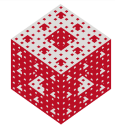
Using a Hilbert-Schmidt SVD for Stable Kernel Computations

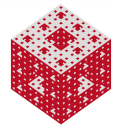
Greg Fasshauer* Mike McCourt Roberto Cavoretto

Department of Applied Mathematics
Illinois Institute of Technology
Partially supported by NSF Grant DMS–1115392

MAIA 2013
Multivariate Approximation and Interpolation with Applications
Erice, Sicily
Sept.25, 2013



Outline

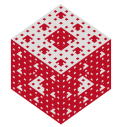


Kernel-based Interpolation

Given data $(\boldsymbol{x}_i, y_i)_{i=1}^N$, use a data-dependent linear function space

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j K(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d$$

with $K : \Omega \times \Omega \to \mathbb{R}$ a positive definite reproducing kernel.



Kernel-based Interpolation

Given data $(\boldsymbol{x}_i, y_i)_{i=1}^N$, use a data-dependent linear function space

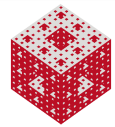
$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j K(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d$$

with $K : \Omega \times \Omega \to \mathbb{R}$ a positive definite reproducing kernel.
To find c_j solve the interpolation equations

$$s(\boldsymbol{x}_i) = y_i, \quad i = 1, \ldots, N,$$

which leads to a linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ with symmetric positive definite – often ill-conditioned – system matrix

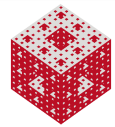
$$\mathrm{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i, j = 1, \ldots, N.$$



Common Complaints About Kernels

Kernel methods suffer from

- numerical instability,
- the presence of free parameter(s),
- high computational cost.



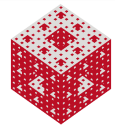
Common Complaints About Kernels

Kernel methods suffer from

- numerical instability,
- the presence of free parameter(s),
- high computational cost.

In this talk we will address the first two issues:

- We obtain stable methods by working with a "better" basis which leads to a Hilbert-Schmidt SVD of the matrix K.
- Free parameters can be "optimally" chosen by using statistical methods such as MLE, which are significantly enhanced by using the HS-SVD.



Hilbert-Schmidt Theory

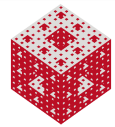
We assume that we know a Hilbert-Schmidt expansion (or Mercer series expansion) of our kernel K:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(\boldsymbol{x}) \varphi_n(\boldsymbol{z}),$$

where (λ_n, φ_n) are orthonormal eigenpairs of a Hilbert-Schmidt integral operator $\mathcal{T}_K : L_2(\Omega, \rho) \to L_2(\Omega, \rho)$ defined as

$$(\mathcal{T}_K f)(\boldsymbol{x}) = \int_{\Omega} K(\boldsymbol{x}, \boldsymbol{z}) f(\boldsymbol{z}) \rho(\boldsymbol{z}) \mathrm{d}\boldsymbol{z},$$

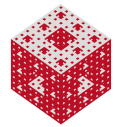
where $\Omega \subset \mathbb{R}^d$ and $\|K\|_{L_2(\Omega \times \Omega, \rho \times \rho)} < \infty$.



Gaussian Eigenfunctions

[Rasmussen/Williams (2006), F./McCourt (2012)]

$$e^{-\varepsilon^2(x-z)^2} = \sum_{n=0}^{\infty} \lambda_n \varphi_n(x) \varphi_n(z)$$



Gaussian Eigenfunctions

[Rasmussen/Williams (2006), F./McCourt (2012)]

$$e^{-\varepsilon^2(x-z)^2} = \sum_{n=0}^{\infty} \lambda_n \varphi_n(x) \varphi_n(z)$$

where

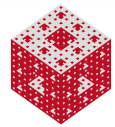
$$\lambda_n = \sqrt{\frac{\alpha^2}{\alpha^2 + \delta^2 + \varepsilon^2}} \left(\frac{\varepsilon^2}{\alpha^2 + \delta^2 + \varepsilon^2} \right)^n, \quad \varphi_n(x) = \gamma_n e^{-\delta^2 x^2} H_n(\alpha\beta x)$$

with H_n Hermite polynomials,

$$\beta = \left(1 + \left(\frac{2\varepsilon}{\alpha} \right)^2 \right)^{\frac{1}{4}}, \quad \gamma_n = \sqrt{\frac{\beta}{2^n \Gamma(n+1)}}, \quad \delta^2 = \frac{\alpha^2}{2} \left(\beta^2 - 1 \right)$$

and $\{\varphi_n\}_{n=0}^{\infty}$ (ρ-weighted) L_2-orthonormal, i.e.,

$$\int_{-\infty}^{\infty} \varphi_m(x) \varphi_n(x) \, \rho(x) \, dx = \delta_{mn}, \qquad \rho(x) = \frac{\alpha}{\sqrt{\pi}} e^{-\alpha^2 x^2}$$

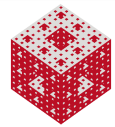


Multivariate Eigenfunction Expansion

Use tensor product form of the Gaussian kernel

$$K(\boldsymbol{x},\boldsymbol{z}) = e^{-\varepsilon^2\|\boldsymbol{x}-\boldsymbol{z}\|_2^2} = e^{-\sum_{\ell=1}^{d}\varepsilon^2(x_\ell-z_\ell)^2} = \prod_{\ell=1}^{d} e^{-\varepsilon^2(x_\ell-z_\ell)^2}$$

$$\boldsymbol{x} = (x_1,\ldots,x_d) \in \mathbb{R}^d,$$

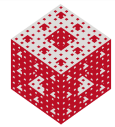


Multivariate Eigenfunction Expansion

Use tensor product form of the Gaussian kernel

$$K(\boldsymbol{x},\boldsymbol{z}) = e^{-\varepsilon^2\|\boldsymbol{x}-\boldsymbol{z}\|_2^2} = e^{-\sum_{\ell=1}^{d}\varepsilon_\ell^2(x_\ell-z_\ell)^2} = \prod_{\ell=1}^{d} e^{-\varepsilon_\ell^2(x_\ell-z_\ell)^2}$$

$$\boldsymbol{x} = (x_1,\ldots,x_d) \in \mathbb{R}^d,$$



Multivariate Eigenfunction Expansion

Use tensor product form of the Gaussian kernel

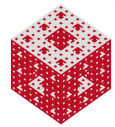
$$K(\boldsymbol{x},\boldsymbol{z}) = e^{-\varepsilon^2\|\boldsymbol{x}-\boldsymbol{z}\|_2^2} = e^{-\sum_{\ell=1}^{d}\varepsilon_\ell^2(x_\ell-z_\ell)^2} = \prod_{\ell=1}^{d} e^{-\varepsilon_\ell^2(x_\ell-z_\ell)^2}$$

$$= \sum_{\boldsymbol{n}\in\mathbb{N}^d} \lambda_{\boldsymbol{n}}\varphi_{\boldsymbol{n}}(\boldsymbol{x})\varphi_{\boldsymbol{n}}(\boldsymbol{z}), \qquad \boldsymbol{x} = (x_1,\ldots,x_d) \in \mathbb{R}^d,$$

where

$$\lambda_{\boldsymbol{n}} = \prod_{\ell=1}^{d} \lambda_{n_\ell}, \qquad \varphi_{\boldsymbol{n}}(\boldsymbol{x}) = \prod_{\ell=1}^{d} \varphi_{n_\ell}(x_\ell).$$

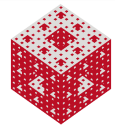
Different shape parameters ε_ℓ for different space dimensions allowed (i.e., K may be anisotropic).



