Supplementary Materials: Hierarchical Inducing Point Gaussian Processes for Inter-domain Observations

1 The HIP-GP Algorithm

We describe two algorithms that are core to the acceleration techniques we develop in Section 3.2. Algorithm 2 computes a fast MVM with a hierarchical Toeplitz matrix using the circulant embedding described in Algorithm 1. Note that we can similarly compute the MVM $\boldsymbol{R}^\top \boldsymbol{v}$ simply by adapting Algorithm 2 to perform FFT on $\boldsymbol{C}^{1/2}$ instead of on $\boldsymbol{C}$. Together these algorithms are sufficient for use within PCG to efficiently compute $\boldsymbol{k}_n$.

Algorithm 1: Hierarchical circulant embedding.

Data: T ($N_1 \times ... \times N_D$ representation of hierarchical Toeplitz matrix);
Result: C (circulant embedding of T)
$C \leftarrow T$; `// copy`
for $d \leftarrow 1$ **to** D **do**
  $C_r \leftarrow$ `reverse-dim`$(C, \text{dim} = d)$
  $C_r \leftarrow$ `chop-single-dim`$(C_r, \text{dim} = d)$
  $C_r \leftarrow$ `binary-zero-pad`(C_r) ; `// front pad`
  $C \leftarrow$ `concat`$(C, C_r, \text{dim} = d)$
return C

Algorithm 2: Matrix-vector multiplication $\boldsymbol{K}\boldsymbol{v}$ for a symmetric hierarchical Toeplitz matrix $\boldsymbol{K}$ and vector $\boldsymbol{v}$.

Data: $\boldsymbol{k}_0$ (first row of $\boldsymbol{K}$ in C-order); $\boldsymbol{v}$ (vector, also in C-order); $N_1, \ldots, N_D$ (grid dimensions)
Result: $\boldsymbol{K}\boldsymbol{v}$ (matrix-vector product)
$T \leftarrow$ `reshape`$(\boldsymbol{k}_0, N_{1:D})$; `// to` $N_1 \times \cdots \times N_D$
$V \leftarrow$ `reshape`$(\boldsymbol{v}, N_{1:D})$; `// to` $N_1 \times \cdots \times N_D$
$C \leftarrow$ `Circ-Embed`$(T, N_{1:D})$; `//`
$V \leftarrow$ `Zero-Embed`$(V, N_{1:D})$; `// match` C
$res \leftarrow$ `ifft`$($`fft`$(C) \cdot$ `fft`$(V))$; `//` D`-dim fft`
return `flatten`(res) ; `// flatten in C-order`

2 Optimization Details

In this section, we derive the gradients for structured variational parameters and kernel hyperparameters.

2.1 Variational parameters

The structured variational posterior is characterized by $N(\boldsymbol{m}, \boldsymbol{S}) = \prod_{i=1}^{B} \mathcal{N}(\boldsymbol{m}_i, \boldsymbol{S}_i)$, where we decompose the $M \times M$ matrix $\boldsymbol{S}$ into B block-independent covariance matrices of block size M_b:

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_1 & & & \\ & \boldsymbol{S}_2 & & \\ & & \cdots & \\ & & & \boldsymbol{S}_B \end{pmatrix}, \tag{1}$$

and the vector $\boldsymbol{m}$ into corresponding B blocks: $\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_B$.

2.1.1 Direct solves

We first consider directly solving the optimal $\boldsymbol{m}$ and $\boldsymbol{S}$.

Taking the derivatives of the HIP-GP objective w.r.t. $\boldsymbol{m}$ and $\boldsymbol{S}$, we obtain

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{S}_i} = -\frac{1}{2}\underbrace{\left((\sum_n \frac{1}{\sigma_n^2}\boldsymbol{k}_{n,i}\boldsymbol{k}_{n,i}^T) + \boldsymbol{I}_{M_b}\right)}_{\triangleq \boldsymbol{\Lambda}_i} + \frac{1}{2}\boldsymbol{S}_i^{-1}, \qquad \text{for } i = 1:B \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{m}} = \underbrace{\sum_n \frac{1}{\sigma_n^2} y_n \boldsymbol{k}_n}_{\triangleq \boldsymbol{b}} - \underbrace{\left(\sum_n \frac{1}{\sigma_n^2}\boldsymbol{k}_n\boldsymbol{k}_n^T + \boldsymbol{I}_M\right)}_{\triangleq \boldsymbol{\Lambda}} \boldsymbol{m} \tag{3}$$

where a vector or a matrix with subscript i, denotes its i-th block.

