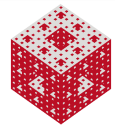
MATH 590: Meshfree Methods

Stable Computation via the Hilbert–Schmidt SVD

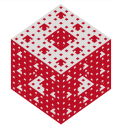
Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

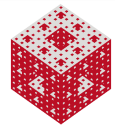
Fall 2014



Outline



Outline



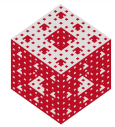
As we've mentioned several times before, there are a number of features that make positive definite kernels (or RBFs) so attractive to work with:

As we've mentioned several times before, there are a number of features that make positive definite kernels (or RBFs) so attractive to work with:

- Interpolants with "flat" kernels converge to polynomial interpolants.

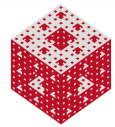
As we've mentioned several times before, there are a number of features that make positive definite kernels (or RBFs) so attractive to work with:

- Interpolants with "flat" kernels converge to polynomial interpolants.
- Gaussians (and certain other RBFs) provide dimension-independent, arbitrarily high, convergence rates for sufficiently "nice" data (i.e., with low effective dimension and coming from a smooth function).



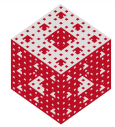
As we've mentioned several times before, there are a number of features that make positive definite kernels (or RBFs) so attractive to work with:

- Interpolants with "flat" kernels converge to polynomial interpolants.
- Gaussians (and certain other RBFs) provide dimension-independent, arbitrarily high, convergence rates for sufficiently "nice" data (i.e., with low effective dimension and coming from a smooth function).
- Numerical instability of interpolation/approximation algorithms can be overcome by using a "better" basis.



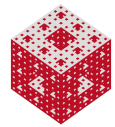
As we've mentioned several times before, there are a number of features that make positive definite kernels (or RBFs) so attractive to work with:

- Interpolants with "flat" kernels converge to polynomial interpolants.
- Gaussians (and certain other RBFs) provide dimension-independent, arbitrarily high, convergence rates for sufficiently "nice" data (i.e., with low effective dimension and coming from a smooth function).
- Numerical instability of interpolation/approximation algorithms can be overcome by using a "better" basis.
- All of this together establishes (smooth) RBFs as generalized spectral methods.



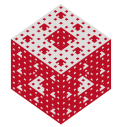
Using a Better Basis to Ensure Stability

- This idea has been well-known in approximation theory for a long time, e.g.,
  - B-splines as stable bases for piecewise polynomial splines [Sch81], or
  - Chebyshev polynomials instead of monomials [Tre13].



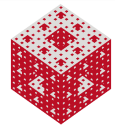
Using a Better Basis to Ensure Stability

- This idea has been well-known in approximation theory for a long time, e.g.,
  - B-splines as stable bases for piecewise polynomial splines [Sch81], or
  - Chebyshev polynomials instead of monomials [Tre13].
- In the RBF literature one can find this idea at least as early as the paper [BLB01] (see also [Fas07, Ch. 34]).



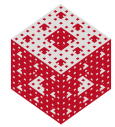
Using a Better Basis to Ensure Stability

- This idea has been well-known in approximation theory for a long time, e.g.,
  - B-splines as stable bases for piecewise polynomial splines [Sch81], or
  - Chebyshev polynomials instead of monomials [Tre13].
- In the RBF literature one can find this idea at least as early as the paper [BLB01] (see also [Fas07, Ch. 34]).
- The use of expansions in terms of eigenvalues and eigenfunctions of the Hilbert–Schmidt integral operator $\mathcal{K}$ associated with the kernel K to obtain stable bases for kernel spaces $\mathcal{H}_K(\mathcal{X})$ is discussed in [CFM14, Fas11a, Fas11b, FM12], and implicitly appeared in [FP08].



Using a Better Basis to Ensure Stability

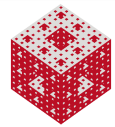
- This idea has been well-known in approximation theory for a long time, e.g.,
  - B-splines as stable bases for piecewise polynomial splines [Sch81], or
  - Chebyshev polynomials instead of monomials [Tre13].
- In the RBF literature one can find this idea at least as early as the paper [BLB01] (see also [Fas07, Ch. 34]).
- The use of expansions in terms of eigenvalues and eigenfunctions of the Hilbert–Schmidt integral operator $\mathcal{K}$ associated with the kernel K to obtain stable bases for kernel spaces $\mathcal{H}_K(\mathcal{X})$ is discussed in [CFM14, Fas11a, Fas11b, FM12], and implicitly appeared in [FP08].
  - Gaussian eigenvalues and eigenfunctions were presented in the previous chapter.
  - iterated Brownian bridge kernels were discussed in Chapter 6.



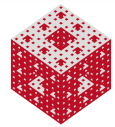
Using a Better Basis to Ensure Stability

- This idea has been well-known in approximation theory for a long time, e.g.,
  - B-splines as stable bases for piecewise polynomial splines [Sch81], or
  - Chebyshev polynomials instead of monomials [Tre13].
- In the RBF literature one can find this idea at least as early as the paper [BLB01] (see also [Fas07, Ch. 34]).
- The use of expansions in terms of eigenvalues and eigenfunctions of the Hilbert–Schmidt integral operator $\mathcal{K}$ associated with the kernel K to obtain stable bases for kernel spaces $\mathcal{H}_K(\mathcal{X})$ is discussed in [CFM14, Fas11a, Fas11b, FM12], and implicitly appeared in [FP08].
  - Gaussian eigenvalues and eigenfunctions were presented in the previous chapter.
  - iterated Brownian bridge kernels were discussed in Chapter 6.
- We now explain how to use such expansions to obtain the Hilbert–Schmidt SVD.

Outline

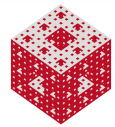


The Contour-Padé algorithm was the subject of Grady Wright's Ph.D. thesis [Wri03] and was reported in [FW04] (see also [Fas07, Ch. 17]).



The Contour-Padé algorithm was the subject of Grady Wright's Ph.D. thesis [Wri03] and was reported in [FW04] (see also [Fas07, Ch. 17]).

The aim of the Contour-Padé algorithm is to come up with a method that allows the computation and evaluation of RBF interpolants for infinitely smooth basic functions when the shape parameter ε tends to zero (including the limiting case).



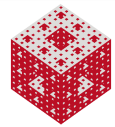
The Contour-Padé algorithm was the subject of Grady Wright's Ph.D. thesis [Wri03] and was reported in [FW04] (see also [Fas07, Ch. 17]).

The aim of the Contour-Padé algorithm is to come up with a method that allows the computation and evaluation of RBF interpolants for infinitely smooth basic functions when the shape parameter ε tends to zero (including the limiting case).

The starting point is to consider evaluation of the RBF interpolant

$$s_\varepsilon(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \kappa(\varepsilon \|\boldsymbol{x} - \boldsymbol{x}_j\|)$$

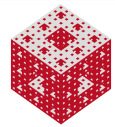
for a fixed evaluation point $\boldsymbol{x}$ as an analytic function of ε.



The key idea is to represent $s_\varepsilon(\boldsymbol{x})$ by a Laurent series in ε, and approximate the "negative part" of the series by a Padé approximant, i.e.,

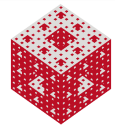
$$s_\varepsilon(\boldsymbol{x}) \approx r(\varepsilon) + \sum_{k=0}^{\infty} d_k \varepsilon^k,$$

where $r(\varepsilon)$ is the rational Padé approximant.



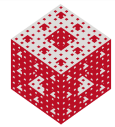
We then rewrite the interpolant in cardinal form, i.e., as

$$s_\varepsilon(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \kappa(\varepsilon \|\boldsymbol{x} - \boldsymbol{x}_j\|)$$



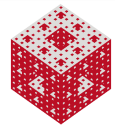
We then rewrite the interpolant in cardinal form, i.e., as

$$\begin{aligned} s_\varepsilon(\boldsymbol{x}) &= \sum_{j=1}^{N} c_j \kappa(\varepsilon \|\boldsymbol{x} - \boldsymbol{x}_j\|) \\ &= \boldsymbol{k}_\varepsilon(\boldsymbol{x})^T \boldsymbol{c} \end{aligned}$$



We then rewrite the interpolant in cardinal form, i.e., as

$$\begin{aligned} s_\varepsilon(\boldsymbol{x}) &= \sum_{j=1}^{N} c_j \kappa(\varepsilon\|\boldsymbol{x}-\boldsymbol{x}_j\|) \\ &= \boldsymbol{k}_\varepsilon(\boldsymbol{x})^T \boldsymbol{c} \\ &= \boldsymbol{k}_\varepsilon(\boldsymbol{x})^T \mathrm{K}_\varepsilon^{-1} \boldsymbol{y} \end{aligned}$$



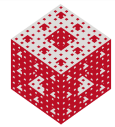
We then rewrite the interpolant in cardinal form, i.e., as

$$\begin{aligned} s_\varepsilon(\boldsymbol{x}) &= \sum_{j=1}^{N} c_j \kappa(\varepsilon\|\boldsymbol{x}-\boldsymbol{x}_j\|) \\ &= \boldsymbol{k}_\varepsilon(\boldsymbol{x})^T \boldsymbol{c} \\ &= \boldsymbol{k}_\varepsilon(\boldsymbol{x})^T \mathrm{K}_\varepsilon^{-1} \boldsymbol{y} \\ &= \left(\overset{\star}{\boldsymbol{u}}_\varepsilon(\boldsymbol{x})\right)^T \boldsymbol{y}, \end{aligned}$$

where $\boldsymbol{k}_\varepsilon(\boldsymbol{x})_j = \kappa(\varepsilon\|\boldsymbol{x}-\boldsymbol{x}_j\|)$, $(\mathrm{K}_\varepsilon)_{i,j} = \kappa(\varepsilon\|\boldsymbol{x}_i-\boldsymbol{x}_j\|)$, $\boldsymbol{c} = (c_1,\ldots,c_N)^T$, $\boldsymbol{y} = (y_1,\ldots,y_N)^T$, and

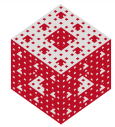
$$\overset{\star}{\boldsymbol{u}}_\varepsilon(\boldsymbol{x}) = \mathrm{K}_\varepsilon^{-1}\boldsymbol{k}_\varepsilon(\boldsymbol{x})$$

denotes the vector of values of the cardinal functions at $\boldsymbol{x}$.



