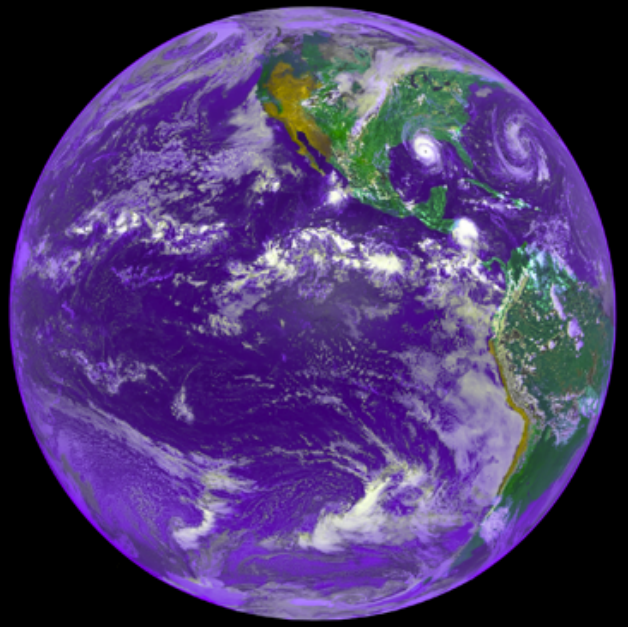
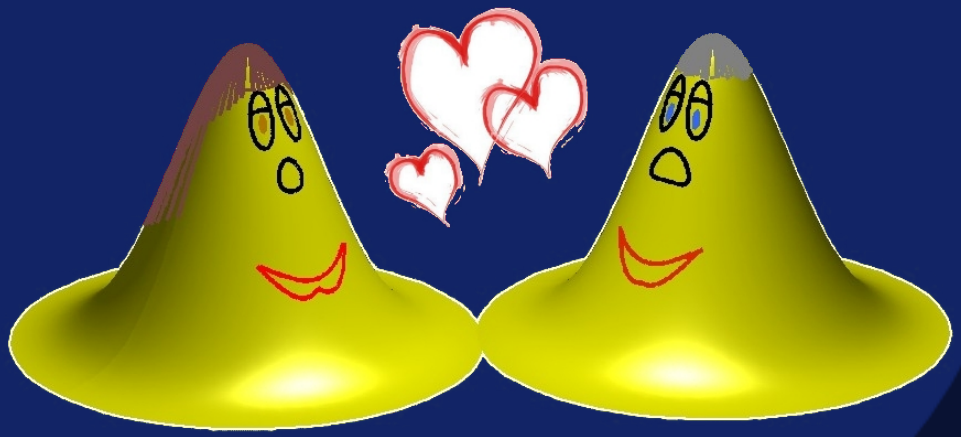
2013 Dolomites Research Week on Approximation



Lecture 3:
Good bases for kernel spaces

Grady B. Wright
Boise State University

Good Bases for Kernel Spaces



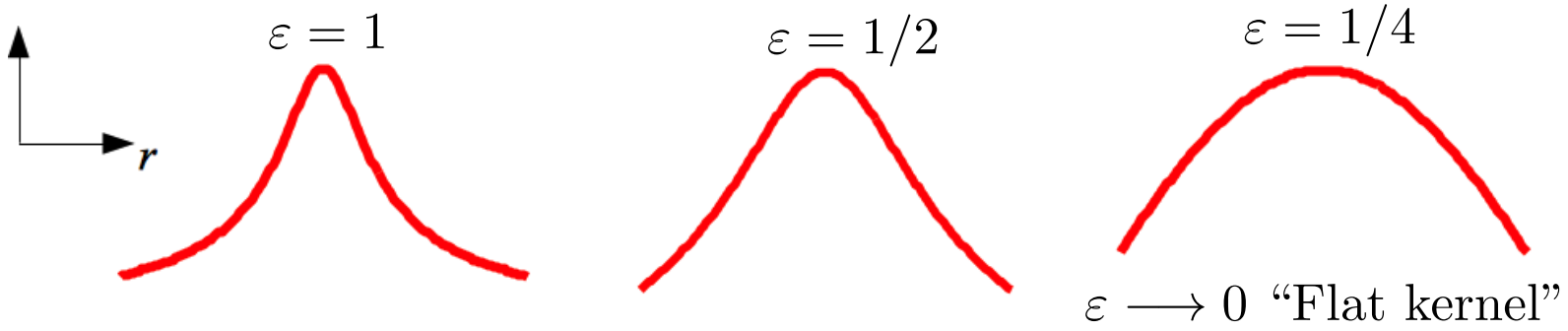
By Dr. RBF

Overview

- Part I: Smooth kernels with a shape parameter.

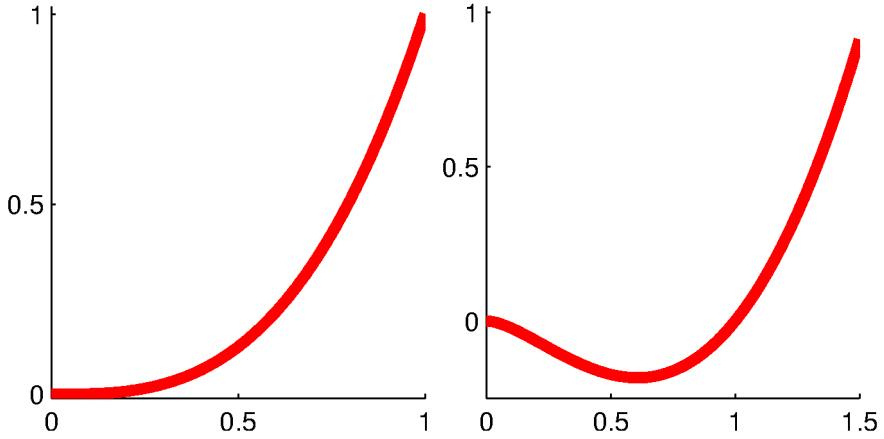
Ex: $\phi(r) = \exp(-(\varepsilon r)^2)$ $\phi(r) = \dfrac{1}{\sqrt{1+(\varepsilon r)^2}}$ $\phi(r) = \sqrt{1+(\varepsilon r)^2}$

Issue: Effect of decreasing ε leads to severe ill-conditioning of interp. matrices



- Part II: "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.

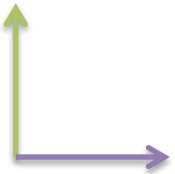
Good bases for kernel spaces

- Key observation: The space spanned by linear combinations of positive definite radial kernels (in $\mathbb{R}^d$ or $\mathbb{S}^2$) is good for approximation

BUT, the standard basis $\{\phi(\cdot, \mathbf{x}_1), \ldots, \phi(\cdot, \mathbf{x}_N)\}$ can be problematic.

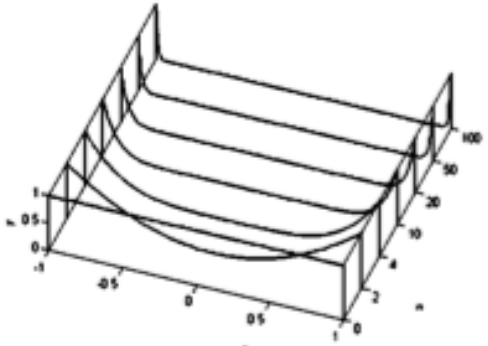
Analogy:
(Fornberg)

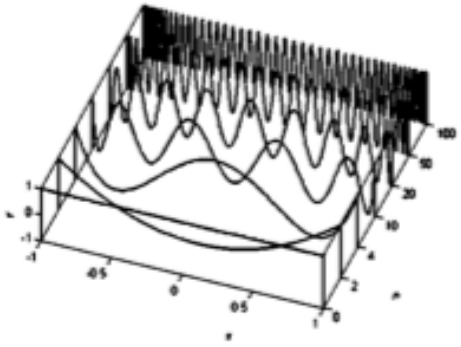
Vectors



Bad basis for $\mathbb{R}^2$ | Good basis for $\mathbb{R}^2$

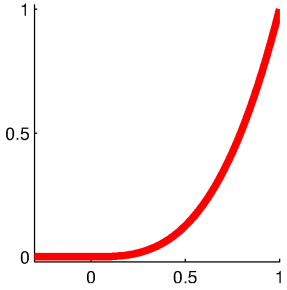
Polynomials

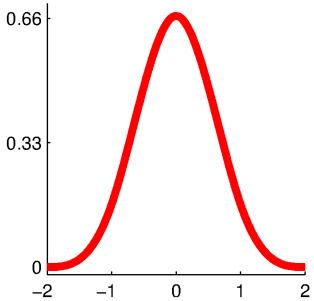




Bad basis: x^n, $n = 0, 1, \ldots$ | Chebyshev basis: $T_n(x)$, $n = 0, 1, \ldots$

Splines





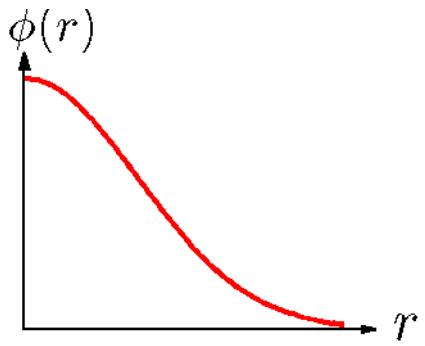
Truncated powers: $(x)_+^3$ | Bspline basis: $b_3(x)$

