A rational approximation algorithm for stable computations with flat RBFs

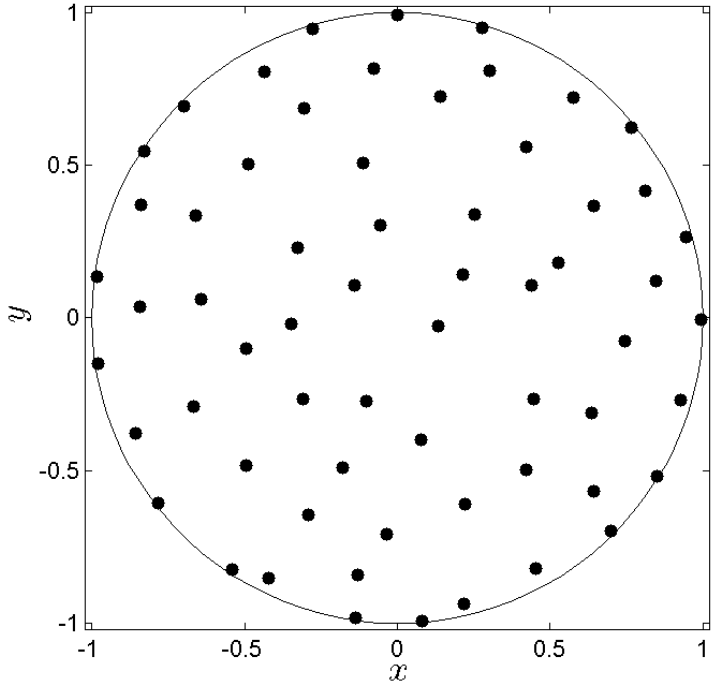
Grady B. Wright
Boise State University

Bengt Fornberg
University of Colorado, Boulder

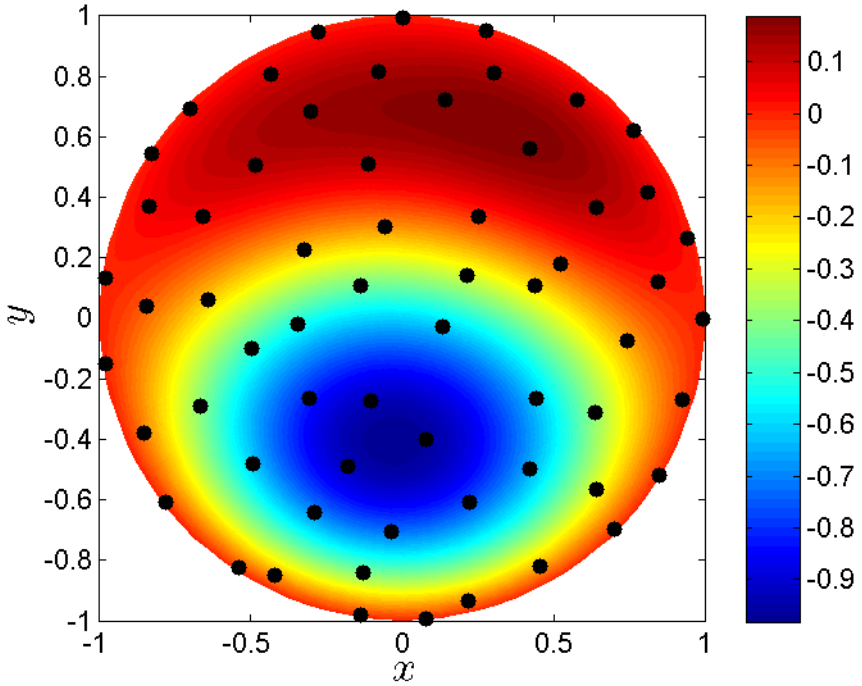
*This work is supported by grants NSF-DMS 0934581

Example interpolation problem

Nodes $\widehat{X}$, $N = 62$



Target function

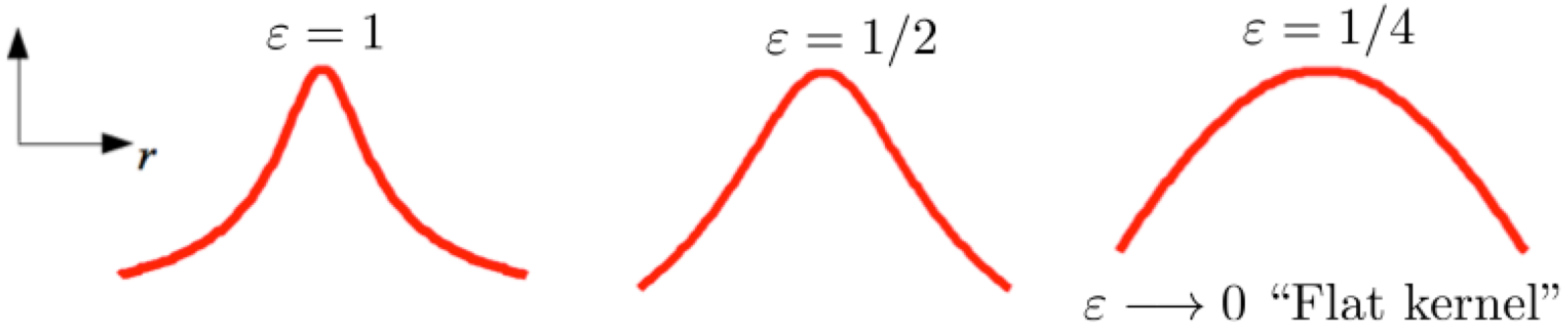


$$f(\mathbf{x}) = (1 - (x^2 + y^2)) \left[\sin\left(\frac{\pi}{2}(y - 0.07)\right) - \frac{1}{2}\cos\left(\frac{\pi}{2}(x + 0.1)\right) \right]$$

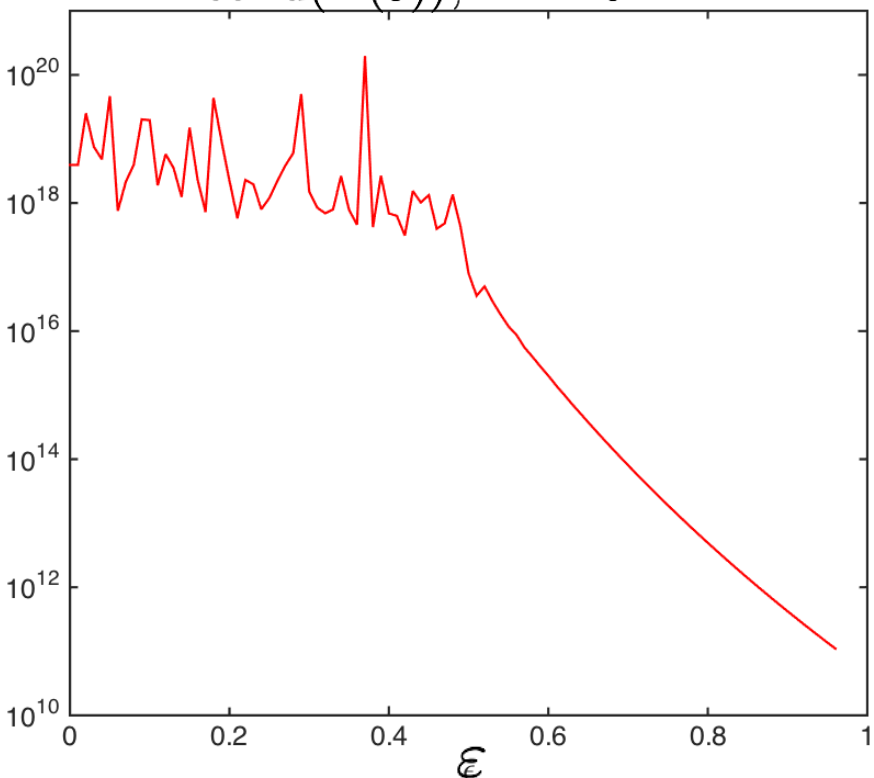
Issue interpolating with a smooth RBF

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

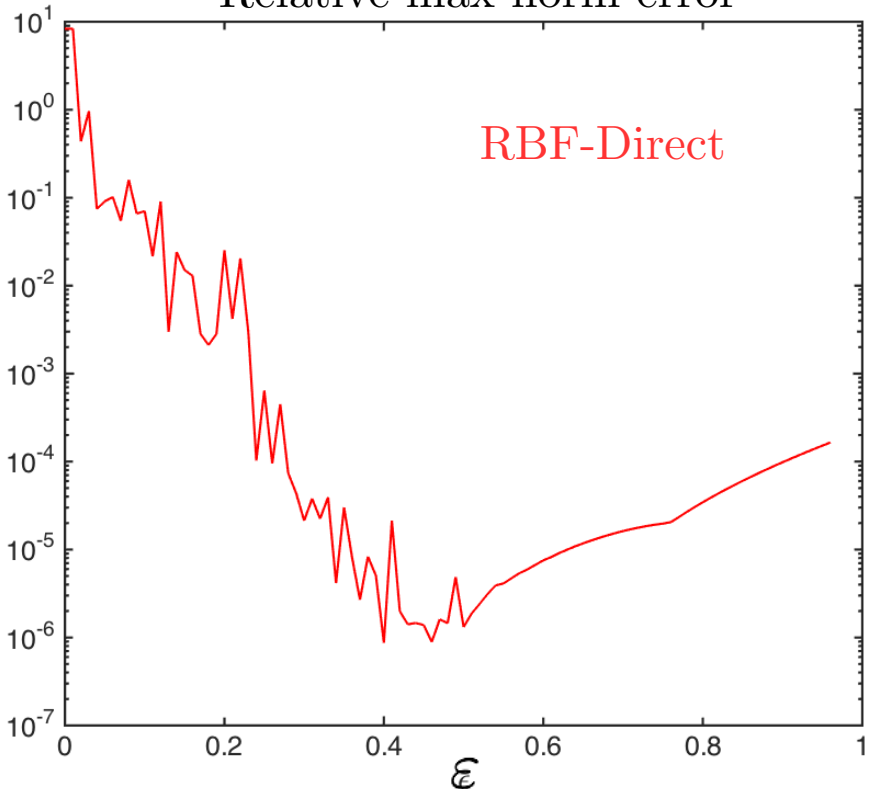
Issue: Effect of decreasing ε leads to severe ill-conditioning of interp. matrices



$\text{cond}(A(\varepsilon)),\ N = 62$



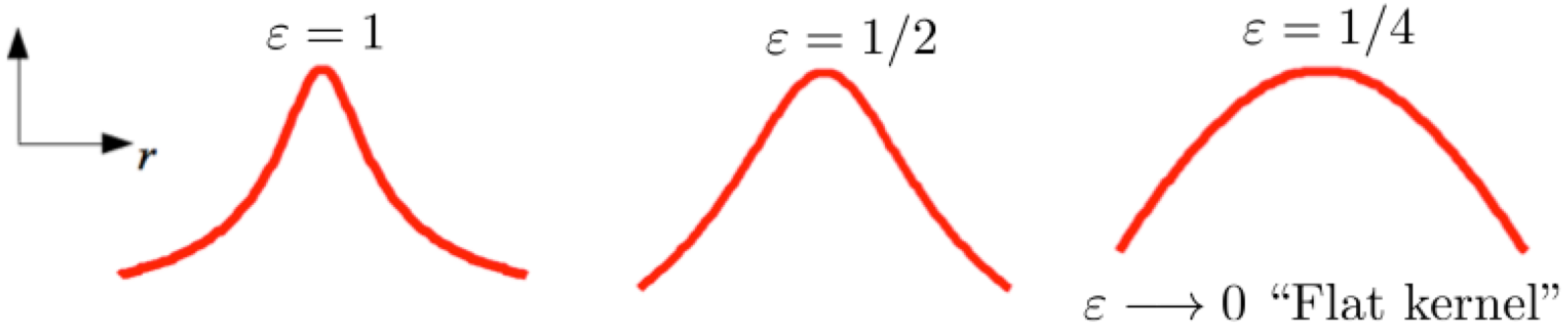
Relative max norm error

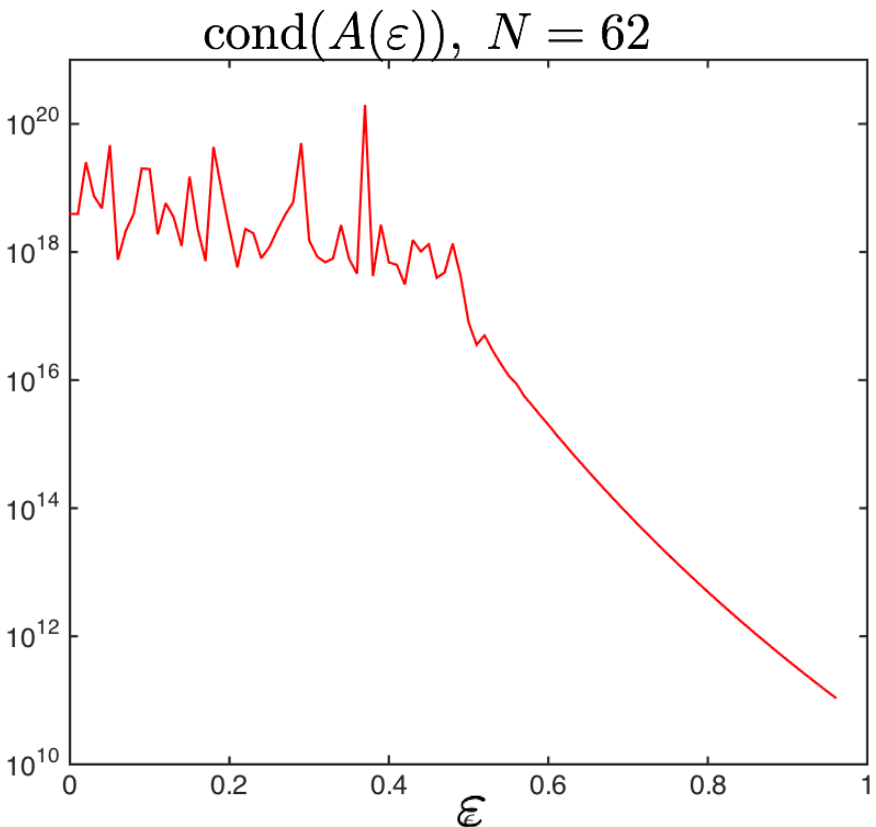


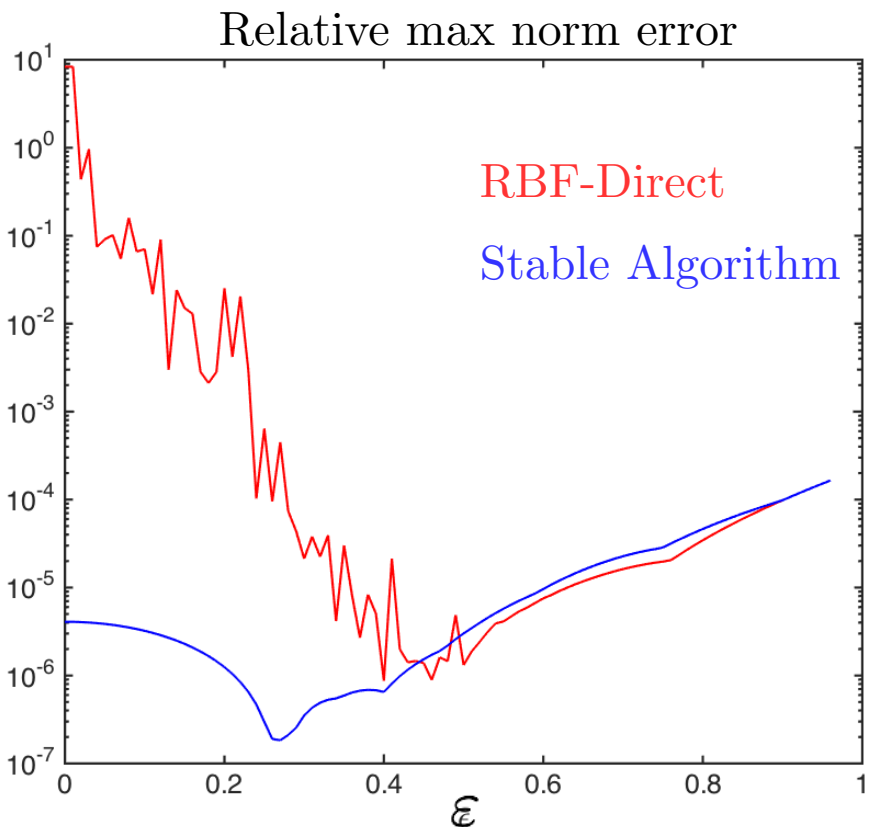
Issue interpolating with a smooth RBF

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







Stable algorithms

- Literature

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)
Micchelli, Lee & Yoon (2015)

