What Can the Hilbert–Schmidt SVD Do For Kernel-based Approximation Methods?

Greg Fasshauer

Meshfree Methods Seminar, IIT
May 19, 2014
Kernel-based Interpolation: A Really Quick Refresher

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

\[
s(x) = \sum_{j=1}^{N} c_j K(x, x_j), \quad x \in \Omega \subseteq \mathbb{R}^d
\]

with \(K : \Omega \times \Omega \to \mathbb{R}\) a positive definite reproducing kernel.
Kernel-based Interpolation and Approximation

Kernel-based Interpolation: A Really Quick Refresher

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

\[
s(x) = \sum_{j=1}^{N} c_j K(x, x_j), \quad x \in \Omega \subseteq \mathbb{R}^d
\]

with \(K : \Omega \times \Omega \rightarrow \mathbb{R}\) a positive definite reproducing kernel.

To find \(c_j\) solve the interpolation equations

\[
s(x_i) = y_i, \quad i = 1, \ldots, N,
\]

which leads to a linear system \(Kc = y\) with symmetric positive definite — often ill-conditioned — system matrix

\[
K_{ij} = K(x_i, x_j), \quad i, j = 1, \ldots, N.
\]
Examples of Kernels

- **Radial** (also called isotropic): such as the Gaussian kernel
  \[ K(x, y) = e^{-\epsilon^2 \|x - y\|^2}, \quad x, y \in \mathbb{R}^d \]

- **Anisotropic**, but still translation invariant (stationary): such as the anisotropic Gaussian kernel
  \[
  K(x, y) = e^{-\sum_{\ell=1}^{d} \epsilon_\ell^2 (x_\ell - y_\ell)^2} = \prod_{\ell=1}^{d} e^{-\epsilon_\ell^2 (x_\ell - y_\ell)^2}
  \]
General positive definite kernel: such as the Brownian bridge (or piecewise linear spline) product kernel

$$K(x, y) = \prod_{\ell=1}^{d} \left( \min\{x_\ell, y_\ell\} - x_\ell y_\ell \right), \quad x, y \in [0, 1]^d$$

Note: This kernel satisfies boundary conditions
Scattered Data Fitting with Gaussians

For “nice” functions, Gaussians are spectrally accurate — independent of $d$, allowing for high accuracy with smaller $N$ and avoiding the curse of dimensionality [F./Hickernell/Woźniakowski (2012)]. For applications such as surrogate modeling the cost of producing a data point is significant, so this is useful.
Scattered Data Fitting with Gaussians

For “nice” functions, Gaussians are spectrally accurate — independent of $d$, allowing for high accuracy with smaller $N$ and avoiding the curse of dimensionality [F./Hickernell/Woźniakowski (2012)]. For applications such as surrogate modeling the cost of producing a data point is significant, so this is useful.

The shape parameter $\varepsilon$ can have a great effect on accuracy and condition of the interpolant.

\[ \varepsilon=10, \ k \approx 1 \]
\[ \varepsilon=1, \ k \approx 10^6 \]
\[ \varepsilon=.5, \ k \approx 10^9 \]
Common Complaints About Kernels

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

In this talk I will show that stable and accurate results can be obtained by working with a “better” basis via the Hilbert–Schmidt SVD of the matrix $K$ — without ever forming $K$; a closed form expression for the kernel $K$ is not required; the Hilbert–Schmidt SVD can be implemented in less than 20 lines of MATLAB code.

Free parameters can be “optimally” chosen with the help of statistical methods such as MLE, which are also significantly enhanced by using the HS-SVD (one of last summer’s research topics).

Greg Fasshauer
Common Complaints About Kernels

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

In this talk I will show that
- stable and accurate results can be obtained by working with a “better” basis via the Hilbert–Schmidt SVD of the matrix $K$ — without ever forming $K$;
- a closed form expression for the kernel $K$ is not required;
- the Hilbert–Schmidt SVD can be implemented in less than 20 lines of MATLAB code.
Common Complaints About Kernels

Kernel methods suffer from

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

In this talk I will show that

- stable and accurate results can be obtained by working with a “better” basis via the Hilbert–Schmidt SVD of the matrix $K$ — without ever forming $K$;
- a closed form expression for the kernel $K$ is not required;
- the Hilbert–Schmidt SVD can be implemented in less than 20 lines of MATLAB code.

