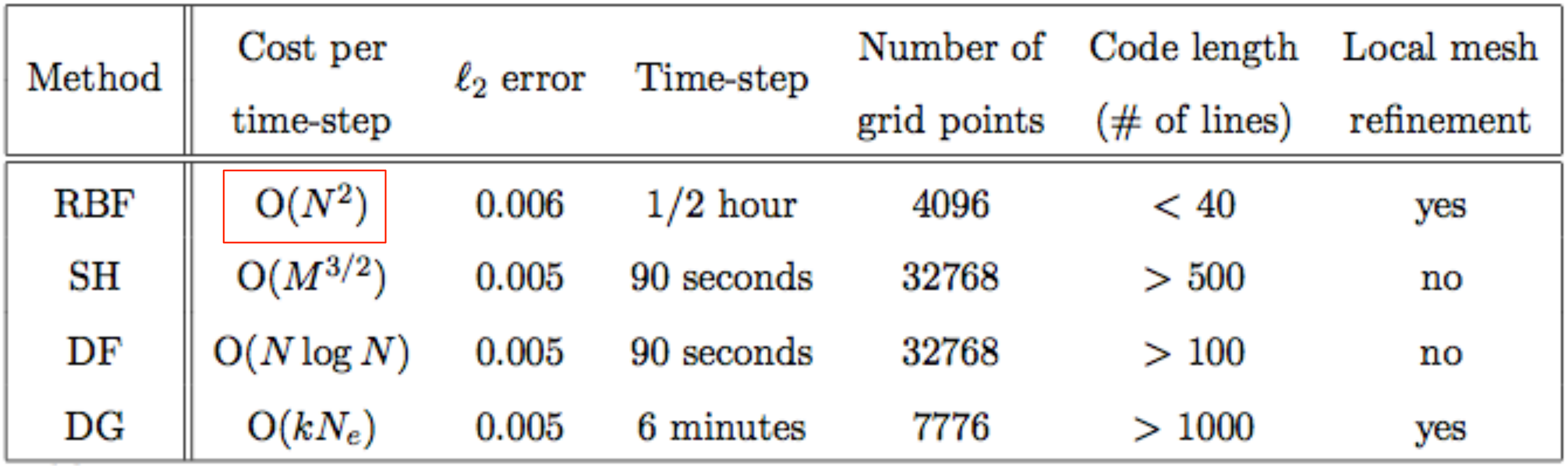
RBF=radial basis functions, SH=spherical harmonics, DF=double Fourier, DG=discontinuous Galerkin spectral elements

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	$\boxed{O(N^2)}$	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

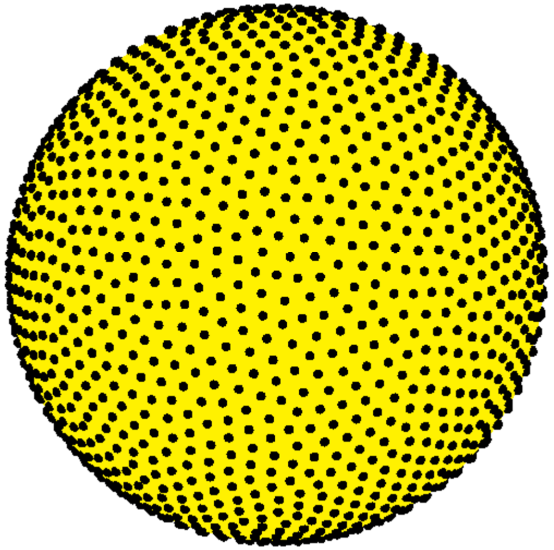
RBF=radial basis functions, SH=spherical harmonics, DF=double Fourier, DG=discontinuous Galerkin spectral elements

- Need ways to reduce this cost.

- Next two methods we discussed are focused on this

RBF generated finite differences (RBF-FD)

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

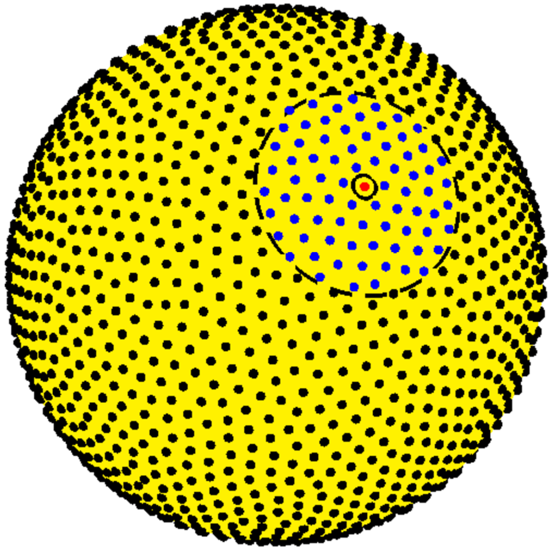


- Generalization of finite-difference (FD) method to scattered nodes using RBFs to compute the FD weights.

- 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:

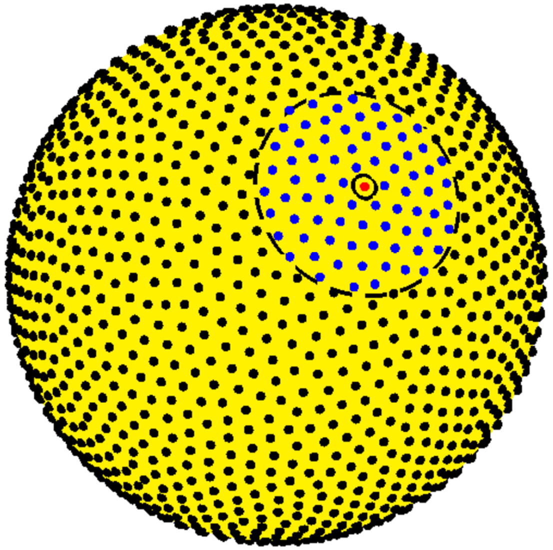
1. For each node $\mathbf{x}_j$, choose $n-1$ of it's nearest neighbors: $X_j = \{\mathbf{x}_i\}_{i=1}^n$, with $\mathbf{x}_1 = \mathbf{x}_j$.

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:

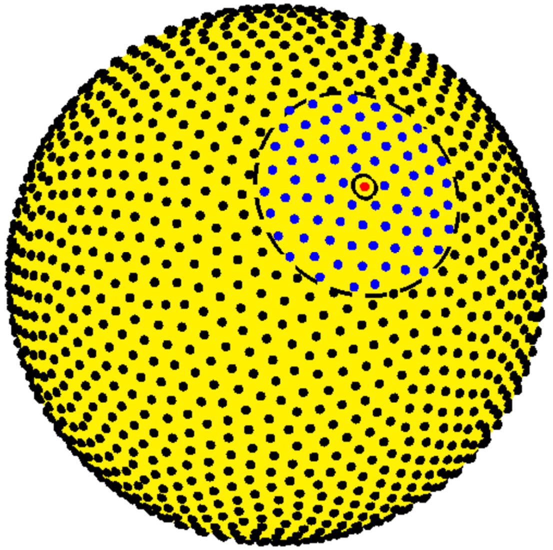
3. Choose the weights c_i such the approximation is exact for $\{\phi(\|\mathbf{x} - \mathbf{x}_k\|)\}_{k=1}^{n}$:

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

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:

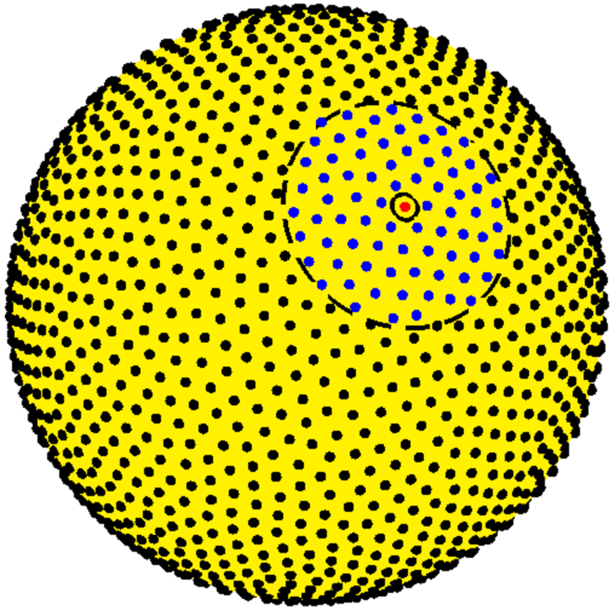
3. The weights $\{c_i\}_{i=1}^{n}$ can be computed by solving:

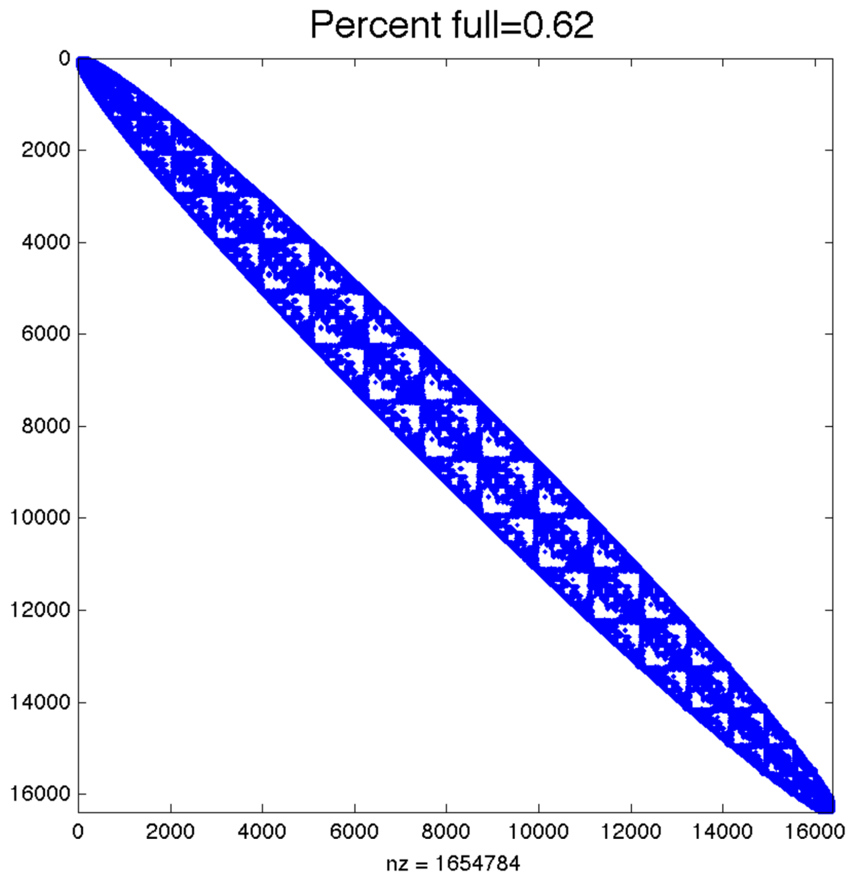
$$\begin{bmatrix} \phi(\|\mathbf{x}_1-\mathbf{x}_1\|) & \phi(\|\mathbf{x}_1-\mathbf{x}_2\|) \cdots & \phi(\|\mathbf{x}_1-\mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_2-\mathbf{x}_1\|) & \phi(\|\mathbf{x}_2-\mathbf{x}_2\|) \cdots & \phi(\|\mathbf{x}_2-\mathbf{x}_N\|) \\ \vdots & \vdots \quad \ddots & \vdots \\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