The optimum can be solved in closed form by setting the gradients equal to zero, i. e.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{S}_i} = 0 \qquad \Rightarrow \qquad \boldsymbol{S}_i = \boldsymbol{\Lambda}_i^{-1}, \qquad \text{for } b = 1:B \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{m}} = 0 \qquad \Rightarrow \qquad \boldsymbol{m} = \boldsymbol{\Lambda}^{-1}\boldsymbol{b}. \tag{5}$$

If M is very large, this direct solve will be infeasible. But note that $\boldsymbol{\Lambda}_i, \boldsymbol{b}$ and $\boldsymbol{\Lambda}$ are all summations over some data terms, hence we can compute an unbiased gradient estimate using a small number of samples which is more efficient. We will use natural gradient descent (NGD) to perform optimization.

2.1.2 Natural gradient updates

To derive the NGD updates, we need the other two paramterizations of $N(\boldsymbol{m}, \boldsymbol{S})$, namely,

- the *canonical parameterization*: $\{\boldsymbol{\theta}_{1,i}\}_{i=1}^B, \{\boldsymbol{\theta}_{2,i}\}_{i=1}^B$ where $\boldsymbol{\theta}_{1,i} = \boldsymbol{S}_i^{-1}\boldsymbol{m}_i$ and $\boldsymbol{\theta}_{2,i} = -\frac{1}{2}\boldsymbol{S}_i^{-1}$, $i = 1:B$; and
- the *expectation parameterization*: $\{\boldsymbol{\eta}_{1,i}\}_{i=1}^B, \{\boldsymbol{\eta}_{2,i}\}_{i=1}^B$ where $\boldsymbol{\eta}_{1,i} = \boldsymbol{m}_i$ and $\boldsymbol{\eta}_{2,i} = \boldsymbol{m}_i\boldsymbol{m}_i^T + \boldsymbol{S}_i$, $i = 1:B$.

In Gaussian graphical models, the natural gradient for the *canonical parameterization* corresponds to the standard gradient for the *expectation parameterization*. That is,

$$\frac{\partial}{\partial \boldsymbol{\eta}}\mathcal{L} = \tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}, \tag{6}$$

where $\tilde{\nabla}_\theta$ denotes the natural gradient w.r.t. θ.

By the chain rule, we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_{1,i}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{m}_i}\frac{\partial \boldsymbol{m}_i}{\partial \boldsymbol{\eta}_{1,i}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}_i}\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{\eta}_{1,i}} \tag{7}$$

$$= \boldsymbol{b}_i - \boldsymbol{S}_i^{-1}\boldsymbol{m}_i - [(\boldsymbol{\Lambda}\boldsymbol{m})_i - \boldsymbol{\Lambda}_i\boldsymbol{m}_i], \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_{2,i}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{m}_i}\frac{\partial \boldsymbol{m}_i}{\partial \boldsymbol{\eta}_{2,i}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}_i}\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{\eta}_{2,i}} \tag{9}$$

$$= -\frac{1}{2}\boldsymbol{\Lambda}_i + \frac{1}{2}\boldsymbol{S}_i^{-1}. \tag{10}$$

Therefore, the natural gradient updates for $\boldsymbol{\theta}$ are as follows:

$$\boldsymbol{\theta}_{1,i} \leftarrow \boldsymbol{\theta}_{1,i} + l\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_{1,i}} = \boldsymbol{\theta}_{1,i} + l\left(\boldsymbol{b}_i - \boldsymbol{S}_i^{-1}\boldsymbol{m}_i - [(\boldsymbol{\Lambda}\boldsymbol{m})_i - \boldsymbol{\Lambda}_i\boldsymbol{m}_i]\right) \tag{11}$$

$$\boldsymbol{\theta}_{2,i} \leftarrow \boldsymbol{\theta}_{2,i} + l\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_{2,i}} = \boldsymbol{\theta}_{2,i} + l\left(-\frac{1}{2}\boldsymbol{\Lambda}_i + \frac{1}{2}\boldsymbol{S}_i^{-1}\right), \tag{12}$$

where l is a positive step size.

2.2 Kernel hyperparameters

We now consider learning the kernel hyperparameters $\boldsymbol{\theta}$ with gradient descent.