Fundamental idea: use the eigen-expansion of the kernel K to rewrite the matrix K from the interpolation problem as

$$\mathrm{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & & \vdots \\ K(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

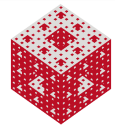
$$= \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_1) & \dots \\ \vdots & & \vdots & \\ \varphi_1(\boldsymbol{x}_N) & \dots & \varphi_M(\boldsymbol{x}_N) & \dots \end{pmatrix} \begin{pmatrix} \lambda_1 & & & \\ & \ddots & & \\ & & \lambda_M & \\ & & & \ddots \end{pmatrix} \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_1(\boldsymbol{x}_N) \\ \vdots & & \vdots \\ \varphi_M(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_N) \\ \vdots & & \vdots \end{pmatrix}$$



But we can't compute with infinite matrices, so we choose a truncation value M (supported by $\lambda_n \to 0$ as $n \to \infty$, more later) and rewrite

$$\mathrm{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & & \vdots \\ K(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \dots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix}}_{=\Phi} \underbrace{\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_M \end{pmatrix}}_{=\Lambda} \underbrace{\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_1(\boldsymbol{x}_N) \\ \vdots & & \vdots \\ \varphi_M(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix}}_{=\Phi^T}$$



But we can't compute with infinite matrices, so we choose a truncation value M (supported by $\lambda_n \to 0$ as $n \to \infty$, more later) and rewrite

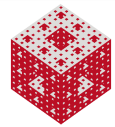
$$\mathsf{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & & \vdots \\ K(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \dots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix}}_{=\Phi} \underbrace{\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_M \end{pmatrix}}_{=\Lambda} \underbrace{\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_1(\boldsymbol{x}_N) \\ \vdots & & \vdots \\ \varphi_M(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix}}_{=\Phi^T}$$

Since

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(\boldsymbol{x}_i) \varphi_n(\boldsymbol{x}_j) \approx \sum_{n=1}^{M} \lambda_n \varphi_n(\boldsymbol{x}_i) \varphi_n(\boldsymbol{x}_j)$$

accurate reconstruction of all entries of K will likely require $M > N$.



The matrix K is often ill-conditioned, so forming K and computing with it is not a good idea.

The matrix K is often ill-conditioned, so forming K and computing with it is not a good idea.

The eigen-decomposition

$$K = \Phi \Lambda \Phi^T$$

provides an accurate (elementwise) approximation of K without ever forming it.

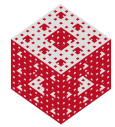
The matrix K is often ill-conditioned, so forming K and computing with it is not a good idea.

The eigen-decomposition

$$K = \Phi \Lambda \Phi^T$$

provides an accurate (elementwise) approximation of K without ever forming it.

However, it is not recommended to directly use this decomposition either since all of the ill-conditioning associated with K is still present – sitting in the matrix Λ.



The matrix K is often ill-conditioned, so forming K and computing with it is not a good idea.

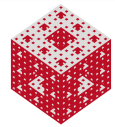
The eigen-decomposition

$$K = \Phi \Lambda \Phi^T$$

provides an accurate (elementwise) approximation of K without ever forming it.

However, it is not recommended to directly use this decomposition either since all of the ill-conditioning associated with K is still present – sitting in the matrix Λ.

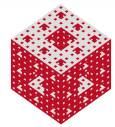
We now use mostly standard numerical linear algebra to isolate some of this ill-conditioning and develop the Hilbert-Schmidt SVD and a general RBF-QR algorithm.



Details of the Hilbert-Schmidt SVD

Assume $M > N$, so that Φ is "short and fat" and partition Φ:

$$\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_N(\boldsymbol{x}_1) & \varphi_{N+1}(\boldsymbol{x}_1) & \cdots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & & \vdots & \vdots & & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \cdots & \varphi_N(\boldsymbol{x}_N) & \varphi_{N+1}(\boldsymbol{x}_N) & \cdots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix} = \begin{pmatrix} \underbrace{\Phi_1}_{N\times N} & \underbrace{\Phi_2}_{N\times(M-N)} \end{pmatrix}$$



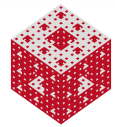
Details of the Hilbert-Schmidt SVD

Assume $M > N$, so that Φ is "short and fat" and partition Φ:

$$\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_N(\boldsymbol{x}_1) & \varphi_{N+1}(\boldsymbol{x}_1) & \cdots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & & \vdots & \vdots & & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \cdots & \varphi_N(\boldsymbol{x}_N) & \varphi_{N+1}(\boldsymbol{x}_N) & \cdots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix} = \begin{pmatrix} \underbrace{\Phi_1}_{N\times N} & \underbrace{\Phi_2}_{N\times(M-N)} \end{pmatrix}$$

Then

$$\mathsf{K} \;=\; \Phi\Lambda\Phi^T$$



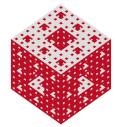
Details of the Hilbert-Schmidt SVD

Assume $M > N$, so that Φ is "short and fat" and partition Φ:

$$\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_N(\boldsymbol{x}_1) & \varphi_{N+1}(\boldsymbol{x}_1) & \cdots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & & \vdots & \vdots & & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \cdots & \varphi_N(\boldsymbol{x}_N) & \varphi_{N+1}(\boldsymbol{x}_N) & \cdots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix} = \begin{pmatrix} \underbrace{\Phi_1}_{N\times N} & \underbrace{\Phi_2}_{N\times(M-N)} \end{pmatrix}$$

Then

$$\begin{aligned} \mathsf{K} &= \Phi\Lambda\Phi^T \\ &= \Phi \begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix} \begin{pmatrix} \Phi_1^T \\ \Phi_2^T \end{pmatrix} \end{aligned}$$



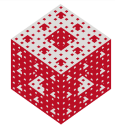
Details of the Hilbert-Schmidt SVD

Assume $M > N$, so that Φ is "short and fat" and partition Φ:

$$\begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_N(\boldsymbol{x}_1) & \varphi_{N+1}(\boldsymbol{x}_1) & \cdots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & & \vdots & \vdots & & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \cdots & \varphi_N(\boldsymbol{x}_N) & \varphi_{N+1}(\boldsymbol{x}_N) & \cdots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix} = \begin{pmatrix} \underbrace{\Phi_1}_{N\times N} & \underbrace{\Phi_2}_{N\times(M-N)} \end{pmatrix}$$

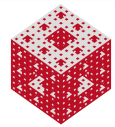
Then

$$\begin{aligned} K &= \Phi\Lambda\Phi^T \\ &= \Phi \begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix} \begin{pmatrix} \Phi_1^T \\ \Phi_2^T \end{pmatrix} \\ &= \underbrace{\Phi \begin{pmatrix} I_N \\ \Lambda_2\Phi_2^T\Phi_1^{-T}\Lambda_1^{-1} \end{pmatrix}}_{=\Psi} \underbrace{\Lambda_1\Phi_1^T}_{=M} \end{aligned}$$



There are at least two ways to interpret the Hilbert-Schmidt SVD

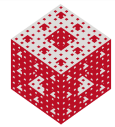
$$K = \Psi \Lambda_1 \Phi_1^T$$



There are at least two ways to interpret the Hilbert-Schmidt SVD

$$K = \Psi \Lambda_1 \Phi_1^T$$

- We've found an invertible $M = \Lambda_1 \Phi_1^T$ such that $\Psi = KM^{-1}$ is better conditioned than K $\rightsquigarrow$ "better basis".



There are at least two ways to interpret the Hilbert-Schmidt SVD

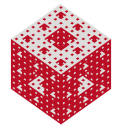
$$K = \Psi \Lambda_1 \Phi_1^T$$

- We've found an invertible $M = \Lambda_1 \Phi_1^T$ such that $\Psi = KM^{-1}$ is better conditioned than K $\rightsquigarrow$ "better basis".
- We've diagonalized the matrix K, i.e.,

$$K = \Psi \Lambda_1 \Phi_1^T,$$

where

- Λ_1 is a diagonal matrix of Hilbert-Schmidt singular values,
- Ψ and Φ_1 are matrices generated by orthogonal eigenfunctions (but not orthogonal matrices).



