Banded Matrix Operators for Gaussian Markov Models in the Automatic Differentiation Era

Nicolas Durrande, Vincent Adam, Lucas Bordeaux, Stefanos Eleftheriadis and James Hensman
PROWLER.io, Cambridge, UK

Abstract

Banded matrices can be used as precision matrices in several models including linear state-space models, some Gaussian processes, and Gaussian Markov random fields. The aim of the paper is to make modern inference methods (such as variational inference or gradient-based sampling) available for Gaussian models with banded precision. We show that this can efficiently be achieved by equipping an automatic differentiation framework, such as TensorFlow or PyTorch, with some linear algebra operators dedicated to banded matrices. This paper studies the algorithmic aspects of the required operators, details their reverse-mode derivatives, and show that their complexity is linear in the number of observations.

1 Introduction

Gaussian process (GP) modelling is a popular framework for predicting the value of a (latent) function f given a limited set of input/output observation tuples. It encapsulates several common methods such as linear regression, smoothing splines and the reproducing kernel Hilbert space approximation (Rasmussen and Williams, 2006). The popularity of this framework can be explained by its efficiency when little data is available (Sacks et al., 1989), the existence of an analytical solution for the posterior when the likelihood is Gaussian, and the control over the prior that is offered by the choice the covariance function (i.e. the kernel).

Two practical limitations of GP models are that algorithms for computing the posterior distribution typically scale in $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time where n is the number of observations, and that the posterior distribution is not tractable when the likelihood is not conjugate. These two limitations have been thoroughly studied over the past decades and several approaches have been proposed to overcome them. The most popular method for reducing computational complexity is the sparse GP framework (Candela and Rasmussen, 2005; Titsias, 2009), where computations are focussed on a set of “inducing variables”, allowing a trade-off between computational requirements and the accuracy of the approximation. To cope with non-conjugacy in these models, several approximation methods such as the Laplace approximation, variational inference (VI) or expectation propagation have been proposed to approximate non-Gaussian posteriors by Gaussian distributions (Nickisch and Rasmussen, 2008).

Another angle to tackle the complexity inherent to GP models is to choose a class of covariance functions that lead to particular structures that can be exploited for storage and/or computational gain. Although several efficient methods are based on structured *covariance* matrices K (Gneiting, 2002; Nickson et al., 2015; Wilson and Nickisch, 2015), state space models (SSM, see Särkkä, 2013)—including the iconic Kalman filter (Kalman, 1960)—and Gaussian Markov random fields (GMRF, Rue and Held, 2005) are using sparse structure in the *precision* matrix $Q = K^{-1}$. We will refer to these methods using sparsity in the precision matrix as Gaussian Markov models. Exploiting this sparsity can bring orders of magnitude speed-ups compared to naive implementations based on covariance matrices. Furthermore, it can be proved that some classical covariance functions have an equivalent state space representation that leads to a sparse precision (see Section 2 and Solin, 2016).

Learning the hyper-parameters of GP models parameterised by their precision can be challenging, and it is common to resort to Markov chain Monte Carlo (MCMC) sampling Rue and Held (2005). In this context, it is however recognized that typical MCMC samplers such as Metropolis Hastings or Gibbs sampling suffer from high correlation between the latent vari-

Proceedings of the 22$^{\text{nd}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

ables (Rue et al., 2009). Deterministic approximations based on the Laplace approximation have been derived, such as the widely used integrated nested Laplace approximations (Rue et al., 2009). Our aim however is to provide general inference methods that can be applied to a broader class of models (e.g. beyond the scope of the classical combination of a Gaussian latent function with an associated likelihood).

The main contribution of this article is to show that the limitations of current inference methods for precision-based models can be overcome by implementing a small set of low level linear algebra operators dedicated to banded matrices and their derivatives in an automatic differentiation framework such as TensorFlow (Abadi et al., 2016). We propose a general framework that allows us to perform marginal likelihood estimation for models with conjugate likelihoods, Hamiltonian Monte Carlo (HMC) and VI in linear complexity both in time and space in the non-conjugate setting.

Most inference and learning algorithms in Gaussian models involve a small set of linear algebraic operations, such as matrix product, Cholesky factorisation or triangular solve. For general Gaussian models, efficient implementations of these operations and their derivatives have been proposed (Murray, 2016; Seeger et al., 2017; Giles, 2008). Tailored primitives have been designed for SSMs (Nickisch et al., 2018; Grigorievskiy et al., 2017). These however lack derivations of their reverse-mode differentiation, which prevents their use in automatic differentiation libraries. With this paper we fill this gap by introducing a set of linear algebra operations for Gaussian models with banded precisions. Compared to the dense case (i.e. precisions without band structure), our framework also includes dedicated algorithms to compute subsets of the inverses of sparse matrices (Takahashi, 1973; Zammit-Mangion and Rougier, 2018).

In this paper, we revisit and develop inference and learning algorithms in GP models with banded precisions following the wide adoption of end-to-end training of generative models using automatic differentiation. The paper is organised as follows: Section 2 gives some context and background on precision matrices with a focus on the banded case. In Section 3 we survey inference and learning algorithms for Gaussian models with banded precision and identify the basic linear algebraic operations they require. In Section 4 we describe how these operations (and their derivatives) can be efficiently implemented. In Section 5 we show on two experiments based on SSMs and GMRFs that the proposed framework scales to large problems and is proven to be attractive for real-world scenarios.