Algorithms

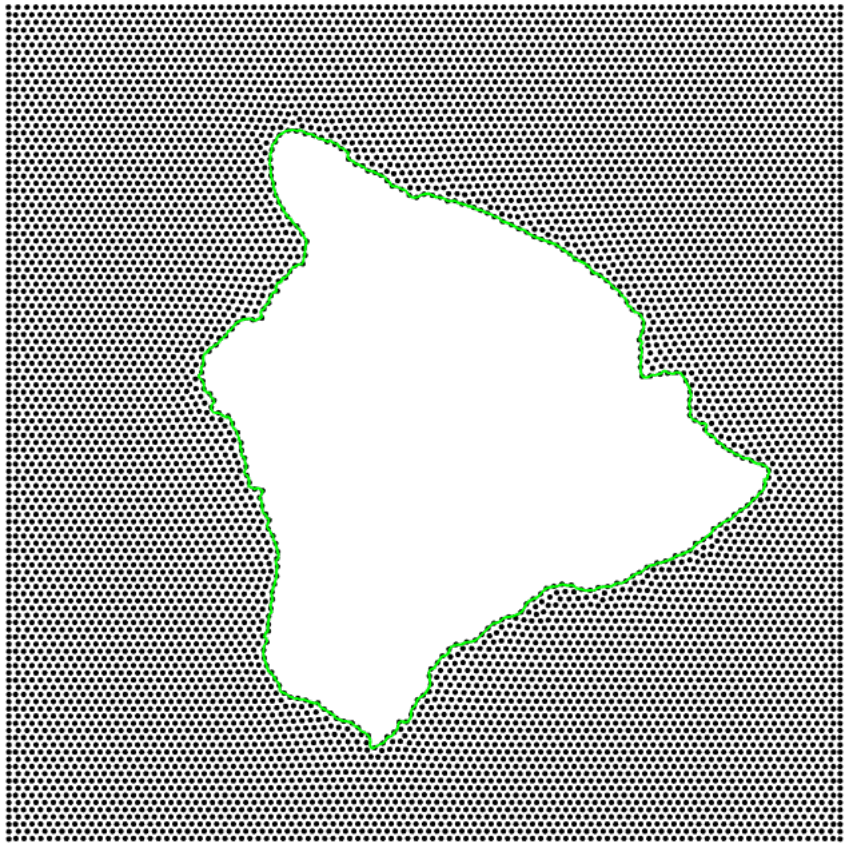
Fornberg & Wright (2004)
Fornberg & Piret (2007)
Fornberg, Larsson, & Flyer (2011)
Fasshauer & McCourt (2012)
Gonnet, Pachon, & Trefethen (2011)
Pazouki & Schaback (2011)
De Marchi & Santin (2013; 2014)
Fornberg, Letho, Powell (2013)
Larsson, Lehto, Heryudono, Fornberg (2013)
Cavoretto, Fasshauer, McCourt (2015)
Gonzalez-Rodriguez, Bayona, Moscoso & Kindelan (2015)

- Highlights of the present algorithm
  - Similarities to Contour-Padé method (RBF-CP).
  - Applies to many types of smooth kernels not just Gaussians.
  - Can be used for generalized interpolation problems

    Hermite interpolation, divergence-free & curl-free interpolation, etc.
  - Limited to relatively small node sets ($N \leq 100$ in 2D and $N \leq 400$ in 3D)

    Poses few problems for the RBF partition-of-unity and RBF finite-difference methods

Illustration RBF partition-of-unity method



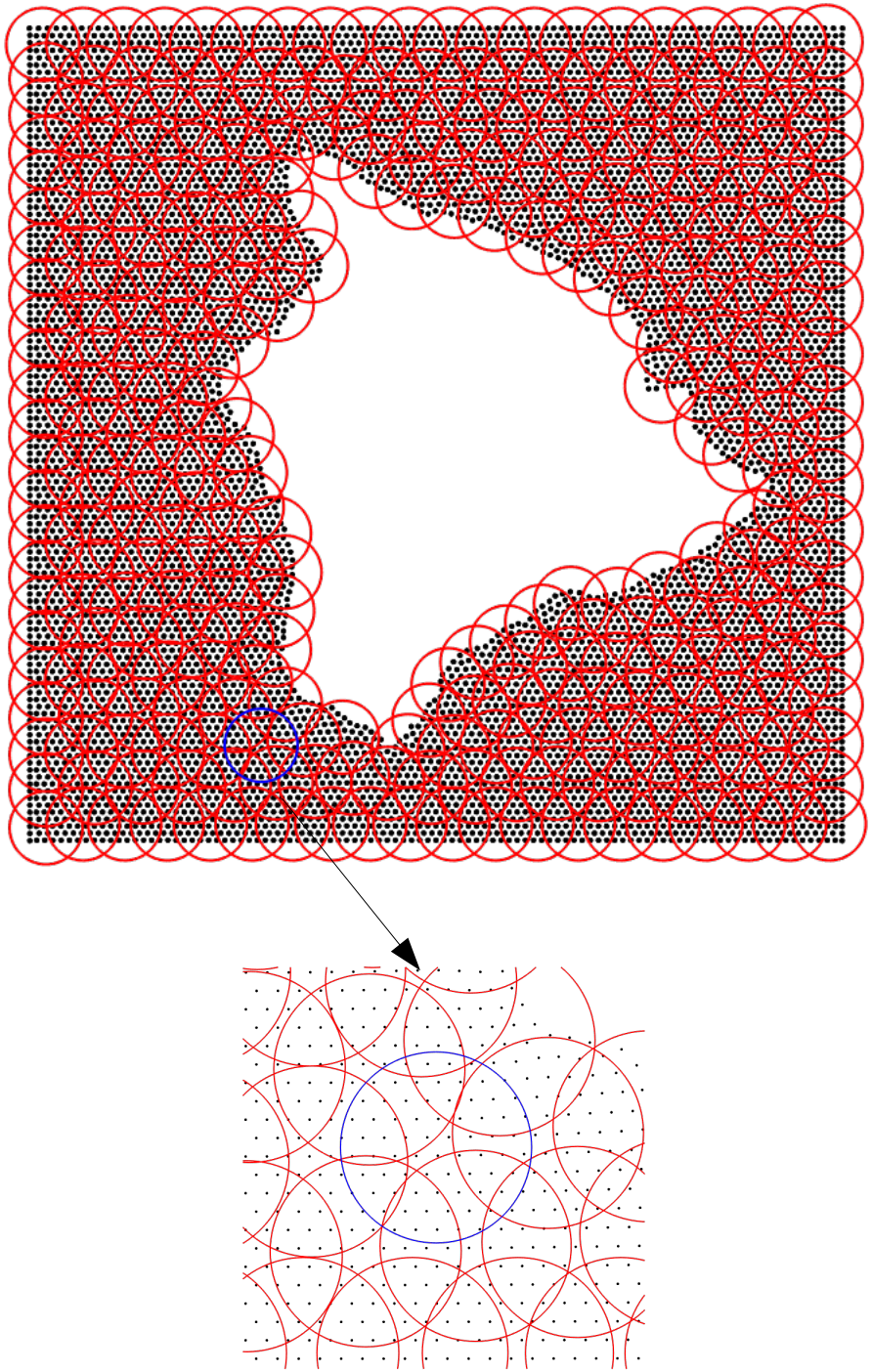
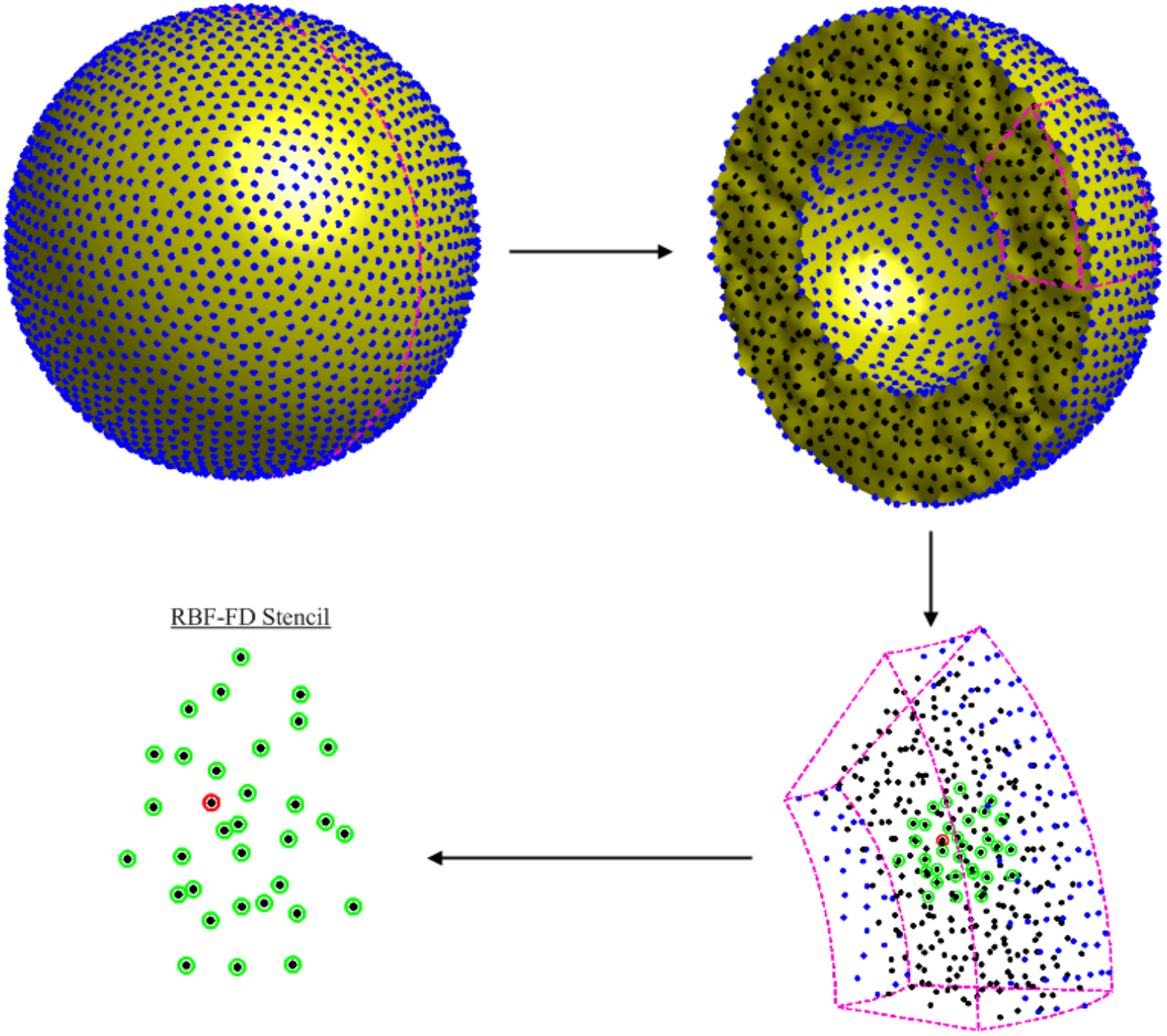
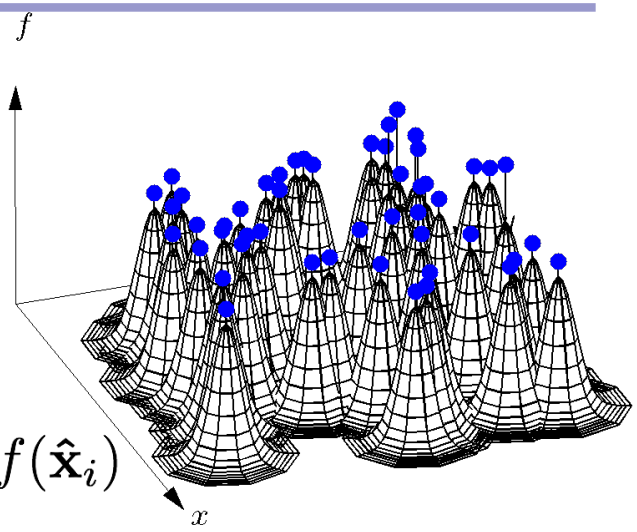


Illustration of the RBF finite-difference method



Notation



Nodes/centers: $\widehat{X} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N\}$

Target: $f\Big|_{\widehat{X}} = \{f(\hat{\mathbf{x}}_1), \ldots, f(\hat{\mathbf{x}}_N)\}$

Interpolant: $s(\mathbf{x}, \varepsilon) = \sum_{i=1}^{N} \lambda_i \phi_\varepsilon(\|\mathbf{x} - \hat{\mathbf{x}}_i\|), \;\; s(\hat{\mathbf{x}}_i, \varepsilon) = f(\hat{\mathbf{x}}_i)$

Ex. ϕ: GA: $\phi_\varepsilon(r) = e^{-(\varepsilon r)^2}$ MQ: $\phi_\varepsilon(r) = \sqrt{1 + (\varepsilon r)^2}$

Linear system: $A(\varepsilon)\, \boldsymbol{\lambda}(\varepsilon) = f\Big|_{\widehat{X}}, \;\; A(\varepsilon)_{i,j} = \phi_\varepsilon(\|\mathbf{x}_i - \hat{\mathbf{x}}_j\|)$

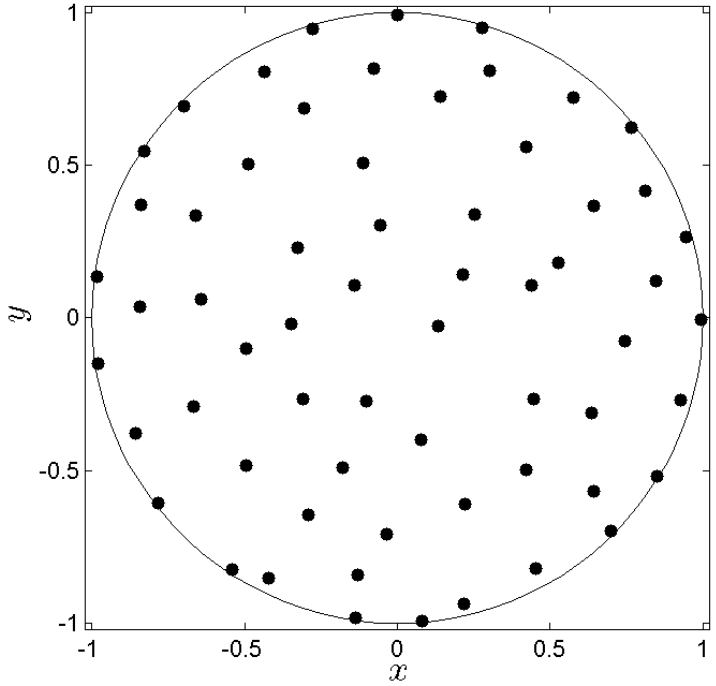
Eval. points: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$

RBF Direct: $s(\mathbf{x}_j, \varepsilon) = \sum_{i=1}^{N} \lambda_i \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_i\|)$

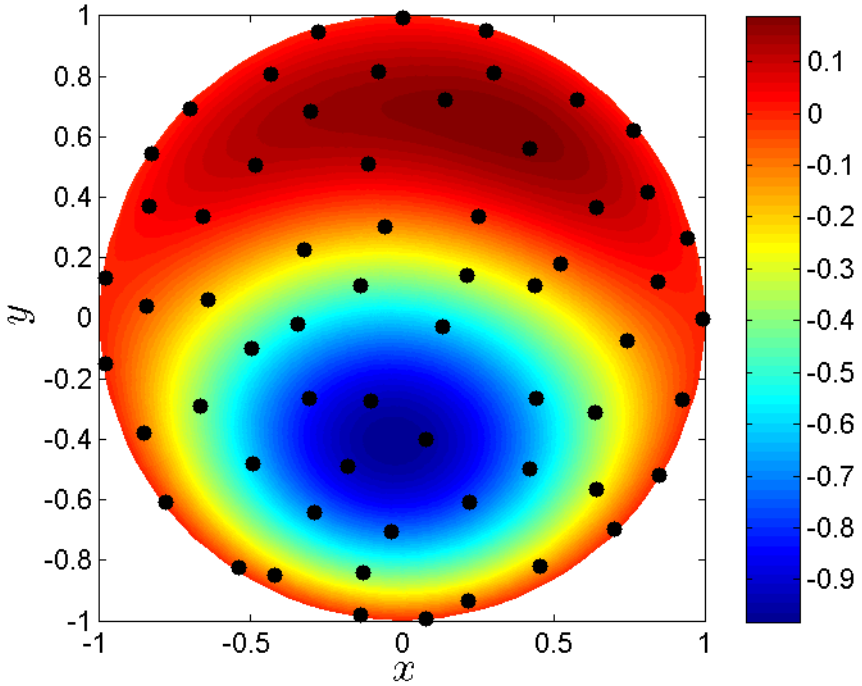
$$= \begin{bmatrix} \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_1\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_N\|) \end{bmatrix} \begin{bmatrix} & A(\varepsilon)^{-1} & \end{bmatrix} \begin{bmatrix} f(\hat{\mathbf{x}}_1) \\ \vdots \\ f(\hat{\mathbf{x}}_N) \end{bmatrix}$$

