Supplementary Appendices

A Reformulation of SKI as Bayesian Linear Regression – Omitted Details

A.1 Exact Inference for GSGP

For completeness we derive the exact GSGP inference equations for Fact 2.

Proof of Fact 2. GSGP is a standard linear basis function model and it is well known (see e.g., (Neal, 2011)) that the posterior mean and covariance of $\boldsymbol{\theta}$ given $\mathbf{y}$ are given by

$$\mathbb{E}[\boldsymbol{\theta}|\mathbf{y}] = \frac{1}{\sigma^2} \cdot \left[K_G^{-1} + \frac{1}{\sigma^2} W^T W\right]^{-1} \cdot W^T \mathbf{y}. \tag{4}$$

and

$$\mathrm{Var}(\boldsymbol{\theta}|\mathbf{y}) = \left[K_G^{-1} + \frac{1}{\sigma^2} W^T W\right]^{-1}. \tag{5}$$

Using that for invertible A, B, $(AB)^{-1} = B^{-1}A^{-1}$ we can write:

$$\frac{1}{\sigma^2} \cdot \left[K_G^{-1} + \frac{1}{\sigma^2} W^T W\right]^{-1} = \frac{1}{\sigma^2} \cdot \left[I + \frac{1}{\sigma^2} K_G W^T W\right]^{-1} K_G = (K_G W^T W + \sigma^2 I)^{-1} K_G.$$

Plugging back into the posterior mean equation (4) gives

$$\mathbb{E}[\boldsymbol{\theta}|\mathbf{y}] = (K_G W^T W + \sigma^2 I)^{-1} K_G W^T \mathbf{y} = \bar{\mathbf{z}}.$$

By linearity since $f(\mathbf{x}) = \mathbf{w}_{\mathbf{x}}^T \boldsymbol{\theta}$, this gives the GSGP posterior mean equation, $\mu_{f|\mathcal{D}}(\mathbf{x}) = \mathbf{w}_{\mathbf{x}}^T \bar{\mathbf{z}}$. Similarly, plugging back into the posterior variance equation (5) gives:

$$\mathrm{Var}(\boldsymbol{\theta}|\mathbf{y}) = \sigma^2 (K_G W^T W + \sigma^2 I)^{-1} K_G = \bar{\mathbf{C}}.$$

Again by linearity this gives the GSGP posterior variance equation $k_{f|\mathcal{D}}(\mathbf{x}, \mathbf{x}') = \mathbf{w}_{\mathbf{x}}^T \bar{\mathbf{C}} \mathbf{w}_{\mathbf{x}'}$.

Finally, it is well known (Neal, 2011) that the log likelihood of $\mathbf{y}$ under the GSGP linear basis function model is given by:

$$\log p(\mathbf{y}) = -\frac{1}{2}\left[n \log(2\pi) + n \log(\sigma^2) + \log\det K_G - \log\det(\bar{\mathbf{C}}) + \frac{1}{\sigma^2} \mathbf{y}^T \mathbf{y} - \bar{\mathbf{z}}^T \bar{\mathbf{C}}^{-1} \bar{\mathbf{z}}\right]$$

We can split $\log\det(\bar{\mathbf{C}}) = \log\det(\sigma^2 (K_G W^T W + \sigma^2 I)^{-1} K_G) = m \log \sigma^2 - \log\det(K_G W^T W + \sigma^2 I) + \log\det(K_G)$, and canceling this gives:

$$\log p(\mathbf{y}) = -\frac{1}{2}\left[n \log(2\pi) + (n-m)\log\sigma^2 + \log\det(K_G W^T W + \sigma^2 I) + \frac{1}{\sigma^2} \mathbf{y}^T \mathbf{y} - \bar{\mathbf{z}}^T \bar{\mathbf{C}}^{-1} \bar{\mathbf{z}}\right] \tag{6}$$