2 Background on banded precision matrices

For a Gaussian random vector g of length N with covariance matrix K, the element $K_{i,j}$ of the covariance matrix corresponds to the covariance between g_i and g_j. Elements of covariance matrices thus correspond to marginal distributions, all other variables being marginalised out. The interpretation of the elements of a precision matrix $Q = K^{-1}$ is not as straightforward, but it is still possible: $Q_{i,j}$ is a function of the conditional distribution of g_i, g_j given all other variables. More precisely, let I be a subset of $\{1, \dots, N\}$ and let $Q_{I,I}$ and g_I be the restriction of Q and g to the indices in I, then $Q_{I,I} = (\mathrm{cov}(g_I, g_I \mid g_k,\ k \notin I))^{-1}$. Taking $I = \{i\}$ shows that $Q_{i,i} = (\mathrm{var}(g_i \mid g_j,\ j \neq i))^{-1}$: contrary to the covariance case where extracting a sub-covariance amounts to marginalisation, a sub-precision corresponds to the inverse of a conditional covariance. Similarly, choosing $I = \{i, j\}$, one can show that conditional independence between g_i and g_j (given all other variables) implies $Q_{i,j} = 0$. This means that random vectors with conditional independences will lead to sparse precision matrices.

Banded matrices are sparse matrices that only have non-zero values within a small "band" around their diagonal. The lower and upper bandwidths of a banded matrix B are defined as the smaller integers l_l and l_u such that $i + l_u < j < i - l_l$ implies $B_{i,j} = 0$. For example, a tridiagonal matrix has $l_l = l_u = 1$. The bandwidth of a matrix is $l = \max(l_l, l_u)$.

In one dimension, a typical example of conditional independence resulting in banded precisions is given by random vectors with the Markov property. For example if g corresponds to the evaluations of a one-dimensional GP f with Brownian or Matérn½ covariance at increasing input locations ($g_i = f(x_i)$ with $x_i < x_j$ for $i < j$), then Q is banded with lower and upper bandwidths equal to one. In a similar fashion, one dimensional GPs with higher order Matérn kernels can also lead to banded precision matrices, but it is necessary to augment the state-space dimension by adding some derivatives. For example, the vector $g = (f(x_0), f'(x_0), f(x_1), f'(x_1), \dots, f'(x_n))$ where f is a GP with Matérn3⁄2 covariance will result in a precision with lower (and upper) bandwidth equal to three (Grigorievskiy et al., 2017).

Other kernels such as the squared exponential do not result in banded precision matrices. It is however possible to find a good approximation of the covariance such that the precision is banded as discussed by Särkkä and Piché (2014). A final one-dimensional example resulting in banded precisions are autoregressive models (Jones, 1981).

In higher dimensions, when the GP input is $x \in \mathbb{R}^d$, there is no direct equivalent of the Markov property. The classical approach is to consider a set $V = \{x_1, \dots, x_N\}$ of points $x_i \in \mathbb{R}^d$ and to define a set of undirected edges E between these points to obtain a graph structure. Now, let g be a Gaussian random vector corresponding to the evaluation of a GP h indexed by the nodes of the graph $g_i = h(x_i)$. It is then possible to have an equivalent of the Markov property where, given the values of g at the neighbouring nodes $\{k,\ (i,k) \in E\}$, g_i is independent of the rest of the graph. Assuming that, given all other entries, g_i and g_j are independent is equivalent to considering a precision matrix $Q_{i,j}$ satisfying $Q_{i,j} = 0$ if $(i,j) \notin E$. One example is the Laplacian precision $Q = D - A$, where D is a diagonal matrix with $D_{i,i} = \text{degree}(i)$ called the degree matrix, and A is the adjacency matrix: $A_{i,j} = 1$ if $(i,j) \in E$ and 0 otherwise (Belkin et al., 2004). Although this leads to a sparse precision Q, the associated bandwidth depends on the ordering of the nodes and it is possible to use heuristics such as the Cuthill McKee algorithm to find a node ordering associated to a thin bandwidth (Rue and Held, 2005).

3 Fast inference with banded precisions

In this section we look at three inference techniques and investigate the banded matrix operations that are required for each case. Let $X = [x_1, \dots x_n]^T$ with $x_i \in \mathcal{D} \subseteq \mathbb{R}^d$ and $Y = [y_1, \dots, y_n]^T$ with $y_i \in \mathbb{R}$ denote matrices corresponding to input and output values of the data. We consider the following type of models: a latent function f is defined over $\mathcal{D}$; the prior on the latent function is parameterised by θ; given the latent function, an observation model provides a likelihood that factorises as $p(Y|f) = \prod_{i=1}^n p(y_i|F_i)$, with $F = f(X)$.