Free parameters can be “optimally” chosen with the help of statistical methods such as MLE, which are also bluesignificantly enhanced by using the HS-SVD (one of last summer’s research topics).
Uncertainty principle — An unfortunate misconception

“One can’t have high accuracy and stability at the same time.”

[Schaback (1995b), Schaback (1995c)]
Uncertainty principle — An unfortunate misconception

"Using the standard basis, one can’t have high accuracy and stability at the same time.”

[Schaback (1995b), Schaback (1995c)]

Stable evaluation [F./McCourt (2012)] inspired by
[Fornberg/Piret (2008), Fornberg/Larsson/Flyer (2011)]
A Side-trip to the Polynomial World

The most straightforward approach to polynomial interpolation assumes

\[ p_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n, \]
i.e., it uses the basis monomials \( b_j(x) = x^j, j = 0, \ldots, n. \)
A Side-trip to the Polynomial World

The most straightforward approach to polynomial interpolation assumes

\[ p_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n, \]
i.e., it uses the basis monomials \( b_j(x) = x^j, j = 0, \ldots, n. \)

The interpolation conditions \( p_n(x_i) = y_i \) lead to the linear system \( B a = y \) with Vandermonde matrix

\[
B = \begin{bmatrix}
1 & x_0 & x_0^2 & \ldots & x_0^n \\
1 & x_1 & x_1^2 & \ldots & x_1^n \\
1 & x_2 & x_2^2 & \ldots & x_2^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & x_n^2 & \ldots & x_n^n \\
\end{bmatrix}, \quad a = \begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_n \\
\end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
\vdots \\
y_n \\
\end{bmatrix}.
\]
Example

For the data

\[
\begin{array}{c|c|c|c}
 x & 0 & 1 & 3 \\
 y & 1 & 0 & 4 \\
\end{array}
\]

the interpolation conditions lead to the linear system

\[
\begin{align*}
(p_2(0) =) & \quad a_0 = 1 \\
(p_2(1) =) & \quad a_0 + a_1 + a_2 = 0 \\
(p_2(3) =) & \quad a_0 + 3a_1 + 9a_2 = 4 \\
\end{align*}
\]

with solution \( a_0 = 1, \ a_1 = -2, \) and \( a_2 = 1. \) Thus,

\[
p_2(x) = 1 - 2x + x^2.
\]
Example

Note that the polynomial \( p_2(x) = 1 - 2x + x^2 \) can also be written as

\[
p_2(x) = (1 - x)^2.
\]

So, using a **basis of shifted monomials**

\[
b_0(x) = 1, \ b_1(x) = 1 - x, \text{ and } b_2(x) = (1 - x)^2
\]

would (have to) result in expansion coefficients \( a_0 = 0, a_1 = 0, \text{ and } a_2 = 1. \)
Remark

- The monomial basis results in an interpolation matrix (a Vandermonde matrix) that is known to be prone to *ill-conditioning*.
- Therefore one usually uses a different basis to ensure better numerical stability.
- To draw a parallel, we can think of the monomial basis as an analogue to our standard kernel basis \( \{ K(\cdot, x_1), \ldots, K(\cdot, x_N) \} \).
Another, in some sense ideal, basis is provided by the Lagrange form

\[ p_n(x) = \sum_{j=0}^{n} y_j L_j(x), \]  

(1)

where the expansion coefficients are just the data values.
Another, in some sense ideal, basis is provided by the Lagrange form

\[ p_n(x) = \sum_{j=0}^{n} y_j L_j(x), \]  

(1)

where the expansion coefficients are just the data values. The Lagrange or cardinal basis polynomials are given by

\[ L_j(x) = \prod_{\substack{i=0\atop i \neq j}}^{n} \frac{x - x_i}{x_j - x_i}, \quad j = 0, 1, \ldots, n, \]

and have the special cardinality property

\[ L_j(x_i) = \delta_{ij}, \]

which ensures that the interpolation conditions are satisfied.
Example

Using the same data as before, i.e.,

\[
\begin{array}{c|ccc}
    x & 0 & 1 & 3 \\
    y & 1 & 0 & 4 \\
\end{array}
\]

the Lagrange form of the interpolating polynomial is

\[
p_2(x) = \sum_{j=0}^{2} y_j L_j(x) = L_0(x) + 4L_2(x), \quad \text{with } L_j(x) = \prod_{\substack{i=0 \atop i \neq j}}^{2} \frac{x - x_i}{x_j - x_i}.
\]

We can compute\(^a\)

\[
L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 1)(x - 3)}{(-1)(-3)} = \frac{1}{3}(x - 1)(x - 3),
\]

and

\[
L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{x(x - 1)}{(3 - 0)(3 - 1)} = \frac{1}{6}x(x - 1).
\]

\(^a\)We do not need the third basis function \(L_1(x) = -\frac{1}{2}x(x - 3)\).
Remark

- The interpolation matrix for the Lagrange form is the identity matrix and the coefficients in the basis expansion are given by the data values.

