Appendices

A Details on the doubly sparse variational Gaussian process

A.1 Conditional for Markovian Gaussian processes

We consider a stationary Markovian GP with state dimension d and denote by $(\boldsymbol{u}_-, \boldsymbol{s}, \boldsymbol{u}_+)$ its evaluation on the triplet (z_{n_-}, t, z_{n_+}). We here detail the derivation of $p(\boldsymbol{s}|\boldsymbol{v} = [\boldsymbol{u}_-, \boldsymbol{u}_+])$

Derivation from the joint precision

$$
\begin{aligned}
p(\boldsymbol{s}|\boldsymbol{u}_-, \boldsymbol{u}_+) &\propto p(\boldsymbol{s}|\boldsymbol{u}_-)p(\boldsymbol{u}_+|\boldsymbol{s}) \\
&\propto \mathcal{N}(\boldsymbol{s}; \boldsymbol{A}_{n_-,t}\boldsymbol{u}_-, \boldsymbol{Q}_{n_-,t})\mathcal{N}(\boldsymbol{u}_+; \boldsymbol{A}_{t,n_+}\boldsymbol{s}, \boldsymbol{Q}_{t,n_+}) \\
&\propto \exp -\frac{1}{2}\left[||\boldsymbol{s} - \boldsymbol{A}_{n_-,t}\boldsymbol{u}_-||^2_{\boldsymbol{Q}^{-1}_{n_-,t}} + ||\boldsymbol{u}_+ - \boldsymbol{A}_{t,n_+}\boldsymbol{s}||^2_{\boldsymbol{Q}^{-1}_{t,n_+}}\right] \\
&\propto \exp -\frac{1}{2}\big[\boldsymbol{s}^\top \underbrace{(\boldsymbol{Q}^{-1}_{n_-,t} + (\boldsymbol{A}_{t,n_+})^\top \boldsymbol{Q}^{-1}_{t,n_+}\boldsymbol{A}_{t,n_+})}_{\boldsymbol{T}^{-1}}\boldsymbol{s} - 2\boldsymbol{s}^\top \underbrace{\big[\boldsymbol{Q}^{-1}_{n_-,t}\boldsymbol{A}_{n_-,t}, \quad \boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{t,n_+}\big]}_{\boldsymbol{M}=[\boldsymbol{M}_1,\boldsymbol{M}_2]}\boldsymbol{v}\big] \\
&\propto \exp -\frac{1}{2}\left[\boldsymbol{s}^\top \boldsymbol{T}^{-1}\boldsymbol{s} - 2\boldsymbol{s}^\top \boldsymbol{M}\boldsymbol{v}\right] = \mathcal{N}(\boldsymbol{s}; \boldsymbol{P}\boldsymbol{v}, \boldsymbol{T})
\end{aligned}
$$

with

$$
\begin{aligned}
\boldsymbol{T} &= (\boldsymbol{Q}^{-1}_{n_-,t} + \boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{t,n_+}\boldsymbol{A}_{t,n_+})^{-1} \text{ (Woodbury identity)} \\
&= \boldsymbol{Q}_{n_-,t} - \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}(\boldsymbol{Q}_{t,n_+} + \boldsymbol{A}_{t,n_+}\boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+})^{-1}\boldsymbol{A}_{t,n_+}\boldsymbol{Q}_{n_-,t} \\
&= \boldsymbol{Q}_{n_-,t} - \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{n_-,n_+}\boldsymbol{A}_{t,n_+}\boldsymbol{Q}_{n_-,t}
\end{aligned}
$$

and $\boldsymbol{P} = [\boldsymbol{P}_1, \boldsymbol{P}_2] = \boldsymbol{T}\boldsymbol{M} = [\boldsymbol{T}\boldsymbol{M}_1, \boldsymbol{T}\boldsymbol{M}_2]$ given by

$$
\begin{aligned}
\boldsymbol{P}_1 &= (\boldsymbol{Q}_{n_-,t} - \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{n_-,n_+}\boldsymbol{A}_{t,n_+}\boldsymbol{Q}_{n_-,t})\boldsymbol{Q}^{-1}_{n_-,t}\boldsymbol{A}_{n_-,t} \\
&= \boldsymbol{A}_{n_-,t} - \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{n_-,n_+}\boldsymbol{A}_{n_-,n_+} \\
\boldsymbol{P}_2 &= (\boldsymbol{Q}_{n_-,t} - \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{n_-,n_+}\boldsymbol{A}_{t,n_+}\boldsymbol{Q}_{n_-,t})\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{t,n_+} \\
&= \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{t,n_+} - \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{n_-,n_+}(\boldsymbol{Q}_{n_-,n_+} - \boldsymbol{Q}_{t,n_+})\boldsymbol{Q}^{-1}_{t,n_+} \text{ (Woodbury identity)} \\
&= \boldsymbol{Q}_{n_-,t}\boldsymbol{A}^\top_{t,n_+}\boldsymbol{Q}^{-1}_{n_-,n_+}
\end{aligned}
$$