3.1 Marginal likelihood computation in tractable problems

In the case where the likelihood is Gaussian with variance τ^2, the common approach for estimating the model parameters θ is to maximise the marginal likelihood $p(Y|\theta)$. This requires computing the prior distribution of the latent function at locations where observations are provided. This is straightforward when the latent function is parameterised by its covariance, as in a Gaussian process model, but it scales cubically with the number of observations. Grigorievskiy et al. (2017) show that it is possible to do this computation efficiently for an SSM with banded precision. The formulation of the SSM allows them to compute the precision matrix for an arbitrary subset of the total input locations. In the case of GMRF, the precision matrix Q is specified for the all the nodes of the graph, it thus has a size $N \times N$ even if the n observations are only associated to a subset of nodes. We show below that the approach of Grigorievskiy et al. (2017) can be generalised to this case. The covariance matrix of $f(X)$ is $K = EQ^{-1}E^T$ where E is an $n \times N$ matrix of 0 and 1 that selects the appropriate rows of Q^{-1}. Using the matrix inversion and the matrix determinant lemma, we obtain:

$$\begin{aligned}
\log\, p(Y|\theta) &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|EQ^{-1}E^T + \tau^2 I| \\
&\quad - \frac{1}{2}Y^T(EQ^{-1}E^T + \tau^2 I)^{-1}Y \\
&= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|Q + \tau^{-2}E^T E| \\
&\quad + \frac{1}{2}\log|Q| - \frac{1}{2}\log|\tau^2 I| - \frac{1}{2\tau^2}Y^T Y \\
&\quad + \frac{1}{2\tau^4}Y^T E(Q + \tau^{-2}E^T E)^{-1}E^T Y. \quad (1)
\end{aligned}$$

The definition of E implies that $E^T E$ is a diagonal matrix so $Q + \tau^{-2}E^T E$ has the same bandwith as Q. The Cholesky factors of banded positive-definite matrices are lower triangular banded matrices that can be computed efficiently (see Section 4). We thus introduce $LL^T = (Q + \tau^{-2}E^T E)$ and $L_Q L_Q^T = Q$ to obtain the following expression of the marginal likelihood:

$$\begin{aligned}
\log\, p(Y|\theta) &= -\frac{n}{2}\log(2\pi) - \log|L| + \log|L_Q| \\
&\quad - \frac{n}{2}\log\tau^2 - \frac{1}{2\tau^2}Y^T Y \\
&\quad + \frac{1}{2\tau^4}Y^T E L^{-T}L^{-1}E^T Y. \quad (2)
\end{aligned}$$

As a consequence, the computation of the marginal likelihood requires the efficient computation of the Cholesky factorisation of banded matrices and the efficient solution of linear systems with banded triangular matrices such as $L^{-1}(E^T Y)$ in Eq. 2.

3.2 Gradient-based MCMC

Asymptotically exact inference in non-conjugate models is achievable through MCMC sampling. Among available algorithms, the most empirically effective are HMC (see e.g., Neal et al., 2011) and its variants (Hoffman and Gelman, 2014). These samplers require the log joint density of the latent variables and the data, $\log\, p(F, Y|\theta)$, as well as its derivatives with respect to the latent variables. Here we investigate which operations involving banded matrices are required for this purpose.

To avoid strong correlations in the joint distribution, which reduces the effectiveness of the sampler, whitening of the latent variables is often employed (see e.g., Filippone et al., 2013). Let $v \sim \mathcal{N}(0, I)$ be a random vector of length N. One can generate samples of $f(X)$ by computing $F = L_K v$ where L_K is the Cholesky factor of the covariance matrix K of $f(X)$. When working with a precision Q, we can instead write $F = L_Q^{-T} v$, resulting in $p(F) = \mathcal{N}(0, Q^{-1})$ as required.

The log joint density is then:

$$\log p(v, \theta, Y) = \log p(v) + \log p(\theta) + \sum_{i=1}^{n} \log p(y_i | \theta, (L_Q^{-T} v)_i). \quad (3)$$

The first two terms of this expression are straightforward to compute. The main computational challenge is the Cholesky decomposition of the precision matrix Q and solving the linear system $f(X) = L_Q^{-T} v$. We discuss in Section 4 how these operators (and their derivatives) can be implemented to make HMC efficient for banded precision matrices.[1]

3.3 Variational inference

Variational inference achieves approximate inference by maximising a lower bound to the marginal likelihood over a family of tractable distributions (Blei et al., 2017). We here derive the necessary banded matrix operations that are needed for VI.

Let $F = f(X) \in \mathbb{R}^n$. We assume a Gaussian prior $F \sim \mathcal{N}(m_p, Q_p^{-1})$ with banded precision, where m_p and $Q_p = L_p L_p^T$ may depend on parameters θ. We also choose a Gaussian distribution $\mathcal{N}(m_q, Q_q^{-1})$ with banded precision to approximate the posterior $F|Y$. We denote its probability density by q, and we parameterise it by its mean m_q and the Cholesky factor L_q of Q_q.