Additionally we can write:

$$\bar{\mathbf{z}}^T \bar{\mathbf{C}}^{-1} \bar{\mathbf{z}} = \frac{1}{\sigma^2} \bar{\mathbf{z}}^T K_G^{-1} (K_G W^T W + \sigma^2 I)(K_G W^T W + \sigma^2 I)^{-1} K_G W^T \mathbf{y} = \frac{1}{\sigma^2} \bar{\mathbf{z}}^T W^T \mathbf{y}.$$

Thus we have $\frac{1}{\sigma^2} \mathbf{y}^T \mathbf{y} - \bar{\mathbf{z}}^T \bar{\mathbf{C}}^{-1} \bar{\mathbf{z}} = \frac{\mathbf{y}^T(\mathbf{y} - W\bar{\mathbf{z}})}{\sigma^2}$. Plugging back into (6) yields the log likelihood formula of Fact 2, completing the proof. □

A.2 Equivalence of GSGP Exact Inference and SKI

Theorem 1, which states that the GSGP and SKI inference equations are identical, follows directly from Claim 1, as GSGP is a Gaussian process whose kernel is exactly the approximate kernel used by SKI. For illustrative purposes, we also give a purely linear algebraic proof of Theorem 1 below:

Theorem 1. *The inference expressions of Fact 2 are identical to the SKI approximations of Def. 1.*

Linear Algebraic Proof of Theorem 1. The equivalence between the expressions for the posterior mean and posterior variance is a standard manipulation to convert between the "weight space" and "function space" view of a GP with an explicit feature expansion, e.g., Eqs. (2.11), (2.12) of (Rasmussen, 2004). We provide details with our notation for completeness. The correspondence between the two expressions for the log-likelihood is due to the same correspondence between "weight space" and "function space" views, though we are not aware of a specific reference that provides the formula in Fact 2.

First, recall these definitions from Def. 1 and Fact 2:

$$\begin{aligned}\tilde{K}_X &= WK_GW^T\\ \tilde{\mathbf{z}} &= \left(WK_GW^T+\sigma^2 I\right)^{-1}\mathbf{y}\\ \bar{\mathbf{z}} &= (K_GW^TW+\sigma^2 I)^{-1}K_GW^T\mathbf{y}\end{aligned}$$

Mean: The two mean expressions are $\mathbf{w_x}K_GW^T\tilde{\mathbf{z}}$ (Def. 1) and $\mathbf{w_x}\bar{\mathbf{z}}$ (Fact 2). Expanding these, it suffices to show that

$$K_GW^T\left(WK_GW^T+\sigma^2 I\right)^{-1} = \left(K_GW^TW+\sigma^2 I\right)^{-1}K_GW^T. \tag{7}$$

This follows from the following claim:

Claim 5. *Let $A\in\mathbb{R}^{m\times n}$ and $B\in\mathbb{R}^{n\times m}$. Then*

$$A(BA+\sigma^2 I_n)^{-1} = (AB+\sigma^2 I_m)^{-1}A$$

as long as $BA+\sigma^2 I_n$ and $AB+\sigma^2 I_m$ are both invertible.

Proof. Observe that $(AB+\sigma^2 I_m)A = A(BA+\sigma^2 I_n) = ABA+\sigma^2 A$. If the matrices $AB+\sigma^2 I_m$ and $BA+\sigma^2 I_n$ are invertible the result follows by left- and right-multiplying by the inverses. □

We obtain (7) from Claim 5, applied with $A=K_GW^T$ and $B=W$. As required by the claim, we have that both $K_GW^TW+\sigma^2 I$ and $WK_GW^T+\sigma^2 I$ are invertible. For $\sigma>0$, $WK_GW^T+\sigma^2 I$ is positive definite, which implies invertibility. K_GW^TW is similar to the positive semidefinite matrix $K_G^{1/2}W^TWK_G^{1/2}$, and thus has all non-negative eigenvalues. Thus, for $\sigma>0$, $K_GW^TW+\sigma^2 I$ has all positive eigenvalues (in particular, it has no zero eigenvalue) and so is invertible.