Derivation from the joint covariance

Another derivation using the covariance approach. One can write down the joint density

$$
p(\boldsymbol{s}, \boldsymbol{v}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{\boldsymbol{s}} \\ \boldsymbol{\mu}_{\boldsymbol{v}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{ss}} & \boldsymbol{\Sigma}_{\boldsymbol{sv}} \\ \boldsymbol{\Sigma}_{\boldsymbol{vs}} & \boldsymbol{\Sigma}_{\boldsymbol{vv}}\end{bmatrix}\right)
$$

with

$$
\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{s}} &= \boldsymbol{\mu}_{\boldsymbol{v}} = \boldsymbol{0} \\
\boldsymbol{\Sigma}_{\boldsymbol{ss}} &= \boldsymbol{P}_0 \\
\boldsymbol{\Sigma}_{\boldsymbol{vv}} &= \begin{bmatrix}\boldsymbol{P}_0 & \boldsymbol{A}^T_{n_-,n_+}\boldsymbol{P}_0 \\ \boldsymbol{P}_0\boldsymbol{A}_{n_-,n_+} & \boldsymbol{P}_0\end{bmatrix} \\
\boldsymbol{\Sigma}_{\boldsymbol{sv}} &= \begin{bmatrix}\boldsymbol{A}^T_{n_-,t}\boldsymbol{P}_0, & \boldsymbol{A}^T_{n_+,t}\boldsymbol{P}_0\end{bmatrix}
\end{aligned}
$$

and get

$$p(\boldsymbol{s}|\boldsymbol{v}) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{s}} + \boldsymbol{\Sigma}_{\boldsymbol{sv}}\boldsymbol{\Sigma}_{\boldsymbol{vv}}^{-1}(\boldsymbol{v} - \boldsymbol{\mu}_{\boldsymbol{v}}), \boldsymbol{\Sigma}_{\boldsymbol{ss}} - \boldsymbol{\Sigma}_{\boldsymbol{sv}}\boldsymbol{\Sigma}_{\boldsymbol{vv}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{vs}}\right)$$

Both implementations reveal the overall $\mathcal{O}(d^3)$ scaling of the conditional statistics.

A.2 Sampling from the variational posterior process

We here describe a method to jointly sample from the posterior $q(\boldsymbol{s}(\cdot))$ at inputs $\boldsymbol{x}$. Such a sample can be obtained by first sampling from the prior process at inputs $[\boldsymbol{x}, \boldsymbol{z}]$:

$$\boldsymbol{s}_p, \boldsymbol{u}_p \sim p(\boldsymbol{s}(\boldsymbol{x}), \boldsymbol{s}(\boldsymbol{z})) \tag{21}$$

Then sampling from the marginal posterior at $\boldsymbol{z}$:

$$\boldsymbol{u}_q \sim q(\boldsymbol{s}(\boldsymbol{z})) \tag{22}$$

And finally construct:

$$\boldsymbol{s} = \boldsymbol{s}_p + E[\boldsymbol{s}(\boldsymbol{x})|\boldsymbol{s}(\boldsymbol{z}) = \boldsymbol{u}_q - \boldsymbol{u}_p], \tag{23}$$

which is a sample from the marginal posterior $q(\boldsymbol{s}(\boldsymbol{x}))$.

This methods allows to generate samples in complexity $\mathcal{O}((N + M)d^3)$, which is the time required to jointly sample s_p, u_p. It was used to produce the posterior samples in the deep- GP experiment.

B Multivariate Gaussian distributions with banded precision: parameterisations, link functions and natural gradients

Here we present different parameterisations of a multivariate Gaussian distribution with block- tridiagonal precision matrices along with the link functions between them and describe how we use these to compute natural gradient updates of our variational objective.

B.1 Distribution parameterisations and link functions

In Section 3.2 we have defined the variational distribution approximating the posterior on inducing states to be

$$q_{\boldsymbol{u}} = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{u}}, \boldsymbol{Q}_{\boldsymbol{u}}^{-1}), \quad \boldsymbol{Q} = \boldsymbol{L}_{\boldsymbol{u}}\boldsymbol{L}_{\boldsymbol{u}}^{\top}, \tag{24}$$

where $\boldsymbol{Q}_{\boldsymbol{u}}$ denotes the precision matrix with block-tridiagonal structure and $\boldsymbol{L}_{\boldsymbol{u}}$ the Cholesky factor of the precision. We denote with $\boldsymbol{\xi} : \{\boldsymbol{m}_{\boldsymbol{u}}, \boldsymbol{L}_{\boldsymbol{u}}\}$ the above parameterisation. In the following table we present the identities that allow us to transfer back and forth from the default parameterisation $\boldsymbol{\xi}$ to the natural parameters $\boldsymbol{\theta} : \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ and to the expectation parameters $\boldsymbol{\eta} : \{\boldsymbol{\eta}_1, \boldsymbol{\eta}_2\}$.

Table 3: Transformations between the different parameterisations of the Gaussian distribution with block-tridiagonal precision.