We finally assume that the likelihood factorises to obtain the following log-likelihood lower bound as our variational objective:

$$\log p(Y) \geq \mathbb{E}_{q(F)} \log \frac{p(Y, F)}{q(F)} = \sum_{i=1}^{n} \mathbb{E}_{q(F_i)} \log p(Y_i | F_i) - \mathrm{KL}[q \,\|\, p]. \quad (4)$$

The first term of Eq. 4 only depends on the marginal distributions $q(F_i)$, which are described by m_q and the diagonal values of Q_q^{-1}. Although Q_q^{-1} is typically a dense matrix, its diagonal values can be obtained efficiently with the *sparse inverse subset* method discussed in Section 4.1. Depending on the likelihood, $\log p(Y_i|F_i)$ may or may not have a closed form. If no analytical expression is available, the problem typically boils down to the numerical approximation of one-dimensional integrals which can be done via Monte Carlo sampling or quadrature methods (Hensman et al., 2015). Regarding the Kullback–Leibler divergence term in Eq. 4, it can be expressed as:

$$\mathrm{KL}[q \,\|\, p] = \frac{1}{2}\Big(\mathrm{tr}(Q_q^{-1} Q_p) + 2 \sum_i \log[L_q]_{ii} - \log[L_p]_{ii} + (m_p - m_q)^T L_p L_p^T (m_p - m_q) - N\Big). \quad (5)$$

The trace term in this expression can be computed as the sum of an element-wise product between Q_q^{-1} and Q_p. Since Q_p is banded, it is sufficient to compute only the elements of Q_q^{-1} that lie inside the band of Q_p. Here again, we can use the *sparse inverse subset* operator.

4 Banded low level operators

In the previous section we collected a list of the operators needed to perform efficient inference for GP models with banded precisions. We now show that these operators can be implemented efficiently with a complexity at most $\mathcal{O}(Nl^2)$ for $N \times N$ matrices of bandwidth l. Furthermore, we derive expressions for the reverse mode differentiation of these operators that also have linear complexity in N.

4.1 Description of operators

Cholesky decomposition. $\mathbb{C}: \; Q \to L \; s.t. \; LL^T = Q$. One fundamental property of the Cholesky decomposition of a banded matrix B (assumed symmetric and positive-definite) is that it returns a lower triangular matrix with the same number of sub-diagonals as B. Its implementation for a banded matrix is similar to the dense case (see Algorithm 1 in Appendix B) but its complexity is $\mathcal{O}(Nl^2)$ instead of $\mathcal{O}(N^3)$, as detailed by Rue and Held (2005).

Triangular solve. $\mathbb{S}: \; (L, v) \to L^{-1}v$. It turns out that the algorithm is similar to a classic triangular solve algorithm running in nested loops through all rows and columns of L, but the inner loops on columns can be started at the beginning of the band. This leads to a $\mathcal{O}(Nl)$ complexity instead of $O(N^2)$ for dense matrices (Rue and Held, 2005, p.45).

Sparse inverse subset. $\mathbb{I}: \; L \to (Q^{-1})$. Although the inverse of a banded matrix is often a dense matrix,

[1] Faulkner and Minin (2018) report that they successfully used HMC for sampling from GMRF models using the probabilistic programming language Stan (Carpenter et al., 2017). Their approach does not take advantage of the sparsity that is found in GMRF precision matrices, so this sampling is not efficient for large models. This specific topic is however currently under discussion between the Stan developers (Simpson and Vehtari, 2017).

Takahashi's algorithm (Takahashi, 1973) shows that it is possible to compute only the band elements of Q^{-1}. The pseudo code for this operator is given by Algorithm 3 in Appendix B, and it results in a $\mathcal{O}(Nl^2)$ complexity.

Products. The matrix product is another operation that preserves bandedness: the resulting lower bandwidth is the sum of the lower bandwidths of the inputs (and similarly for the upper bandwidth). This operator is denoted by $\mathbb{P} : B_1, B_2 \to B_1 B_2$ and its complexity is $\mathcal{O}(Nl^2)$. We additionally need the following basic linear algebra operations: *product* between a banded matrix and a vector $\mathbb{P} : B, v \to Bv$, that is $\mathcal{O}(Nl)$; and *outer product* of two vectors ($\mathbb{O} : m, v \to mv^T$). The latter typically yields a dense matrix and it has a $\mathcal{O}(N^2)$ complexity. Although this may seem problematic, we only require in our applications a small band of this dense matrix, which can be computed with a cost that is linear in N.

Note, also, that although we are primarily interested in lower-banded matrices, various operations require some matrix *transposes* (examples below with the expressions for various gradients). This forces the implementation to deal with several variants of each algorithm, such as solving linear systems with lower-banded or upper-banded matrices.

4.2 Derivatives of the operators

We endow each operator with a method implementing its *reverse-mode differentiation* (see Appendix A). Given a chain of operations resulting in a scalar value (say $X \to Y \to c$), it consists in propagating a downstream gradient ($\frac{dc}{dY}$) to an upstream gradient ($\frac{dc}{dX}$). This approach has two main advantages: (1) it is *compositional*, which allows the gradients to be obtained for arbitrarily complex models based on our banded matrix operators; (2) it is *efficient*: the execution time of the reverse mode differentiation of a model takes a time proportional to its forward evaluation.

Following the literature (Giles, 2008; Murray, 2016), we denote by $\bar{X}$ the "reverse-mode sensitivities", or gradients computed in reverse-mode on the output X of an operator. In our previous example, $\bar{X} = \frac{dc}{dX}$.

Basic operators The expressions of the gradients of all product and solve operators are derived in Appendix A and summarised in Table 1. They can all be defined as a simple composition of the forward evaluation of banded operators. For example, the gradient of a *solve* is defined using solve and product operations only.

