ORIE 6741 Final Report: Accurate Kernel Interpolation with Compactly Supported Kernels

Rohit Bandaru RB696@CORNELL.EDU
Cornell University, Gates Hall, Cornell Unviersity, Ithaca, NY 14850 USA

Ian Delbridge IAD35@CORNELL.EDU
Cornell University, Gates Hall, Cornell Unviersity, Ithaca, NY 14850 USA

Avinash Thangali AT553@CORNELL.EDU
Cornell University, Gates Hall, Cornell Unviersity, Ithaca, NY 14850 USA

Abstract

Current methods for structured kernel interpolation (SKI) of Gaussian processes (GPs) use either simple local cubic interpolation or computationally expensive GP interpolation. In this project, we investigate scalable kernel interpolation using compactly-supported kernels (CSKs) to increase the accuracy of SKI while maintaining tractability. Kernel interpolation with CSK GP regression is theoretically favorable because it is a tractable, expressive, and tunable interpolation method. In this paper, we show how GP regression with CSKs can be used to perform scalable kernel interpolation. Experiments demonstrate that for many kernels this method can achieve higher interpolation accuracy per unit time than state-of-the-art kernel approximation, KISS-GP. We also show that CSK hyperparameters can be automatically tuned via marginal likelihood maximization. However, in several experiments it is shown that this method does not achieve significantly higher regression accuracy compared to KISS-GP.

1. Introduction

Gaussian processes (GPs) are valuable Bayesian nonparametric probabilistic models due to their flexibility, being universal approximators (Rasmussen & Williams, 2006), and due to their ability to encode strong inductive biases via the covariance kernel. For these reasons, they have been used successfully in many applications, especially those requiring predictive uncertainty estimates. However, the naive computational complexity of GP learning and inference has historically prevented GP usage on data sets with a large number of examples. Denote the number of training examples as n. The traditional method for calculating the predictive distribution and the marginal likelihood of a GP involves calculating the Cholesky decomposition of the $n \times n$ covariance matrix, which requires $\mathcal{O}(n^3)$ computations (Rasmussen & Williams, 2006). Both exact and approximate scalable Gaussian process methods have been developed to reduce this running time.

In particular, inducing point methods approximate Gaussian processes' kernels by evaluating the kernel on a set of chosen "inducing" points rather than on the entire dataset. Scalable kernel interpolation[1] (SKI) is an inducing point method that approximates the kernel by interpolating the kernel at data points using inducing points, which are chosen with structure to permit fast linear algebra. Separately, compactly supported kernels have been derived as replacements for standard kernels that reduce the computational complexity of GP inference by inducing a sparse covariance matrix.

In this work, we build upon previous inducing point methods by extending the SKI framework to interpolation by zero-noise Gaussian processes with compactly supported kernels, and refer to the method as CSK-SKI. Finding simultaneously fast and accurate kernel interpolants is integral to the success of the SKI framework because faster interpolation results in more scalable SKI GP inference, and more accurate interpolation requires fewer inducing points,

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

[1] While (Wilson & Nickisch, 2015) use SKI and KISS-GP interchangeably, we shall refer to the framework as SKI and to SKI using local cubic interpolation as KISS-GP to make a distinction between the particular choice of interpolation and the framework itself.

which again results in more scalable SKI GP inference. In this work, we implement efficient, arbitrary-dimensional CSK-SKI with NumPy. To study CSK-SKI, we perform the following experiments: (1) we analyze the asymptotic run-time of CSK-SKI; (2) we measure the kernel interpolation accuracy and running time of CSK-SKI varying tuning parameters, CSKs, and base kernels; (2) we measure the kernel interpolation accuracy and running time of CSK-SKI in increasingly high dimensions; and (3) we measure the regression accuracy with both synthetic data and noisy real-world data. Additionally, we experiment with informative prior means for the interpolating GP. Finally, we demonstrate how CSK hyperparameters can be automatically tuned with constraints on computational cost.

To the authors knowledge, this is the first application of CSKs to kernel interpolation and the first detailed study comparing the *de facto* local polynomial interpolation to alternative interpolation methods in the context of kernel interpolation. To the authors knowledge, it is also the first work to use automatically tunable kernel interpolation or to perform compactly supported kernel learning.

2. Background & Related Work

2.1. Gaussian Process Inference & Learning

For a given dataset of n input vectors $X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ and n scalar responses $\mathbf{y} = (\mathbf{y}(\mathbf{x_1}), \ldots, \mathbf{y}(\mathbf{x_n}))^\top$, a Gaussian process (GP) is a collection of random variables for whom any finite subset has joint Gaussian distribution. We use GPs to define distributions over functions $f(\mathbf{x}) \sim \mathcal{GP}(\mu, \mathbf{k})$ where any collection of function values $\mathbf{f}$ has joint Gaussian distribution:

$$\mathbf{f} = \mathbf{f}(\mathbf{X}) \sim \mathcal{N}(\mu, \mathbf{K}) \tag{1}$$

with mean vector $\mu_\mathbf{i} = \mu(\mathbf{x_i})$ and $n \times n$ covariance matrix K. The covariance kernel of a GP, as well as its hyperparameters $\boldsymbol{\theta}$, controls the GP's smoothness and generalization properties. The predictive distribution of a GP modeled with white observation noise on n_* test points X_* is given by:

$$\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}, \theta, \sigma^\mathbf{2} \sim \mathcal{N}(\overline{\mathbf{f}_*}, \mathbf{cov}(\mathbf{f}_*)) \tag{2}$$

$$\overline{\mathbf{f}_*} = \mu_{\mathbf{X}_*} + \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^\mathbf{2}\mathbf{I}]^{-\mathbf{1}}(\mathbf{y} - \mu_\mathbf{X}) \tag{3}$$

$$\text{cov}(\mathbf{f}_*) = \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^\mathbf{2}\mathbf{I}]^{-\mathbf{1}}\mathbf{K}_{\mathbf{X},\mathbf{X}_*} \tag{4}$$

$K_{X_*,X}$ denotes the cross-covariance matrix between X and $X_\star$, and μ_{X_*} is the mean vector for the test points. To obtain the marginal likelihood of the data, we can marginalize $f(\mathbf{x})$ conditioned on θ:

$$\log p(\mathbf{y}|\theta) \propto -[\mathbf{y}^\top(\mathbf{K}_\theta + \sigma^\mathbf{2}\mathbf{I})^{-\mathbf{1}}\mathbf{y} + \log|\mathbf{K}_\theta + \sigma^\mathbf{2}\mathbf{I}|] \tag{5}$$

To perform inference and learning, i.e. calculate the predictive distribution and marginal likelihood, we must compute $(K + \sigma^2 I)^{-1}\mathbf{y}$ and $\log|K + \sigma^2 I|$ respectively. This is the computational bottleneck for GP inference and learning, and the source of inefficiency on large datasets.

2.2. Scalable GP Methods

2.2.1. Inducing Point Methods

In response to the intractability of GP learning and inference on large datasets, two major classes of methods were developed to circumvent such limitations. *Inducing-point methods* seek to reduce the computational complexity by approximating the GP with m (usually $m \ll n$) "inducing" points instead of the n training points directly (Quiñonero-Candela & Rasmussen, 2005). Inducing-point methods such as subset of regressors (SoR) (Silverman, 1985) and fully-independent training conditional (FITC) (Snelson & Ghahramani, 2006) that do not exploit training data structure unfortunately still require $\mathcal{O}(nm^2)$ computations. Therefore, to be tractable, the number of inducing points must be small, prohibiting the accuracy of the approximations.

2.2.2. Structure-exploiting Methods

Structure-exploiting methods make use of structure in the covariance matrix, such as Kronecker, Toeplitz, or Hadamard-product structure, for fast matrix operations (Saatçi, 2012). However, these methods are restrictive to data that exhibit gridded structure.

2.2.3. Scalable Kernel Interpolation

The SKI framework, introduced in (Wilson & Nickisch, 2015), combines inducing point methods with structure exploitation.