Example problem

Nodes $\widehat{X}$, $N = 62$

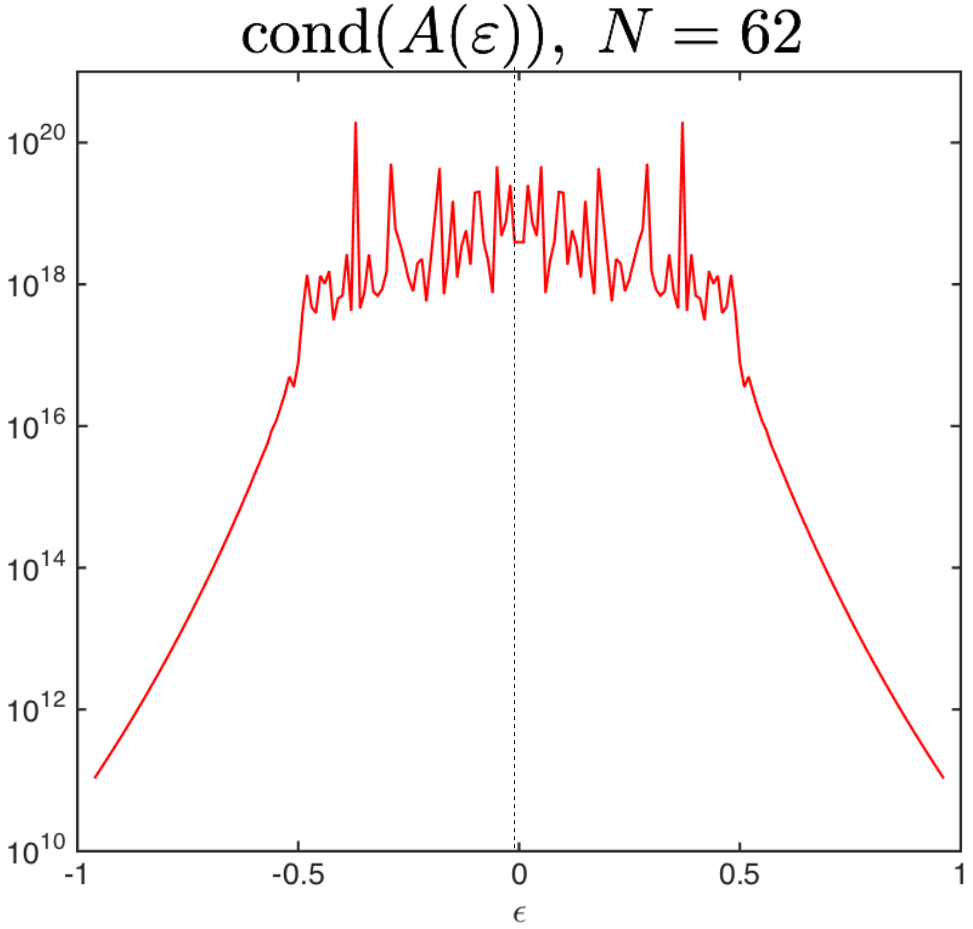


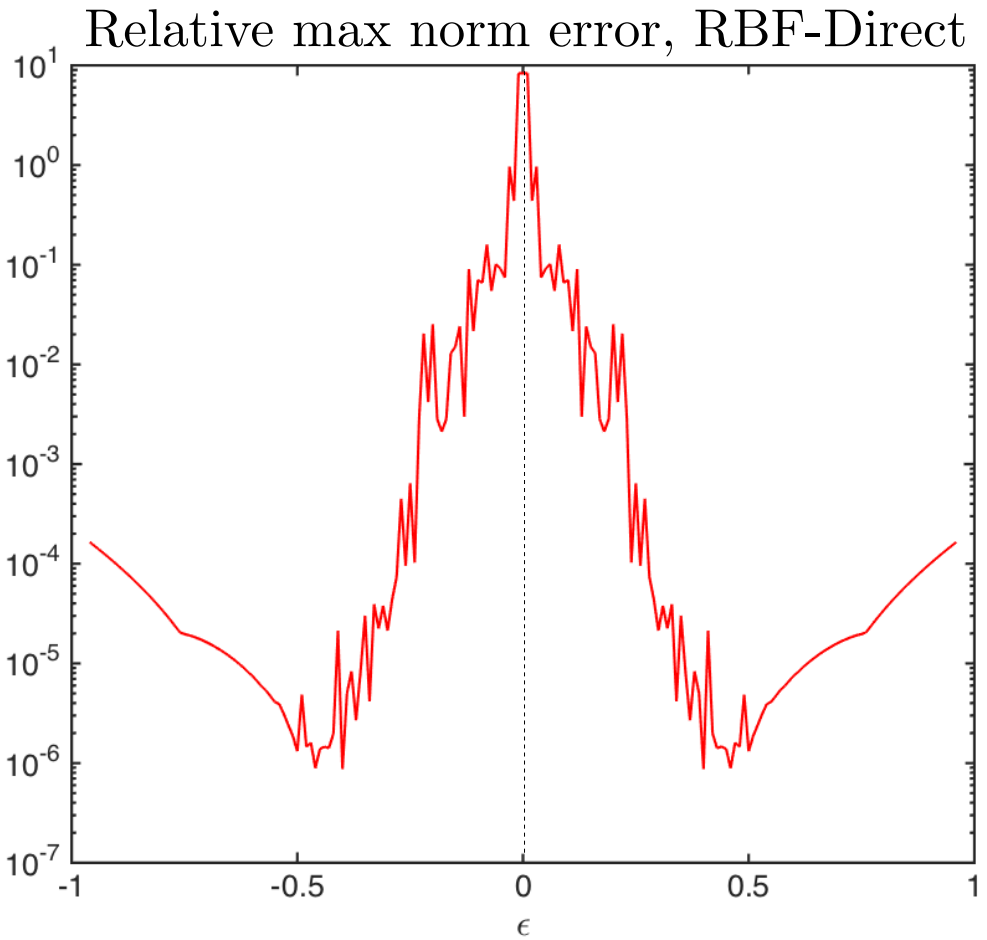
Target function



$$f(\mathbf{x}) = (1 - (x^2 + y^2)) \left[\sin\left(\frac{\pi}{2}(y - 0.07)\right) - \frac{1}{2}\cos\left(\frac{\pi}{2}(x + 0.1)\right) \right]$$

Different view of ill-conditioning issue



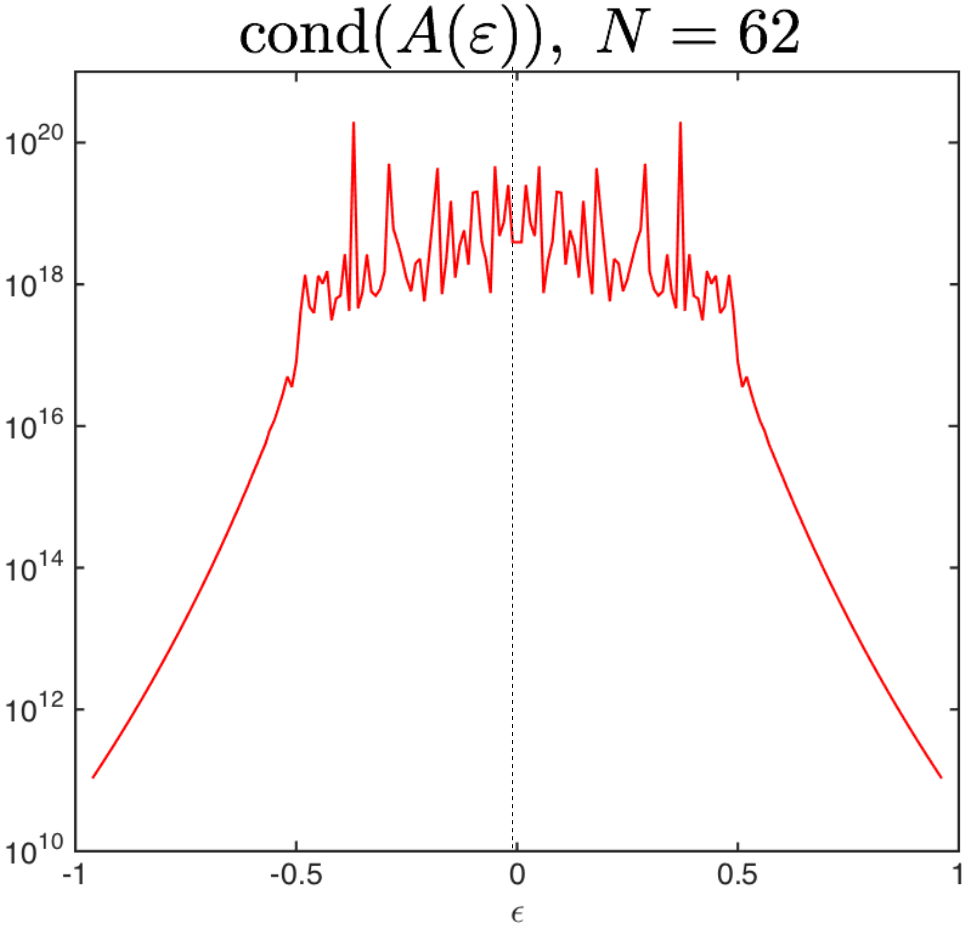


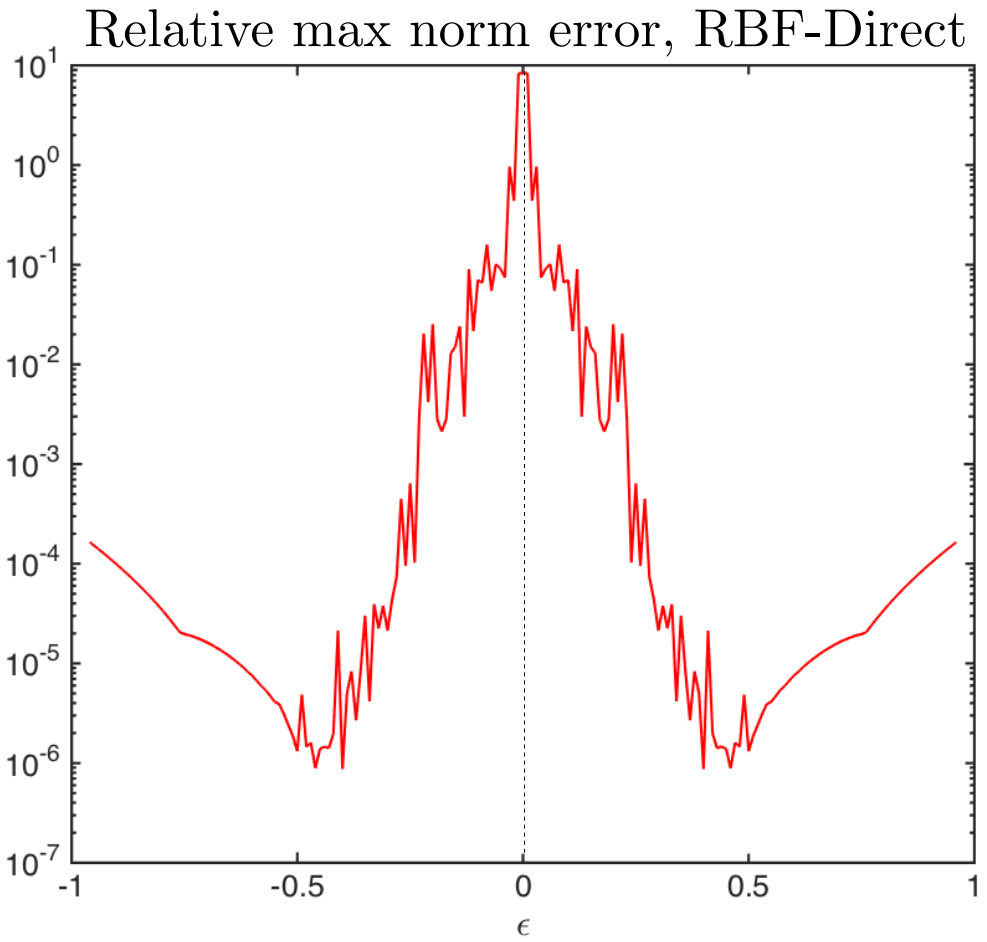
- For this example: $\text{cond}(A(\varepsilon)) = \mathcal{O}(|\varepsilon|^{-22})$
- In general for N scattered nodes in 2-D (Fornberg, Lehto, Powell 2013):

$$\text{cond}(A(\varepsilon)) = \mathcal{O}(|\varepsilon|^{-2\lfloor(\sqrt{8N-7}-1)/2\rfloor})$$

Since $s(\mathbf{x}, \varepsilon) = \mathcal{O}(1)$ a vast amount of numerical cancelation must arise in RBF-Direct

Different view of ill-conditioning issue



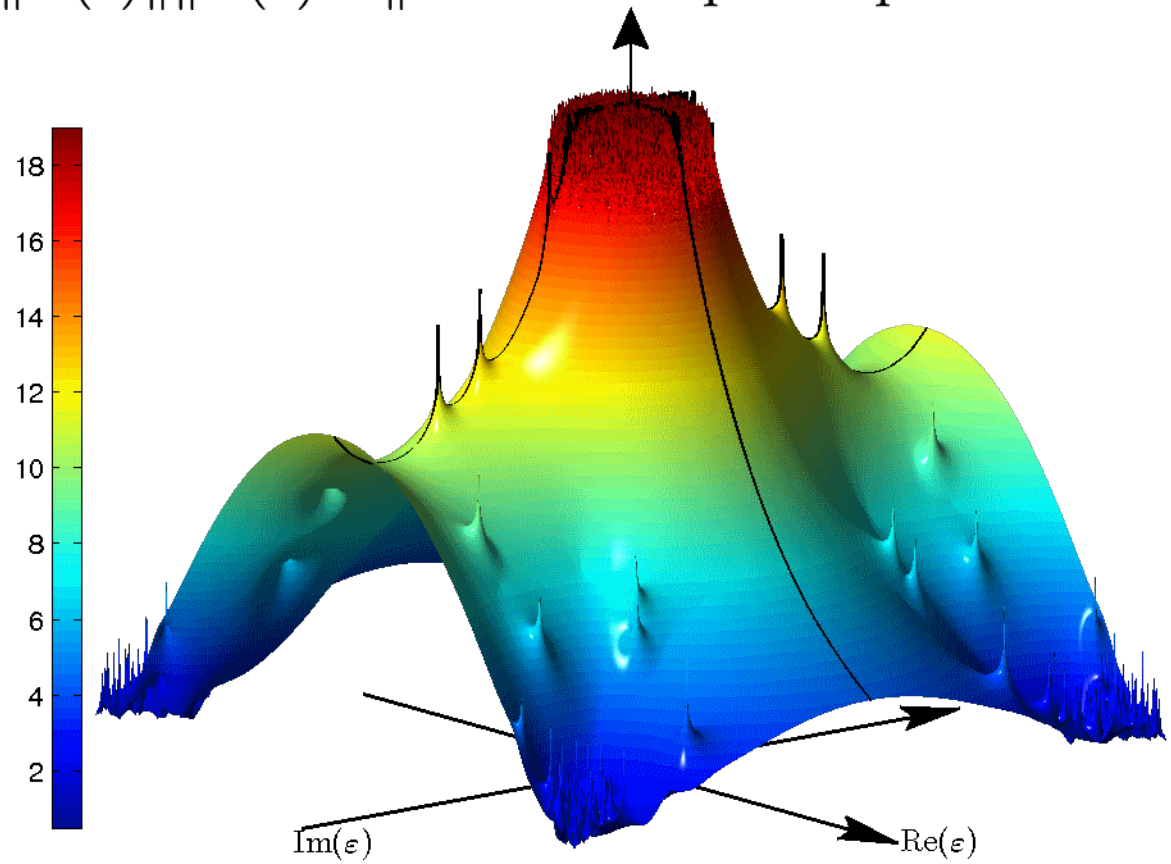


The shortest path between two truths in the real domain passes through the complex domain.
Jacques Hadamard

- Idea: Consider ε as a complex variable.
- Key: All smooth ϕ_ε are analytic functions in ε.

RBF interpolation from an analytic function view

- $\mathrm{cond}(A(\varepsilon)) = \|A(\varepsilon)\|\|A(\varepsilon)^{-1}\|$ in the complex ε-plane.



- Four-fold symmetry follows from $A(\varepsilon) = A(-\varepsilon) = \overline{A(\bar{\varepsilon})} = \overline{A(-\bar{\varepsilon})}$.