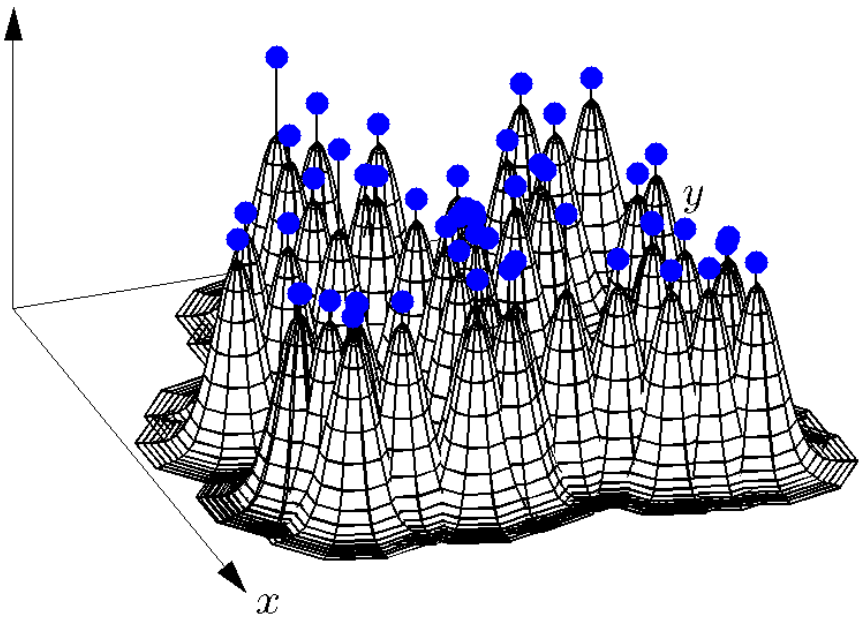
Part I: Smooth "flat kernels" (small shape parameter)

Radial basis function (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.

RBF-Direct

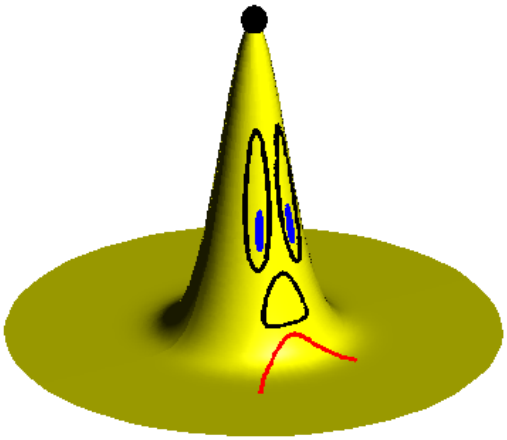
RBF interpolation in the "flat" limit

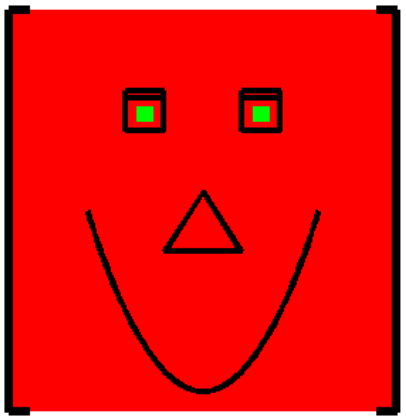
RBF interpolant: $$I_{X,\varepsilon} f(\mathbf{x}) = \sum_{j=1}^{N} c_j(\varepsilon)\phi_\varepsilon(\|\mathbf{x} - \mathbf{x}_j\|)$$

Theorem (Driscoll & Fornberg (2002)). For N nodes in 1-D, the RBF interpolant (for certain smooth kernels) converges to the standard Lagrange interpolant as $\varepsilon \longrightarrow 0$ (flat limit)

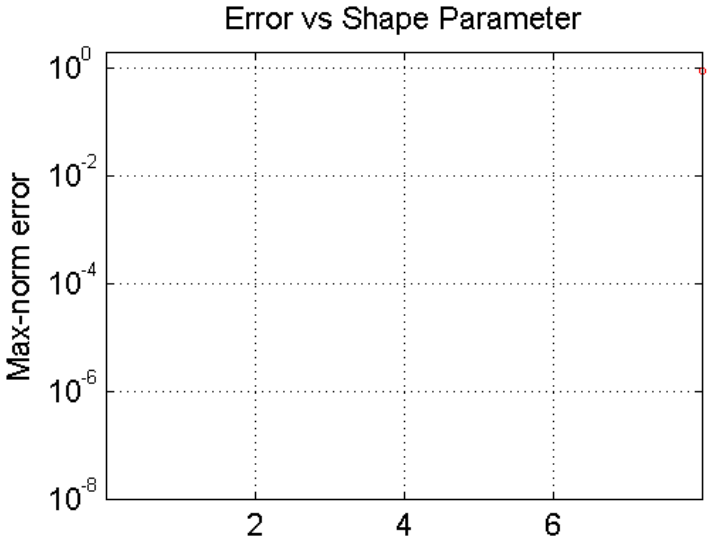
- Higher dimensions: Limit usually exits and takes the form of a multivariate polynomial as $\varepsilon \longrightarrow 0$.
  - Fornberg, W, & Larsson (2004), Larsson & Fornberg (2005), Schaback (2005,2006), Lee, Yoon, & Yoon (2007)
  - In the case of the Gaussian kernel, the interpolant always converges to the de Boor & Ron "least polynomial interpolant".
- Sphere: Limit (usually) exits and converges to a spherical harmonic interpolant (Fornberg & Piret (2007)).

Illustration of using a bad basis for flat kernels:





RBF-Direct



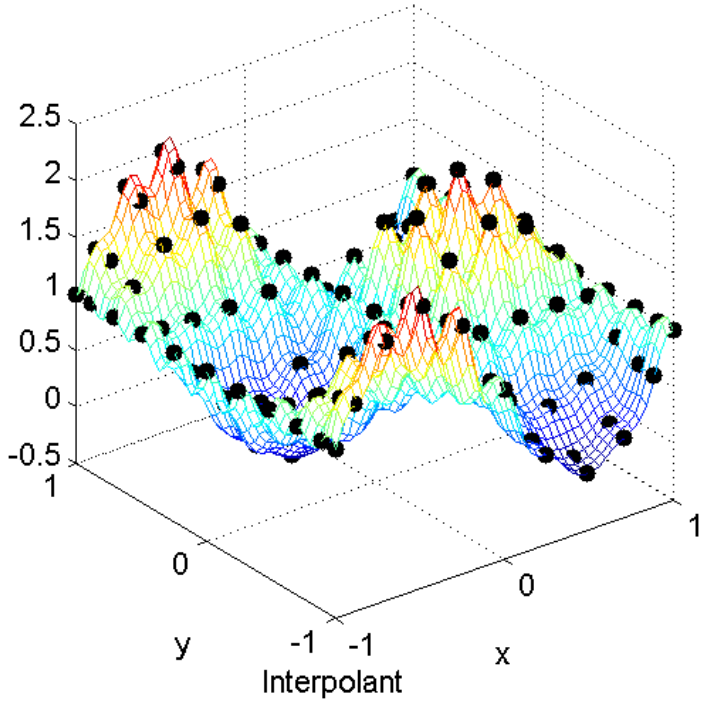
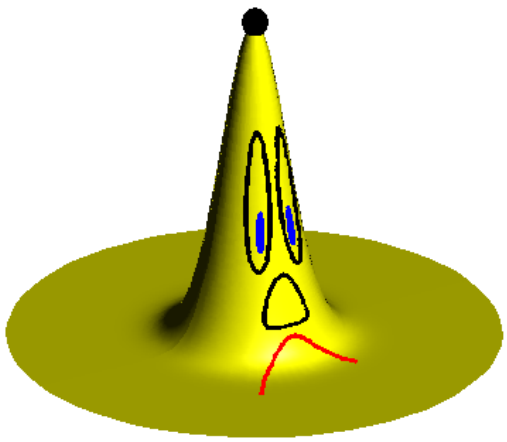
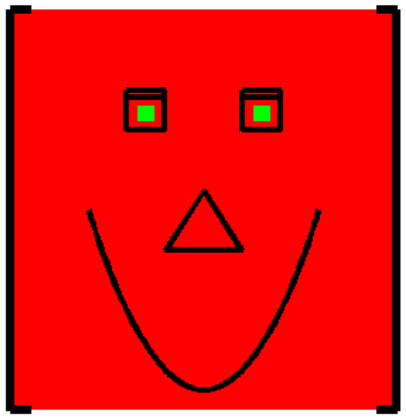
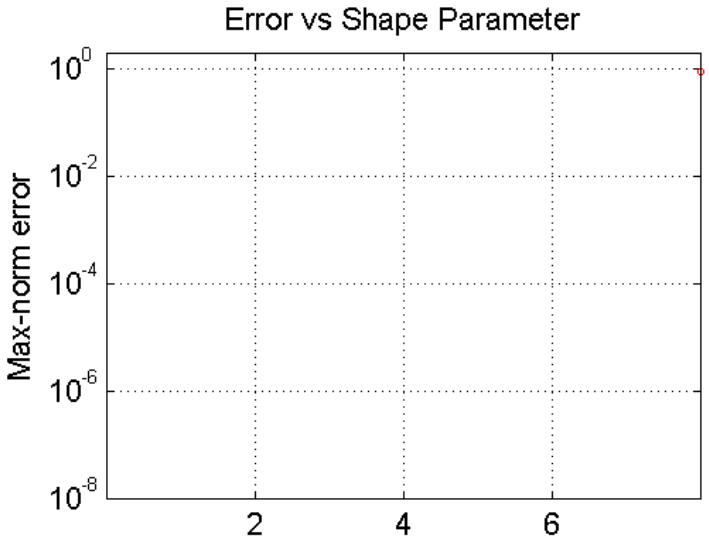