Transformation	Original parameterisation		Resulting parameterisaton	
$\boldsymbol{\xi} \to \boldsymbol{\theta}$	$\boldsymbol{\mu}_{\boldsymbol{u}},$	$\text{btd}[\boldsymbol{L}_{\boldsymbol{u}}]$	$\boldsymbol{\theta}_1 = \boldsymbol{L}_{\boldsymbol{u}}\boldsymbol{L}_{\boldsymbol{u}}^{\top}\boldsymbol{\mu}_{\boldsymbol{u}},$	$\boldsymbol{\theta}_2 = -1/2\,\text{btd}[\boldsymbol{L}_{\boldsymbol{u}}\boldsymbol{L}_{\boldsymbol{u}}^{\top}]$
$\boldsymbol{\theta} \to \boldsymbol{\xi}$	$\boldsymbol{\theta}_1,$	$\text{btd}[\boldsymbol{\theta}_2]$	$\boldsymbol{\mu}_{\boldsymbol{u}} = (-2\boldsymbol{\theta}_2)^{-1}\boldsymbol{\theta}_1,$	$\boldsymbol{L}_{\boldsymbol{u}} = \text{btd}[\text{chol}[-2\boldsymbol{\theta}_2]]$
$\boldsymbol{\xi} \to \boldsymbol{\eta}$	$\boldsymbol{\mu}_{\boldsymbol{u}},$	$\text{btd}[\boldsymbol{L}_{\boldsymbol{u}}]$	$\boldsymbol{\eta}_1 = \boldsymbol{\mu}_{\boldsymbol{u}},$	$\boldsymbol{\eta}_2 = \text{btd}[\boldsymbol{L}_{\boldsymbol{u}}^{-\top}\boldsymbol{L}_{\boldsymbol{u}}^{-1} + \boldsymbol{\mu}_{\boldsymbol{u}}\boldsymbol{\mu}_{\boldsymbol{u}}^{\top}]$
$\boldsymbol{\eta} \to \boldsymbol{\xi}$	$\boldsymbol{\eta}_1,$	$\text{btd}[\boldsymbol{\eta}_2]$	$\boldsymbol{\mu}_{\boldsymbol{u}} = \boldsymbol{\eta}_1,$	$\boldsymbol{L}_{\boldsymbol{u}} = \text{btd}[\text{chol}[(\boldsymbol{\eta}_2 - \boldsymbol{\eta}_1\boldsymbol{\eta}_1^{\top})^{-1}]]$

Note that the expectation parameter $\boldsymbol{\eta}_2$ is a full matrix since it involves the inverse of a banded matrix, which is not necessarily banded. However, since the sufficient statistics of the distribution are $\boldsymbol{t}(\boldsymbol{u}) = [\boldsymbol{u}, \text{btd}[\boldsymbol{u}\boldsymbol{u}^{\top}]]$, we only need to compute the elements in the band.

B.2 Natural gradient update

When maximising our objective $\mathcal{L}(\boldsymbol{\xi})$ with respect to $\boldsymbol{\xi}$, the parameters of our variational distribution, we perform a sequence of natural gradient updates:

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \gamma_t \tilde{\nabla}_{\boldsymbol{\xi}} \mathcal{L}|_{\boldsymbol{\xi}=\boldsymbol{\xi}_t}, \quad \tilde{\nabla}_{\boldsymbol{\xi}} \mathcal{L}|_{\boldsymbol{\xi}=\boldsymbol{\xi}_t} = \boldsymbol{F}_{\boldsymbol{\xi}}^{-1} \nabla_{\boldsymbol{\xi}^\top} \mathcal{L}|_{\boldsymbol{\xi}=\boldsymbol{\xi}_t}$$

In an exponential family with natural parameters $\boldsymbol{\theta}$ and expectation parameters $\boldsymbol{\eta}$, the Fisher information is given by

$$\boldsymbol{F}_{\boldsymbol{\xi}} = \left(\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}\boldsymbol{\xi}}\right)^\top \frac{\mathrm{d}\boldsymbol{\eta}}{\mathrm{d}\boldsymbol{\theta}} \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}\boldsymbol{\xi}}$$

As shown in Salimbeni et al. (2018), this leads to

$$\tilde{\nabla}_{\boldsymbol{\xi}^\top} \mathcal{L} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\theta}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}^\top},$$

which is a Jacobian-vector product allowing for an efficient implementation using automatic differentiation libraries. The computation of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}}$ is achieved using the chain rule: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}}$.

B.3 Inverse of the subset inverse, and its reverse mode derivatives

A banded positive semi-definite (PSD) matrix Q has a Cholesky factor L_Q that is lower triangular with the same lower bandwidth. However, its inverse Q^{-1} is in most cases dense. If one is only interested in computing the entries of Q^{-1} that are located in the band of Q (denoted by $\mathrm{band}_Q[\cdot]$), efficient subset inverse algorithms are available (Durrande et al., 2019). We are now interested in the mathematical inverse of this operation, which we call reverse to avoid confusion with the matrix inverse operation.

$$L_Q \underset{\text{subset inverse}}{\overset{\text{reverse subset inverse}}{\longleftarrow\!\!\!\!\!\!\longrightarrow}} \mathrm{band}_Q[Q^{-1}]$$

We consider symmetric matrices with lower bandwidth r

$$Q = \begin{pmatrix} q_{11} & \dots & q_{1r} & & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ q_{r1} & \ddots & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & \ddots & q_{n-r,n} \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & & q_{n,n-r} & \dots & q_{n,n} \end{pmatrix}.$$

Such a matrix has banded Cholesky factor

$$L_Q = \begin{pmatrix} l_{11} & & & & \dots & 0 \\ \vdots & \ddots & & & & \vdots \\ l_{r1} & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \dots & & l_{n,n-r} & \dots & l_{n,n} \end{pmatrix}.$$

Computing the subset inverse is $\mathcal{O}(nr^2)$ (**?**). We present below an algorithm that performs the reverse subset inverse operation with complexity $\mathcal{O}(nr^2)$.

