Fast and Accurate Gaussian Kernel Ridge Regression Using Matrix Decompositions for Preconditioning

Gil Shabat*  Era Choshen  Dvir Ben-Or  Nadav Carmel
Playtika AI Research Lab, Israel

Abstract

This paper presents a method for building a preconditioner for a kernel ridge regression problem, where the preconditioner is not only effective in its ability to reduce the condition number substantially, but also efficient in its application in terms of computational cost and memory consumption. The suggested approach is based on randomized matrix decomposition methods, combined with the fast multipole method to achieve an algorithm that can process large datasets in complexity linear to the number of data points. In addition, a detailed theoretical analysis is provided, including an upper bound to the condition number. Finally, for Gaussian kernels, the analysis shows that the required rank for a desired condition number can be determined directly from the dataset itself without performing any analysis on the kernel matrix.

1 Introduction

Kernel methods are a way to embed the input space into a high dimensional feature space, which enables learning over a richer and more complex hypothesis class. One of the most elementary applications of kernel methods is Kernel Ridge Regression (KRR). The problem setup can be defined in the following manner - let the training data be defined as \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}\) where \(\mathcal{X} \subseteq \mathbb{R}^d\) is the input domain and \(\mathcal{Y} \subseteq \mathbb{R}\) is the output domain, let \(k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}\), and a regularization parameter \(\beta > 0\), with the response for a given input \(x\) estimated as:

\[
f(x) = \sum_{j=1}^{n} k(x, x_j) \alpha_j
\]  

where \(\alpha = (\alpha_1 \cdots \alpha_n)^T\) is the solution of the equation

\[
(K + \beta I)\alpha = b
\]  

*Corresponding author: gils@playtica.com
1.1 Related work

Building scalable and computationally efficient algorithms for kernel methods is an active research area. Some methods use low rank approximations to approximate the kernel directly, such as Nyström approximation or random Fourier features. This approach is very popular and has many variants where the main idea is to reduce the computational cost by low rank approximation while retaining reasonable model quality. Recent work on these approaches can be found in [20, 9, 11, 26].

A different approach uses fast matrix vector multiplications, e.g. Fast Gauss Transform or Fast Multipole methods to solve the regression using Krylov iterations [32, 11, 34]. These methods reduce the computational cost of each Krylov iteration significantly, from $O(n^2)$ to $O(n \log n)$ or even $O(n)$. However, when the kernel matrix is ill-conditioned, the number of iterations required might be huge. Another approach is to use a preconditioner to reduce the condition number of the kernel matrix and in so doing reduce the required number of Krylov iterations. These approaches are discussed in [17, 35, 8].

1.2 Contributions

The paper proposes the use of randomized matrix decompositions [18] (specifically, the interpolative decomposition [16]), for constructing a preconditioner for the kernel matrix. The preconditioner can be used to solve the kernel ridge regression problem [2] using a substantially smaller
number of iterations while maintaining high accuracy. Moreover, the preconditioner is designed in its structure to be both efficient in its application to the kernel matrix and strict in memory consumption.

The paper presents a theoretical analysis of the condition number of the preconditioned matrix. Furthermore, for a Gaussian kernel, the paper provides a theoretical lower bound for the low rank approximation required to reach the desired condition number, this bound can be calculated directly from the data.

2 Preliminaries

2.1 Notation

Throughout the paper, matrices are indicated by capital letters and vectors and scalars by small letters. The norm \( \| \cdot \| \) when applied to vectors, refers to the standard Euclidean norm, i.e. \( \|v\| = \sqrt{\sum_{i=1}^{n} v_i^2} \) and to the spectral norm when applied to matrices, i.e. \( \|A\| = \sup \frac{\|Av\|}{\|v\|} \). The norm \( \| \cdot \|_A \) applied to a vector with respect to a positive semidefinite matrix \( A \) is defined \( \|v\|_A = \sqrt{\langle v, Av \rangle} \). Singular values are indicated in descending order, by \( \sigma_1, \ldots, \sigma_n \) and eigenvalues by \( \lambda_1, \ldots, \lambda_n \) also in descending order when they are real. \( A \geq 0 \) on a square matrix indicates that \( A \) is positive semidefinite and \( A \geq B \) means \( A - B \geq 0 \).