Goal:

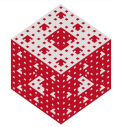
To stably compute the vector $\overset{\star}{\boldsymbol{u}}_{\varepsilon}$ for all values of $\varepsilon \geq 0$



Goal:

To stably compute the vector $\overset{\star}{\boldsymbol{u}}_\varepsilon$ for all values of $\varepsilon \geq 0$

- without explicitly forming the inverse K_ε^{-1} and

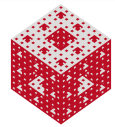


Goal:

To stably compute the vector $\overset{\star}{\boldsymbol{u}}_\varepsilon$ for all values of $\varepsilon \geq 0$

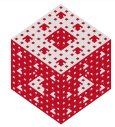
- without explicitly forming the inverse $\mathrm{K}_\varepsilon^{-1}$ and
- without computing the matrix vector product $\mathrm{K}_\varepsilon^{-1}\boldsymbol{k}_\varepsilon$.

Here the vectors $\overset{\star}{\boldsymbol{u}}_\varepsilon$ and $\boldsymbol{k}_\varepsilon$ are obtained by evaluating the vector functions $\overset{\star}{\boldsymbol{u}}_\varepsilon(\cdot)$ and $\boldsymbol{k}_\varepsilon(\cdot)$ on an appropriate evaluation grid.

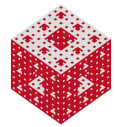


- The solution proposed by Wright and Fornberg is to use Cauchy's integral theorem to integrate around a circle in the complex ε-plane.

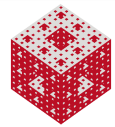
- The solution proposed by Wright and Fornberg is to use Cauchy's integral theorem to integrate around a circle in the complex ε-plane.
- The residuals (i.e., coefficients in the Laurent expansion) are obtained using the (inverse) fast Fourier transform.



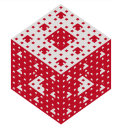
- The solution proposed by Wright and Fornberg is to use Cauchy's integral theorem to integrate around a circle in the complex ε-plane.
- The residuals (i.e., coefficients in the Laurent expansion) are obtained using the (inverse) fast Fourier transform.
- The terms with negative powers of ε are then approximated using a rational Padé approximant.



- The solution proposed by Wright and Fornberg is to use Cauchy's integral theorem to integrate around a circle in the complex ε-plane.
- The residuals (i.e., coefficients in the Laurent expansion) are obtained using the (inverse) fast Fourier transform.
- The terms with negative powers of ε are then approximated using a rational Padé approximant.
- The integration contour (usually a circle) has to lie between the region of instability near $\varepsilon = 0$ and possible branch point singularities that lie somewhere in the complex plane depending on the choice of κ.



- The solution proposed by Wright and Fornberg is to use Cauchy's integral theorem to integrate around a circle in the complex ε-plane.
- The residuals (i.e., coefficients in the Laurent expansion) are obtained using the (inverse) fast Fourier transform.
- The terms with negative powers of ε are then approximated using a rational Padé approximant.
- The integration contour (usually a circle) has to lie between the region of instability near $\varepsilon = 0$ and possible branch point singularities that lie somewhere in the complex plane depending on the choice of κ.
- Details of the method can be found in [FW04].



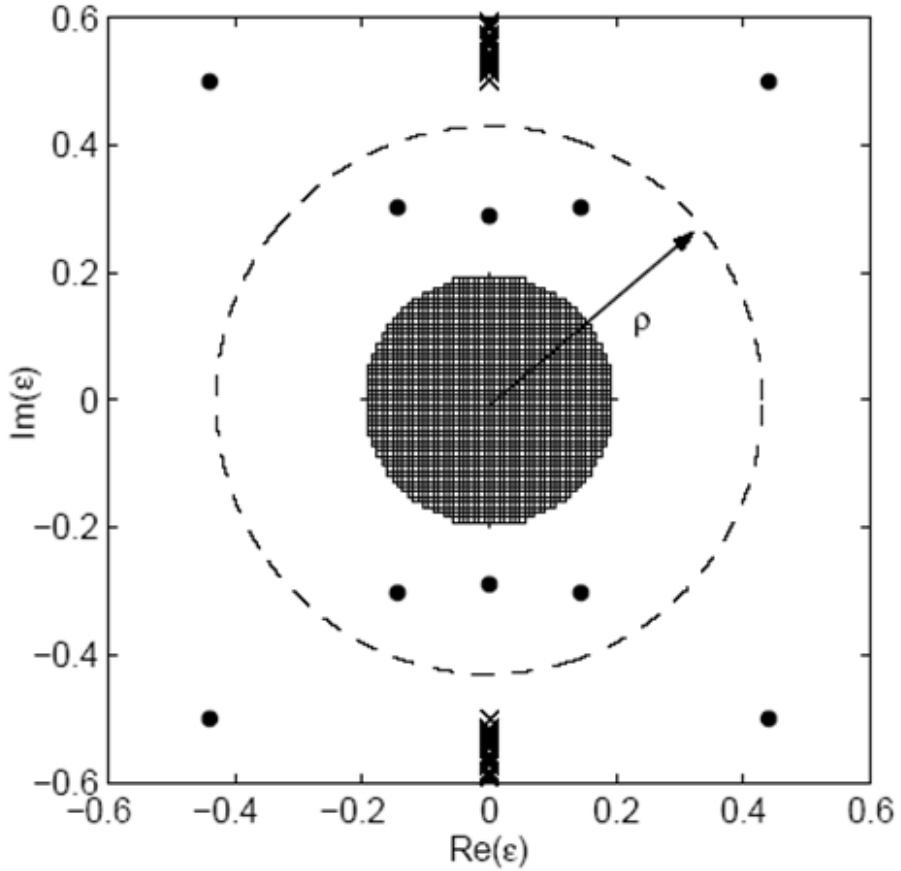
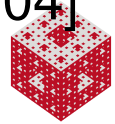


Figure 4: Structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane. The approximate area with ill-conditioning is marked with a line pattern; poles are marked with solid circles and branch points with ×'s.

from [FW04]



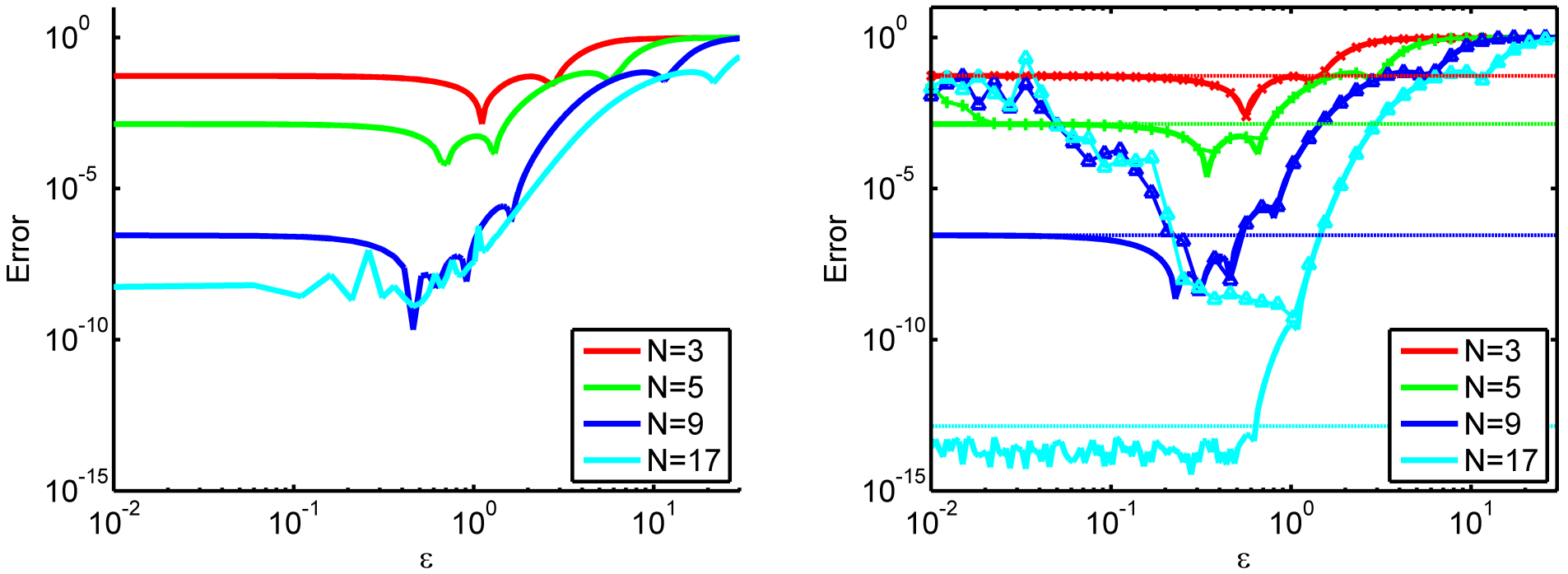


Figure: Optimal ε curves based on Contour-Padé (left) and Hilbert–Schmidt SVD (right) for interpolation to the sinc function with Gaussians in 1D for various choices of N uniform points.