There are at least two ways to interpret the Hilbert-Schmidt SVD

$$K = \Psi \Lambda_1 \Phi_1^T$$

- We've found an invertible $M = \Lambda_1 \Phi_1^T$ such that $\Psi = KM^{-1}$ is better conditioned than K $\rightsquigarrow$ "better basis".
- We've diagonalized the matrix K, i.e.,

$$K = \Psi \Lambda_1 \Phi_1^T,$$

where
  - Λ_1 is a diagonal matrix of Hilbert-Schmidt singular values,
  - Ψ and Φ_1 are matrices generated by orthogonal eigenfunctions (but not orthogonal matrices).

Remark

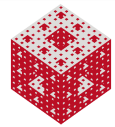
The matrix Ψ is the same for both interpretations.

- *It can be computed stably.*
- *We get a well-conditioned linear system $\Psi \boldsymbol{b} = \boldsymbol{y}$ (where $\boldsymbol{b} = M\boldsymbol{c}$) for the interpolation problem.*

Taking a closer look at the matrix Ψ, we see that

$$\begin{aligned}\Psi &= (\Phi_1\ \Phi_2)\begin{pmatrix} I_N \\ \Lambda_2\Phi_2^T\Phi_1^{-T}\Lambda_1^{-1}\end{pmatrix} \\ &= \Phi_1 + \Phi_2\left[\Lambda_2\Phi_2^T\Phi_1^{-T}\Lambda_1^{-1}\right].\end{aligned}$$

We can interpret this as having a new basis $\psi(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$ for the interpolation space span $\{K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N\}\}$ consisting of the appropriately corrected first N eigenfunctions:

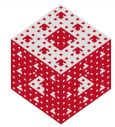


Taking a closer look at the matrix Ψ, we see that

$$\begin{aligned}\Psi &= (\Phi_1 \ \Phi_2) \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \\ &= \Phi_1 + \Phi_2 \left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \right].\end{aligned}$$

We can interpret this as having a new basis $\psi(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$ for the interpolation space span $\{K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N\}\}$ consisting of the appropriately corrected first N eigenfunctions:
If we let $\phi(\cdot)^T = (\varphi_1(\cdot), \ldots, \varphi_N(\cdot), \varphi_{N+1}(\cdot), \ldots, \varphi_M(\cdot))$, then we can rewrite our kernel basis using the Hilbert-Schmidt SVD

$$\boldsymbol{k}(\boldsymbol{x})^T = \phi(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T = \psi(\boldsymbol{x})^T \Lambda_1 \Phi_1^T.$$



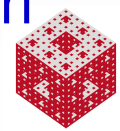
Taking a closer look at the matrix Ψ, we see that

$$\begin{aligned}\Psi &= (\Phi_1 \ \Phi_2) \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \\ &= \Phi_1 + \Phi_2 \left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \right].\end{aligned}$$

We can interpret this as having a new basis $\psi(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$ for the interpolation space span $\{K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N\}\}$ consisting of the appropriately corrected first N eigenfunctions:
If we let $\phi(\cdot)^T = (\varphi_1(\cdot), \ldots, \varphi_N(\cdot), \varphi_{N+1}(\cdot), \ldots, \varphi_M(\cdot))$, then we can rewrite our kernel basis using the Hilbert-Schmidt SVD

$$\boldsymbol{k}(\boldsymbol{x})^T = \phi(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T = \psi(\boldsymbol{x})^T \Lambda_1 \Phi_1^T.$$

The data-dependence of the new basis is captured by the "correction" term. The new basis is more stable since we have removed Λ_1.



The QR in RBF-QR

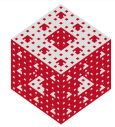
Additional stability in the computation of the correction matrix

$$\left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}\right],$$

in particular, in the formation of $\Phi_2^T \Phi_1^{-T}$, is achieved via a QR decomposition of Φ, i.e.,

$$\begin{pmatrix} \Phi_1 & \Phi_2 \end{pmatrix} = Q \begin{pmatrix} \underbrace{R_1}_{N \times N} & \underbrace{R_2}_{N \times (M-N)} \end{pmatrix}$$

with orthogonal $N \times N$ matrix Q and upper triangular matrix R_1.



The QR in RBF-QR

Additional stability in the computation of the correction matrix

$$\left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}\right],$$

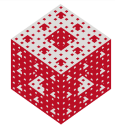
in particular, in the formation of $\Phi_2^T \Phi_1^{-T}$, is achieved via a QR decomposition of Φ, i.e.,

$$\begin{pmatrix} \Phi_1 & \Phi_2 \end{pmatrix} = Q \begin{pmatrix} \underbrace{R_1}_{N\times N} & \underbrace{R_2}_{N\times(M-N)} \end{pmatrix}$$

with orthogonal $N \times N$ matrix Q and upper triangular matrix R_1. Then we have

$$\Phi_2^T \Phi_1^{-T} = R_2^T Q^T Q R_1^{-T} = R_2^T R_1^{-T}.$$

This idea appeared in [Fornberg/Piret (2008)].



Summary of Method

Instead of solving the "original" interpolation problem with ill-conditioned matrix K

$$\mathrm{K}\boldsymbol{c} = \boldsymbol{y},$$

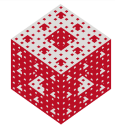
leading to inaccurate coefficients which then need to be multiplied against poorly conditioned basis functions, we now solve

$$\Psi \boldsymbol{b} = \boldsymbol{y}$$

for a new set of coefficients which we then evaluate via

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} b_j \psi_j(\boldsymbol{x}),$$

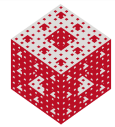
i.e., using the new basis.



General Implementation

It is crucial to know the Hilbert-Schmidt expansion of K:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(\boldsymbol{x}) \varphi_n(\boldsymbol{z})$$



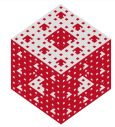
General Implementation

It is crucial to know the Hilbert-Schmidt expansion of K:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(\boldsymbol{x}) \varphi_n(\boldsymbol{z})$$

The multivariate Gaussian kernels mentioned earlier were used in

- [F./Hickernell/Woźniakowski (2012)] to prove dimension-independent convergence rates
- [F./McCourt (2012)] to obtain and implement a stable GaussQR algorithm.



General Implementation

It is crucial to know the Hilbert-Schmidt expansion of K:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(\boldsymbol{x}) \varphi_n(\boldsymbol{z})$$

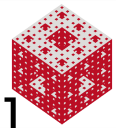
The multivariate Gaussian kernels mentioned earlier were used in

- [F./Hickernell/Woźniakowski (2012)] to prove dimension-independent convergence rates
- [F./McCourt (2012)] to obtain and implement a stable GaussQR algorithm.

We now discuss the implementation for generalizations of the Brownian bridge kernel

$$K(x, z) = \min(x, z) - xz, \qquad x, z \in [0, 1],$$

which we call compact Matérn kernels [Cavoretto/F./McCourt (2013)].

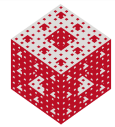


We define compact Matérn kernels as Green's kernels of

$$\left(-\frac{\mathrm{d}^2}{\mathrm{d}x^2}+\varepsilon^2\mathcal{I}\right)^{\beta} K(x,z)=\delta(x-z), \qquad x,z\in[0,1],\ \beta\in\mathbb{N},\ \varepsilon\geq 0,$$

subject to

$$\frac{\mathrm{d}^{2\nu}}{\mathrm{d}x^{2\nu}}K(0,z)=\frac{\mathrm{d}^{2\nu}}{\mathrm{d}x^{2\nu}}K(1,z)=0, \qquad \nu=0,\ldots,\beta-1.$$



We define compact Matérn kernels as Green's kernels of

$$\left(-\frac{\mathrm{d}^2}{\mathrm{d}x^2}+\varepsilon^2\mathcal{I}\right)^{\beta} K(x,z)=\delta(x-z), \qquad x,z\in[0,1],\ \beta\in\mathbb{N},\ \varepsilon\geq 0,$$

subject to

$$\frac{\mathrm{d}^{2\nu}}{\mathrm{d}x^{2\nu}}K(0,z)=\frac{\mathrm{d}^{2\nu}}{\mathrm{d}x^{2\nu}}K(1,z)=0, \qquad \nu=0,\ldots,\beta-1.$$

The Hilbert-Schmidt expansion for compact Matérn kernels is

$$K_{\beta,\varepsilon}(x,z)=\sum_{n=1}^{\infty}\frac{2}{\left(n^2\pi^2+\varepsilon^2\right)^{\beta}}\sin\left(n\pi x\right)\sin\left(n\pi z\right),$$

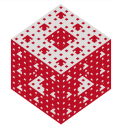
i.e., the eigenvalues and eigenfunctions are

$$\lambda_n=\frac{1}{\left(n^2\pi^2+\varepsilon^2\right)^{\beta}}, \qquad \varphi_n(x)=\sqrt{2}\sin\left(n\pi x\right). \tag{1}$$

Clearly,

- the eigenfunctions are bounded by $\sqrt{2}$,
- and, for a fixed value of ε, the eigenvalues decay as $n^{-2\beta}$.