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

- Continuous transport equation for some $\mathbf{u} = (u, v, w) \in T_X\mathbb{S}^2$:

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

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

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

- Advance the system in time using some standard ODE solver.
- This is a purely hyperbolic problem and temporal stability is 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

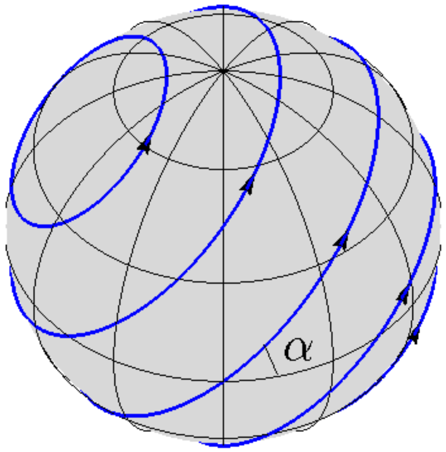
  (Williamson et. al. JCP (1992))

  Stream Function for flow

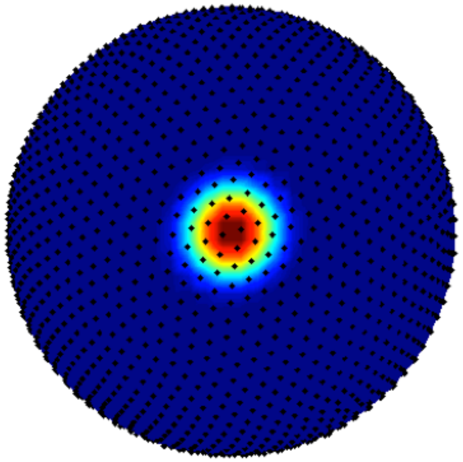
$$\psi(\mathbf{x}) = \cos(\alpha)z + \sin(\alpha)y \qquad \alpha = \pi/2 \text{ (flow over the poles)}$$

  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



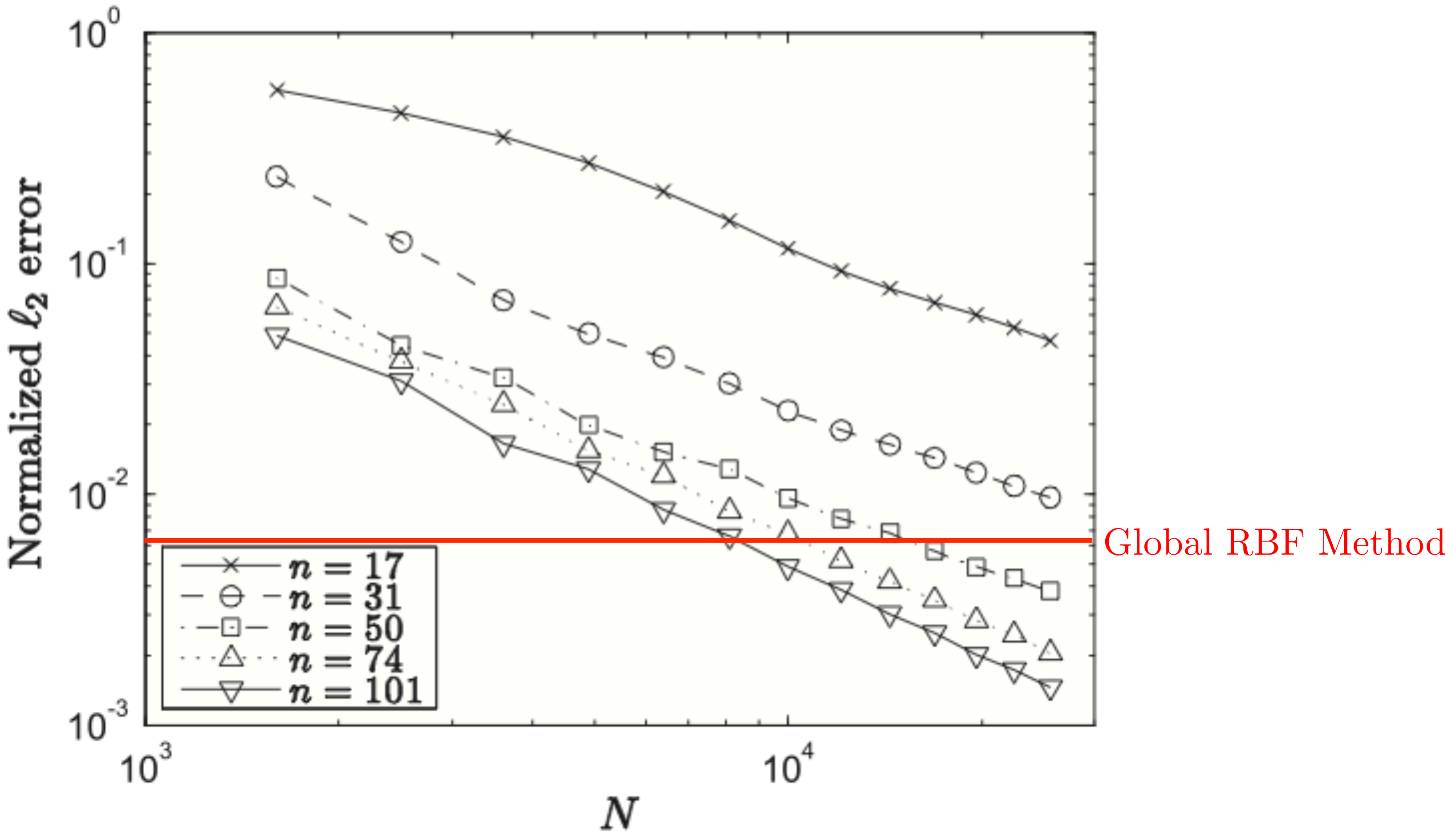
Initial condition

Details:

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

Numerical results: solid body rotation

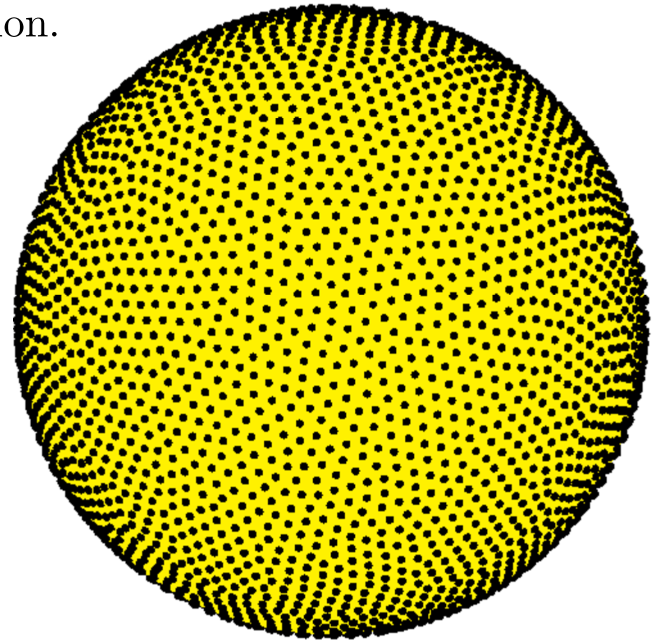
- Convergence results as number of nodes N increases (Fornberg & Lehto, 2011)
- Error results are for 10 complete revolution of the cosine bell.



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

RBFs and partition-of-unity (RBF-PUM) on the sphere

- Recent work with my graduate student Kevin Aiton.
  G.B. Wright & K. Aiton. A radial basis function partition of unity method for transport on the sphere. In preparation.

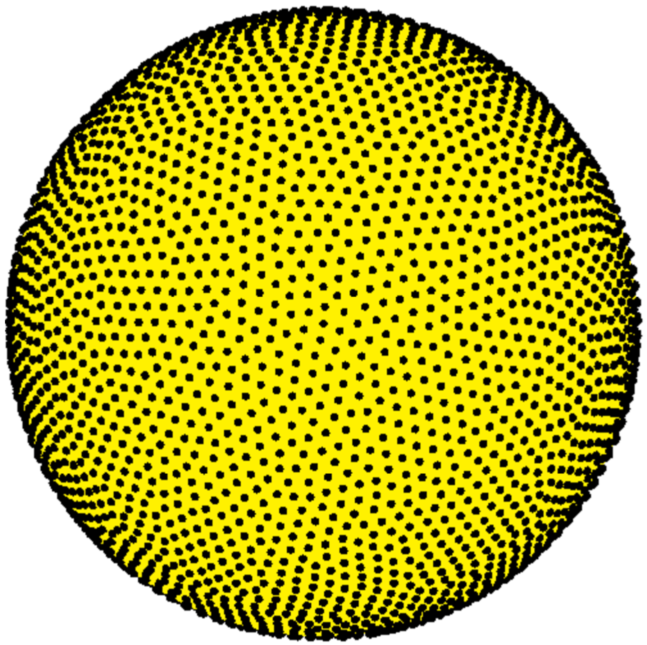


<u>Background references for interpolation with RBF-PUM</u>

- R. Cavoretto & A. DeRossi, Fast and accurate interpolation of large scattered data sets on the sphere. *J. Comput. Appl. Math.* 234 (2010), 1505–1521.
- R. Cavoretto & A. DeRossi, Spherical interpolation using the partition of unity method: An efficient and flexible algorithm. *Appl. Math. Lett.* 25 (2012), 1251-1256.