B.3.1 Derivation of the forward evaluation $\mathrm{band}_Q[C] \to L_Q$

The following derivation shows how to compute, for each index i, the column $L_{i:i+r,i}$ given the sub-block $C_{i:i+r,i:i+r}$ independently.

To simplify notations, we introduce the following intervals $n = [1 : i-1]$, $o = [i : i+r]$ and $p = [i+r+1 : n]$. We denote by C the full covariance $C = Q^{-1} = L^{-T}L^{-1}$.

First, we have that the sub-covariance $C_{op,op}$ only depends on $L_{op,op}$

$$L^{-1} = \begin{bmatrix} L_{n,n} & 0 \\ L_{op,n} & L_{op,op} \end{bmatrix}^{-1} = \begin{bmatrix} L_{n,n}^{-1} & 0 \\ -L_{n,n}^{-1}L_{n,op}L_{op,op}^{-1} & L_{op,op}^{-1} \end{bmatrix} \implies C_{op,op} = L_{op,op}^{-T}L_{op,op}^{-1} \tag{25}$$

Second, we have that the following expression for sub-covariance $C_{o,o}$

$$[L_{op,op}]^{-1} = \begin{bmatrix} L_{o,o} & 0 \\ L_{p,o} & L_{p,p} \end{bmatrix}^{-1} = \begin{bmatrix} L_{o,o}^{-1} & 0 \\ -L_{p,p}^{-1}L_{p,o}L_{o,o}^{-1} & L_{p,p}^{-1} \end{bmatrix} \tag{26}$$

$$\implies C_{o,o} = [L_{op,op}^{-T}L_{op,op}^{-1}]_{1:r,1:r} = L_{o,o}^{-T}L_{o,o}^{-1} + L_{o,o}^{-T}L_{p,o}^{T}L_{p,p}^{-T}L_{p,p}^{-1}L_{p,o}L_{o,o}^{-1} \tag{27}$$

Using, the matrix inversion lemma, we get the following expression for $C_{o,o}^{-1}$

$$C_{o,o}^{-1} = [L_{op,op}^{-T}L_{op,op}^{-1}]_{1:r,1:r}^{-1} \tag{28}$$

$$= L_{o,o}L_{o,o}^{T} - L_{o,o}L_{p,o}^{T}(L_{p,p}L_{p,p}^{T} + L_{p,o}L_{p,o}^{T})^{-1}L_{p,o}L_{o,o}^{T} \tag{29}$$

By construction, the first column of L_{po} is out of the matrix band (it is a null vector). Because $L_{o,o}$ is lower triangular, the product $L_{po}L_{o,o}^{T}$ also has a null vector as its first column, and so has the last term in Eq. 29.

Therefore, keeping only the first column, we end up with the identity

$$[C_{o,o}^{-1}]_{:,1} = L_{o,o}[L_{o,o}^{T}]_{:,1} = L_{o,i}L_{i,i}, \tag{30}$$

which is a system of r equations with r unknown $L_{o,i}$, that we can solve analytically getting first $L_{i,i} = \sqrt{[C_{o,o}^{-1}]_{1,1}}$ then $L_{o,i} = [C_{o,o}^{-1}]_{:,1}/L_{i,i}$

This derivation is summarised in Algorithm 1:

Algorithm 1 Reverse subset inverse for banded matrices

procedure REV_SUBSET_INV(C) ▷ $C = \text{band}_Q[Q] \in \mathbb{R}^{n\times n}$, $Q = LL^T$ of bandwidth r
  $L \leftarrow 0$
  for $i \in [0, \dots, n-1]$ **do**
    $c^{(i)} \leftarrow C_{i:i+r,i:i+r}$ ▷ extract symmetric sub-block $(r \times r)$ at i
    $v^{(i)} \leftarrow (c^{(i)})^{-1}e, \quad e = [1, 0, \dots, 0] \in \mathbb{R}^r$ ▷ select first column of $(c^{(i)})^{-1}$
    $l^{(i)} \leftarrow v^{(i)}/\sqrt{v_1^{(i)}}$
    $L_{i:i+r,i} \leftarrow l^{(i)}$
  return L ▷ Cholesky factor of Q

B.3.2 Derivation of the reverse mode differentiation

We manually derive the reverse mode differentiation of the reverse inverse subset algorithm introduced in the previous section.

We refer the reader to **?** for a brief introduction to reverse mode differentiation and to Durrande et al. (2019) for derivations of reverse mode differentiation of the subset inverse algorithm for banded matrices.