2.2 Interpolative and CUR decompositions

Interpolative decomposition (ID) \[16\] is a matrix factorization that can be applied to any matrix \( A \) and defined as the following:

\[
A_{m \times n} \approx B_{m \times k}P_{k \times n}
\]  

(3)

where \( B \) is a subset of the \( k \)-columns of \( A \) and \( P \) is an interpolation matrix with certain properties (such as small norm) that is used for reconstructing \( A \) from \( B \). The columns of \( B \) are computed by rank revealing factorization \[15, 28, 24, 6, 19\] in order to get an error \( \|A - BP\| \) bounded by a factor proportional to the the \( k \) singular value of \( A \). In addition to the deterministic algorithm for computing the ID that is described in \[16\], there is a randomized version that starts by projecting \( A \) onto a low dimensional space using a random normally distributed matrix \[23, 18\]. This property is used implicitly in Algorithm \[\]

The CUR decomposition \[21, 10\], is a pseudo-skeleton decomposition \[12, 27\] that factors a matrix \( A \) into three matrices, where \( C \) and \( R \) are subsets of the columns and rows of \( A \) respectively, and \( U \) is defined as the inverse matrix of the overlap between \( C \) and \( R \). Most of the matrices discussed in the paper are symmetric, so the Nyström approximation will be used as a special case of the CUR decomposition. The following Lemma gives an error bound for the CUR decomposition using columns and rows selected by the interpolative decomposition:
Lemma 2.1. \[ \text{Let } A \in \mathbb{R}^{m \times n}, \text{ that satisfies } A = CV^T + E \text{ and } A = WR + \tilde{E}, \text{ where } C,R \text{ are the } k \text{ columns and } k \text{ rows of } A, \text{ respectively, and } W,V^T \text{ are the interpolation matrices from the interpolative decomposition. Suppose further that } R \text{ is full rank, and that } U \in \mathbb{R}^{k \times k} \text{ is defined as } U = V^T R^\dagger. \text{ Then} \]
\[ ||A - CUR|| \leq ||E|| + ||\tilde{E}|| \] (4)

Remark 2.2. \[ \text{Note that } ||E|| \leq p(n,k)\sigma_{k+1}, \text{ furthermore for the deterministic ID, } p(n,k) = \sqrt{1 + k(n-k)}. \text{ Equivalently, for a similar bound exists for the randomized ID, the reader is referred to [29].} \]

2.3 Fast Gauss Transform

Fast Gauss transform (FGT) is a variant of the fast multipole method (FMM) [13]. FMM was originally formulated as an efficient method for complex physical calculations, efficiently performing matrix-vector products. FGT deals with the evaluation of the discrete Gauss transform:
\[ G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/\epsilon^2} \] (5)
where \{x_i\}_{i=1}^{N}, x_i \in \mathbb{R}^d \text{ represent the centers of the Gaussians, each of which has bandwidth } \epsilon, \text{ and } q_i \text{ are the weight coefficients for each Gaussian. Direct evaluation of the summation for a set of points, } \{y_j\}_{j=1}^{M}, y_i \in \mathbb{R}^d, \text{ is computationally costly in large scale scenarios, as it requires } \mathcal{O}(MN) \text{ operations. FGT allows setting a desired degree of precision } \Delta \text{ for the approximation of the Gaussian function, and reduces the complexity of the summation (5) to } \mathcal{O}(M+N), \text{ accompanied by a constant factor, which grows exponentially with the dimension } d \text{ [14]. To overcome this disadvantage, which makes plain FGT inapplicable to high-dimensional data, the Fast Improved Gauss (FIG) transform [37] uses tree data structures and an adaptive space subdivision technique, leading to reduction of the constant factor to asymptotically polynomial order. It is worth noting, that the computational complexity in } d \text{ is dependent on the data as well as the selection of } \epsilon \text{ [25].} \]

Definition 2.1. \[ \text{Let } \text{FIG}(X,Y,\epsilon,q) \text{ represent the application of the FIG transform to compute equation (5), where } X \in \mathbb{R}^{N \times d} \text{ and } Y \in \mathbb{R}^{M \times d} \text{ represent matrices of data points for which the kernel is generated, weighted by the vector } q \in \mathbb{R}^{N}. \text{ According to this notation, it holds that } \text{FIG}(Y,X,\epsilon,q) = \text{FIG}^*(X,Y,\epsilon,q). \]

The algorithm discussed in the paper can be applied to any other kernels that can utilize fast matrix-vector product such as other FMM functions. Another approach for fast matrix-vector product for any kernel and based on nearest neighbors is described in [31].
3 Description of the algorithm

