Sparse Inverse Covariance Estimation with Hierarchical Matrices

Jonas Ballani*  Daniel Kressner*

October 12, 2014

Abstract
In statistics, a frequent task is to estimate the covariance matrix of a set of normally distributed random variables from given samples. In a high-dimensional regime, this problem becomes particularly challenging as the number of samples is typically much smaller than the number of variables. If the inverse covariance matrix happens to be sparse, an $\ell_1$-based sparsity constraint often leads to a successful identification of the underlying correlations. The resulting convex optimization problem can then efficiently be solved by Newton-like techniques with super-linear or even quadratic convergence rates. A drawback of the associated quadratic model is the required computation of (dense) inverses of sparse matrices. We address this problem by the usage of hierarchical matrices which allow for an (approximate) data-sparse representation of large dense matrices. This explicit representation enables us to further exploit the simultaneous treatment of groups of variables in a block-wise manner and to easily ensure positive definiteness of each iterate. Numerical examples indicate an at most quadratic scaling in time in the number of variables under moderate storage consumptions even for high-dimensional problems.

1 Introduction
Given $n$ i.i.d. samples $y_1, \ldots, y_n$ drawn from a $p$-variate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, we consider the task of estimating the inverse covariance matrix (also called precision matrix) $\Sigma^{-1} \in \mathbb{R}^{p \times p}$ when $p \gg n$. In such a setting, the sample covariance matrix $S$ given by

$$S := \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{\mu})(y_i - \hat{\mu})^\top,$$

is singular and one cannot estimate $\Sigma^{-1}$ directly from the inverse of $S$. A standard approach to regularize this problem is to impose a sparsity prior onto $\Sigma^{-1}$. For a given parameter $\lambda > 0$, the $\ell_1$-regularized maximum log-likelihood estimator of $\Sigma^{-1}$ is then given by the minimum of the convex function

$$f(X) := -\log \det X + \text{tr}(SX) + \lambda \| X \|_1$$

over all symmetric positive definite matrices $X \in \mathbb{R}^{p \times p}$.

Sparse inverse covariance matrix estimation is a standard task in statistics and the recent surge of interest, in particular from machine learning, has led to a large number of iterative algorithms to tackle this optimization problem. In the following, we briefly summarize some recent developments.

A variety of first-order methods have been proposed, relying on block-wise coordinate descent [1, 4, 7, 19], interior point methods [25, 15], projected subgradients [6], alternating minimization [20], iterative thresholding [18], or greedy strategies [21]. These methods typically exhibit a rather slow convergence behavior. In special cases, thresholding strategies as in [9, 16, 24] can be used to split the full optimization problems into smaller subproblems.

Second-order methods such as those discussed in [17] and the QUadratic Inverse Covariance algorithm (QUIC) [10, 11] aim at computing an (approximate) Newton direction in every step. This increases the computational cost per iteration but leads to much faster convergence. In QUIC, an approximation of the Newton direction is achieved by coordinate descent which allows to approach large-scale inverse covariance estimation problems.

The key to scaling iterative methods for sparse inverse covariance matrix estimation to high dimensions $p$ is to maintain the sparsity of the iterates $X$ throughout the whole approximation process. There are, however, a number of obstacles that need to be overcome to be able to address very large values of $p$. First, for each iterate $X$ one should be able to evaluate the objective function $f$. In particular, this requires the accurate evaluation of $\log \det X$, which is by no means trivial for large-scale matrices. Second, to gain information on a possible descent direction starting from $X$ one needs to be able to compute (or approximate) $X^{-1} = \nabla \log \det X$. In addi-
tion, the positive definiteness of $X$ needs to be guaranteed.

In this paper, we propose a unified framework that addresses the issues mentioned above. Our main idea is to represent the sparse iterates $X$ (as they arise, e.g., when performing QUIC) and their inverses $X^{-1}$ in a structured low-rank format, namely by hierarchical matrices (cf. [8] and the references therein). By a combination of clustering techniques with block-wise low-rank approximation, this class of matrices can be used for the approximate factorization and inversion of large classes of sparse matrices [2]. Originally proposed in the context of numerical methods for partial differential equations, comparably few attempts have been made to apply hierarchical matrices and variants thereof to other domains; see [3, 22] for an application to kernel approximation and Gaussian process regression.