SKI builds upon a simple approach to make GP inference and learning scalable; an inducing point method called *Subset of Regressors (SoR)* (Silverman, 1985). Given a *base* kernel function $k(\mathbf{x}, \mathbf{z})$ to approximate, and a set of m inducing points $U = [\mathbf{u_i}]_{\mathbf{i=1\ldots m}}$, the SoR kernel approximation is :

$$K_{\text{SoR}} \triangleq K_{X,U}K_{U,U}^{-1}K_{U,X}, \tag{6}$$

where $K_{X,U}, K_{U,U}, K_{U,X}$ are the $n \times m, m \times m, m \times n$ (cross-)covariance matrices generated from the base kernel k. In the SKI framework, the inducing points are taken to be on a grid, which allows for fast matrix operations with $K_{U,U}$ by structure-exploiting methods. However, the cross covariance matrix $K_{X,U}$ has no such structure, so matrix computations with it are slow. Therefore, the authors of (Wilson & Nickisch, 2015) interpolate the cross-covariance

with a matrix of interpolation weights W:

$$K_{X,U} \approx W K_{U,U} \tag{7}$$

The SKI kernel approximation is then derived by replacing the cross covariance matrix in equation (6) with its interpolation, yielding:

$$K_{X,X} \approx K_{X,U} K_{U,U}^{-1} K_{U,X} \tag{8}$$

$$\approx W K_{U,U} K_{U,U}^{-1} K_{U,U} W^\top \tag{9}$$

$$= W K_{U,U} W^\top \triangleq K_{\text{KISS-GP}}. \tag{10}$$

In (Wilson & Nickisch, 2015), the authors use local cubic interpolation to compute W, which results in W being extremely sparse. Because sparse matrix operations with W are so fast, many more inducing points are permitted than in other inducing point methods, which results in more accurate, scalable kernel approximations.

If it is assumed that X is 1-dimensional, $K_{U,U}$ will have Toeplitz structure, allowing $\mathcal{O}(m \log m)$ matrix-vector multiplications. Since W is sparse, matrix-vector multiplications with K_{SKI} only cost $\mathcal{O}(n + m \log m)$ computations and $\mathcal{O}(n + m)$ storage. If the data is d-dimensional and $K_{U,U}$ exhibits Kronecker structure, matrix-vector multiplications with K_{SKI} cost $\mathcal{O}(n + dm^{1+\frac{1}{d}})$ operations and $\mathcal{O}(n + dm^{\frac{2}{d}})$ storage. Inference is performed by using Conjugate Gradient (CG) Optimization with fast matrix-vector multiplications (MVMs) to solve $K_{\text{KISS-GP}}^{-1}\mathbf{y}$, which converges to within machine precision in $j \ll n$ iterations. Learning is performed by computing $\log |K_{\text{KISS-GP}}| = \sum_i \log V_{ii}$ where V is a diagonal matrix of eigenvalues yielded from the eigendecomposition $K_{\text{KISS-GP}} = QVQ^\top$ Importantly, the full covariance matrix over the training data is never computed during inference or learning with KISS-GP, which allows for fast computations (Wilson & Nickisch, 2015).

Notably the SoR approximation can itself be seen as a SKI approximation using zero-noise, zero-mean GP regression with the base covariance kernel k itself:

$$K_{SoR} = K_{X,U} K_{U,U}^{-1} K_{U,U} K_{U,U}^{-1} K_{U,X}. \tag{11}$$

However, the interpolation weights are still not sparse, making SoR very computationally expensive, still.

While local cubic interpolation may work well in certain interpolation tasks, we should not expect that it will perform well in *every* interpolation task. Additionally, since the method of local cubic interpolation via convolution as described by (Keys, 1981) and used by (Wilson & Nickisch, 2015) cannot be tuned in any way, it cannot adapt better fit to kernel at hand, even if we have the exact analytical form of the function which it interpolates. This suggests that one may achieve more accurate kernel approximations by exploring alternative fast interpolants.

2.2.4. Compactly Support Kernels

A separate approach to scalable GPs is the use of compactly supported kernels (CSKs). CSKs are kernel functions possessing finite local support and accordingly sparse covariance matrices. Research in this area has primarily focused on finding valid CSKs (Chernih et al., 2014) and the ad-hoc construction of CSKs to entirely replace standard kernels used in GP inference and learning in order to achieve scalability over large datasets (Zhang et al., 2004). However, it is inherently impossible for large length-scales to be well-represented by CSKs because the covariance is necessarily zero between pairs of points outside of the support, which must be small to maintain tractability. Thus, the wholesale replacement of kernels by constructed CSK counterparts is not warranted unless the GP performs local interpolation.

An alternative method of using CSKs for scalable GP regression is to use a prior mean determined by a separate (fast) regressor so that large-scale variation is captured by the prior mean, and small-scale variation is captured by the compactly supported covariance kernel (Kaufman et al., 2011). This approach largely defeats the purpose of using GPs for general modelling purposes, as the compactly supported kernel only accounts for small-scale local regression.

There are many potential choices of CSKs to use for a particular task. While the interpolating kernel need not be stationary itself, we limit the scope of our study to stationary CSKs because most kernels which we seek to interpolate are functions whose behavior is not significantly different over its support. In this work, we consider several families of compactly supported radial basis functions, including the *Wendland functions* (Chernih et al., 2014), the *truncated power function* (Gneiting, 2001), the *Wu functions* (Wu, 1995), and *Euclid's Hat* (Gneiting, 1999). To the authors' knowledge, these are the most commonly used extant CSKs.

These are stationary kernels that are monotonically decreasing as distance increases, similar to the squared exponential or Matèrn kernels. Also similar to the Matèrn kernel, each of these families possess a smoothness parameter q. That is, as one increases the smoothness parameter, one obtains smoother kernel functions, which result in smoother functions sampled from a GP with corresponding covariance kernel. Additionally, one cannot generally find kernels that are compactly supported in arbitrary dimension, so most families also possess a dimension parameter s, such that a kernel is compactly supported in $d \leq s$ dimensions.

For illustration, example kernels of the Wendland family of functions are shown in figure 1. We have also included in Appendix C plots of each of these CSKs and samples

drawn from their GPs. Such plots help one develop intuition for which kernel is the most likely perform best in a regression or interpolation task. Other compactly supported

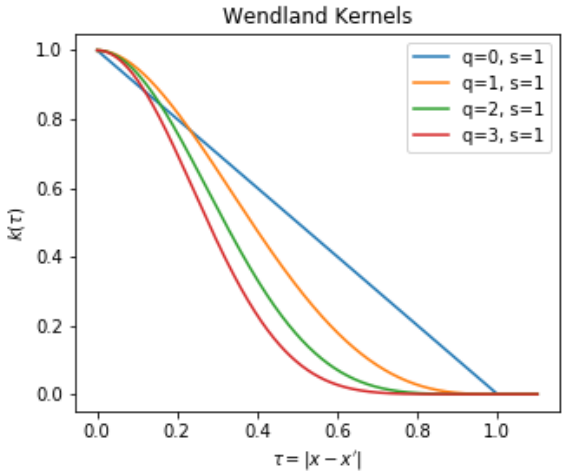

Figure 1. Wendland family kernels as the smoothness parameter q is varied.

kernels can be derived from existing compactly supported kernels. Simply multiplying a valid kernel by a compactly supported kernel creates a new compactly supported kernel with the same support.

3. Methodology

3.1. Compactly Supported Kernels Gaussian Processes for Kernel Interpolation

In this work, we test the hypothesis that one may achieve a better tradeoff between kernel interpolation accuracy and computational efficiency by combining scalable kernel interpolation and compactly supported kernels. In particular, we propose an extension of the SKI framework by replacing the local cubic interpolation strategy of KISS-GP with CSK GP kernel interpolation. We term this use of GPs with compactly supported kernels for interpolation integrated within the SKI framework, **CSK-SKI**.

Because the support of a CSK is bounded in Euclidean space, the resulting covariance matrix is sparse, which can be exploited for computational efficiency. (Rasmussen & Williams, 2006) (Zhang et al., 2004). Additionally, as kernel methods themselves, GP kernel interpolation with CSKs allow the direct encoding of various structural properties of the interpolated kernel function. This is why we believe their use has much potential to significantly outperform KISS-GP in terms of accuracy, while still remaining competitive with respect to run-time. Moreover, CSKs use trainable hyperparameters, e.g. length scales, which can be optimized using the marginal likelihood. By specifying a prior over CSK length scales, one may impose constraints on the sparsity of the covariance matrix. Finally, by combination with other kernels, it is possible to perform compactly supported kernel learning for the purpose of kernel interpolation.