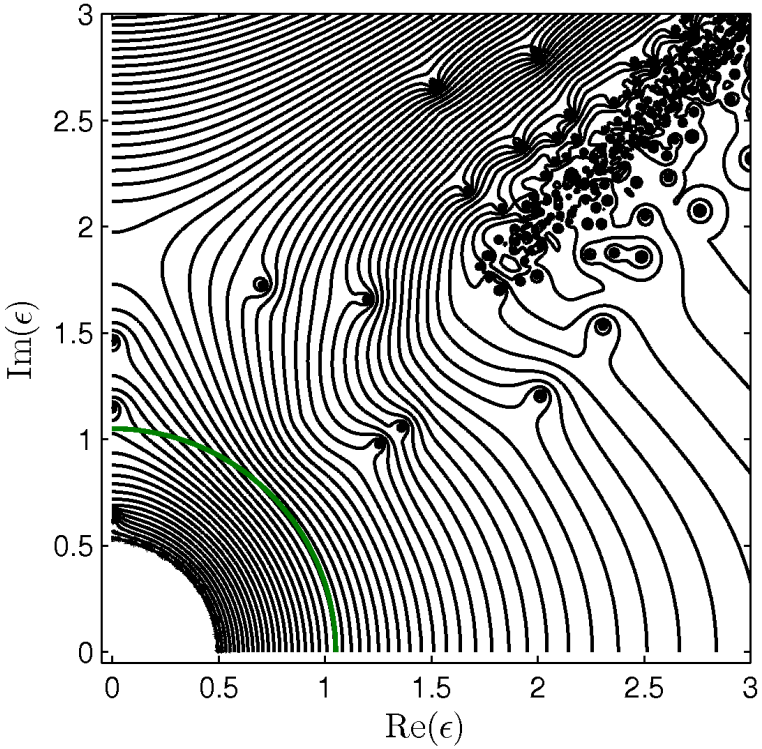
- Can restrict study of $s(\mathbf{x}, \varepsilon)$ to the first quadrant.

The main idea

Problem: Evaluate $s(\mathbf{x}_j, \varepsilon) = \begin{bmatrix} \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_1\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_N\|) \end{bmatrix} \begin{bmatrix} & A(\varepsilon)^{-1} & \end{bmatrix} \begin{bmatrix} f(\hat{\mathbf{x}}_1) \\ \vdots \\ f(\hat{\mathbf{x}}_N) \end{bmatrix}$.

Difficulty: Numerical cancellation.

Solution: Think of $s(\mathbf{x}_j, \varepsilon)$ as a function of a complex variable ε.



Then $s(\mathbf{x}_j, \varepsilon)$ is meromorphic and $\varepsilon = 0$ is (typically) just a removable singularity.

Idea: Approximate $s(\mathbf{x}_j, \varepsilon)$ using a rational function:

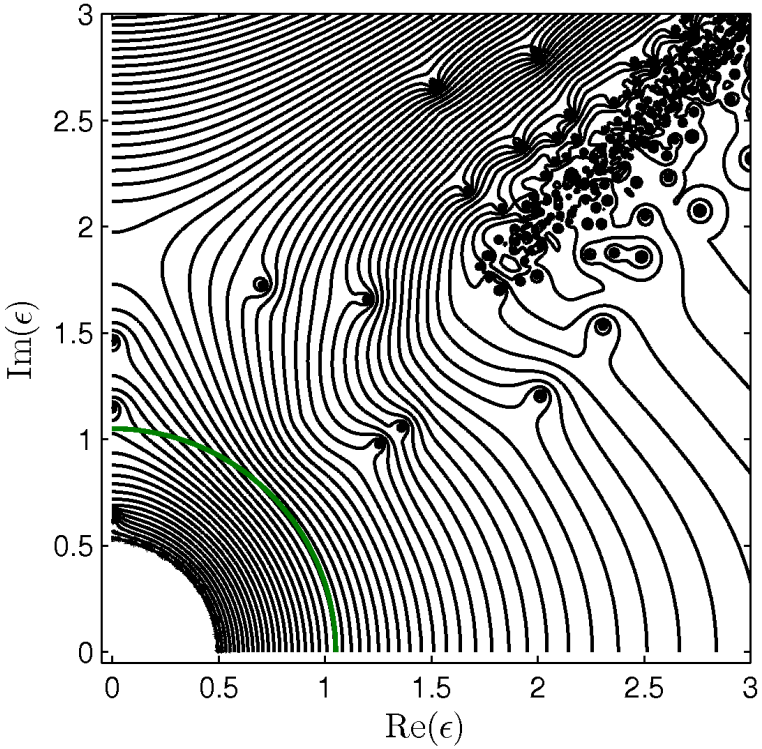
$$s(\mathbf{x}_j, \varepsilon) \approx r(\mathbf{x}_j, \varepsilon) = \frac{a_{0,j} + a_{1,j}\varepsilon^2 + a_{2,j}\varepsilon^4 + \cdots + a_{m,j}\varepsilon^{2m}}{1 + b_1\varepsilon^2 + b_2\varepsilon^4 + \cdots b_n\varepsilon^{2n}}.$$

The main idea

Problem: Evaluate $s(\mathbf{x}_j, \varepsilon) = \begin{bmatrix} \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_1\|) & \cdots & \phi_\varepsilon(\|\mathbf{x}_j - \hat{\mathbf{x}}_N\|) \end{bmatrix} \begin{bmatrix} & A(\varepsilon)^{-1} & \end{bmatrix} \begin{bmatrix} f(\hat{\mathbf{x}}_1) \\ \vdots \\ f(\hat{\mathbf{x}}_N) \end{bmatrix}$.

Difficulty: Numerical cancellation.

Solution: Think of $s(\mathbf{x}_j, \varepsilon)$ as a function of a complex variable ε.



Fundamental observation

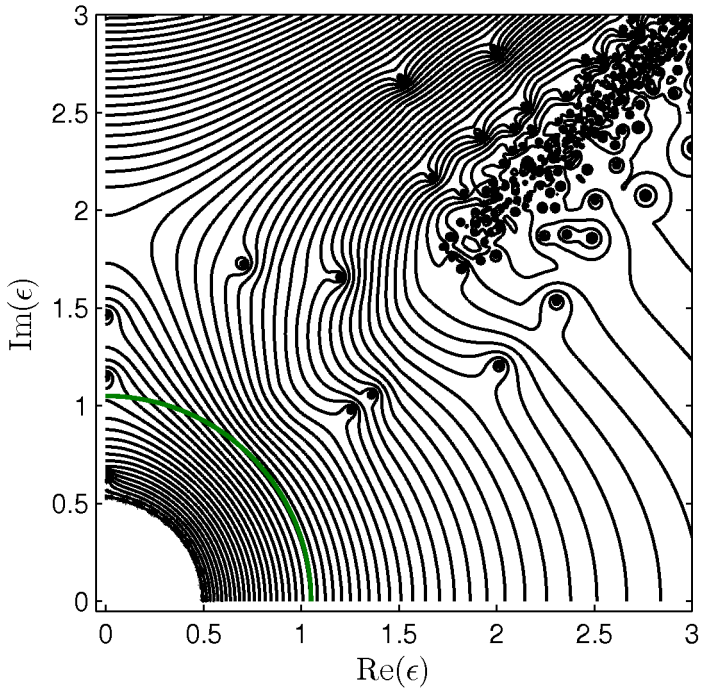
Denominator is independent of x_j

Idea: Approximate $s(\mathbf{x}_j, \varepsilon)$ using a rational function:

$$s(\mathbf{x}_j, \varepsilon) \approx r(\mathbf{x}_j, \varepsilon) = \frac{a_{0,j} + a_{1,j}\varepsilon^2 + a_{2,j}\varepsilon^4 + \cdots + a_{m,j}\varepsilon^{2m}}{1 + b_1\varepsilon^2 + b_2\varepsilon^4 + \cdots b_n\varepsilon^{2n}}.$$

Review of RBF-CP

1. Evaluate $s(\mathbf{x}_j, \varepsilon)$ at K equally spaced locations around the circular contour.



2. Take FFT of result to get Laurent expansion for $s(\mathbf{x}_j, \varepsilon)$.

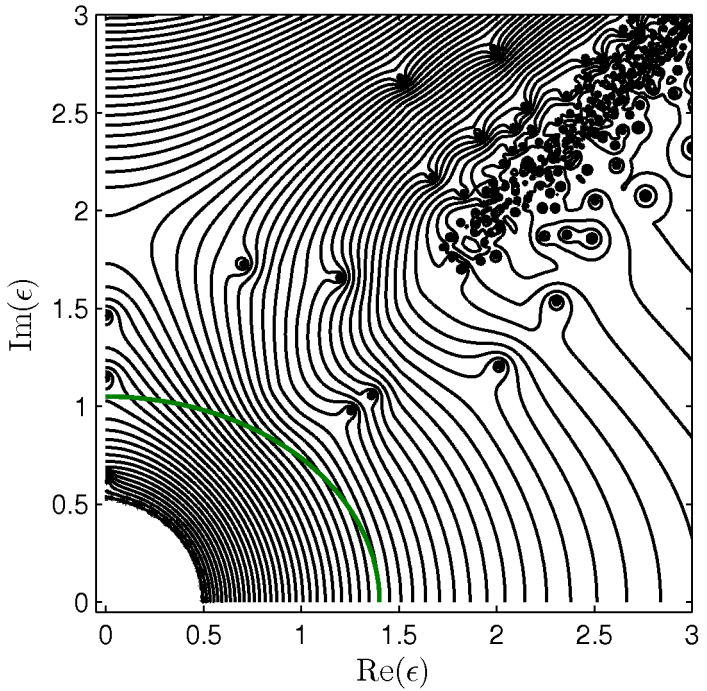
3. Transform the negative powers of ε^2 into a rational function using Padé approximation:

$$s(\mathbf{x}_j, \varepsilon) \approx r(\mathbf{x}_j, \varepsilon) = a_{0,j} + a_{1,j}\varepsilon^2 + \cdots + a_{m,j}\varepsilon^{2m} + \underbrace{\frac{1}{1 + b_1\varepsilon^2 + \cdots b_n\varepsilon^{2n}}}_{\text{Padé}}.$$

Difficulties: Choosing the contour and degree of denominator n.

RBF-RA Algorithm

1. Evaluate $s(\mathbf{x}_j, \varepsilon)$ at K locations around a safe contour in the upper half plane, $\{\varepsilon_1, \ldots, \varepsilon_K\}$.



Symmetry means only $K/2+1$ evaluations are necessary.

2. For each evaluation point $\mathbf{x}_j$, $j = 1, \ldots, M$, fit a rational function:

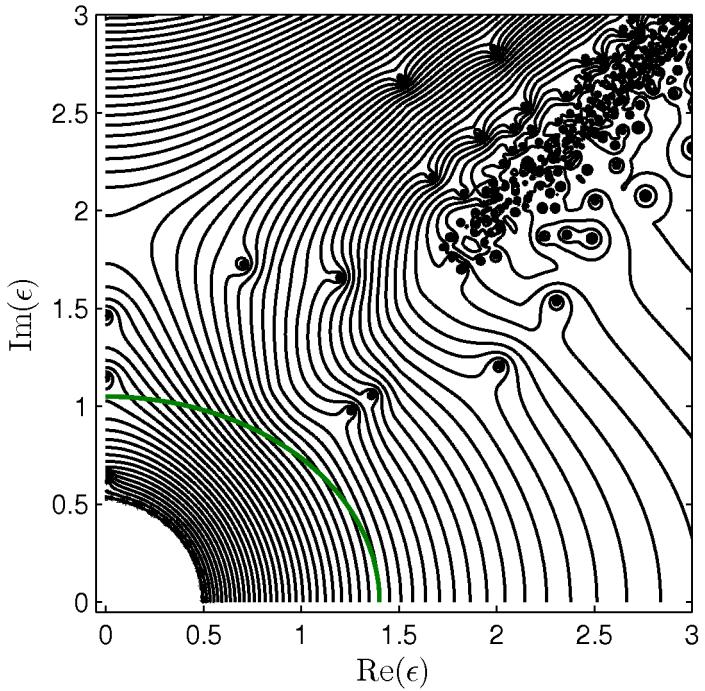
$$r(\mathbf{x}_j, \varepsilon) = \frac{a_{0,j} + a_{1,j}\varepsilon^2 + a_{2,j}\varepsilon^4 + \cdots + a_{m,j}\varepsilon^{2m}}{1 + b_1\varepsilon^2 + b_2\varepsilon^4 + \cdots b_n\varepsilon^{2n}}$$

This gives M linear systems to solve:

$$\underbrace{\begin{bmatrix} 1 & \varepsilon_1^2 & \cdots & \varepsilon_1^{2m} \\ 1 & \varepsilon_2^2 & \cdots & \varepsilon_2^{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varepsilon_K^2 & \cdots & \varepsilon_K^{2m} \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} a_{0,j} \\ \vdots \\ a_{m,j} \end{bmatrix}}_{\underline{a}_j} + \underbrace{\left(-\mathrm{diag}(\underline{s}_j) \begin{bmatrix} \varepsilon_1^2 & \cdots & \varepsilon_1^{2n} \\ \varepsilon_2^2 & \cdots & \varepsilon_2^{2n} \\ \vdots & \ddots & \vdots \\ \varepsilon_K^2 & \cdots & \varepsilon_K^{2n} \end{bmatrix} \right)}_{F_j} \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}}_{\underline{b}} = \underbrace{\begin{bmatrix} s(\mathbf{x}_j, \varepsilon_1) \\ s(\mathbf{x}_j, \varepsilon_2) \\ \vdots \\ s(\mathbf{x}_j, \varepsilon_K) \end{bmatrix}}_{\underline{s}_j}$$

RBF-RA Algorithm

1. Evaluate $s(\mathbf{x}_j, \varepsilon)$ at K locations around a safe contour in the upper half plane, $\{\varepsilon_1, \ldots, \varepsilon_K\}$.



Symmetry means only $K/2+1$ evaluations are necessary.

2. For each evaluation point $\mathbf{x}_j$, $j = 1, \ldots, M$, fit a rational function:

$$r(\mathbf{x}_j, \varepsilon) = \frac{a_{0,j} + a_{1,j}\varepsilon^2 + a_{2,j}\varepsilon^4 + \cdots + a_{m,j}\varepsilon^{2m}}{1 + b_1\varepsilon^2 + b_2\varepsilon^4 + \cdots b_n\varepsilon^{2n}}$$

This gives M linear systems to solve:

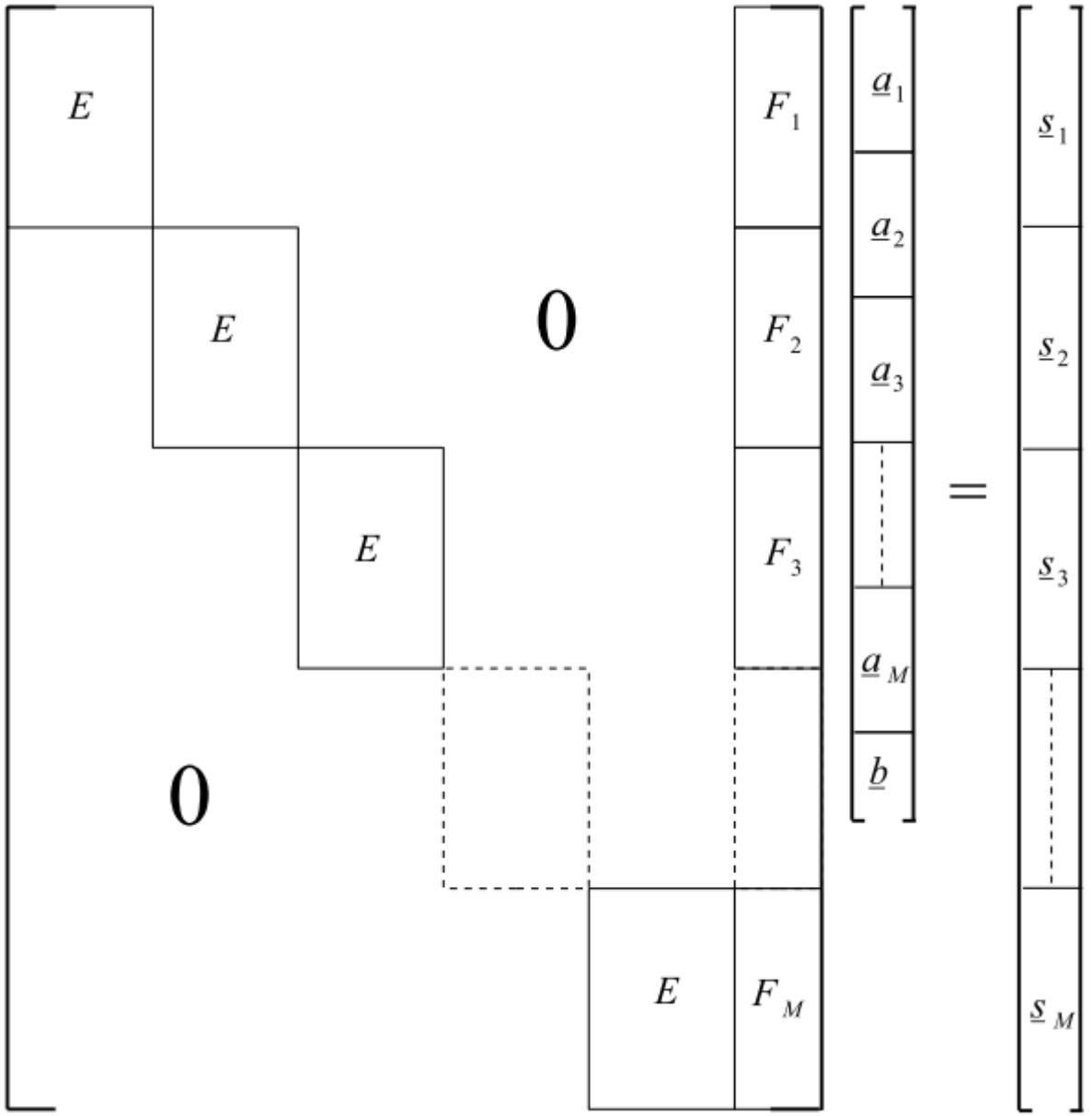
$$\Longrightarrow E\underline{a}_j + F\underline{b} = \underline{s}_j, \quad j = 1, \ldots, M$$

System is KM-by-$((m+1)M + n)$, and overdetermined when $m + 1 + n/M \leq K$.