In a nutshell, in a chain $A \to B \to ... \to c \in \mathbb{R}$ with A, B matrices, the reverse mode derivative of operation of $f : A \to B$ is the operation propagating the reverse mode sensitivity $\bar{B} = \frac{dc}{dB}$ into sensitivity $\frac{dc}{dA}$. Given the differential identity $dc = \sum_{ij} \frac{\partial c}{\partial B_{ij}} dB_{ij} = Tr[\bar{B}^T dB]$ and the general differential relation at A: $dB = X_A dA Y_A$, it follows that $dc = Tr[Y_A^T X_A dA]$, therefore we identify $\bar{A}^T = Y_A \bar{B}^T X_A$.

Algorithm 1 being parallel, we can compute the contribution of each column of L to C separately. First we relate $l^{(i)}$ to $v^{(i)}$:

$$dl^{(i)} = \frac{1}{\sqrt{v_1^{(i)}}} \underbrace{\begin{bmatrix} 1 - l_1^{(i)}/\left(2\sqrt{v_1^{(i)}}\right) & & \\ -l_2^{(i)}/\left(2\sqrt{v_1^{(i)}}\right) & 1 & \\ \vdots & & \ddots \\ -l_n^{(i)}/\left(2\sqrt{v_1^{(i)}}\right) & & 1 \end{bmatrix}}_{H_i} dv^{(i)} = H_i dv^{(i)} \,.$$

We identify $\bar{v}^{(i)}$:

$$df = \sum_i tr(i\bar{l}^{(i)T} dl^{(i)}) = \sum_i tr(\bar{l}^{(i)T} H_i dv^{(i)}) \implies \bar{v}^{(i)T} = \bar{l}^{(i)T} H_i$$

Then we relate $v^{(i)}$ to $c^{(i)}$,

$$v^{(i)} = \left(c^{(i)}\right)^{-1} e, \implies dv^{(i)} = -\left(c^{(i)}\right)^{-1} dc^{(i)} \left(c^{(i)}\right)^{-1} e$$

And we identify $\bar{c}^{(i)}$:

$$\begin{aligned} df &= \sum_i tr(\bar{v}^{(i)T} dv^{(i)}) = \sum_i tr(-\left(c^{(i)}\right)^{-1} e \bar{v}^{(i)T} \left(c^{(i)}\right)^{-1} dc^{(i)}) \\ &\implies \bar{c}^{(i)T} = -\left(c^{(i)}\right)^{-1} e^{(i)} \bar{v}^{(i)T} \left(c^{(i)}\right)^{-1} \end{aligned}$$

Putting everything together, we have

$$\bar{c}^{(i)T} = -\left(c^{(i)}\right)^{-1} e \bar{l}^{(i)T} H_i \left(c^{(i)}\right)^{-1}$$

Algorithm 2 summarises the derivations of the reverse mode sensitivity $\bar{C}$ of the reverse subset inverse operation.

Algorithm 2 Reverse mode sensitivity: reverse of subset inverse

procedure GRAD_REV_SUBSET_INV($\bar{L}, C$) ▷ $\bar{L} = \frac{df}{dL}$ is $n \times n$
  $\bar{C} \leftarrow 0$
  for $i \in [0, \ldots, n-1]$ **do**
    $c^{(i)} \leftarrow C_{i:i+r,i:i+r}$ ▷ extract symmetric sub-block $(r \times r)$ at i
    $\bar{l}^{(i)} = \bar{L}_{i:i+r,i}$
    $\bar{c}^{(i)} = -\left(c^{(i)}\right)^{-1} e \bar{l}^{(i)T} H_i \left(c^{(i)}\right)^{-1}$
    $\bar{C}_{i:i+r,i:i+r} \leftarrow \bar{C}_{i:i+r,i:i+r} + \bar{c}^{(i)}$ ▷ add to sensitivity $\bar{C}$
  return $\bar{C}$ ▷ $\bar{C} = \frac{df}{dC}$

B.4 Fast implementation

In Algorithms 1 and 2, we computed the $(c^{(i)})^{-1}e$ independently for each i. This can be achieved by first computing the Cholesky factors $s^{(i)}$ of each $c^{(i)}$ and then solving $v^{(i)} = (s^{(i)})^{-T}(s^{(i)})^{-1}e$. The direct Cholesky factorization of all $c^{(i)}$ would incur a total complexity of $\mathcal{O}(nr^3)$. However one can use a recursive algorithm achieving the same goal in complexity $\mathcal{O}(nr^2)$.