Remark

- *The two methods perform roughly the same for small values of N.*
- *Hilbert–Schmidt SVD performs much better for $N = 17$.*

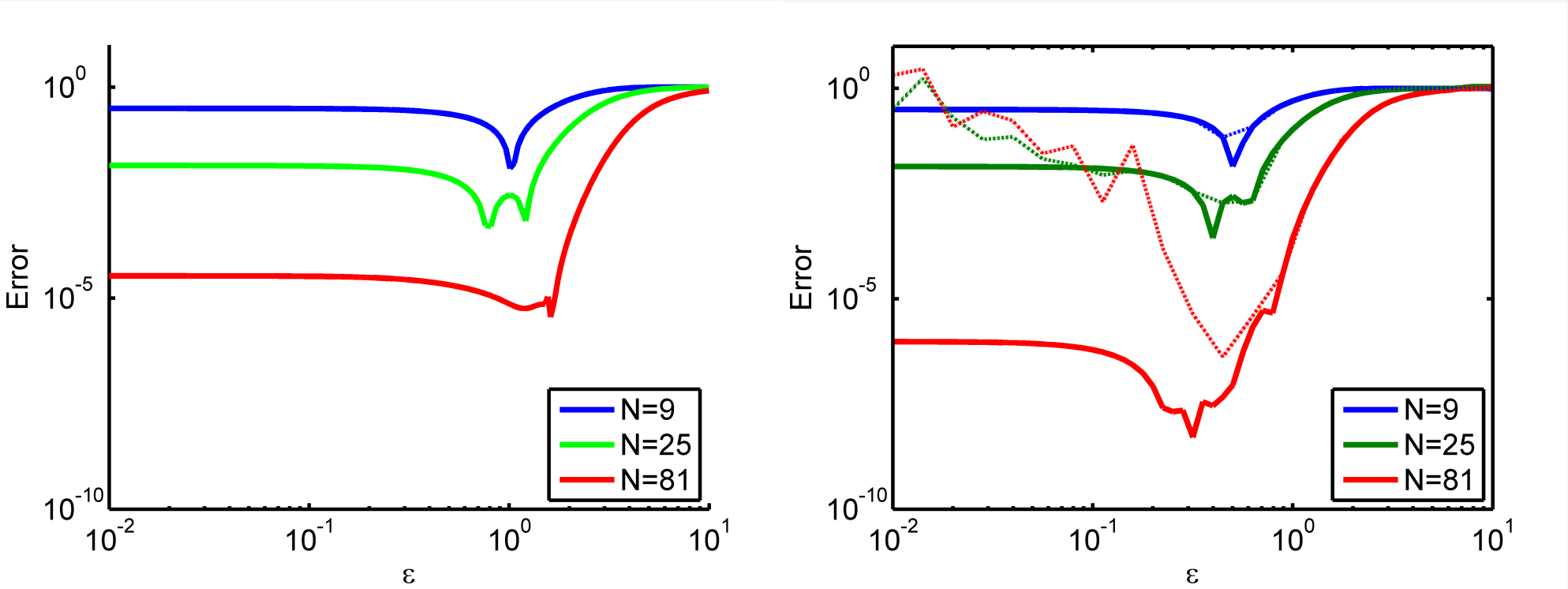


Figure: Optimal ε curves based on Contour-Padé (left) and Hilbert–Schmidt SVD (right) for interpolation to the 2D sinc function with Gaussians in for various choices of N Halton points.

Remark

- *Again, both methods perform equally well for small N, but Hilbert–Schmidt SVD is much more accurate for* $N = 81$.

Remark

The main drawback of the Contour-Padé algorithm is the fact that if N becomes too large then the region of ill-conditioning around the origin in the complex ε-plane and the branch point singularities will overlap.

Remark

The main drawback of the Contour-Padé algorithm is the fact that if N becomes too large then the region of ill-conditioning around the origin in the complex ε-plane and the branch point singularities will overlap.

This implies that the method can only be used with limited success.

Remark

The main drawback of the Contour-Padé algorithm is the fact that if N becomes too large then the region of ill-conditioning around the origin in the complex ε-plane and the branch point singularities will overlap.

This implies that the method can only be used with limited success.

Moreover, as the examples above show, the value of N that has to be considered "large" is unfortunately rather small. For the one-dimensional case the results for $N = 17$ already are affected by instabilities, and in the two-dimensional experiment $N = 81$ causes problems.

Remark

The main drawback of the Contour-Padé algorithm is the fact that if N becomes too large then the region of ill-conditioning around the origin in the complex ε-plane and the branch point singularities will overlap.

This implies that the method can only be used with limited success.

Moreover, as the examples above show, the value of N that has to be considered "large" is unfortunately rather small. For the one-dimensional case the results for $N = 17$ already are affected by instabilities, and in the two-dimensional experiment $N = 81$ causes problems.

We now want to work our way toward the Hilbert–Schmidt SVD (Gauss-QR) method of [FM12].

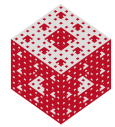
Outline

The following discussion is based mainly on [FM12], which developed a stable algorithm specifically for the Gaussian kernel.
That algorithm was referred to as Gauss-QR algorithm, but it is a special case of the Hilbert–Schmidt SVD. Similar algorithms are also known as RBF-QR algorithms.

The general framework applies to any kernel that has a Hilbert–Schmidt (or Mercer) series

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

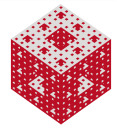
We now discuss the general framework and later look at kernel-specific issues that are important for the implementation.



The first main idea is to use the eigenexpansion of the kernel K to rewrite the matrix K from the interpolation problem as

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

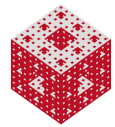
$$= \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_M(\boldsymbol{x}_1) & \dots \\ \vdots & & \vdots & \\ \varphi_1(\boldsymbol{x}_N) & \dots & \varphi_M(\boldsymbol{x}_N) & \dots \end{pmatrix} \begin{pmatrix} \lambda_1 & & & \\ & \ddots & & \\ & & \lambda_M & \\ & & & \ddots \end{pmatrix}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



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

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

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



But we can't compute with an infinite matrix, so we choose a truncation value M (aided by $\lambda_n \to 0$ as $n \to \infty$) 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 $M > N$.

We already looked at truncation lengths for iterated Brownian bridge kernels in Chapter 6 and HW 2.

Remark

- *A careful analysis of truncation lengths for general kernels given in series form (which includes our truncated Mercer series kernels) is presented in [GRZ13].*
  There it is shown that the truncation length M should needs to depend on N and the smallest eigenvalue of K. In fact, one should have

$$\sum_{n=M+1}^{\infty} \lambda_n \lesssim \frac{\lambda_{min}(\mathrm{K})}{N},$$

where $\lesssim$ encodes a dependence on the size of the eigenfunctions. If M is chosen in this way then interpolation error with the truncated kernel will be on the same order as with the full kernel.

Remark

- *A careful analysis of truncation lengths for general kernels given in series form (which includes our truncated Mercer series kernels) is presented in [GRZ13].*
  There it is shown that the truncation length M should needs to depend on N and the smallest eigenvalue of K. In fact, one should have

$$\sum_{n=M+1}^{\infty} \lambda_n \lesssim \frac{\lambda_{min}(\mathrm{K})}{N},$$

  where $\lesssim$ encodes a dependence on the size of the eigenfunctions. If M is chosen in this way then interpolation error with the truncated kernel will be on the same order as with the full kernel.
- *This criterion has only limited practical applicability — especially if we have a very ill-conditioned matrix K and we want to use the truncated kernel to obtain a stable basis.*

We now assume that $M > N$, so that Φ is "short and fat".
The key is to first partition the matrix Φ into two blocks Φ_1 and Φ_2 according to

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

Remark

- *We can think of the n-th column of Φ as a sample of the n-th eigenfunction obtained at the interpolation locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.*

We now assume that $M > N$, so that Φ is "short and fat".
The key is to first partition the matrix Φ into two blocks Φ_1 and Φ_2 according to

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

Remark

- *We can think of the n-th column of Φ as a sample of the n-th eigenfunction obtained at the interpolation locations $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$.*
- *Recall that the eigenfunctions are orthogonal in both $L_2(\Omega, \rho)$ and in $\mathcal{H}_K(\Omega)$. However, this does not imply orthogonality of the columns of Φ.*

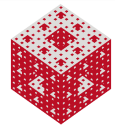
We now assume that $M > N$, so that Φ is "short and fat".
The key is to first partition the matrix Φ into two blocks Φ_1 and Φ_2 according to

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

Remark

- *We can think of the n-th column of Φ as a sample of the n-th eigenfunction obtained at the interpolation locations $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$.*
- *Recall that the eigenfunctions are orthogonal in both $L_2(\Omega, \rho)$ and in $\mathcal{H}_K(\Omega)$. However, this does not imply orthogonality of the columns of Φ.*
- *Thus, the Hilbert-Schmidt SVD does not employ orthogonal matrices Φ_1 (and later Ψ).*

By replacing Φ^T with its blocks and applying an analogous block partition to Λ we formally rewrite our eigen-decomposition of K:

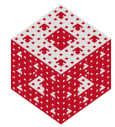


By replacing Φ^T with its blocks and applying an analogous block partition to Λ we formally rewrite our eigen-decomposition of K:

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

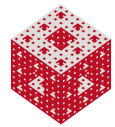
By replacing Φ^T with its blocks and applying an analogous block partition to Λ we formally rewrite our eigen-decomposition of K:

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



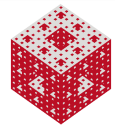
By replacing Φ^T with its blocks and applying an analogous block partition to Λ we formally rewrite our eigen-decomposition of K:

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



By replacing Φ^T with its blocks and applying an analogous block partition to Λ we formally rewrite our eigen-decomposition of K:

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

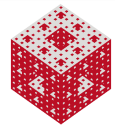


By replacing Φ^T with its blocks and applying an analogous block partition to Λ we formally rewrite our eigen-decomposition of K:

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

We have now identified

- a preconditioning matrix M and
- matrices
  - Ψ and Φ_1 of left and right Hilbert-Schmidt singular vectors, respectively, and
  - Λ_1 of Hilbert-Schmidt singular values.