RBFs and partition-of-unity on the sphere

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

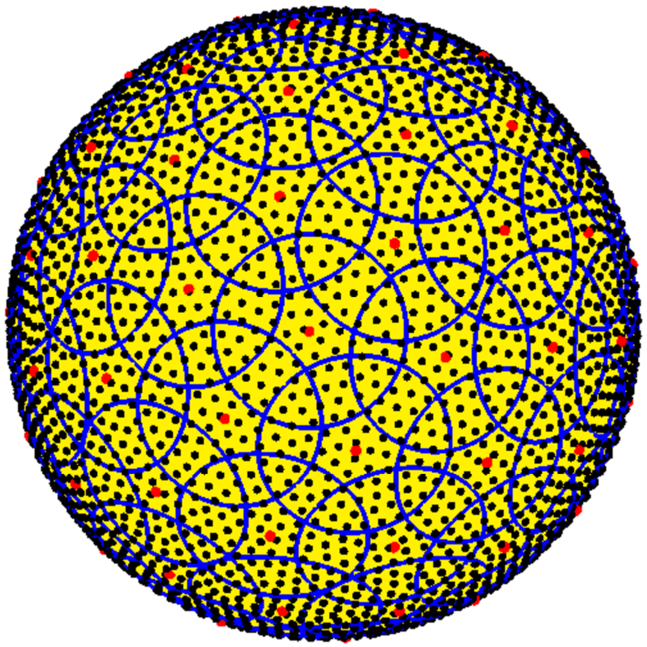


Key Steps:

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

RBFs and partition-of-unity on the sphere

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



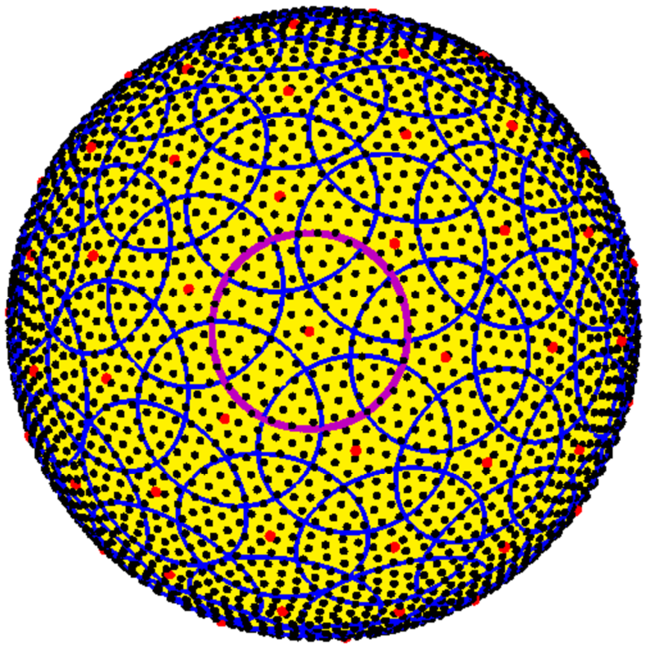
M total patches
n nodes per patch
ξ_k = center of patch Ω_k

Key Steps:

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

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

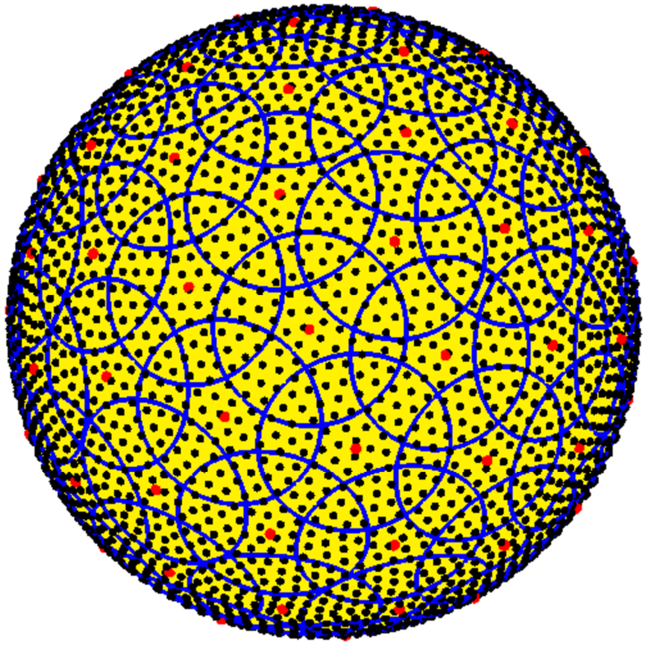
Key Steps:

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

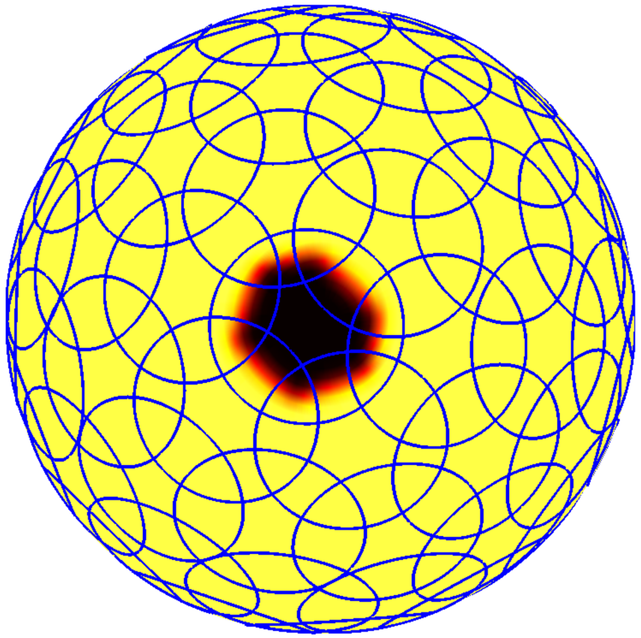
Key Steps:

3. Define weight functions $w_k : \mathbb{S}^2 \to \mathbb{R}$, $k = 1\ldots, M$, 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:

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

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

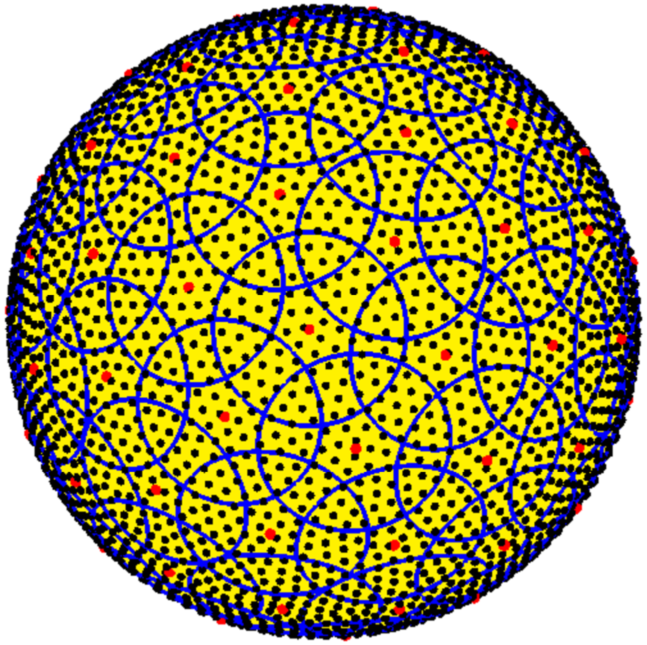
<u>Key Steps:</u>

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:

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

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
$\xi_k =$ center of patches Ω_k

Key Steps:

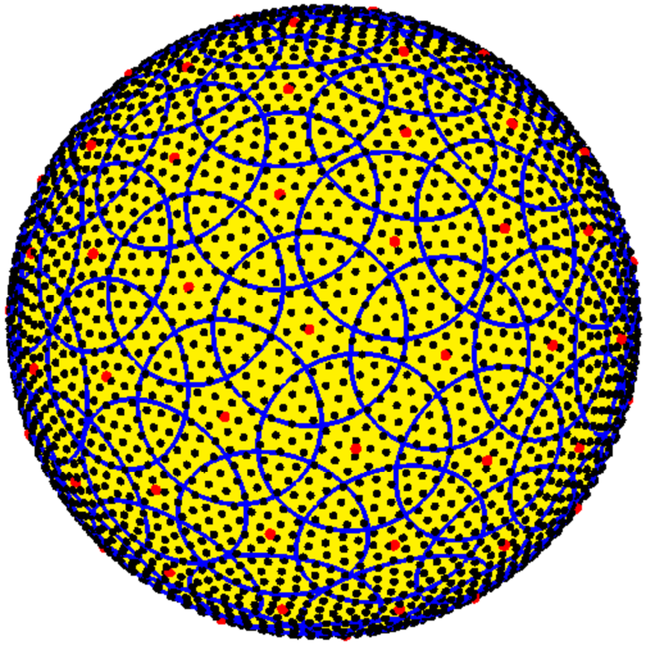
Weight function details:

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

$\rho_k =$ radius of patch Ω_k
ψ has compact support over $[0, 1]$

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

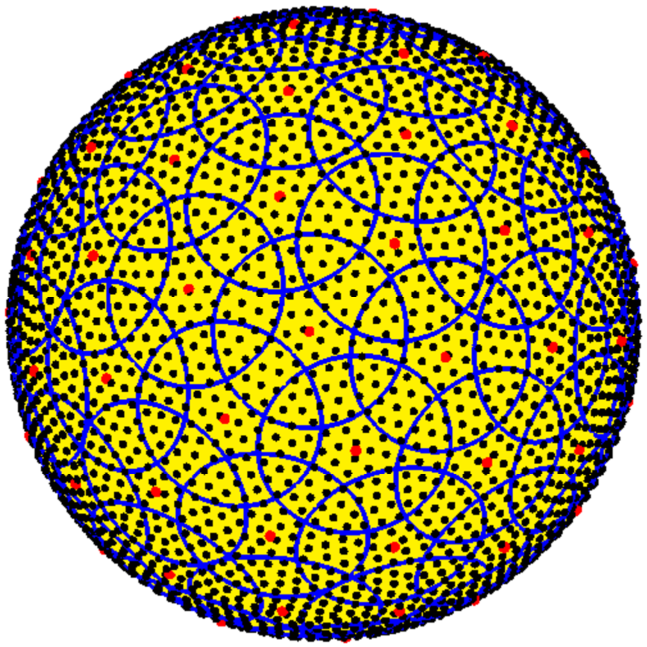
Key Steps:

4. Create a global interpolant for X as

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

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:

5. Apply projected gradient operator $\mathcal{D}_x := \mathbf{p}_x \cdot \nabla$ to interpolant and evaluate at each node $\mathbf{x}_j$:

$$\mathcal{D}_x s(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_j} = \sum_{k=1}^{M} \mathcal{D}_x \left(w_k(\mathbf{x}) s_k(\mathbf{x})\right)\Big|_{\mathbf{x}=\mathbf{x}_j}$$

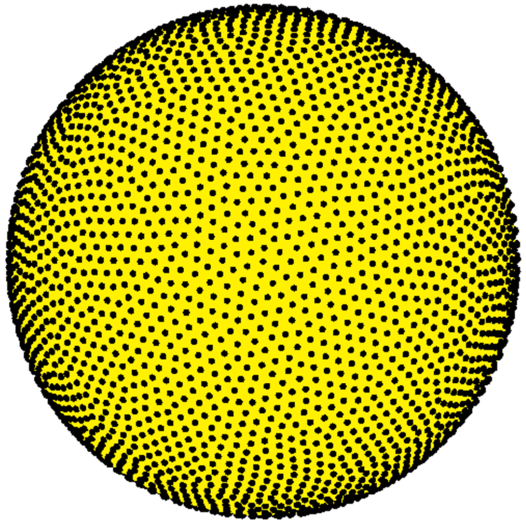
Weights can be generated and stored in a differentiation matrix.

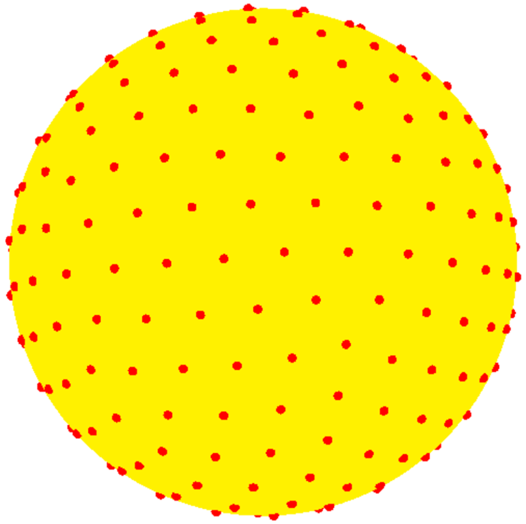
Choosing the nodes and patches

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



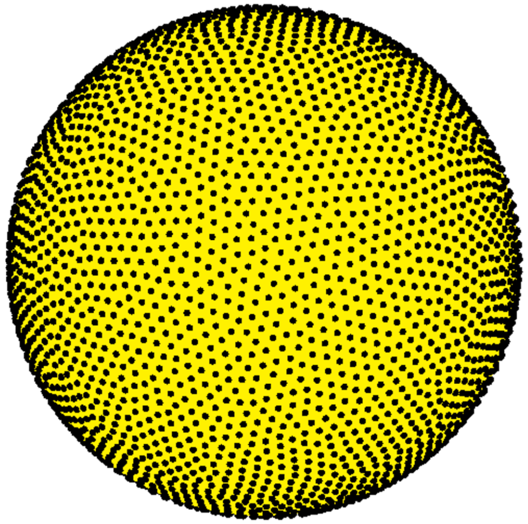


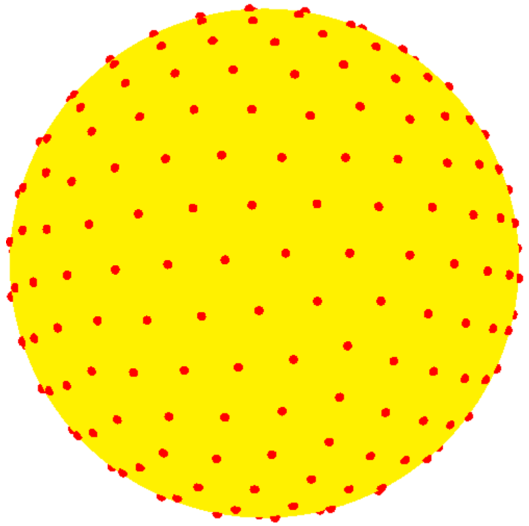
Patch centers

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:

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

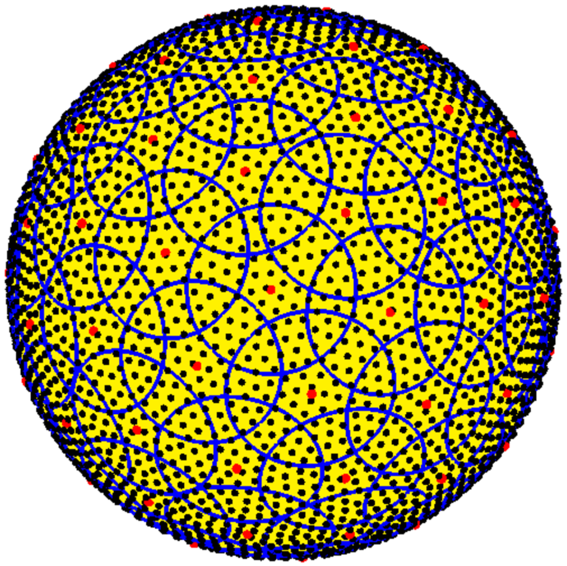
- The overlap parameter q determines the **average number of patches a node belongs to**, and satisfies the relationship:

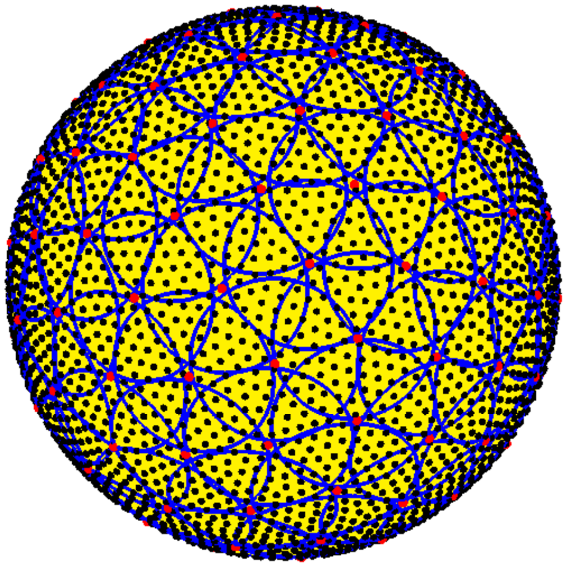
$$\frac{4\pi}{M} \approx \frac{\pi\rho^2}{q} \quad \Longrightarrow M = \left\lceil q\frac{N}{n} \right\rceil$$