Therefore the truncation length M needed for accurate representation of the entries of K can be easily determined as a function of β and ε:



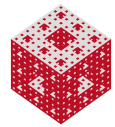
Clearly,

- the eigenfunctions are bounded by $\sqrt{2}$,
- and, for a fixed value of ε, the eigenvalues decay as $n^{-2\beta}$.

Therefore the truncation length M needed for accurate representation of the entries of K can be easily determined as a function of β and ε:

To ensure that we keep the first M significant terms we take M such that

$$\lambda_M < \epsilon_{mach}\lambda_N, \qquad M > N.$$



Clearly,

- the eigenfunctions are bounded by $\sqrt{2}$,
- and, for a fixed value of ε, the eigenvalues decay as $n^{-2\beta}$.

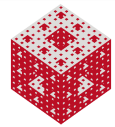
Therefore the truncation length M needed for accurate representation of the entries of K can be easily determined as a function of β and ε:

To ensure that we keep the first M significant terms we take M such that

$$\lambda_M < \epsilon_{mach}\lambda_N, \qquad M > N.$$

Using the explicit representation of the eigenvalues, we solve for M:

$$M(\beta, \varepsilon; \epsilon_{mach}) = \left\lceil \frac{1}{\pi} \sqrt{\epsilon_{mach}^{-1/\beta}(N^2\pi^2 + \varepsilon^2) - \varepsilon^2} \right\rceil.$$



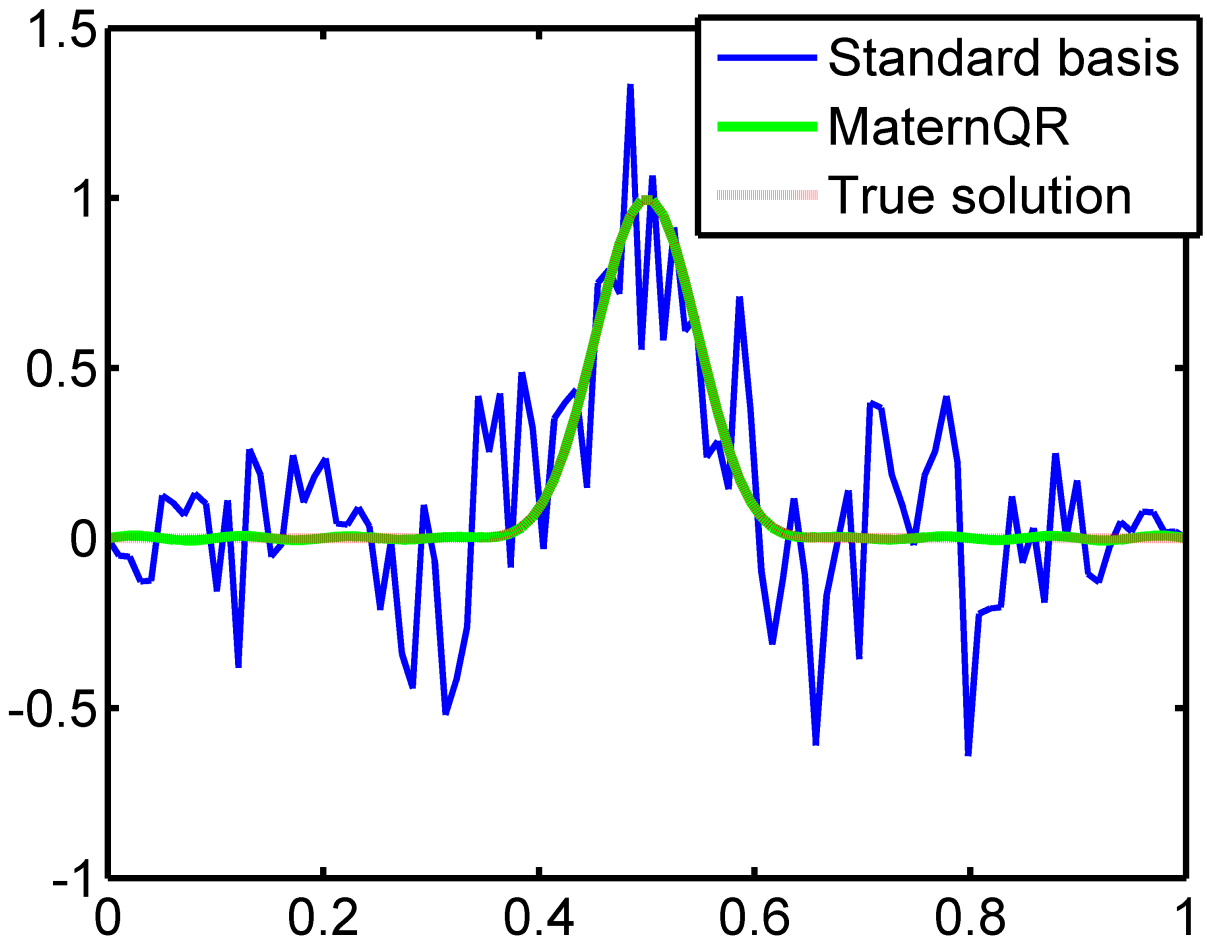
Program (MaternQRSolve.m)

```
function yy = MaternQRSolve(x,y,ep,beta,xx)
  phifunc = @(n,x) sqrt(2)*sin(pi*x*n);
  N = length(x);
  M = ceil(1/pi*sqrt(eps^(-1/beta)*(N^2*pi^2+ep^2)-ep^2));
  n = 1:M;
  Lambda = diag(((n*pi).^2+ep^2).^(-beta));
  Phi = phifunc(n,x);
  [Q,R] = qr(Phi);
  R1 = R(:,1:N);   R2 = R(:,N+1:end);
  Rhat = R1\R2;
  Lambda1 = Lambda(1:N,1:N);
  Lambda2 = Lambda(N+1:M,N+1:M);
  Rbar = Lambda2*Rhat'/Lambda1;
  Psi = Phi*[eye(N);Rbar];
  b = Psi\y;
  Phi_eval = phifunc(n,xx);
  yy = Phi_eval*[eye(N);Rbar]*b;
end
```

Standard RBF vs. MatérnQR Interpolation

We use

- $K_{\beta,\varepsilon}$ with $\beta = 7$ and $\varepsilon = 1$
- $N = 21$ uniform samples of $f(x) = (1-4x)_+^{14}(4x-3)_+^{14}$



Likelihood Functions for Gaussian Random Fields

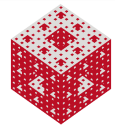
Kernel-based interpolation has an analog in statistics called kriging.

If, instead of trying to recover a function, we treat our scattered data as samples of one realization of a Gaussian random field, we can prescribe a positive definite kernel K as the presumed covariance between realizations of the Gaussian random field.

The likelihood function of a zero-mean Gaussian random field (the probability of the data $(\boldsymbol{x}_i, y_i)_{i=1}^N$ given the kernel K with shape parameters $\boldsymbol{\theta} = (\varepsilon, \beta)$) is

$$L(\boldsymbol{\theta}; \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi\sigma^2)^N \det(\mathrm{K})}} \exp\left(-\frac{1}{2\sigma^2}\boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y}\right)$$

K is the kernel interpolation matrix from before, σ^2 is the process variance.