The CSK-SKI method uses a similar formulation to the other SKI methods, SoR and KISS-GP. However, its interpolations are performed by a zero-noise, zero-mean Gaussian process with a compactly supported kernel. The approximation is:

$$K_{X,X} \approx K_{X,U} K_{U,U}^{-1} K_{U,X} \tag{12}$$

$$\approx B_{X,U} B_{U,U}^{-1} K_{U,U} B_{U,U}^{-1} B_{X,U}^{\top} \tag{13}$$

$$\triangleq K_{\text{CSK-SKI}} \tag{14}$$

$B_{X,U}$ is the $n \times m$ cross-covariance matrix for a given CSK between training data points and inducing points, and $B_{U,U}$ is the $m \times m$ covariance matrix over the inducing points for the same CSK.

Note that, in fact, the CSK-SKI kernel is in fact a valid covariance kernel, because it is guaranteed to be positive definite. This is because the positive definiteness of $K_{U,U}$ implies the positive-definiteness of $Q^{\top} K_{U,U} Q$ for any $m \times i$ matrix Q for any i. Thus, the CSK-SKI kernel is valid because:

$$[B_{X,U} B_{U,U}^{-1}]^{\top} = B_{U,U}^{-1} B_{X,U}^{\top}. \tag{15}$$

3.2. Computational Efficiency

By construction, $B_{X,U}$ is sparse, and $B_{U,U}$ is a sparse, symmetric matrix. In the 1-dimensional case, $B_{U,U}$ is also a banded matrix whose bandwidth is determined by the support of the CSK and the spacing of the inducing point grid.

More generally, for dimensionality d and CSK support s and grid spacing Δ, both $B_{U,U}$ and $B_{X,U}$ have $\left(2\lfloor \frac{s}{\Delta} \rfloor\right)^d$ non-zero elements per row. For small bandwidth $b = \lfloor \frac{s}{\Delta} \rfloor$, the level of sparsity is similar to the weight matrices of KISS-GP, which have 4^d non-zero elements per row.

Similar to KISS-GP, we can perform fast matrix multiplications to perform efficient inference and learning. Specifically, we can compute the matrix-vector multiplication (MVM) $K_{\text{CSK-SKI}}\mathbf{v}$ by using fast MVMs for each component. In the 1-dimensional case, the MVM for $B_{U,X}$ takes $\mathcal{O}(bn)$ time due to sparsity. Solving the linear system $B_{U,U}^{-1}\mathbf{v}$ takes $\mathcal{O}(b^2 m)$ time, since $B_{U,U}$ is a symmetric, banded, Toeplitz matrix. The MVM with $K_{U,U}$ takes $\mathcal{O}(m \log m)$ due to its Toeplitz structure. Therefore, a MVM with $K_{CSK-SKI}$ takes $\mathcal{O}(b^2 m + bn + m \log m)$ time. Inference may then be performed computing $K_{\text{CSK-SKI}}^{-1}\mathbf{y}$ efficiently using Conjugate Gradient Optimization, which only depends on fast matrix multiplica-

tions.

If data are d-dimensional ($d > 1$), then fast MVMs are still possible, provided base kernel $K_{U,U}$ and interpolating kernel $B_{U,U}$ exhibit Kronecker structure. In this case, $K_{U,U}\mathbf{v}$ and $B_{U,U}^{-1}\mathbf{v}$ can both be computed in $\mathcal{O}(dm^{1+\frac{1}{d}})$ time (Wilson et al., 2014). Note, though, that m is the total number of inducing points, which depends on the dimensionality: if there are m' inducing points in each dimension, $m = (m')^d$. However, when $K_{U,U}$ and $B_{U,U}$ are Kronecker products, MVMs and linear system solves decompose into successive matrix multiplications of the matrices of the Kronecker product (Schäcke). Since the Kronecker multiplicands are themselves structure kernel matrices, MVMs are actually much faster to calculate.

However, most compactly supported kernels cannot be decomposed as a product across dimensions. Most are piecewise polynomials of the Euclidean distance between points. For instance, an example kernel from the Wu family of functions is

$$\phi_{3,3}(\tau) = (1-\tau)_+^4(16 + 29\tau + 20\tau^2 + 5\tau^3).$$

Nevertheless, one can construct CSKs that do exhibit such structure. If

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix}, \qquad \boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x}'_a \\ \boldsymbol{x}'_b \end{bmatrix}$$

and k_1 and k_2 are valid kernels, then $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}_a, \boldsymbol{x}'_a)k_2(\boldsymbol{x}_b, \boldsymbol{x}'_b)$ is a valid kernel. Moreover, if k_1 and k_2 are compactly supported kernels, the result is itself a compactly supported kernel. In this way, we construct CSKs whose covariance matrix is a Kronecker product. The computational efficiency of CSK-SKI is empirically evaluated in section 4.

3.3. Non-zero Mean

In the process described thus far, the CSK-based GPs used for kernel interpolation used a zero prior mean. However, we surmised that because the kernel interpolation task itself has zero noise, and we know exactly what function it must approximate, that using a nonzero mean function could allow us to interpretably and simply encode such knowledge into said GPs, potentially yielding greater interpolation accuracy (Rasmussen & Williams, 2006). In (Kaufman et al., 2011), the authors describe a framework for application of CSKs within noiseless GP regression contexts involving fitting a simpler regression model (e.g. local polynomial regression) to the data, and then training a GP on the residuals. This was done to compensate for the inherent limitations of CSKs when expressing covariances between points in the input domain (i.e. that covariances between inputs very far away from each other are assumed to be 0). This process is equivalent to use of the simpler regression model as the mean function for the GP itself; the simpler regression model is intended to model any large-scale variation within the input domain, with the compactly-supported kernel itself only modelling small-scale variation adequately captured by a kernel with compact support.

As a way of potentially improving CSK-SKI interpolation accuracy per unit time, we experimented with this method for our interpolation task by computing a bicubic kernel interpolant and reconstructing the covariance matrix of a chosen base kernel, the very interpolant used in KISS-GP (Wilson et al., 2014). Then, we proceeded to compute the residuals and use them as targets for our CSK-based GP interpolants. The Wendland CSK was used for these experiments. During prediction, we simply added back the interpolation values computed by the original bicubic method to those computed by our CSK GP to get the final kernel value predictions.

These experiments did not yield good approximations for a variety of base kernels (see Sec 4.3)and therefore, we proceeded with all other experiments using a simple zero GP mean for the CSK GP interpolant.

3.4. CSK Hyperparameter Learning and Tuning

As described before, one benefit to using CSK GP regression as a kernel interpolant is that the interpolant can be automatically tuned to fit the interpolation task at hand. One simple approach is to simply first learn GP hyperparameters in a kernel regression task constructed from inducing points.

This procedure is as follows: one first determines the kernel which is to be interpolated. Then, one determines the number of inducing points that will yield tractable inference. Inducing points are generated and the kernel is evaluated at each distance (e.g. if there are m inducing points that are Δ far apart, the kernel is evaluated at $k(0), k(\Delta), ..., k(m\Delta)$). Then, kernel learning is performed normally on this constructed data set. That is, CSK hyperparameters of the interpolating GP are learned via maximizing the marginal likelihood. This task will not be computationally intensive since the CSK is sparse, provided fast banded matrix operations are available. Additionally, the task does not depend on the number of true data points, though the number of inducing points could be large.

However, if the CSK length scales are updated via unconstrained optimization, it is possible for the learned length scales to become too large for tractable kernel interpolation, since length scales directly correspond to CSK support. As a solution, one can place a prior the length scale parameter to penalize high length scales. This acts as a soft constraint on the support of the CSK.

An experiment demonstrating this process is described in section 4.6.