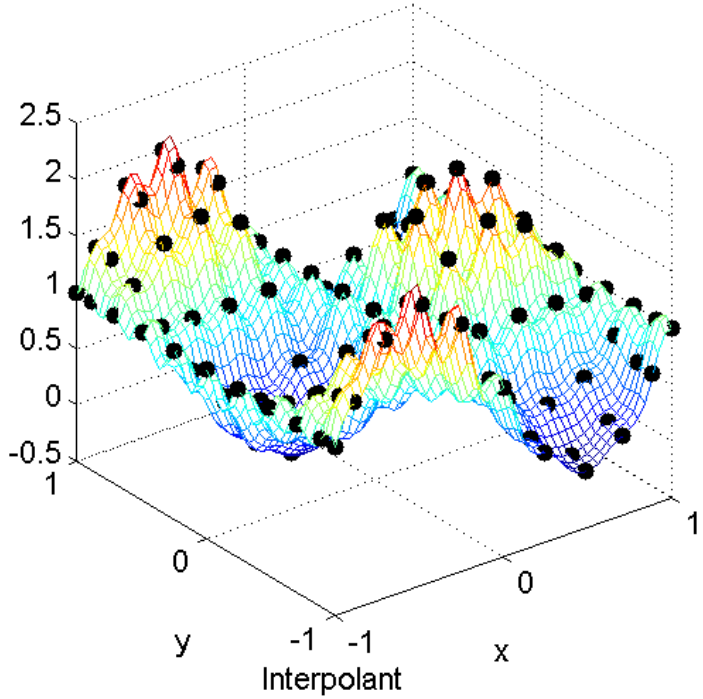
Illustration of using a good basis for flat kernels:





RBF-Direct





Behavior of interpolants in the flat limit

- For cardinal data $f(\mathbf{x}_j) = \begin{cases} 1 & \text{if } j = 1 \\ 0 & \text{if } j \neq 1 \end{cases}$ the interpolant can be written as

$$I_{X,\varepsilon} f(\mathbf{x}) = \frac{\det \begin{bmatrix} \phi_\varepsilon(\|\mathbf{x} - \mathbf{x}_1\|) & \phi_\varepsilon(\|\mathbf{x} - \mathbf{x}_2\|) & \cdots & \phi_\varepsilon(\|\mathbf{x} - \mathbf{x}_N\|) \\ \phi_\varepsilon(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi_\varepsilon(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_\varepsilon(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi_\varepsilon(\|\mathbf{x}_N - \mathbf{x}_2\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix}}{\det \begin{bmatrix} \phi_\varepsilon(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi_\varepsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \phi_\varepsilon(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi_\varepsilon(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_\varepsilon(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi_\varepsilon(\|\mathbf{x}_N - \mathbf{x}_2\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix}}$$

- Expand determinants using $\phi_\varepsilon(r) = a_0 + a_1 \varepsilon^2 r^2 + a_2 \varepsilon^4 r^4 + a_3 \varepsilon^6 r^6 + \cdots$:

$$I_{X,\varepsilon} f(\mathbf{x}) = \frac{\varepsilon^{2p}\{\text{poly. in } \mathbf{x}\} + \varepsilon^{2(p+1)}\{\text{poly. in } \mathbf{x}\} + \cdots}{\varepsilon^{2q}\{\text{constant}\} + \varepsilon^{2(q+1)}\{\text{constant}\} + \cdots}$$

- In general (and always for GA) $p = q$ so that the $\lim_{\varepsilon \longrightarrow 0} s(\mathbf{x}, \varepsilon)$ exists.

Behavior of interpolants in the flat limit

- Expand determinants using $\phi_\varepsilon(r) = a_0 + a_1\varepsilon^2 r^2 + a_2\varepsilon^4 r^4 + a_3\varepsilon^6 r^6 + \cdots$:

$$I_{X,\varepsilon} f(\mathbf{x}) = \frac{\varepsilon^{2p}\{\text{poly. in } \mathbf{x}\} + \varepsilon^{2(p+1)}\{\text{poly. in } \mathbf{x}\} + \cdots}{\varepsilon^{2q}\{\text{constant}\} + \varepsilon^{2(q+1)}\{\text{constant}\} + \cdots}$$

- In general (and always for GA) $p = q$ so that the $\lim_{\varepsilon \longrightarrow 0} s(\mathbf{x}, \varepsilon)$ exists.
- Example values of $2p$ and $2q$:

	d - dimension	N - number of data points							
		2	3	5	10	20	50	100	200
Leading power of ϵ	1	2	6	20	90	380	2450	9900	39800
in det(A) for both	2	2	4	12	40	130	570	1690	4940
numer. and denom.	3	2	4	10	30	90	360	980	2610

- High powers of ε indicate extreme ill-conditioning.
- Stable algorithms (better bases) are needed to reach the flat limit.

Uncertainty principle misconception

- Schaback's uncertainty principle:

  Principle: *One cannot simultaneously achieve good conditioning and high accuracy.*

  Misconception: Accuracy that can be achieved is limited by ill-conditioning.

  Restatement:

  One cannot simultaneously achieve good conditioning and high accuracy when using the standard basis.

- It's a matter of base vs. space.
- Literature for interpolation with "flat" kernels is growing:

Theory:
Driscoll & Fornberg (2002)
Larsson & Fornberg (2003; 2005)
Fornberg, Wright, & Larsson (2004)
Schaback (2005; 2008)
Platte & Driscoll (2005)
Fornberg, Larsson, & Wright (2006)
deBoor (2006)
Fornberg & Zuev (2007)
Lee, Yoon, & Yoon (2007)
Fornberg & Piret (2008)
Buhmann, Dinew, & Larsson (2010)
Platte (2011)
Song, Riddle, Fasshauer, & Hickernell (2011)

Stable algorithms:
Fornberg & Wright (2004)
Fornberg & Piret (2007)
Fornberg, Larsson, & Flyer (2011)
Fasshauer & McCourt (2011)
Gonnet, Pachon, & Trefethen (2011)
Pazouki & Schaback (2011)
De Marchi & Santin (2013)
Fornberg, Letho, Powell (2013)
Wright & Fornberg (2013)

Better kernel bases for the sphere: RBF-QR algorithm

- Key idea behind the RBF-QR algorithm is to exploit the spherical harmonic expansion of the kernel:

$$\phi_\varepsilon(\|\mathbf{x}-\mathbf{y}\|) = \phi_\varepsilon(\sqrt{2-2\mathbf{x}^T\mathbf{y}}) = \psi_\varepsilon(\mathbf{x}^T\mathbf{y}) = \sum_{\ell=0}^{\infty} \hat{\psi}_\varepsilon(\ell) \sum_{m=-\ell}^{\ell} Y_\ell^m(\mathbf{x}) Y_\ell^m(\mathbf{y})$$

- And use the nice properties of the resulting spherical Fourier coefficients:

Name	Kernel ($r(t)=\sqrt{2-2t}$)	Fourier coefficients $\hat{\psi}_\varepsilon(\ell)$ ($\varepsilon>0$)
Gaussian	$\psi(t) = \exp(-(\varepsilon r(t))^2)$	$\varepsilon^{2\ell}\frac{4\pi^{3/2}}{\varepsilon^{2\ell+1}} e^{-2\varepsilon^2} I_{\ell+1/2}(2\varepsilon^2)$
IMQ	$\psi(t) = \frac{1}{\sqrt{1+(\varepsilon r(t))^2)}}$	$\varepsilon^{2\ell}\frac{4\pi}{(\ell+1/2)}\left(\frac{2}{1+\sqrt{4\varepsilon^2+1}}\right)^{2\ell+1}$
MQ	$\psi(t) = -\sqrt{1+(\varepsilon r(t))^2)}$	$\varepsilon^{2\ell}\frac{2\pi(2\varepsilon^2+1+(\ell+1/2)\sqrt{1+4\varepsilon^2})}{(\ell+3/2)(\ell+1/2)(\ell-1/2)}\left(\frac{2}{1+\sqrt{4\varepsilon^2+1}}\right)^{2\ell+1}$

Note how the powers of ε appear in the coefficients.