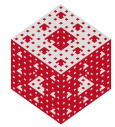


Maximum Likelihood Estimation (MLE)

Usually we minimize the negative (concentrated) log-likelihood:

$$\widetilde{L}(\boldsymbol{\theta}; \boldsymbol{y}) = \frac{1}{N} \log \det(\mathrm{K}) + \log \underbrace{\left(\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} \right)}_{=Q(\boldsymbol{y})}.$$

This requires evaluating $\log \det(\mathrm{K})$ and $\log Q(\boldsymbol{y})$, which, given the ill-conditioning of K are both bound to cause trouble.



Maximum Likelihood Estimation (MLE)

Usually we minimize the negative (concentrated) log-likelihood:

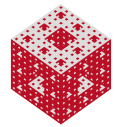
$$\widetilde{L}(\boldsymbol{\theta}; \boldsymbol{y}) = \frac{1}{N} \log\det(\mathrm{K}) + \log \underbrace{\left(\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} \right)}_{=Q(\boldsymbol{y})}.$$

This requires evaluating $\log\det(\mathrm{K})$ and $\log Q(\boldsymbol{y})$, which, given the ill-conditioning of K are both bound to cause trouble.

Luckily, we have developed the Hilbert-Schmidt SVD

$$\mathrm{K} = \Psi \Lambda_1 \Phi_1^T$$

to help in both cases.

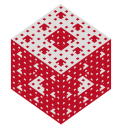


Computing log det(K)

We use the Hilbert-Schmidt SVD to write

$$\det(K) = \det(\Psi\Lambda_1\Phi_1^T) = \det(\Psi)\det(\Lambda_1)\det(\Phi_1)$$

Evaluating $\det(\Lambda_1)$ can be done analytically and $\det(\Psi)$ and $\det(\Phi_1)$ can be computed stably using standard techniques.

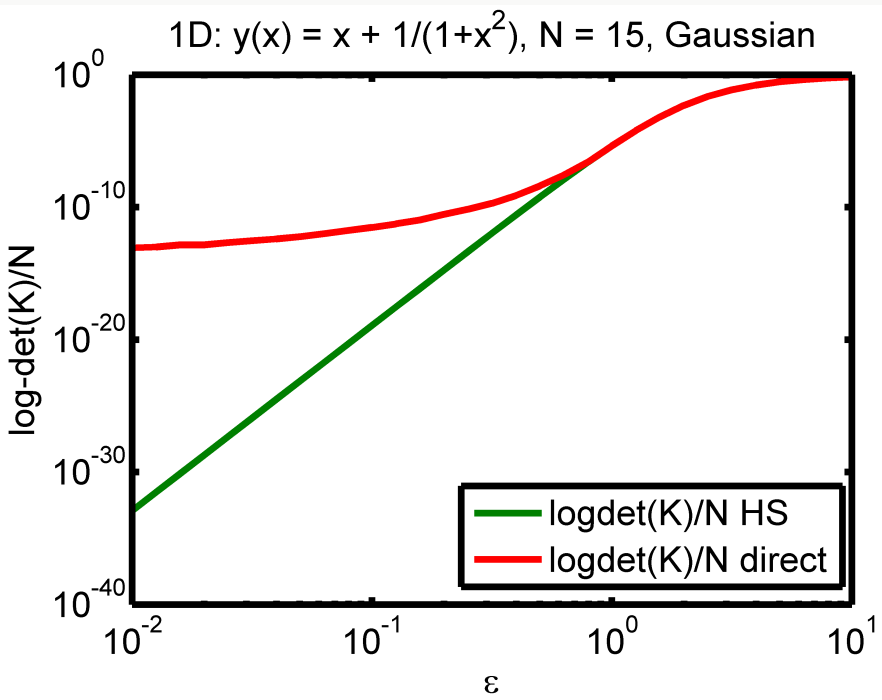


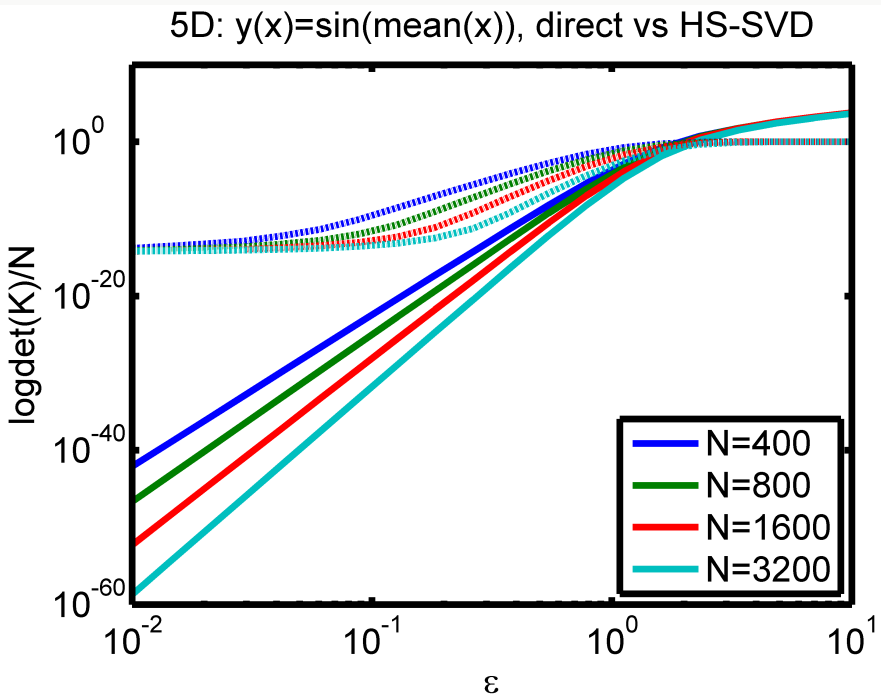
Computing log det(K)

We use the Hilbert-Schmidt SVD to write

$$\det(K) = \det(\Psi\Lambda_1\Phi_1^T) = \det(\Psi)\det(\Lambda_1)\det(\Phi_1)$$

Evaluating $\det(\Lambda_1)$ can be done analytically and $\det(\Psi)$ and $\det(\Phi_1)$ can be computed stably using standard techniques.

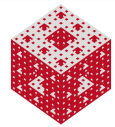




Computing $\log Q(\boldsymbol{y})$

We remember that the Hilbert-Schmidt SVD $\mathrm{K} = \Psi\Lambda_1\Phi_1^T$ gives us

$$\mathrm{K}\boldsymbol{c} = \boldsymbol{y} \iff \Psi \underbrace{\Lambda_1\Phi_1^T\boldsymbol{c}}_{=\boldsymbol{b}} = \boldsymbol{y}. \tag{2}$$



Computing log $Q(\boldsymbol{y})$

We remember that the Hilbert-Schmidt SVD $\mathrm{K} = \Psi\Lambda_1\Phi_1^T$ gives us

$$\mathrm{K}\boldsymbol{c} = \boldsymbol{y} \iff \Psi \underbrace{\Lambda_1\Phi_1^T\boldsymbol{c}}_{=\boldsymbol{b}} = \boldsymbol{y}. \tag{2}$$

Straightforward computation shows that

$$Q(\boldsymbol{y}) = \boldsymbol{y}^T\mathrm{K}^{-1}\boldsymbol{y} = \boldsymbol{b}^T\mathrm{A}\boldsymbol{b},$$