3.5. Implementation

GPyTorch is a powerful python library enabling quick creation of GP models for a variety of different tasks. Although integration of our CSK-SKI methodology within this framework alongside the extant KISS-GP implementation is desirable in the long run, we chose to implement CSK-SKI from scratch using NumPy, SciPy, and bandmat python libraries (Travis E, 2006) (Jones et al., 2001–) (Shannon, 2018) for construction of and operations with banded matrices. Currently, efficient banded matrix solution routines *are* implemented within LAPACK, and by extension, SciPy, but we are not aware of any corresponding PyTorch implementation. However, we fully believe that LAPACK banded matrix operations will be added to PyTorch in future, at which time CSK-SKI could be implemented in GPyTorch. Nevertheless, the simplicity of our implementation gives us more control over the implementation and facilitates development and benchmarking. Similarly, KISS-GP was implemented using only SciPy and NumPy for purposes of comparison. We expect that comparisons made with our implementation will extend to potential future implementations in GPyTorch.

4. Experiments

To test CSK-SKI and compare against the benchmark set by KISS-GP, we performed several experiments. The experiments include a matrix-vector multiplication run-time analysis, comparisons of running time versus kernel interpolation accuracy on varied base kernels, regression on zero-noise synthetic data, regression on noisy, real-world datasets, and learning the CSK hyperparameters.

4.1. Run-time of Matrix-Vector-Multiplications

We benchmarked each of the component calculations to verify that their quasi-linear running time with respect to the number of inducing points. Each matrix-vector multiplication against randomly generated target vectors was performed five times, with the average time taken over the trials. For this experiment, the bandwidth of the CSK was set to 6, which was chosen to reflect the fact that CSK-SKI generally performs best with more bandwidth than that of KISS-GP. The results of this experiment are shown in figure 11. Interestingly, we found that the Toeplitz matrix-vector multiplication was by far the most expensive step in CSK-SKI MVMs. This result implies that having a larger bandwidth than 2, as in KISS-GP, does not make CSK-SKI dramatically more expensive.

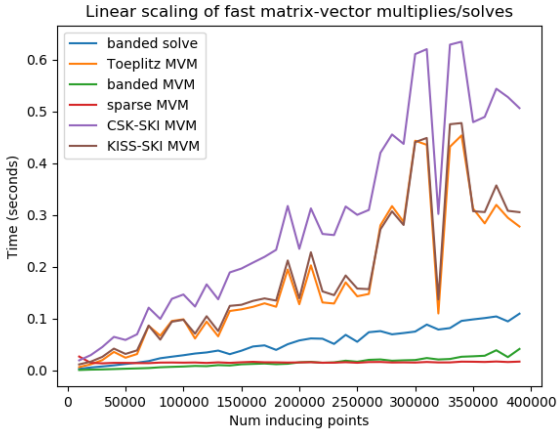

Figure 2. Running time of component MVMs in CSK-SKI MVMs.

4.2. Covariance Matrix Reconstruction

As a preliminary test, we replicate an experiment performed in (Wilson & Nickisch, 2015) to compare our CSK-SKI kernel approximation with the true full RBF kernel. We sample 1000 data points from $\mathcal{N}(0, 25)$ and compute the full covariance matrix on this data after sorting. For both KISS-GP and CSK-SKI, 40 inducing points were used on a regular grid to interpolate the kernel. The bandwidth was chosen to be 4, because in this case both the running time and mean absolute error is approximately equal for both CSK-SKI and KISS-GP. The interpolation errors are shown in figures 3 and 4. We also show the values of the kernel against values of τ, to provide further insight into the interpolations.

It is noteworthy that all of the error for the CSK interpolation is positive, meaning that in this case the interpolation does not over-estimate the covariance. While it is possible for a zero-noise, zero-mean Gaussian process to overestimate its function, the length scale of the CSK is small enough such that covariance between inducing points and interpolated points is relatively small. This causes the interpolating GP to revert back to its zero mean. While, as noted in 3.3, our attempts to create an informative prior mean did not prove beneficial, in some applications one may still prefer the kernel approximation to under or over estimate covariance, in which case one can choose the prior mean appropriately.

To examine accuracy per-unit time, we tested multiple configurations of Wendland CSKs (figure 5) and the product kernel of RBF and the truncated power function (figure 6) and measured the mean absolute kernel interpolation error versus computation time, varying the number of inducing points between 500 and 2000 by increments of 150.

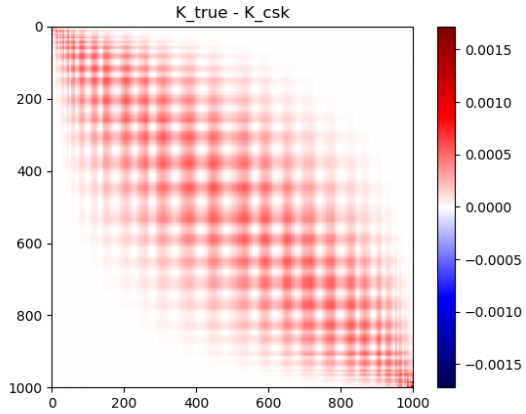

Figure 3. Difference between CSK-SKI reconstruction and true covariance matrix. CSK-SKI used Wendland CSK with $q = 2$ CSK, bandwidth=4, and 40 inducing points.

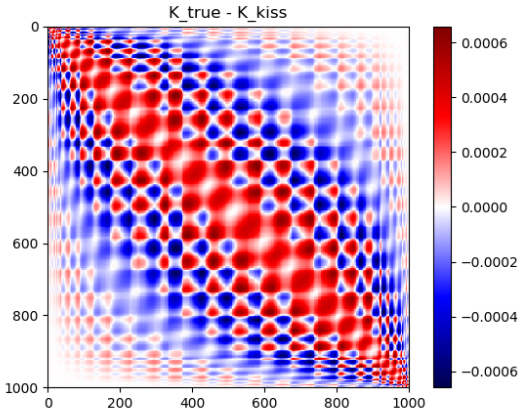

Figure 4. Difference between KISS-GP reconstruction and true covariance matrix. KISS-GP used 40 inducing points.

The RBF-CSK interpolation is very poor compared to either Wendland CSK or KISS-GP interpolation. However, the Wendland CSK kernel interpolation is very competitive with KISS-GP. While KISS-GP is very fast, the Wendland CSK with smoothness parameter $q = 2$ and bandwidth $b = 100$ has an order of magnitude lower interpolation error while being only roughly twice as computationally expensive. Indeed, if one uses fewer than 500 inducing points, the Wendland CSK with bandwidth $b = 100$ and smoothness $q = 2$ achieves significantly higher accuracy per unit time than local cubic interpolation.

A similar analysis was repeated for a variety of base kernels other than RBF, varying the number of inducing points between 400 and 4000 by increments of 150. The base kernels examined included the Ornstein-Uhlenbeck (OU), Matèrn (MA), Periodic (PER), Local-Periodic (L_PER), Rational-Quadratic (RQ), Spectral-Mixture (SM), and 3 "combination" kernels defined respectively as:

$$\text{comb1} = \text{RQ} \times \text{RBF} \times \text{L\_PER} \tag{16}$$

$$\text{comb2} = \text{RQ} \times \text{RBF} + \text{L\_PER} \tag{17}$$

$$\text{comb3} = \text{RBF} \times \text{PER} \times \text{L\_PER} + \text{RBF} \times \text{RQ} \tag{18}$$

The results of these analyses are displayed in Appendix D in Figures 22-30. Generally we observed a significant outperformance in interpolation accuracy with at least one tuning of CSK-SKI over KISS-GP for all of the base kernels tested except for OU, SM, and comb4, where CSK-SKI was only competitive with KISS-GP in terms of accuracy achieved, albeit at a marginally higher computational cost. We understand that this result is due to the fact that the OU kernel more quickly varies than other kernels; because the Wendland CSK is constructed to replicate the behavior of the extremely smooth RBF kernel, the CSK-based GP interpolation proves inadequate for very quickly varying kernels. The results for the SM kernel were somewhat surprising, but can potentially be explained by the fact that hyperparameters were set arbitrarily rather than being determined by training on a particular data set, potentially yielding a quickly varying kernel similar to OU for which CSK-SKI is a poor interpolant. We suggest CSK-SKI's performance for the OU and SM kernels as base kernels, as an interesting area for further examination.