- This makes the Lagrange form ideal for situations in which many experiments with the same data sites, but different data values need to be performed.

- The Lagrange form is often used for theoretical purposes such as computation of Lebesgue constants and determination of error estimates.
Additional Bases

- Chebyshev polynomials, defined recursively as
  \[ T_k = 2xT_{k-1} - T_{k-2}, \quad T_1 = x, \quad T_0 = 1 \]
  are orthogonal on \((-1, 1)\) with respect to \(\rho(x) = \frac{1}{\sqrt{1-x^2}}\).

- Legendre polynomials, defined recursively as
  \[ P_k = \frac{2k-1}{k}xP_{k-1} - \frac{k-1}{k}P_{k-2}, \quad P_1 = x, \quad P_0 = 1 \]
  are orthogonal on \([-1, 1]\) with respect to \(\rho(x) \equiv 1\).

- Hermite polynomials, defined recursively as
  \[ H_k = 2xH_{k-1} - 2(k-2)H_{k-2}, \quad H_1 = x, \quad H_0 = 1 \]
  are orthogonal on \((-\infty, \infty)\) with respect to \(\rho(x) = e^{-x^2/2}\).
Example

We perform a series of interpolation experiments with an eye on error and condition number.

Data is obtained by sampling the function $f(x) = \cos(10x + \frac{1}{2})$ at an increasing number $N$ of evenly spaced points in $[-1, 1]$. 
Figure: Four different bases for polynomials of degree 5 on $[-1, 1]$. 
**Figure:** Errors and condition numbers for polynomial interpolation.
We illustrate the effects of the locations at which we sample the data function (or the *design*) for polynomial interpolation.
Figure: Errors and condition numbers for polynomial interpolation at evenly spaced points in $[-1, 1]$. 
Figure: Errors and condition numbers for polynomial interpolation on Halton points in $[-1, 1]$. 
Figure: Errors and condition numbers for polynomial interpolation on Chebyshev points in $[-1, 1]$. 
Figure: Errors and condition numbers for polynomial interpolation on $[-1, 1]$ at points clustered around the origin.
Hilbert–Schmidt Theory

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

$$K(x, z) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(x) \varphi_n(z), \quad x, z \in \Omega \subseteq \mathbb{R}^d,$$

where $(\lambda_n, \varphi_n)$ are orthonormal eigenpairs of a Hilbert–Schmidt integral operator $\mathcal{K} : L_2(\Omega, \rho) \rightarrow L_2(\Omega, \rho)$ defined as

$$(\mathcal{K}f)(x) = \int_{\Omega} K(x, z)f(z)\rho(z)dz,$$

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

$$\mathcal{K}\varphi_n = \lambda_n \varphi_n \iff \int_{\Omega} K(x, z)\varphi_n(z)\rho(z)dz = \lambda_n \varphi_n(x), \quad n = 1, 2, \ldots$$
Simple Example: Brownian Bridge

The Brownian bridge kernel is given by

\[ K(x, z) = \min(x, z) - xz = \begin{cases} 
  x - xz = x(1 - z), & 0 \leq x \leq z \leq 1, \\
  z - xz = z(1 - x), & 0 \leq z \leq x \leq 1,
\end{cases} \]

and one can verify that the eigenvalues and normalized eigenfunctions

\[ \lambda_n = (n\pi)^{-2}, \quad \varphi_n(x) = \sqrt{2} \sin n\pi x, \quad n = 1, 2, \ldots, \]

satisfy \( K \varphi_n = \lambda_n \varphi_n \) (with \( \rho(z) \equiv 1 \)), i.e.,

\[ \int_0^1 (\min(x, z) - xz) \sin(n\pi z) dz = (n\pi)^{-2} \sin(n\pi x), \quad n = 1, 2, \ldots. \]
Simple Example: Brownian Bridge