- Can redefine the spherical harmonic expansion as:

$$\psi_\varepsilon(\mathbf{x}^T\mathbf{y}) = \sum_{\ell=0}^{\infty} \varepsilon^{2\ell}\tilde{\psi}_\varepsilon(\ell) \sum_{m=-\ell}^{\ell} Y_\ell^m(\mathbf{x}) Y_\ell^m(\mathbf{y}) \quad (\tilde{\psi}_\varepsilon(\ell) = \varepsilon^{-2\ell}\hat{\psi}(\ell))$$

Better kernel bases for the sphere: RBF-QR algorithm

- For $X = \{\mathbf{x}_j\}_{j=1}^N \subset \mathbb{S}^2$, we can write each basis function $\psi_\varepsilon(\mathbf{x}^T\mathbf{x}_j)$ as

$$
\begin{aligned}
\psi_\varepsilon(\mathbf{x}^T\mathbf{x}_1) =& \widetilde{\psi}_\varepsilon(0) Y_0^0(\mathbf{x}_1) Y_0^0(\mathbf{x}) + \\
& \varepsilon^2 \widetilde{\psi}_\varepsilon(1) \left\{ Y_1^{-1}(\mathbf{x}_1) Y_1^{-1}(\mathbf{x}) + Y_1^0(\mathbf{x}_1) Y_1^0(\mathbf{x}) + Y_1^1(\mathbf{x}_1) Y_1^1(\mathbf{x}) \right\} + \\
& \varepsilon^4 \widetilde{\psi}_\varepsilon(2) \{ \ldots\ldots \} + \varepsilon^6 \widetilde{\psi}_\varepsilon(3) \{ \ldots\ldots \} + \varepsilon^8 \widetilde{\psi}_\varepsilon(4) \{ \ldots\ldots \} + \ldots \\
\psi_\varepsilon(\mathbf{x}^T\mathbf{x}_2) =& \widetilde{\psi}_\varepsilon(0) Y_0^0(\mathbf{x}_2) Y_0^0(\mathbf{x}) + \\
& \varepsilon^2 \widetilde{\psi}_\varepsilon(1) \left\{ Y_1^{-1}(\mathbf{x}_2) Y_1^{-1}(\mathbf{x}) + Y_1^0(\mathbf{x}_2) Y_1^0(\mathbf{x}) + Y_1^1(\mathbf{x}_2) Y_1^1(\mathbf{x}) \right\} + \\
& \varepsilon^4 \widetilde{\psi}_\varepsilon(2) \{ \ldots\ldots \} + \varepsilon^6 \widetilde{\psi}_\varepsilon(3) \{ \ldots\ldots \} + \varepsilon^8 \widetilde{\psi}_\varepsilon(4) \{ \ldots\ldots \} + \ldots \\
\vdots & \qquad\qquad \vdots \\
\psi_\varepsilon(\mathbf{x}^T\mathbf{x}_N) =& \widetilde{\psi}_\varepsilon(0) Y_0^0(\mathbf{x}_N) Y_0^0(\mathbf{x}) + \\
& \varepsilon^2 \widetilde{\psi}_\varepsilon(1) \left\{ Y_1^{-1}(\mathbf{x}_N) Y_1^{-1}(\mathbf{x}) + Y_1^0(\mathbf{x}_N) Y_1^0(\mathbf{x}) + Y_1^1(\mathbf{x}_N) Y_1^1(\mathbf{x}) \right\} + \\
& \varepsilon^4 \widetilde{\psi}_\varepsilon(2) \{ \ldots\ldots \} + \varepsilon^6 \widetilde{\psi}_\varepsilon(3) \{ \ldots\ldots \} + \varepsilon^8 \widetilde{\psi}_\varepsilon(4) \{ \ldots\ldots \} + \ldots
\end{aligned}
$$

- For simplicity we assume N is a perfect square $(N = n+1)^2$.

- Or in matrix vector form as

$$\begin{bmatrix} \psi(\mathbf{x}^T\mathbf{x}_1) \\ \psi(\mathbf{x}^T\mathbf{x}_2) \\ \vdots \\ \psi(\mathbf{x}^T\mathbf{x}_N) \end{bmatrix} = BE \begin{bmatrix} Y_0^0(\mathbf{x}) \\ Y_1^{-1}(\mathbf{x}) \\ Y_1^0(\mathbf{x}) \\ Y_1^1(\mathbf{x}) \\ \vdots \end{bmatrix}$$

$$B = \begin{bmatrix} \widetilde{\psi}(0)Y_0^0(\mathbf{x}_1) & \widetilde{\psi}(1)Y_1^{-1}(\mathbf{x}_1) & \widetilde{\psi}(1)Y_1^0(\mathbf{x}_1) & \widetilde{\psi}(1)Y_1^1(\mathbf{x}_1) & \dots \\ \widetilde{\psi}(0)Y_0^0(\mathbf{x}_2) & \widetilde{\psi}(1)Y_1^{-1}(\mathbf{x}_2) & \widetilde{\psi}(1)Y_1^0(\mathbf{x}_2) & \widetilde{\psi}(1)Y_1^1(\mathbf{x}_2) & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \widetilde{\psi}(0)Y_0^0(\mathbf{x}_N) & \widetilde{\psi}(1)Y_1^{-1}(\mathbf{x}_N) & \widetilde{\psi}(1)Y_1^0(\mathbf{x}_N) & \widetilde{\psi}(1)Y_1^1(\mathbf{x}_N) & \dots \end{bmatrix}$$

$$E = \begin{bmatrix} 1 & & & & & \\ & \varepsilon^2 & & & & \\ & & \varepsilon^2 & & & \\ & & & \varepsilon^2 & & \\ & & & & \varepsilon^4 & \\ & & & & & \ddots \end{bmatrix}$$

Better kernel bases for the sphere: RBF-QR algorithm

$$\begin{bmatrix} \psi(\mathbf{x}^T\mathbf{x}_1) \\ \psi(\mathbf{x}^T\mathbf{x}_2) \\ \vdots \\ \psi(\mathbf{x}^T\mathbf{x}_N) \end{bmatrix} = BE \begin{bmatrix} Y_0^0(\mathbf{x}) \\ Y_1^{-1}(\mathbf{x}) \\ Y_1^0(\mathbf{x}) \\ Y_1^1(\mathbf{x}) \\ \vdots \end{bmatrix} \iff \underline{\psi}(\mathbf{x}) = BE\underline{Y}$$

- We can't deal with infinite matrices so we truncate according to some tolerance (see Fornberg & Piret (2007)).
- Let $\mu > n$ be the truncation degree of the expansion.
- Partition the truncated matrices:

$$B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$$

where the columns of B_1 consist of spherical harmonics up to degree n and B_2 consist of spherical harmonics from degree $n+1$ to μ.

$$E = \begin{bmatrix} E_1 & \\ & E_2 \end{bmatrix}$$

where E_1 is diagonal with blocks of powers of ε from 0 to $2n$ and where E_2 is diagonal with blocks of powers of ε from $2(n+1)$ to 2μ.

Better kernel bases for the sphere: RBF-QR algorithm

$$\begin{bmatrix} \psi(\mathbf{x}^T\mathbf{x}_1) \\ \psi(\mathbf{x}^T\mathbf{x}_2) \\ \vdots \\ \psi(\mathbf{x}^T\mathbf{x}_N) \end{bmatrix} = BE \begin{bmatrix} Y_0^0(\mathbf{x}) \\ Y_1^{-1}(\mathbf{x}) \\ Y_1^0(\mathbf{x}) \\ Y_1^1(\mathbf{x}) \\ \vdots \\ Y_\nu^\nu(\mathbf{x}) \end{bmatrix} \iff \underline{\psi}(\mathbf{x}) = \begin{bmatrix} B_1 & B_2 \end{bmatrix} \begin{bmatrix} E_1 & \\ & E_2 \end{bmatrix} \underline{Y}$$

- Now we change basis, by QR decomposition of $[B_1\ B_2]$ and manipulations with E.

$$\begin{aligned} \underline{\psi}(\mathbf{x}) &= QR \begin{bmatrix} E_1 & \\ & E_2 \end{bmatrix} \underline{Y} \\ &= Q \begin{bmatrix} R_1 & R_2 \end{bmatrix} \begin{bmatrix} E_1 & \\ & E_2 \end{bmatrix} \underline{Y} \quad (R_1 \text{ is } N\text{-by-}N \text{ and upper triangular}) \\ &= QR_1 \begin{bmatrix} I & R_1^{-1}R_2 \end{bmatrix} \begin{bmatrix} E_1 & \\ & E_2 \end{bmatrix} \underline{Y} \\ &= QR_1E_1 \underbrace{\begin{bmatrix} I & E_1^{-1}R_1^{-1}R_2E_2 \end{bmatrix} \underline{Y}}_{\text{New stable basis}} \end{aligned}$$

- All negative powers of ε in $E_1^{-1}R_1^{-1}R_2E_2$ analytically cancel after some matrix manipulations.