We note that our results show that error does not always decrease with the number of inducing points. Our hypothesis as to why this can occur is because some of the decrease in error due to increased number of inducing points can be offset by greater inaccuracy due to more of the CSK's supporting points lying outside of the inducing point grid leading to worse interpolation in those locations. This could potentially be remedied in a similar fashion as in KISS-GP, by adding more points outside of the grid. Addressing this issue would be an interesting area of further work.

We repeated a similar suite of experiments analyzing a variety of CSKs belonging to two other families: the Wu and Euclid's Hat functions. However, these CSKs yielded similar results to those we present here for the Wendland CSKs .

4.3. Synthetically-Generated Data Set

In practice, one seeks regression inference accuracy, not just kernel interpolation accuracy. Two kernels approximations with similar approximation error could perform worse

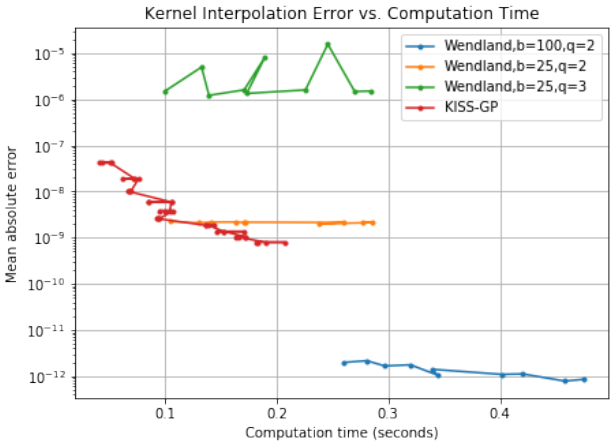

Figure 5. Wendland CSK SKI performance versus time compared to KISS-GP.

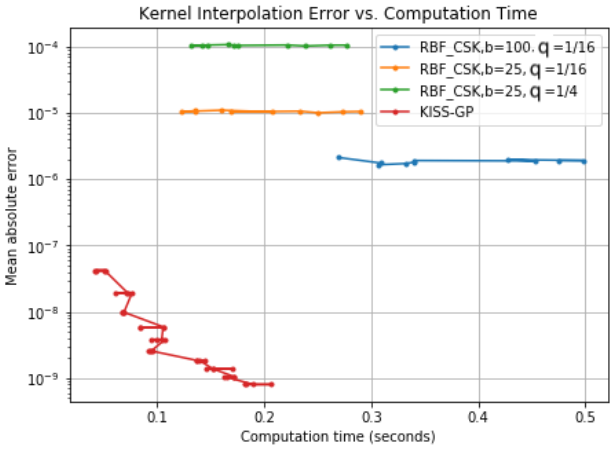

Figure 6. RBF * Truncated Power CSK SKI performance versus time compared to KISS-GP.

in regression depending on the distribution of the error. To prove that the CSK-SKI kernel approximation is not detrimental to regression accuracy, we created a toy data set using the function $sinc(x) = \frac{\sin(x)}{x}$. We uniformly sampled 10,000 points from this univariate function and divided half into a training set and half into a test set. KISS-GP regression was compared to CSK-SKI regression by its regression accuracy. Indeed, CSK-SKI and KISS-GP achieved near zero error. However, the running time for KISS-GP, which ran in 0.09 seconds, was faster than CSK-SKI, which ran in 0.22 seconds. To test CSK-SKI in higher dimensions with CSKs constructed as Kronecker products, we also experimented with a more difficult synthetic two-dimensional regression task. For this task, x values were drawn randomly from an isotropic truncated normal distribution between 0 and 1, and y values were determined by the polynomial

$$y(\boldsymbol{x}) = \boldsymbol{x}_1^2 + 4\boldsymbol{x}_1\boldsymbol{x}_2.$$

The base kernel was chosen an RBF kernel with length scale 3. The number data points and inducing points were $n = 100$ and $m = 10000$ respectively. The CSK was chosen to be a Wendland kernel, which has been shown to perform well for RBF base kernels, and the bandwidth was $b = 10$, determined by guessing a reasonable length scale for the CSK in interpolating the RBF kernel. CSK-SKI and KISS-GP achieved nearly identical accuracy, achieving 0.000348 and 0.000347 mean squared error in 74.3 and 162 ms respectively.

4.4. Real-World Data Set

We tested our model on several noisy real world datasets, including CO2 concentrations measured in Mauna Loa (Hipel & McLeod, 2014). The hyperparameters were tuned manually to 1000 inducing points and a base length scale of 6.5. The CSK-SKI was set to a very small bandwidth of 3. CSK-SKI and KISS-GP achieved similar mean squared error of 8.94 and 8.39 respectively, and inference times of 0.095 and 0.052 seconds. This shows our method is competitive with KISS-GP on real world noisy data. Hyperparameter learning would be helpful in achieving greater accuracy.

4.5. CSK-GP Interpolants with Non-zero Means

As described in 3.3, we benchmarked the performance in accuracy and time of a CSK-based GP interpolant using the predictions of a simpler regression function instead of 0 as its mean. Unfortunately, the resulting approximations were not accurate. Approximations for a variety of base kernels were examined, including the RBF, OU, and Matèrn kernels (Figures 8, 20, 21 respectively). This result was unexpected, and may suggest that the Wendland CSK used is not appropriate for GP regression of the residual function. We suggest this result as an interesting area for further examination.

4.6. Learning CSK Hyperparameters

To demonstrate the CSK learning procedure, we took the base kernel to be an OU kernel with length scale equal to 10. We generated a small number of inducing points (10) so that the interpolation would be more illustrative. We implemented CSKs in GPyTorch without sparse operations, which is tractable because of the small number of inducing points. We tested four interpolating kernels in this task. The first three were all Wendland kernels with $q = 2$ and $s = 1$. The fourth was the product of the same Wend-

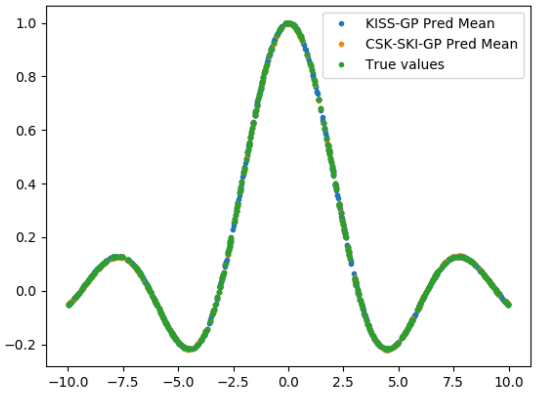

Figure 7. Comparison of KISS-GP and CSK-SKI on learning the sinc function with 1000 training points. Both methods achieved negligible error but CSK-SKI took 0.22 seconds while KISS-GP took 0.09 seconds.

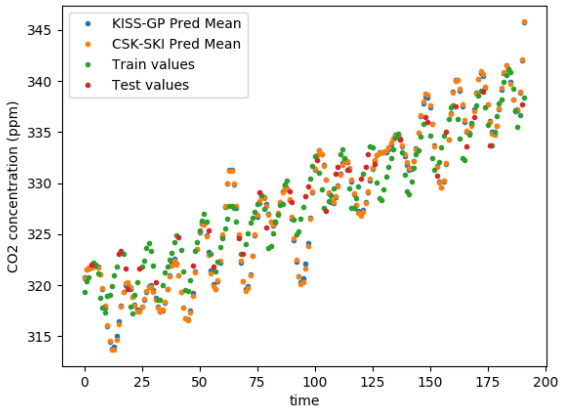

Figure 8. Mauna Loa CO2 concentrations.

CSK	Length Scale	MAE
Unoptim. Wendland	10.0	.00516
Optim. Wendland	119.1	.00081
Optim. Wendland With Prior	76.6	.00084
Wendland $\times$ Spectral Mixture	2.9	.1787

Table 1. CSK hyperparameter learning experiment results.