The proposed algorithm is based on conjugate gradient with a preconditioner designed not only to be effective in its ability to reduce the condition number, but also in its efficiency to be applied in terms of computational cost and memory consumption. In order to keep the time and storage complexity to a minimum, the proposed scheme uses the Fast Improved Gauss transform \([37, 38]\) ("FIG transform"). If the kernel is Gaussian, then the FIG transform can be used for applying the kernel to any vector in \(O(n)\), but the preconditioner has to be applied using the FIG transform as well, to keep the advantage of using a Gaussian kernel. In order to achieve this, the kernel is approximated using a Nyström decomposition, i.e. \(\tilde{K} = CUC^T\), where \(C \in \mathbb{R}^{n \times k}\) is a subset of columns of \(K\) and \(U \in \mathbb{R}^{k \times k}\) is the inverse of a submatrix of \(C\). Adding the ridge regression parameter, \(\beta\) and using the Woodbury matrix identity:

\[
(\tilde{K} + \beta I)^{-1} = (CUC^T + \beta I)^{-1} = \beta^{-1} I - \beta^{-1} C(\beta U^{-1} + C^T C)^{-1} C^T
\]  

Since \(C\) is a submatrix of \(K\), then the application of both \(C\) and \(C^T\) to a matrix, can be done using FIG transform in \(O(n + k)\). \(U^{-1}\) is also a subset of \(K\) (unlike \(U\) itself, which involves matrix inversion) and can therefore be applied in the same way, in \(O(k)\). The Woodbury matrix identity requires computing the inverse of \(\beta U^{-1} + C^T C\), however this is a small matrix of size \(k \times k\). Constructing the matrix \(\beta U^{-1} + C^T C\) can be done by the application of the FIG transform to the identity matrix, \(I\) (of size \(n\)) in \(O(nk)\) operations. Each application of Eq. 6 involves solving the following linear system

\[
(\beta U^{-1} + C^T C)x = b.
\]  

It is better to store the Cholesky factor of the matrix in Eq. 7 but this matrix tends to be ill-conditioned in large scales. Instead, Eq. 7 is modified to:

\[
(\beta I + U^{\frac{1}{2}} C^T C U^{\frac{1}{2}})U^{-\frac{1}{2}} x = U^{\frac{1}{2}} b.
\]  

where the matrix \(\beta I + U^{\frac{1}{2}} C^T C U^{\frac{1}{2}}\) is generally more stable, and its Cholesky decomposition can be applied to solve Eq. 8 for \(z = U^{-\frac{1}{2}} x\) and then restoring \(x\) using \(x = U^{\frac{1}{2}} z\).

\[
L^T L = \beta I + U^{\frac{1}{2}} C^T C U^{\frac{1}{2}}
\]  

where \(L\) is the Cholesky factor of size \(k \times k\). The algorithm can be viewed as two steps:

- Preparation stage (or “setup” stage), which selects anchor data points from the dataset and also performs some computations to be used later in the iterative stage, such as building the matrix \(L\) of the Cholesky decomposition and the matrix \(U^{\frac{1}{2}}\).

- Iterative stage, which applies conjugate gradient using the preconditioner that was computed
Selection of the anchor points can be done in various ways, such as random sampling or farthest point sampling (FPS) [5]. In this work, the suggested method is based on the interpolative decomposition (or equivalently, on rank revealing decompositions, [15, 28, 24], which has the following advantages:

- It is strongly related to the CUR decomposition [36], and therefore also to the Nyström approximation. The interpolative decomposition is also integral to the method, yielding theoretical spectral bounds that can give theoretical guarantees for the overall performance of the algorithm.
- It selects columns of the input matrix, which enables the usage of the FIG-transform later on in the algorithm.
- It has a randomized version, which can be computed (using the FIG-transform) in linear time and the most demanding computational part can be parallelized.

**Algorithm 1 AnchorSelection**: Select anchor points using randomized interpolative decomposition approach (RRQR)

1: **Input**: $X$ - Dataset, matrix of size $n \times d$.  
2: $k$ - Number points to choose (rank),  
3: $l \geq k$ - number of random projections, typically slightly larger than $k$  
4: $\epsilon > 0$ - Width of the Gaussian kernel  
5: **Output**: $S$ - A set of anchor points from $X$  
6: Generate a random matrix $\Omega \in \mathbb{R}^{l \times n}$, s.t. $\omega_{i,j} \sim \mathcal{N}(0, 1)$  
7: $Y \leftarrow \text{FIG}(X, X, \epsilon, \Omega)$ # Apply FIG transform to $\Omega$  
8: $Y^T P = Q R$ # Apply Strong RRQR to $Y$, $P$ is a permutation matrix, but can viewed as a vector  
9: $S \leftarrow P(1 : k)$ # Choose the first $k$ elements in $P$  
10: return $S$

**Remark 3.1.** Algorithm 1 uses sampling technique based on interpolative decomposition. When using pivoted QR, the computational complexity is $\mathcal{O}(n^2)$. In practice, the performance is very similar to RRQR.