Given the Cholesky factor $s^{(i)}$ of $c^{(i)}$, we can compute the Cholesky factor of $c^{(i-1)}$ as follows:

$$s^{(i-1)} = \text{chol}(C_{i-1:i+r,i-1:i+r})_{1:r,1:r} \tag{31}$$

$$= \text{chol}\left(\begin{bmatrix} C_{i-1,i-1} & C_{i-1:i+r,i-1}^T \\ C_{i-1:i+r,i-1} & s^{(i)} s^{(i)T} \end{bmatrix}\right)_{1:r,1:r} \tag{32}$$

with

$$\text{chol}\left(\begin{bmatrix} C_{i-1,i-1} & C_{i-1:i+r,i-1}^T \\ C_{i-1:i+r,i-1} & s^{(i)}s^{(i)T} \end{bmatrix}\right) \tag{33}$$

$$= \begin{bmatrix} \sqrt{C_{i-1,i-1}} & 0 \\ C_{i-1:i+r,i-1}/\sqrt{C_{i-1,i-1}} & \text{chol}(s^{(i)}s^{(i)T} - C_{i-1:i+r,i-1}C_{i-1:i+r,i-1}^T/C_{i-1,i-1}) \end{bmatrix} \tag{34}$$

The Cholesky factor $s^{(i-1)}$ is then readily obtained by removing the last row and column of the matrix in Eq. 34. The bottom right entry of the matrix in Eq. 34 corresponds to a Cholesky downdate that can be computed with cost r^2 (**?**), so that, starting from index n, all Cholesky factors $s^{(1)}, \ldots, s^{(n)}$ can be computed with complexity $\mathcal{O}(nr^2)$.

C Experimental details

C.1 Illustration of the efficiency of the natural gradient update

We describe in this section a simple experiment to compare the behaviour of various optimisers. Given a regular grid X of 10^3 points equally spaced on $[0, 1]$ and a centred GP f with a Matérn 3/2 covariance (unit variance and length-scale $\ell = 0.1$), we generate two datasets at random as follow:

$$\boldsymbol{y} = f(X) + \varepsilon \qquad \text{with } \varepsilon_i \sim \mathcal{N}(0, 0.01) \tag{35}$$

$$\boldsymbol{z} = f(X) + \tau \qquad \text{with } \tau_i \sim \mathcal{T}(df = 1)\,. \tag{36}$$

Note that the first model has a conjugate likelihood whereas the second one does not. We can then fit S^{2}VGP models with 50 inducing points (fixed to a regular grid on $[0, 1]$). We show in Figure 7 the optimisations traces we obtained when optimising the variational parameters (all other model parameters being fixed to their nominal values), for different samples of the datasets. It can be seen that natural gradients provide a striking advantage in the conjugate case but that it also behaves favourably, especially in the first few iterations, in the non-conjugate case.

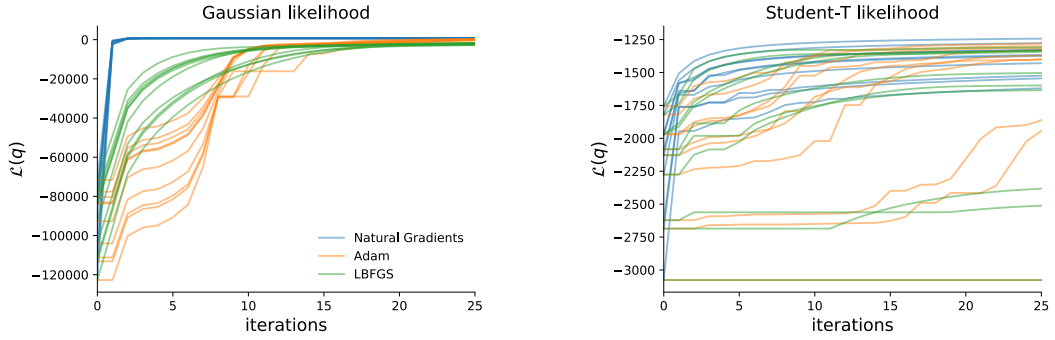

Figure 7: Optimisation traces for the ELBO of our S^{2}VGP method. Three optimisers (Natural gradients, Adam and LBFGS) are compared on 10 datasets that are generated at random. The left pannel correspond to a dataset and a model with a conjugate likelihood, whereas the right one is for a non conjugate likelihood.

C.2 Details on Section 4.2: conjugate regression on time-series

The full sound waveform used in this experiment is shown in Figure 8(top).

To model this signal, we used the following stationary kernels that have an equivalent SDE representations of state-dimension $2J$, where J is the number of harmonic components:

$$k^J(\tau) = k_{Mat^{1}/_{2}}(\tau)\left(\sum_{j=1}^{J} \gamma_j^2 \cos(2\pi f_0 j \tau)\right)$$

where f_0 denotes the fundamental frequency of the pitched sound. The variance of the Matérn$^1/_2$ kernel was set to one and its only free parameter is its length-scale ℓ. We used a Gaussian likelihood with variance σ^2. Since