For the convenience of notation, we denote the gram matrix $\boldsymbol{K_{u,u}}$ as $\boldsymbol{K}$. HIP-GP computes $\boldsymbol{K}^{-1}\boldsymbol{v}$ by PCG. Directly auto-differentiating through PCG may be numerically instable. Therefore, we derive the analytical gradient for this part. Denote $\boldsymbol{c}$ as the first row of $\boldsymbol{K}$ — $\boldsymbol{c}$ fully characterizes the symmetric Toeplitz matrix $\boldsymbol{K}$. It suffices to manually compute the derivative w.r.t. $\boldsymbol{c}$, i.e. $\frac{\partial \boldsymbol{r}^T \boldsymbol{K}^{-1}\boldsymbol{v}}{\partial \boldsymbol{c}}$, since by the following term $\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\theta}}$ can be taken care of with auto-differentiation.

We note the following equality

$$\frac{\partial \boldsymbol{r}^T \boldsymbol{K}^{-1}\boldsymbol{v}}{\partial \boldsymbol{c}} = -(\boldsymbol{K}^{-1}\boldsymbol{r})^T \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{c}} \boldsymbol{K}^{-1}\boldsymbol{v}. \tag{13}$$

The computation of $\boldsymbol{b} = \boldsymbol{K}^{-1}v$ is done in the forward pass and therefore can be cached for the backward pass. Additional computations in the backward pass are (1) $\boldsymbol{a} = \boldsymbol{K}^{-1}\boldsymbol{r}$ and (2) $\frac{\partial \boldsymbol{a}^T \boldsymbol{K}\boldsymbol{b}}{\partial \boldsymbol{c}}$. (1) can be computed efficiently using the techniques developed in HIP-GP. Now we present the procedure to compute (2):

$$\frac{\partial \boldsymbol{a}^T \boldsymbol{K}\boldsymbol{b}}{\partial \boldsymbol{c}} = \sum_{ij} a_i b_j \frac{\partial K_{ij}}{\partial \boldsymbol{c}} \tag{14}$$

$$= \sum_{ij} a_i b_j \boldsymbol{e}_{|i-j|+1} \tag{15}$$

$$= \text{toeplitz-mm}(b_1\boldsymbol{e}_1, \boldsymbol{b}, \boldsymbol{a}) + \text{toeplitz-mm}(a_1\boldsymbol{e}_1, \boldsymbol{a}, \boldsymbol{b}) - (\boldsymbol{a}^T\boldsymbol{b})\boldsymbol{e}_1, \tag{16}$$

where $\boldsymbol{e}_i$ denotes the vector with a 1 in the i-th coordinate and 0's elsewhere, and toeplitz-mm$(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ denotes the Toeplitz MVM $\boldsymbol{T}\boldsymbol{z}$, with the Toeplitz matrix $\boldsymbol{T}$ characterized by its first column vector $\boldsymbol{x}$ and first row vector $\boldsymbol{y}$ — this Toeplitz MVM can be also efficiently computed via its circulant embedding.

3 Additional Experiment Results

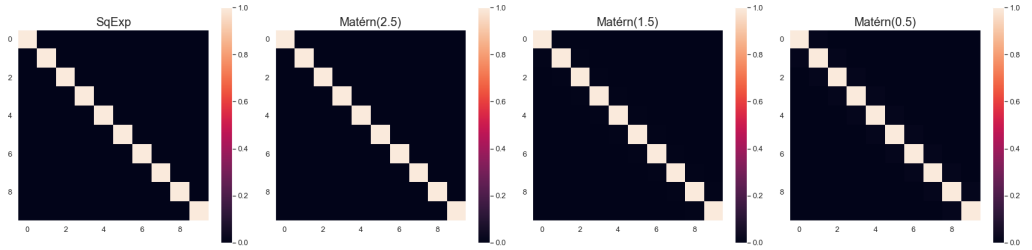
3.1 Empirical analysis on preconditioner

In this section, we present an empirical analysis on the preconditioner developed in Section 3.2. Specifically, we investigate our intuition on the "banded property" that makes the preconditioner effective: when the number of inducing points M are large enough, the inducing point Gram matrix $\boldsymbol{K_{u,u}}$ is increasingly sparse, and therefore the upper left block of $\boldsymbol{C}^{-1}$ will be close to $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}^{-1}$.