Solving the overdetermined system

- Normalize any large rows among all blocks.

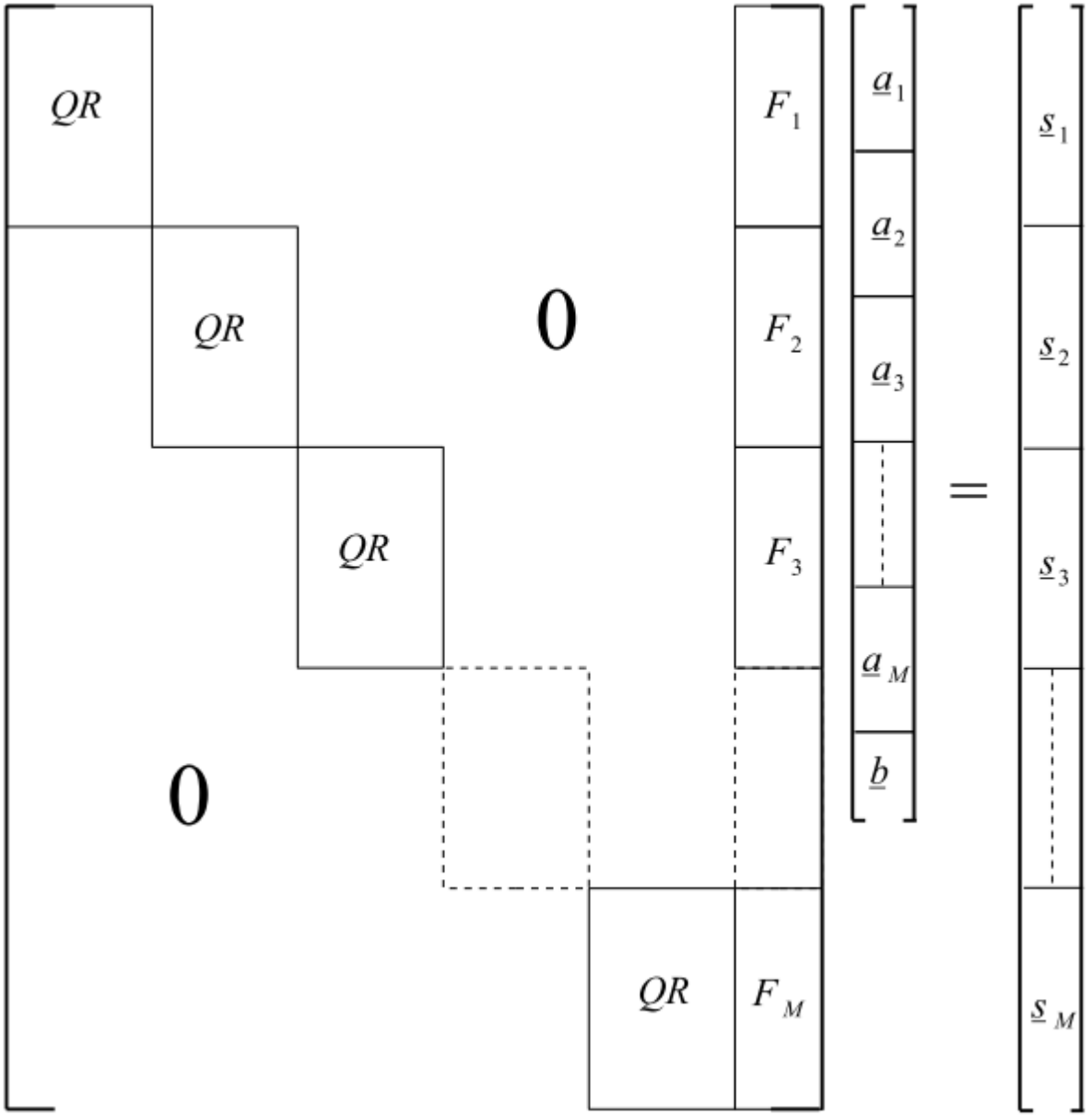
Linear system:



Solution procedure: step 2

- QR decompose E.

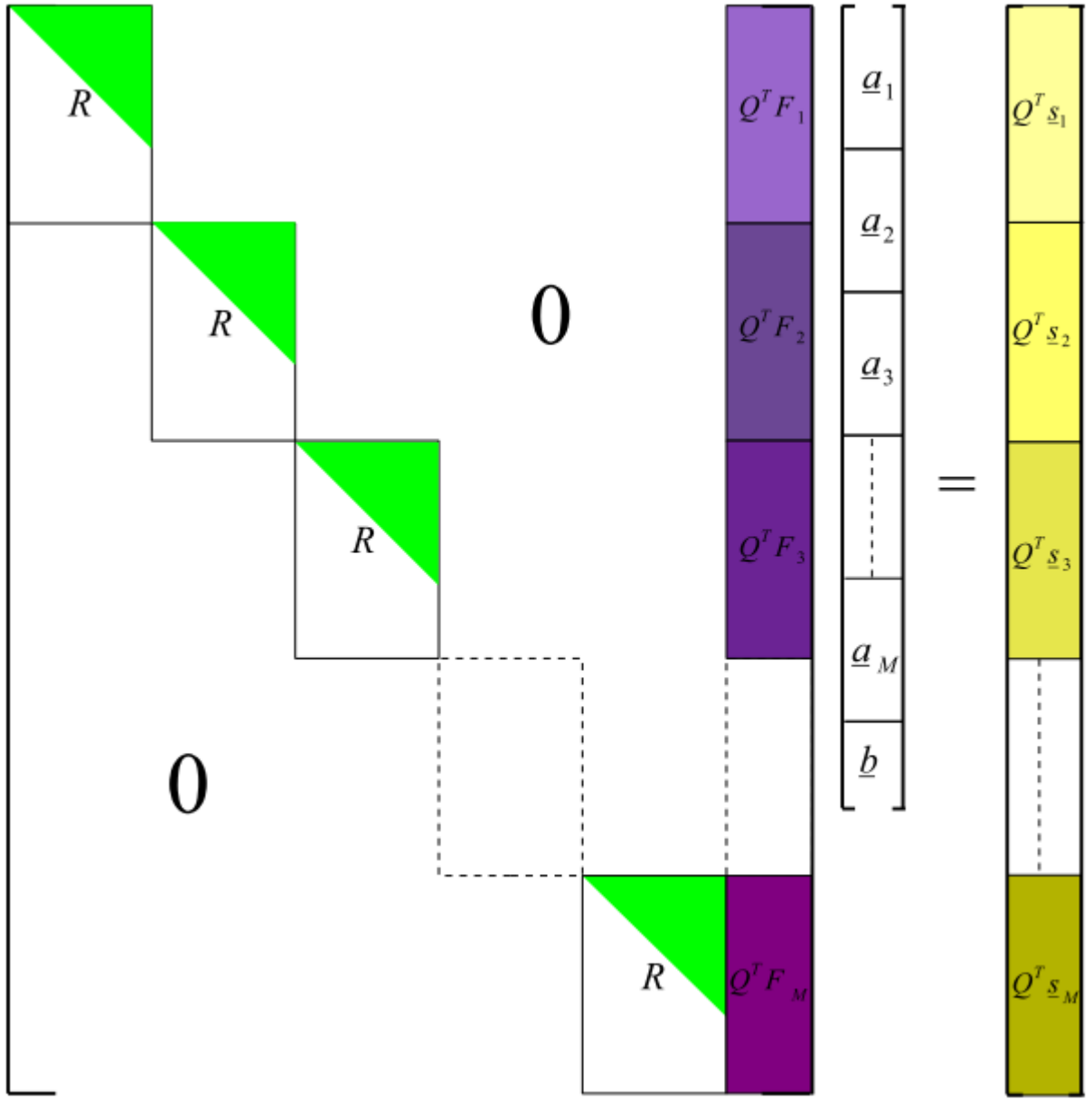
Linear system:



Solution procedure: step 3

- Move Q from R.

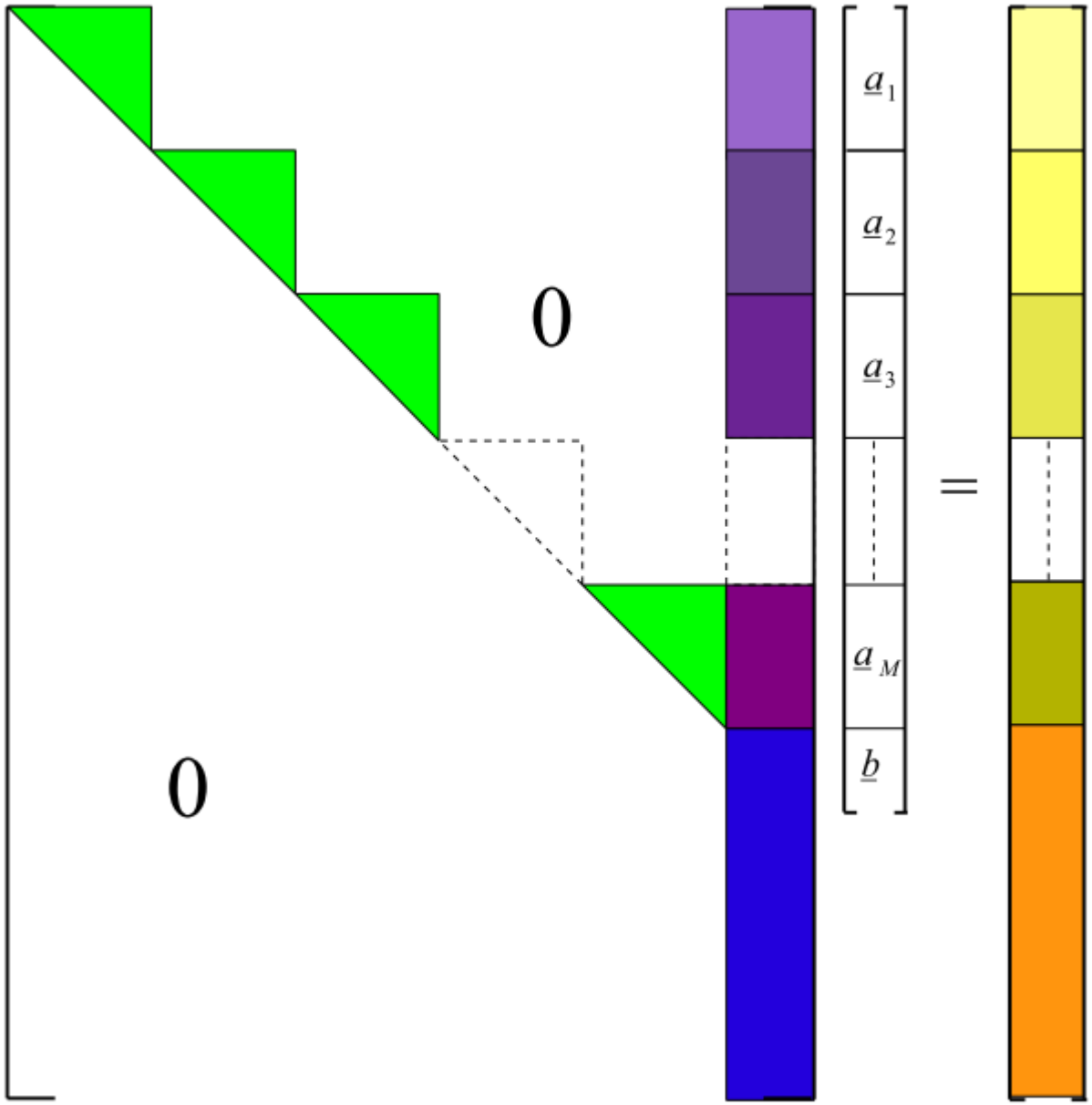
Linear system:



Solution procedure: step 4

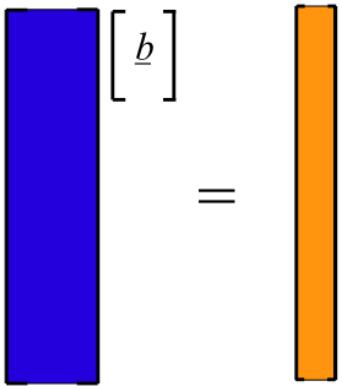
- Re-order the rows.

Linear system:

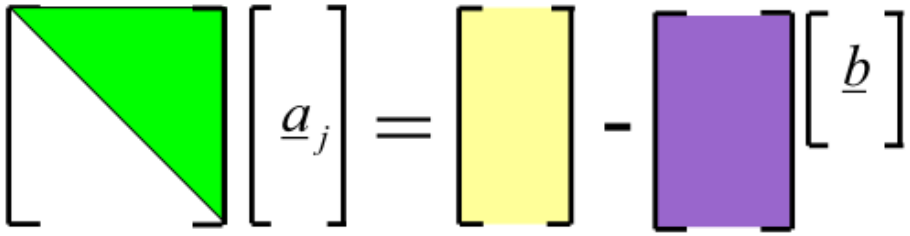


Solution procedure: steps 5 and 6

- Step 5: Solve for $\underline{b}$:



- Step 6: Determine $\underline{a}_1$, $\underline{a}_2$,...,$\underline{a}_M$:



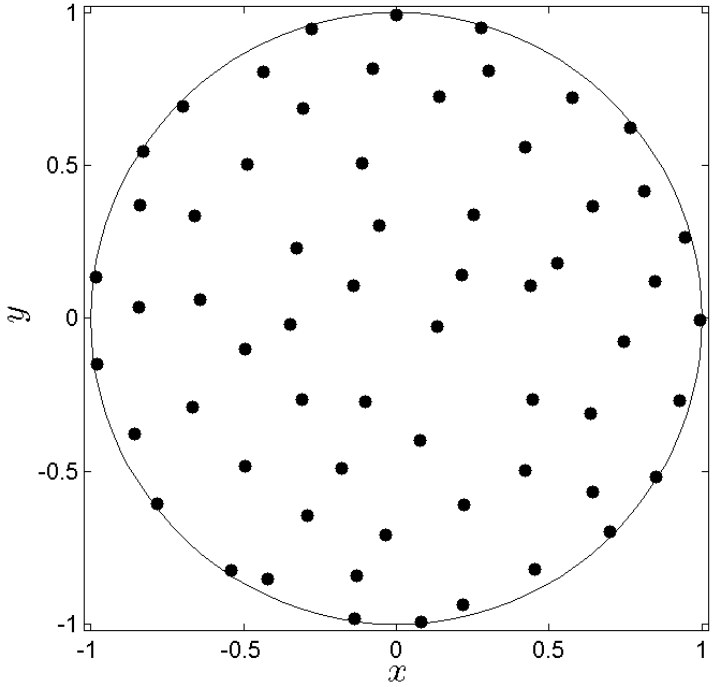
Note 1: One does not have use all the evaluations points to determine $\underline{b}$.

Note 2: We have found that choosing $n = \left\lfloor \frac{K}{4} \right\rfloor$ generally gives good results.