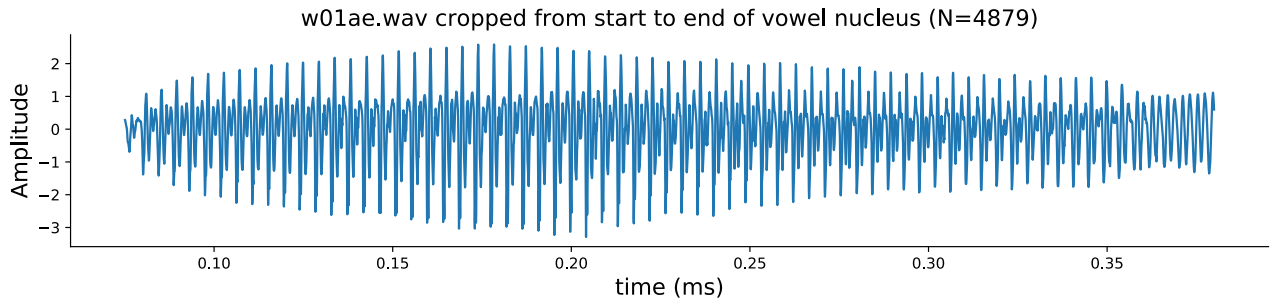

Figure 8: Audio time-series used in Section 4.2

we focused on inference, all parameters were initially fitted to the data by maximising the marginal likelihood available in closed form in this conjugate setting.

We increased the number of inducing points in powers of 2, placing them on an homogenous grid from the start to the end of the time support. Inducing points locations were not learned.

For both SVGP and S^{2}VGP, we only learned the variational parameters. We used *LBFGS* for both SVGP and S^{2}VGP.

C.3 Details on Section 4.3: additive regression

The generative model is as follow:

$$f_i \sim \mathcal{GP}(0, k_i), \quad k_i = \text{Matern}_{3/2}(\sigma_i^2, \ell_i), \tag{37}$$

$$y_k = \sum_{i=1}^{c} f_i(x_k^{(i)}) + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma^2), \quad x^{(i)} \in \mathbb{R}. \tag{38}$$

We propose a mean-field approximation to the posterior over processes $g(f_1, \ldots, f_c) = \prod_i q^{(i)}(f_i)$ as in (**?**). where each process is approximated using our doubly sparse parameterisation with inducing states $\boldsymbol{u}^{(i)} = f_i(\boldsymbol{z}^{(i)})$ evaluated at component specific inputs $\boldsymbol{z}^{(i)}$.

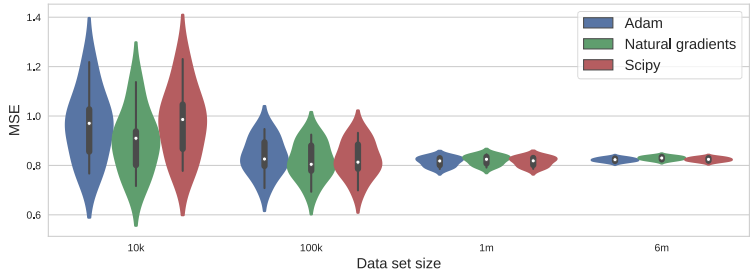

Figure 9: Comparison of predictive MSE on the airline delays dataset when training S^{2}VGP with various optimisers (the distribution of errors is across the 10 splits).

C.4 Details on Section 4.4: time warping with deep Gaussian process

Setting

The function f is a sine wave of the form $f(x) = (0.75(1 - \tanh(10 * 2\pi x/15)) + 0.25)\sin(10 * 2\pi x)$ and is shared across the two observed series. The functions a_k are time-warping functions, with $a_1(x) = x$ and $a_2(x) = x^2$. The functions g_k are output distortions with $g_1(x) = \tanh(x)$ and $g_2(x) = x$. To generate the two time series we uniformly sample 1000 points in $[0, 1]$ and subsequently pass them to the 2-layer model that we described. The Gaussian additive noise has standard deviation $\sigma = 0.05$. We removed observations in the intervals $[0.55, 0.6]$ & $[0.85, 0.9]$ for the first time series and in the interval of $[0.40, 0.50]$ for the second time series.

Uncertainty decomposition across layers in the deep GP experiment

As can be seen in Figure 6, there is almost no uncertainty in the first layer. The reason for this is two fold. First, we initialised the kernels on the first layer to have very long lengthscales (initial value of 4), and small variance (initial value of 0.01) to bias the inference towards smooth functions. Second, the low posterior uncertainty in the intermediate layers is a known consequence of variational approach used in Salimbeni and Deisenroth (2017), where the variational distribution factorises across layers (we use the same approximation). This pathology has been recently explored in **?** and can be remedied by explicitly imposing a conditional dependency between the layers in the variational distribution. We conducted the same experiment with a higher observation noise level ($\sigma = 0.1$ instead of $\sigma = 0.05$) and report the result in Figure 10. Changing the noise level has no effect to the uncertainty in the intermediate layers but has a significant effect in the model's ability to learn the correct functions, as the model prefers to explain these attributes by measurement noise.