**Remark 3.2.** The computational complexity of Algorithm 2 is $\mathcal{O}(nk + k^3)$.

**Remark 3.3.** Applying the preconditioner (Algorithm 3) in each iteration takes $\mathcal{O}(n + k^2)$ operations.

**Remark 3.4.** The computational complexity of Algorithm 4, when using pivoted QR in Algorithm 4 is $\mathcal{O}(nl^2 + nk + k^3 + (n + k^2)t)$, where $t$ is the number of iterations used in the conjugate gradient.
**Algorithm 2 BuildCholesky**: Computes the Cholesky decomposition according to Eq. 9

1: **Input**: $X$ - Dataset, matrix of size $n \times d$,
2: $S$ - Vector of length $k$,
3: $\beta > 0$ - Ridge parameter,
4: $\epsilon > 0$ - Width of the Gaussian kernel
5: **Output**: $L$ - Cholesky factor
6: $X_S \leftarrow X(S,:)$, # $X_S$ is a subset of $X$ containing the anchor points
7: $U^{-1} \leftarrow \text{FIG}(X_S, X_S, \epsilon)$
8: Using SVD or EVD build $U^{\frac{1}{2}}$
9: $Y \leftarrow \text{FIG}(X, X_S, \epsilon, U^{\frac{1}{2}})$
10: $Y \leftarrow \beta I + U^{\frac{1}{2}} \text{FIG}(X_S, X, \epsilon, Y)$
11: $Y = L^T L$ # Cholesky decomposition
12: return $L, U^{\frac{1}{2}}$

**Algorithm 3 ApplyPreconditioner**: Applies the preconditioner according to Eq. 6

1: **Input**: $X$ - Dataset, matrix of size $n \times d$.
2: $S$ - Anchor points, matrix of size $k \times d$
3: $\beta > 0$ - Ridge parameter
4: $\epsilon > 0$ - Width of the Gaussian kernel
5: $L$ - Cholesky factor
6: $x$ - Input vector of length $n$.
7: **Output**: $r$ - The application of the preconditioner $(\tilde{K} + \beta I)^{-1}$ to $x$.
8: $X_S \leftarrow X(S,:)$, # $X_S$ is a subset of $X$ containing the anchor points
9: $y \leftarrow U^{\frac{1}{2}} \text{FIG}(X_S, X, \epsilon, x)$
10: Solve $L^T L z' = y$ # back substitution
11: $z \leftarrow U^{\frac{1}{2}} z'$
12: $r \leftarrow \beta^{-1} x - \beta^{-1} \text{FIG}(X, X_S, \epsilon, z)$
13: return $r$

**Algorithm 4 SolveKRR**: Solves the Gaussian kernel ridge regression

1: **Input**: $X$ - Dataset, matrix of size $n \times d$.
2: $\beta > 0$ - Ridge parameter
3: $\epsilon > 0$ - Width of the Gaussian kernel
4: $k$ - Number of points to sample.
5: $l \geq k$ - Number of random projections to use.
6: $b$ - vector of length $n$.
7: **Output**: $x$ - The solution to the equation $(K + \beta I)x = b$.
8: Select anchor points $S$ from $X$ using Algorithm 1
9: Build the Cholesky factor $L$ using Algorithm 2
10: Solve $(K + \beta I)x = b$, using CG, where $Kx$ can be computed by $\text{FIG}(X,X,\epsilon, x)$ and the preconditioner can be applied using Algorithm 3
11: return $x$
Remark 3.5. Anchor points selection can be done differently, for example by random sampling or FPS. In this case, the theoretical bounds will not hold, but may still produce good results in practice and reduce the computational cost of Algorithm [7]. For example, from $O(nl^2)$ using pivoted QR, to $O(k)$ using random sampling.

4 Theoretical Analysis

Lemma 4.1. Let $K \in \mathbb{R}^{n \times n}$ be a symmetric matrix, and let $\tilde{K} = CUC^T$ be its Nyström approximation, then $\|K - \tilde{K}\| \leq 2\sigma_{k+1}(K) \cdot p(k, n)$ where $p(k, n)$ is a function bounded by a low degree polynomial in $k$ and $n$.