The Brownian bridge kernel is given by

$$K(x, z) = \min(x, z) - xz = \begin{cases} 
  x - xz = x(1 - z), & 0 \leq x \leq z \leq 1, \\
  z - xz = z(1 - x), & 0 \leq z \leq x \leq 1,
\end{cases}$$

and one can verify that the eigenvalues and normalized eigenfunctions

$$\lambda_n = (n\pi)^{-2}, \quad \varphi_n(x) = \sqrt{2} \sin n\pi x, \quad n = 1, 2, \ldots,$$

satisfy $$K\varphi_n = \lambda_n\varphi_n$$ (with $$\rho(z) \equiv 1$$), i.e.,

$$\int_0^1 (\min(x, z) - xz) \sin(n\pi z) dz = (n\pi)^{-2} \sin(n\pi x), \quad n = 1, 2, \ldots.$$

The Hilbert-Schmidt expansion is a generalized Fourier series

$$K(x, z) = \min(x, z) - xz = \sum_{n=1}^{\infty} \frac{2}{n^2 \pi^2} \sin n\pi x \sin n\pi z.$$
Gaussian Eigenfunctions

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

\[ e^{-\varepsilon^2(x-z)^2} = \sum_{n=0}^{\infty} \lambda_n \varphi_n(x) \varphi_n(z), \quad x, z \in \mathbb{R} \]
Gaussian Eigenfunctions

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

\[ e^{-\varepsilon^2(x-z)^2} = \sum_{n=0}^{\infty} \lambda_n \varphi_n(x) \varphi_n(z), \quad x, z \in \mathbb{R} \]

where

\[ \lambda_n = \frac{\alpha \varepsilon^{2n}}{(\alpha^2 + \delta^2 + \varepsilon^2)^{n+1/2}}, \quad \varphi_n(x) = \gamma_n e^{-\delta^2 x^2} H_n(\alpha \beta x) \]

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

\[ \int_{-\infty}^{\infty} \varphi_m(x) \varphi_n(x) \rho(x) \, dx = \delta_{mn}, \quad \rho(x) = \frac{\alpha}{\sqrt{\pi}} e^{-\alpha^2 x^2} \]
Multivariate Eigenfunction Expansion

Use tensor product form of the Gaussian kernel

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

$$x = (x_1, \ldots, x_d), z \in \mathbb{R}^d,$$
Multivariate Eigenfunction Expansion

Use tensor product form of the Gaussian kernel

\[ K(x, z) = e^{-\epsilon^2 \|x-z\|^2} = e^{-\sum_{\ell=1}^{d} \epsilon_{\ell}^2 (x_\ell - z_\ell)^2} = \prod_{\ell=1}^{d} e^{-\epsilon_{\ell}^2 (x_\ell - z_\ell)^2} \]

\[ x = (x_1, \ldots, x_d), \quad z \in \mathbb{R}^d, \]
Multivariate Eigenfunction Expansion

Use tensor product form of the Gaussian kernel

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

\[ = \sum_{n \in \mathbb{N}^d} \lambda_n \varphi_n(x) \varphi_n(z), \quad x = (x_1, \ldots, x_d), \quad z \in \mathbb{R}^d, \]

where

\[ \lambda_n = \prod_{\ell=1}^{d} \lambda_{n_{\ell}}, \quad \varphi_n(x) = \prod_{\ell=1}^{d} \varphi_{n_{\ell}}(x_{\ell}). \]

Different shape parameters \( \varepsilon_{\ell} \) for different space dimensions allowed (i.e., \( K \) may be anisotropic).
Fundamental idea: Formally use the eigen-expansion of the kernel $K$ to rewrite the matrix $K$ from the interpolation problem as

$$
K = \begin{pmatrix}
K(x_1, x_1) & \cdots & K(x_1, x_N) \\
\vdots & \ddots & \vdots \\
K(x_N, x_1) & \cdots & K(x_N, x_N)
\end{pmatrix}
$$

$$
= \begin{pmatrix}
\varphi_1(x_1) & \cdots & \varphi_M(x_1) \\
\vdots & \ddots & \vdots \\
\varphi_1(x_N) & \cdots & \varphi_M(x_N)
\end{pmatrix}
\begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_M
\end{pmatrix}
\begin{pmatrix}
\varphi_1(x_1) & \cdots & \varphi_1(x_N) \\
\vdots & \ddots & \vdots \\
\varphi_M(x_1) & \cdots & \varphi_M(x_N)
\end{pmatrix}
$$

