EVERYTHING IS VECCHIA: UNIFYING COLUMN NYSTRÖM AND SPARSE VECCHIA APPROXIMATIONS

EAGAN KAMINETZ*

Abstract. This thesis examines two types of factored matrix approximations that are fast and accurate in different contexts: low-rank approximations of positive definite matrices and sparse factored approximations of positive definite matrix inverses. The thesis combines these two approaches using the theoretical framework of Vecchia approximation. The main algorithmic contribution is a greedy method for computing the low-rank portion of a low-rank plus sparse approximation that achieves near-optimal theoretical guarantees. The greedy method is applied to produce preconditioners for iteratively solving linear systems arising from Gaussian processes, where numerical experiments demonstrate significant convergence rate improvements over existing methods.

Key words. Vecchia approximation, Nyström approximation, kernel matrix, Gaussian process, Factorized sparse approximate inverse

AMS subject classifications. 65F55, 65C99, 68T05

1. Motivation.

This thesis asks and answers the following question: what is a fast, accurate algorithm that provides a factored approximation of a positive definite matrix?

A factored approximation of a positive definite matrix is useful for scientific computing in at least two ways. The approximation can be used as a "direct solver", when it is substituted for the original matrix when solving linear systems, calculating determinants, or calculating eigenvectors. Direct solvers are computationally efficient, and they induce error, but the error level is acceptable given a highly accurate approximation. As an alternative approach, a factored approximation can serve as a preconditioner for an iterative method. Iterative methods for linear solves and eigenvalue solves drive the error all the way to zero, but the rate of convergence depends on the approximation quality of the matrix preconditioner; for details see section 2.

In the large-data regime, both direct solvers and preconditioned iterative solvers are much faster than dense factorization methods for scientific computing, and they stretch the limits of the calculations that can be performed. For example, given a $10^5 \times 10^5$ matrix in single precision, it is impossible to execute dense factorization methods using standard software (MATLAB or python) on a laptop-scale computer (64GB RAM). In contrast, preconditioned iterative solvers execute in minutes or hours, and direct solvers are even faster.

This thesis will examine a large, existing body of literature devoted to factored approximations of positive definite matrices. The literature is divided into low-rank approximations [4, 5, 6] and sparse factored approximations of the matrix inverse [11, 18], as well as combinations of the two approaches [24]. All these approximation methods execute quickly and apply in the large-data regime, since they examine just a subset of entries in the matrix. Yet, they are based on different philosophies, since the low-rank approximations are accurate for matrices with quickly decaying eigenvalues and the sparse factored approximations are valid for matrices with off-diagonal decay in the inverse. It is unclear whether these approximations can be combined in an overarching framework.

The thesis considers a unified framework for low-rank plus sparse approximation, and it makes the following three contributions:

*University of California San Diego, La Jolla, CA (ekaminetz@ucsd.edu).

1. The thesis synthesizes the existing literature by showing how existing low-rank and sparse approximation methods [4, 5, 6, 11, 18, 24] can be understood as particular examples of Vecchia approximation [22] with different sparsity patterns. Therefore, the optimality guarantees of the Vecchia approximation [12, 23, 19] extend to all these methods.
2. Using the theory of Vecchia approximations, this thesis introduces a greedy method to build the low-rank part of a low-rank plus sparse approximation [24], and it proves near-optimal theoretical guarantees for this greedy method.
3. Numerical experiments show that the greedy method leads to significant convergence rate improvements when used to precondition iterative methods.

The rest of the thesis is organized as follows. Subsection 1.1 introduces notation. Section 2 provides background on approximate matrix factorizations. Section 3 unifies approaches in section 2 using the theoretical framework of Vecchia approximation. Section 4 introduces the greedy algorithm. Last, section 5 presents numerical experiments.

1.1. Notation. We denote scalars in lower case italics: m, n, r. We denote vectors in lower case boldface: $\boldsymbol{u}, \boldsymbol{v}$. We denote matrices in boldface capital letters: $\boldsymbol{A}, \boldsymbol{B}$. We use $\boldsymbol{u}_i$ to refer to to the ith entry of a vector $u \in \mathbb{C}^n$, and we use $\boldsymbol{A}_{i,j}$ to refer to the (i, j) entry of a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$. Given index sets $S, T \subseteq \{1, \dots, n\}$, we use $\boldsymbol{u}_S$ to refer to the subvector $(\boldsymbol{u}_i)_{i\in S}$, and we use $\boldsymbol{A}_{S,T}$ to refer to the submatrix $(\boldsymbol{A}_{i,j})_{i\in S, j\in T}$. Similarly, $\boldsymbol{A}_{i,:}$ and $\boldsymbol{A}_{:,i}$ indicate the ith row and column of $\boldsymbol{A}$.

We represent the vector of all ones by $\mathbf{1}$. We use $\boldsymbol{e_i}$ to represent the ith standard basis vector, which has ith entry 1 and all other entries 0. We write $[n]$ to mean $\{1 \dots n\}$, and write $m : k$ to refer to $\{m \dots l\}$, especially when indexing vectors or matrices. We use $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to indicate the multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The conjugate transpose of a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ is denoted $\boldsymbol{A}^*$, and the Moore-Penrose pseudoinverse is $\boldsymbol{A}^+$.

The Nyström approximation of a positive definite matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ with respect to a test matrix $\boldsymbol{X} \in \mathbb{C}^{n\times r}$ is defined as

$$\boldsymbol{A}\langle \boldsymbol{X}\rangle = \boldsymbol{A}\boldsymbol{X}(\boldsymbol{X}^*\boldsymbol{A}\boldsymbol{X})^{-1}\boldsymbol{X}^*\boldsymbol{A}.$$

The column Nyström approximation using an index set $R \subseteq \{1, \dots, n\}$ is defined as

$$\boldsymbol{A}\langle R\rangle = \boldsymbol{A}\langle \mathbf{I}(:, R)\rangle = \boldsymbol{A}_{:,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,:}.$$

The Schur complement with respect to index set R is defined as

$$\boldsymbol{A}/R = \boldsymbol{A} - \boldsymbol{A}\langle R\rangle.$$

2. Methods for factored approximations. This section describes several methods to construct factored approximations for positive definite matrices $\boldsymbol{A} \in \mathbb{C}^{n\times n}$. The main purpose of this section is to provide motivation and to set the notation.

2.1. Approximate Cholesky factorization. Every positive definite matrix $\boldsymbol{A}$ has a unique Cholesky factorization, $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^*$, where $\boldsymbol{L}$ is a lower triangular matrix. Since the inverse $\boldsymbol{A}^{-1}$ is positive definite, it also admits a unique Cholesky factorization, $\boldsymbol{A}^{-1} = \boldsymbol{C}\boldsymbol{C}^*$. However, the factorizations $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^*$ and $\boldsymbol{A} = \boldsymbol{C}^{-*}\boldsymbol{C}^{-1}$ are distinct, because $\boldsymbol{L}$ is lower triangular while $\boldsymbol{C}^{-*}$ is upper triangular.

In practice, it is often too expensive to form a Cholesky factorization exactly. However, we can consider an approximate Cholesky factorization $\hat{\boldsymbol{A}} = \hat{\boldsymbol{L}}\hat{\boldsymbol{L}}^*$ or $\hat{\boldsymbol{A}}^{-1} = \hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^*$, where the matrix $\hat{\boldsymbol{L}}$ or $\hat{\boldsymbol{C}}$ is lower triangular and sparse. When such an approximation is available, we can put this factorization into action to solve linear systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ using a direct or indirect approach.

In the direct approach, we replace the original matrix $\boldsymbol{A}$ with an approximate Cholesky factorization $\hat{\boldsymbol{A}}$ and then solve the modified linear system $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \boldsymbol{b}$. At this point, solving the system is extremely efficient. For example, given an approximate factorization $\hat{\boldsymbol{A}}^{-1} = \hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^*$ we can solve the linear system using two matrix–vector products, which require at most $\mathcal{O}(n^2)$ operations and run even faster if $\hat{\boldsymbol{C}}$ is sparse. Alternatively, given an approximate factorization $\hat{\boldsymbol{A}} = \hat{\boldsymbol{L}}\hat{\boldsymbol{L}}^*$, we can solve the linear system using two instances of triangular substitution. The runtime for triangular substitution is linear in the number of nonzero entries, so it requires $\mathcal{O}(n^2)$ operations for a dense matrix and runs even faster when the Cholesky factor $\boldsymbol{L}$ is sparse.

In an indirect method, we do not substitute $\hat{\boldsymbol{A}}$ for $\boldsymbol{A}$ directly. Rather, we use the approximation $\hat{\boldsymbol{A}}$ as a preconditioner in the preconditioned conjugate gradient (PCG) algorithm. We run PCG for some number of iterations, producing better and better estimates of $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ with each iteration. PCG requires just $O(n^2)$ operations per iteration, and after k iterations it produces an approximation $\hat{\boldsymbol{x}}^{(k)}$ with error bounded according to [9, Ch. 11]:

$$\|\hat{\boldsymbol{x}}^{(k)} - \boldsymbol{x}\|_{\boldsymbol{K}} \leq 2\|\boldsymbol{x}\|_{\boldsymbol{K}} \exp\left(-\frac{2k}{\sqrt{\mathfrak{K}(\boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-*})}}\right)$$

or

$$\|\hat{\boldsymbol{x}}^{(k)} - \boldsymbol{x}\|_{\boldsymbol{K}} \leq 2\|\boldsymbol{x}\|_{\boldsymbol{K}} \exp\left(-\frac{2k}{\sqrt{\mathfrak{K}(\boldsymbol{C}^*\boldsymbol{K}\boldsymbol{C})}}\right)$$

The condition number $\mathfrak{K}(\boldsymbol{M})$ of a positive definite matrix $\boldsymbol{M} \in \mathbb{C}^{n\times n}$ is defined as the ratio of the largest eigenvalue of $\boldsymbol{M}$ to the smallest.

2.2. Column Nyström approximation. Suppose we are given a positive definite matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ with rapidly decreasing eigenvalues. Then, we can can consider a factored approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^*, \quad \text{where } \boldsymbol{B} \in \mathbb{C}^{n\times r} \text{ and } \boldsymbol{D} \in \mathbb{C}^{r\times r} \text{ with } r \ll n.$$

This is called a low-rank approximation, because $\operatorname{rank}(\boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^*) \leq r$. If such a low-rank approximation exists and we can find it quickly, then we can replace $\boldsymbol{A}$ by the low-rank approximation $\hat{\boldsymbol{A}}$ to speed up linear algebra operations. For example, we can quickly multiply $\hat{\boldsymbol{A}}$ by a vector in a sequence of three multiplications using each matrix in the factorization. We can also efficiently compute the shifted inverse $(\hat{\boldsymbol{A}} + \lambda\mathbf{I})^{-1}$ for $\lambda > 0$ by using the Woodbury formula.

A popular and useful low-rank approximation is the *Nyström approximation*:

$$\boldsymbol{A}\langle\boldsymbol{X}\rangle := \boldsymbol{A}\boldsymbol{X}(\boldsymbol{X}^*\boldsymbol{A}\boldsymbol{X})^{-1}\boldsymbol{X}^*\boldsymbol{A}. \tag{2.1}$$

The Nyström approximation can be defined with respect to any test matrix $\boldsymbol{X} \in \mathbb{C}^{n\times r}$, and it has rank at most r. The approximation (2.1) can be justified as the matrix $\boldsymbol{H} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^*\boldsymbol{A}$ that is closest to $\boldsymbol{A}$ in the spectral norm while maintaining a positive semidefinite residual $\boldsymbol{A} - \boldsymbol{H}$.

The Nyström approximation is defined for any $\boldsymbol{X} \in \mathbb{C}^{n\times r}$, so how should we choose $\boldsymbol{X}$? In an ideal world, there is no better choice than making the columns the dominant r eigenvectors of $\boldsymbol{A}$, since computing $\boldsymbol{A}\langle \boldsymbol{X}\rangle$ via (2.1) then recovers the r-truncated eigendecomposition. In practice, running a few iterations of an iterative eigenvector-finding method can give a high-accuracy Nyström approximation when the eigenvalues decay quickly; see [21]. However, this approach is too expensive for many computational settings, since it requires $\mathcal{O}(n^2 r)$ operations to execute.

A more efficient Nyström approximation is generated by the matrix $\boldsymbol{X} = \mathbf{I}(:, R)$, which is a block of columns from the identity indexed by a cardinality-r index set $R \subseteq \{1, \dots, n\}$, often called a *pivot set*. Then, the Nyström approximation takes a simplified form:

$$\boldsymbol{A}\langle R\rangle = \boldsymbol{A}\langle \mathbf{I}(:, R)\rangle = \boldsymbol{A}_{:,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,:}. \tag{2.2}$$

We call (2.2) the *column Nyström* approximation, and it can be computed in just $\mathcal{O}(nr^2)$ operations by first forming and factoring the matrix $\boldsymbol{A}_{R,R}$ and then incorporating by the matrix $\boldsymbol{A}_{:,R}$. This approach does not require reading all entries of the matrix!