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

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

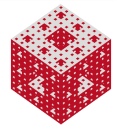
- We have found an invertible M such that $\Psi = KM^{-1}$ is better conditioned than K (without forming K or computing with it).

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

- We have found an invertible M such that $\Psi = KM^{-1}$ is better conditioned than K (without forming K or computing with it).
- We have diagonalized the matrix K, i.e.,

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

with diagonal matrix Λ_1 of Hilbert-Schmidt singular values. Here, as above, equality is only up to machine accuracy (i.e., M has to be chosen large enough).



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

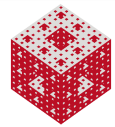
- We have found an invertible M such that $\Psi = \mathrm{K}\mathrm{M}^{-1}$ is better conditioned than K (without forming K or computing with it).
- We have diagonalized the matrix K, i.e.,

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

with diagonal matrix Λ_1 of Hilbert-Schmidt singular values. Here, as above, equality is only up to machine accuracy (i.e., M has to be chosen large enough).

The matrix Ψ in the same for both interpretations, and it can be computed stably.

Moreover, the notation above implies $\mathrm{M} = \Lambda_1 \Phi_1^T$.



Remark

- *Note that even though the Hilbert-Schmidt SVD*

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

looks very much like a regular SVD

$$K = U \Sigma V^T$$

the two are fundamentally different since for the Hilbert-Schmidt SVD we never form and factor the matrix K*.*

Remark

- *Note that even though the Hilbert-Schmidt SVD*

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

looks very much like a regular SVD

$$K = U \Sigma V^T$$

the two are fundamentally different since for the Hilbert-Schmidt SVD we never form and factor the matrix K*.*

- *Also, the matrices* Ψ *and* Φ_1 *are not orthogonal matrices, but they are generated by orthogonal functions.*

Remark

- *Note that even though the Hilbert-Schmidt SVD*

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

looks very much like a regular SVD

$$K = U \Sigma V^T$$

the two are fundamentally different since for the Hilbert-Schmidt SVD we never form and factor the matrix K.

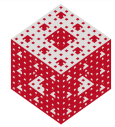
- *Also, the matrices* Ψ *and* Φ_1 *are not orthogonal matrices, but they are generated by orthogonal functions.*
- *Even though the notation* Λ_1 *might suggest a regularization of the matrix* K*, that is not the case. Such a regularization will occur only for sufficiently small truncation length* M *(in particular* $M < N$*, see later).*

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

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

which we recognize as

- the matrix Φ_1 corresponding to samples of the first N eigenfunctions
- plus an appropriate correction matrix.



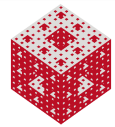
Viewed as functions, we have a new basis

$$\boldsymbol{\psi}(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$$

for the interpolation space

$$\text{span}\,\{K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N)\}$$

consisting of the appropriately corrected first N eigenfunctions:



Viewed as functions, we have a new basis

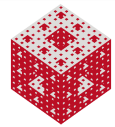
$$\boldsymbol{\psi}(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$$

for the interpolation space

$$\text{span}\,\{K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N)\}$$

consisting of the appropriately corrected first N eigenfunctions:

$$\begin{aligned}
\boldsymbol{k}(\boldsymbol{x})^T &= \boldsymbol{\phi}(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \\
&= \left[(\varphi_1(\boldsymbol{x}), \ldots, \varphi_N(\boldsymbol{x})) + (\varphi_{N+1}(\boldsymbol{x}), \ldots) \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \right] \Lambda_1 \Phi_1^T \\
&= \boldsymbol{\psi}(\boldsymbol{x})^T \Lambda_1 \Phi_1^T
\end{aligned}$$



Viewed as functions, we have a new basis

$$\boldsymbol{\psi}(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$$

for the interpolation space

$$\operatorname{span}\{K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N)\}$$

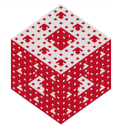
consisting of the appropriately corrected first N eigenfunctions:

$$\begin{aligned}
\boldsymbol{k}(\boldsymbol{x})^T &= \boldsymbol{\phi}(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \\
&= \left[(\varphi_1(\boldsymbol{x}), \ldots, \varphi_N(\boldsymbol{x})) + (\varphi_{N+1}(\boldsymbol{x}), \ldots) \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \right] \Lambda_1 \Phi_1^T \\
&= \boldsymbol{\psi}(\boldsymbol{x})^T \Lambda_1 \Phi_1^T
\end{aligned}$$

Remark

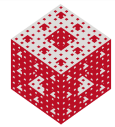
The data-dependence of the new basis is captured by the "correction" term (since Φ_1 and Φ_2 depend on the center locations). The new basis is more stable since we have removed Λ_1.

The particular structure of the correction term $\Phi_2 \left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}\right]$ is important for the success of the method:



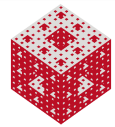
The particular structure of the correction term $\Phi_2 \left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}\right]$ is important for the success of the method:

- Since $\lambda_n \to 0$ as $n \to \infty$ the eigenvalues in Λ_2 are smaller than those in Λ_1 and so $\Phi_2^T \Phi_1^{-T}$ usually does not blow up.



The particular structure of the correction term $\Phi_2 \left[\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}\right]$ is important for the success of the method:

- Since $\lambda_n \to 0$ as $n \to \infty$ the eigenvalues in Λ_2 are smaller than those in Λ_1 and so $\Phi_2^T \Phi_1^{-T}$ usually does not blow up.
- Moreover, the multiplications by Λ_2 and Λ_1^{-1} can be done analytically.
- This avoids stability issues associated with possible
  - underflow (for the Gaussian kernel, entries in Λ_2 are as small as ε^{2M-2}) or
  - overflow (for the Gaussian, entries in Λ_1^{-1} are as large as ε^{-2N-2}).



Relation to RBF-QR [FP08]

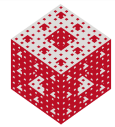
Additional stability in the computation of the correction matrix

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

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.



Relation to RBF-QR [FP08]

Additional stability in the computation of the correction matrix

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

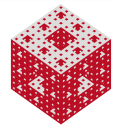
in particular, in the formation of $\Phi_2^T \Phi_1^{-T}$, is achieved via a QR decomposition of Φ, i.e.,

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

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

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

However, we rarely find this to be necessary.



Summary: How to use the Hilbert–Schmidt SVD

Instead of solving the "original" problem

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

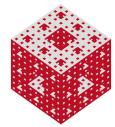
potentially yielding inaccurate coefficients which are multiplied against poorly conditioned basis functions, we now solve

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

with a new basis and new set of coefficients which we evaluate via

$$\begin{aligned} s(\boldsymbol{x}) &= \boldsymbol{k}(\boldsymbol{x})^T \mathrm{K}^{-1} \boldsymbol{y} \\ &= \boldsymbol{\psi}(\boldsymbol{x})^T \Lambda_1 \Phi_1^T \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1} \boldsymbol{y} \\ &= \boldsymbol{\psi}(\boldsymbol{x})^T \Psi^{-1} \boldsymbol{y} \end{aligned}$$

so that all the ill-conditioning from Λ_1 is gone.



Summary: How to use the Hilbert–Schmidt SVD

Instead of solving the "original" problem

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

potentially yielding inaccurate coefficients which are multiplied against poorly conditioned basis functions, we now solve

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

with a new basis and new set of coefficients which we evaluate via

$$\begin{aligned} s(\boldsymbol{x}) &= \boldsymbol{k}(\boldsymbol{x})^T \mathrm{K}^{-1} \boldsymbol{y} \\ &= \boldsymbol{\psi}(\boldsymbol{x})^T \Lambda_1 \Phi_1^T \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1} \boldsymbol{y} \\ &= \boldsymbol{\psi}(\boldsymbol{x})^T \Psi^{-1} \boldsymbol{y} \end{aligned}$$

so that all the ill-conditioning from Λ_1 is gone.

Note

$\boldsymbol{\psi}(\cdot)^T \Psi^{-1} = \boldsymbol{k}(\cdot)^T \mathrm{K}^{-1}$ provides fresh look at cardinal functions.

Implementation for Iterated Brownian Bridge Kernels

The Hilbert-Schmidt series is of the form

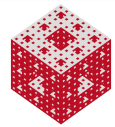
$$K_{\beta,\varepsilon}(x,z) = \sum_{n=1}^{\infty} \frac{2}{\left(n^2\pi^2 + \varepsilon^2\right)^{\beta}} \sin(n\pi x)\sin(n\pi z),$$

with Hilbert-Schmidt eigenvalues and eigenfunctions given by

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

Clearly,

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



Implementation for Iterated Brownian Bridge Kernels

The Hilbert-Schmidt series is of the form

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

with Hilbert-Schmidt eigenvalues and eigenfunctions given by

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

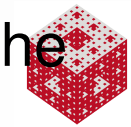
Clearly,

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

Therefore, in Chapter 6 we decided to use the truncation length

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

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x (the *ceiling* of x).