In the numerical experiments we demonstrate that the combination of hierarchical matrices with a state-of-the-art estimation technique, such as QUIC, achieves excellent performance both on synthetic and real-world data sets. However, it is important to emphasize that such performance gains can only be expected if the minimizer of $f$ is sparse. When sparsity gets lost (e.g., due to an unfortunate choice of $\lambda$), hierarchical matrices also tend to lose their data-sparse approximation power.

2 Quadratic Approximation

In this section, we review the QUIC algorithm, which we use as the primary example to illustrate the benefits of hierarchical matrices in sparse inverse covariance matrix estimation. Let us recall that the goal is to find the unique minimizer of the objective function

$$f(X) = -\log \det X + \text{tr}(SX) + \lambda \|X\|_1.$$  

of the strictly convex function $f(X) := -\log \det X + \text{tr}(SX) + \lambda \|X\|_1$. Here, $X \succ 0$ restricts $X$ to be symmetric positive definite and $\lambda$ is a regularization parameter for the $\ell_1$-penalty $\|X\|_1 := \sum_{i,j=1}^p |X_{ij}|$.

The objective function $f$ can be decomposed as $f(X) = g(X) + \lambda \|X\|_1$ with the smooth function

$$g(X) := -\log \det X + \text{tr}(SX).$$

In particular, we have

$$\nabla g(X) = S - X^{-1}, \quad \nabla^2 g(X) = X^{-1} \otimes X^{-1},$$

where $\otimes$ denotes the usual Kronecker product. A quadratic model of $f$ around a point $X + \Delta$ can therefore be obtained as

$$f(X + \Delta) \approx q_X(\Delta) := \text{tr}((S - X^{-1})\Delta) + \frac{1}{2} \text{tr}(X^{-1}\Delta X^{-1}) - \log \det X + \text{tr}(SX) + \lambda \|X + \Delta\|_1.$$

Starting from $X \approx X^*$, the aim of a quadratic approximation scheme is to find the generalized Newton direction $D$ of $f$ at $X$ from the solution of the minimization problem

$$D := \arg \min_{\Delta} q_X(\Delta).$$

For a suitable step size $\alpha \in (0,1]$, we then update $X$ by $X' := X + \alpha D$ in order to obtain an improved approximation $X' \approx X^*$. Note that the Hessian $\nabla^2 g(X)$ is a $p^2 \times p^2$ matrix which makes the direct solution of (2.3) intractable for large $p$.

2.1 QUIC Algorithm. The QUIC algorithm [10, 11] computes an approximation of the unique minimizer $X^*$ from (2.3) by (block) coordinate descent. To this end, the index set

$$I := \{(i,j) \in \{1, \ldots, p\}^2 : j \leq i\},$$

is first subdivided into two disjoint subsets

$$I_{\text{fixed}} := \{(i,j) \in I : |\nabla_{ij} g(X)| \leq \lambda, X_{ij} = 0\},$$

$$I_{\text{free}} := I \setminus I_{\text{fixed}}.$$  

It turns out that

$$\Delta' := \arg \min_{\Delta} q_X(\Delta) \quad \text{such that} \quad \Delta_{ij} = 0 \forall (i,j) \in I_{\text{free}}$$

has the trivial solution $\Delta' = 0$ which means that the first block update for all $(i,j) \in I_{\text{fixed}}$ can be avoided.

Let now $(i,j) \in I_{\text{free}}$ and assume that we are given a symmetric iterate $\Delta$ approximating $D$. An improved iterate

$$\Delta' := \Delta + \mu'(e_i e_j^T + e_j e_i^T)$$

in direction $(i,j)$ can then be obtained by the solution of the one-dimensional optimization problem

$$\mu' := \arg \min_{\mu \in \mathbb{R}} q_X(\Delta + \mu(e_i e_j^T + e_j e_i^T)).$$

Letting $W := X^{-1} = [w_1, \ldots, w_p]$, we can find the solution to (2.4) analytically by computing

$$a := W_{ij}^2 + W_{ii} W_{jj},$$

$$b := S_{ij} - W_{ij} + w_i^T \Delta w_j,$$

$$c := X_{ij} + \Delta_{ij},$$

$$\mu' := -c + S(e - b/a, \lambda/a),$$
with the soft-thresholding function

\[ S(z, r) := \text{sign}(z) \max\{|z| - r, 0\} \]

Note that the update direction \( \Delta \) possesses only \( O(\#I_{\text{free}}) \) non-zero entries implying that the term \( w_i^\top \Delta w_j \) can always be computed within a complexity of \( O(\#I_{\text{free}}) \).