The column Nyström approximation can be represented as a block matrix approximation, after a suitable reordering of the indices. Given an index set $R = \{k_1, \dots, k_r\}$, we introduce the permutation matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ that satisfies $\boldsymbol{P}\boldsymbol{e_i} = \boldsymbol{e_{k_i}}$ for each $1 \le i \le r$. Then the column Nyström approximation of $\boldsymbol{A}$ can be represented as

$$\boldsymbol{P}^*\boldsymbol{A}\langle R\rangle\boldsymbol{P} = \begin{bmatrix} \boldsymbol{A}_{R,R} & \boldsymbol{A}_{R,R^c} \\ \boldsymbol{A}_{R^c,R} & \boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R^c} \end{bmatrix}.$$

The residual matrix, which is called the Schur complement, can be represented as

$$\boldsymbol{P}^*(\boldsymbol{A}/R)\boldsymbol{P} = \boldsymbol{P}^*(\boldsymbol{A} - \boldsymbol{A}\langle R\rangle)\boldsymbol{P} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_{R^c,R^c} - \boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R^c,R} \end{bmatrix}$$

The index reordering makes clear that the selected rows and columns are represented exactly, while the remaining rows and columns are interpolated by a linear combination of the selected columns.

2.3. Vecchia approximation. An alternative way to approximate a positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ or $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is based on approximating the distribution of a real- or complex-valued random Gaussian vector $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{A})$ [17]. If we introduce an exact inverse Cholesky factorization $\boldsymbol{A}^{-1} = \boldsymbol{C}\boldsymbol{C}^*$, then we observe $\boldsymbol{z} = \boldsymbol{C}^*\boldsymbol{y}$ is a white noise vector, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, so the vectors $\boldsymbol{z}$ and $\boldsymbol{y}$ are linked by the regression formula

$$\boldsymbol{z}_i = \sum_{j=i}^{n} \overline{\boldsymbol{C}_{ji}}\boldsymbol{y}_j \quad \text{and consequently} \quad \boldsymbol{y}_i = -\sum_{j=i+1}^{n} \frac{\overline{\boldsymbol{C}_{ji}}}{\boldsymbol{C}_{ii}}\boldsymbol{y}_j + \frac{\boldsymbol{z}_i}{\boldsymbol{C}_{ii}}. \tag{2.3}$$

By this formula, when we linearly regress $\boldsymbol{y}_i$ on $\boldsymbol{y}_{i+1}, \dots, \boldsymbol{y}_n$, the coefficients are $-\overline{\boldsymbol{C}_{ji}}/\boldsymbol{C}_{ii}$ and the residual is $\mathcal{N}(0, 1/\boldsymbol{C}_{ii}^2)$.

Next, we observe that many covariance matrices arising in applications have entries $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ that satisfy the conditional independence relationship

$$\boldsymbol{y}_i \perp \boldsymbol{y}_j \mid \boldsymbol{y}_{j+1} \dots \boldsymbol{y}_n. \tag{2.4}$$

By the regression formula (2.3), the conditional independence relationship is equivalent to a zero entry in the Cholesky factor $\boldsymbol{C}_{ji} = 0$. Given the tendency for many

conditional independence relationships to hold simultaneously, we consider an approximation $\hat{\boldsymbol{A}}^{-1} = \hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^*$ where the approximate Cholesky factor is sparse

$$\hat{\boldsymbol{C}}_{ij} = 0, \quad \text{if } j \notin S_i,$$

and the sparsity pattern is determined by index sets $\{S_i\}_{i=1}^n$ which contain a small, bounded number of elements subject to $\{i\} \subseteq S_i \subseteq \{i, \dots, n\}$.

Taking one step further, we can explicitly approximate the Gaussian distribution $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{A})$ using the sparsity pattern $\{S_i\}_{i=1}^n$. Using the chain rule of conditional probabilities, the *exact* Gaussian density function p is

$$p(\boldsymbol{y}) = \prod_{i=1}^{n} p_{i \,|\, \{i+1,\dots,n\}}(\boldsymbol{y}_i \,|\, \boldsymbol{y}_{\{i+1,\dots,n\}}),$$

where $p_{i \,|\, \{i+1,\dots,n\}}$ is the conditional density of $\boldsymbol{y}_i$ given variables $\boldsymbol{y}_{i+1}, \dots, \boldsymbol{y}_n$. If we only consider the conditional dependencies specified in the sparsity pattern $\{S_i\}_{i=1}^n$, then the Gaussian density is naturally approximated as

$$\hat{p}(\boldsymbol{y}) = \prod_{i=1}^{n} p_{i \,|\, S_i \setminus \{i\}}(\boldsymbol{y}_i \,|\, \boldsymbol{y}_{S_i \setminus \{i\}}). \tag{2.5}$$

Each conditional density $p_{i \,|\, S_i \setminus \{i\}}(\boldsymbol{y}_i \,|\, \boldsymbol{y}_{S_i \setminus \{i\}})$ is representing a Gaussian conditional distribution, which can be written as

$$\boldsymbol{y}_i \sim \mathcal{N}\left(-\frac{\boldsymbol{e}_1^*(\boldsymbol{A}_{S_i,S_i})^{-1}}{\boldsymbol{e}_1^*(\boldsymbol{A}_{S_i,S_i})^{-1}\boldsymbol{e}_1} \begin{bmatrix} 0 \\ \boldsymbol{y}_{S_i \setminus \{i\}} \end{bmatrix}, \frac{1}{\boldsymbol{e}_1^*(\boldsymbol{A}_{S_i,S_i})^{-1}\boldsymbol{e}_1} \right).$$

Taken as a whole, the density $\hat{p}$ is representing a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, (\hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^*)^{-1})$, where the inverse Cholesky factor has a closed-form, computationally tractable expression:

$$\hat{\boldsymbol{C}}_{S_i,i} = \frac{(\boldsymbol{A}_{S_i,S_i})^{-1}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^*(\boldsymbol{A}_{S_i,S_i})^{-1}\boldsymbol{e}_1}} \quad \text{and} \quad \hat{\boldsymbol{C}}_{S_i^c,i} = \boldsymbol{0}, \quad \text{for each } i = 1, \dots, n. \tag{2.6}$$

The approximation (2.6) is known as the Vecchia approximation in spatial statistics [22].

The Vecchia approximation has several optimality properties among approximations $\hat{\boldsymbol{A}}^{-1} = \boldsymbol{C}\boldsymbol{C}^*$ with a fixed sparsity pattern:

1. In 1990, Kaporin [12] discovered that the Vecchia approximation optimizes the ratio of the trace to the normalized determinant

$$\frac{\operatorname{tr}\big(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^*\big)}{\det\big(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^*\big)^{1/n}}. \tag{2.7}$$

2. Later, Yeremin et al. [23] found that the Vecchia approximation minimizes the Frobenius norm of the residual

$$\|\boldsymbol{I} - \boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^*\|_{\mathrm{F}} \tag{2.8}$$

subject to the constraint $\operatorname{diag}(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^T) = \boldsymbol{1}$.

3. Most recently, Schäfer et al. [18] observed that the Vecchia approximation minimizes the KL divergence

$$D_{\mathrm{KL}}\left(\mathcal{N}(0,\boldsymbol{A})\|\,\mathcal{N}(0,(\boldsymbol{C}\boldsymbol{C}^*)^{-1})\right). \tag{2.9}$$

We also note two desirable computational properties of the Vecchia formula (2.6). First, this formula does not use all entries of $\boldsymbol{A}$, only blocks of entries corresponding to each column's nonzero set. Second, as a result, we can compute the nonzero entries of $\hat{\boldsymbol{C}}$ in only $\mathcal{O}(nr^3)$ operations if $|S_i| \leq r$ for each index set i.

Finally, we notice that the stipulation $S_i \subseteq \{i, i+1 \dots n\}$ depends heavily on the ordering of the indices. Therefore, we consider a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ that puts the important indices last, and then we apply Vecchia approximation of the permuted matrix

$$\boldsymbol{M} = \boldsymbol{P}^*\boldsymbol{A}\boldsymbol{P}.$$

We obtain an approximation $\hat{\boldsymbol{M}}^{-1} = \hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^*$, which transfers to an approximation of the original matrix via

$$\hat{\boldsymbol{A}}^{-1} = \boldsymbol{P}\hat{\boldsymbol{M}}^{-1}\boldsymbol{P}^* = (\boldsymbol{P}\hat{\boldsymbol{C}}\boldsymbol{P}^*)(\boldsymbol{P}\hat{\boldsymbol{C}}\boldsymbol{P}^*)^*. \tag{2.10}$$

We call the approximation (2.10) a generalized Vecchia approximation.

2.4. Column Nyström plus Vecchia. Last, we consider an approach that combines low-rank and sparse structure when approximating a positive definite matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$.

In this approach, we start by generating a column Nyström approximation using an index set $R = \{k_1, \dots, k_r\}$:

$$\boldsymbol{A}\langle R\rangle = \boldsymbol{A}_{:,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,:}.$$

Following the exposition in subsection 2.2, the structure of the Nyström approximation become clearer when we introduce a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ that puts the R indices last: $\boldsymbol{P}\boldsymbol{e}_{\boldsymbol{n-r+i}} = \boldsymbol{e}_{\boldsymbol{k_i}}$ for each $i = 1, \dots, r$. The Nyström approximation can then be written as

$$\boldsymbol{P}^*\boldsymbol{A}\langle R\rangle\boldsymbol{P} = \begin{bmatrix} \boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R_c} & \boldsymbol{A}_{R^c,R} \\ \boldsymbol{A}_{R,R^c} & \boldsymbol{A}_{R,R} \end{bmatrix}.$$

and the Schur complement is

$$\boldsymbol{P}^*(\boldsymbol{A}/R)\boldsymbol{P} = \begin{bmatrix} \boldsymbol{A}_{R^c,R^c} - \boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R_c} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}.$$

The residual from this approximation is a positive-definite residual matrix $\boldsymbol{B} = \boldsymbol{A}_{R^c,R^c} - \boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R_c}$, which we can systematically improve.

To improve our approximation, we apply sparse Vecchia approximation to $\boldsymbol{B}$:

$$\hat{\boldsymbol{B}}^{-1} = \boldsymbol{F}\boldsymbol{F}^*.$$

Combining the Nyström and sparse approximations then gives

$$\boldsymbol{P}^*\hat{\boldsymbol{A}}\boldsymbol{P} = \begin{bmatrix} \boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R_c} + \boldsymbol{F}^{-*}\boldsymbol{F}^{-1} & \boldsymbol{A}_{R^c,R} \\ \boldsymbol{A}_{R,R^c} & \boldsymbol{A}_{R,R} \end{bmatrix}. \tag{2.11}$$

This approximation was discovered by [24], who refer to their implementation as "adaptive Factorized Nyström" due to an adaptive determination of the rank of the Nyström part. However, for the rest of paper, we refer to the general class of approximations based on (2.11) as "column Nyström plus Vecchia" (CNV) approximations.

Next, we make a new observation that the *inverse* of a CNV approximation can be written explicitly as

$$\begin{aligned}(2.12)\quad &\boldsymbol{P}^*\hat{\boldsymbol{A}}^{-1}\boldsymbol{P} \\ &= \begin{bmatrix} \boldsymbol{F}\boldsymbol{F}^* & -\boldsymbol{F}\boldsymbol{F}^*\boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1} \\ -(\boldsymbol{A}_{R,R}^{-1})\boldsymbol{A}_{R,R^c}\boldsymbol{F}\boldsymbol{F}^* & (\boldsymbol{A}_{R,R})^{-1} + (\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R^c}\boldsymbol{F}\boldsymbol{F}^*\boldsymbol{A}_{R^c,R}(\boldsymbol{A}_{R,R})^{-1} \end{bmatrix}\end{aligned}$$

This formula is a consequence of the block matrix inversion formula [3, p420], together with the fact that

$$\left(\hat{\boldsymbol{A}}_{R^c,R^c} - \hat{\boldsymbol{A}}_{R^c,R}(\hat{\boldsymbol{A}}_{R,R})^{-1}\hat{\boldsymbol{A}}_{R,R^c}\right)^{-1} = \boldsymbol{F}\boldsymbol{F}^*.$$

Equation (2.12) can be simplified further, by writing the approximation in terms of its Cholesky decomposition:

$$(2.13)\quad \boldsymbol{P}^*\hat{\boldsymbol{A}}^{-1}\boldsymbol{P} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{0} \\ -(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R^c}\boldsymbol{F} & \boldsymbol{C} \end{bmatrix}\begin{bmatrix} \boldsymbol{F} & \boldsymbol{0} \\ -(\boldsymbol{A}_{R,R})^{-1}\boldsymbol{A}_{R,R^c}\boldsymbol{F} & \boldsymbol{C} \end{bmatrix}^*, \quad \text{where } (\boldsymbol{A}_{R,R})^{-1} = \boldsymbol{F}\boldsymbol{F}^*.$$

Equation (2.13) gives an intuitive formula for the approximate Cholesky factor that can be evaluated in $\mathcal{O}(r^3 + r\|\boldsymbol{C}\|_0)$ operations, where $\|\boldsymbol{C}\|_0$ denotes the number of nonzero entries in $\boldsymbol{C}$.