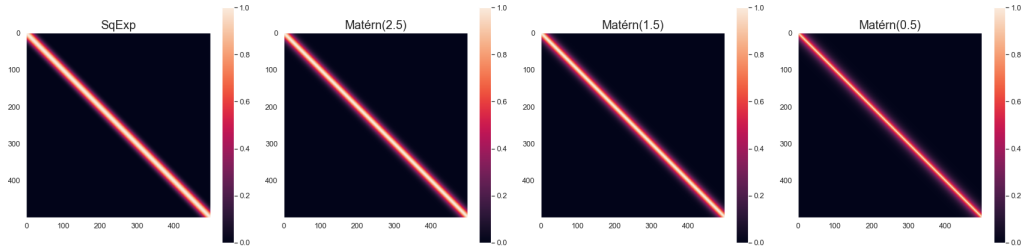
We note that the sparsity of the kernel matrix $\boldsymbol{K_{u,u}}$ depends on three factors (1) M, the number of inducing points, (2) the lengthscale l of the kernel, and (3) the property of the kernel function itself such as smoothness. Hence, to verify our intuition, we conduct the PCG convergence experiment by varying the combinations of these three factors. We evenly place M inducing points in the $[0, 2]$ interval that form the Gram matrix $\boldsymbol{K_{u,u}}$, and randomly generate 25 vectors $\boldsymbol{v}$ of length M. We vary M ranges from 10 to 500, and experiment with 4 types of kernel function: squared exponential kernel, Matérn (2.5), Matérn (1.5) and Matérn (0.5) kernels. For all kernels, we fix the signal variance σ^2 to 1 and the lengthscale l to 0.05 and 0.5 in two separate settings. We run CG and PCG to solve $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}^{-1}\boldsymbol{v}$ up to convergence with tolerance rate at 1e-10. We compare the fraction of # PCG iterations required for convergence over # CG iterations required for convergence, denoted as r_{pcg}. The results are displayed in Figure 1 (for $l = 0.05$) and Figure 2 (for $l = 0.5$).

Figure 1a - 1b and Figure 2a - 2b depict the kernel matrix $\boldsymbol{K_{u,u}}$ for $M = 10$ and $M = 500$ with $l = 0.05$ and 0.5, respectively. Figure 1c and 2c plot r_{pcg} over M for different kernels and lengthscales. From these plots, we make the following observations:

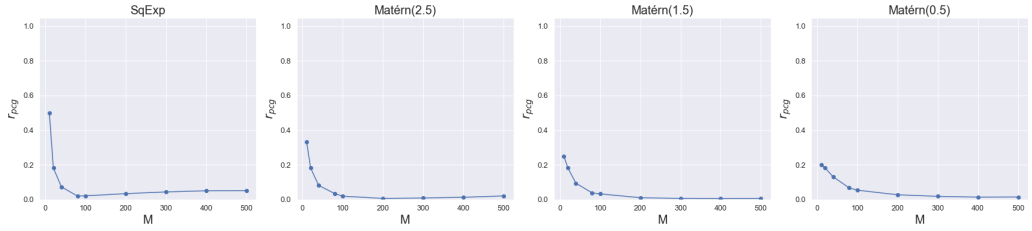
(1) r_{pcg} are consistently smaller than 1, which verifies the effectiveness of the preconditioner.



(a) Inducing point kernel matrix $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ with $M = 10$



(b) Inducing point kernel matrix $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ with $M = 500$



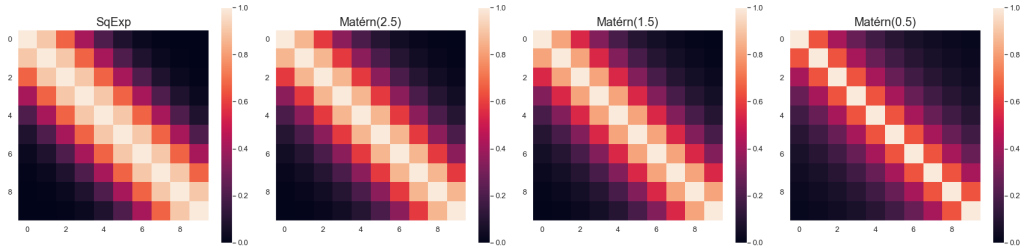
(c) r_{pcg} v.s. M for different kernels.

Figure 1: Empirical analysis for preconditioner. The kernel lengthscale is 0.05.