### 2.2 Main Computational Bottlenecks.

For large values of \( p \), the computationally most expensive operations performed within QUIC are

(a) computing \( \log \det X \),

(b) ensuring positive definiteness of each iterate \( X \) (when selecting the step size), and

(c) computing (and storing) \( X^{-1} \).

By no means specific to QUIC, these operations are at the heart of most existing iterative algorithms for solving (2.2). A standard approach to solve (a)–(c) simultaneously is to

(i) compute the Cholesky factorization \( X = LDL^\top \) with a lower triangular matrix \( L \) that has ones on the diagonal and a diagonal matrix \( D = \text{diag}(d_{11}, \ldots, d_{pp}) \),

(ii) check positive definiteness of \( X \) by verifying that all \( d_{11}, \ldots, d_{pp} \) are positive,

(iii) compute \( \log \det X = \sum_{i=1}^p \log d_{ii} \),

(iv) compute \( X^{-1} = (L^\top)^{-1}D^{-1}L^{-1} \).

A (dense) Cholesky factorization requires \( O(p^3) \) operations and \( O(p^2) \) memory, which imposes tight limitations on the values of \( p \) that remain computationally feasible.

A sparse iterate \( X \) allows for the use of a sparse Cholesky factorization [5] in (i). Its computational complexity is determined by the number of nonzero entries in \( L \). The fill-in (that is, additional nonzero entries in \( L \) relative to \( X \)) strongly depends on the sparsity pattern of \( X \) and a suitable permutation of the indices. Even when there is little fill-in, the inversion (iv) will in general result in a dense matrix \( X^{-1} \).

Although we cannot expect \( X^{-1} \) to be sparse, it turns out that its off-diagonal blocks are often (approximately) of low rank. We may therefore still be able to exploit data-sparsity in \( X^{-1} \). We will reach this goal by combining ideas from clustering and low-rank approximations in a suitable way.

### 3 Hierarchical Matrices

The basic idea of hierarchical matrices is to represent a given matrix \( A \in \mathbb{R}^{p \times p} \) in a block-wise low-rank format. Consider a block \( b := t \times s \) with row indices \( t \subseteq I := \{1, \ldots, p\} \) and column indices \( s \subseteq I \). Assuming that the corresponding submatrix \( A_b \) satisfies \( \text{rank}(A_b) = k \), where \( k \leq \min\{\#t, \#s\} \), we can represent this block in the form \( A_b = UV^\top \) with \( U \in \mathbb{R}^{\#t \times k} \) and \( V \in \mathbb{R}^{\#s \times k} \).

This means that instead of the entries themselves, we may store the factors \( U \) and \( V \) in order to represent the whole matrix block. The storage required by this low-rank representation is \( k(\#t + \#s) \), which is less than \( \#t \cdot \#s \) for the full representation provided that \( k \) is small enough.

In order to extend this idea to the whole matrix \( A \), we need to subdivide \( A \) into blocks which gives rise to a partition \( P \) of \( I \times I \). This partition is typically constructed in a recursive way starting with the subdivision of the block \( I \times I \) into smaller blocks \( b = t \times s \) which are either representable in low-rank format or need to be subdivided again. The recursion is stopped if the block size is below some pre-defined threshold \( n_{\text{min}} \in \mathbb{N} \). For a given maximal rank \( k \), the set of hierarchical matrices is then defined by

\[ \mathcal{H}(P, k) := \{ A \in \mathbb{R}^{p \times p} : \text{rank}(A_b) \leq k \text{ for all } b \in P \text{ with } |b| > n_{\text{min}} \} \]

For a banded matrix \( X \) it is well-known that all off-diagonal blocks of \( A = X^{-1} \) have rank at most \( k \). For more general sparsity patterns, the ability of representing blocks \( A_b \) for \( b \in P \) with a rank bounded by \( k \) will strongly depend on the pattern and the chosen partition \( P \).