3. Column Nyström plus Vecchia approximations. This section extends the theory of Vecchia approximation to show that column-Nyström-plus-Vecchia approximations are themselves Vecchia approximations in disguise.

Our main theoretical result is the following:

THEOREM 3.1 (CNV = Vecchia). *Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ be a positive definite matrix and partition the indices $\{1,\dots,n\}$ into sets $U = \{1,\dots,n-r\}$ and $V = \{n-r+1,\dots,n\}$. Consider the column Nyström plus Vecchia approximation*

$$(3.1)\qquad \hat{\boldsymbol{A}}^{-1} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \\ -(\boldsymbol{A}_{V,V})^{-1}\boldsymbol{A}_{V,U}\boldsymbol{C} & \boldsymbol{F} \end{bmatrix}\begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \\ -(\boldsymbol{A}_{V,V})^{-1}\boldsymbol{A}_{V,U}\boldsymbol{C} & \boldsymbol{F} \end{bmatrix}^*,$$

that incorporates the following components:

1. *$\boldsymbol{A}_{V,V}^{-1} = \boldsymbol{F}\boldsymbol{F}^*$ is a dense Cholesky factorization of $\boldsymbol{A}_{V,V}^{-1}$.*
2. *$\hat{\boldsymbol{B}}^{-1} = \boldsymbol{C}\boldsymbol{C}^*$ is a Vecchia approximation of $\boldsymbol{B} = \boldsymbol{A}_{U,U} - \boldsymbol{A}_{U,V}(\boldsymbol{A}_{V,V})^{-1}\boldsymbol{A}_{V,U}$ with sparsity pattern $\{S_i\}_{i=1}^n$.*

Then $\hat{\boldsymbol{A}}$ is the Vecchia approximation of $\boldsymbol{A}$ with sparsity pattern $T_i = S_i \cup V$ for $i = 1,\dots,n-r$ and $T_i = \{i, i+1,\dots,n\}$ for $i = n-r+1,\dots,n$.

Proof. Columns $i = n-r+1,\dots,n$ of the approximate Cholesky factor in (3.1) are generated via dense Cholesky factorization, which is the same as a Vecchia approximation with a dense sparsity pattern $T_i = \{i, i+1,\dots,n\}$.

Next consider columns $i = 1,\dots,n-r$ of the approximate Cholesky factor in (3.1). Using the definition (2.6) of the sparse Vecchia approximation $\hat{\boldsymbol{B}}^{-1} = \boldsymbol{C}\boldsymbol{C}^*$,

each column i satisfies

$$\begin{bmatrix} \mathbf{I} \\ -(\boldsymbol{A}_{V,V})^{-1}\boldsymbol{A}_{V,U} \end{bmatrix} \boldsymbol{C}_{:,i}, \quad \text{where } \boldsymbol{C} = \frac{(\boldsymbol{B}_{S_i,S_i})^{-1}\boldsymbol{e_1}}{\boldsymbol{e_1^*}(\boldsymbol{B}_{S_i,S_i})^{-1}\boldsymbol{e_1}}$$

Next, using the block matrix inversion formula [3, p420], we observe

$$\begin{bmatrix} \mathbf{I} \\ -(\boldsymbol{A}_{V,V})^{-1}\boldsymbol{A}_{V,U} \end{bmatrix} \frac{(\boldsymbol{B}_{S_i,S_i})^{-1}\boldsymbol{e_1}}{\boldsymbol{e_1^*}(\boldsymbol{B}_{S_i,S_i})^{-1}\boldsymbol{e_1}} = \frac{(\boldsymbol{A}_{S_i\cup V,S_i\cup V})^{-1}\boldsymbol{e_1}}{\sqrt{\boldsymbol{e_1}(\boldsymbol{A}_{S_i\cup V,S_i\cup V})^{-1}\boldsymbol{e_1}}}.$$

This is precisely the formula (2.6) for the ith column in a Vecchia approximation of $\boldsymbol{A}$ with sparsity pattern $T_i = S_i \cup V$. ∎

Theorem 3.1 has several ramifications. Most immediately, the Kaporin, KL, and Frobenius optimality results from [12, 18, 23] carry over to CNV approximations. Also, the theorem implies that for fixed approximation rank r, the choice of column Nyström approximation is equivalent to the choice of the last r indices in the ordering. Two existing methods with the same index ordering are "adaptive factorized Nyström" approximation [24] and "sparse Cholesky approximation" [18]: these techniques produce the same approximation when the bottom r rows are included to the sparsity pattern.

4. Application: Gaussian Processes. The results in section 3 apply to all positive-definite matrices, but one application which merits further elaboration is covariance matrices in Gaussian processes (GPs). A Gaussian process is a multivariate Gaussian distribution whose dimensions are $\mathbb{R}^d$, i.e., with mean $\boldsymbol{m} \in \mathbb{R}^{\mathbb{R}^d}$ and covariance matrix $\bar{\boldsymbol{K}} \in \mathbb{R}^{\mathbb{R}^d \times \mathbb{R}^d}$. Typically, $\bar{\boldsymbol{K}}$ is specified by setting $\bar{\boldsymbol{K}}_{\boldsymbol{x},\boldsymbol{y}} = \kappa(\boldsymbol{x},\boldsymbol{y})$ or $\boldsymbol{K}_{\boldsymbol{x},\boldsymbol{y}} = \kappa(\boldsymbol{x},\boldsymbol{y}) + \mu\delta_{\boldsymbol{x},\boldsymbol{y}}$, for κ a *kernel function*, or a function with $\kappa(\boldsymbol{x},\boldsymbol{x}) = 1$ for which $\kappa(\boldsymbol{x},\boldsymbol{y})$ decreases in $\|x - y\|$ for some norm, and $\mu \in \mathbb{R}_{\geq 0}$ a *nugget parameter*. A random variable conforming to such a distribution can be thought of as a random function $f : \mathbb{R}^d \to \mathbb{R}$ such that the covariance of $f(\boldsymbol{x})$ and $f(\boldsymbol{y})$ is given by $\bar{\boldsymbol{K}}_{\boldsymbol{x},\boldsymbol{y}}$. Since estimating and subtracting off the mean is usually not a problem in applications, we will take $\boldsymbol{m} = \mathbf{0}$ in this paper.

When working with GPs, we often work in terms of a realization of a GP at some finite number n points $\{\boldsymbol{x}^{(i)}\}_{i=1}^n \subseteq \mathbb{R}^d$. When we're only concerned with these points, $\bar{\boldsymbol{K}}$ behaves like a matrix in $\mathbb{R}^{n\times n}$; we thus often define some matrix $\boldsymbol{K}$ such that $\boldsymbol{K}_{i,j}$ refers to $\bar{\boldsymbol{K}}_{\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}}$. If n is large, practitioners must often work with approximations of $\boldsymbol{K}$ over these n points with advantageous computational properties for improved computational speed at the cost of some approximation error. In this section, we apply CNV to this task. Specifically, we discuss the compatibility of existing algorithms for choosing pivot sets and sparsity patterns with CNV in the context of GPs. We also propose our own pivot-choosing algorithm in subsection 4.1.3.

4.1. Matérn Kernels. We begin by examining existing methods for finding sparse Vecchia approximations of covariance matrices corresponding to a particular class of kernels, the Matérn kernels. Later in this section, we will show that these methods are compatible with CNV approximations and adapt their theoretical bounds to the CNV case, as well as propose heuristics based on these for choosing pivots during the Nyström part of CNV-type approximations.

The Matérn kernel functions take the form

$$\kappa_{\nu,\rho,\sigma}(\boldsymbol{x},\boldsymbol{x'}) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\rho}\right)^{\nu} B_\nu \left(\sqrt{2\nu}\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\rho}\right)$$

where B_ν is the modified Bessel function of the second kind, Γ is the gamma function, ν is called a smoothness parameter, and $\rho, \sigma > 0$ are other parameters. For $\nu \in \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2} \ldots\} = \mathbb{N} + \frac{1}{2}$, this can be written

$$\kappa_{\nu,\rho,\sigma}(\boldsymbol{x},\boldsymbol{x'}) =$$
$$\sigma^2 \exp\left(-\frac{\sqrt{2\nu}\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\rho}\right) \frac{(\nu-\frac{1}{2})!}{(2\nu-1)!} \sum_{i=0}^{\nu-\frac{1}{2}} \frac{(\nu-\frac{1}{2}+i)!}{i!(\nu-\frac{1}{2}-i)!} \left(\frac{2\sqrt{2\nu}\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\rho}\right)^{\nu-\frac{1}{2}-i}.$$

4.1.1. Reverse Maximin Ordering and Theoretical Sparse Vecchia Error Bounds. Let κ be a Matérn kernel function on $\mathbb{R}^d$, let $\bar{\boldsymbol{K}} \in \mathbb{R}^{\mathbb{R}^d \times \mathbb{R}^d}$ be the corresponding covariance matrix, and let $\boldsymbol{K}$ be the covariance matrix it induces given $X = \{\boldsymbol{x}^{(i)}\}_{i=1}^n$ according to $\boldsymbol{K}_{i,j} = \kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$. Additionally, let $Z \subseteq \mathbb{R}^d$ (possibly empty), which in this context can be thought of as an index set. Define

$$\boldsymbol{K}/Z := (\bar{\boldsymbol{K}}/Z)_{XX} = \bar{\boldsymbol{K}}_{XX} - \bar{\boldsymbol{K}}_{XZ}\bar{\boldsymbol{K}}_{ZZ}^{-1}\bar{\boldsymbol{K}}_{ZX}.$$

Later in this subsection, we will state a result of [18] regarding the sparsity of the Cholesky factor of $(\boldsymbol{K}/Z)^{-1}$ that gives us reason to believe the Cholesky factor of $\boldsymbol{K}^{-1}$ is sparse, too. But before we do, we recall from subsection 2.3 that if we are to find a sparse Vecchia approximation of $\boldsymbol{K}$, the ordering of the points $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$ matters significantly to the quality of our approximation. When $d = 1$, ordering the points according to the usual ordering on $\mathbb{R}$, or its reverse, tend to work well in practice. However, methods of this type (e.g. lexicographically coordinate-by-coordinate in some basis) tend to perform poorly in $\mathbb{R}^d$ ([10]).

An alternative which [10] shows performs well empirically, and which [18] require a slightly weaker version of in their aforementioned result, is known as the reverse-maximin ordering. It follows a procedure known as Farthest Point Sampling (FPS) to choose the next-to-last point in the ordering at any given step. Mathematically, assuming $\{k_j\}_{j=i+1}^n$ in the reverse maximin ordering have been determined, and for a given Z as above, one finds

$$k_i = \operatorname*{argmax}_{k \notin \{k_j | j=i+1 \ldots n\}} \min_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i+1)} \ldots \boldsymbol{x}^{(n)}\} \cup Z} \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k_j)}\|_2 \tag{4.1}$$

for the next point in the ordering. Procedurally, a naïve algorithm for construction such an ordering is given in Algorithm 4.1.

This algorithm takes $O(n(n+|Z|)d)$ operations; whether this would be a computational bottleneck generally depends on d and one's application. If a faster algorithm is needed and d is not too large, [19]'s Algorithm 4.1 yields a reverse-maximin ordering in $O(2^d dn \log^2 n + n|Z|d)$ operations.