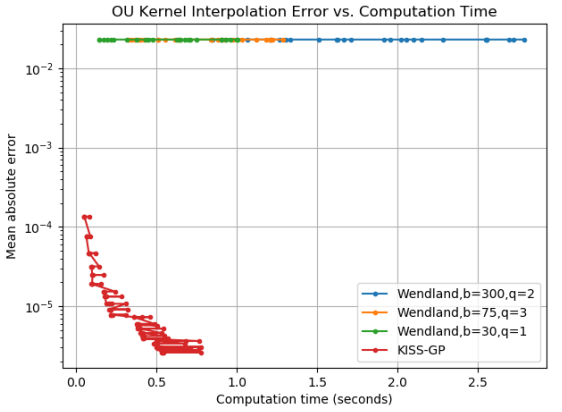

Figure 9. Runtime vs. Accuracy for Nonzero Mean CSK-SKI

land kernel and an SM kernel with four mixtures. Each of the four kernels was initialized with bandwidth equal to 10. The first pure Wendland kernel was not optimized, to serve as a baseline. The second Wendland kernel was optimized without a prior. The third Wendland kernel was optimized with a Gaussian prior with mean 1 and variance 1 over the log length scale. The fourth kernel was also optimized with no prior. The learned lengths scales and resultant interpolation error for these models are shown in table 1. As expected, the optimized CSK was able to achieve significantly lower interpolation error, but its length scale became very large. The optimized CSK with a penalty for high length scale achieved similar accuracy but learned smaller a smaller length scale, and therefore will be more tractable. Finally, the product kernel performed very poorly. We hypothesize that the inaccuracy is due to poor spectral mixture initialization rather than lack of merit.

5. Discussion

Our experiments upheld our initial belief that CSKs could be used to perform accurate kernel interpolation while maintaining tractability. We have have described a method for computing fast matrix-vector multiplications with CSK-SKI and demonstrated its quasi-linear scaling with the number of inducing points. We have shown that in many cases one can achieve higher kernel interpolation accuracy per unit time using CSK-SKI than KISS-GP. We have also found that, in general, Wendland CSKs are much better interpolants than those constructed with the truncated power function. We have also shown that it is possible to automatically learn interpolating kernel hyperparameters with constraints on its tractability.

However, there exist a number of limitations to our proposed method: firstly, for a number of base kernels, namely the OU and Spectral Mixture kernels, our interpolation method is only competitive with local cubic interpolation in terms of accuracy, while also being more expensive. Additionally, even for base kernels for which CSK-SKI does significantly outperform KISS-GP in accuracy (while be-

ing more expensive), at least in our learning scenarios, the more accurate approximation produced by CSK-SKI does not seem to significantly affect the accuracy of the predictions made over those made with KISS-GP. In fact, our results suggest that KISS-GP produces a sufficiently accurate kernel approximation for practical use in a variety of scenarios.

Nevertheless, we believe that CSK-SKI is a useful scalable GP method and that future may continue to show its benefits over of KISS-GP. Specifically, future implementation within GPyTorch will be critical for evaluating CSK-SKI in the context of kernel learning and difficult regression problems. Integration with GPyTorch will also enable CSK hyperparameters to be learned simultaneously with the base kernel itself. Additionally, we had attempted to automatically learn new compactly supported kernels by composition with the spectral mixture kernel. Since the experiment seems to have only failed due to poor spectral kernel initialization, we are interested in its continued study. Additionally, exploration of exactly why our framework for CSK-based GP interpolation with a regression function as the prior mean did not work in our experiments could inform even further improvements to the accuracy of CSK-SKI, helping to make it a more desirable alternative to KISS-GP.

References

Chernih, Andrew, Sloan, Ian H, and Womersley, Robert S. Wendland functions with increasing smoothness converge to a gaussian. *Advances in Computational Mathematics*, 40(1):185–200, 2014.

Gneiting, Tilmann. Radial positive definite functions generated by euclid's hat. *Journal of Multivariate Analysis*, 69(1):88 – 119, 1999. ISSN 0047-259X. doi: https://doi.org/10.1006/jmva.1998.1800. URL http://www.sciencedirect.com/science/article/pii/S0047259X98918000.

Gneiting, Tilmann. Criteria of pólya type for radial positive definite functions. *Proceedings of the American Mathematical Society*, 129(8):2309–2318, 2001.

Hipel and McLeod. Co2 (ppm) mauna loa, 1965-1980, Feb 2014. URL https://datamarket.com/data/set/22v1/co2-ppm-mauna-loa-1965-1980#!ds=22v1&display=line.

Jones, Eric, Oliphant, Travis, Peterson, Pearu, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed ¡today¿].

Kaufman, Cari G, Bingham, Derek, Habib, Salman, Heitmann, Katrin, Frieman, Joshua A, et al. Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics*, 5(4):2470–2492, 2011.

Keys, R. G. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Accoustics, Speech, and Signal Processing*, ASSP-29:1153–1160, 1981.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Rasmussen, C. E. and Williams, C. K. I. Gaussian processes for machine learning. http://www.gaussianprocess.org/gpml/chapters/, 2006. Accessed: 2018-10-29.

Saatçi, Yunus. *Scalable inference for structured Gaussian process models*. PhD thesis, University of Cambridge, 2012.

Schäcke, Kathrin. On the kronecker product. Master's thesis, University of Waterloo.

Shannon, Matt. bandmat, 2018. URL https://github.com/MattShannon/bandmat. [Online; accessed ¡today¿].

Silverman, B. W. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society*, 47(1):1–52, 1985.

Snelson, Edward and Ghahramani, Zoubin. Sparse gaussian processes using pseudo-inputs. In Weiss, Y., Schölkopf, B., and Platt, J. C. (eds.), *Advances in Neural Information Processing Systems 18*, pp. 1257–1264. MIT Press, 2006. URL http://papers.nips.cc/paper/2857-sparse-gaussian-processes-using-pseudo-input pdf.

Travis E, Oliphant. A guide to NumPy, 2006.

Wilson, A. G. and Nickisch, H. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning ICML 2015*, pp. 1775–1784, 2015.

Wilson, Andrew G, Gilboa, Elad, Nehorai, Arye, and Cunningham, John P. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*, pp. 3626–3634, 2014.

Wu, Zongmin. Compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4(1):283, Dec 1995. ISSN 1572-9044. doi: 10.1007/BF03177517. URL https://doi.org/10.1007/BF03177517.

Zhang, Hao Helen, Genton, Marc, and Liu, Peng. Compactly supported radial basis function kernels. *Available at www4. stat. ncsu. edu/hzhang/research. html*, 2004.

Appendices

A. Run-ime at higher dimensions

To see how our CSK-SKI method scales with respect to dimensions, we created synthetic datasets of increasing dimensions. The data is randomly generated values between 0 and 10, and label is the sum of the values of every dimension. Keeping all other parameters constant, we measured the run-time at inference for different dimensionalities. CSK-SKI scaled exponentially over KISS-GP in runtime, while maintaining similar accuracy.

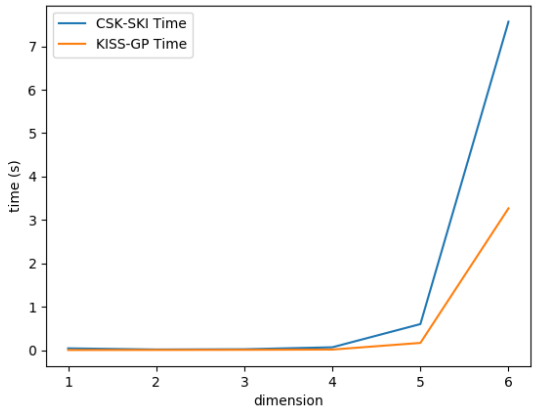

Figure 10. Running time of CSK-SKI vs KISS-GP for higher dimensional data.

B. Noisy synthetic data

We can achieve better accuracy for small and noisy datasets. 200 random data points were generated with random Gaussian noise sampled with a variance of 0.001. 25 inducing points were used, along with a CSK bandwidth of 3. CSK-SKI had a mean squared error of 0.0011 in 0.012 seconds. KISS-GP achieved worse accuracy in less time with a mean squared error of 0.011 in 0.0060 seconds.