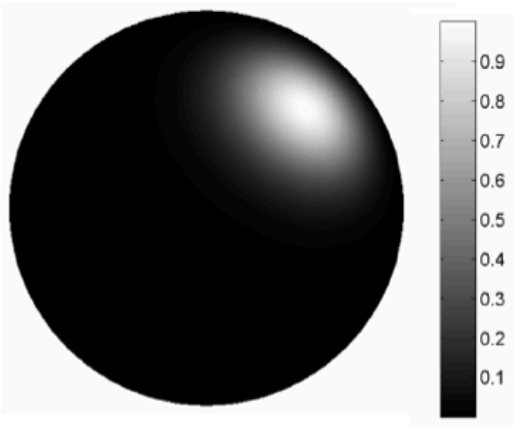
Numerical example

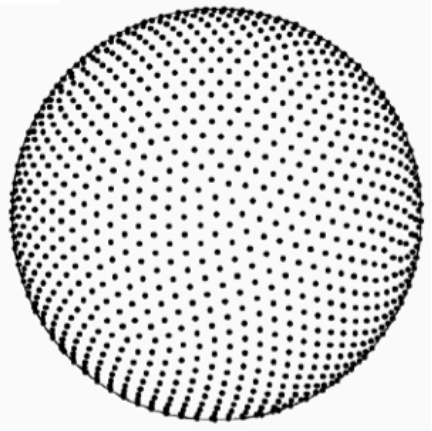
Target function:

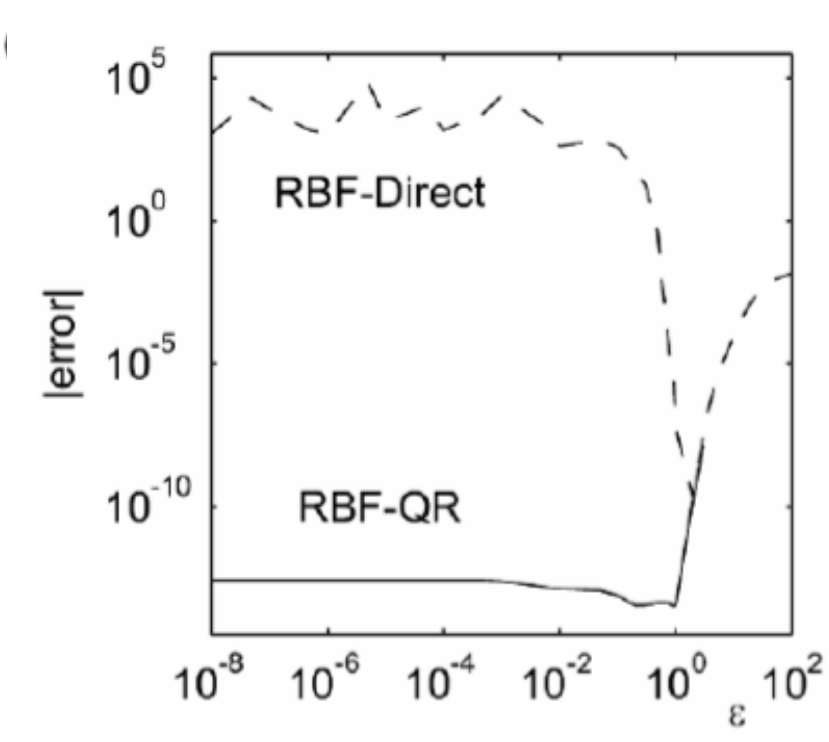
$f(\mathbf{x}) = \exp(-7(x + \frac{1}{2}^2) - 8(y + \frac{1}{2})^2 - 9(z - \frac{1}{\sqrt{2}})^2$

Node Set:

$N = 1849$ Minimal energy (ME) nodes







Concluding remarks on RBF-QR

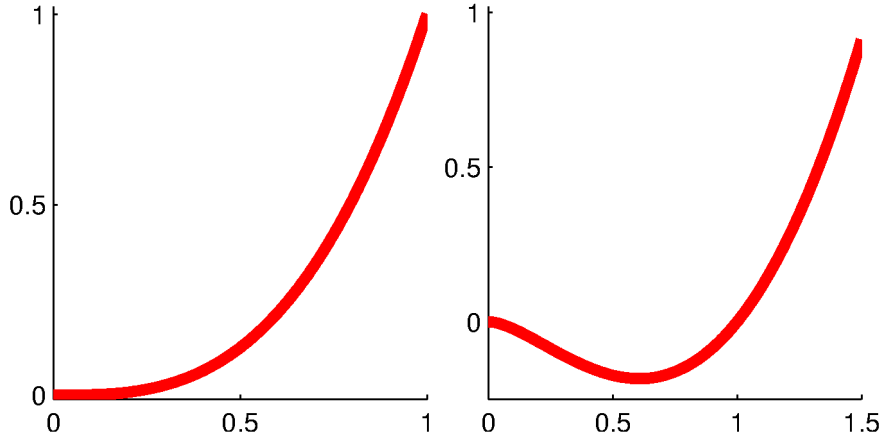
- RBF-QR allows one to stably compute "flat" kernel interpolants on the sphere.
- One can reach full numerical precision using this procedure (for smooth enough target functions and large enough N)
- It is more expensive than standard approach (RBF-Direct).
- Work has gone into extending this idea to general Euclidean space, but the procedure is much more complicated.
- Matlab Code for RBF-QR is given in Fornberg & Piret (2007)
- One interesting idea is to extend the procedure for the divergence- and curl-free vector interpolants.

Part II: Scale independent kernels

Good bases for kernel spaces

- Part II: "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

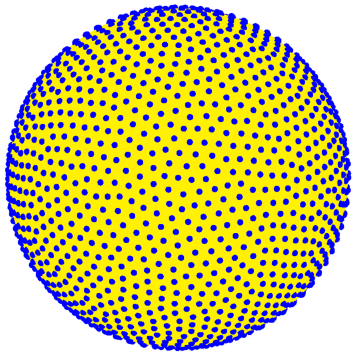
Restrict our attention to $\Omega = \mathbb{S}^2$ and $\psi_\ell(t) = (1-t)^{\ell-1}\log(1-t)$

Standard zonal interpolant:

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

Lagrange form:

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

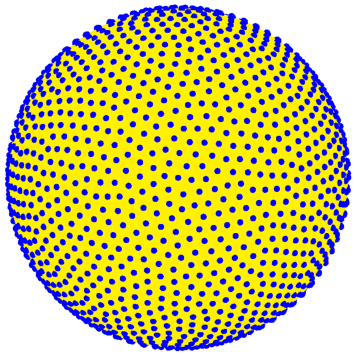


Lagrange functions on the sphere

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

Standard zonal interpolant:

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



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)

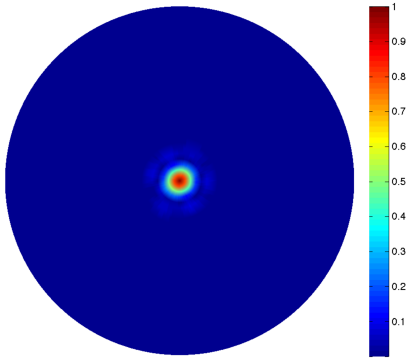
1. Lagrange basis is local (HNW, 2010):

$$|L_j(\mathbf{x})| \le C \exp\left[-\nu \frac{\mathrm{dist}(\mathbf{x}_j,\mathbf{x})}{h_X}\right]$$

2. Lebesgue constant is bounded (HNW, 2010):

$$\mathcal{L}_X := \max_{\mathbf{x}\in\mathbb{S}^2} \sum_{j=1}^{N} |L_j(\mathbf{x})| \le C$$

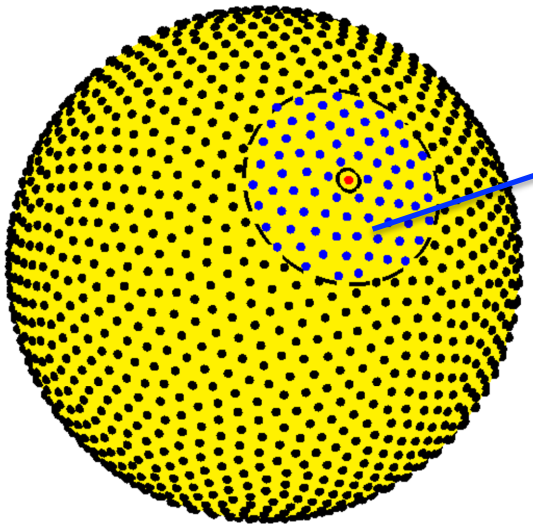
3. Lagrange basis is stable (HNSW, 2011)



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

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 $\mathbf{x}_i$:

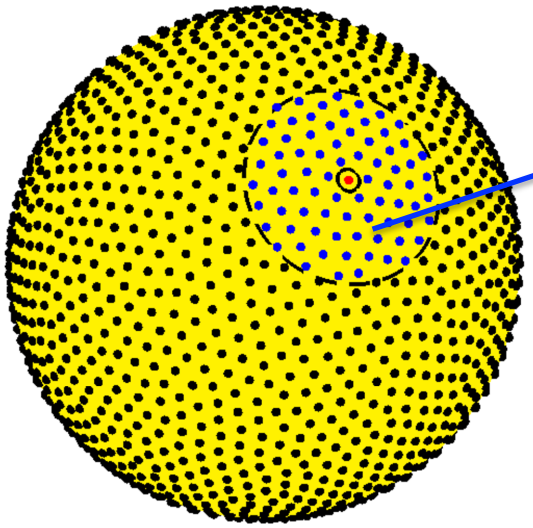
$$X_i = \{\mathbf{x}_j^i\}_{j=1}^n \subset X$$