Since $K(x_i, x_j) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(x_i) \varphi_n(x_j) \approx \sum_{n=1}^{M} \lambda_n \varphi_n(x_i) \varphi_n(x_j)$, accurate reconstruction of all entries of $K$ will likely require $M > N$. 
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

$$K = \begin{pmatrix}
K(x_1, x_1) & \ldots & K(x_1, x_N) \\
\vdots & \ddots & \vdots \\
K(x_N, x_1) & \ldots & K(x_N, x_N)
\end{pmatrix} = \begin{pmatrix}
\varphi_1(x_1) & \ldots & \varphi_M(x_1) \\
\vdots & \ddots & \vdots \\
\varphi_1(x_N) & \ldots & \varphi_M(x_N)
\end{pmatrix}
\begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_M
\end{pmatrix} = \Phi \Lambda \Phi^T$$

Since $K(x_i, x_j) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(x_i) \varphi_n(x_j) \approx \sum_{n=1}^{M} \lambda_n \varphi_n(x_i) \varphi_n(x_j)$, accurate reconstruction of all entries of $K$ will likely require $M > N$. 
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

\[
K = \begin{pmatrix}
K(x_1, x_1) & \ldots & K(x_1, x_N) \\
\vdots & & \vdots \\
K(x_N, x_1) & \ldots & K(x_N, x_N)
\end{pmatrix}
\]

\[
= \begin{pmatrix}
\varphi_1(x_1) & \ldots & \varphi_M(x_1) \\
\vdots & & \vdots \\
\varphi_1(x_N) & \ldots & \varphi_M(x_N)
\end{pmatrix}

\begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_M
\end{pmatrix}

= \Phi \Lambda \Phi^T
\]

Since

\[
K(x_i, x_j) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(x_i) \varphi_n(x_j) \approx \sum_{n=1}^{M} \lambda_n \varphi_n(x_i) \varphi_n(x_j)
\]

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

Greg Fasshauer

Hilbert–Schmidt SVD
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 $\Lambda$. 
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 $\Lambda$.

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 $\Phi$ is “short and fat” and partition $\Phi$:

$$
\begin{pmatrix}
\varphi_1(x_1) & \ldots & \varphi_N(x_1) & \varphi_{N+1}(x_1) & \ldots & \varphi_M(x_1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\varphi_1(x_N) & \ldots & \varphi_N(x_N) & \varphi_{N+1}(x_N) & \ldots & \varphi_M(x_N)
\end{pmatrix}
= \begin{pmatrix}
\Phi_1_{N \times N} & \Phi_2_{N \times (M-N)}
\end{pmatrix}
$$
Details of the Hilbert–Schmidt SVD

Assume $M > N$, so that $\Phi$ is “short and fat” and partition $\Phi$:

\[
\begin{pmatrix}
\varphi_1(x_1) & \ldots & \varphi_N(x_1) & \varphi_{N+1}(x_1) & \ldots & \varphi_M(x_1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\varphi_1(x_N) & \ldots & \varphi_N(x_N) & \varphi_{N+1}(x_N) & \ldots & \varphi_M(x_N)
\end{pmatrix}
= \begin{pmatrix}
\Phi_1 & \Phi_2 \\
N \times N & N \times (M-N)
\end{pmatrix}
\]

Then

\[K = \Phi \Lambda \Phi^T\]
Details of the Hilbert–Schmidt SVD

Assume $M > N$, so that $\Phi$ is “short and fat” and partition $\Phi$:

$$
\begin{pmatrix}
\varphi_1(x_1) & \ldots & \varphi_N(x_1) & \varphi_{N+1}(x_1) & \ldots & \varphi_M(x_1) \\
\vdots & & \vdots & \vdots & & \vdots \\
\varphi_1(x_N) & \ldots & \varphi_N(x_N) & \varphi_{N+1}(x_N) & \ldots & \varphi_M(x_N)
\end{pmatrix}
= 
\begin{pmatrix}
\Phi_1 & \Phi_2 \\
N \times N & N \times (M-N)
\end{pmatrix}
$$

Then

$$
K = \Phi \Lambda \Phi^T
= \Phi \begin{pmatrix}
\Lambda_1 & \\
& \Lambda_2
\end{pmatrix}
\begin{pmatrix}
\Phi_1^T \\
\Phi_2^T
\end{pmatrix}
$$
Details of the Hilbert–Schmidt SVD