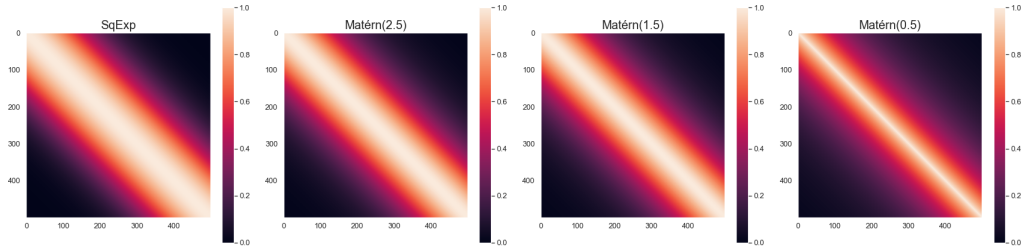
(2) In most cases, PCG converges faster when the system size M is bigger. For example, r_{pcg} decreases as M increases in Figure 1c where $l = 0.05$. However, we note that PCG convergence can be slowed down when the system size M exceeds certain threshold in some cases (e.g.first three plots of Figure 2c where $l = 0.5$). To see why this happens, we compare the plots of kernel matrices with $l = 0.5$ for $M = 10$ and $M = 500$ in Figure 2a and 2b. Since the lengthscale $l = 0.5$ is relatively large with respect to the input domain range, the resulting $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ for $M = 10$ is sparse enough to approach a diagonal matrix. However, when we increase M to 500, 1 lengthscale unit covers too many inducing points, making $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ less "banded" and the preconditioner less effective. This observation is also consistent with our intuition.

(3) For kernels that are less smooth, the PCG convergence speed-ups over CG are bigger given the same M, e.g. Matérn (0.5) kernel has smaller r_{pcg} than squared exponential kernel does under the same configuration. We also observe that $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ with Matérn (0.5) kernel is more diagonal-like than $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ with squared exponential kernel from the kernel matrix plots. Together with (2), these results show that when the kernel matrix is more banded, the PCG convergence is accelerated more.

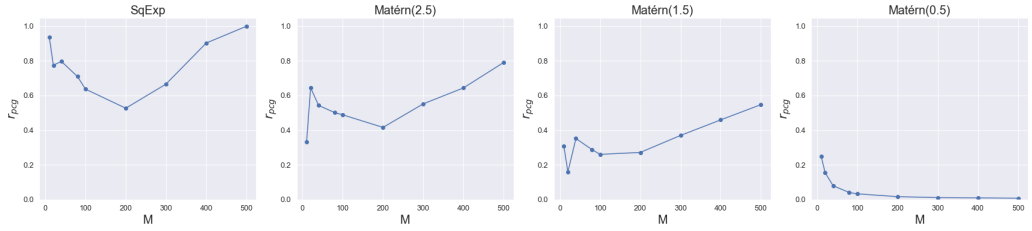
In conclusion, the PCG convergence speed depends on the "banded" property of the inducing point kernel matrix $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$, which further depends on M and the smoothness of the kernel. As $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ approaches a banded matrix, the preconditioner speeds up convergence drastically.



(a) Inducing point kernel matrix $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ with $M = 10$



(b) Inducing point kernel matrix $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}$ with $M = 500$



(c) r_{pcg} v.s. M for different kernels.

Figure 2: Empirical analysis for preconditioner. The kernel lengthscale is 0.5.

3.2 Additional experiment results for Section 5.2

We include additional experiment results on the other 3 kernels for Section 5.2, in Table 1- 3. These results are consistent to our conclusion in the paper.

M	10^3	10^4	10^5	10^6
HIP-GP	**0.0078**	**0.0187**	**0.3484**	**1.4727**
SVGP	0.0152	0.1516	n/a	n/a

Table 1: Whitening time comparison (second) of HIP-GP v.s. SVGP with Matérn(0.5) kernel.

M	10^3	10^4	10^5	10^6
HIP-GP	**0.0087**	**0.0192**	**0.3479**	**1.4656**
SVGP	0.0142	0.1379	n/a	n/a

Table 2: Whitening time comparison (second) of HIP-GP v.s. SVGP with Matérn(1.5) kernel.

M	10^3	10^4	10^5	10^6
HIP-GP	**0.0112**	**0.0199**	**0.3683**	**2.3433**
SVGP	0.7090	0.0992	n/a	n/a

Table 3: Whitening time comparison (second) of HIP-GP v.s. SVGP with squared exponential kernel.

3.3 UCI benchmark dataset

We include another experiment on the UCI 3D Road dataset ($N = 278,319, D = 3$). Following the same setup as Wang et al., 2019, we train HIP-GP with $M = 36{,}000$ and a mean-field variational family, and compare to their reported results of Exact GP, SGPR ($M = 512$) and SVGP ($M = 1,024$) (Table 4). With the large M, HIP-GP achieves the smallest NLL, and the second-smallest RMSE (only beaten by exact GPs).

RMSE				**NLL**			
HIP-GP	Exact GP	SGPR	SVGP	HIP-GP	Exact GP	SGPR	SVGP
0.189	**0.101**	0.661	0.481	**−0.171**	0.909	0.943	0.697

Table 4: UCI 3D Road experiment ($N = 278{,}319$). Results are averaged over 3 random splits.