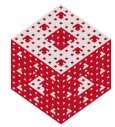
where

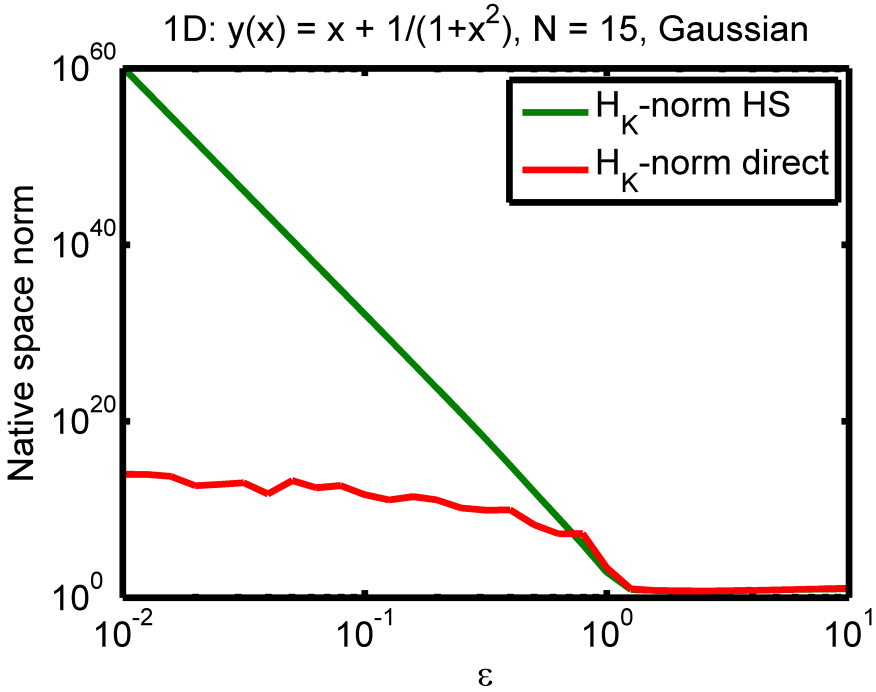
$$\mathrm{A} = \Lambda_1^{-1} + \mathrm{B}^T\Lambda_2\mathrm{B}, \qquad \mathrm{B} = \Phi_2^T\Phi_1^{-T}\Lambda_1^{-1}$$

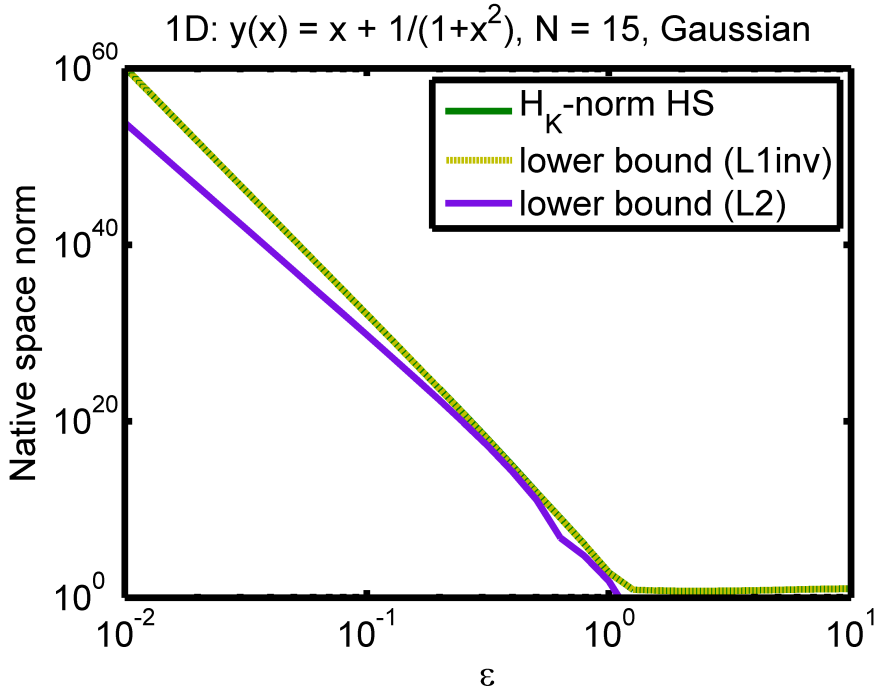
so that A is clearly symmetric and positive definite.
In particular,

$$Q(\boldsymbol{y}) = \boldsymbol{b}^T\Lambda_1^{-1}\boldsymbol{b} + \boldsymbol{b}^T\mathrm{B}^T\Lambda_2\mathrm{B}\boldsymbol{b} \geq \boldsymbol{b}^T\Lambda_1^{-1}\boldsymbol{b},$$

where $\boldsymbol{b}$ is computed stably via (2) and Λ_1^{-1} is given analytically.



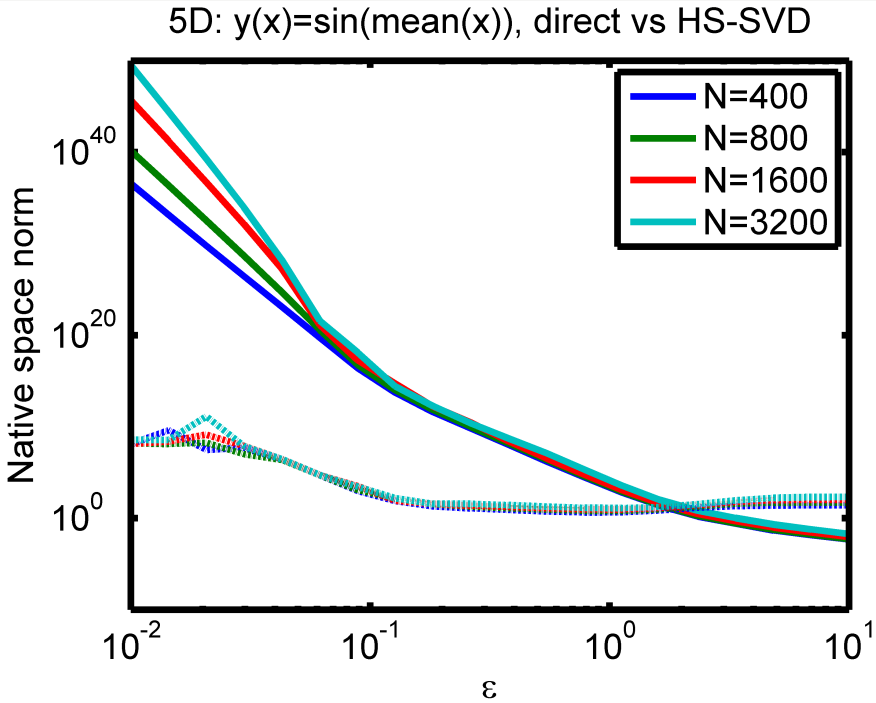


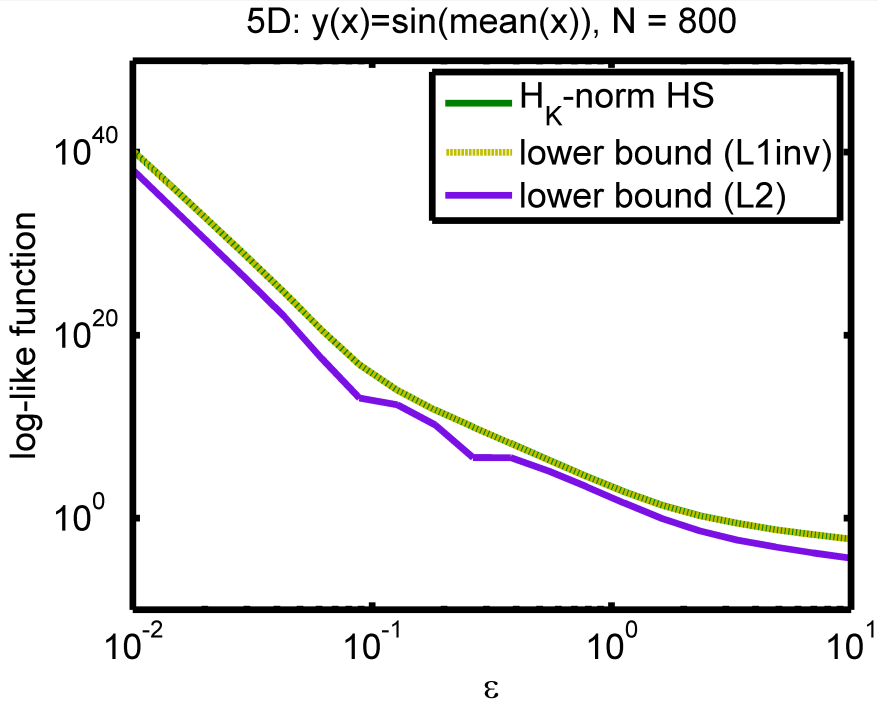


$$Q(\boldsymbol{y}) = \boldsymbol{b}^T \mathrm{A} \boldsymbol{b} = \boldsymbol{b}^T \Lambda_1^{-1} \boldsymbol{b} + \boldsymbol{b}^T \mathrm{B}^T \Lambda_2 \mathrm{B} \boldsymbol{b}$$

Note that $Q(\boldsymbol{y}) = \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} = \boldsymbol{c}^T \mathrm{K} \boldsymbol{c} = \boldsymbol{b}^T \mathrm{A} \boldsymbol{b}$ is the native space norm of the interpolant.