Assume $M > N$, so that $\Phi$ is “short and fat” and partition $\Phi$:

$$
\begin{pmatrix}
\varphi_1(x_1) & \cdots & \varphi_N(x_1) & \varphi_{N+1}(x_1) & \cdots & \varphi_M(x_1) \\
\vdots & & \vdots & \vdots & \ddots & \vdots \\
\varphi_1(x_N) & \cdots & \varphi_N(x_N) & \varphi_{N+1}(x_N) & \cdots & \varphi_M(x_N)
\end{pmatrix}
= 
\begin{pmatrix}
\Phi_1 & \Phi_2 \\
N \times N & N \times (M-N)
\end{pmatrix}.
$$

Then

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

= \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}
I_N \\
\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}
\end{pmatrix}
\begin{pmatrix}
\Lambda_1 \Phi_1^T \\
= M
\end{pmatrix} = \Psi
$$
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”. [Preconditioning]
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.,
  - \( \Lambda_1 \) **is a diagonal matrix of** Hilbert–Schmidt singular values,
  - \( \Psi \) and \( \Phi_1 \) **are matrices generated by orthogonal eigenfunctions** (but not orthogonal matrices).

---

**Preconditioning**

**Summary**

We solve the interpolation problem via the well-conditioned linear system

\[ \Psi b = y. \]

The matrix \( \Psi \) can be computed stably as

\[ \Psi = \Phi_1 \left( I_N \Lambda_2 \Phi_2^T \Phi_2^{-T} \Lambda_1^{-1} \right). \]

Most importantly, the HS-SVD can be computed directly from the eigenvalues and eigenfunctions — without knowing \( K \).
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 \( K \) \( \iff \) “better basis”. [Preconditioning]
- We’ve diagonalized the matrix \( K \), i.e.,
  - \( \Lambda_1 \) is a diagonal matrix of Hilbert–Schmidt singular values,
  - \( \psi \) and \( \Phi_1 \) are matrices generated by orthogonal eigenfunctions (but not orthogonal matrices).

Summary

- We solve the interpolation problem via the well-conditioned linear system \( \psi b = y \).
- The matrix \( \psi \) can be computed stably as \( \psi = \Phi \left( \begin{bmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{bmatrix} \right) \).
- Most importantly, the HS-SVD can be computed directly from the eigenvalues and eigenfunctions — without knowing \( K \).
Taking a closer look at the matrix $\Psi$, we see that
\[
\Psi = (\Phi_1 \Phi_2) \left( \begin{array}{c} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{array} \right)
\]
\[
= \Phi_1 + \Phi_2 \left[ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \right].
\]

We can interpret this as having a new basis $\psi(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$ for the interpolation space $\text{span} \{ K(\cdot, x_1), \ldots, K(\cdot, x_N) \}$ consisting of the appropriately corrected first $N$ eigenfunctions.
Taking a closer look at the matrix $\Psi$, we see that

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

We can interpret this as having a new basis $\psi(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$ for the interpolation space $\text{span} \{K(\cdot, x_1), \ldots, K(\cdot, x_N)\}$ 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

$$
k(x)^T = \phi(x)^T \left( \begin{array}{c} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{array} \right) \Lambda_1 \Phi_1^T = \psi(x)^T \Lambda_1 \Phi_1^T.
$$
Taking a closer look at the matrix $\Psi$, we see that

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

We can interpret this as having a new basis $\psi(\cdot)^T = (\psi_1(\cdot), \ldots, \psi_N(\cdot))$ for the interpolation space span $\{K(\cdot, x_1), \ldots, K(\cdot, x_N)\}$ 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

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

The data-dependence of the new basis is captured by the “correction” term. The new basis is more stable since we have removed $\Lambda_1$. 
The QR in RBF-QR

Additional stability in the computation of the correction matrix

\[
\begin{bmatrix}
\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}
\end{bmatrix},
\]

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

\[
(\Phi_1 \ \Phi_2) = Q \begin{pmatrix}
R_1 \\
N \times N
\end{pmatrix} \begin{pmatrix}
R_2 \\
N \times (M-N)
\end{pmatrix}
\]

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

Greg Fasshauer
Hilbert–Schmidt SVD
The QR in RBF-QR

Additional stability in the computation of the correction matrix

\[
\begin{bmatrix}
\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}
\end{bmatrix},
\]