We can now state a slightly weaker version of the result of [18]:

THEOREM 4.1 ([18], Theorem 3.4). *Let $\Omega \subseteq \mathbb{R}^d$ be a bounded domain with boundary $\partial\Omega$, and let $\{\boldsymbol{x}^{(i)}\}_{i=1}^n \subseteq \Omega$ be ordered according to the reverse-maximin ordering. Also, let*

$$\delta = \frac{\min_{i \in [n]} \min_{\boldsymbol{x} \in \{\boldsymbol{x}^{(j)}\}_{j=1}^n \cup \partial\Omega \setminus \boldsymbol{x}^{(i)}} \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|_2}{\max_{\boldsymbol{x} \in \Omega} \min_{\boldsymbol{x'} \in \{\boldsymbol{x}^{(i)}\}_{i=1}^n \cup \partial\Omega} \|\boldsymbol{x} - \boldsymbol{x'}\|_2}, \tag{4.2}$$

Algorithm 4.1 Construction of Reverse-Maximin Ordering

```
for z ∈ Z do
    for i ∈ [n] do
        d_i ← min(d_i, ‖x^(i) − z‖_2)
    end for
end for
for i = n ... 1 do
    k_i ← arg max_k d_k
    for j = 1 ... n do
        d_j ← min(d_j, ‖x^(j) − x^(k_i)‖_2)
    end for
end for
```

and suppose $\boldsymbol{K}$ *is the Matérn covariance matrix at* $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$ *for* $\kappa_{\nu,\rho,\sigma}$ *with* $\nu \in \mathbb{N} + \frac{1}{2}$. *Further suppose* S_i *is given by the indices of the* $O((\log(\frac{n}{\varepsilon}))^d)$*-nearest neighbors of* $\boldsymbol{x}^{(i)}$ *which satisfy lower triangularity along with* $\boldsymbol{x}^{(i)}$ *itself, with implicit constant depending only on* $\delta, \nu, \rho, \sigma, \Omega$. *Then if* $\boldsymbol{C}$ *is the sparse Vecchia approximation of* $\boldsymbol{K}/\partial\Omega$ *with sparsity pattern* $\{S_i\}_{i=1}^n$ *as given by (2.6), we have*

$$D_{KL}\left(\mathcal{N}(\boldsymbol{0}, \boldsymbol{K}/\partial\Omega) \,\|\, \mathcal{N}(\boldsymbol{0}, (\boldsymbol{C}\boldsymbol{C}^*)^{-1})\right) + \|(\boldsymbol{K}/\partial\Omega) - (\boldsymbol{C}\boldsymbol{C}^*)^{-1}\|_{FRO} < \varepsilon \tag{4.3}$$

We remark that while Theorem 4.1 requires conditioning on the boundary of a bounded domain containing $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$, [18] observed that this did not substantially affect empirical results in settings that did not condition on the boundary. We also remark that this result does not hold with nonzero nugget, and indeed various authors ([13, 14]) observe that the sparsity of $\boldsymbol{C}$ decreases as μ the nugget parameter increases.

4.1.2. Adapting Nearest Neighbors to CNV case. Now, we adapt Theorem 4.1 from the sparse Vecchia case to the CNV case.

Before proceeding more generally, we first consider the case of CNV approximations whose Nyström pivot set is simply the last r indices in the reverse-maximin ordering. Let $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$, $\boldsymbol{K}$, Ω, and the sparsity pattern $\{S_i\}_{i=1}^n$ as in Theorem 4.1, the latter for error tolerance $\varepsilon > 0$. Here, Theorem 3.1 implies that the sparse Vecchia approximation with sparsity pattern $S_i \cup \{\max(i, n-r+1) \dots n\}$ equals the CNV approximation with r pivots and sparsity pattern $\{S_i\}_{i=1}^n$ because the index ordering is the same. By the optimality of the Vecchia approximation in the KL divergence loss (2.9) and the Frobenius loss (2.8) [1], we cannot make our approximation worse in these objectives by adding entries to the sparsity pattern. So by Theorem 4.1, which we crucially can only apply because the ordering is still reverse-maximin, this CNV approximation has error no worse than (4.3).

Though [24] did not know of Theorem 3.1 nor this result, this provides considerable justification for their proposal to use farthest point sampling (FPS) to select Nyström pivots. However, the literature on choosing Nyström pivots is very extensive, and we might hope to use other algorithms than FPS for this purpose. Motivated by this, we extend the bounds of Theorem 4.1 to other choices for pivot sets:

THEOREM 4.2. *Let* $\{\boldsymbol{x}^{(i)}\}$, Ω, $\boldsymbol{K}$, ε *as in the setting of Theorem 4.1, let* $R \subseteq [n]$ *of size* r. *Let* $\{k_i\}_{i=1}^n$ *an index reordering such that* $\{k_i\}_{i=n-r+1}^n = R$ *and* $\{k_i\}_{i=1}^{n-r}$

[1] Actually, the Frobenius losses in (2.8) and (4.3) are not exactly of the same form, though the bound in Theorem 4.1 remains valid due to [19]'s Lemma B.8.

contains the rest of the indices according to the positions of their respective $\boldsymbol{x}^{(k_i)}$*s in the reverse maximin ordering initialized with* $\partial\Omega \cup \{\boldsymbol{x}^{(R)}\}$*. Also, let* $\boldsymbol{P}$ *be the* $n \times n$ *permutation matrix such that* $\boldsymbol{P}\boldsymbol{e_1} = \boldsymbol{e_{k_i}}$*. Finally, suppose that for* $i \notin R$*, we have that* S_i *is given by the indices of the* $O((\log(\frac{n}{\varepsilon}))^d)$*-nearest neighbors of* $\boldsymbol{x}^{(i)}$ *among* $\{\boldsymbol{x}^{R^C \cap \{k_j\}_{j=1}^n)}\}$*, along with* $\boldsymbol{x}^{(i)}$ *itself, with implicit constant depending only on* $\delta, \nu, \rho, \sigma, \Omega$*,* as well as $\{\boldsymbol{x}^{(R)}\}$ the choice of pivots. *Then for* $(\boldsymbol{CC}^*)$ *the CNV approximation of* $\boldsymbol{P}^*(\boldsymbol{K}/\partial\Omega)\boldsymbol{P}$*, we have*

(4.4)
$$D_{KL}\left(\mathcal{N}(\boldsymbol{0}, \boldsymbol{P}^*(\boldsymbol{K}/\partial\Omega)\boldsymbol{P}) \,\|\, \mathcal{N}(\boldsymbol{0}, (\boldsymbol{CC}^*)^{-1})\right) + \|\boldsymbol{P}^*(\boldsymbol{K}/\partial\Omega)\boldsymbol{P} - (\boldsymbol{CC}^*)^{-1}\|_{FRO} < \varepsilon$$

Proof. We'll start by showing that $((\boldsymbol{K}/\partial\Omega)/R)^{-1}$, the inverse column Nyström residual of $\boldsymbol{K}/\partial\Omega$ for some index set $R \subseteq [n]$, is also well-approximated by a matrix with $\mathcal{O}((\log(\frac{n}{\varepsilon}))^d)$ entries per column in its Cholesky factor.

We first observe that

$$\begin{aligned}(\boldsymbol{K}/\partial\Omega)/R &= (\bar{\boldsymbol{K}}_{X\cup\partial\Omega, X\cup\partial\Omega} - \bar{\boldsymbol{K}}_{X\cup\partial\Omega,\partial\Omega}\bar{\boldsymbol{K}}^{-1}_{\partial\Omega,\partial\Omega}\bar{\boldsymbol{K}}_{\partial\Omega, X\cup\partial\Omega})_{X,X}/R \\ &= ((\bar{\boldsymbol{K}}_{X\cup\partial\Omega, X\cup\partial\Omega} - \bar{\boldsymbol{K}}_{X\cup\partial\Omega,\partial\Omega}\bar{\boldsymbol{K}}^{-1}_{\partial\Omega,\partial\Omega}\bar{\boldsymbol{K}}_{\partial\Omega, X\cup\partial\Omega})/\{\boldsymbol{x}^{(R)}\})_{X,X} = (\boldsymbol{K}/\partial\Omega\cup\{\boldsymbol{x}^{(R)}\}).\end{aligned}$$

Motivated by this, define $\tilde{\Omega} = \Omega \setminus \{\boldsymbol{x}^{(R)}\}$ for any index set $R \subseteq [n]$, which is a bounded domain containing $\{\boldsymbol{x}^{([n]\setminus R)}\}$. Then $\partial\tilde{\Omega} = \partial\Omega \cup \{x^{(R)}\}$, and we may apply Theorem 4.1 to $\{\boldsymbol{x}^{[n]\setminus R}\}$ on $\tilde{\Omega}$.

As such, let $\{k_i\}_{i=1}^n$ and $\boldsymbol{P}$ be the index reordering and permutation matrix in the proposition, and let $\tilde{\boldsymbol{P}} = \boldsymbol{P}_{1:n-r,k_{1:n-r}}$. Then by Theorem 4.1, the Frobenius and KL divergence errors of the Vecchia approximation of $\tilde{\boldsymbol{P}}^*(\boldsymbol{K}/\partial\tilde{\Omega})\tilde{\boldsymbol{P}}$ decay exponentially in the number of nonzeros per column.

Finally, we demonstrate that this propagates to the entire CNV approximation. Letting $\boldsymbol{GG}^* \approx \tilde{\boldsymbol{P}}^*(\boldsymbol{K}/\partial\tilde{\Omega})\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{P}}^*(\boldsymbol{K}/\partial\Omega \cup \{\boldsymbol{x}^{(R)}\})\tilde{\boldsymbol{P}}$ the sparse Vecchia part from the last paragraph, we note that by (2.11) the residual of the entire CNV approximation of $\boldsymbol{P}^*(\boldsymbol{K}/\partial\Omega)\boldsymbol{P}$ with this sparse component takes form

$$\begin{aligned}&\begin{bmatrix}(\boldsymbol{K}/\partial\Omega)_{R^C,R^C} & (\boldsymbol{K}/\partial\Omega)_{R^C,R} \\ (\boldsymbol{K}/\partial\Omega)_{R,R^C} & (\boldsymbol{K}/\partial\Omega)_{R,R}\end{bmatrix} \\ &\quad - \begin{bmatrix}(\boldsymbol{K}/\partial\Omega)_{R^C,R}(\boldsymbol{K}/\partial\Omega)^{-1}_{R,R}(\boldsymbol{K}/\partial\Omega)_{R,R^C} + \boldsymbol{G}^{*-}\boldsymbol{G}^{-1} & (\boldsymbol{K}/\partial\Omega)_{R^C,R} \\ (\boldsymbol{K}/\partial\Omega)_{R,R^C} & (\boldsymbol{K}/\partial\Omega)_{R,R}\end{bmatrix} \\ &\qquad = \begin{bmatrix}\tilde{\boldsymbol{P}}^*(\boldsymbol{K}/\partial\Omega \cup \{\boldsymbol{x}^{(R)}\})\tilde{\boldsymbol{P}} + \boldsymbol{G}^{*-}\boldsymbol{G}^{-1} & 0 \\ 0 & 0\end{bmatrix}.\end{aligned}$$

We showed in the last paragraph that the Frobenius error of the upper right block decays exponentially, so it follows that the Frobenius error of the entire RHS does, too. For the KL divergence, this can also be shown directly, but we refer the reader to Lemma B.8 in [18] for brevity. ∎

4.1.3. Alternative pivot chooser: Approximate greedy KL minimization. Motivated by Theorem 4.2, we now propose an alternative to [24]'s FPS for choosing the indices in a column Nyström approximation. As theoretical motivation, Theorem 3.1 shows how the column Nyström part of a CNV approximation is equivalent to a Vecchia approximation. Moreover, a Vecchia approximation optimizes the KL divergence

$$D_{\mathrm{KL}}\left(\mathcal{N}(0, \boldsymbol{K}) \| \mathcal{N}(0, (\boldsymbol{CC}^*)^{-1})\right).$$

given a specific sparsity pattern. Therefore, it is natural to search for the column Nyström approximation that leads to the smallest KL divergence.