Proof. By Lemma 2.1

$$\|K - \tilde{K}\| \leq \|E\| + \|\tilde{E}\| \quad (10)$$

Note however that since $K$ is symmetric, $R = C^T$ meaning

$$\tilde{K} = CUR = CUC^T \quad (11)$$

hence $\|E\| = \|\tilde{E}\|$. Therefore it follows immediately that

$$\|K - \tilde{K}\| \leq 2\|E\| \quad (12)$$

We also know from by definition of RRQR, that the decomposition must satisfy

$$\|E\| \leq \sigma_{k+1}(K) \cdot p(k, n) \quad (13)$$

We therefore combine (7) and (8), yielding

$$\|K - \tilde{K}\| \leq 2\sigma_{k+1}(K) \cdot p(k, n) \quad (14)$$

Lemma 4.2. Let $K \in \mathbb{R}^{n \times n}$ be a positive semidefinite matrix, and let $\tilde{K} = CUC^T$ its Nyström approximation, then $K - \tilde{K} \succeq 0$.

Proof. The lemma infers directly from the Schur’s compliment of $K$.

Lemma 4.3 ([4] pp. 673). For any two matrices $A, B \in \mathbb{C}^{m \times n}$ the following holds:

$$|\sigma_i(A) - \sigma_i(B)| \leq \|A - B\| \quad (15)$$
Definition 4.1. [2] The numerical rank of the Gaussian kernel $G^X_\epsilon$ up to precision $\delta \geq 0$ is

$$\rho_\delta(G^X_\epsilon) \triangleq \# \left\{ j : \frac{\sigma_j(G^X_\epsilon)}{\sigma_1(G^X_\epsilon)} \geq \delta \right\}$$

Theorem 4.4. [2, 3] Let $X = \{x_i\}_{i=1}^n \in \mathbb{R}^d$ be a set bounded by a box $B = I_1 \times I_2, \ldots I_d$, where $I_1, I_2, \ldots, I_d$ are intervals in $\mathbb{R}$. Let $q_i$ be the length of the $i$th interval, i.e. $q_i = |I_i|$ and let $G^X_\epsilon$ be the associated kernel matrix, i.e. $(G^X_\epsilon)_{i,j} = g_\epsilon(x_i, x_j)$, then

$$\rho_\delta(G^X_\epsilon) \leq \prod_{i=1}^d (\lfloor \gamma q_i \rfloor + 1), \quad (16)$$

where $\gamma = \frac{2}{\pi} \sqrt{\epsilon^{-1} \ln \delta^{-1}}$.

The following theorem gives an upper bound to the condition number:

Theorem 4.5. Let $\tilde{K} \in \mathbb{R}^{n \times n}$ be a rank $k$ Nyström approximation for a positive semidefinite matrix $K \in \mathbb{R}^{n \times n}$, such that $\|K - \tilde{K}\| \leq M\sigma_{k+1}(K)$, for a positive constant $M$ (that may depend on $n$ and $k$) and a ridge parameter $\beta > 0$, then

$$\text{cond} \left[ (\tilde{K} + \beta I)^{-1}(K + \beta I) \right] \leq 1 + \frac{M\lambda_{k+1}(K)}{\beta} \quad (17)$$

Proof. Combining Lemma 4.2 and Lemma 4.3 with the fact that $K$ and $\tilde{K}$ are positive semidefinite, gives

$$0 \leq K - \tilde{K} \leq \|K - \tilde{K}\|I \leq M\sigma_{k+1}(K) = M\lambda_{k+1}(K) \quad (18)$$

Modifying Eq. (18) and adding $\beta I$ gives

$$\tilde{K} + \beta I \leq K + \beta I \leq \tilde{K} + \beta I + M\lambda_{k+1}(K)I \quad (19)$$

Applying $(\tilde{K} + \beta I)^{-\frac{1}{2}}$ to both sides of the equation and using the fact that $(\tilde{K} + \beta I)^{-\frac{1}{2}}$ is symmetric positive semidefinite gives

$$I \leq (\tilde{K} + \beta I)^{-\frac{1}{2}}(K + \beta I)(\tilde{K} + \beta I)^{-\frac{1}{2}} \leq I + M\lambda_{k+1}(K)(\tilde{K} + \beta I)^{-1} \quad (20)$$