A few points are worth noting: (1) The notation introduced in Table 1 is overloaded: $\mathbb{P}$ denotes, for instance, a banded product where the right-hand side denotes either a banded matrix or a vector, which is clear from the context. (2) All intermediate terms in the expressions of the reverse-mode sensitivities column can be kept banded, throughout their evaluation. Consequently, the gradients of all operations can be computed in time proportional to the forward evaluation, which means that the gradients in Table 1 have at most a $\mathcal{O}(Nl^2)$ complexity.

Cholesky and sparse subset inverse. To define the gradients of the Cholesky and subset inverse operators, we had to use a different approach. For both operators, one can define an analytical expression for the reverse-mode sensitivities $(-2\mathbb{S}(L, \mathbb{P}(\bar{\Sigma}^T, (LL^T)^{-1}))$ for subset inverse; see Murray (2016) for Cholesky). However, evaluating these terms requires the computation of dense matrices and scales as $\mathcal{O}(N^3)$. Such an approach would thus lead to a computational bottleneck and render our scheme less efficient.

For the Cholesky reverse-mode differentiation we used an existing gradient computation algorithm that was easy to adapt to a banded representation (Giles, 2008), and is described as Algorithm 2 in our Appendix. For the sparse subset inverse operator, we did not find a gradient computation algorithm in the literature that could be customized for banded matrices. We therefore used the Tangent software package (van Merriënboer et al., 2018), to generate a gradient computation algorithm from the forward computation code. We then hand-curated the generated code, obtaining Algorithm 4 in Appendix B. Both algorithms have a $\mathcal{O}(Nl^2)$ complexity.

4.3 Storage footprint

Using Gaussian process models usually requires the storage of covariance matrices of size $n \times n$, where n is the number of observations. This is usually the limiting factor, and the maximum number of observations that can be handled on currently available desktop computers is typically in the range $n \in [10^4, 10^5]$.

When working with symmetric or lower triangular banded matrices, it is of course sufficient to store only the non-zero elements of the lower half of the matrix. In our implementation of the operators described above, all the operators use the following convention for their inputs and outputs: let B be a banded matrix of size $n \times n$ which has lower and upper bandwidths equal to (l_l, l_u), we store B as an $(l_l + l_u + 1) \times n$ matrix B' with $B_{i,j} = B'_{i-l_u-j+1,j}$ for $1 \leq i, j \leq n$. Note that the values of B' located in the upper-left and lower-right corners may not be defined but they are never accessed by algorithms in practice.

Table 1: Summary of the reverse mode sensitivities with analytical expression.

Operator	Symbol	Input	Forward	Reverse Mode Sensitivities
product matrix-matrix	$\mathbb{P}$	B_1, B_2	$P = B_1 B_2$	$\bar{B}_1 = \mathbb{P}(\bar{P}, B_2^T)$ $\quad \bar{B}_2 = \mathbb{P}(B_1^T, \bar{P})$
product matrix-vector	$\mathbb{P}$	B, v	$p = Bv$	$\bar{B} = \mathbb{O}(\bar{p}, v)$ $\quad \bar{v} = \mathbb{P}(B^T, \bar{p})$
vector outer product	$\mathbb{O}$	m, v	$O = mv^T$	$\bar{m} = \mathbb{P}(\bar{O}, v)$ $\quad \bar{v} = \mathbb{P}(\bar{O}^T, m)$
solve matrix-vector	$\mathbb{S}$	L, v	$s = L^{-1}v$	$\bar{v} = \mathbb{S}(L^T, \bar{s})$ $\bar{L}^T = -\mathbb{O}(\mathbb{S}(L, v), \mathbb{S}(L^T, \bar{s}))$
solve matrix-matrix	$\mathbb{S}$	L, B	$S = L^{-1}B$	$\bar{B} = \mathbb{S}(L^T, \bar{S})$ $\bar{L}^T = -\mathbb{P}(\mathbb{S}(L, B), \mathbb{S}(L^T, \bar{S})^T)$

5 Experiments

5.1 Implementation

We have implemented the banded operators described in Section 4 as custom operators for TensorFlow. This allowed us to experiment with complex GP models using these operators together with the functionalities of the GPflow library (Matthews et al., 2017).

Our implementation is based on TensorFlow's extensibility mechanism referred to as "custom ops": the code for the forward evaluation of each operator is written in C++ and registered to TensorFlow, together with a mix of Python and C++ code that implements the gradients of each operator. The C++ code for forward evaluation is a direct implementation of the algorithms detailed in Appendix B. Most gradients can be written in Python by calling other operators, following Table 1. The Cholesky and sparse subset operators require dedicated C++ code, as explained in Section 4.2 and detailed in Appendix B.

5.2 Computational time

The aim of this section is to confirm that models written using our operators are faster to train than existing alternatives. We have seen previously that the complexity of the proposed operators is $\mathcal{O}(Nl^2)$ where N is the size of the precision and l is the bandwidth. We now illustrate the influence of these parameters on the time required for computing the log-likelihood and its gradient for a GP regression model with Gaussian likelihood.