The KL divergence can be expanded according to

$$2D_{\mathrm{KL}}\left(\mathcal{N}(0,\boldsymbol{K}),\mathcal{N}(0,(\boldsymbol{C}\boldsymbol{C}^*)^{-1}\right) \tag{4.5}$$

$$= \mathrm{Tr}(\boldsymbol{C}^*\boldsymbol{K}\boldsymbol{C}) - n + \log\det(\boldsymbol{C}\boldsymbol{C}^*)^{-1} - \log\det\boldsymbol{K}. \tag{4.6}$$

Given a sparsity pattern S_i, the corresponding Vecchia approximation (2.6) that optimizes the KL divergence is given by

$$\boldsymbol{C}_{S_i,i} = \frac{(\boldsymbol{A}_{S_i,S_i})^{-1}\boldsymbol{e_1}}{\sqrt{\boldsymbol{e_1}^*(\boldsymbol{A}_{S_i,S_i})^{-1}\boldsymbol{e_1}}}.$$

This form of Cholesky factor $\boldsymbol{C}$ ensures that $\boldsymbol{C}^*\boldsymbol{K}\boldsymbol{C}$ has a diagonal of all ones. Hence, (4.6) reduces to

$$= \log\det(\boldsymbol{C}\boldsymbol{C}^*)^{-1} - \log\det\boldsymbol{K}, \tag{4.7}$$

which can be shown (see [11], appendix A.1) to equal

$$\sum\nolimits_{i=1}^{n}\Big[\log\big((\boldsymbol{K}/S_i\setminus\{i\})_{ii}\big) - \log\big((\boldsymbol{K}/\{i+1:n\})_{ii}\big)\Big]. \tag{4.8}$$

Next, we try to minimize the KL divergence with a "low rank plus diagonal" sparsity pattern, defined by the diagonal entries together with the bottom rows that indicate the low-rank component. Let $k_1,\dots,k_n$ be an arbitrary index reordering, let $\boldsymbol{P}$ be the permutation matrix with $\boldsymbol{P}\boldsymbol{e_i} = \boldsymbol{e_{k_i}}$, and let $\boldsymbol{M} = \boldsymbol{P}^*\boldsymbol{K}\boldsymbol{P}$. In particular, the k_ith row of $\boldsymbol{K}$ exactly equals the ith row of $\boldsymbol{M}$, so $(\boldsymbol{M}/J)_{i,i} = (\boldsymbol{K}/\{k_J\})_{k_i,k_i}$ for any $J\subseteq[n]$.

Suppose we are approximating $\boldsymbol{M}$ with a sparse Vecchia approximation, and that that the sparsity pattern currently consists of rows $n-j+1\dots n$ together with the diagonal. If we added the $n-j$th row to this sparsity pattern, the change in KL divergence would be given by

$$\begin{aligned}
&\sum_{i=1}^{n-j}\log\big((\boldsymbol{M}/\{n\dots n-j\}\setminus\{i\})_{i,i}\big) - \sum_{i=1}^{n-j}\log\big((\boldsymbol{M}/\{n\dots n-j+1\})_{i,i}\big)\\
&= \Big(\log\det\big((\boldsymbol{M}/\{n\dots n-j\})_{1:n-j-1,1:n-j-1}\big)\\
&\qquad\qquad + \log\big((\boldsymbol{M}/\{n\dots n-j+1\})_{n-j,n-j}\big)\Big)\\
&\qquad - \Big(\log\det\big((\boldsymbol{M}/\{n\dots n-j+1\})_{1:n-j,1:n-j}\big)\Big)\\
&= \log\det\big((\boldsymbol{M}/\{n\dots n-j\})_{1:n-j-1,1:n-j-1}\big)\\
&\qquad - \log\det\big((\boldsymbol{M}/\{n\dots n-j+1\})_{1:n-j-1,1:n-j-1}\big)
\end{aligned}$$

where the higher terms of the sums on the first line cancel. For the second and third lines, again see [11]'s appendix A.1. The above expression can be simplified by an application of the matrix determinant lemma to the first term according to [11], appendix B.5:

$$\begin{aligned}
&\log\big((\boldsymbol{M}/[n]\setminus n-j)_{n-j,n-j}\big) - \log\big((\boldsymbol{M}/\{n-j+1:n\})_{n-j,n-j}\big)\\
&\qquad = \log\big((\boldsymbol{K}/[n]\setminus k_{n-j})_{k_{n-j},k_{n-j}}\big) - \log\big((\boldsymbol{K}/\{k_{n-j+1:n}\})_{k_{n-j},k_{n-j}}\big)
\end{aligned} \tag{4.9}$$

This simplified expression leads to the consequence that the greedy optimization strategy can be highly successful.

THEOREM 4.3 (Greedy optimization strategy for Nyström approximation). *Fix a positive definite matrix $\boldsymbol{K} \in \mathbb{C}^{n\times n}$. For each index set I with arbitrary ordering $\prec$, let $(\boldsymbol{F}^{(I)})^{-*}(\boldsymbol{F}^{(I)})^{-1}$ denote the generalized Vecchia approximation of $\boldsymbol{K}$ with sparsity pattern $S_j = \{i \in I \mid i \prec j\} \cup \{j\}$ for each $j = 1, \dots, n$. Also, introduce the objective function*

$$f(I) := \frac{1}{2} D_{KL}\big[\mathcal{N}\big(0, \boldsymbol{K}\big) \,\|\, \mathcal{N}\big(0, (\boldsymbol{F}^{(I)})^{-*}(\boldsymbol{F}^{(I)})^{-1}\big)\,\big].$$

Then the greedy approximation strategy

$$I_0 \leftarrow \emptyset, \qquad I_t \leftarrow I_{t-1} \cup \{i_t\} \quad \text{for} \quad i_t \in \operatorname*{argmin}_{1\le i\le n} f(I_{t-1} \cup \{i\}),$$

results in exponential convergence to the best cardinality-m objective function value:

$$f(I_t) \le \min_{|J|=k} f(J) + \mathrm{e}^{-t/m}\Big[f(I_0) - \min_{|J|=m} f(J)\Big], \quad \text{for each } t = 0, 1, \dots$$

Proof. The proof technique is based on the classic idea of supermodular maximization, first developed by Nemhauser [16]. We will show f is supermodular, i.e.,

$$f(I \cup \{\ell\}) - f(I) \le f(I \cup J \cup \{\ell\}) - f(I \cup J), \tag{4.10}$$

for any disjoint index sets $I, J \subseteq \{1, \dots, n\}$ and any $\ell \in \{1, \dots, n\}$. Using (4.9), we have

$$\begin{aligned} f(I \cup \{\ell\}) - f(I) &= \log\big((\boldsymbol{K}/\{-\ell\})_{\ell\ell}\big) - \log\big((\boldsymbol{K}/I)_{\ell\ell}\big), \\ f(I \cup J \cup \{\ell\}) - f(I \cup J) &= \log\big((\boldsymbol{K}/\{-\ell\})_{\ell\ell}\big) - \log\big((\boldsymbol{K}/I \cup J)_{\ell\ell}\big). \end{aligned}$$

Subtracting the second line from the first line, we see

$$\begin{aligned} &[f(I \cup \{\ell\}) - f(I)] - [f(I \cup J \cup \{\ell\}) - f(I \cup J)] \\ &= \log\big((\boldsymbol{K}/I \cup J)_{\ell\ell}\big) - \log\big((\boldsymbol{K}/I)_{\ell\ell}\big). \end{aligned}$$

By taking the Schur complement with respect to the index set I first and with respect to the index set J second,

$$(\boldsymbol{K}/I \cup J)_{\ell\ell} = (\boldsymbol{K}/I)_{\ell\ell} - (\boldsymbol{K}/I)_{\ell J}[(\boldsymbol{K}/I)_{JJ}]^{-1}(\boldsymbol{K}/I)_{J\ell} \le (\boldsymbol{K}/I)_{\ell\ell},$$

where the inequality comes from the fact that $[(\boldsymbol{K}/I)_{JJ}]^{-1}$ is positive definite and $(\boldsymbol{K}/I)_{J\ell}$ is a vector. This establishes the supermodularity property (4.10).

Let $J^* = \{j_1, \dots, j_m\}$ be an optimal index set that solves $\min_{|J|=m} f(J)$. Then for each time index $t = 0, 1, \dots$, calculate

$$\begin{aligned} f(J^*) &\ge f(J^* \cup I_t) && (4.11)\\ &= f(I_t) + \sum\nolimits_{\ell=1}^{m} f(I_j \cup \{j_1, \dots, j_\ell\}) - f(I_t \cup \{j_1, \dots, j_{\ell-1}\}) && (4.12)\\ &\ge f(I_t) + \sum\nolimits_{\ell=1}^{m} f(I_t \cup \{j_\ell\}) - f(I_t) && (4.13)\\ &\ge f(I_t) + m(f(I_{t+1}) - f(I_t)), && (4.14) \end{aligned}$$

In this expression, (4.11) comes from supermodularity, (4.12) is a telescoping sum, (4.13) comes from another application of supermodularity, and (4.14) comes from the definition of the greedy optimization approach. By rearrangement,

$$f(I_{t+1}) - f(J^*) \leq \Big(1 - \frac{1}{m}\Big)[f(I_t) - f(J^*)], \quad \text{for each } t = 0, 1, \dots.$$

By induction, it follows

$$f(I_t) - f(J^*) \leq \left(1 - \frac{1}{m}\right)^t [f(I_0) - f(J^*)] \leq e^{-t/m}[f(I_0) - f(J^*)]$$

which can be rearranged into the desired result. ∎

To our knowledge, Theorem 4.3 is new. Krause et al. [15] studied a mutual information objective function, which leads to the same greedy optimization strategy based on (4.7). Yet the Kullback-Leibler divergence and mutual information are different functions. Our result establishes rigorous guarantees for optimization of the the Kullback-Leibler divergence, which is directly relevant to obtaining a good matrix approximation.

Unfortunately, there are barriers to a practical implementation of the greedy optimization strategy based on minimizing (4.9)

$$\log\big((\boldsymbol{K}/[n] \setminus i)_{i,i}\big) - \log\big((\boldsymbol{K}/\{k_{n-j+1:n}\})_{i,i}\big) = -\log(\boldsymbol{K}_{ii}^{-1}) - \log\big((\boldsymbol{K}/\{k_{n-j+1:n}\})_{i,i}\big)$$

We can keep track of the second term, which is the logarithm of the diagonal entry in the residual of our partial Nyström approximation. However, keeping track of the first term is not so easy. Our idea is to approximate $\operatorname{diag}(\boldsymbol{K}^{-1})$ or each $(\boldsymbol{K}/[n] \setminus i)_{ii}$ and use the approximation in the optimization. We call this approach Approximate Greedy KL Minimization (GKL). For a particular approximation $\boldsymbol{h}$ with $\boldsymbol{h}_i \approx \boldsymbol{K}_{ii}^{-1} = \frac{1}{(\boldsymbol{K}/[n]\setminus i)_{ii}}$, Algorithm 4.2 codifies such a routine in pseudocode.

Algorithm 4.2 Column Nyström via Greedy KL Minimization

$\boldsymbol{F} \leftarrow \boldsymbol{0} \in \mathbb{R}^{n \times r}$
$\boldsymbol{d} \leftarrow \operatorname{diag}(\boldsymbol{K}) \quad (= \boldsymbol{1})$ ▷ Diagonal of Nyström residual
for $i \in [r]$ **do**
  $k_{n-i+1} \leftarrow \arg\max_l \boldsymbol{d}_l \boldsymbol{h}_l$ ▷ $= \arg\min_l -\log(\boldsymbol{h}_l) - \log(\boldsymbol{d}_l)$
  $\boldsymbol{F}_{:,i} \leftarrow \boldsymbol{K}_{:,i}$
  $\boldsymbol{F}_{:,i} \leftarrow \boldsymbol{F}_{:,i} - \boldsymbol{F}_{:,1:i-1}(\boldsymbol{F}_{k_{n-i+1},1:i-1})^*$
  $\boldsymbol{F}_{:,i} \leftarrow \frac{\boldsymbol{F}_{:,i}}{\boldsymbol{F}_{k_{n-i+1},i}}$
  for $j \in [n]$ **do**
    $\boldsymbol{d}_j \leftarrow \boldsymbol{d}_j - |\boldsymbol{F}_{j,i}|^2$
  end for
end for
return $\{k_{n-r+1} \dots k_n\}, \boldsymbol{F}$ ▷ Pivot set, Nyström approximation's Cholesky factor

To approximate $\boldsymbol{K}_{ii}^{-1}$ for each i, we observe that if $\boldsymbol{C}\boldsymbol{C}^* \approx \boldsymbol{K}$, then $\boldsymbol{K}_{i,i}^{-1} \approx \boldsymbol{C}_{i,:}\boldsymbol{C}_{:,i}^* = \|\boldsymbol{C}_{i,:}\|_2^2$. For Matérn covariance matrices specifically, we already know from the last two sections that the pure-sparse Vecchia approximation based on order $\log n$ nearest neighbors per column is of high quality. Thus, we can produce such an

Algorithm 4.3 Inverse Diagonal Approximation by Vecchia Row Norm

$\{S_i\}_{i=1}^n \leftarrow$ `kNNSparsityPattern`$(\boldsymbol{K}, s)$ ▷ Or any other suitable sparsity pattern
$\boldsymbol{C} \leftarrow$ `SparseVecchia`$(\boldsymbol{K}, \{S_i\}_{i=1}^n)$
for $i \in [n]$ **do**
  $\boldsymbol{h}_i \leftarrow \|\boldsymbol{C}_{i,:}\|_2^2$
end for
return $\boldsymbol{h}$

approximation in order $n \log^{3d} n$ operations and use the square 2-norm of its ith row as an approximation of $K_{i,i}^{-1}$. See Algorithm 4.3 for pseudocode.