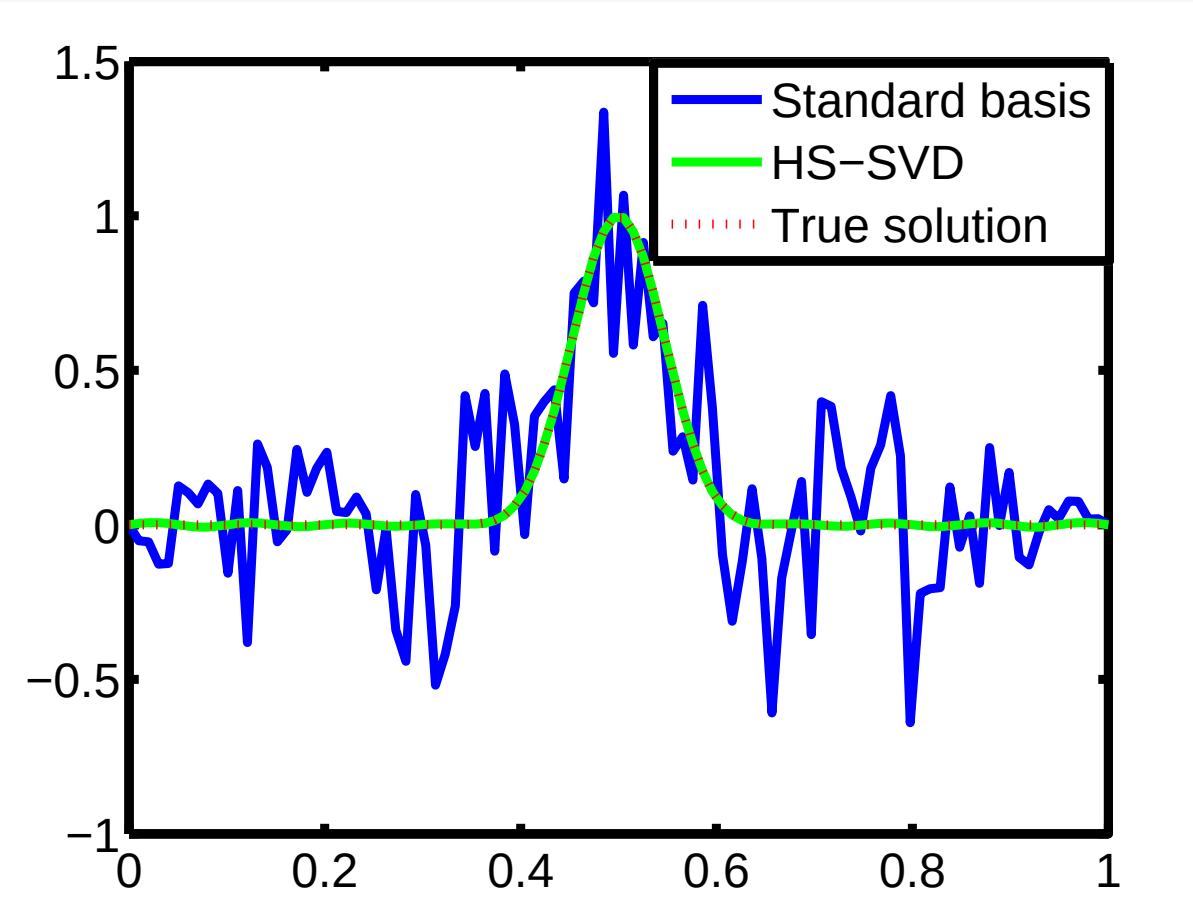
Program (MaternQRDemo.m)

```
N = 21; x = linspace(0,1,N)'; x = x(2:N-1);
N = N-2;  xx = linspace(0,1,100)';
f = @(x) .25^(-28)*max(x-.25,0).^14.*max(.75-x,0).^14; y = f(x);
ep = 1; beta = 7; % DO NOT use with beta < 3 !!
phifunc = @(n,x) sqrt(2)*sin(pi*x*n);
lambdafunc = @(n) ((n*pi).^2+ep^2).^(-beta);
M = max(N,ceil(1/pi*sqrt(eps^(-1/beta)*(N^2*pi^2+ep^2)-ep^2)));
Lambda = diag(lambdafunc(1:M));
Phi = phifunc(1:M,x);
K = Phi*Lambda*Phi';  c = K\y;
Phi_eval = phifunc(1:M,xx);
y_standard = Phi_eval*Lambda*Phi'*c;
Phi_1 = Phi(:,1:N);  Phi_2 = Phi(:,N+1:end);
Lambda_1 = Lambda(1:N,1:N);  Lambda_2 = Lambda(N+1:M,N+1:M);
Correction = Lambda_2*(Phi_1\Phi_2)'/Lambda_1;
Psi = Phi*[eye(N);Correction];
Psi_eval = Phi_eval*[eye(N);Correction];
y_HS = Psi_eval*(Psi\y);
plot(xx,y_standard,'linewidth',2),hold on
plot(xx,y_HS,'g',xx,f(xx),':r','linewidth',3)
```

Standard RBF vs. HS-SVD Interpolation

We use

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



Note: cond(K) $= 3.4 \times 10^{17}$

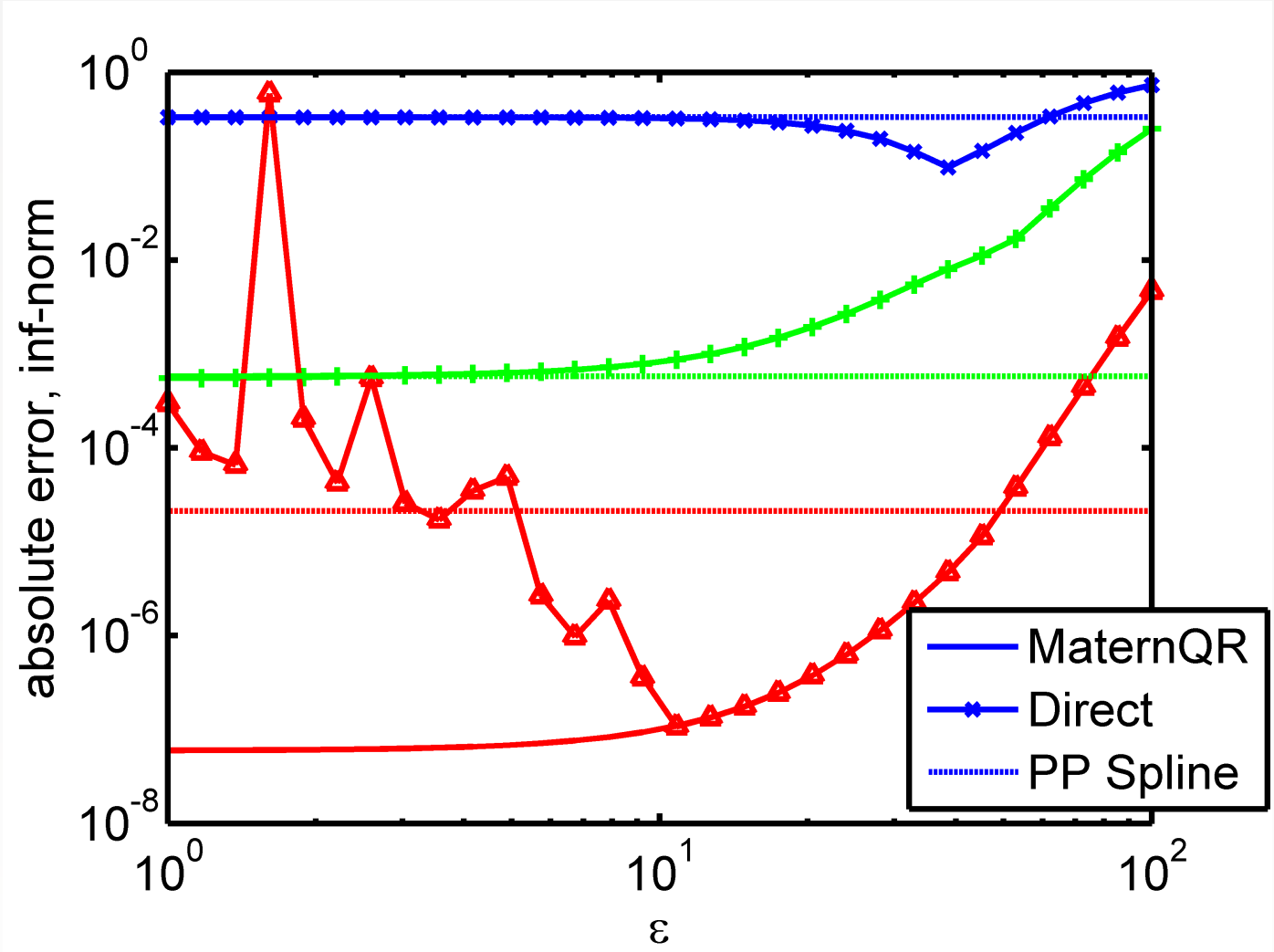
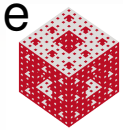


Figure: Comparison of different methods for iterated Brownian bridge interpolation with $K_{5,\varepsilon}$ to $f(x) = 30x^2(1-x)^2\sin(2\pi x)^4$ at $N = 10, 20, 40$ Chebyshev points. The piecewise polynomial interpolant (implemented with Bernoulli polynomials) is ill-conditioned for $N = 40$.



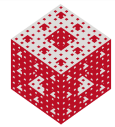
The implementation for Gaussian kernels is much more complicated.

- Evaluation of the eigenfunctions

$$\varphi_n(x) = \frac{\sqrt[8]{1+\left(\frac{2\varepsilon}{\alpha}\right)^2}}{\sqrt{2^n n!}} \mathrm{e}^{-\left(\sqrt{1+\left(\frac{2\varepsilon}{\alpha}\right)^2}-1\right)\frac{\alpha^2 x^2}{2}} H_n\left(\sqrt[4]{1+\left(\frac{2\varepsilon}{\alpha}\right)^2}\alpha x\right)$$

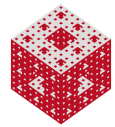
requires lots of care. The different factors of the eigenfunctions may become extremely large or extremely small.

- To ensure safe computation of the product form of the eigenfunctions they are evaluated using logarithms.
- A number of asymptotic expansions are used in the GaussQR implementation for different ranges of the argument of φ (see [McC13] for details).



- Choice of the truncation length M is not as simple as for the RBF-QR implementation with iterated Brownian bridge kernels since the eigenfunctions are more complicated.