Clearly, $(\tilde{K} + \beta I)^{-1} \leq \beta^{-1}I$, which yields

$$I \leq (\tilde{K} + \beta I)^{-\frac{1}{2}}(K + \beta I)(\tilde{K} + \beta I)^{-\frac{1}{2}} \leq (1 + \frac{M\lambda_{k+1}(K)}{\beta})I. \quad (21)$$

and finally,

$$\text{cond} \left[ (\tilde{K} + \beta I)^{-1}(K + \beta I) \right] = \text{cond} \left[ (\tilde{K} + \beta I)^{-\frac{1}{2}}(K + \beta I)(\tilde{K} + \beta I)^{-\frac{1}{2}} \right] \leq 1 + \frac{M\lambda_{k+1}(K)}{\beta} \quad (22)$$
which completes the proof. □

**Corollary 4.6.** Let $K$ be a Gaussian kernel matrix over dataset $X \in \mathbb{R}^d$, i.e. $K = G^X$. Let $\tilde{K}$ be a rank $k$ Nyström approximation, such that $\|K - \tilde{K}\| \leq M(n,k)\sigma_{k+1}(K)$ and let $\beta$ be a ridge parameter. Then, for a maximal condition number, $\xi$

$$k \geq \prod_{i=1}^{d} \left( \left\lceil \frac{2q_i}{\pi} \sqrt{\epsilon^{-1} \ln \frac{Mn}{\beta(\xi - 1)}} \right\rceil + 1 \right)$$

(23)

where $\{q_i\}_{i=1}^{d}$ are the lengths of the intervals of the bounding box of the dataset $X$.

**Proof.** From Theorem 4.5, the condition number depends on $\lambda_{k+1}(K)$, i.e.

$$\text{cond} \left[ (\tilde{K} + \beta I)^{-1} (K + \beta I) \right] \leq 1 + \frac{M \lambda_{k+1}(K)}{\beta} \leq \xi$$

(24)

where $\bar{M} \triangleq \sup M(k)$. Therefore, $\lambda_{k+1}(K)$ must satisfy $\lambda_{k+1}(K) \leq \frac{\beta(\xi - 1)}{M}$. By defining $\delta = \frac{\beta(\xi - 1)}{M\|K\|}$

$$\gamma = \frac{2}{\pi} \sqrt{\epsilon^{-1} \ln(\delta^{-1})} \leq \frac{2}{\pi} \sqrt{\epsilon^{-1} \ln \frac{Mn}{\beta(\xi - 1)}}$$

(25)

and therefore,

$$k \geq \prod_{i=1}^{d} \left( \left\lceil \frac{2q_i}{\pi} \sqrt{\epsilon^{-1} \ln \frac{Mn}{\beta(\xi - 1)}} \right\rceil + 1 \right)$$

where $\{q_i\}_{i=1}^{d}$ are the lengths of the intervals of the bounding box of the dataset $X$. □

**Remark 4.7.** Corollary 4.6 enables to determine the required rank Nyström approximation and illustrates the tradeoff between the condition number and the low rank, which has implications over the memory consumption and computational load of the algorithm. Smaller condition number yields less iterations on one hand, but on the other hand requires processing larger matrix $\tilde{K}$.

**Remark 4.8.** As a simple example, for the deterministic interpolative decomposition, $M = \sqrt{k(n-k) + 1}$, which yields $\bar{M} = \sqrt{\frac{n^2}{4} + 1} \approx \frac{n}{2}$

**Remark 4.9.** Eq. 25 grows very slowly in $n$, which means that $k$ remains small even when $n$ grows fast. For example, suppose $\bar{M} = n/2$, $\epsilon = 1$, $\xi = 2$ and $\beta = 1$, then for $n = 10^6$, $\gamma = 3.3$ and for $n = 10^8$, $\gamma = 3.82$ which does not affect the value of $k$ according to Corollary 4.6. On the down side, it grows fast in $d$.

**Corollary 4.10.** For a Nyström preconditioner built by choosing $\tilde{K} = CUC^T$, where $C$ are columns of $K$ chosen by the interpolative decomposition, then

$$\text{cond} \left[ (\tilde{K} + \beta I)^{-1} (K + \beta I) \right] \leq 1 + \frac{2\lambda_{k+1}(K)\sqrt{k(n-k) + 1}}{\beta} \leq 1 + \frac{2\lambda_{k+1}(K)(\frac{n}{2} + 1)}{\beta}$$

(26)
Proof. The proof follows immediately by combining Lemma 4.1 with Theorem 4.5.

Remark 4.11. A similar bound can be developed immediately for other matrix decompositions.