Alternatively, we could approximate each $(\boldsymbol{K}/[n] \setminus \{i\})_{ii}$ by $(\boldsymbol{K}/S_i \setminus \{i\})_{ii}$ for any suitable set S_i (not necessarily respecting lower triangularity). This also has an interpretation in terms of (2.6) and Vecchia approximations. By lower triangularity, $\boldsymbol{K}_{1,1}^{-1} \approx \boldsymbol{C}_{1,:}\boldsymbol{C}_{:,1}^* = \boldsymbol{C}_{1,1}^2$. If we had reordered the indices such that point $\boldsymbol{x}^{(i)}$ were first instead of $\boldsymbol{x}^{(1)}$ and adjusted the sparsity pattern to reflect that now *all* indices of nearest neighbors of $\boldsymbol{x}^{(i)}$ satisfy lower traingularity, we could approximate $\boldsymbol{K}$ as the square of just the entry corresponding to index i of the vector (2.6) if S_i is allowed to ignore triangularity. We provide Algorithm 4.4 to illustrate this connection, although in practice it is equivalent to inverting $\boldsymbol{K}_{S_i,S_i}$ and taking the first diagonal entry (which should be used for practical implementations).

Algorithm 4.4 Inverse Diagonal Approximation by Columnwise Vecchia Diagonal

for $i \in [n]$ **do**
  $S_i \leftarrow$ `kNN`$(\boldsymbol{x}^{(i)}, \{\boldsymbol{x}^{(1)} \dots \boldsymbol{x}^{(n)}\}, s)$ ▷ Or any suitable set, ignoring triangularity
  $\boldsymbol{v} \leftarrow$ `SparseVecchiaColumn`$(\boldsymbol{K}_{:,i}, S_i)$
  $\boldsymbol{h}_i \leftarrow \boldsymbol{v}_i^2$
end for
return $\boldsymbol{h}$

4.1.4. Alternative Pivot Choosers: Randomly Pivoted Cholesky. Instead of approximating first term of (4.9), we could also ignore it entirely. Surprisingly, this recovers a diagonal residual heuristic common to an already well-known class of column Nyström algorithms for general positive-definite matrices ([7, 4, 20]). Since the relative influence of the first term in (4.9) on the objective function diminishes as r increases, such pre-existing column Nyström algorithms might be appropriate when r is taken to be large. These methods also benefit from already-known error bounds when applied to generic positive definite matrices.

This "diagonal residual heuristic" class of algorithms computes Nyström approximations via the procedure in Algorithm 4.5, where `ChoosePivot` is a subroutine which chooses the next pivot based on the information in the diagonal of the residual.

Several choices of `ChoosePivot` are used in practice. One such choice simply ignores $\boldsymbol{d}$ and returns a uniformly random index among those not previously chosen. This approach is well-motivated: especially in the case of kernel matrices $\boldsymbol{K}$ where columns do not differ much in scale, invariant subspaces with respect to $\boldsymbol{K}$ must align closely with many columns of $\boldsymbol{K}$ in order to be large. By sampling the columns uniformly, we thus have a good chance of choosing such columns ([8]). We refer to this strategy as uniform pivoting.

Another, which we'll refer to as the greedy pivoting ([2, 7]), simply chooses

Algorithm 4.5 Column Nyström via Diagonal Residual Heuristic

$\boldsymbol{F} \leftarrow \boldsymbol{0} \in \mathbb{R}^{n\times r}$
$\boldsymbol{d} \leftarrow \mathrm{diag}(\boldsymbol{K}) \quad (= \boldsymbol{1})$ ▷ Diagonal of Nyström residual
for $i \in [r]$ **do**
  $k_{n-i+1} = \texttt{ChoosePivot}(\boldsymbol{d})$
  $\boldsymbol{F}_{:,i} \leftarrow \boldsymbol{K}_{:,i}$
  $\boldsymbol{F}_{:,i} \leftarrow \boldsymbol{F}_{:,i} - \boldsymbol{F}_{:,1:i-1}(\boldsymbol{F}_{k_{n-i+1},1:i-1})^*$
  $\boldsymbol{F}_{:,i} \leftarrow \frac{\boldsymbol{F}_{:,i}}{\boldsymbol{F}_{k_{n-i+1},i}}$
  for $j \in [n]$ **do**
    $\boldsymbol{d}_j \leftarrow \boldsymbol{d}_j - |\boldsymbol{F}_{j,i}|^2$
  end for
end for

$\texttt{ChoosePivot}(\boldsymbol{d}) = \arg\max_i \boldsymbol{d}_i$. Historically, this been motivated by the fact that for a generic positive definite matrix $\boldsymbol{A}$, we have $|\boldsymbol{A}_{i,j}| < \sqrt{\boldsymbol{A}_{i,i}\boldsymbol{A}_{j,j}}$; as the residual of a Nyström approximation is positive definite, controlling the magnitude of the diagonal entries of its residual controls the magnitude of the off-diagonal entries, too. The analysis in subsection 4.1.3 provides additional motivation when $\boldsymbol{A}$ is a covariance matrix.

Both uniform and greedy pivoting have important failure cases associated with them ([4]). In the context of GPs, if the data $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$ is grouped into order r small clusters, uniform pivoting will often fail to select a pivot corresponding to each cluster, yielding poor approximations for columns corresponding to those missed clusters. On the other hand, greedy pivoting is often derailed in the presence of order r points of $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$ which are far away from each other and the rest, as it often chooses columns corresponding to these outliers that yield poor approximations for the rest of the columns.

One method which mitigates both these failure modes is known as Randomly Pivoted Cholesky (RPCholesky). To find $\texttt{ChoosePivot}(\boldsymbol{d})$, this method samples index i with likelihood $\boldsymbol{d}_i$. This is a middle ground between uniform and greedy sampling: [20] observes that all three methods can be viewed as sampling with likelihood $\boldsymbol{d}_i^\beta$, with $\beta = 0$ for uniform, $\beta \to \infty$ for greedy, and $\beta = 1$ for RPCholesky. For general positive definite matrices, [4] prove the following bound on RPCholesky error:

PROPOSITION 4.4. *Suppose $\boldsymbol{A}$ is positive-definite, $\boldsymbol{X}^*$ is such that $\boldsymbol{A}\langle\boldsymbol{X}^*\rangle$ is the best rank-l Nyström approximation of $\boldsymbol{A}$ in the Frobenius and spectral norms with positive semi-definite residual, and*

$$r \geq \frac{l}{\varepsilon} + l\log\left(\frac{\varepsilon\,\mathrm{Tr}(\boldsymbol{A})}{\mathrm{Tr}(\boldsymbol{A}-\boldsymbol{A}\langle\boldsymbol{X}^*\rangle)}\right). \tag{4.15}$$

Then for $\hat{\boldsymbol{A}}$ a rank-r RPCholesky approximation of $\boldsymbol{A}$, we have

$$\mathbb{E}\,\mathrm{Tr}(\boldsymbol{A}-\hat{\boldsymbol{A}}) \leq (1+\varepsilon)\,\mathrm{Tr}(\boldsymbol{A}-\boldsymbol{A}\langle\boldsymbol{X}^*\rangle). \tag{4.16}$$

Note that this bound is in terms of the trace rather than the Frobenius or spectral norms. [5] also prove bounds on the spectral norm of the residual when $\boldsymbol{A}$ is a kernel matrix with nonzero nugget parameter, but they are not useful in the general case with no nugget.

In our case where diagonal heuristic methods are motivated by the form of (4.9), RPCholesky can be justified over greedy pivoting because the second term's magnitude

will approach the first as r increases, but the terms have different signs. In other words, optimizing only the second term of (4.9) may correlate with choosing an index with worse first term, which the randomization attempts to adjust for. Together with the above theoretical bounds, there is a strong case for using RPCholesky for the Nyström part; we analyze its performance in section 5.

4.2. Greedy Conditional Selection. We now explore an alternative sparsity pattern chooser to the nearest neighbors type method in subsection 4.1.1 and subsection 4.1.2. While Theorem 4.2 is encouraging for said method, ultimately, no purely Euclidean nearest-neighbors based algorithm can incorporate any information gained from the Nyström part. This seems at odds with the idea of CNV approximations, where the column Nyström pivots and sparsity pattern should ideally complement each other to improve approximation quality. Is there any way our sparsity pattern chooser can be made to take into account column Nyström information?

4.2.1. GCS for FSAI Sparsity Pattern Selection. [11]'s Greedy Conditional Selection (GCS) is one possible idea, which we first develop in a purely-sparse-Vecchia context.

Let $\boldsymbol{K}$ be a the covariance matrix induced on some $\{\boldsymbol{x}^{(i)}\}_{i=1}^n \subseteq \mathbb{R}^d$ by some kernel function κ. In subsubsection 4.1.3, we motivated an algorithm which uses sparse approximation to estimate the change in KL loss function (4.5) from adding a pivot to the column Nyström approximation of $\boldsymbol{K}$, which is then greedily minimized to choose pivots.

One might similarly hope to greedily minimize (4.5) to choose the sparsity patterns $\{S_i\}_{i=1}^n$ of individual columns. Conveniently, (4.8)

$$2D_{\mathrm{KL}}\left(\mathcal{N}(0,\boldsymbol{K}),\mathcal{N}(0,(\boldsymbol{C}\boldsymbol{C}^*)^{-1}\right) = \sum_{i=1}^n \Big[\log\big((\boldsymbol{K}/S_i\setminus\{i\})_{ii}\big) - \log\big((\boldsymbol{K}/\{i+1:n\})_{ii}\big)\Big]$$

has the form of a sum in which terms correspond to individual columns' sparsity patterns and are constant in changes to the sparsity patterns of other columns. In particular, the change in objective function from adding index k to the sparsity pattern in the ith column S_i is given by

$$\begin{aligned}\log\big((\boldsymbol{K}/S_i\cup\{k\}\setminus\{i\})_{i,i}\big) - \log\big((\boldsymbol{K}/S_i\setminus\{i\})_{ii}\big) \\ = \log\left((\boldsymbol{K}/S_i\setminus i)_{ii} - \frac{(\boldsymbol{K}/S_i\setminus\{i\})_{ik}^2}{(\boldsymbol{K}/S_i\setminus\{i\})_{kk}}\right) - \log\big((\boldsymbol{K}/S_i\setminus\{i\})_{ii}\big) \\ = \log\left(1 - \frac{(\boldsymbol{K}/S_i\setminus\{i\})_{ik}^2}{(\boldsymbol{K}/S_i\setminus i)_{ii}(\boldsymbol{K}/S_i\setminus\{i\})_{kk}}\right)\end{aligned} \tag{4.17}$$

which is minimized by maximizing

$$\frac{(\boldsymbol{K}/S_i\setminus\{i\})_{ik}^2}{(\boldsymbol{K}/S_i\setminus i)_{ii}(\boldsymbol{K}/S_i\setminus\{i\})_{kk}} = \mathrm{Corr}(\boldsymbol{y}_i,\boldsymbol{y}_k\mid\boldsymbol{y}_{S_i\setminus\{i\}}) \text{ for } \boldsymbol{y}\sim\mathcal{N}(0,\boldsymbol{K}) \tag{4.18}$$

over k. Since $(\boldsymbol{K}/S_i\setminus i)_{ii}$ does not depend on k, we may omit division by it from the optimization if we wish.

Unfortunately, computing (4.18) for the order n points per column satisfying lower triangularity, for each of the n columns, even once would give such an algorithm a $\Omega(n^2)$ computational complexity. To make an algorithm based on (4.18) tractable,

we must only maximize (4.18) over a *candidate set* $C_i \subseteq \{i+1 \dots n\}$ for each ith column, with $\max_i |C_i| = c$ much less than n. In other words, we must constrain our optimization space by enforcing $S_i \subseteq C_i$. Motivated by Theorem 4.1, [11] recommends choosing each C_i to correspond to nearest neighbors of $\boldsymbol{x}^{(i)}$, especially in the case of Matérn kernels. We note that c is taken to be greater than s, or else this simply recovers the s-nearest-neighbors-based sparsity pattern of subsection 4.1.1.

When restricted to optimizing over C_i instead of $[n]$ for each i, [11] gives two different algorithms to iteratively construct S_i by computing and greedily maximizing (4.18). One is based on Sherman-Morrison updates to $(\boldsymbol{K}_{S_i \setminus \{i\}, S_i \setminus \{i\}})^{-1}$, while the other updates select entries of a column Nyström residual based on S_i by maintaining a Cholesky factor. They are just as fast as each other; we give pseudocode for and analyze computational complexity of the latter in Appendix A.1, and the former is Algorithm C.1 in [11]. Both require $O(ncd + ncs^2)$ operations.