![Figure 1: sparse X and X^{-1} as hierarchical matrix](image)

An illustration of the described strategy is depicted in Figure 1. Assume that after a possible reordering all non-zero entries of \( X \) are concentrated in the red blocks in the matrix from Figure 1 (left). Then a typical structure of \( A = X^{-1} \in \mathcal{H}(P, k) \) is shown in Figure 1 (right). All green blocks are representable with a rank that is small compared to the block size.
A suitable criterion to promote low-rank approximability in blocks of $X^{-1}$ can be derived from the matrix graph of $X$. To this end, we first need to characterize the sparsity pattern of $X$ in an appropriate way.

**Assumption 3.1. ([2])** There are constants $c, d > 0$ such that for all $i \in I$ it holds that

$$|U_\rho(i)| \leq c \rho^d \quad \text{for all } \rho > 0,$$

where $U_\rho(i) := \{ j \in I : d_{ij} \leq \rho \}$ and $d_{ij}$ denotes the length of a shortest path between two vertices $i, j \in I$ of the undirected graph associated with the symmetric matrix $X$.

Assumption 3.1 requires that the set of vertices that are at a distance at most $\rho$ from a given vertex has a cardinality that grows polynomially with respect to $\rho$. The degree $d$ of the polynomial growth is closely related to the intrinsic dimensionality of the graph [14]. For index sets $t, s \subset I$, we set

$$\text{diam}(t) := \max\{d_{ij} : i, j \in t\},$$

$$\text{dist}(t, s) := \min\{d_{ij} : i \in t, j \in s\}.$$

Under relatively strong assumptions, the following theorem ensures the existence of approximations of inverses of sparse matrices as hierarchical matrices.

**Theorem 3.1. ([2])** Assume that $X \in \mathbb{R}^{p \times p}$ is symmetric positive definite and satisfies Assumption 3.1. Moreover, let $P$ be a partition of $I \times I$ such that for all $b = t \times s \in P$ one of the following two conditions is satisfied:

(i) $\min(\text{diam}(t), \text{diam}(s)) \leq \eta \text{dist}(t, s)$ for some $\eta > 0$,

(ii) $\min(|t|, |s|) \leq n_{\min}$ for some $n_{\min} \in \mathbb{N}$.

Then for $\varepsilon > 0$ there exists $A \in \mathcal{H}(P, k)$ with $k \sim (\log \varepsilon / \log \gamma)^d$ such that

$$\|X^{-1} - A\|_2 \leq \varepsilon \|X^{-1}\|_2,$$

where $\gamma := (\sqrt{\kappa} - 1) / (\sqrt{\kappa} + 1)$ with $\kappa := \|X\|_2 \|X^{-1}\|_2$.

Clearly, Theorem 3.1 only yields meaningful approximation results for for well-conditioned matrices. Still, the conditions (i) and (ii) from Theorem 3.1 may offer a general strategy for constructing a suitable block structure from the graph. Alternatively, a graph partitioning algorithm such as METIS [13] can be used. This has turned out quite effective and will be used in our experiments.

We next estimate the storage requirements of a matrix in $\mathcal{H}(P, k)$.

**Lemma 3.1. ([8])** Let $P$ be a partition of $I \times I$ where each block $b \in P$ has been obtained by at most $r$ steps of a recursive subdivision of $I \times I$. Then a matrix $A \in \mathcal{H}(P, k)$ can be stored in a complexity of $c_{\rho} r p \max\{k, n_{\min}\}$, where $c_{\rho}$ is a constant depending on the pattern of the partition.

Typically, $r \sim O(\log p)$ such that the storage cost is in $O(kp \log p)$.