2. Construct the local Lagrange function on X:

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

Local Lagrange functions on the sphere

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



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

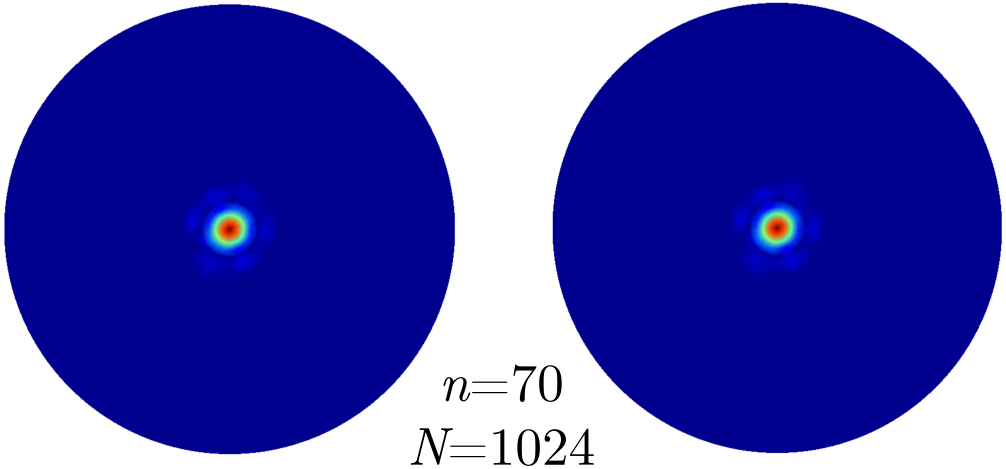
1. Choose $n << N$ nearest neighbors to $\mathbf{x}_i$:

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

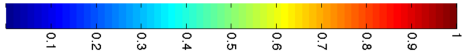
2. Construct the local Lagrange function on X:

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

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

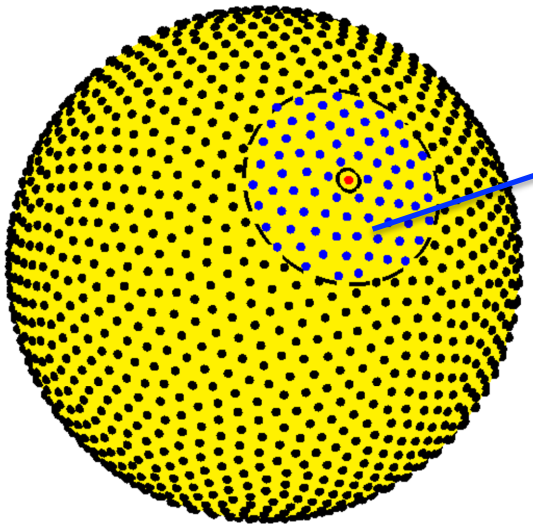


n=70
N=1024



Local Lagrange functions on the sphere

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



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

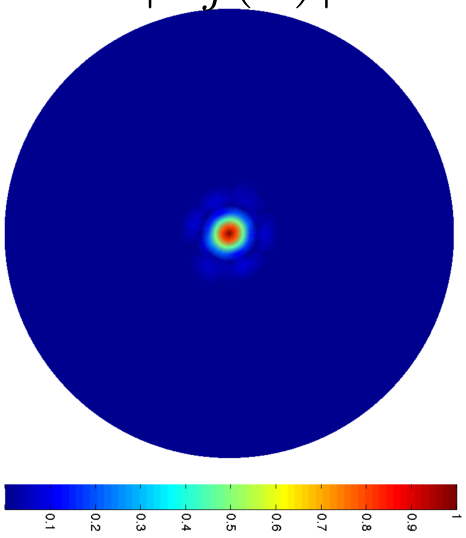
1. Choose $n << N$ nearest neighbors to $\mathbf{x}_i$:

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

2. Construct the local Lagrange function on X:

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

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



Estimates: (FHNWW, 2013)

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

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

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

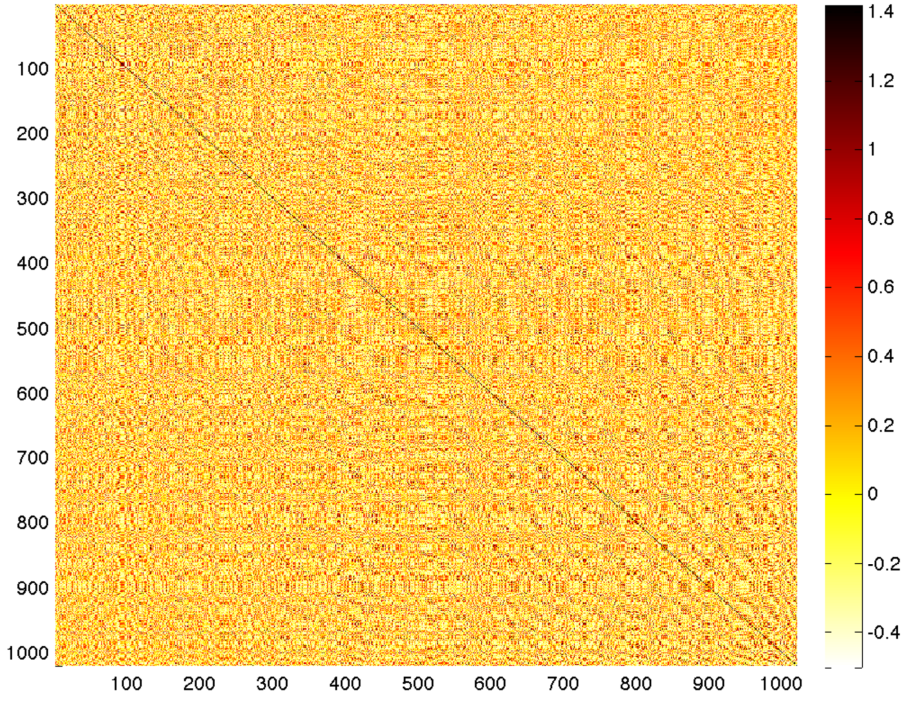
Interpolation matrices

- Example: N=1024, n=70

Standard basis:

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \psi_\ell(\mathbf{x}^T \mathbf{x}_j) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x})$$

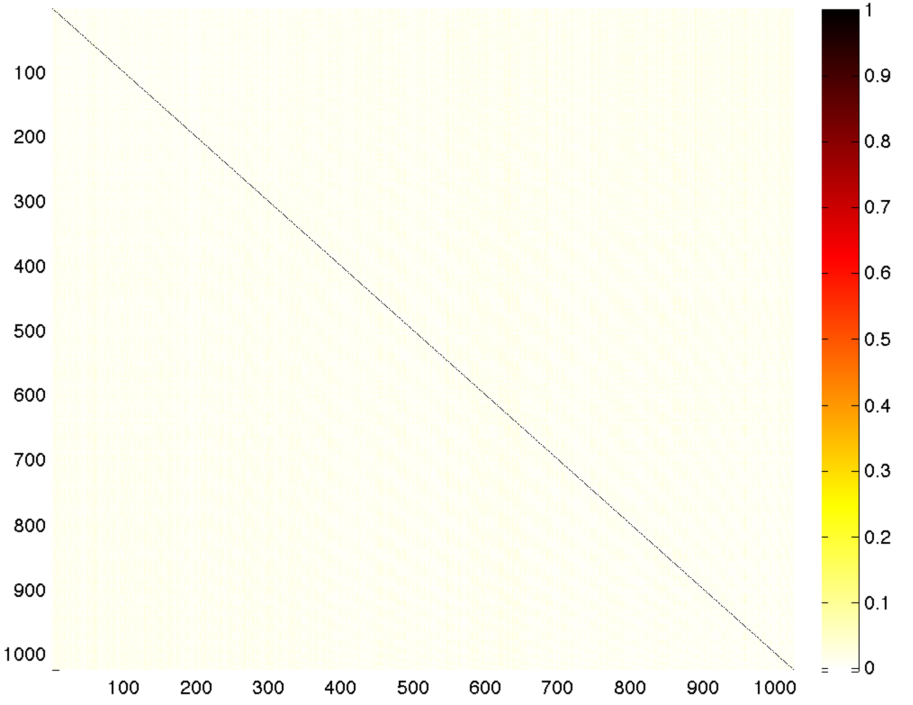
Interpolation matrix



Approximate Lagrange basis:

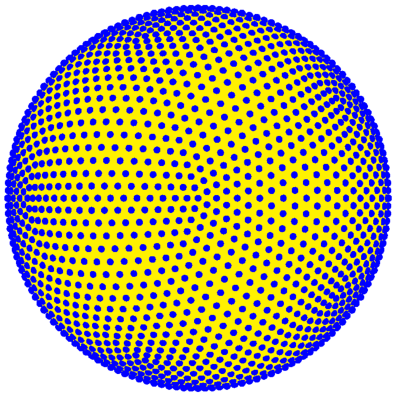
$$s(\mathbf{x}) = \sum_{j=1}^{N} a_j \widetilde{L}_j(\mathbf{x})$$