- Choice of the truncation length M is not as simple as for the RBF-QR implementation with iterated Brownian bridge kernels since the eigenfunctions are more complicated.
- The global scale parameter α is an additional parameter that needs to be chosen carefully. It is needed to obtain the Hilbert-Schmidt expansion and has a significant (not yet fully understood) effect on the practical implementation of GaussQR.



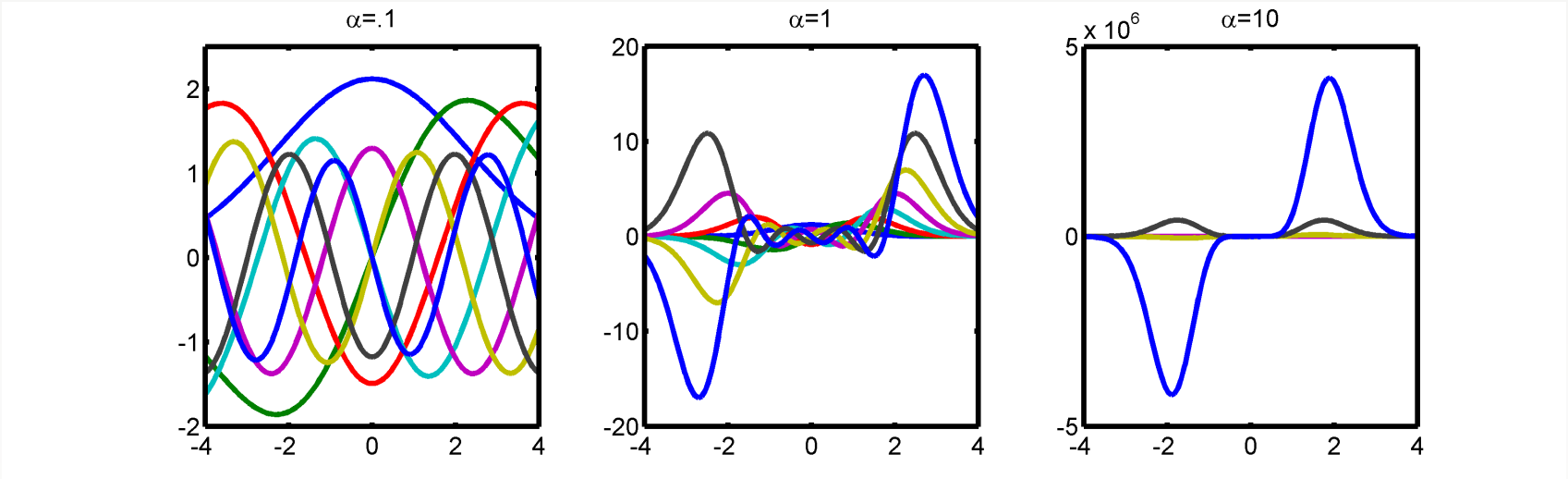
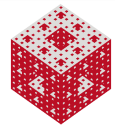


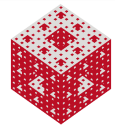
Figure: The first eight Gaussian eigenfunctions with $\varepsilon = 1$ and $\alpha = 0.1, 1, 10$.

▸ Run `GaussianEigenfunctions.cdf`

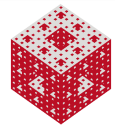


- In particular, the interplay of ε, M, and α needs to be further investigated.

- In particular, the interplay of ε, M, and α needs to be further investigated.
- For higher-dimensional applications
  - different values of ε and α for different coordinates, i.e., anisotropic kernels, can be used (but haven't been yet),
  - the sorting order of eigenfunctions corresponding to eigenvalues of the same magnitude is currently done rather arbitrarily.



As an alternative to the RBF-QR algorithm which is designed to reproduce all the entries of K up to machine precision we consider two different regression approaches:

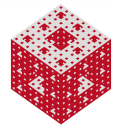


As an alternative to the RBF-QR algorithm which is designed to reproduce all the entries of K up to machine precision we consider two different regression approaches:

- using early truncation of the Hilbert-Schmidt series, i.e., a basis built from the first $M < N$ eigenfunctions, and

As an alternative to the RBF-QR algorithm which is designed to reproduce all the entries of K up to machine precision we consider two different regression approaches:

- using early truncation of the Hilbert-Schmidt series, i.e., a basis built from the first $M < N$ eigenfunctions, and
- using a truncated Hilbert-Schmidt SVD.

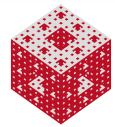


As an alternative to the RBF-QR algorithm which is designed to reproduce all the entries of K up to machine precision we consider two different regression approaches:

- using early truncation of the Hilbert-Schmidt series, i.e., a basis built from the first $M < N$ eigenfunctions, and
- using a truncated Hilbert-Schmidt SVD.

Remark

- *The first approach is much simpler, but data-independent. This may have advantages and disadvantages.*



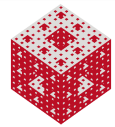
As an alternative to the RBF-QR algorithm which is designed to reproduce all the entries of K up to machine precision we consider two different regression approaches:

- using early truncation of the Hilbert-Schmidt series, i.e., a basis built from the first $M < N$ eigenfunctions, and
- using a truncated Hilbert-Schmidt SVD.

Remark

- *The first approach is much simpler, but data-independent. This may have advantages and disadvantages.*
- *The second approach requires all the work to create the matrix Ψ, but has inherently the same data-dependence as the matrix K.*

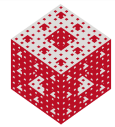
We now discuss both of these.



RBF-QRr

We want to use $M < N$ eigenfunctions to produce a low-rank approximation to the RBF interpolant based on N pieces of data.
The motivation is to

- eliminate high-order eigenfunctions which contribute very little to the solution, but increase computational cost.
- This may reduce the sensitivity of the solution to α.
- In particular, experiments have shown that the choice of an "optimal" α depends on ε and is also more sensitive with increasing M.



In order to introduce this problem in the same context as the interpolatory RBF-QR we assume

- that $M \leq N$ is fixed and
- set all the eigenvalues λ_n, $n = M = 1, \ldots, N$ to zero.

This results in an approximate decomposition of the kernel matrix

$$\begin{aligned} \mathrm{K} &\approx \Phi \tilde{\Lambda} \Phi^T \\ &= \begin{pmatrix} \Phi_1 & \Phi_2 \end{pmatrix} \begin{pmatrix} \Lambda_1 & \\ & 0 \end{pmatrix} \begin{pmatrix} \Phi_1 & \Phi_2 \end{pmatrix}^T, \end{aligned}$$

where

- Φ_1 is based on the first M eigenfunctions,
- Λ_1 contains the first M (and only nonzero) eigenvalues, and
- Φ_2 contains the remaining $N - M$ eigenfunctions.

To get the new basis matrix Ψ for the RBF-QR method we had to multiply K by M^{-1}.

To get the new basis matrix Ψ for the RBF-QR method we had to multiply K by M^{-1}.

We define the matrix[1] M analogously to before:

$$M = \tilde{\Lambda}\Phi^T.$$

[1] This would have to be called a "low-rank preconditioning" matrix.

To get the new basis matrix Ψ for the RBF-QR method we had to multiply K by M^{-1}.

We define the matrix[1] M analogously to before:

$$\mathrm{M} = \tilde{\Lambda}\Phi^T.$$

However, since $\tilde{\Lambda}$ is not invertible we use its pseudoinverse (Φ is $N \times N$ and invertible, so we don't need to use a QR decomposition here):

$$\mathrm{M}^\dagger = \Phi^{-T}\tilde{\Lambda}^\dagger = \Phi^{-T}\begin{pmatrix}\Lambda_1^{-1} & \\ & 0\end{pmatrix}.$$

[1] This would have to be called a "low-rank preconditioning" matrix.

To get the new basis matrix Ψ for the RBF-QR method we had to multiply K by M^{-1}.

We define the matrix[1] M analogously to before:

$$\mathrm{M} = \tilde{\Lambda}\Phi^T.$$

However, since $\tilde{\Lambda}$ is not invertible we use its pseudoinverse (Φ is $N \times N$ and invertible, so we don't need to use a QR decomposition here):

$$\mathrm{M}^\dagger = \Phi^{-T}\tilde{\Lambda}^\dagger = \Phi^{-T}\begin{pmatrix}\Lambda_1^{-1} & \\ & 0\end{pmatrix}.$$

This means that our new basis functions are given by

$$\boldsymbol{\psi}(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T\mathrm{M}^\dagger.$$

[1] This would have to be called a "low-rank preconditioning" matrix.

We can rewrite

$$\psi(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T \mathrm{M}^\dagger$$

in terms of the eigenfunctions and get

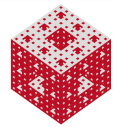
$$\psi(\boldsymbol{x}) \quad = \quad (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\mathrm{M}^\dagger$$

We can rewrite

$$\boldsymbol{\psi}(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T \mathrm{M}^\dagger$$

in terms of the eigenfunctions and get

$$\begin{aligned} \boldsymbol{\psi}(\boldsymbol{x}) &= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\mathrm{M}^\dagger \\ &= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\Phi^{-T}\tilde{\Lambda}^\dagger \end{aligned}$$

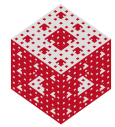


We can rewrite

$$\psi(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T \mathrm{M}^\dagger$$

in terms of the eigenfunctions and get

$$\begin{aligned}
\psi(\boldsymbol{x}) &= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\mathrm{M}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\Phi^{-T}\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\tilde{\Lambda}^\dagger
\end{aligned}$$

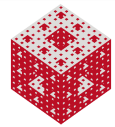


We can rewrite

$$\psi(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T \mathrm{M}^\dagger$$

in terms of the eigenfunctions and get

$$\begin{aligned}
\psi(\boldsymbol{x}) &= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\mathrm{M}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\Phi^{-T}\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\begin{pmatrix} \mathrm{I}_M & \\ & 0 \end{pmatrix}
\end{aligned}$$

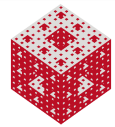


We can rewrite

$$\psi(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T \mathrm{M}^\dagger$$

in terms of the eigenfunctions and get

$$\begin{aligned}
\psi(\boldsymbol{x}) &= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\mathrm{M}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\Phi^{-T}\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\begin{pmatrix} \mathrm{I}_M & \\ & 0 \end{pmatrix} \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_M(\boldsymbol{x}) \quad 0 \quad \dots \quad 0).
\end{aligned}$$



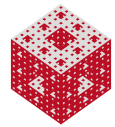
We can rewrite

$$\psi(\boldsymbol{x})^T = \boldsymbol{k}(\boldsymbol{x})^T \mathrm{M}^\dagger$$

in terms of the eigenfunctions and get

$$\begin{aligned}
\psi(\boldsymbol{x}) &= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\mathrm{M}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\Phi^T\Phi^{-T}\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\Lambda\tilde{\Lambda}^\dagger \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_N(\boldsymbol{x}))\begin{pmatrix} \mathrm{I}_M & \\ & 0 \end{pmatrix} \\
&= (\varphi_1(\boldsymbol{x}) \quad \dots \quad \varphi_M(\boldsymbol{x}) \quad 0 \quad \dots \quad 0).
\end{aligned}$$

As a result we have set the last $N - M$ eigenfunctions equal to zero.