The weekly average atmospheric CO_2 concentrations recorded at the Mauna Loa Observatory, Hawaii, by Keeling and Whorf (2005) are commonly used to demonstrate how different patterns observed in the data (e.g. periodicity, increasing trend) can be encoded in a GP prior by designing compositional kernel functions (Rasmussen and Williams, 2006). We ensure in this example the bandedness of the precision matrix by working within the class of kernels that have a state-space representation (Grigorievskiy et al., 2017). Conveniently, finite sums and products of such kernels belong to this class (with each composition increasing the resulting state-space dimension). For this experiment, we design our kernel as follows: to capture the slow varying trend we use a Matérn3⁄2 kernel $k_s(\tau)$ parameterized by a lengthscale l_s and a variance σ_s^2; for the quasi-periodic trend, we follow Solin and Särkkä (2014) and use the quasi-periodic kernel $k_{q\text{-}per}(\tau) = k_q(\tau) \sum_{j=1}^{J} \cos(2\pi j f_0 \tau)$, where f_0 is the frequency of the periodic trend and k_q is a Matérn1⁄2 kernel with lengthscale l_q and variance σ_q^2. These kernels have state-space dimensions of respectively 2 and $2J$, so the state dimension of $k = k_s + k_{q-per}$ is $2J + 2$ and the bandwith of the resulting precision matrix is $4J + 3$.

The regression model using this kernel can be implemented in three different ways that we are going to compare: the first one is a Kalman filter implementation using loops in TensorFlow; the second one is based on our custom operators and uses the bandedness of the model's precision matrix; and the third one is the GPflow implementation of classic GP regression which uses the dense covariance matrix. Note that the first two implementations exploit the Markov property of the model and are thus expected to scale linearly with the amount of data points, whereas the third is known to scale cubically. These algorithms are implemented using TensorFlow and are detailed in Appendix C.

The dataset contains $n = 3082$ observations and we consider subsets of increasing sizes to time the evaluation of the marginal likelihood and its gradient with respect to the model parameters (Figure 1, left). As expected, the computational time for the naive GP regression models quickly becomes prohibitively large whereas the implementations based on the state-space representation are linear in time. The computational speed-up for custom operators is of three orders of magnitude for the full dataset.

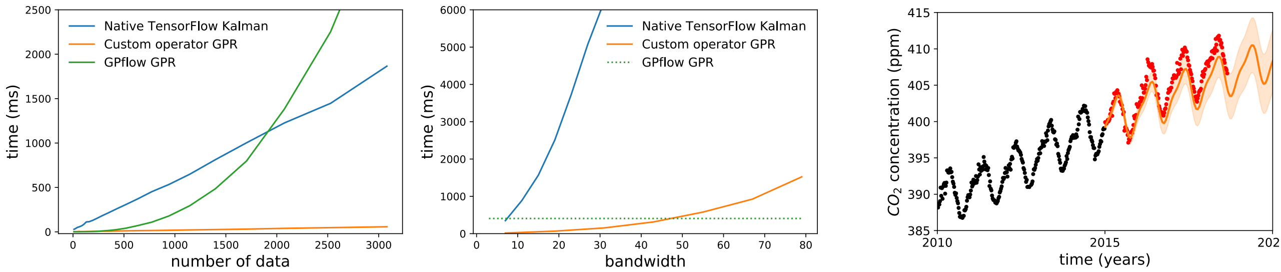


Figure 1: Mean execution time for three implementations of a GP regression model as a function of the number of data points (left) and of the bandwidth (middle). The right panel shows the model predictions when the data points after 2015 are excluded from the training dataset.

Similarly, we show in the middle panel the influence of the bandwidth l (which is twice the state-space dimension d minus one) on the execution time, restricting the dataset to the first $n = 1500$ points. To do so we vary the state dimension of the model by increasing J, that is by adding more harmonics to our quasi-periodic kernel. One can see that our banded operators are much faster than the Kalman implementation, and that it favourably compares to the GPflow implementation when the bandwidth is smaller than 45. Furthermore, this threshold would increase quickly as the dataset gets larger. With our operators, the likelihood evaluation scales as $\mathcal{O}(Nl^2)$, but we now have $N = nd$ so the operators complexity is $\mathcal{O}(nl^3)$. Note that this is the same complexity as the classic implementation of the Kalman filter.

Finally, Figure 1 (right) shows the model predictions ($J = 2$, $l = 11$) for the years from 2015 to 2020. We use the implementation based on our custom operators and we learn the kernel parameters by optimizing the marginal likelihood of the model given the data from 1958 until 2015. This illustrates that the model can account for complex patterns even with a small bandwidth. The mean test log-likelihood of this model is -1.75 whereas we obtain -1.56 with the reference implementation (Rasmussen and Williams, 2006, Eq. 5.19). Although, this is slightly to the advantage of the later, it means that a model with small bandwith can have good prediction abilities, even when it is not finely tuned for the dataset at hand.

5.3 Gaussian Markov random field

In this section we illustrate our ability to perform inference on a GMRF with non-conjugate likelihood. To this aim, we consider the Porto dataset that gathers the GPS locations of taxi pick-ups in the city of Porto for the period July 2013 - June 2014. We use the first three weeks of the data as our training set and the following three weeks as our test set. This dataset has already been modeled successfully with GP based Cox processes by John and Hensman (2018) but we choose a different approach here: we consider a GP based Cox process model defined on a graph representing the road network and each data point is projected onto the closest node (if it is within a 10m radius). The main advantage of this approach is that the GP covariances are using the graph distance, which are more meaningful that the Euclidian distance (think about two locations separated by the river).