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

\[
( \Phi_1 \quad \Phi_2 ) = Q \begin{pmatrix}
R_1 & R_2 \\
N \times N & N \times (M-N)
\end{pmatrix}
\]

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

\[
\Phi_2^T \Phi_1^{-T} = R_2^T Q^T QR_1^{-T} = R_2^T R_1^{-T}.
\]

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

Instead of solving the “original” interpolation problem with ill-conditioned matrix $K$

$$Kc = y,$$

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

$$\Psi b = y$$

for a new set of coefficients which we then evaluate via

$$s(x) = k(x)^T K^{-1} y$$

$$= \psi(x)^T \Lambda_1 \Phi_1^T \Phi_1^{-T} \Lambda_1^{-1} \psi^{-1} y$$

$$= \psi(x)^T \psi^{-1} y.$$

i.e., using the new basis, and all the ill-conditioning from $\Lambda_1$ is gone.
General Implementation

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

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

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

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

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 $K$:

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

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

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

[Cavoretto/F./McCourt (2014)]
We define **iterated Brownian bridge kernels** as Green’s kernels of

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

subject to

\[
\frac{d^{2\nu}}{dx^{2\nu}} K(0, z) = \frac{d^{2\nu}}{dx^{2\nu}} K(1, z) = 0, \quad \nu = 0, \ldots, \beta - 1.
\]
We define iterated Brownian bridge kernels as Green’s kernels of
\[
\left( -\frac{d^2}{dx^2} + \varepsilon^2 I \right)^\beta K(x, z) = \delta(x - z), \quad x, z \in [0, 1], \beta \in \mathbb{N}, \varepsilon \geq 0,
\]
subject to
\[
\frac{d^{2\nu}}{dx^{2\nu}} K(0, z) = \frac{d^{2\nu}}{dx^{2\nu}} K(1, z) = 0, \quad \nu = 0, \ldots, \beta - 1.
\]

The Hilbert–Schmidt expansion for these kernels is
\[
K_{\beta,\varepsilon}(x, z) = \sum_{n=1}^{\infty} \frac{2}{(n^2 \pi^2 + \varepsilon^2)^\beta} \sin(n\pi x) \sin(n\pi z),
\]
i.e., the eigenvalues and eigenfunctions are
\[
\lambda_n = \frac{1}{(n^2 \pi^2 + \varepsilon^2)^\beta}, \quad \varphi_n(x) = \sqrt{2} \sin(n\pi x). \tag{2}
\]
Two-Parameter Iterated Brownian Bridge Kernels

(a) $\beta = 1, \varepsilon = 0$

(b) $\beta = 1, \varepsilon = 10$
Two-Parameter Iterated Brownian Bridge Kernels

(c) $\beta = 2, \varepsilon = 0$

(d) $\beta = 2, \varepsilon = 10$
Two-Parameter Iterated Brownian Bridge Kernels

(e) $\beta = 3, \varepsilon = 0$

(f) $\beta = 3, \varepsilon = 10$
Two-Parameter Iterated Brownian Bridge Kernels

(g) $\beta = 7$, $\varepsilon = 0$

(h) $\beta = 7$, $\varepsilon = 10$
Two-Parameter Iterated Brownian Bridge Kernels

(i) $\beta = 20, \varepsilon = 0$

(j) $\beta = 20, \varepsilon = 50$
Clearly,

- the eigenfunctions are bounded by $\sqrt{2}$,
- and, for a fixed value of $\varepsilon$, 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 $\beta$ and $\varepsilon$:
Clearly,

- the eigenfunctions are bounded by $\sqrt{2}$,
- and, for a fixed value of $\varepsilon$, 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 $\beta$ and $\varepsilon$:

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

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

- the eigenfunctions are bounded by $\sqrt{2}$,
- and, for a fixed value of $\varepsilon$, 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 $\beta$ and $\varepsilon$:

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

$$\lambda_M < \varepsilon_{mach}\lambda_N, \quad M > N.$$ 

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

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

function yy = HSSVDInterp(x,y,ep,beta,xx)
    phifunc = @(n,x) sqrt(2)*sin(pi*x*n);
    lambdafunc = @(n) ((n*pi).^2+ep^2).^(−beta);
    N = length(x);
    M = ceil(1/pi*sqrt(eps^(-1/beta)*(N^2*pi^2+ep^2)-ep^2));
    I_N = eye(N);
    Lambda = diag(lambdafunc(1:M));
    Phi_interp = phifunc(1:M,x);
    Phi_eval = phifunc(1:M,xx);
    Phi_1 = Phi_interp(:,1:N);
    Phi_2 = Phi_interp(:,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_interp = Phi_interp*[I_N;Correction];
    Psi_eval = Phi_eval*[I_N;Correction];
    yy = Psi_eval*(Psi_interp\y);
end
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: $\text{cond}(K) = 3.4 \times 10^{17}$
We use
- Gaussians
- \( N \) quasi-random Halton samples of
\[
f(u, v, w, x, y) = 1 + (u + v + w)^2(x - y)^2(u + x) \text{ in } [-1, 1]^5
\]
Standard RBF vs. HS-SVD Interpolation

We use
- Gaussians
- $N$ quasi-random Halton samples of
  $$f(u, v, w, x, y) = 1 + (u + v + w)^2(x - y)^2(u + x)$$
in $[-1, 1]^5$
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
  - can use series to create designer kernels

- Vast applications:
  - function interpolation/approximation
  - numerical solution of PDEs (collocation, MFS, MPS)
  - parameter estimation (MLE, GCV)
  - ...

- Future outlook:
  - implement for anisotropic Gaussians
  - HS-SVD for other kernels
  - other applications
References I

Rasmussen, C. E. and Williams, C. K. I.
*Gaussian Processes for Machine Learning.*

Cavoretto, R., Fasshauer, G. E. and McCourt, M. J.
An introduction to the Hilbert–Schmidt SVD using iterated Brownian bridge kernels.
Numerical Algorithms, DOI: 10.1007/s11075-014-9850-z.

Fasshauer, G. E., Hickernell, F. J. and Woźniakowski, H.
Rate of convergence and tractability of the radial function approximation problem.

Fasshauer, G. E. and McCourt, M. J.
Stable evaluation of Gaussian RBF interpolants.

Fasshauer, G. E. and McCourt, M. J.
Approximating derivatives stably using Gaussians.
Submitted.
References II

1. Fornberg, B., Larsson, E. and Flyer, N.
   Stable computations with Gaussian radial basis functions.

2. Fornberg, B. and Piret, C.
   A stable algorithm for flat radial basis functions on a sphere.

   Radial basis function methods in computational finance.

4. McCourt, M. J.
   Using Gaussian eigenfunctions to solve boundary value problems.
References III

Miele, A. and Iyer, R. R.
General technique for solving nonlinear, two-point boundary-value problems via the method of particular solutions.

Schaback, R.
Error estimates and condition numbers for radial basis function interpolation.

Schaback, R.
Multivariate interpolation and approximation by translates of a basis function.
Left and Right Preconditioning for $Kc = y$

Let $P = M^{-1}$ be a preconditioning matrix so that

$$\text{cond}(PK) \ll \text{cond}(K) \quad (\text{"left"}) \quad \text{or} \quad \text{cond}(KP) \ll \text{cond}(K) \quad (\text{"right"})$$
Left and Right Preconditioning for $Kc = y$

Let $P = M^{-1}$ be a preconditioning matrix so that

$$\text{cond}(PK) \ll \text{cond}(K) \quad (\text{"left"}) \quad \text{or} \quad \text{cond}(KP) \ll \text{cond}(K) \quad (\text{"right"})$$

- Left preconditioning: change both sides, i.e.,

  $$(PK)c = Py \iff c = (PK)^{-1}Py$$

- Right preconditioning: change only the left-hand-side, i.e.,

  $$(KP)b = y \iff b = (KP)^{-1}y$$

This would be the version that matches our approach. But we never form $K$ — we directly create $\Psi = KP$ from the eigenvalues and eigenfunctions!
Left and Right Preconditioning for \( Kc = y \)

Let \( P = M^{-1} \) be a preconditioning matrix so that

\[
\text{cond}(PK) \ll \text{cond}(K) \quad \text{("left")} \quad \text{or} \quad \text{cond}(KP) \ll \text{cond}(K) \quad \text{("right")}
\]

- **Left preconditioning**: change both sides, i.e.,

  \[
  (PK)c = Py \iff c = (PK)^{-1}Py
  \]

- **Right preconditioning**: change only the left-hand-side, i.e.,

  \[
  (KP)b = y \iff b = (KP)^{-1}y
  \]

This would be the version that matches our approach. But we never form \( K \)— we directly create \( \Psi = KP \) from the eigenvalues and eigenfunctions!