### 3.1 Matrix Factorization and Inversion

The key advantage of hierarchical matrices is the possibility to perform standard matrix operations in an approximate way in a complexity of $O(k^2 p \log^3 p)$ with small constants $\beta, \gamma > 0$. For a symmetric matrix $X \in \mathcal{H}(P, k)$ both an approximate $LDL^T$ decomposition and an approximation of $X^{-1}$ can be computed in a complexity of $O(k^2 p \log^3 p)$, cf. [8].

All arithmetic operations are typically defined in a block-wise way where the error in each block $b \in P$ can be controlled by the singular value decomposition. This means that for a prescribed accuracy $\varepsilon > 0$, the required rank for the low-rank representation in each block $b \in P$ can be determined adaptively. In Figure 1 (right) the number of significant singular values exceeding a relative threshold of $\varepsilon = 10^{-8}$ clearly differs from block to block.

### 3.2 Error Analysis

A relation between the block-wise low-rank approximation to the global approximation is given in the following lemma.

**Lemma 3.2.** Let $X \in \mathbb{R}^{p \times p}$ and let $P$ be a partition of $I \times I$. Assume that for $k \in \mathbb{N}$ there exists $X_H \in \mathcal{H}(P, k)$ with $\|(X - X_H)b\|_2 \leq \varepsilon \sqrt{k} b / p$ for all $b \in P$. Then

$$\|X - X_H\|_2 \leq \varepsilon.$$

**Proof.** ([8]) Following [8], we set $A := X - X_H$ and obtain

$$\|A\|_2^2 \leq \sum_{b \in P} \|A|_b\|^2 \leq \varepsilon^2 / p^2 \sum_{b \in P} \#b = \varepsilon^2.$$

Once an approximate factorization $X \approx X_H = L_H D_H L_H^T$ is available, we can also approximate the determinant by

$$\log \det(X) \approx \log \det(X_H) = \log \det(D_H),$$

where the evaluation of the last expression is trivial since $D_H$ is diagonal. The following lemma quantifies the impact of the approximation error in $X_H$ on the determinant. Its proof is given in the appendix.
Lemma 3.3. Let $X_H$ be an approximation of $X \in \mathbb{R}^{p \times p}$ such that
\[ \|I_p - X_H^{-1}X\|_2 \leq \varepsilon < 1. \]
Then
\[ |\log \det(X) - \log \det(X_H)| \leq -p \log(1 - \varepsilon). \]
Note that $\log(1 - \varepsilon) \approx -\varepsilon$ for small $\varepsilon$. The factor $p$ in the error bound of Lemma 3.3 appears to be rather pessimistic and is hardly observed in the numerical experiments.

4 Numerical Results

The aim of the numerical experiments is to illustrate the potential of hierarchical matrices within the framework of QUIC and to point out their limitations. For the synthetic examples, we fix a sparse (true) inverse covariance matrix $\Sigma^{-1}$ and to point out their limitations. For potential of hierarchical matrices within the framework

4.1 Synthetic Examples. We start with a simple model problem with a band-matrix $\Sigma^{-1}$.

Example. Let $\Sigma^{-1}$ be band was
\begin{enumerate}
  \item $\Sigma_{i,i}^{-1} = 1.25$, $\Sigma_{i,i-1}^{-1} = \Sigma_{i,i+1}^{-1} = -0.5$,
  \item $\Sigma_{i,i}^{-1} = 1.25$, $\Sigma_{i,i-1}^{-1} = \Sigma_{i,i+1}^{-1} = -0.25$,
\end{enumerate}

As we are interested in the high-dimensional case, we consider a fixed sample size of $n = 500$ and vary $p$ according to $p = 1000 \cdot 2^j$, $j = 0, \ldots, 7$. After a short experimental study it turns out that
\begin{enumerate}
  \item $\lambda = 0.5$ yields $F \approx 0.997$,
  \item $\lambda = 0.3$ yields $F \approx 0.978$,
\end{enumerate}
for all $p$ which indicates an almost optimal recovery rate. In Figure 2 we can see that the overall time for our $H$-QUIC algorithm scales quadratically with the problem size $p$ which corresponds to the complexity estimate $O((\#I_{\text{free}})^2)$ for the computation of all terms $w_i \Delta w_j^\top$ with the approximation of the Newton direction $\Delta$. Moreover, Figure 3 shows that the storage consumption of the approximate inverse $X_H^{-1}$ for the final iterate scales only linearly with $p$. This reflects the fact that all off-diagonal blocks of a $k$-banded matrix have at most rank $k$ which allows for a very efficient hierarchical matrix representation.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{timings.png}
\caption{Timings for $H$-QUIC for banded $\Sigma^{-1}$.}
\end{figure}

In the next example, we consider a scenario where – up to a small perturbation – the data are clustered.