4.2.2. GCS for Adaptive Factorized Nyström Approximations. Now, we'll adapt GCS to tractably select the sparsity pattern when computing the sparse Vecchia approximation of the residual during CNV approximations.

Like any method for choosing a sparsity pattern, GCS will work off-the-shelf when applied directly to the Nyström residual. However, this naïve implementation requires explicitly computing the entire column Nyström residual, which takes order $n^2 r$ operations for explicit computation of the Nyström approximation plus n^2 times the operations needed for a kernel evaluation to compute $\boldsymbol{K}$ explicitly (which could be even larger than r).

Fortunately, the interpretation discussed in section 3 that CNV approximations are sparse Vecchia approximations with the bottom r rows filled in allows us to avoid this issue. In particular, as each successive pivot k_j is added to the pivot set R, we can update $(\boldsymbol{K}_{R\setminus\{i\}, R\setminus\{i\}})^{-1}$ as required in [11]'s Algorithm C.1 or the pivoted Cholesky matrix required in Algorithm A.1 by treating k_j as if it were the index that GCS had chosen to add to S_i at that step. This is necessary for each $i \in [n] \setminus R$, but $(\boldsymbol{K}_{R\setminus\{i\}, R\setminus\{i\}})^{-1}$ does not even depend on i for $i \notin R$, and the pivoted Cholesky matrices required in Algorithm A.1 for each i at the step where $S_i = R$ are all submatrices of a common matrix which only needs to be computed once. After that, we can simply continue GCS as normal on a column-by-column basis. We provide pseudocode for such a modified version of the latter in A.1. The CNV version requires $\mathcal{O}(nr^2 + ns(c+r)(r+s) + n(c+r)d) = \mathcal{O}(nr^2 + ns^2c + nscr + ncd)$ operations, as shown in Appendix A.1 (note that we always have $c \geq s$).

5. Experiments and analysis. In this section, we compare the performance of algorithms discussed in subsection 4.1 and subsection 4.2 on Matérn kernel matrices. Specifically, we examine the various methods' recovery in the Frobenius norm

$$\|\boldsymbol{L}^* \boldsymbol{K} \boldsymbol{L} - I\|_{\text{FRO}}, \tag{5.1}$$

and how well they control the condition number

$$\kappa(\boldsymbol{L}\boldsymbol{K}\boldsymbol{L}^T) = \frac{\lambda_{\max}(\boldsymbol{L}\boldsymbol{K}\boldsymbol{L}^T)}{\lambda_{\min}(\boldsymbol{L}\boldsymbol{K}\boldsymbol{L}^T)}. \tag{5.2}$$

The Frobenius error is indicative of recovery of individual entries in the matrix; we use $\|\boldsymbol{L}\boldsymbol{K}\boldsymbol{L}^T - I\|_{\text{FRO}}$ rather than $\|\boldsymbol{K} - (\boldsymbol{L}\boldsymbol{L}^*)\|_{\text{FRO}}$ due to the Vecchia approximation's optimality in the former ([23]) and the latter depending significantly on $\lambda_{\min}(\boldsymbol{K})$ which could cause higher variance when testing multiple samplings of synthetic data.

The condition number gives a worst-case bound on the number of iterations of preconditioned conjugate gradient (see subsection 2.1). We also measure the actual performance of CNV preconditioners on PCG by running PCG with these preconditioners on a set of withheld data points and averaging the number of iterations required before the vector 2-norm error is below a threshold (10^{-4} times the error at initialization).

5.1. Setup. We implemented various methods for both the column Nyström pivot choosing and sparsity pattern choosing steps of the CNV approximation framework in the Julia programming language. Specifically, for the Nyström part, we implemented [24]'s Farthest Point Sampling (FPS), [4]'s Randomly Pivoted Cholesky (RPC), and our own approximate greedy KL minimization (GKL). We distinguish algorithms Algorithm 4.3 and Algorithm 4.4 by referring to the first as GKLR and the second as GKLC. For the sparsity pattern choosing part, we implemented [18]'s Nearest Neighbors (kNN) and [11]'s Greedy Conditional Selection (GCS). GCS requires us to specify a number of candidate points c to consider; in this section, we generally set this to a scalar multiple of s (nonzeros per column in the sparse part) and refer to GCS with a specific scalar by appending the scalar (e.g. GCS10 refers to GCS with candidate set size $c = 10s$). We specifically note that GCS with $c = s$ is equivalent to kNN, though we always write kNN rather than GCS1.

For data, we always take $n = 5000$. We work with synthetic data, namely points uniformly sampled from d-dimensional hypercubes. Working with synthetic data in this manner allows us to isolate the effects of changes in specific parameters through experimentation, and we can also sample many synthetic datasets and appropriately average performance metrics across these samples for robustness. In particular, all quantities reported are arithmetic means (Frobenius error, PCG iterations) or geometric means (condition numbers, max/min eigenvalues) over 3 different synthetic datasets sampled in the same manner. We work only with Matérn kernels in this section, and take $\nu = 2.5$ unless otherwise specified. Finally, we set bandwidths of our Matérn kernels to equal the trace of the $d \times d$ sample covariance matrix of the 5000 data points ([1]). This is useful, among other reasons, to compare performance data apples-to-apples as d is varied.

5.2. Results. We first attempt to determine the best value of c, which because kNN is the special case of GCS with $c = s$ has choosing between kNN and GCS as a subproblem. We plot, for two choices $r = 20$ and $r = 60$ of r the rank in the column Nyström step, the performance of the resulting preconditioners for several values of c and s in Figure 1.

Unsurprisingly, $c = 10s$ universally performs the best. Recalling from subsection 4.2.2 that the runtime of GCS is at most linear in c, and that all of our preconditioner-finding algorithms were designed to be $o(n^2)$, we believe the few PCG iterations saved by GCS10 over GCS5 is worth the increased complexity of the GCS step. Moreover, in cases where d is higher, GCS performs even better. In Figure 2, GCS10 improves PCG convergence rate by an order of magnitude compared to kNN or GCS2 for $d = 5$ and $d = 10$. For d even larger than this, taking $c > 10$ may even be advisable.

We also observe that in Figure 2, the condition number and mean PCG iterations behave counterintuitively in that they initially increase with increasing s. Note that this does not contradict the optimality properties of CNV approximations in section 3, as the condition number is not one of the objectives which Vecchia approximations are optimal in, and the Frobenius error, which was such an objective, did still decrease

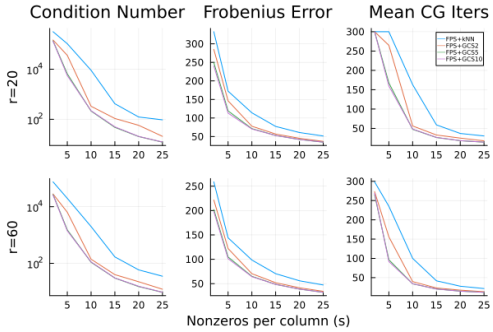

FIG. 1. *Performance comparison of GCS for different values of c as r and s vary. Pivots are chosen by FPS.*

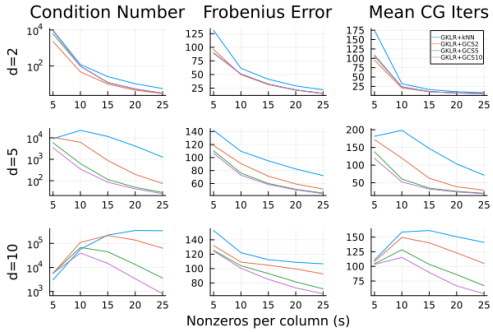

FIG. 2. *Performance comparison of GCS for different values of c as d and s vary. $r = 60$ is fixed. Pivots are chosen by GKLR.*

monotonically. We lack a complete explanation for this, but empirical observations indicate this behavior is due more to the largest eigenvalue increasing than the smallest eigenvalue decreasing as s increases.

Now that we have suitably chosen c, we turn our attention to comparing algorithms for the column Nyström step. We first compare GKLR and GKLC, which we plot in Figure 3 alongside the (completely intractable) algorithm which directly inverts the kernel matrix to recover (4.14) exactly. We refer to the latter as GKLE (as it is "exact"). Remarkably, the performance of each of these three algorithms is nearly indistinguishable, implying GKLR and GKLC both approximate (4.14) well even for single-digit values of s. We choose GKLR to represent GKL going forward, more or less arbitrarily.

Now, we compare [24]'s FPS, [4]'s RPC, and our GKLR, in Figure 4. We observe that GKLR is usually best, especially for large r and small s. GKLR takes as long as kNN, which takes $\Omega(ns^3)$, while GCS10 takes $\Omega(ns^2c)$ which for $c = 10s$ means approximately 10 times as long. This means that if we're using GCS to choose the sparsity pattern, using GKLR as opposed to FPS or RPC requires only about 10% more operations for the creation of the preconditioner altogether.

As the preconditioning methods in this paper generally take order n^2 or less operations, which is less than one iteration of PCG, this is well worth saving even a few PCG iterations as it usually does for s relatively small. However, for relatively large s, the choice of column Nyström pivot chooser matters much less in terms

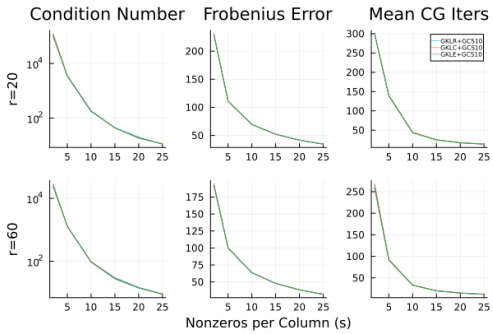

FIG. 3. *Performance comparison of GKLR, GKLC, and GKLE for different values of r and s.*

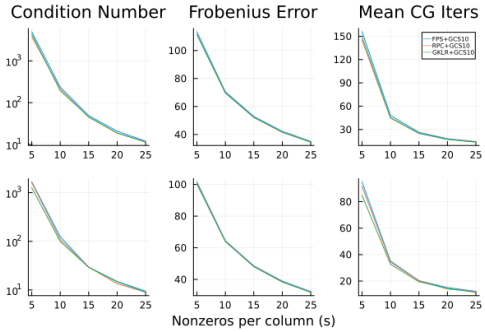

FIG. 4. *Comparison of methods for column Nyström step for different values of r and s. $r = 20$ in the top row and $r = 60$ in the bottom row.*

of iterations for PCG convergence while GKLR (and preconditioner construction in general) takes comparatively longer, meaning using FPS or RPC instead and slightly increasing s might be a more efficient use of computational resources.

However, we remark that as ν increases, the gap between GKLR and FPS, and to a lesser extent RPC, widens. In Figure 5, we fix $s = 15$ and compare the algorithms for different values of ν and r. For $\nu = \frac{7}{2}$, one notices a significant gap between FPS and the other methods, though RPC seems to keep pace with GKLR relatively well.

Finally, we fix $s = 15$ and examine the effects of varying r in Figure 6, in attempt to answer how much of a performance improvement the low-rank component of CNV really contributes. As can be seen, increasing r results in rapid improvements while r is still small, though the marginal benefit decreases substantially for $r > 5$ or so. Thus, CNV results in dramatic improvement over purely sparse Vecchia, here a three to four times improvement for the condition number and number of CG iterations, even for small r. Nevertheless, given that (in the case of GCS) the complexity of the preconditioning step is order $nr^2 + ns^2c$, and given that the improvement from increasing r past 5 or so is still significant, it seems that taking r somewhere between s and $s\sqrt{c}$ is worthwhile.

Acknowledgments. I would like to thank my advisor, Robert Webber, for his mathematical expertise as well as all he has taught me about the process of mathematical research. This project would not have come together if it were not for his help directing it, and I feel I have improved significantly as a mathematician thanks to his advice throughout the past two quarters.

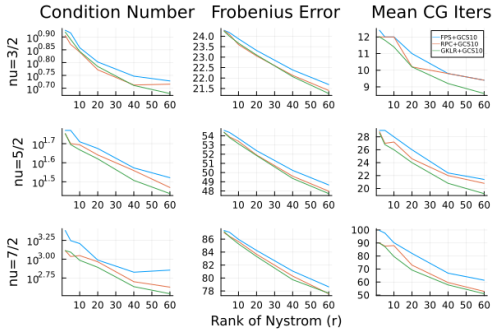

FIG. 5. *Performance comparison of methods for column Nyström step for different values of ν and r. $s = 15$ is fixed.*

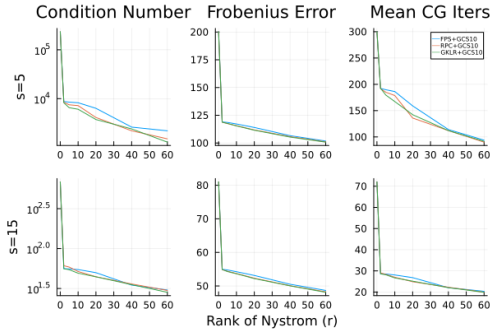

FIG. 6. *Plotting objective functions as r increases from 0, to determine the impact of the column Nyström part in CNV approximations.*

I would also like to thank Christopher J. Geoga, Chris Camaño, Ethan N. Epperly, and Florian Schäfer for helpful discussions.