The graph is an undirected graph obtained from open street map, it is denoted by $G = (V = \{1, \dots, N\}, E)$ and it consists of $N = 11284$ nodes and $\#V = 12185$ edges. The length (in meters) of the edge $(i, j) \in E$ is denoted by $d_{i,j}$.

Let $f \sim \mathcal{N}(0, Q^{-1})$ be a latent GMRF indexed by the nodes $v \in V$. Our generative model assumes that the number of pick-up associated to a node i follows a Poisson distribution with parameter $\exp(f_i)w_i$, where w_i is the length of the edges associated to the node i: $w_i = \sum_{j,(i,j)\in E} \max(10, d_{i,j}/2)$.

Since Q can also be interpreted as an inner product for vectors $g, h \in \mathbb{R}^N$: $\langle g, h\rangle = g^T Q h$, we define Q such that it corresponds to the sum of Matérn½ inner products over all the edges (Durrande et al., 2016):

$$g^T Q h = \frac{1}{\sigma^2} \sum_{(i,j)\in E} \frac{1}{1-\lambda_{i,j}} (g_i \ g_j) \begin{pmatrix} 1 & -\lambda_{i,j} \\ -\lambda_{i,j} & 1 \end{pmatrix} \begin{pmatrix} h_i \\ h_j \end{pmatrix} - \frac{1}{2} g_i h_i - \frac{1}{2} g_j h_j \qquad (6)$$

where $\lambda_{i,j} = \sigma^2 \exp(-d_{i,j}/\ell)$ with $\sigma^2 = 10$, $\ell = 10^4$.

We now compare three methods for predicting the values of the latent function f given the observations of y_v for $v \in V$: a Hamiltonian Monte-Carlo sampler, a variational inference method, and a baseline consisting in estimating the rate of a Poisson random variable independently for each node. In the first two cases, we use our implementation based on the GPflow framework together with the specialised operators for banded matrices described in Section 4. The first step before actually building the models consists in finding a good ordering of the nodes in order to reduce the bandwidth of the precision matrix. Using the Cuthill McKee algorithm from the Scipy library, we found an ordering corresponding to a lower bandwidth of $l = 117$ for matrix Q.

Since our inference methods has a runtime of $\mathcal{O}(Nl^2)$, as opposed to $\mathcal{O}(N^3)$ for a dense representation, the settings $N = 11284$ and $l = 117$ imply that our banded framework saves runtime by a factor of roughly 10^4. Similarly, having to store only the band of the matrices instead of their dense versions allows us to save almost two orders of magnitude on the storage footprint.

Figure 2 (bottom) shows the mean prediction of the model trained using variational inference. One can see that the model successfully extracts a smooth trend for the latent variable f. However the non-linear mapping from the latent function to the rate $\lambda_i = \exp(f_i)w_i$ leads to a large range of predicted rates: for most nodes, the predicted rate is below 5, but its maximum value is 149. We investigated some of the locations with large predicted rate and they all correspond to particular landmarks such as hotels or hospitals where the taxi demand is naturally high and sharp. The plots we obtain for the HMC and the baseline predictions are very similar so we do not reproduce them here.

Finally, we compare the likelihood of the three models on the test set. The values of the log-likelihood for the VI, HMC and the baseline are -15778.5, -15873.6 and -17146.6 respectively. This shows that even for this challenging dataset, the proposed model has powerful predictive power.

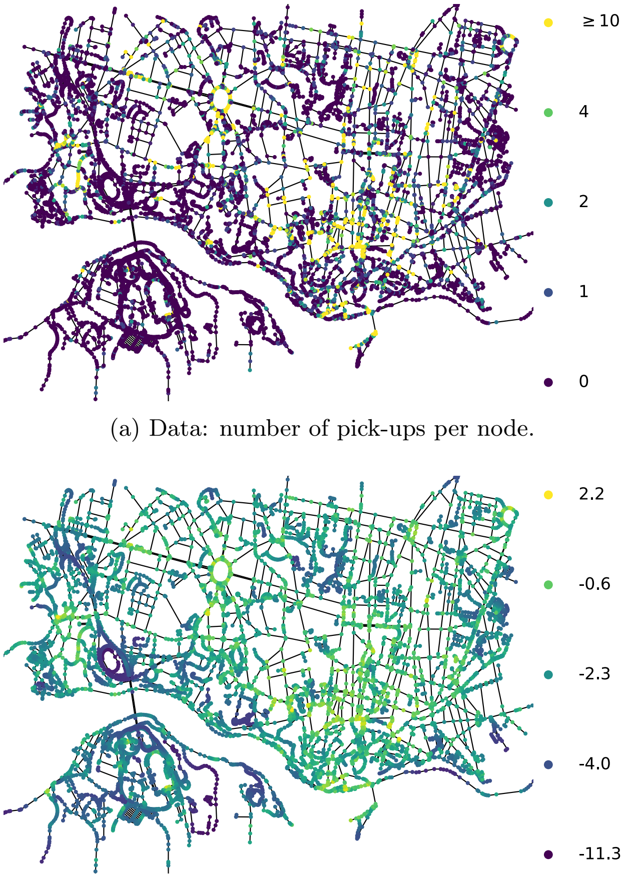
6 Discussion and conclusion

This work has examined how some Gaussian Markov models can be expressed using banded matrices. We investigated some inference and end-to-end learning procedures for these models and identified the shared set of general banded operators—endowed with their reverse-mode derivatives—necessary to implement them.