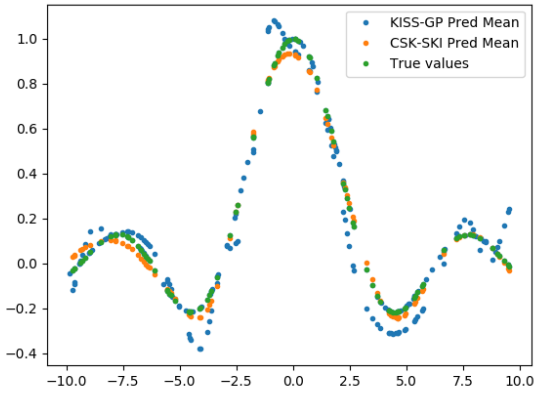

Figure 11. Noisy sinc function

C. Compactly Supported Kernel Plots

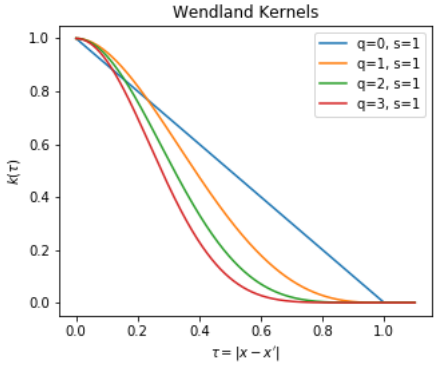

Figure 12. Wendland family kernels as the smoothness parameter q is varied.

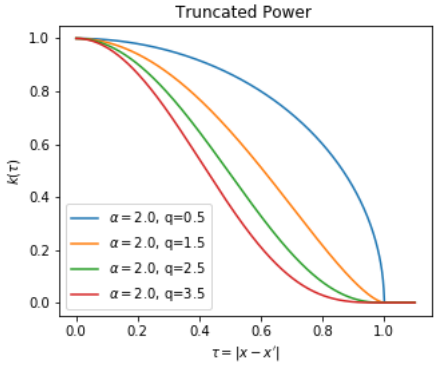

Figure 13. Truncated Power Function kernels as the smoothness parameter q is varied. Note that for the Truncated Power Function, α does not correspond exactly to a dimension parameter. However, for a given dimensionality, only certain combinations of q and α are positive definite.

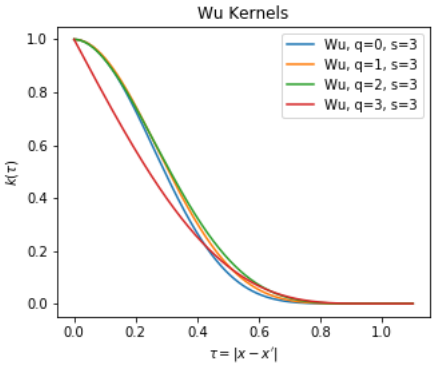

Figure 14. Wu kernels as the smoothness parameter q is varied.

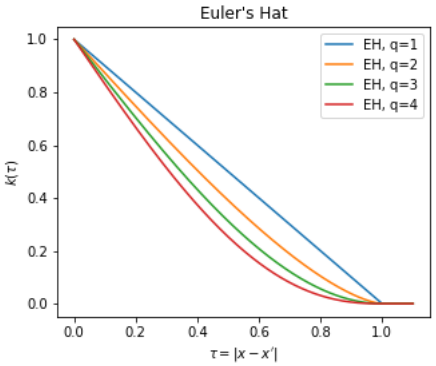

Figure 15. Euclid's Hat as the smoothness parameter q is varied.

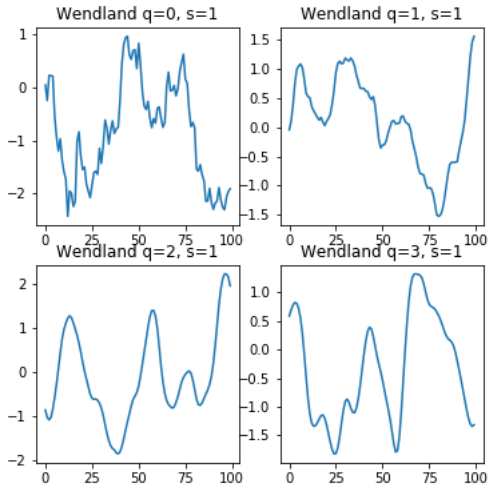

Figure 16. Samples from a GP with Wendland covariance kernel as the smoothness parameter q is varied.

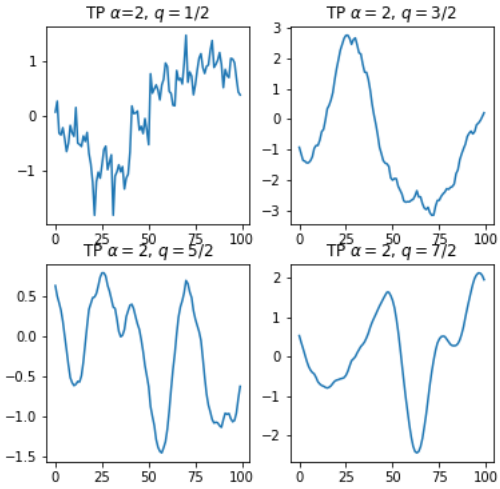

Figure 17. Samples from a GP with Truncated Power Function covariance kernel as the smoothness parameter q is varied.

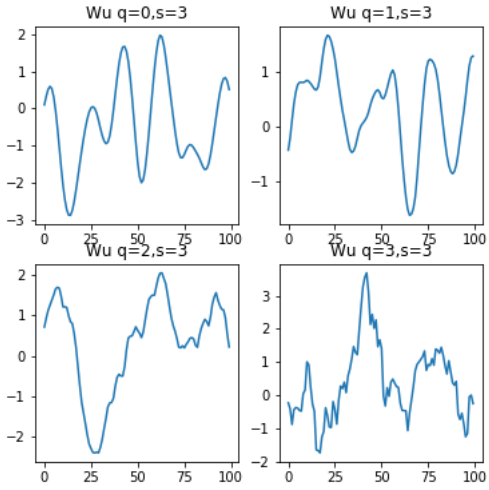

Figure 18. Samples from a GP with Wu covariance kernel as the smoothness parameter q is varied.

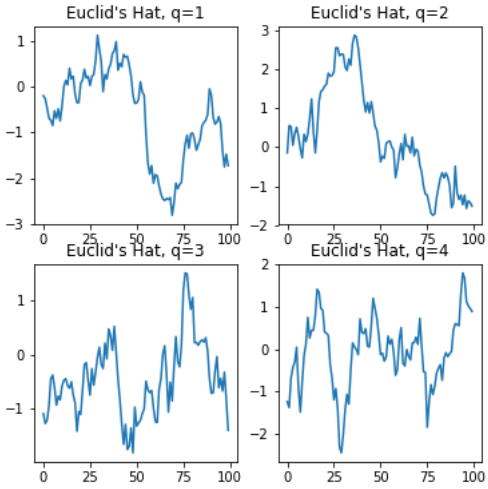

Figure 19. Samples from a GP with Euclid's Hat covariance kernel as the smoothness parameter q is varied.