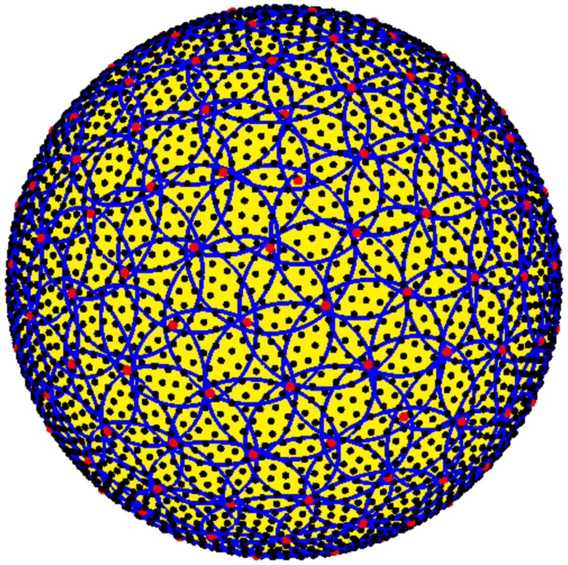
Choosing the nodes and patches

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

q=2 | q=3 | q=4


M=82

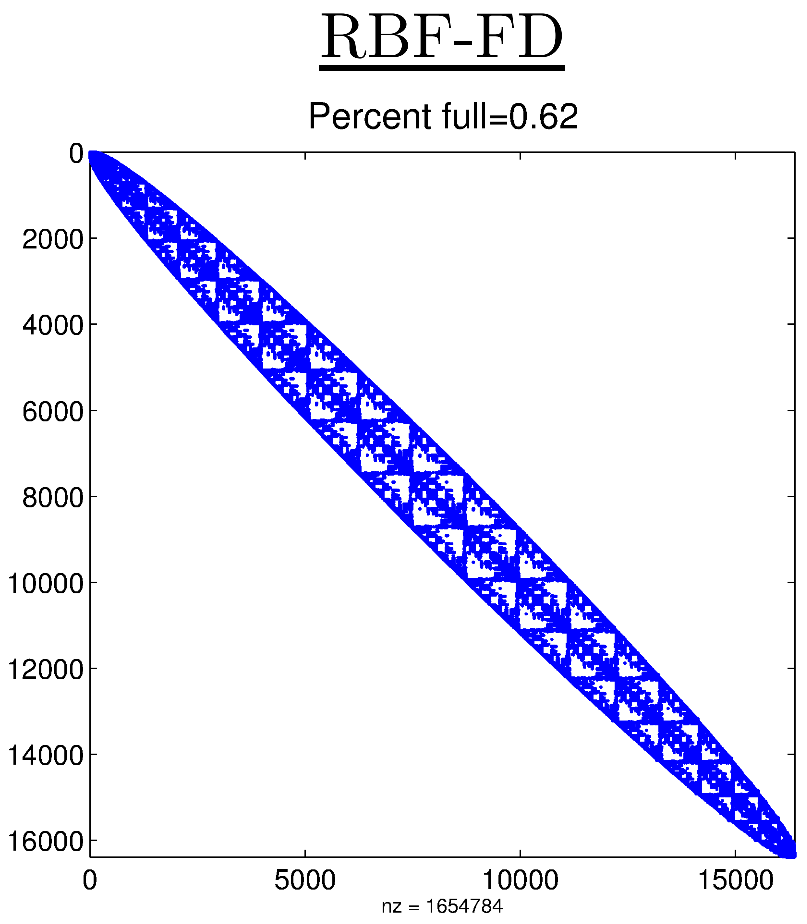

M=123

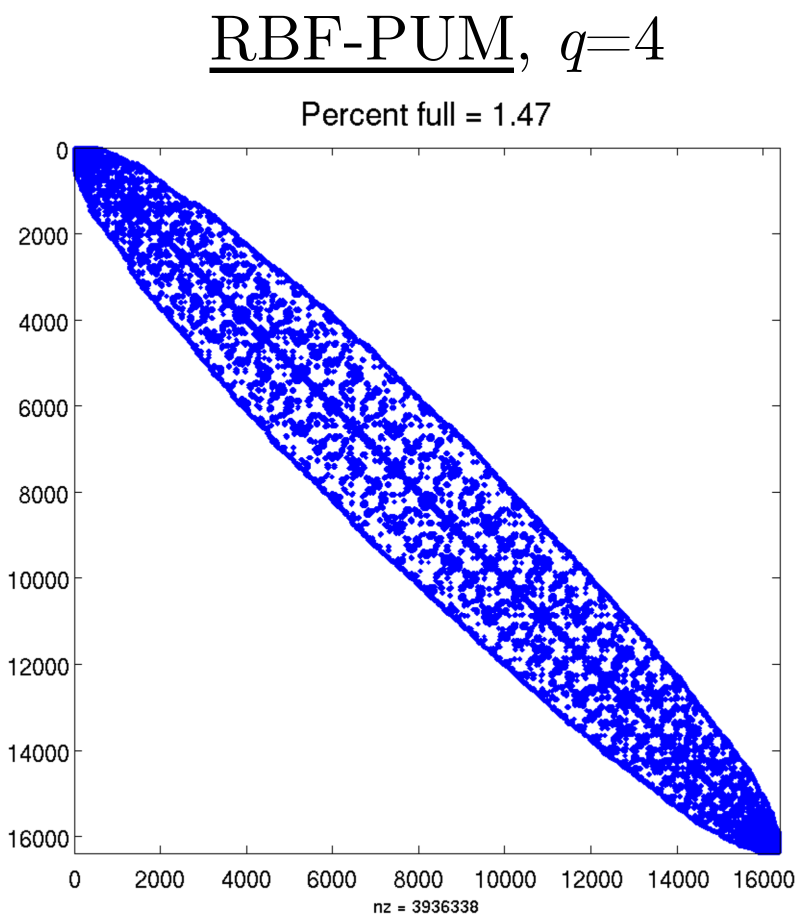

M=184

The computational cost for evaluating a derivative grows at most linearly with q and n.

Comparison: RBF-FD and RBF-PUM

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





- Recall global RBF-type methods result in dense matrices.

RBF-PUM method for transport equation

- Continuous transport equation for some $\mathbf{u} = (u, v, w) \in T_X\mathbb{S}^2$:

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

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

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

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

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

  - L_N is a like a discrete approximation to a high power of the Laplacian.

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)

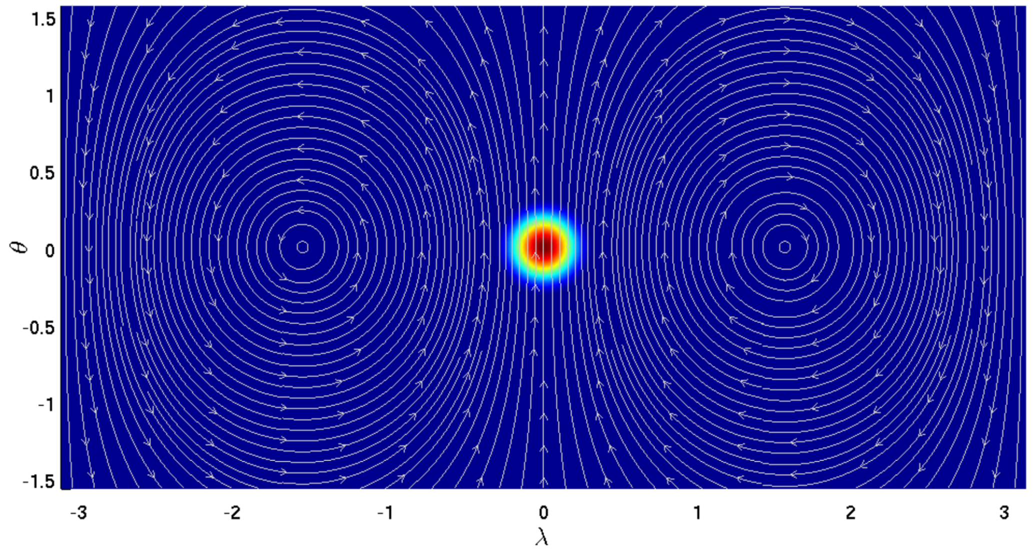
$$h(\mathbf{x}) = \begin{cases} \frac{1}{2}\left(1+\cos(3\pi r(\mathbf{x}))\right) & r(\mathbf{x}) < 1/3 \\ 0 & r(\mathbf{x}) \geq 1/3 \end{cases}$$

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

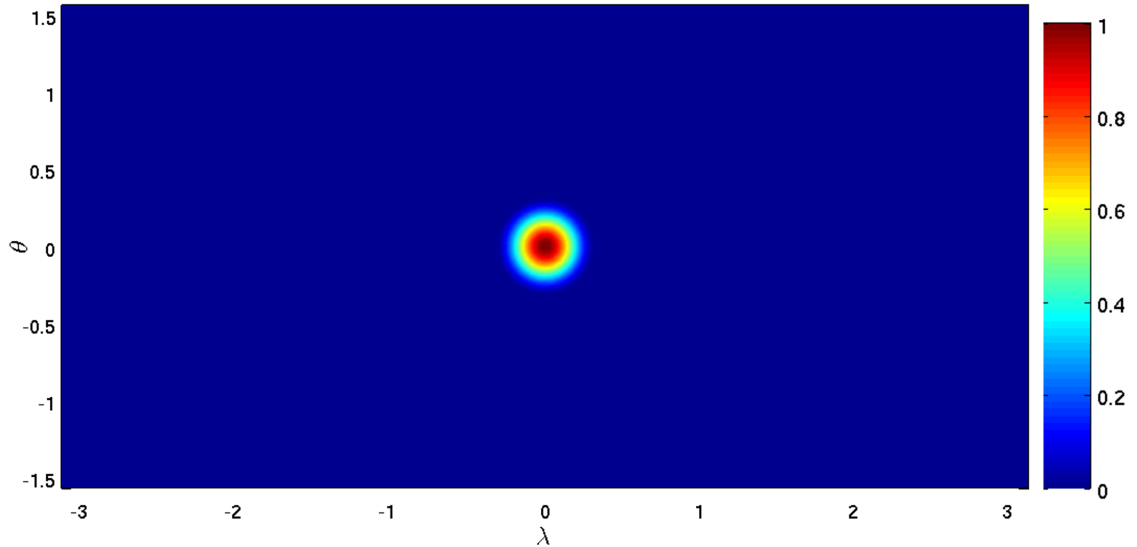
Numerical results: solid body rotation

Plots of the RBF-PUM solution for N=12544, n=100, Δt=2π/1600 :