The framework we propose is general in the sense that it covers a large class of models (Kalman, SSM, GMRF) for which it provides several state of the art inference methods such as variational inference, gradient-based samplers and maximum likelihood estimation when a subset of the variables are observed. More inference algorithms such as expectation propagation readily fit into this paradigm.

The only algorithm we could not find in the literature is the differentiation of the sparse inverse subset $\mathbb{I}$. Given that all others were readily available, we believe that the implementation of a few low level operators is a small price to pay for the huge benefit they provide in practice. Although we focused on matrices where the precisions are banded, similar work could be carried out for matrices with other sparsity patterns such as the ones obtained when the nested dissection is used for the graph node ordering (Rue and Held, 2005).

We anticipate two main outcomes for the present work. The first one is a strong incentive for a better support for the sparse algebra in automatic differentiation frameworks, and the second is a renewal of the popularity of Gaussian Markov models now that state of the art inference methods are available.



(a) Data: number of pick-ups per node.

(b) Predicted latent.

Figure 2: Graphs of the Porto experiment. (a) Number of pick-ups after clipping them to the graph. The range of the data is [0, 160] but we choose a non-linear color map. (b) Predicted latent function for the VI model.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory (COLT)*, pages 624–638. Springer, 2004.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112 (518):859–877, 2017.

Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

Nicolas Durrande, James Hensman, Magnus Rattray, and Neil D Lawrence. Detecting periodicities with Gaussian processes. *PeerJ Computer Science*, 2:e50, 2016.

James R. Faulkner and Vladimir N. Minin. Locally adaptive smoothing with Markov random fields and shrinkage priors. *Bayesian Analysis*, 13(1):225–252, 2018.

Maurizio Filippone, Mingjun Zhong, and Mark Girolami. A comparative evaluation of stochastic-based inference methods for Gaussian process models. *Machine Learning*, 93(1):93–114, 2013.

Mike B Giles. Collected matrix derivative results for forward and reverse mode algorithmic differentiation. In *Advances in Automatic Differentiation*, pages 35–44. Springer, 2008.

Tilmann Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83(2): 493–508, 2002.

Alexander Grigorievskiy, Neil Lawrence, and Simo Särkkä. Parallelizable sparse inverse formulation Gaussian processes (SpInGP). In *27th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2017.

James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics, (AISTATS)*, pages 351–360, 2015.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

ST John and James Hensman. Large-scale Cox process inference using variational Fourier features. In *International Conference on Machine Learning*, pages 2367–2375, 2018.

Richard H Jones. Fitting a continuous time autoregression to discrete data. In *Applied time series analysis II*, pages 651–682. Elsevier, 1981.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

CD Keeling and TP Whorf. Atmospheric carbon dioxide record from mauna loa. *Carbon Dioxide Research Group, Scripps Institution of Oceanography, University of California La Jolla, California*, pages 92093–0444, 2005.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Iain Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016.

Radford M Neal et al. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2 (11), 2011.

Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

Hannes Nickisch, Arno Solin, and Alexander Grigorevskiy. State space Gaussian processes with non-Gaussian likelihood. In *International Conference on Machine Learning*, pages 3786–3795, 2018.

Thomas Nickson, Tom Gunter, Chris Lloyd, Michael A Osborne, and Stephen Roberts. Blitzkriging: Kronecker-structured stochastic Gaussian processes. *arXiv preprint arXiv:1510.07965*, 2015.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006.

Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested laplace approximations. *Journal of the royal statistical society: Series B (statistical methodology)*, 71(2):319–392, 2009.

Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

Simo Särkkä and Robert Piché. On convergence and accuracy of state-space approximations of squared exponential covariance functions. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

Matthias Seeger, Asmus Hetzel, Zhenwen Dai, and Neil D Lawrence. Auto-differentiating linear algebra. *arXiv preprint arXiv:1710.08717*, 2017.

Daniel Simpson and Aki Vehtari. Sparse matrices for Stan. Technical report, October 2017. URL `http://discourse.mc-stan.org/t/a-proposal-for-sparse-matrices-and-gps-in-stan/2183`.

Arno Solin. *Stochastic differential equation methods for spatio-temporal Gaussian process regression.* PhD thesis, Aalto University, 2016.

Arno Solin and Simo Särkkä. Explicit link between periodic covariance functions and state space models. In *Artificial Intelligence and Statistics*, pages 904–912, 2014.

Kazuhiro Takahashi. Formation of sparse bus impedance matrix and its application to short circuit study. In *Proc. PICA Conference, June, 1973*, 1973.

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics, (AISTATS)*, pages 567–574, 2009.

Bart van Merriënboer, Dan Moldovan, and Alexander Wiltschko. Tangent: Automatic differentiation using source-code transformation for dynamically typed array programming. In *Advances in Neural Information Processing Systems*, pages 6259–6268, 2018.

Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning (ICML)*, pages 1775–1784, 2015.

Andrew Zammit-Mangion and Jonathan Rougier. A sparse linear algebra algorithm for fast computation of prediction variances with Gaussian Markov random fields. *Computational Statistics & Data Analysis*, 123:116–130, 2018.