5 Numerical Results

In this section, empirical evaluation of the algorithm is presented and compared against the naive implementation (no preconditioning) and RFF preconditioning. The datasets used are described in the supplementary appendix [30]. The number of data points is $n = 100,000$, rank $k = 50$, noise level $\beta = 0.1, 1$ and $l = 60$. The most costly computation in the presented flow is the QR decomposition in Algorithm 1 which is $O(nl^2)$, compared to $O(nds)$ where $s$ the number of Gaussian components, therefore in order to maintain similar computational complexity, the number of Gaussian components was set to $\frac{ld^2}{d}$, and therefore $s = 1200$. Figure 1 shows the comparison between the distance $\|\alpha^* - \alpha^{(i)}\|_{K+\beta I}$, where $\alpha^*$ is the exact solution, and $\alpha^{(i)}$ is the solution at iteration $i$.

![Graph](image)

(a) Stanford Helicopter Dataset $k = 50, \beta = 0.1$  (b) Stanford Helicopter Dataset $k = 50, \beta = 1$

Figure 1: Comparison between different noise levels

Remark 5.1. It is important to note that each iteration of the proposed algorithm is $O(n + k^2)$, while every iteration of RFF is $O(nl^2 + k^2)$ for $s = \frac{ld^2}{d}$. For $n \gg k^2$, each RFF iteration is asymptotically more costly in terms of run-time complexity by a factor of $k^2$.

Remark 5.2. When the spectral decay is fast, a small $k$ will be sufficient for satisfying results.

Figure 2 compares between the $l_2$ norm of the residual error, $r^{(i)} = (K + \beta I)\alpha^{(i)} - b$, where $i$ refers to the iteration number and $\alpha^{(i)}$ is the solution at iteration $i$ of the CG algorithm. Also presented is the distance $\|\alpha^* - \alpha^{(i)}\|_{K+\beta I}$.

It’s worth noting that the graph is slightly misleading, as the number of Gaussian RV’s taken for the RFF preconditioner is very high (in order to achieve the same computational complexity), however per iteration the proposed method is significantly faster, therefore in reality the convergence is still far faster for the proposed method.
Figure 2: For 30K datapoints from accelerating 2D (X,Y) predicting next timestep for X

Figure 3: For 100K datapoints from accelerating 2D (X,Y) predicting next timestep for Y

5.1 Synthetic Electromagnetic Field Simulator

A 3-dimensional synthetic dataset generated by an electric field simulator was used to test the performance of the algorithm. The simulation was composed of the following parts:

- setting the support of the points to \([0, 1]^3\)
- choosing points from a uniform distribution over the defined support at random, where:
  - 5 points represent the positive point-charges
Following the superposition rule, at each sampling point $j$, each field component was summed as follows:

$$e_x[j] = \sum_{i=1}^{5} \frac{1}{r_{ij}^2} \sin(\phi) \cos(\theta)$$  \hspace{1cm} (27)$$

$$e_y[j] = \sum_{i=1}^{5} \frac{1}{r_{ij}^2} \sin(\phi) \sin(\theta)$$  \hspace{1cm} (28)$$

$$e_z[j] = \sum_{i=1}^{5} \frac{1}{r_{ij}^2} \cos(\phi)$$  \hspace{1cm} (29)$$

where $r_{ij}$ is the distance between charge $i$ and the point $j$, and $\theta$ and $\phi$ are the spherical angles with respect to axes $x$ and $z$ respectively. Additionally, an electric potential was calculated at each point, to be used as the problem’s label as follows:

$$\Phi[j] = \sum_{i=1}^{5} \frac{1}{r_{ij}}$$  \hspace{1cm} (30)$$

Figure 4: For 30K datpoints on the EM Field dataset
6 Conclusions

The paper shows that the use of randomized matrix decompositions (specifically RID with CUR) for constructing a preconditioner for the matrix kernel is very successful in solving the scalability drawback intrinsic to kernel matrix methods. The method is strict in memory consumption (no need to hold the \( n \times n \) matrix), maintains accuracy, allows efficient application of the kernel matrix and reduces its condition number. Our results show fast convergence of the CG algorithm while maintaining accuracy, outperforming similar SOTA methods. The paper also provides theoretical insights and bounds for the low-rank approximation needed to reach the desired condition number.
References

[1] Haim Avron and Vikas Sindhwani. High-performance kernel machines with implicit distributed optimization and randomization. *Technometrics*, 58(3):341–349, 2016.

[2] Amit Bermanis, Amir Averbuch, and Ronald R Coifman. Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, 34(1):15–29, 2013.