Interpolation matrix



Solving "preconditioned" systems

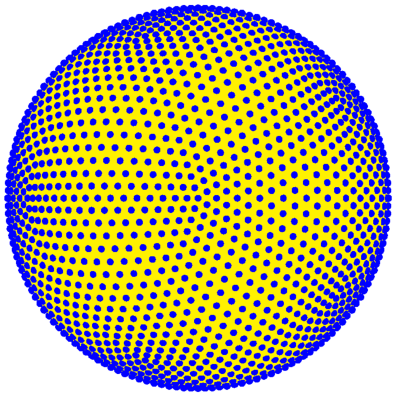
- Numerical experiment: $s(\mathbf{x}) = \sum_{j=1}^{N} a_j \widetilde{L}_j(\mathbf{x})$



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



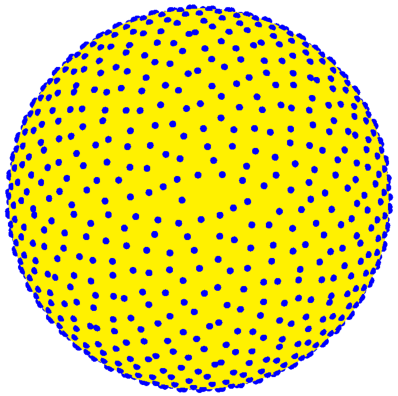
  - Target f: random values distributed between $[-1, 1]$.
  - $\widetilde{L}_j$ constructed from $n = 7\lceil(\log_{10} N)^2\rceil$ neighbors
  - Systems solved using GMRES iterative method (Saad & Schultz, 1986)

			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).

Solving "preconditioned" systems

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



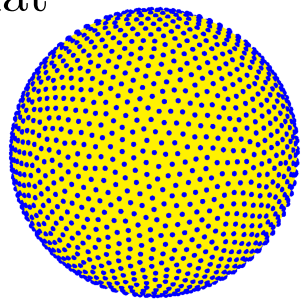
  - Target f: random values distributed between $[-1, 1]$.
  - $\widetilde{L}_j$ constructed from $n = 7\lceil(\log_{10} N)^2\rceil$ neighbors
  - Systems solved using GMRES iterative method (Saad & Schultz, 1986)

			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!).

Application: 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



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

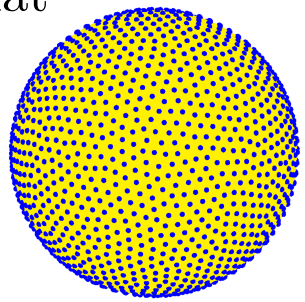
- Our solution: Find the weights from the kernel interpolant of f on X:

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

- So what are the weights?

Application: 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



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

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

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

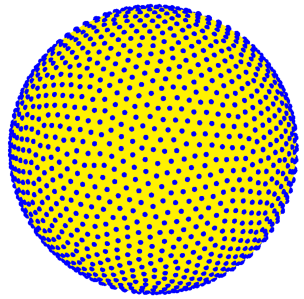
- So what are the weights?

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

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

Quadrature on the sphere

- For simplicity suppose ψ is a positive definite zonal kernel, i.e.
$$\Longrightarrow \psi(\mathbf{x}, \mathbf{y}) = \psi(\mathbf{x} \cdot \mathbf{y})$$



- Thus

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

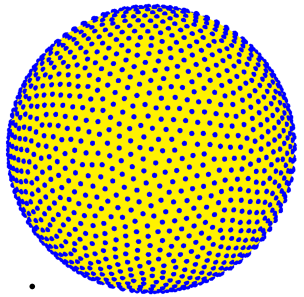
$$= \left(\int_{\mathbb{S}^2} \psi(\mathbf{x} \cdot \mathbf{x}_1) d\mu(\mathbf{x}) \right) \sum_{j=1}^{N} c_j = J \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}$$

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

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

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



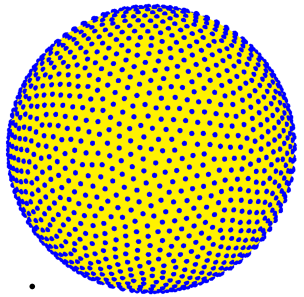
- Our solution: Find the weights from the kernel interpolant of $f\big|_X$:

$$\begin{bmatrix} & \psi(\mathbf{x}_i \cdot \mathbf{x}_j) & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} J \\ \vdots \\ J \end{bmatrix}$$

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

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



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

$$\begin{bmatrix} & \\ & \psi(\mathbf{x}_i \cdot \mathbf{x}_j) \\ & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} J \\ \vdots \\ J \end{bmatrix}$$

- Note that this idea can be extended to CPD kernels as well: (FHNWW-2, 2013)

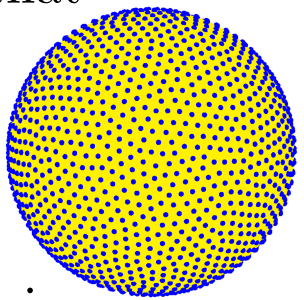
$$s_X(\mathbf{x}) = \sum_{j=1}^N c_j \psi_\ell(\mathbf{x} \cdot \mathbf{x}_j) + \sum_{k=1}^{\ell^2} b_k p_k(\mathbf{x}), \quad \psi_\ell(t) = (1-t)^{\ell-1} \log(1-t)$$

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

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

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



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

$$\begin{bmatrix} & \psi(\mathbf{x}_i \cdot \mathbf{x}_j) & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} J \\ \vdots \\ J \end{bmatrix}$$

- Note that this idea can be extended to CPD kernels as well: (FHNWW-2, 2013)

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

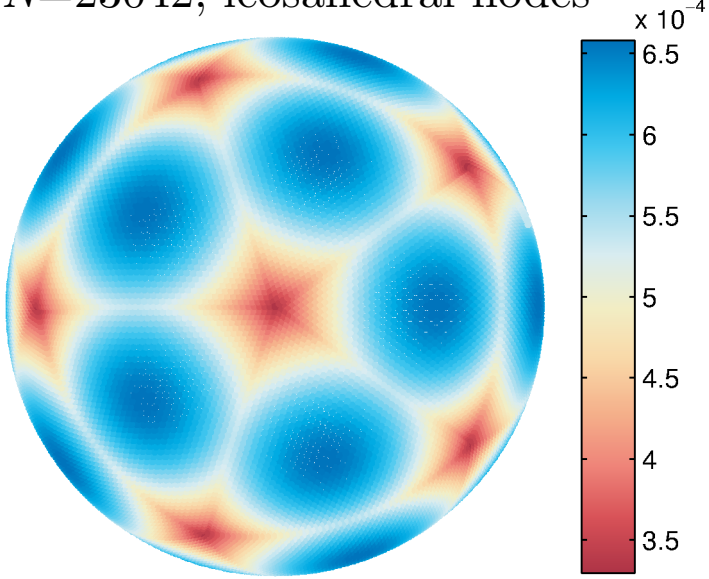
- How ELSE can this be made computationally tractable for large N?

Local Lagrange basis!

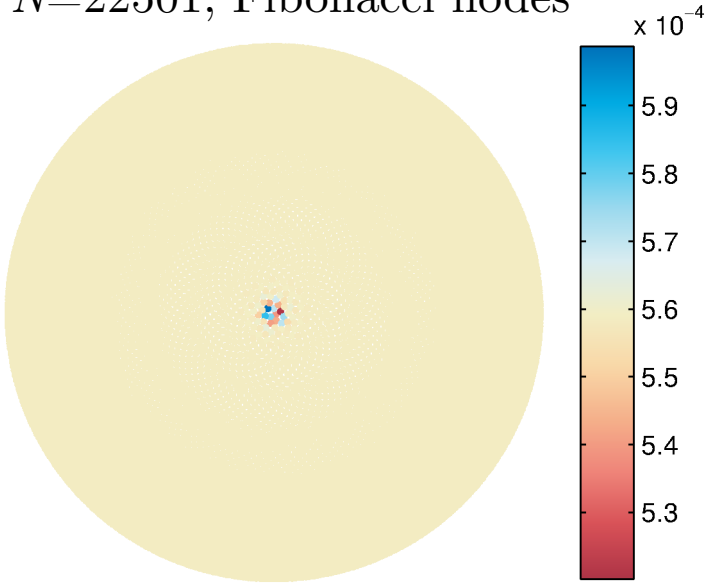
Example of quadrature weights

- Quadrature weights computed using
$$\psi_2(t) = (1-t)\log(1-t)$$
- Local Lagrange preconditioner