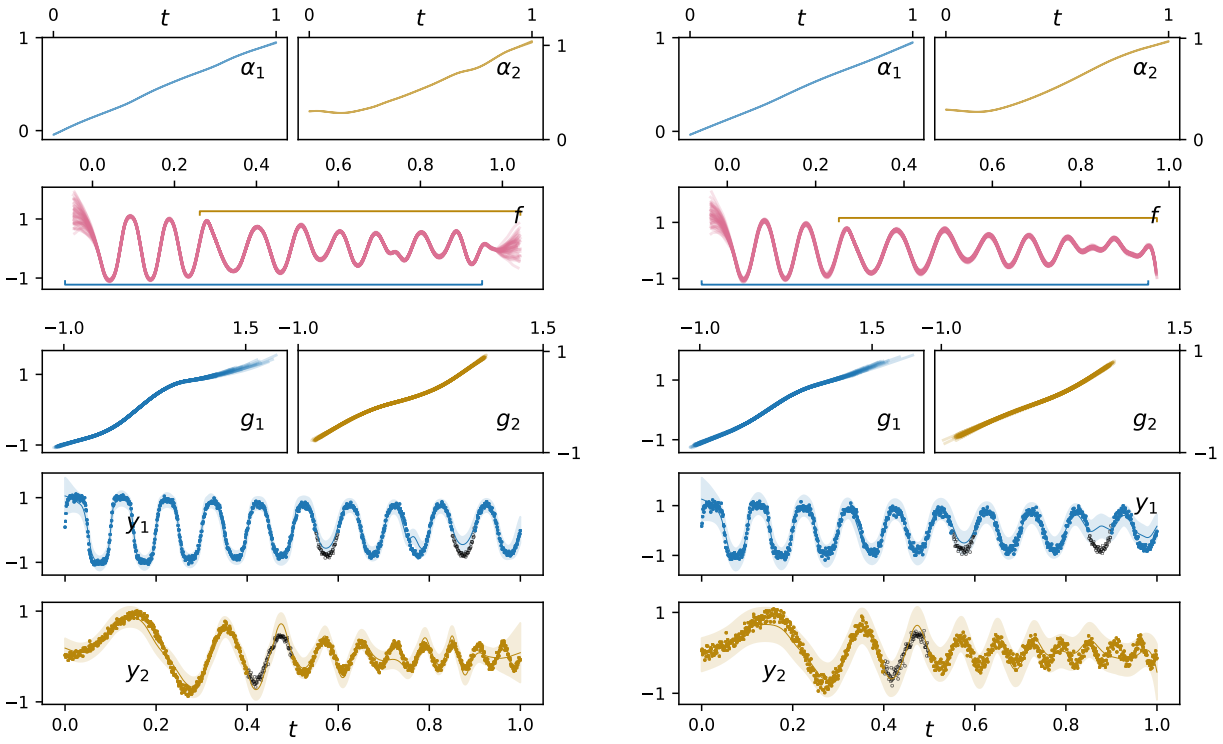

Figure 10: Data alignment with S^{2}VGP layers. **Left**: original experiment from Section 4.4 with $\sigma = 0.05$. **Right**: same experiment with $\sigma = 0.1$.

D Empirical comparison to alternative SSM based approximate inference methods

We empirically compare our S^{2}VGP approach to alternative methods to perform approximate inference in GP models based on their state space representation.

We consider a simple classification task similar from the GPMLv4.2 toolbox demo `gpml-matlab-master/doc/demoState.m` with N=5000 (see Figure 11) and using Matern$^3/_2$ kernel with fixed hyperparameters. For the S^{2}VGP method, we choose $M = 50$ inducing points on a homogenous grid and we use gradient based optimisation (L-BFGS). We run inference and report the NLPDs and execution time:

Algorithm	NLPD	Time (s)
S^2VGP [M=50] (ours)	0.586 ± 0.013	3.38 ± 0.98
Laplace (gpml)	0.586 ± 0.013	7.55 ± 0.15
ADF/EP (gpml)	0.586 ± 0.013	10.585 ± 0.014
VB (gpml)	0.587 ± 0.013	39.530 ± 0.064

For S^2VGP, we run gradient based optimisation (using L-BFGS). We are equally accurate as the other methods yet much faster.

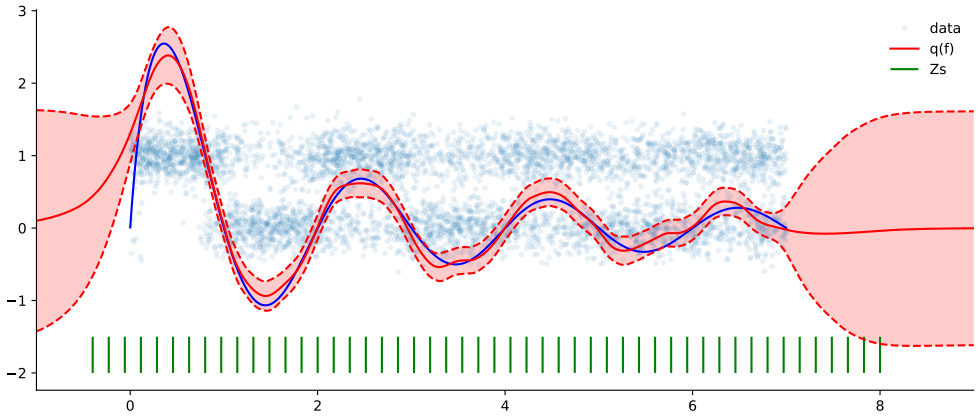

Figure 11: Classification data (N=5000) and S^2VGP fit. The ground truth used to generate the data is shown in blue. Blue dots represent the binary data (with additional noise introduced for visibility). The posterior process is shown in red. Inducing point locations are shown in green.