REFERENCES

[1] D. ARISTOFF, M. JOHNSON, G. SIMPSON, AND R. J. WEBBER, *The fast committor machine: Interpretable prediction with kernels*, The Journal of Chemical Physics, 161 (2024), p. 084113, https://doi.org/10.1063/5.0222798, https://doi.org/10.1063/5.0222798, https://arxiv.org/abs/https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0222798/20131959/084113_1_5.0222798.pdf.

[2] F. R. BACH AND M. I. JORDAN, *Predictive low-rank decomposition for kernel methods*, in Proceedings of the 22nd International Conference on Machine Learning, ICML '05, New York, NY, USA, 2005, Association for Computing Machinery, p. 33–40, https://doi.org/10.1145/1102351.1102356, https://doi.org/10.1145/1102351.1102356.

[3] D. BERNSTEIN, *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory*, Princeton University Press, 2005, https://books.google.com/books?id=pmNRPwOFHKoC.

[4] Y. CHEN, E. N. EPPERLY, J. A. TROPP, AND R. J. WEBBER, *Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations*, Communications on Pure and Applied Mathematics, 78 (2025), pp. 995–1041, https://doi.org/10.1002/cpa.22234.

[5] M. DÍAZ, E. N. EPPERLY, Z. FRANGELLA, J. A. TROPP, AND R. J. WEBBER, *Robust, randomized preconditioning for kernel ridge regression*, 2024, https://arxiv.org/abs/2304.12465.

[6] E. N. EPPERLY, J. A. TROPP, AND R. J. WEBBER, *Embrace rejection: Kernel matrix approximation by accelerated randomly pivoted Cholesky*, 2024, https://arxiv.org/abs/2410.03969.

[7] S. Fine and K. Scheinberg, *Efficient svm training using low-rank kernel representations*, J. Mach. Learn. Res., 2 (2002), p. 243–264.
[8] A. Gittens, *The spectral norm error of the naive nystrom extension*, arXiv preprint arXiv:1110.5305, (2011).
[9] G. H. Golub and C. F. Van Loan, *Matrix Computations - 4th Edition*, Johns Hopkins University Press, Philadelphia, PA, 4 ed., 2013, https://doi.org/10.1137/1.9781421407944.
[10] J. Guinness, *Permutation and grouping methods for sharpening gaussian process approximations*, Technometrics, 60 (2018), pp. 415–429.
[11] S. Huan, J. Guinness, M. Katzfuss, H. Owhadi, and F. Schäfer, *Sparse Cholesky factorization by greedy conditional selection*, 2023, https://arxiv.org/abs/2307.11648.
[12] I. Kaporin, *An alternative approach to estimating the convergence rate of the cg method*, Numerical Methods and Software, Yu. A. Kuznetsov, ed., Dept. of Numerical Mathematics, USSR Academy of Sciences, Moscow, (1990), pp. 55–72.
[13] M. Katzfuss and J. Guinness, *A general framework for vecchia approximations of gaussian processes*, Statistical Science, 36 (2021), pp. 124–141.
[14] M. Katzfuss, J. Guinness, W. Gong, and D. Zilber, *Vecchia approximations of gaussian-process predictions*, Journal of Agricultural, Biological and Environmental Statistics, 25 (2020), pp. 383–414.
[15] A. Krause, A. Singh, and C. Guestrin, *Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies.*, Journal of Machine Learning Research, 9 (2008).
[16] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions—i*, Mathematical programming, 14 (1978), pp. 265–294.
[17] A. J. Rothman, E. Levina, and J. Zhu, *Sparse multivariate regression with covariance estimation*, Journal of Computational and Graphical Statistics, 19 (2010), pp. 947–962, http://www.jstor.org/stable/25765382 (accessed 2025-04-25).
[18] F. Schäfer, M. Katzfuss, and H. Owhadi, *Sparse cholesky factorization by kullback–leibler minimization*, SIAM Journal on Scientific Computing, 43 (2021), pp. A2019–A2046, https://doi.org/10.1137/20M1336254.
[19] F. Schäfer, T. J. Sullivan, and H. Owhadi, *Compression, inversion, and approximate pca of dense kernel matrices at near-linear computational complexity*, arXiv preprint arXiv:1706.02205, (2017).
[20] S. Steinerberger, *Randomly pivoted partial cholesky: Random how?*, arXiv preprint arXiv:2404.11487, (2024).
[21] J. A. Tropp and R. J. Webber, *Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications*, arXiv preprint arXiv:2306.12418, (2023).
[22] A. V. Vecchia, *Estimation and model identification for continuous spatial processes*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 50 (1988), pp. 297–312.
[23] A. Y. Yeremin, L. Y. Kolotilina, and A. Nikishin, *Factorized sparse approximate inverse preconditionings. III. Iterative construction of preconditioners*, Journal of Mathematical Sciences, 101 (2000), pp. 3237–3254.
[24] S. Zhao, T. Xu, H. Huang, E. Chow, and Y. Xi, *An adaptive factorized Nyström preconditioner for regularized kernel matrices*, SIAM Journal on Scientific Computing, 46 (2024), pp. A2351–A2376, https://doi.org/10.1137/23M1565139.

Appendix A. Postponed Pseudocode and Complexity Analysis.

A.1. Greedy Conditional Selection. In this section, we provide pseudocode for and examine the computational complexity of the two algorithms for GCS (one for purely sparse Vecchia and one for CNV) proposed in subsection 4.2.

First, we discuss Algorithm A.1. We write $\ker(x^{(\cdot)}, x^{(\cdot)})$ as opposed to $\boldsymbol{K}_{\cdot,\cdot}$ to emphasize that the entire kernel matrix need not be explicitly computed. Examining the pseudocode, we require order ncs kernel evaluations which takes order ncd operations for most commonly-used kernels, plus ns instances of up to $c \times s$ matrix by $s \times 1$ vector multiplication which takes order ncs^2 operations, plus nsc scalar updates in the innermost loop, for a complexity of $O(ncd + ncs^2)$. Correctness of the updates to the diagonal and respective column is more rigorously proven in [11].

We now discuss our modification to work with CNV, given in Algorithm A.1. Compared to Algorithm A.1, it adds $O(nr^2)$ operations up front (which would have

Algorithm A.1 GCS via Truncated Cholesky

for $i \in [n]$ **do**
  $\{k_l\}_{l=1}^{c} \leftarrow C_i$ ▷ Order arbitrary
  $k_{c+1} \leftarrow i$
  $\boldsymbol{d} \leftarrow \kappa(\boldsymbol{x}^{(\{k_l\}_{l=1}^{c})}, \boldsymbol{x}^{(\{k_l\}_{l=1}^{c})})$ $(= \mathbf{1}$ for kernels$)$ ▷ Nyström residual diagonal
  $\boldsymbol{b} \leftarrow \kappa(\boldsymbol{x}^{(\{k_l\}_{l=1}^{c})}, \boldsymbol{x}^{(i)})$ ▷ Nyström residual ith column
  $\boldsymbol{L} = \mathbf{0} \in \mathbb{R}^{c+1 \times c+1}$
  for $j \in [s]$ **do**
    $m = \arg\max_{l \in k_{[c]}} \frac{\boldsymbol{b}_l^2}{\boldsymbol{d}_l}$
    $S_i \leftarrow S_i \cup \{m\}$
    $\boldsymbol{L}_{:,j} \leftarrow \kappa(\boldsymbol{x}^{(\{k_l\}_{l=1}^{c})}, \boldsymbol{x}^{(m)})$
    $\boldsymbol{L}_{:,j} \leftarrow \boldsymbol{L}_{:,j} - \boldsymbol{L}_{:,1:j-1}(\boldsymbol{L}_{m,1:j-1})^*$
    $\boldsymbol{L}_{:,j} \leftarrow \frac{\boldsymbol{L}_{:,j}}{\sqrt{\boldsymbol{L}_{m,j}}}$
    for $l \in [c]$ **do**
      $\boldsymbol{d}_l \leftarrow \boldsymbol{d}_l - \boldsymbol{L}_{k_l,j}^2$
      $\boldsymbol{b}_l \leftarrow \boldsymbol{b}_l - \boldsymbol{L}_{k_l,j}\boldsymbol{L}_{\boldsymbol{c+1},\boldsymbol{j}}$
    end for
  end for
end for

been needed for truncated Cholesky-based column Nyström anyway), then adds $O(ncr)$ operations across all columns for updating Nyström residuals' diagonals and corresponding columns (this can be optimized to $O(nr)$ by computing them all at once, but is not a bottleneck). It also increases the complexity of the matrix-vector multiplication in the second block of loops to order $(c+r)(r+s)$ operations, which happen order ns times for a bound of $O(ns(c+r)(r+s))$ overall, as well as requiring order rn more kernel evaluations which usually takes $O(rnd)$ more operations. So the CNV version requires $O(nr^2 + ns(c+r)(r+s) + n(c+r)d) = O(nr^2 + ns^2c + nscr + ncd)$ operations (note that we always have $c \geq s$).

Algorithm A.2 CNV using GCS via Truncated Cholesky

$\bar{\boldsymbol{L}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{n \times r+s}$
for $j \in [r]$ **do**
  $\bar{\boldsymbol{L}}_{:,j} \leftarrow \kappa(\boldsymbol{x}^{([n])}, \boldsymbol{x}^{(k_j)})$ ▷ $\{k_j\}_{j=1}^r$ should be the Nyström pivots
  $\bar{\boldsymbol{L}}_{:,j} \leftarrow \boldsymbol{L}_{:,j} - \boldsymbol{L}_{:,1:j-1}(\boldsymbol{L}_{k_j,1:j-1})^*$
  $\boldsymbol{L}_{:,j} \leftarrow \frac{\boldsymbol{L}_{:,j}}{\sqrt{\boldsymbol{L}_{k_j,j}}}$
end for
for $i \in [n]$ **do**
  $\{k_j\}_{j=r+1}^{c+r} \leftarrow C_i$ ▷ Order arbitrary
  $k_{c+r+1} \leftarrow i$
  $\boldsymbol{d} \leftarrow \kappa(\boldsymbol{x}^{(\{k_l\}_{l=r+1}^{c+r})}, \boldsymbol{x}^{(\{k_l\}_{l=r+1}^{c+r})})$ $(= \boldsymbol{1}$ for kernels$)$ ▷ Nyström residual diagonal
  $\boldsymbol{b} \leftarrow \kappa(\boldsymbol{x}^{(\{k_l\}_{l=r+1}^{c+r})}, \boldsymbol{x}^{(i)})$ ▷ Nyström residual ith column
  for $j \in [r]$ **do**
    for $l \in [c]$ **do**
      $\boldsymbol{d}_l \leftarrow \boldsymbol{d}_l - \boldsymbol{L}_{l,j}^2$
      $\boldsymbol{b}_l \leftarrow \boldsymbol{b}_l - \boldsymbol{L}_{l,j}\boldsymbol{L}_{\boldsymbol{c+1},\boldsymbol{j}}$
    end for
  end for
  $\boldsymbol{L} = \bar{\boldsymbol{L}}_{R \cup C,:}$
  for $j \in [s] + r$ **do**
    $m = \arg\max_{l \in k_{[c]+r}} \frac{\boldsymbol{b}_l^2}{\boldsymbol{d}_l}$
    $S_i \leftarrow S_i \cup \{m\}$
    $\boldsymbol{L}_{:,j} \leftarrow \kappa(\boldsymbol{x}^{(\{k_l\}_{l=1}^{c})}, \boldsymbol{x}^{(m)})$
    $\boldsymbol{L}_{:,j} \leftarrow \boldsymbol{L}_{:,j} - \boldsymbol{L}_{:,1:j-1}(\boldsymbol{L}_{m,1:j-1})^*$
    $\boldsymbol{L}_{:,j} \leftarrow \frac{\boldsymbol{L}_{:,j}}{\sqrt{\boldsymbol{L}_{m,j}}}$
    for $l \in [c] + r$ **do**
      $\boldsymbol{d}_l \leftarrow \boldsymbol{d}_l - \boldsymbol{L}_{l,j}^2$
      $\boldsymbol{b}_l \leftarrow \boldsymbol{b}_l - \boldsymbol{L}_{l,j}\boldsymbol{L}_{\boldsymbol{c+1},\boldsymbol{j}}$
    end for
  end for
end for