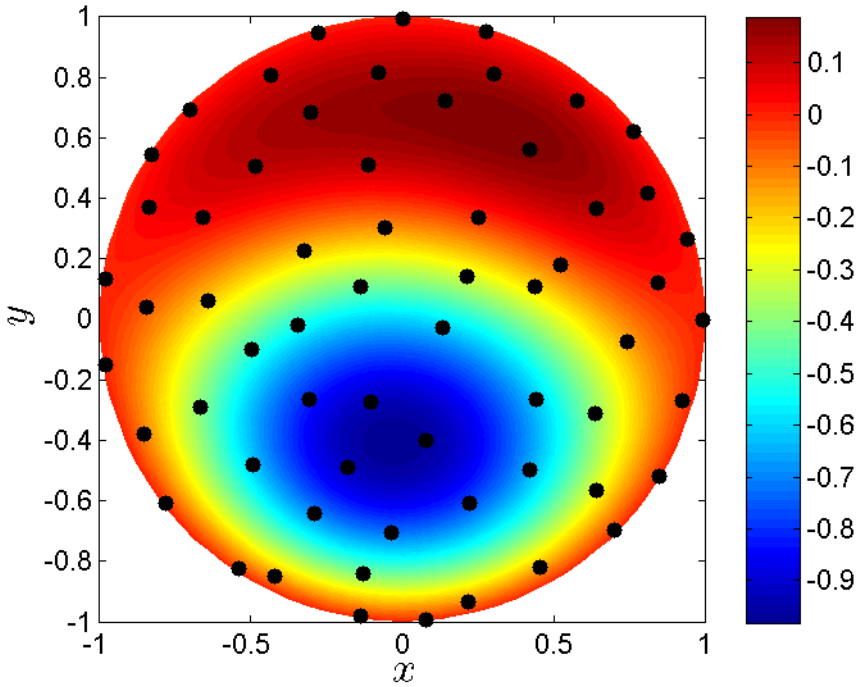
Note 3: We also choose $m = K - 1 - n = K - 1 - \left\lfloor \frac{K}{4} \right\rfloor$

Numerical results: test problem

Nodes $\widehat{X}$, $N = 62$



Target function

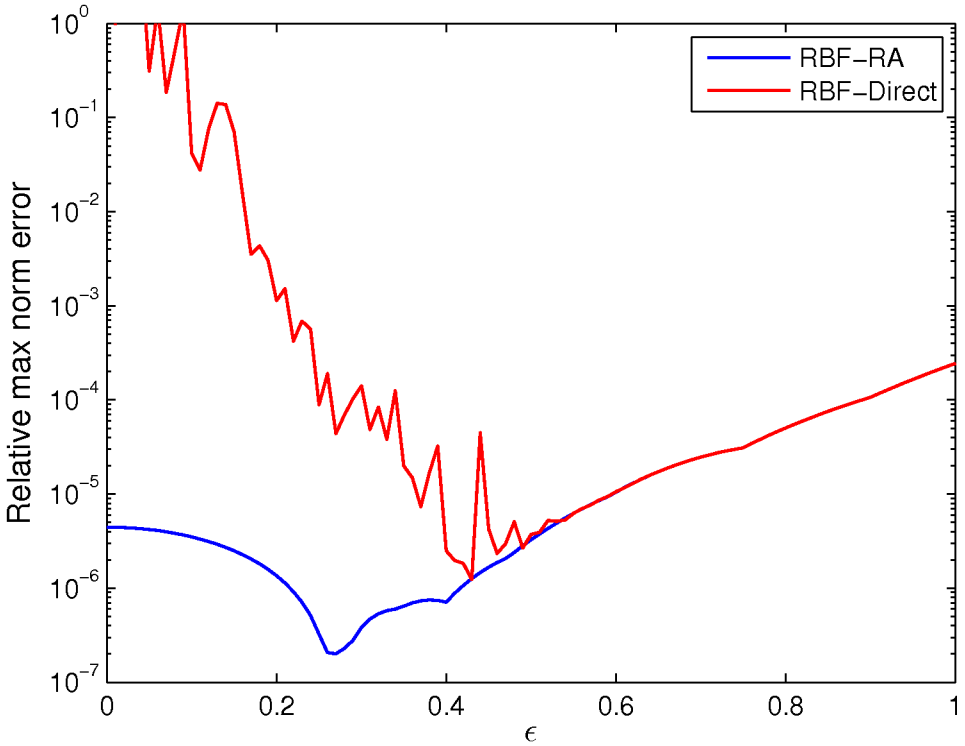


$$f(\mathbf{x}) = (1 - (x^2 + y^2)) \left[\sin\left(\frac{\pi}{2}(y - 0.07)\right) - \frac{1}{2}\cos\left(\frac{\pi}{2}(x + 0.1)\right) \right]$$

- All results are for the Gaussian RBF, and M=41 evaluation points.

Numerical results: reconstructing the target function

- Error in reconstructing $f(x,y)$ vs. shape parameter:

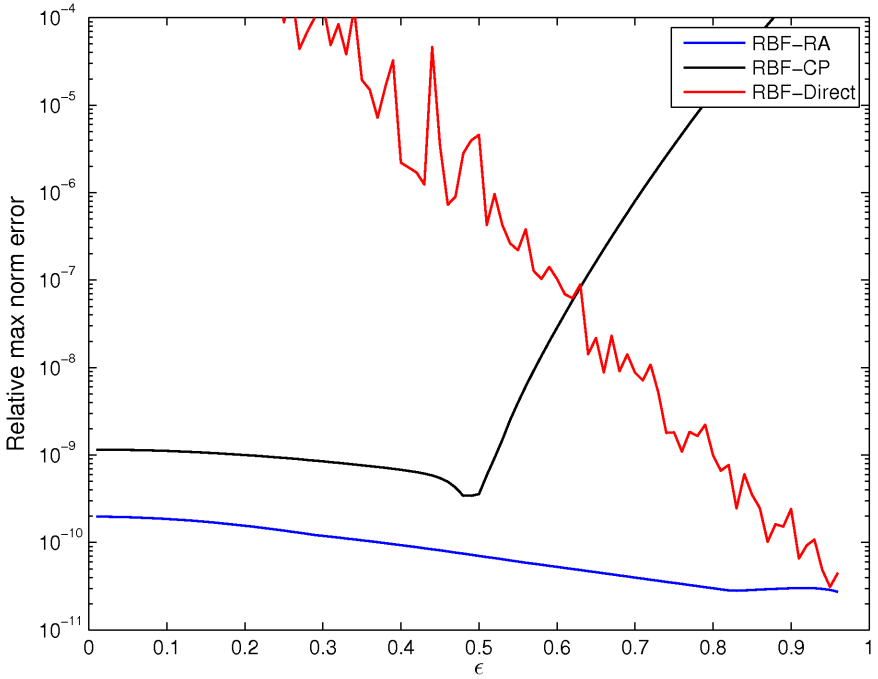


- The dip in the error at some small ε is typical.
- Rise in the error as ε approaches zero is due to Runge phenomenon type effects

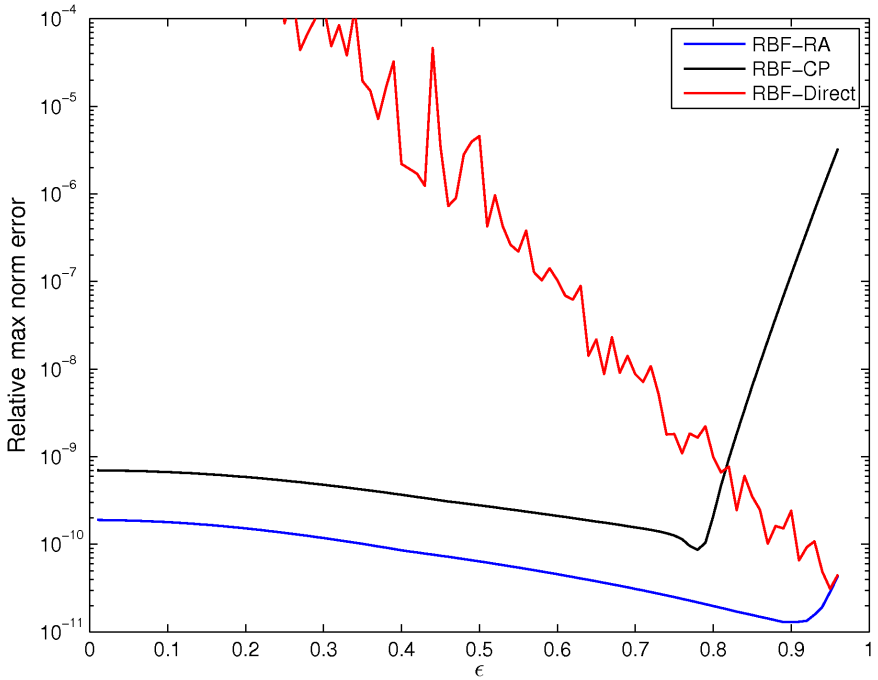
(Larsson & Fornberg 2005 and Fornberg & Zuev 2007))

Numerical results: error in the interpolant

- Results for $s(\mathbf{x},\varepsilon) - \widetilde{s}(\mathbf{x},\varepsilon)$, where $s(\mathbf{x},\varepsilon)$ is the exact interpolant



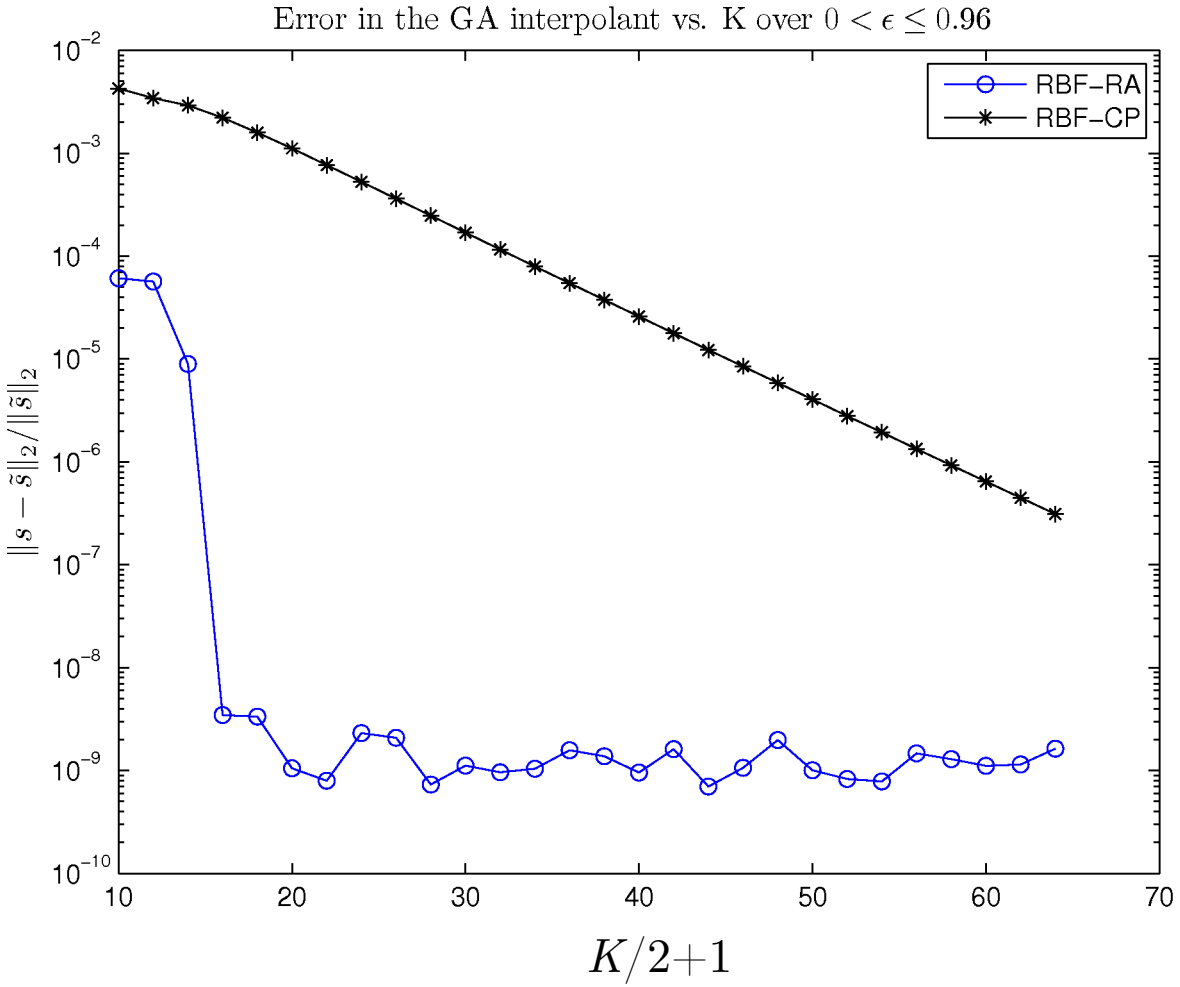
$K/2+1=20$



$K/2+1=50$

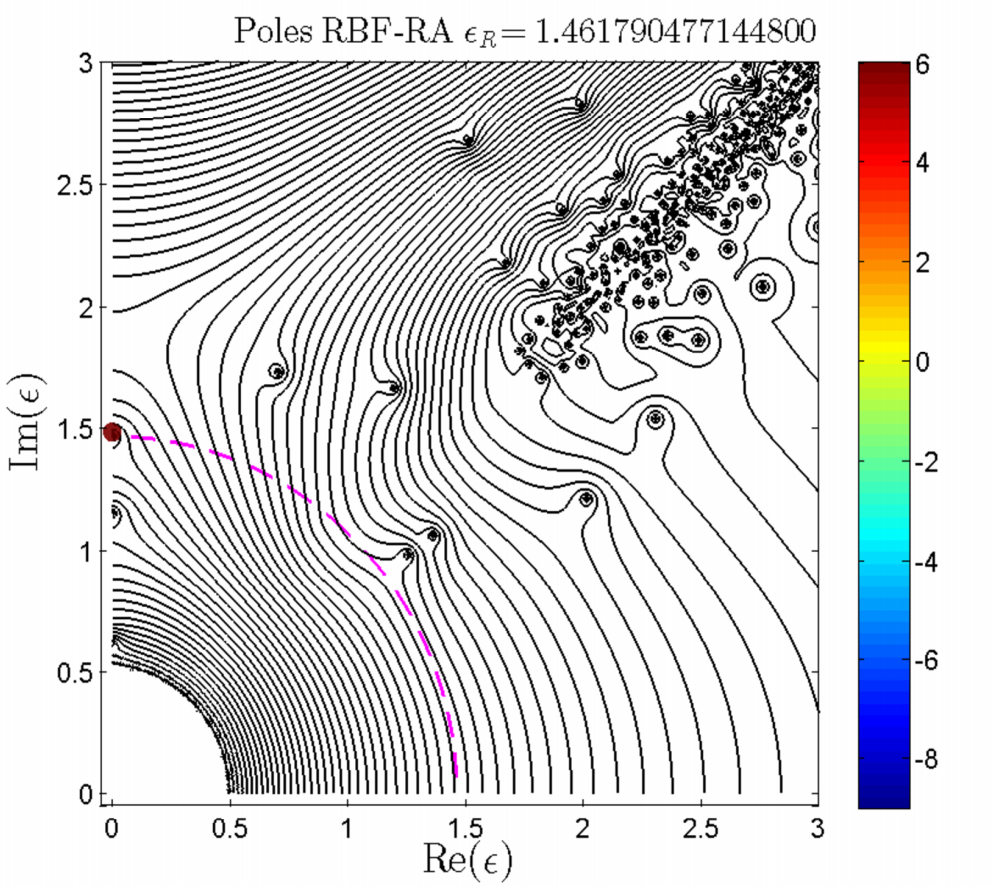
Numerical results: error in the interpolant

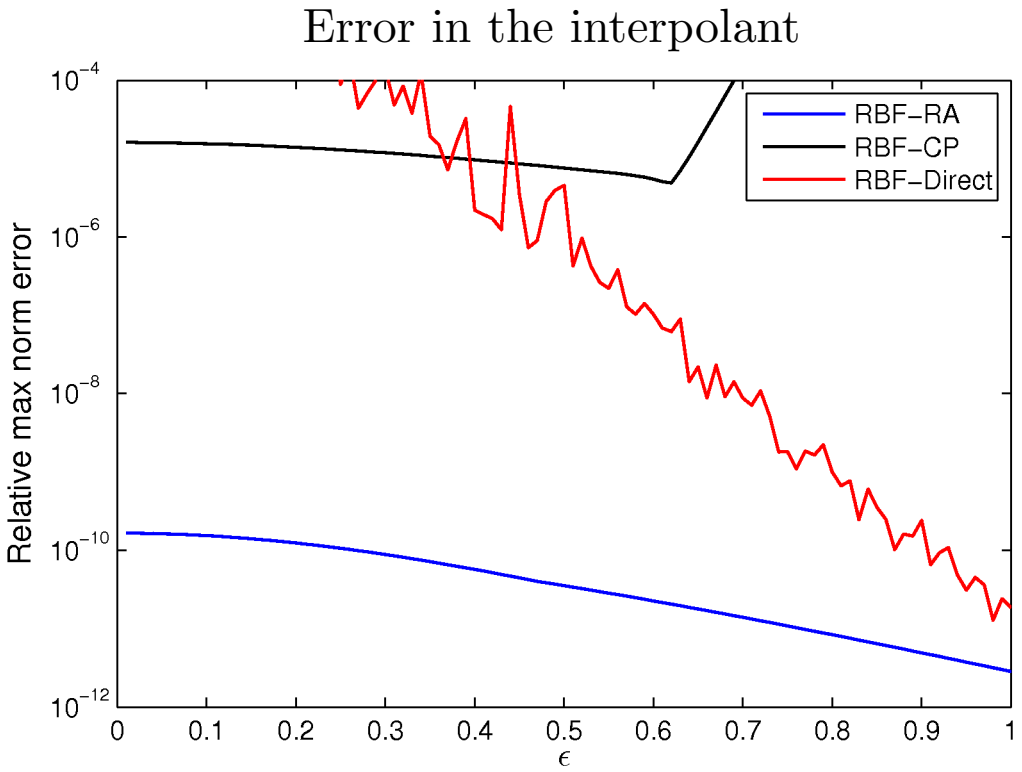
- Error in the interpolant vs. K (number of points on contour)