Recasting the original linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ in terms of the new basis then gives

$$\begin{pmatrix} \Phi_1 & 0 \end{pmatrix} \boldsymbol{b} = \boldsymbol{y}.$$

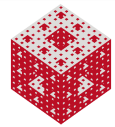
Recasting the original linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ in terms of the new basis then gives

$$\begin{pmatrix} \Phi_1 & 0 \end{pmatrix} \boldsymbol{b} = \boldsymbol{y}.$$

Solving this in a least-squares sense requires solving

$$\min_{\boldsymbol{b}} \left\| \begin{pmatrix} \Phi_1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} - \boldsymbol{y} \right\|_2^2 \quad \Longleftrightarrow \quad \min_{\boldsymbol{b}} \|\Phi_1 \boldsymbol{b}_1 - \boldsymbol{y}\|_2^2 ,$$

where $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are of length M and $N - M$, respectively.



Recasting the original linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ in terms of the new basis then gives

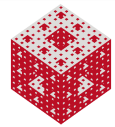
$$\begin{pmatrix} \Phi_1 & 0 \end{pmatrix} \boldsymbol{b} = \boldsymbol{y}.$$

Solving this in a least-squares sense requires solving

$$\min_{\boldsymbol{b}} \left\| \begin{pmatrix} \Phi_1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} - \boldsymbol{y} \right\|_2^2 \quad \Longleftrightarrow \quad \min_{\boldsymbol{b}} \|\Phi_1 \boldsymbol{b}_1 - \boldsymbol{y}\|_2^2,$$

where $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are of length M and $N - M$, respectively.
Therefore the RBF-QRr regression solution is especially simple:

$$\boldsymbol{b}_1 = \Phi_1^{\dagger} \boldsymbol{y}.$$

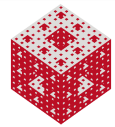


Truncated Hilbert-Schmidt SVD

In this case we obtain the decomposition of K as explained in the HS-SVD section as

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

where we compute the full matrix Ψ including the data-dependent correction part based on $\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}$.



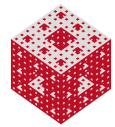
Truncated Hilbert-Schmidt SVD

In this case we obtain the decomposition of K as explained in the HS-SVD section as

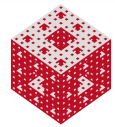
$$K = \Psi \Lambda_1 \Phi_1^T,$$

where we compute the full matrix Ψ including the data-dependent correction part based on $\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}$.

The truncated Hilbert-Schmidt SVD then zeros eigenvalues in Λ_1.



Outline



If we want to move to higher dimensions, then using kernels in product form is most advantageous.

If we want to move to higher dimensions, then using kernels in product form is most advantageous.

Example

Gaussian kernel

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

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

If we want to move to higher dimensions, then using kernels in product form is most advantageous.

Example

Gaussian kernel

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

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

If we want to move to higher dimensions, then using kernels in product form is most advantageous.

Example

Gaussian kernel

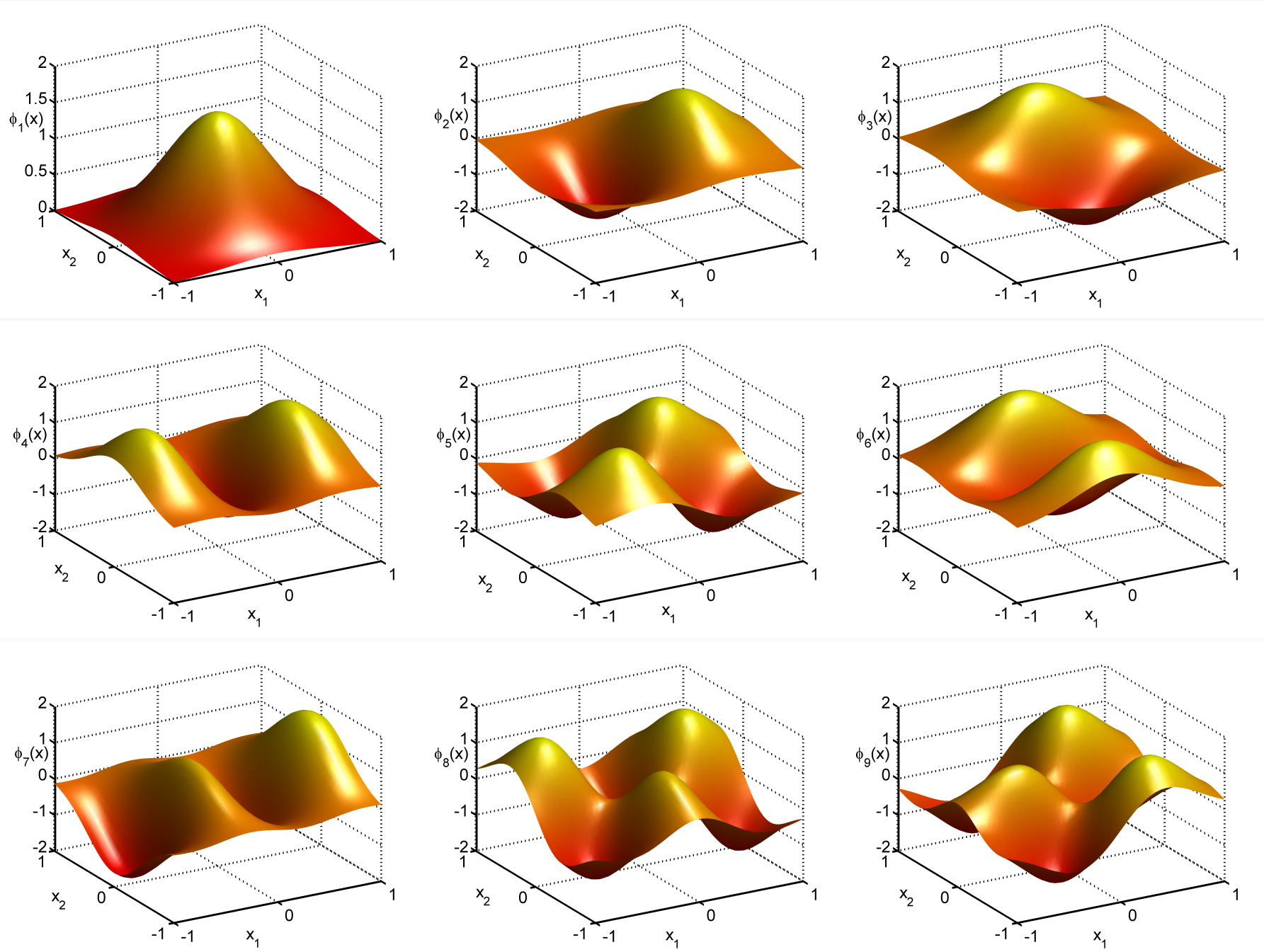
$$K(\boldsymbol{x},\boldsymbol{z}) = \mathrm{e}^{-\varepsilon^2\|\boldsymbol{x}-\boldsymbol{z}\|_2^2} = \mathrm{e}^{-\sum_{\ell=1}^{d}\varepsilon_\ell^2(x_\ell-z_\ell)^2} = \prod_{\ell=1}^{d}\mathrm{e}^{-\varepsilon_\ell^2(x_\ell-z_\ell)^2}$$