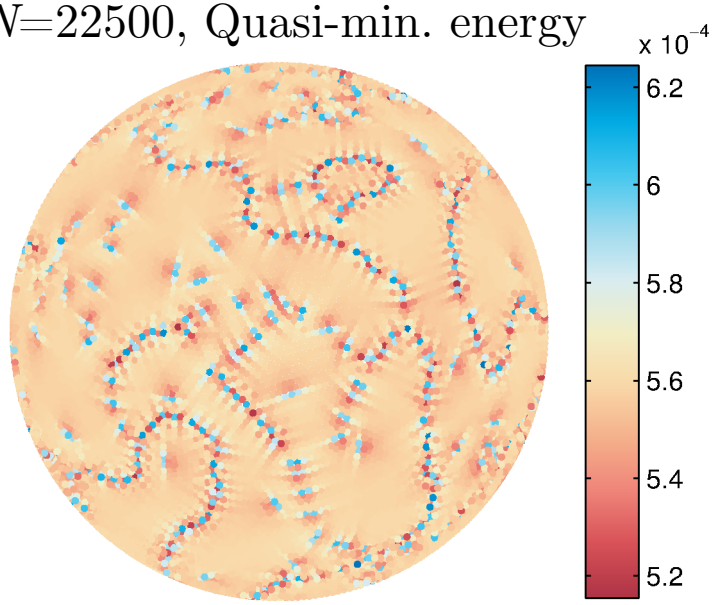
N=23042, icosahedral nodes



N=22501, Fibonacci nodes



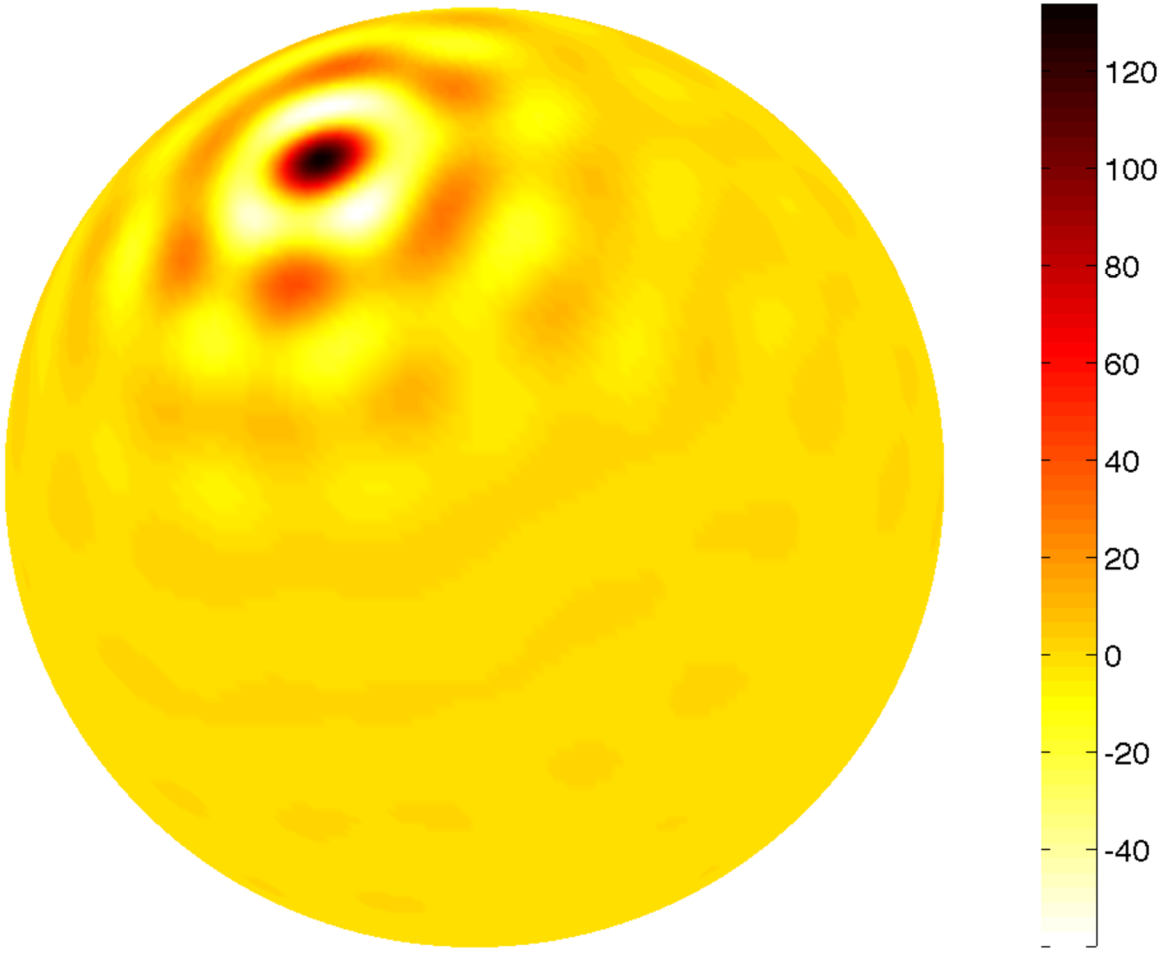
N=22500, Quasi-min. energy



Borodachov, Hardin, Saff (2013)

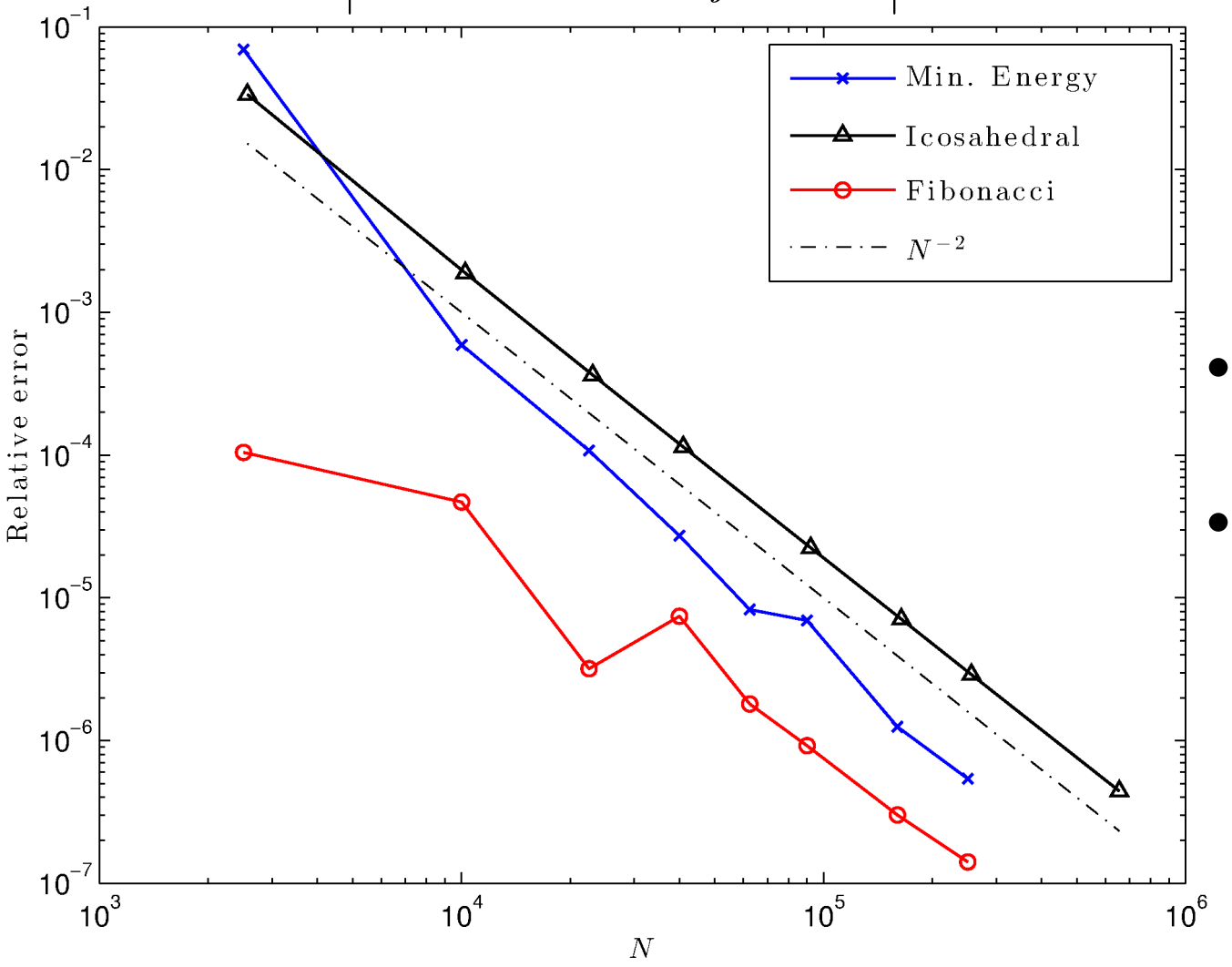
Numerical example

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 $\mathcal{O}(h_X^4)$.

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.
  - Future: implement fast *evaluation* algorithm to reduce the global cost.
- Future: exploit local Lagrange basis as a quasi-interpolant
- Future: develop theory and numerics to handle non-uniform nodes.
- Local lagrange basis can be extended to other manifolds:
  - Future: computable kernels for general manifolds.
- Future: Can more be said about smooth kernels on manifolds: error estimates and good bases (or stable algorithms).

Thank you!