To statisticians this is known as the Mahalanobis distance.





$$Q(\boldsymbol{y}) = \boldsymbol{b}^T \mathrm{A} \boldsymbol{b} = \boldsymbol{b}^T \Lambda_1^{-1} \boldsymbol{b} + \boldsymbol{b}^T \mathrm{B}^T \Lambda_2 \mathrm{B} \boldsymbol{b}$$

Note that $Q(\boldsymbol{y}) = \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} = \boldsymbol{c}^T \mathrm{K} \boldsymbol{c} = \boldsymbol{b}^T \mathrm{A} \boldsymbol{b}$ is the native space norm of the interpolant.

To statisticians this is known as the Mahalanobis distance.

Stable MLE for Gaussian Interpolation in 1D

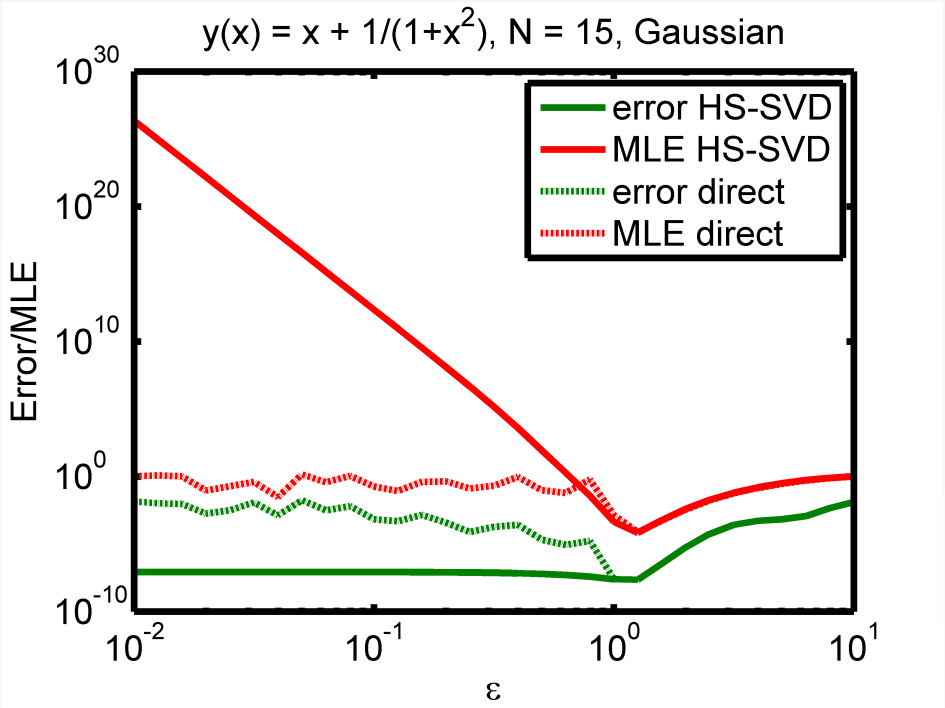
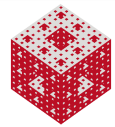
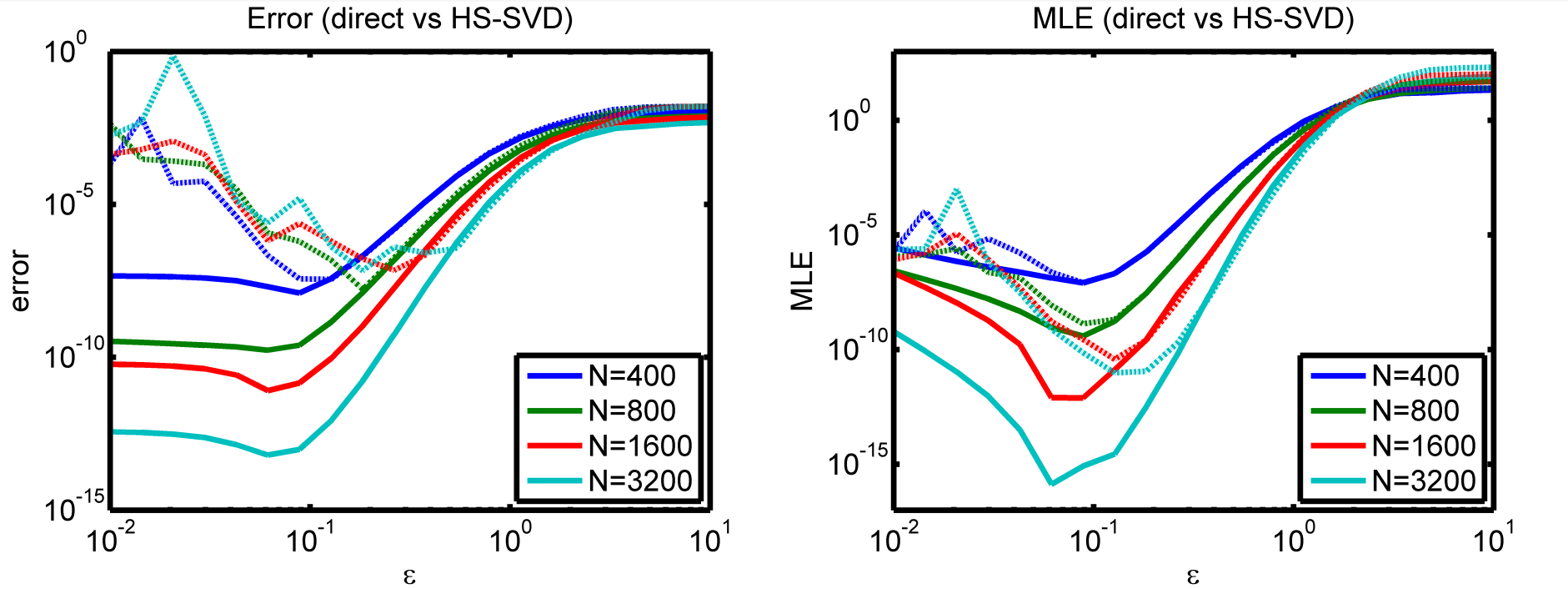


Figure: $N = 15$ Chebyshev points for $f(x) = x + \frac{1}{1+x^2}$ on $[-1, 1]$.

$$\widetilde{L}(\varepsilon; \boldsymbol{y}) = \frac{1}{N} \log \det(\mathrm{K}) + \log\left(\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}\right)$$