[3] Amit Bermanis, Guy Wolf, and Amir Averbuch. Cover-based bounds on the numerical rank of Gaussian kernels. *Applied and Computational Harmonic Analysis*, 36(2):302–315, 2014.

[4] Dennis S Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas*. Princeton University Press, 2009.

[5] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[6] S. Chandrasekaran and I. Ipsen. On rank-revealing QR factorisations.

[7] Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):54, 2017.

[8] Kurt Cutajar, Michael Osborne, John Cunningham, and Maurizio Filippone. Preconditioning kernel matrices. In *International Conference on Machine Learning*, pages 2529–2538, 2016.

[9] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

[10] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[11] Nando D Freitas, Yang Wang, Maryam Mahdaviani, and Dustin Lang. Fast Krylov methods for N-body learning. In *Advances in neural information processing systems*, pages 251–258, 2006.

[12] Sergei A Goreinov, Eugene E Tyrtyshnikov, and Nickolai L Zamarashkin. A theory of pseudoskeleton approximations. *Linear algebra and its applications*, 261(1-3):1–21, 1997.

[13] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[14] Leslie Greengard and John Strain. The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.
[15] Ming Gu and Stanley C Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869, 1996.

[16] P.G Martinsson V. Rokhlin H. Cheng, Z. Gimbutas. On the compression of low rank matrices. *SIAM*, 26.

[17] Kenneth L. Clarkson Haim Avron and David P. Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM*, 38:1116–1138.

[18] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[19] Y.P Hong and C.-T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Math. Comp*.

[20] Zhiyun Lu, Avner May, Kuan Liu, Alireza Bagheri Garakani, Dong Guo, Aurélien Bellet, Linxi Fan, Michael Collins, Brian Kingsbury, Michael Picheny, et al. How to scale up kernel methods to be as good as deep neural nets. *arXiv preprint arXiv:1411.4000*, 2014.

[21] Michael W Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[22] Michael W Mahoney et al. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[23] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47–68, 2011.

[24] L Miranian and Ming Gu. Strong rank revealing lu factorizations. *Linear algebra and its applications*, 367:1–16, 2003.

[25] Vlad I Morariu, Balaji V Srinivasan, Vikas C Raykar, Ramani Duraiswami, and Larry S Davis. Automatic online tuning for fast Gaussian summation. In *Advances in neural information processing systems*, pages 1113–1120, 2009.

[26] Cameron Musco and Christopher Musco. Recursive sampling for the Nystrom method. In *Advances in Neural Information Processing Systems*, pages 3833–3845, 2017.

[27] AI Osinsky and NL Zamarashkin. Pseudo-skeleton approximations with better accuracy estimates. *Linear Algebra and its Applications*, 537:221–249, 2018.
[28] C-T Pan. On the existence and computation of rank-revealing LU factorizations. *Linear Algebra and its Applications*, 316(1-3):199–222, 2000.

[29] Vladimir Rokhlin P.G Martinsson and Mark Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30.

[30] Adam Coates Pieter Abbeel and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 2010.

[31] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, pages 1177–1184, 2008.

[32] Vikas C Raykar and Ramani Duraiswami. Fast large scale Gaussian process regression using approximate matrix-vector products. In *Learning workshop*, 2007.

[33] Gil Shabat, Yaniv Shmueli, Yariv Aizenbud, and Amir Averbuch. Randomized LU decomposition. *Applied and Computational Harmonic Analysis*, 44(2):246–272, 2018.

[34] Yirong Shen, Matthias Seeger, and Andrew Y Ng. Fast Gaussian process regression using kd-trees. In *Advances in neural information processing systems*, pages 1225–1232, 2006.

[35] Balaji Vasan Srinivasan, Qi Hu, Nail A Gumerov, Raghu Murtugudde, and Ramani Duraiswami. Preconditioned Krylov solvers for kernel regression. *arXiv preprint arXiv:1408.1237*, 2014.

[36] Sergey Voronin and Per-Gunnar Martinsson. Efficient algorithms for CUR and interpolative matrix decompositions. *Advances in Computational Mathematics*, 43(3):495–516, 2017.

[37] C Yang, R Duraiswami, NA Gumerov, and L Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proceedings Ninth IEEE International Conference on Computer Vision*.

[38] Changjiang Yang, Ramani Duraiswami, and Larry S Davis. Efficient kernel machines using the improved fast Gauss transform. In *Advances in neural information processing systems*, pages 1561–1568, 2005.
$\text{CG—Norm of difference from final solution}$

$\|x_i - x_{\text{fin}}\|_{K+\lambda I}$