Example. Let $\Sigma^{-1}$ be given by a random sparsity pattern with the following properties:

- the average degree at each node in the matrix graph is 4
• 99% of the edges are contained within clusters of size 20

The non-zero entries are defined such that $\Sigma^{-1}$ is positive definite. For the construction of $\Sigma^{-1}$ we use a slight modification of an example from [15].

As before, we consider a fixed sample size of $n = 500$ and vary $p$ according to $p = 1000 \cdot 2^j$, $j = 0, \ldots, 7$. In a first experiment, we study the quality of the graph resulting from $\mathcal{H}$-QUIC compared to the true $\Sigma^{-1}$. In Figure 4 we can see that for $p = 1000$ there seems to be a value of $\lambda$ that yields a maximal recovery rate for the $F$-measure which, however, does not reach the optimal value 1. This behavior was observed for all values of $p$ and might be due to the higher complexity of the graph, which is not fully captured by the limited number of samples.

We now run $\mathcal{H}$-QUIC for different values of $\lambda$ and investigate the effect on the time and storage consumption. In Figure 5 we can see that for larger values of $\lambda$ the overall time for $\mathcal{H}$-QUIC still scales quadratically with the problem size $p$, but for lower values this is no longer the case. The reason for this an increasing lack of sparsity in the iterates of QUIC which on the one hand increases the size of the free set and on the other hand leads to less favorable approximation properties for the inverses as hierarchical matrices. This is also reflected in the storage consumption of the final iterate shown in Figure 6 which for smaller values of $\lambda$ gets closer to the storage cost $\mathcal{O}(p^2)$ for the full representation of $X^{-1}$.

Interestingly, the value of $\lambda$ for which a deterioration of sparsity was detected was always quite close to the value for which an optimal $F$-measure as in Figure 4 was observed. For this example, we conclude that the search for an optimal $\lambda$ should start from above with successively decreasing values of $\lambda$ in order to keep the computational cost manageable even for high dimensions $p$. 
4.2 Comparison to BIG-QUIC. In [11] it was suggested to compute columns of the inverse of a sparse iterate \( X \) from QUIC on demand by CG. This has the advantage that the inverse does not need to be stored explicitly. In the next example we shortly analyze this strategy for a tridiagonal matrix.

**Example.** Let \( X = \text{tridiag}\{-0.5, 1.25, -0.5\} \) be of size \( p \) and compute \( X^{-1} \) with a relative accuracy of \( 10^{-8} \) by

(a) computing the columns \( w_i \) of \( X^{-1} \) by solving \( Xw_i = e_i \) using the conjugate gradient (CG) method (MATLAB command `pcg` without preconditioning);

(b) computing the columns \( w_i \) of \( X^{-1} \) by a sparse direct solver (MATLAB’s `backslash`),

(c) inverting a hierarchical matrix factorization of \( X \).

In Figure 7 we can see that for moderate \( p \) the sparse direct solver is superior to CG and the hierarchical matrix factorization. However, the complexity of both CG and the direct solver is \( O(p^2) \) which for large values of \( p \) quickly exceeds the linear complexity of the hierarchical matrix inversion.

![Figure 7: Timings for the inversion of \( X = \text{tridiag}\{-0.5, 1.25, -0.5\} \in \mathbb{R}^{p \times p} \) with relative tolerance \( 10^{-8} \) by column-wise CG, by a sparse direct solver, and by hierarchical matrices.](image)

4.3 Real World Example. To study the performance of \( \mathcal{H}\text{-QUIC} \) on a real data set, we consider a sample covariance matrix \( S \) given by the precipitation in the month of October between 1987 and 2013. The data has been extracted from a data base available at [http://www.ncdc.noaa.gov](http://www.ncdc.noaa.gov) which provides rainfall data on a global grid of \( p = 144 \times 72 = 10368 \) points. After a pre-processing step, the data for \( n = 26 \) years were available where we filled missing local values with zero.