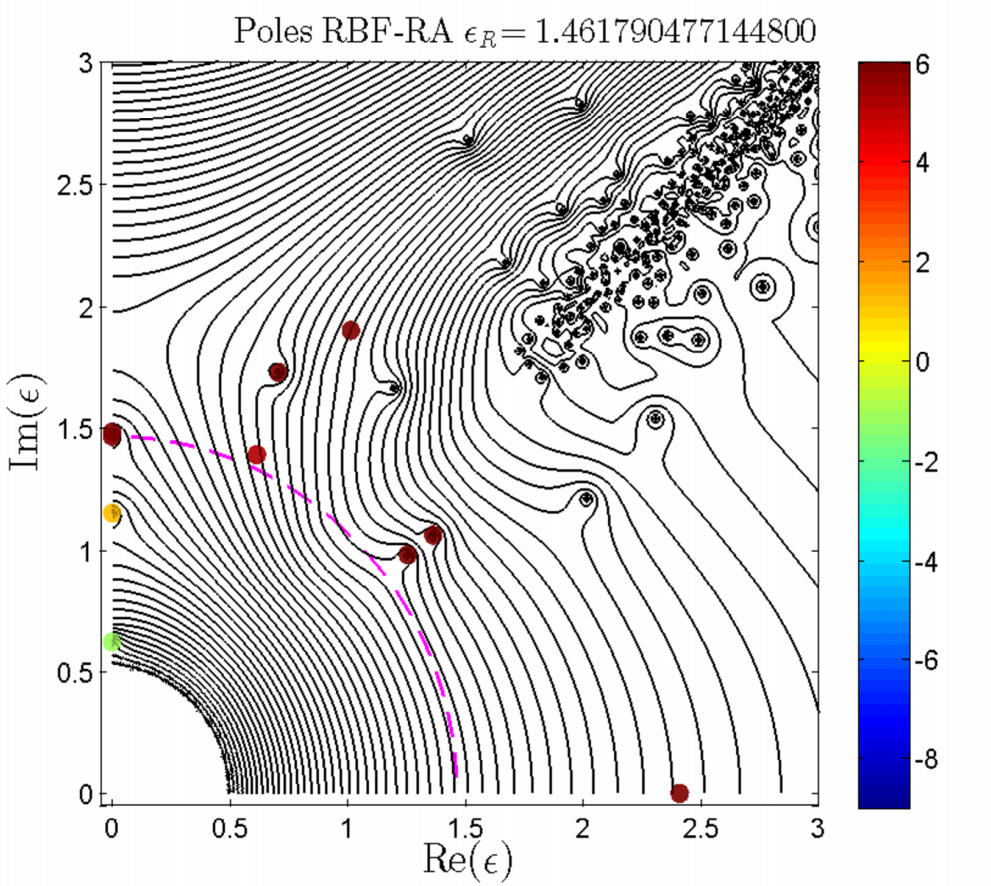
- $K/2+1$ is the total number of RBF interpolation systems to solve.

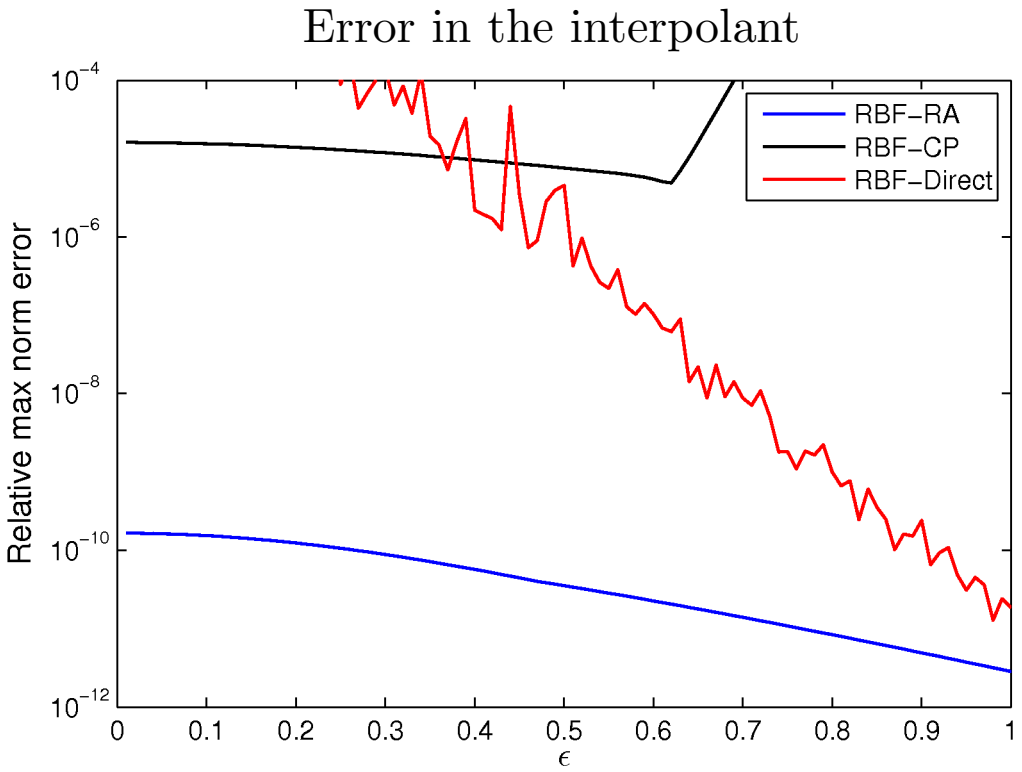
Numerical results: running through a pole





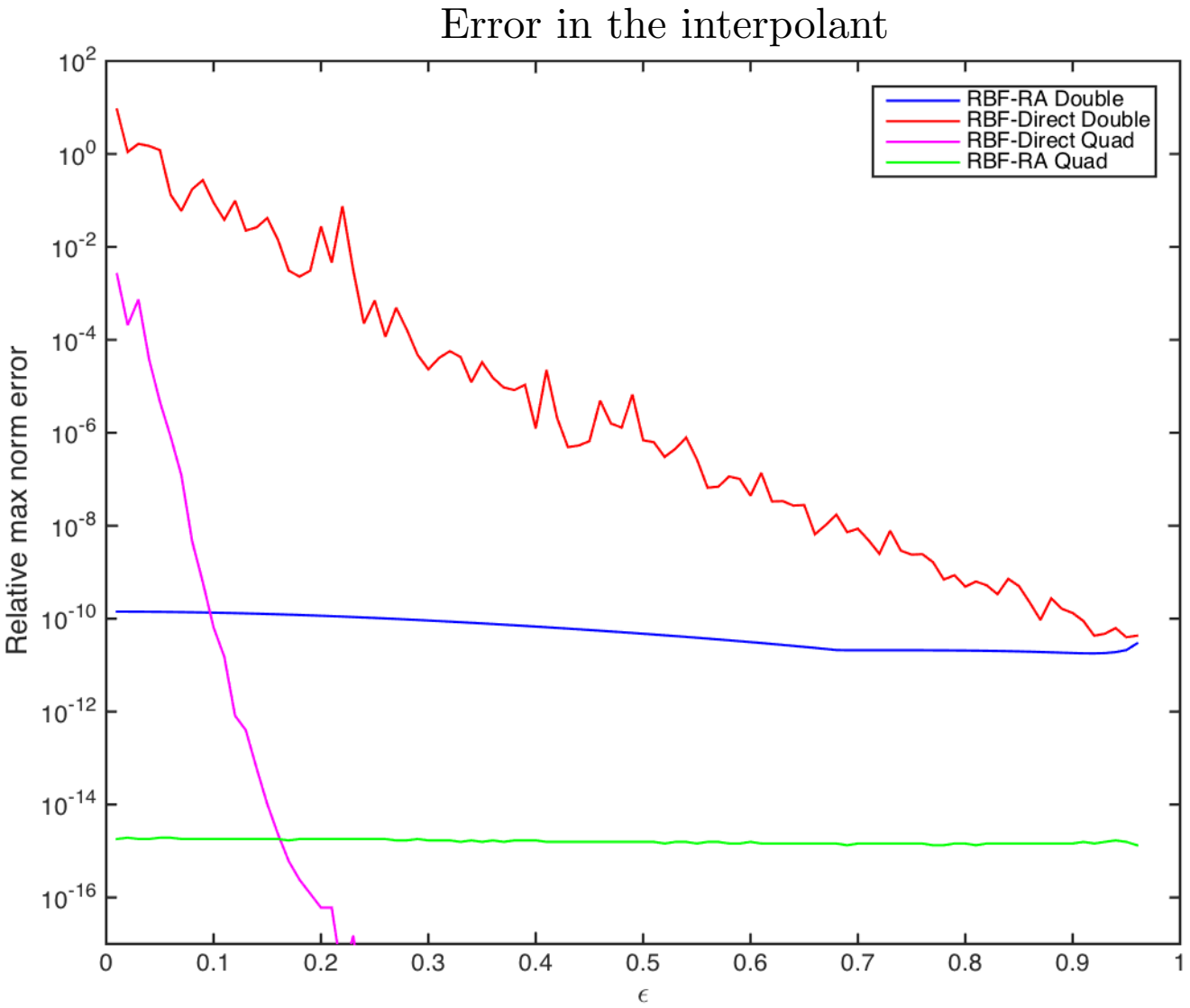
Numerical results: where are the poles





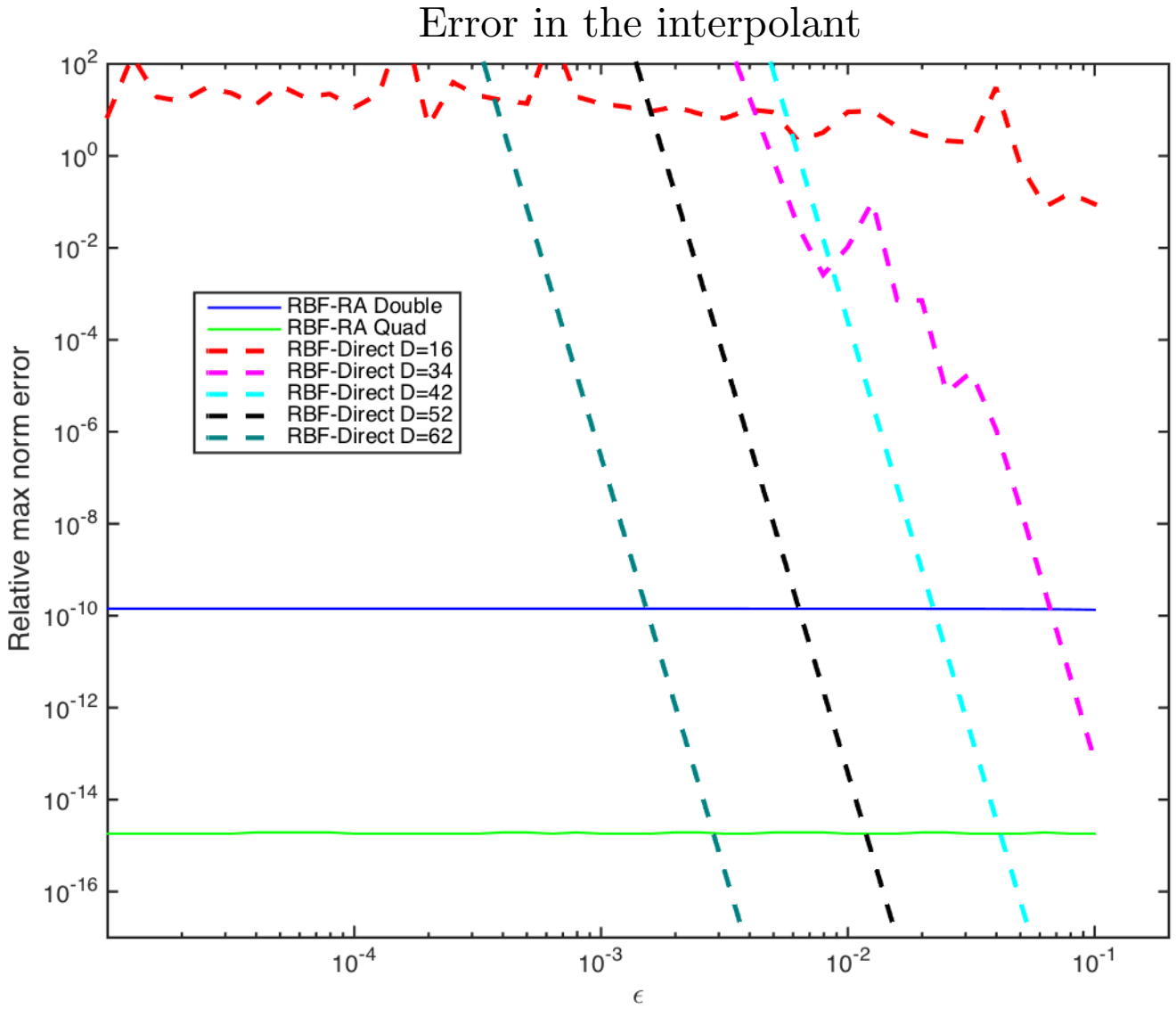
Numerical results: quad precision

- Comparison with quad-precision floating-point



Numerical results: quad precision

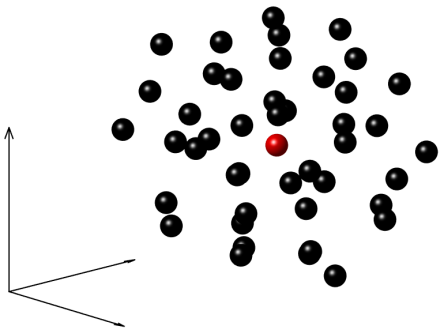
- Comparison with multi-precision floating point (loglog scale)



Application: Compact (Hermite) RBF-FD

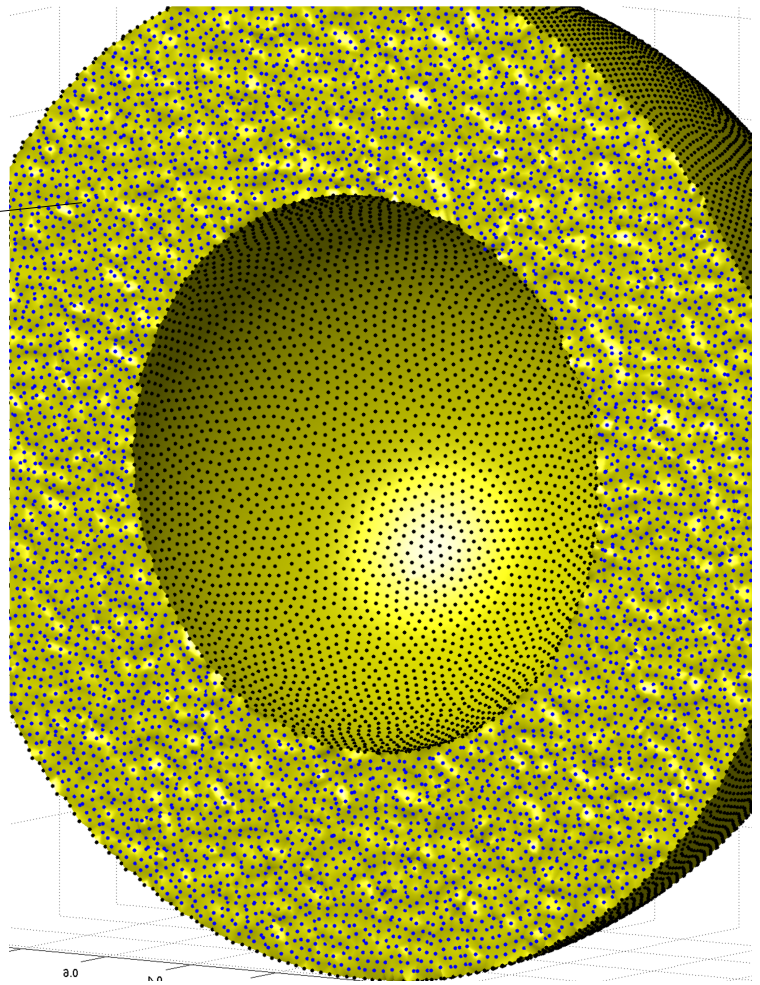
- Example: Spherical Shell $(11/20 \leq r \leq 1)$

RBF-FD Stencil



Spherical coordinates never used!