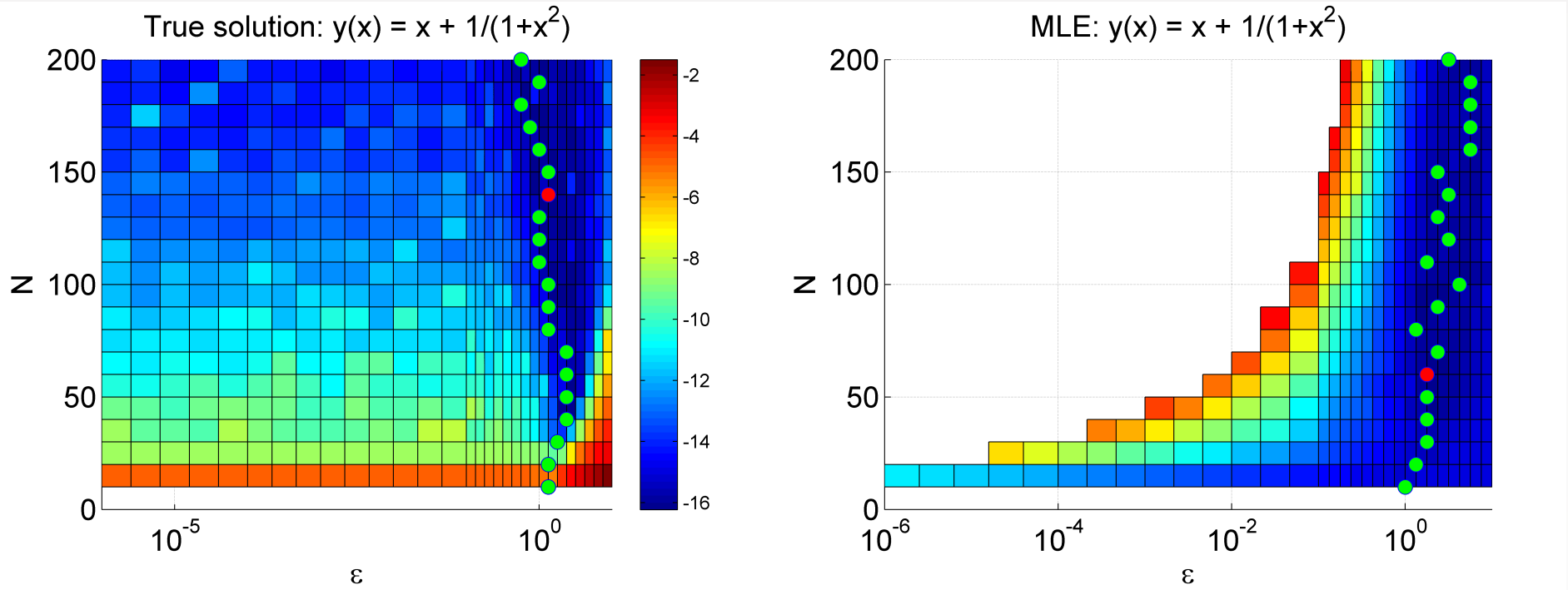


Stable MLE for Gaussian Interpolation in 5D



$y(\boldsymbol{x}) = \sin(\text{mean}(\boldsymbol{x}))$, using N Halton points
solid lines for HS-SVD, dashed lines for direct solve

MLE as a consistent predictor of "optimal" ε



True solution (left): overall optimal values (red dot):

$$\varepsilon = 1.333521, \qquad N = 140, \qquad \text{Error} = 5.8378 \times 10^{-17}$$

MLE (right): overall "optimal" values (red dot):

$$\varepsilon = 1.778279, \qquad N = 60, \qquad \text{Error} = 6.29907 \times 10^{-16}$$

MLE and flat polynomial limits

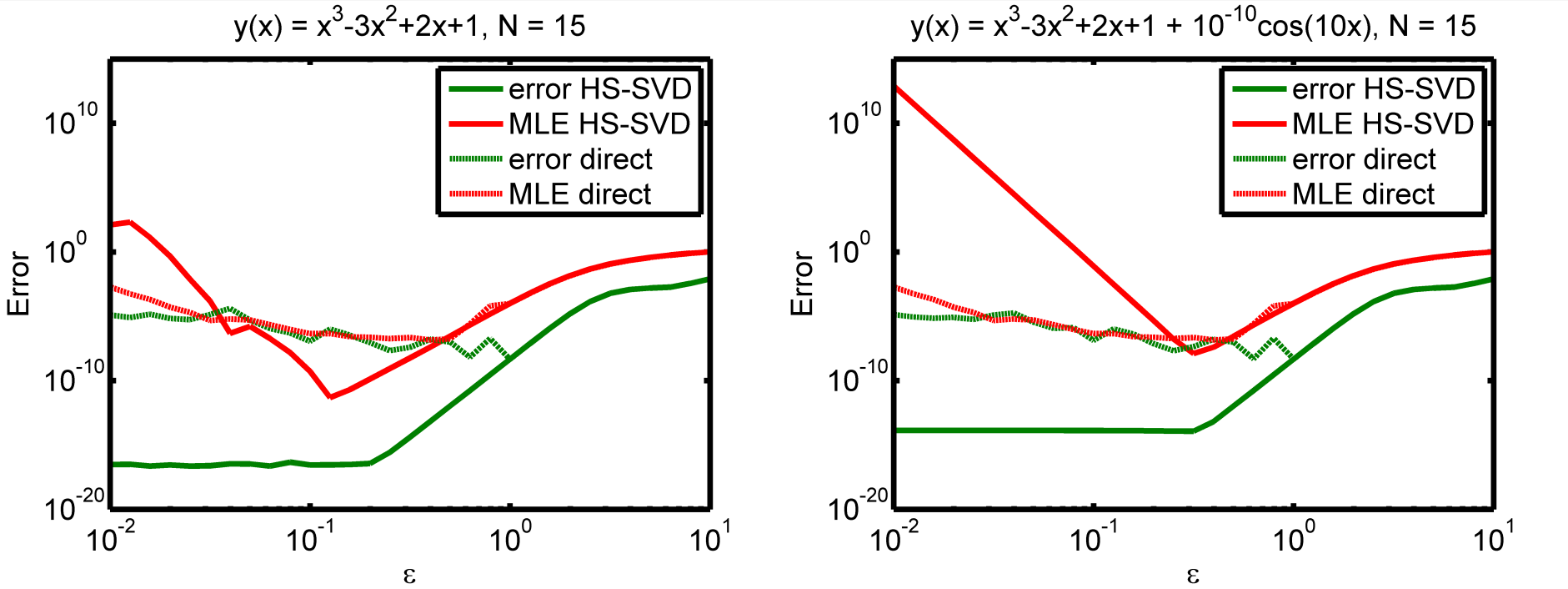
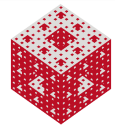


Figure: $N = 15$ Chebyshev points for $y(x) = x^3 - 3x^2 + 2x + 1$ and $y(x) = x^3 - 3x^2 + 2x + 1 + 10^{-10}\cos(10x)$ on $[-1, 1]$.

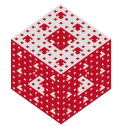
In both cases, the MLE predicts an ε-value that leads to optimal accuracy.

However, the MLE does not "allow" the (polynomial) flat limit since polynomials are not in the native space of Gaussians.



Summary

- Hilbert-Schmidt/Mercer expansion and Hilbert-Schmidt SVD provide a general and transparent framework for stable kernel computation
- Implementation depends on availability of Mercer series for specific kernels
  - some eigenfunctions are easier to obtain than others
  - some eigenfunctions are easier to handle than others
- Vast applications
  - function interpolation/approximation
  - parameter estimation (MLE, GCV)
  - numerical solution of PDEs (collocation, MFS, MPS)
  - ...
- Future outlook
  - implement for anisotropic Gaussians
  - HS-SVD for other kernels
  - MLE for low-rank approximation



References I

Rasmussen, C. E. and Williams, C. K. I.
Gaussian Processes for Machine Learning.
MIT Press, Cambridge, MA, 2006.

Cavoretto, R., Fasshauer, G. E. and McCourt, M. J.
Compact Matérn kernels and piecewise polynomial splines viewed from a Hilbert-Schmidt perspective.
submitted.

Fasshauer, G. E., Hickernell, F. J. and Woźniakowski, H.
Rate of convergence and tractability of the radial function approximation problem.
SIAM J. Numer. Anal. **50**/1 (2012), 247–271.

Fasshauer, G. E. and McCourt, M. J.
Stable evaluation of Gaussian RBF interpolants.
SIAM J. Scient. Comput. **34**/2 (2012), pp. A737–A762.

Fornberg, B. and Piret, C.
A stable algorithm for flat radial basis functions on a sphere.
SIAM J. Scient. Comput. **30**/1 (2008), pp. 60–80.