We now run \( \mathcal{H}\text{-QUIC} \) for different values of \( \lambda \) and compare it to the publicly available C implementation of QUIC from [www.cs.utexas.edu/users/sustik/QUIC](http://www.cs.utexas.edu/users/sustik/QUIC). In Table 1 we can see that the solution is found within a minute of time as long as the sparsity does not deteriorate. Clearly, the standard QUIC algorithm still suffers from the dense computation of the inverse of \( X \). Unfortunately, there is no implementation of BIG-QUIC [11] publicly available to which we could have compared this and the other examples.

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>( \text{nnz}(X) )</th>
<th>( \mathcal{H}\text{-QUIC} ) ( \text{stor}(X_{\mathcal{H}}^{-1}) )</th>
<th>( \text{time [s]} )</th>
<th>( \text{QUIC} ) ( \text{time [s]} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>10406</td>
<td>2.44 MB</td>
<td>17.4</td>
<td>597.1</td>
</tr>
<tr>
<td>0.04</td>
<td>10548</td>
<td>2.44 MB</td>
<td>19.5</td>
<td>674.0</td>
</tr>
<tr>
<td>0.03</td>
<td>11372</td>
<td>2.59 MB</td>
<td>20.2</td>
<td>792.9</td>
</tr>
<tr>
<td>0.025</td>
<td>13202</td>
<td>3.08 MB</td>
<td>25.2</td>
<td>995.2</td>
</tr>
<tr>
<td>0.02</td>
<td>19800</td>
<td>4.54 MB</td>
<td>62.8</td>
<td>1459.4</td>
</tr>
<tr>
<td>0.015</td>
<td>116818</td>
<td>8.44 MB</td>
<td>426.8</td>
<td>2467.0</td>
</tr>
</tbody>
</table>

Table 1: Time comparison of \( \mathcal{H}\text{-QUIC} \) and standard QUIC for rainfall example with \( p = 10368 \) and \( n = 26 \).

5 Conclusions
Hierarchical matrices are an effective means to address computational bottlenecks in state-of-the-art algorithms for sparse inverse covariance estimation, provided that the sparsity pattern of the involved matrices allows for data-sparse factorizations. This has been demonstrated on a number of examples for our newly proposed \( \mathcal{H}\text{-QUIC} \) algorithm, a combination of the QUIC algorithm with hierarchical matrix techniques. We expect that equally successful combinations are possible with several other algorithms.

A Appendix

A.1 Error in \( \log \det \): Proof of Lemma 3.3.

**Proof.** We use arguments similar to [12]. Let \( E := X - X_{\mathcal{H}} \) and let \( \varrho(\cdot) \) denote the spectral radius of a matrix. Then

\[
\varrho(X_{\mathcal{H}}^{-1}E) = \varrho(X_{\mathcal{H}}^{-1}X - \text{Id}) \leq \varepsilon,
\]
where we used that the spectral radius is dominated by any operator norm. Moreover, \( X = X_H + E = X_H(Id + X_H^{-1}E) \), and hence

\[
\det(X) = \det(X_H) \det(Id + X_H^{-1}E).
\]

This implies

\[
\log \det(X) = \log \det(X_H) + \log \det(Id + X_H^{-1}E).
\]

We now make use of \( \det(A) = \exp(\text{tr}(\log A)) \) for the non-singular matrix \( A := Id + X_H^{-1}E \) to obtain

\[
|\log \det(X) - \log \det(X_H)| = |\log \det(Id + X_H^{-1}E)| = |\text{tr}(\log(Id + X_H^{-1}E))| = \left| \sum_{i=1}^{p} \log(1 + \lambda_i(X_H^{-1}E)) \right| \leq \sum_{i=1}^{p} |\log(1 + \lambda_i(X_H^{-1}E))| \leq -p \log(1 - \rho(X_H^{-1}E)) \leq -p \log(1 - \varepsilon).
\]

\[\square\]

**References**