N=500,000 points courtesy of D. Hardin



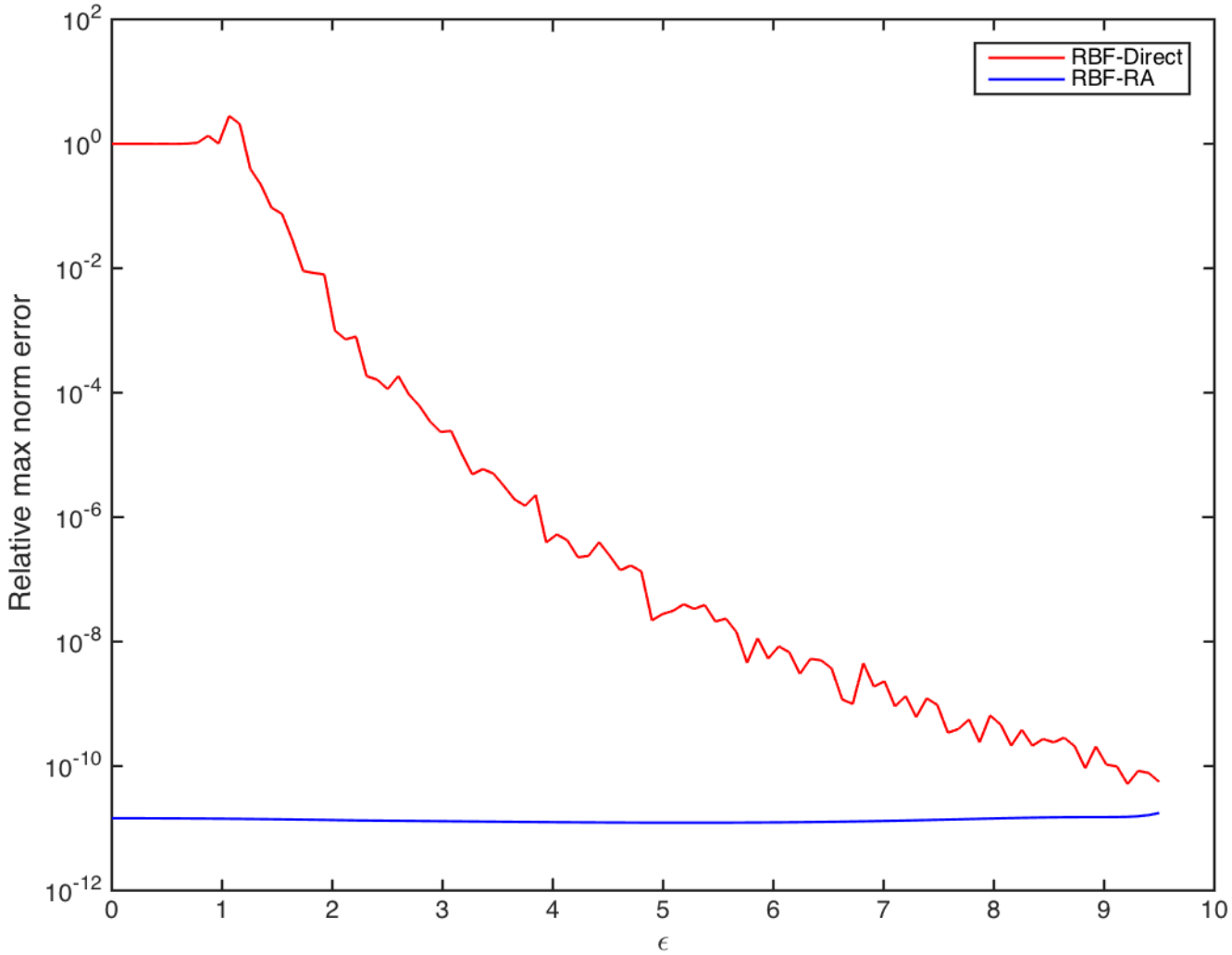
- Compact RBF-FD formula for Laplacian:

$$\nabla^2 u(x_1) \approx \sum_{j=1}^{n} c_j u(x_j) - \sum_{k=2}^{m} d_k \nabla^2 u(x_k)$$

- Weights are computed by solving a Hermite interpolation problem (W & Fornberg 2006)
- We use the RBF-RA algorithm, first to compute all weights, then to solve a Poisson equation.
- All computations are for inverse quadratic kernel, with n=45 and m=20 (approximately 4th order method).

Application: Compact (Hermite) RBF-FD

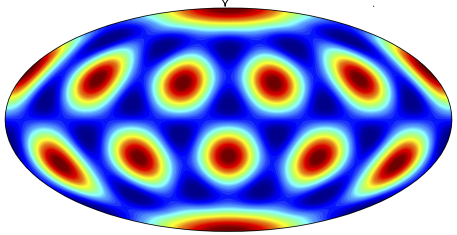
- Comparison of the error in the computation of the RBF-FD weights.



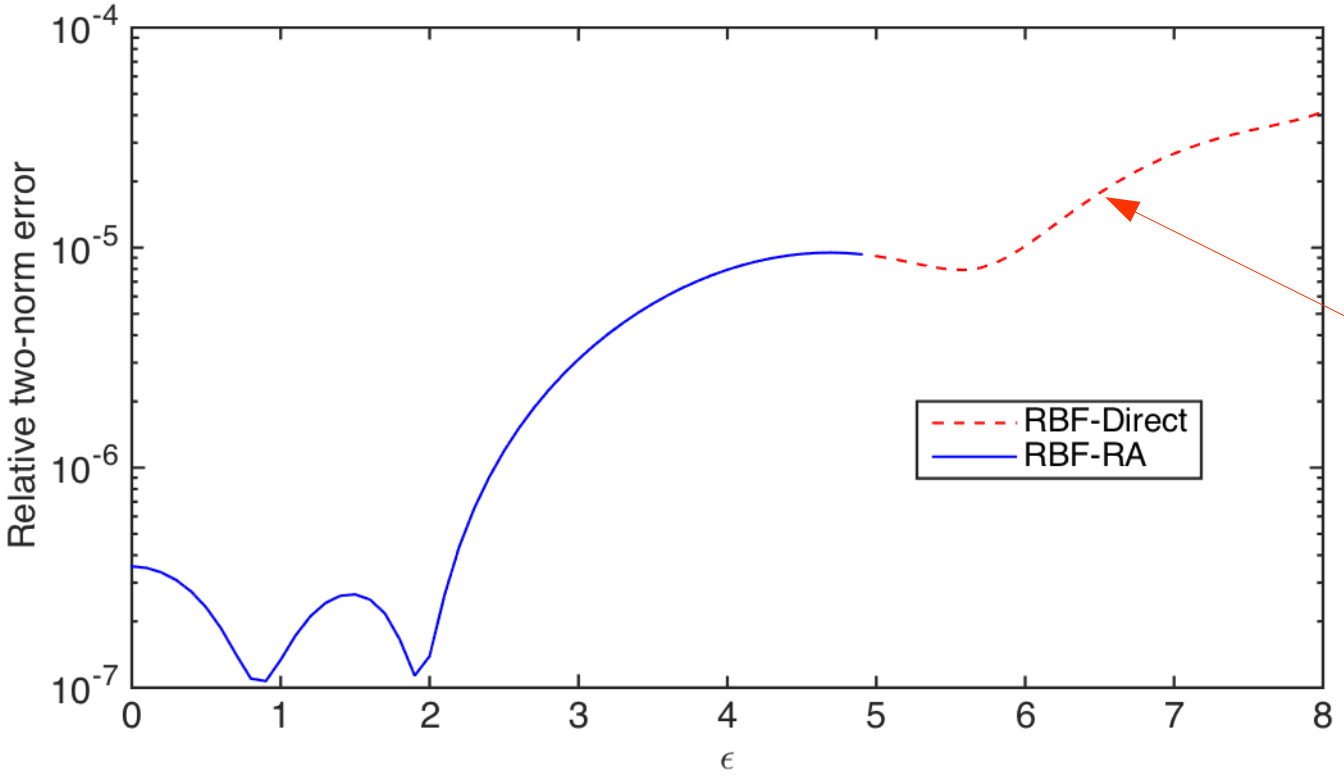
- Results for one particular stencil chosen at random.
- Comparison is made with 200-Digit multi-precision calculation.

Application: Compact (Hermite) RBF-FD

- Poisson equation in 3D spherical shell, N=500,000 nodes
- Exact solution: $u(r,\lambda,\theta) = \sin\left(\frac{20\pi}{9}\left(r - \frac{11}{20}\right)\right)\left[Y_6^0(\lambda,\theta) + \frac{14}{11}Y_6^5(\lambda,\theta)\right]$



Error in computed solution



Numerically safe region for RBF-Direct

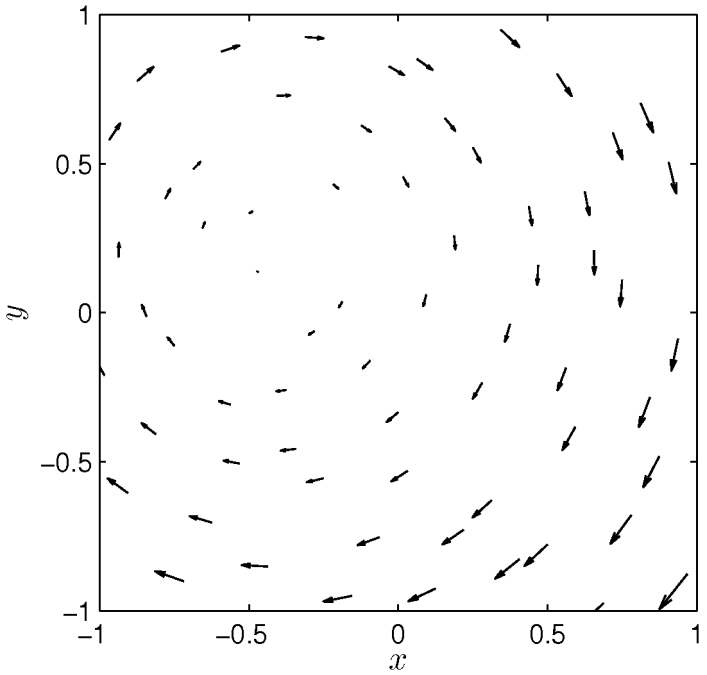
Application: divergence-free interpolation

- Interpolation with divergence-free RBFs (Narcowich and Ward 1995; Fuselier 2008):

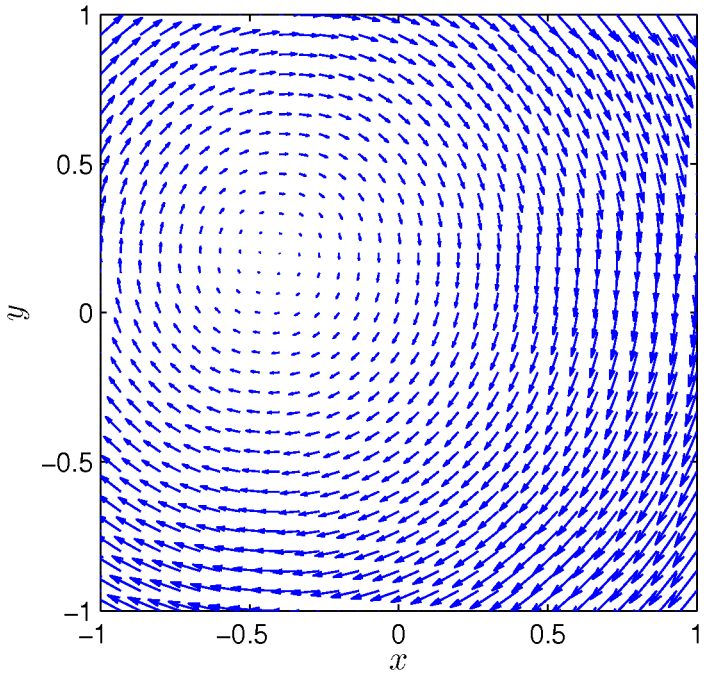
$$\mathbf{s}(\mathbf{x},\varepsilon) = \sum_{k=1}^{N} \Phi_\varepsilon^{\mathrm{div}}(\mathbf{x},\hat{\mathbf{x}}_k)\mathbf{c}_k, \quad \mathbf{s}(\hat{\mathbf{x}}_k) = \mathbf{u}_k.$$

- $\Phi_\varepsilon^{\mathrm{div}}$ is a smooth, positive-definite, matrix-valued kernel with divergence-free columns.

Samples of $\mathbf{u}$



Target div-free velocity field $\mathbf{u}$

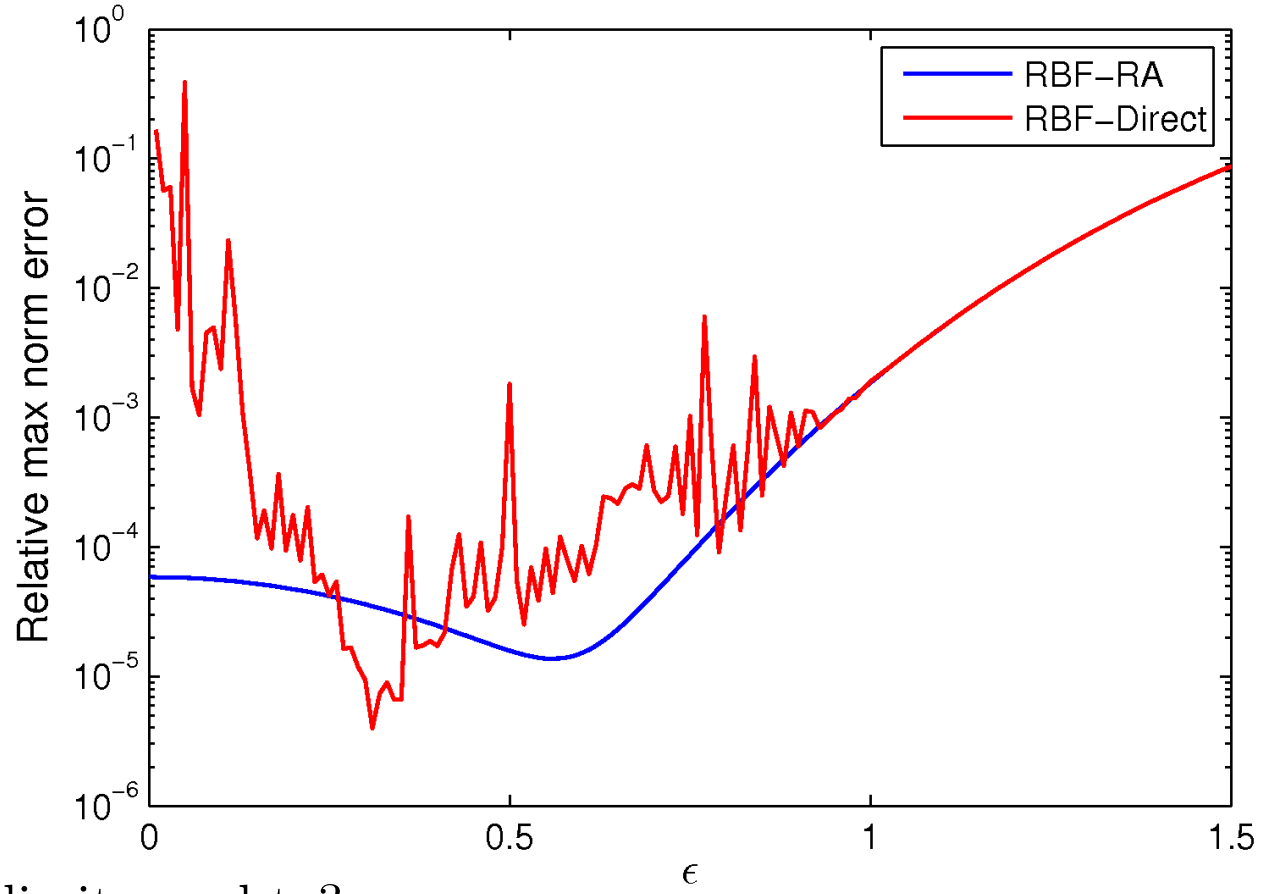


Application: divergence-free interpolation

- Interpolation with divergence-free RBFs (Narcowich and Ward 1995; Fuselier 2008):

$$\mathbf{s}(\mathbf{x},\varepsilon) = \sum_{k=1}^{N} \Phi_\varepsilon^{\mathrm{div}}(\mathbf{x},\hat{\mathbf{x}}_k)\mathbf{c}_k, \quad \mathbf{s}(\hat{\mathbf{x}}_k) = \mathbf{u}_k.$$

- $\Phi_\varepsilon^{\mathrm{div}}$ is a smooth, positive-definite, matrix-valued kernel with divergence-free columns.



- What is the limit equal to?

Concluding remarks

- Like RBF-CP, the new RBF-RA method is limited to small values of N.
- For applications like RBF partition-of-unity and finite-differences this is a minor limitation.
- For these applications the RBF-RA is more flexible than RBF-QR and RBF-GA.
- The implementation of RBF-RA is quite straightforward compared to RBF-CP, and the results are much more accurate and robust.
- Still need some work in choosing parameters:
  Contour, degrees of numerator and denominator, number of evaluation points.
- Some issues with using the method for non-polynomial unisolvent nodes
  For example nodes on a surface of a sphere.