Initial condition, t=0, with streamlines



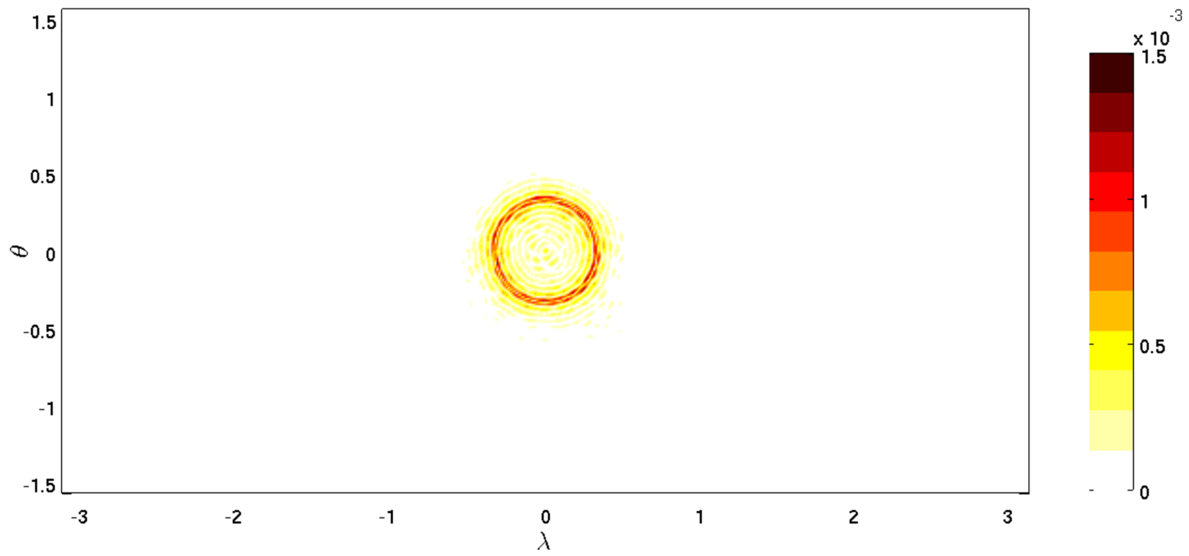
Solution after one revolution, t=2π



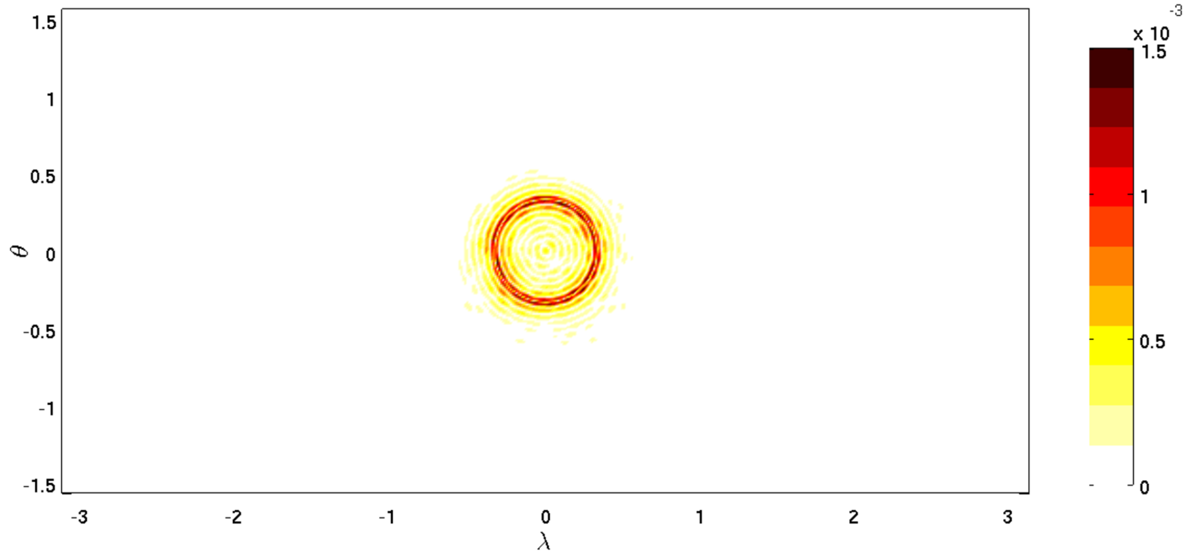
Numerical results: solid body rotation

Plots of the RBF-PUM error for N=12544, n=100, Δt=$2\pi/1600$:

Magnitude of the error after one revolution



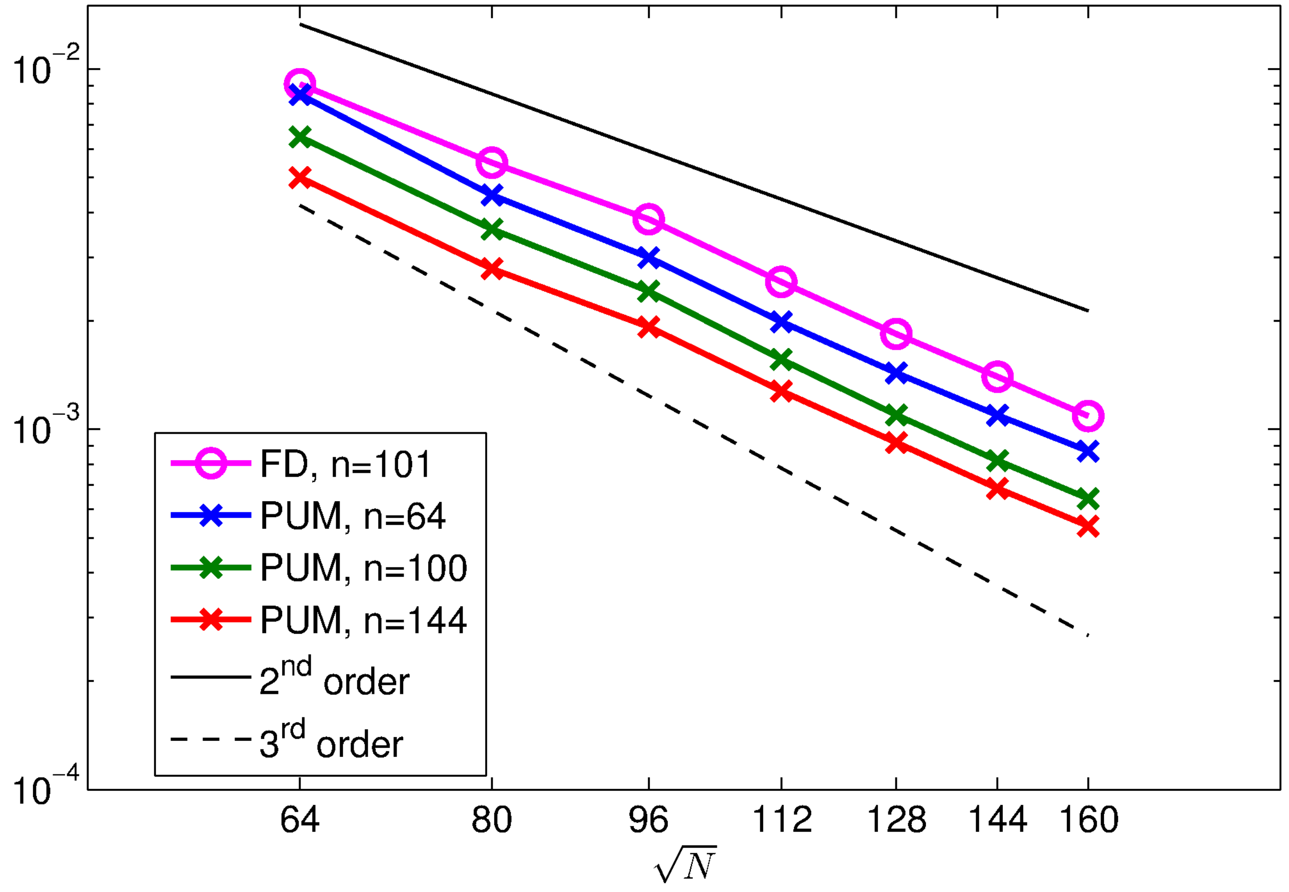
Magnitude of the error after ten revolutions



Numerical results: solid body rotation

Comparison of the errors for RBF-FD and RBF-PUM:

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

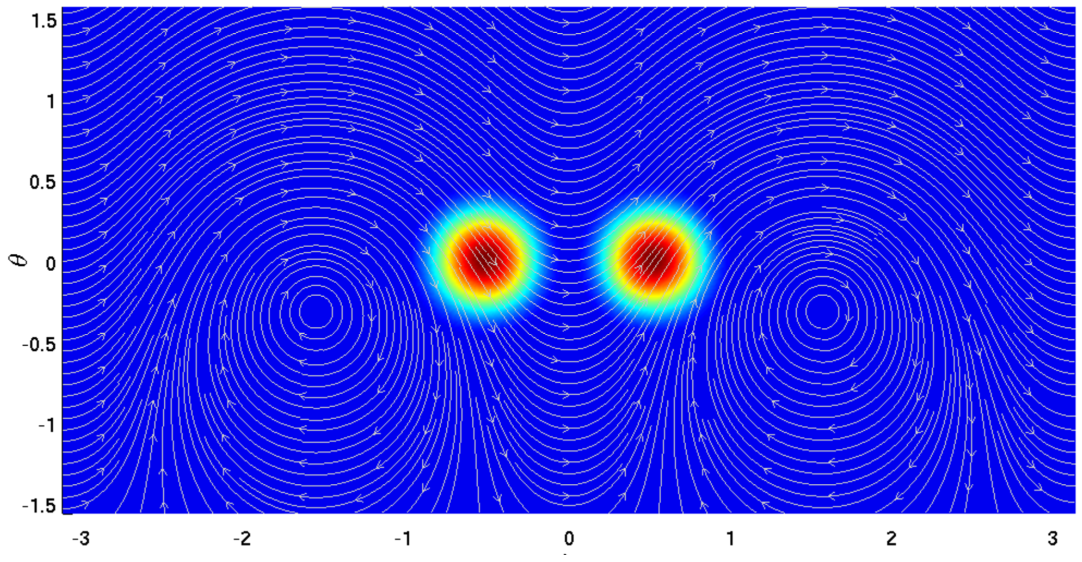


Convergence rates are as expected given smoothness of the initial condition.

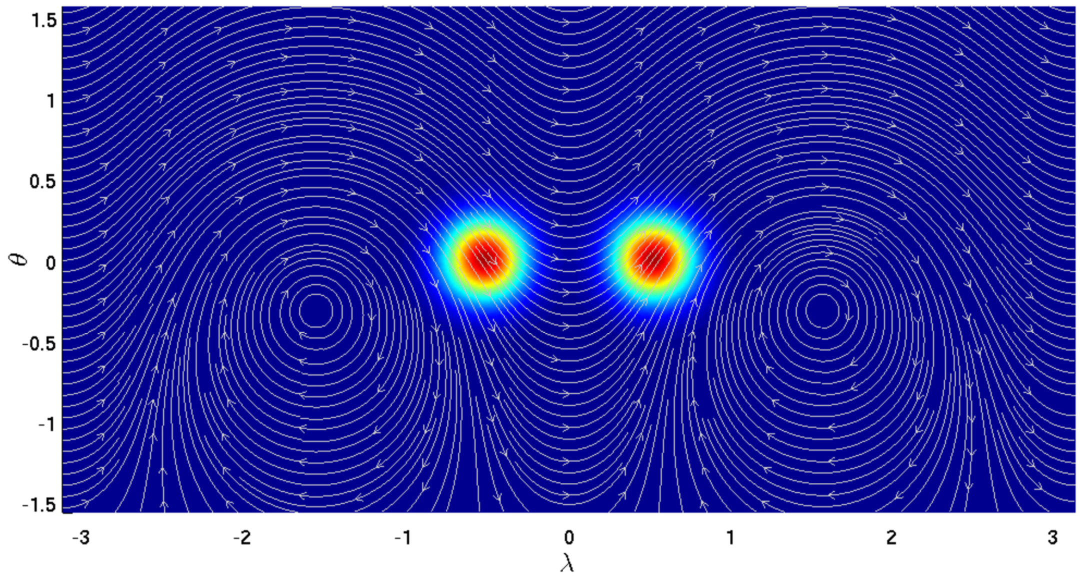
Numerical results: deformational flow

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

Non-smooth initial condition:

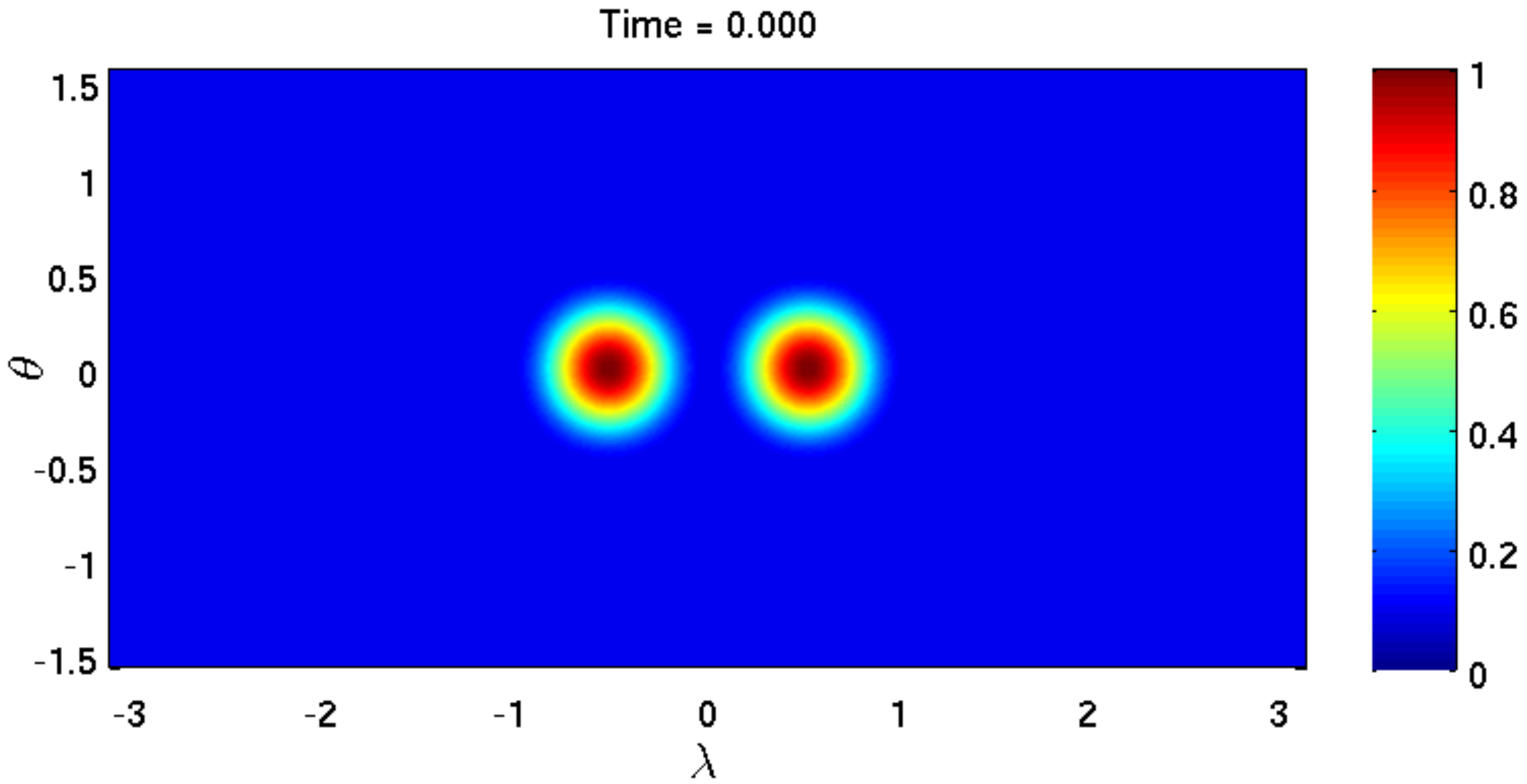


Smooth initial condition:



Numerical results: deformational flow

Simulation for non-smooth IC, N=20736, n=100, Δt=5/2400

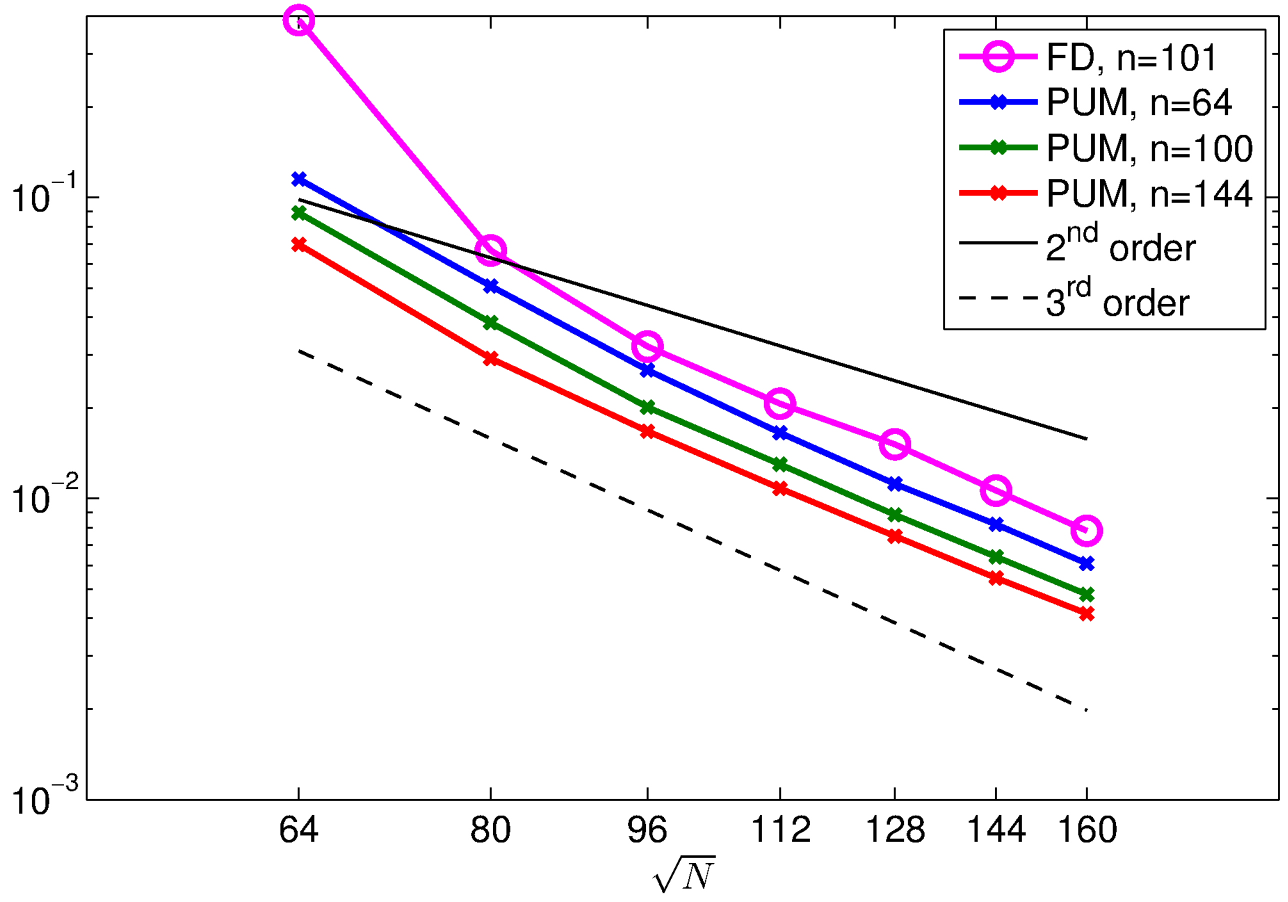


Numerical results: deformational flow

Convergence plots for increasing N and n, Δt=5/2400

- Non-smooth initial condition:

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

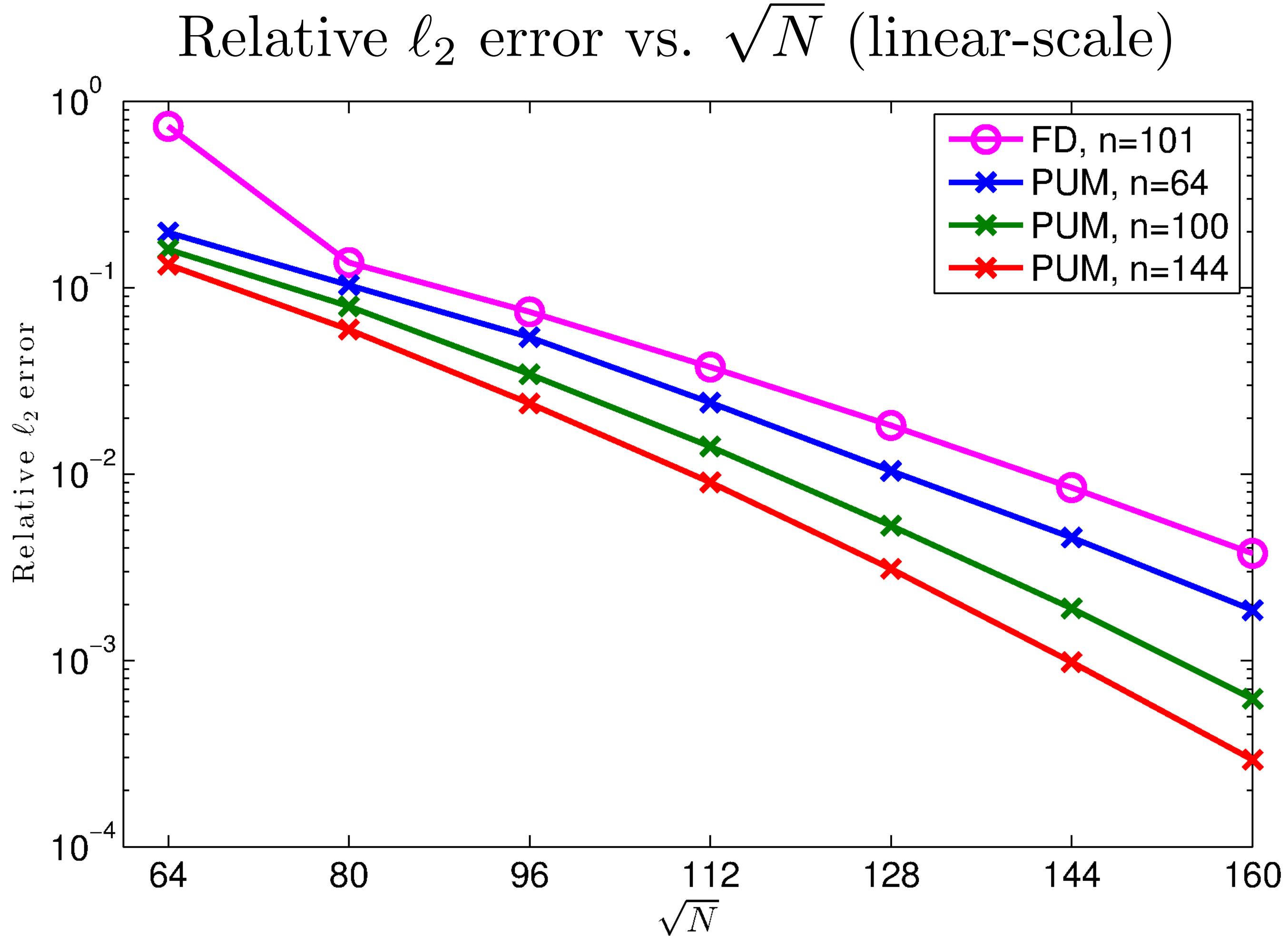


Convergence rates are as expected given smoothness of the initial condition.

Numerical results: deformational flow

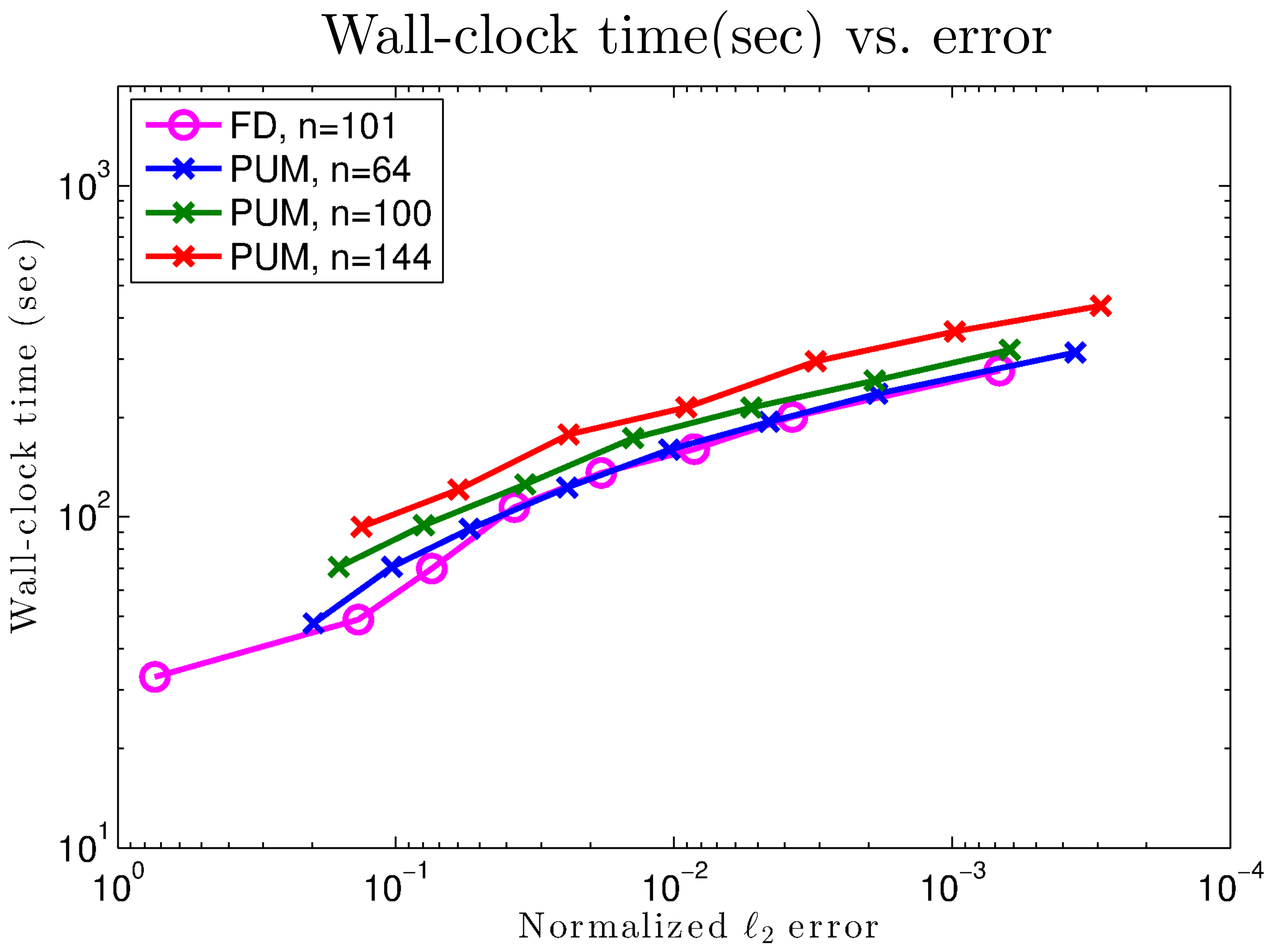
Convergence plots for increasing N and n, Δt=5/2400

- Smooth initial condition:



Comparison: Wall clock time vs. error

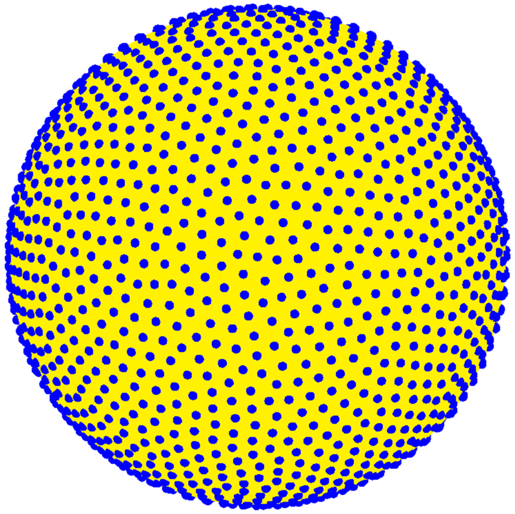
- Test: Deformational/Rotational flow smooth initial condition



- MATLAB R2013a, Intel Xeon 3.1GHz processor

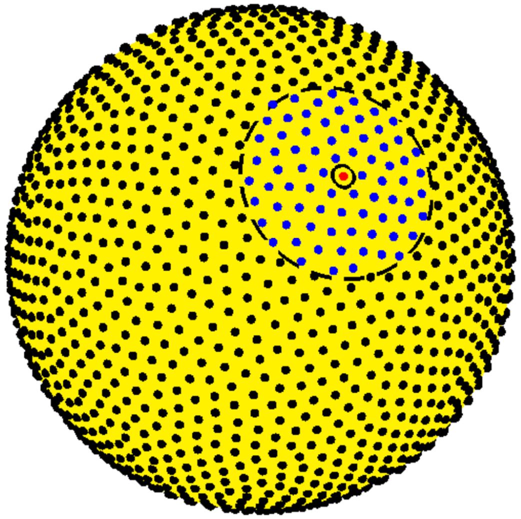
Comparison of Global, RBF-FD, and RBF-PUM methods

Global RBFs



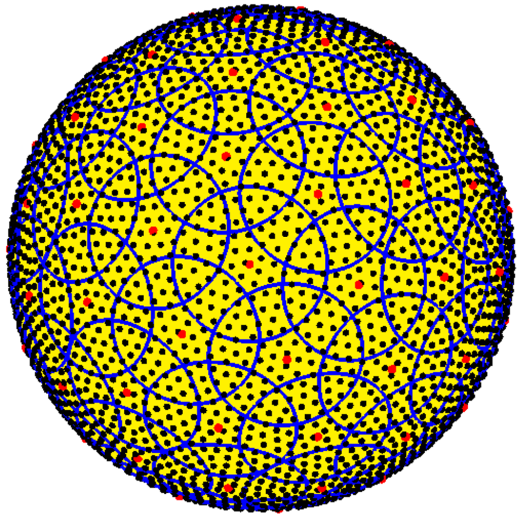
N total nodes

RBF-FD



N total nodes
n nodes per FD-stencil

RBF-PUM



N total nodes
M total patches
n nodes per patch

Cost Comparison:

Derivative approx.	**Global RBF**	**RBF-FD***	**RBF-PUM***
Construction:	$O(N^3)$	$O(n^3 N)$	$O(n^3 M)$
Evaluation:	$O(N^2)$	$O(nN)$	$O(nN)$

*Constants for the RBF-PUM are higher than for RBF-FD.

Concluding remarks

- 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.
  - More comparisons with other state-of-the art methods in the next lecture.
  - Parallelization on multi-GPU has already been implemented (Bollig, Flyer, & Erlebacher, 2012).
- The RBF Partition of unity method (RBF-PUM) also shows great promise.
  - More comparisons are needed between RBF-PUM and RBF-FD in terms of computational cost to achieve a certain accuracy.
  - Parallel implementations also needed.
- More work is needed in developing stable algorithms for “flat” RBFs when working on patches of the sphere.