D. Supplementary Run-time vs. Interpolation Accuracy Plots

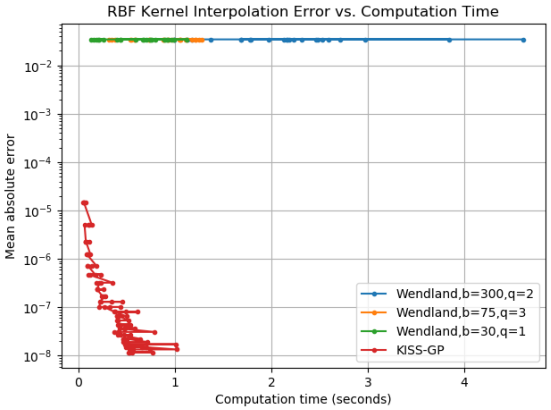

Figure 20. Runtime vs. Accuracy for Nonzero Mean CSK-SKI

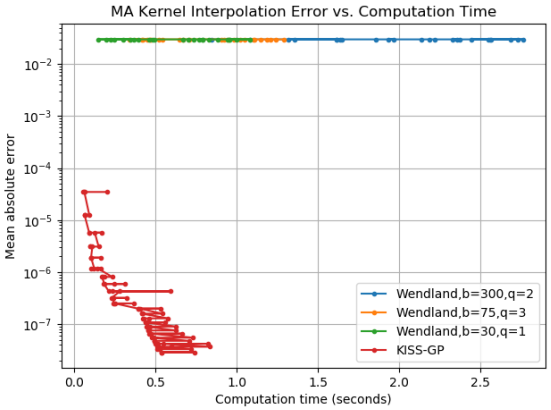

Figure 21. Runtime vs. Accuracy for Nonzero Mean CSK-SKI

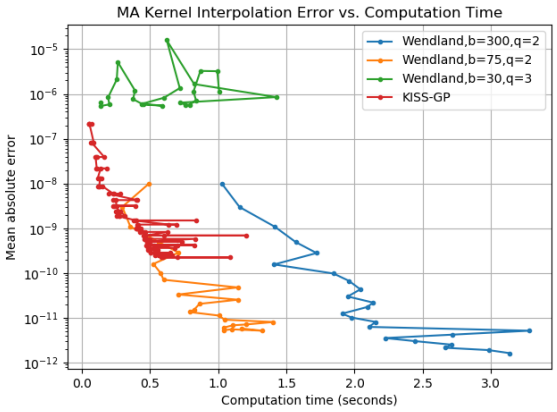

Figure 22. Error vs. Computation time for Matèrn Base Kernel

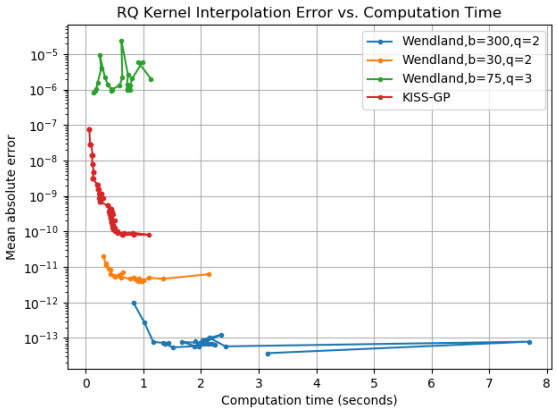

Figure 23. Error vs. Computation time for RQ Base Kernel

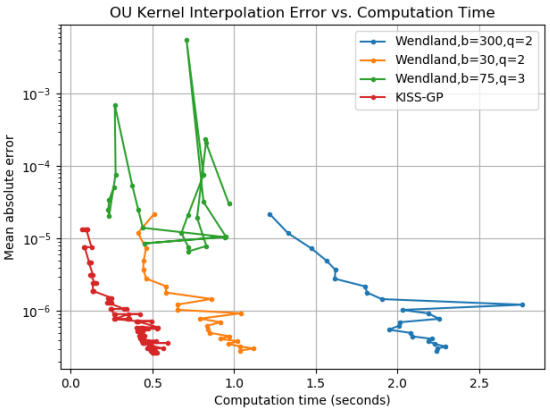

Figure 24. Error vs. Computation time for OU Base Kernel

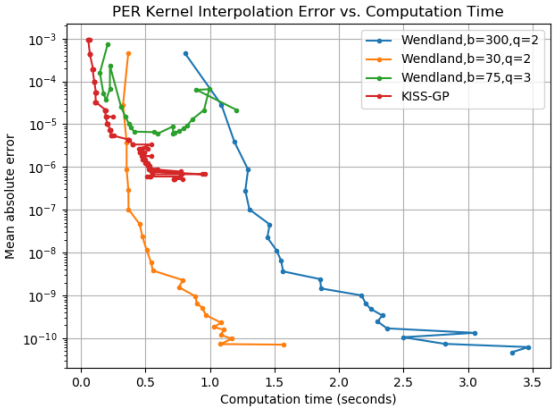

Figure 25. Error vs. Computation time for Periodic Base Kernel

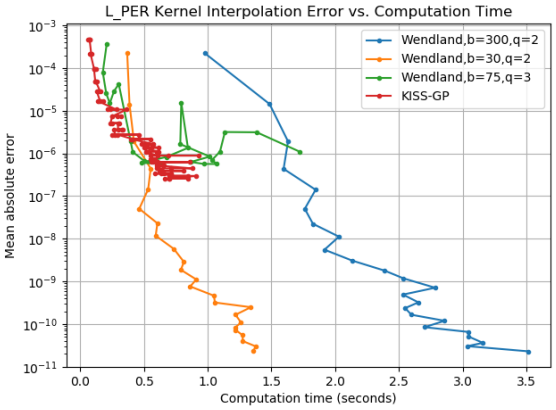

Figure 26.

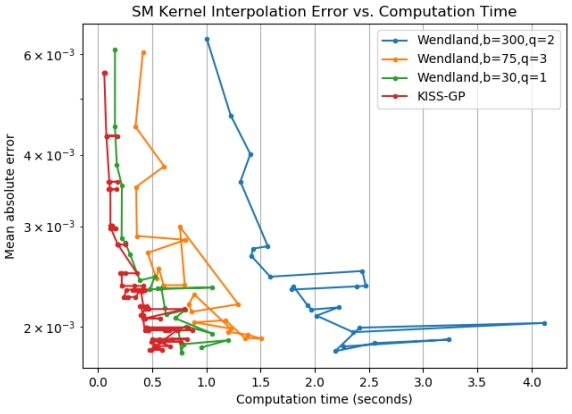

Figure 27. Error vs. Computation time for Spectral Mixture Base Kernel

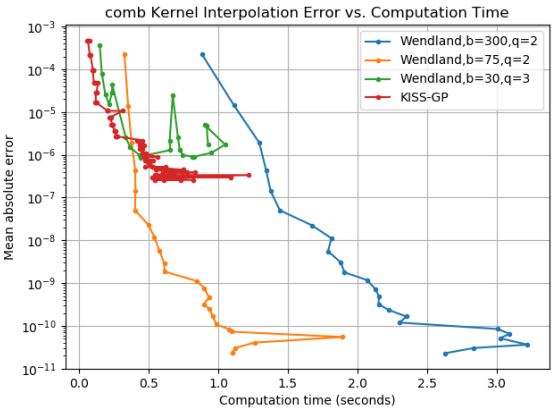

Figure 28. Error vs. Computation time for Combination 1 Base Kernel

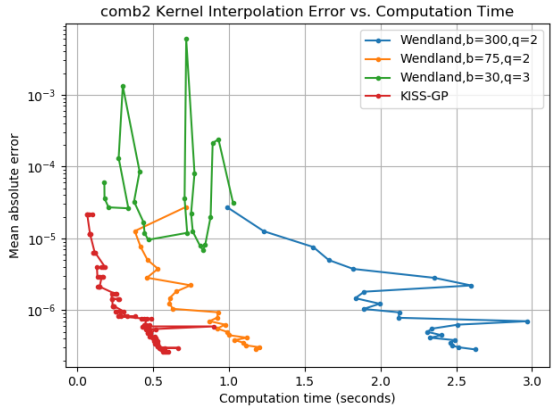

Figure 29. Error vs. Computation time for Combination 2 Base Kernel

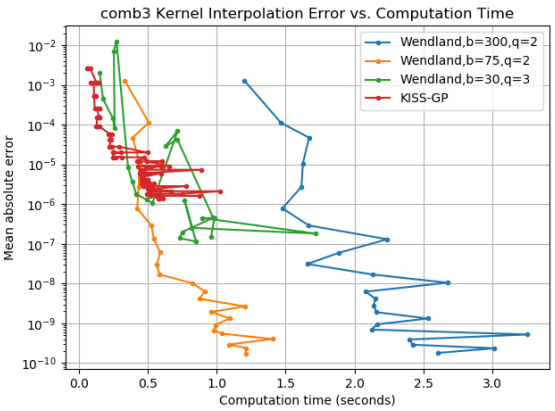

Figure 30. Error vs. Computation time for Combination 3 Base Kernel