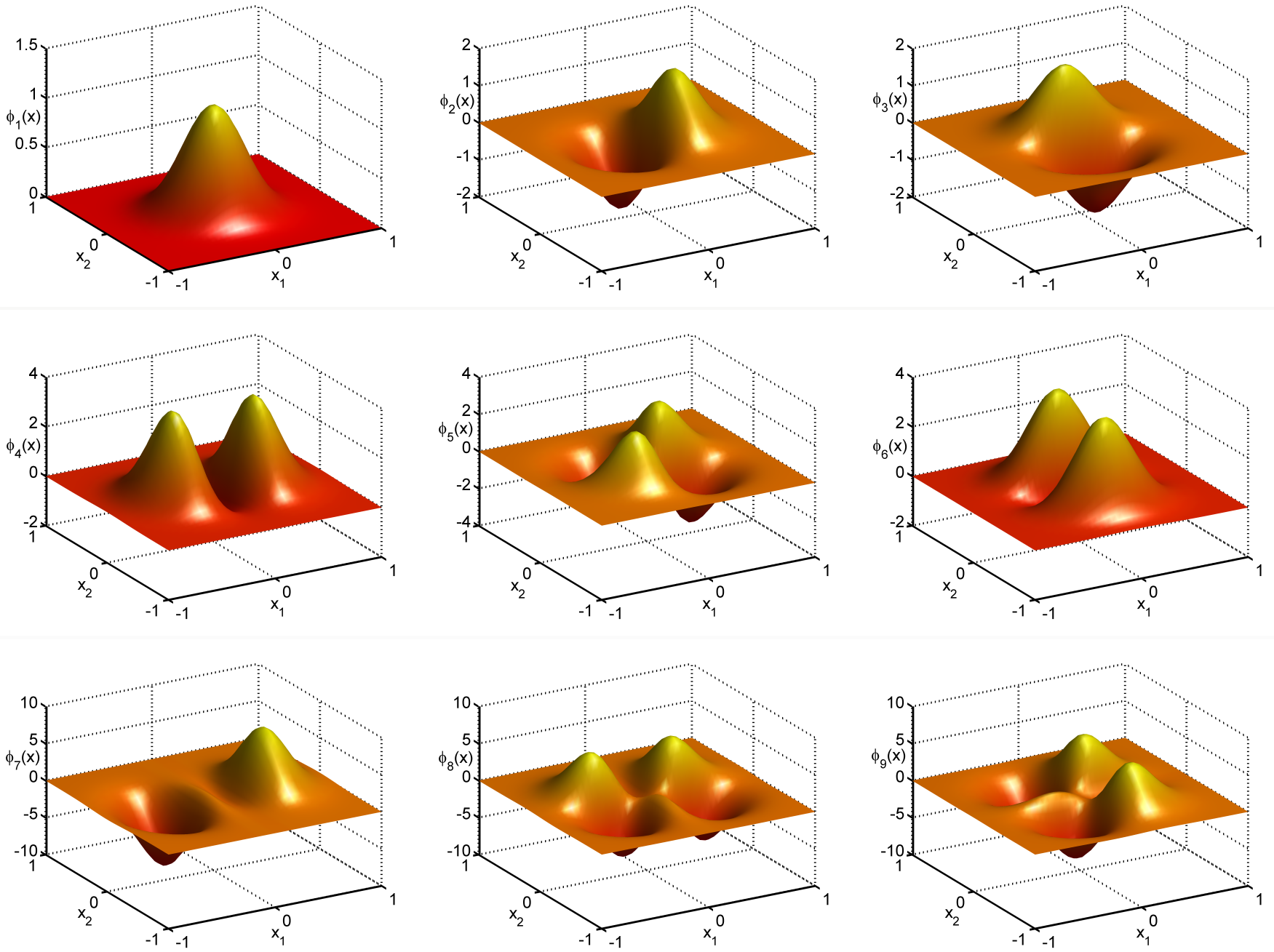
$$= \sum_{\boldsymbol{n}\in\mathbb{N}^d}\lambda_{\boldsymbol{n}}\varphi_{\boldsymbol{n}}(\boldsymbol{x})\varphi_{\boldsymbol{n}}(\boldsymbol{z}), \qquad \boldsymbol{x}=(x_1,\ldots,x_d)\in\mathbb{R}^d,$$

where

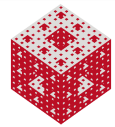
$$\lambda_{\boldsymbol{n}} = \prod_{\ell=1}^{d}\lambda_{n_\ell}, \qquad \varphi_{\boldsymbol{n}}(\boldsymbol{x}) = \prod_{\ell=1}^{d}\varphi_{n_\ell}(x_\ell).$$

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


$\phi_1(x)$
$\phi_2(x)$
$\phi_3(x)$
$\phi_4(x)$
$\phi_5(x)$
$\phi_6(x)$
$\phi_7(x)$
$\phi_8(x)$
$\phi_9(x)$
x_1
x_2


$\phi_1(x)$
$\phi_2(x)$
$\phi_3(x)$
$\phi_4(x)$
$\phi_5(x)$
$\phi_6(x)$
$\phi_7(x)$
$\phi_8(x)$
$\phi_9(x)$
x_1
x_2

In higher dimensions there will be multiple eigenvalues of the same order, and therefore ordering of eigenvalues and their associated eigenfunctions may matter for the performance of the algorithm.



In higher dimensions there will be multiple eigenvalues of the same order, and therefore ordering of eigenvalues and their associated eigenfunctions may matter for the performance of the algorithm.

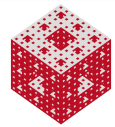
- Using product kernels [FM12] (with uniform ε and α) eigenvalues of the same order follow Pascal's triangle. E.g., in three dimensions the first eigenvalues $\lambda_{\mathbf{1}}, \lambda_{\mathbf{2}}, \lambda_{\mathbf{3}}, \lambda_{\mathbf{4}}$, take the form

$$\lambda_{0,0,0}$$

$$\lambda_{1,0,0}, \lambda_{0,1,0}, \lambda_{0,0,1}$$

$$\lambda_{2,0,0}, \lambda_{1,1,0}, \lambda_{1,0,1}, \lambda_{0,2,0}, \lambda_{0,1,1}, \lambda_{0,0,2}$$

$$\lambda_{3,0,0}, \lambda_{2,1,0}, \lambda_{2,0,1}, \lambda_{1,2,0}, \lambda_{1,1,1}, \lambda_{1,0,2}, \lambda_{0,3,0}, \lambda_{0,2,1}, \lambda_{0,1,2}, \lambda_{0,0,3}$$

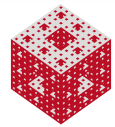


In higher dimensions there will be multiple eigenvalues of the same order, and therefore ordering of eigenvalues and their associated eigenfunctions may matter for the performance of the algorithm.

- Using product kernels [FM12] (with uniform ε and α) eigenvalues of the same order follow Pascal's triangle. E.g., in three dimensions the first eigenvalues $\lambda_{\mathbf{1}}, \lambda_{\mathbf{2}}, \lambda_{\mathbf{3}}, \lambda_{\mathbf{4}}$, take the form

$$\lambda_{0,0,0}$$
$$\lambda_{1,0,0}, \lambda_{0,1,0}, \lambda_{0,0,1}$$
$$\lambda_{2,0,0}, \lambda_{1,1,0}, \lambda_{1,0,1}, \lambda_{0,2,0}, \lambda_{0,1,1}, \lambda_{0,0,2}$$
$$\lambda_{3,0,0}, \lambda_{2,1,0}, \lambda_{2,0,1}, \lambda_{1,2,0}, \lambda_{1,1,1}, \lambda_{1,0,2}, \lambda_{0,3,0}, \lambda_{0,2,1}, \lambda_{0,1,2}, \lambda_{0,0,3}$$

Since we may want to use only the "first", e.g., 12 eigenfunctions we need to decide which two of the order three eigenvalues are most significant.



Fornberg, Larsson and Flyer [FLF11] reported several other eigenvalue patterns for their kernels:

- In 2D for Gaussians, MQs, IMQs, IQs and Bessel kernels with $\beta > d = 2$ we have multiplicities

$$1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ldots$$

- In 2D for Bessel kernels with $\beta = d = 2$ we have multiplicities

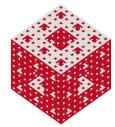
$$1,\ 2,\ 2,\ 2,\ 2,\ 2,\ 2,\ldots$$

- In 3D for Gaussians (see above), MQs, IMQs and IQs we have multiplicities

$$1,\ 3,\ 6,\ 10,\ 15,\ 21,\ 28,\ldots$$

- On the sphere $\mathbb{S}^2$ [FP08] for Gaussians, MQs, IMQs and IQs we have multiplicities

$$1,\ 3,\ 5,\ 7,\ 9,\ 11,\ 13,\ldots$$



References I

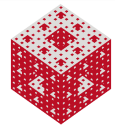
[BLB01] R. K. Beatson, W. A. Light, and S. Billings, *Fast solution of the radial basis function interpolation equations: domain decomposition methods*, SIAM Journal on Scientific Computing **22** (2001), no. 5, 1717–1740.

[CFM14] Roberto Cavoretto, G. E. Fasshauer, and M. J. McCourt, *An introduction to the Hilbert-Schmidt SVD using iterated Brownian bridge kernels*, Numerical Algorithms (2014).

[Fas07] G. E. Fasshauer, *Meshfree Approximation Methods with* MATLAB, Interdisciplinary Mathematical Sciences, vol. 6, World Scientific Publishing Co., Singapore, 2007.

[Fas11a] _____, *Green's functions: taking another look at kernel approximation, radial basis functions and splines*, Approximation Theory XIII: San Antonio 2010 (M. Neamtu and L. L. Schumaker, eds.), Springer Proceedings in Mathematics, vol. 13, Springer, 2011, pp. 37–63.

[Fas11b] _____, *Positive definite kernels: past, present and future*, Dolomites Research Notes on Approximation **4** (2011), 21–63.



References II

[FLF11] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer, *Stable computations with Gaussian radial basis functions*, SIAM Journal on Scientific Computing **33** (2011), no. 2, 869–892.

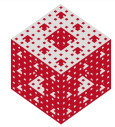
[FM12] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of Gaussian radial basis function interpolants*, SIAM J. Sci. Comput. **34** (2012), no. 2, A737—A762.

[FP08] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM J. Sci. Comput. **30** (2008), no. 1, 60–80.

[FW04] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl. **48** (2004), no. 5–6, 853–867.

[GRZ13] M. Griebel, C. Rieger, and B. Zwicknagl, *Multiscale approximation and reproducing kernel Hilbert space methods*, 2013.

[McC13] M. McCourt, *Building infrastructure for multiphysics simulations*, Ph.D. thesis, Cornell University, 2013.



References III

[Sch81] L. L. Schumaker, *Spline functions: Basic theory*, John Wiley & Sons (New York), 1981, reprinted by Krieger Publishing 1993.

[Tre13] Lloyd N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.

[Wri03] G. B. Wright, *Radial basis function interpolation: Numerical and analytical developments*, Ph.d. thesis, University of Colorado at Boulder, 2003.