Covariance: The two expressions for covariance are $\mathbf{w}_x\tilde{\mathbf{C}}\mathbf{w}_x$ (Def. 1) and $\mathbf{w_x}\bar{\mathbf{C}}\mathbf{w}_x$ (Fact. 2) with

$$\begin{aligned}\tilde{\mathbf{C}} &= K_G - K_GW^T(WK_GW^T+\sigma^2 I)^{-1}WK_G\\ \bar{\mathbf{C}} &= \sigma^2\left(K_GW^TW+\sigma^2 I\right)^{-1}K_G\end{aligned}$$

So it suffices to show that $\tilde{\mathbf{C}}=\bar{\mathbf{C}}$. Using Eq. 7, we have that

$$\tilde{\mathbf{C}} = K_G-(K_GW^TW+\sigma^2 I)^{-1}K_GW^TWK_G.$$

Now, factor out K_G and simplify to get:

$$\begin{aligned}\tilde{\mathbf{C}} &= [I-(K_GW^TW+\sigma^2 I)^{-1}K_GW^TW]K_G\\ &= [(K_GW^TW+\sigma^2 I)^{-1}\cdot(K_GW^TW+\sigma^2 I-K_GW^TW)]K_G\\ &= \sigma^2(K_GW^TW+\sigma^2 I)^{-1}K_G = \bar{\mathbf{C}}\end{aligned}$$

Log likelihood: By matching terms in the two expressions and using the definition of $\tilde{K}_X$, it suffices to show both of the following:

$$\mathbf{y}^T\tilde{\mathbf{z}} = \frac{\mathbf{y}^T(\mathbf{y} - W\bar{\mathbf{z}})}{\sigma^2}, \tag{8}$$

$$\log\det(WK_GW^T + \sigma^2 I) = \log\det(K_GW^TW + \sigma^2 I) + (n-m)\log\sigma^2. \tag{9}$$

For Eq. (8), we first observe that by our argument above that $\bar{\mathbf{z}} = K_GW^T\tilde{\mathbf{z}}$, we have

$$\frac{\mathbf{y}^T(\mathbf{y} - W\bar{\mathbf{z}})}{\sigma^2} = \frac{\mathbf{y}^T(\mathbf{y} - WK_GW^T\tilde{\mathbf{z}})}{\sigma^2} = \frac{\mathbf{y}^T(I - WK_GW^T(WK_GW^T + \sigma^2 I)^{-1})\mathbf{y}}{\sigma^2}. \tag{10}$$

We have:

$$\begin{aligned}(I - WK_GW^T(WK_GW^T + \sigma^2 I)^{-1}) &= [WK_GW^T + \sigma^2 I - WK_GW^T](WK_GW^T + \sigma^2 I)^{-1} \\ &= \sigma^2(WK_GW^T + \sigma^2 I)^{-1}.\end{aligned}$$

Thus we can simplify (10) to:

$$\frac{\mathbf{y}^T(\mathbf{y} - W\bar{\mathbf{z}})}{\sigma^2} = \mathbf{y}^T(WK_GW^T + \sigma^2 I)^{-1}\mathbf{y} = \mathbf{y}^T\tilde{\mathbf{z}}. \tag{11}$$

It remains to prove Eq. 9. This follows from the general claim:

Claim 6. *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times m}$. Then:*

$$\det(BA + \sigma^2 I_n) = (\sigma^2)^{n-m}\det(AB + \sigma^2 I_m)$$

Proof. The claim generalizes the Weinstein-Aronszajn identity, which states that $\det(BA + I_n) = \det(AB + I_m)$. It can be proven using the block determinant formulas in (Gubner, 2015). Let $C = \begin{bmatrix}\sigma I_m & -A \\ B & \sigma I_n\end{bmatrix}$. We have:

$$\begin{aligned}\det(C) = \det(\sigma I_m)\cdot\det(B(\sigma I_m)^{-1}A + \sigma I_n) &= \det(\sigma I_m)\cdot\det\left(\frac{1}{\sigma}(BA + \sigma^2 I_n)\right) \\ &= \det(\sigma I_m)\cdot\det\left(\frac{1}{\sigma}I_n\right)\cdot\det(BA + \sigma^2 I_n)\end{aligned}$$

and similarly

$$\begin{aligned}\det(C) = \det(\sigma I_n)\cdot\det(A(\sigma I_n)^{-1}B + \sigma I_m) &= \det(\sigma I_n)\cdot\det\left(\frac{1}{\sigma}(AB + \sigma^2 I_m)\right) \\ &= \det(\sigma I_n)\cdot\det\left(\frac{1}{\sigma}I_m\right)\cdot\det(AB + \sigma^2 I_m)\end{aligned}$$

Thus,

$$\begin{aligned}\det(\sigma I_m)\cdot\det\left(\frac{1}{\sigma}I_n\right)\cdot\det(BA + \sigma^2 I_n) &= \det(\sigma I_n)\cdot\det\left(\frac{1}{\sigma}I_m\right)\cdot\det(AB + \sigma^2 I_m) \\ \det(BA + \sigma^2 I_n) &= \sigma^{2(n-m)}\cdot\det(AB + \sigma^2 I_m),\end{aligned}$$

giving the claim. ☐

Applying Claim 6 with $A = K_GW^T$ and $B = W$ gives that $\det(\tilde{W}K_GW^T + \sigma^2 I) = (\sigma^2)^{n-m}\det(K_GW^TW + \sigma^2 I)$ which in turn gives that $\log\det(WK_GW^T + \sigma^2 I) = \log\det(K_GW^TW + \sigma^2 I) + (n-m)\log\sigma^2$ and so completes Eq. (9) and the theorem. ☐

A.3 Symmetric Reformulation of GSGP

Recall that directly replacing the SKI approximate inference equations of Definition 1 with the GSGP inference equations of Fact 2 reduces per-iteration cost from $\mathcal{O}(n + m \log m)$ to $\mathcal{O}(m \log m)$. However, the matrix $K_G W^T W + \sigma^2 I$ is *asymmetric*, which prevents the application of symmetric system solvers like the conjugate gradient method with strong guarantees.

Here we describe a symmetric reformulation of GSGP that doesn't compromise on per-iteration cost. We write the matrix $(K_G W^T W + \sigma^2 I)^{-1}$, whose application is required in both the posterior mean and covariance computation as:

$$\begin{aligned}(K_G W^T W + \sigma^2 I)^{-1} &= (K_G W^T W K_G K_G^{-1} + \sigma^2 K_G K_G^{-1})^{-1} \\ &= K_G (K_G W^T W K_G + \sigma^2 K_G)^{-1}.\end{aligned}$$

Applying the above matrix requires a symmetric solve in $(K_G W^T W K_G + \sigma^2 K_G)^{-1}$, along with a single $\mathcal{O}(m \log m)$ time MVM with K_G. Our per-iteration complexity thus remains at $\mathcal{O}(m \log m)$ – the matrix vector multiplication time for $K_G W^T W K_G + \sigma^2 K_G$. However, this matrix is no longer of the 'regularized form' $A + \sigma^2 I$, and may have worse condition number than $W K_G W^T + \sigma^2 I$, possibly leading to slower convergence of iterative solvers like CG as compared to SKI.

We can similarly symmetrize the logdet computation in the likelihood expression by writing

$$\log\det(K_G W^T W + \sigma^2 I) = \log\det(K_G W^T W K_G + \sigma^2 K_G) - \log\det(K_G).$$

Again however, it is unclear how this might affect the convergence of iterative methods for logdet approximation

B Factorized Iterative Methods – Omitted Details

B.1 Proofs for Factorized Update Steps

We give proofs of our fundamental factorized update steps described in Claims 3 and 4.

Claim 3. *For any $\mathbf{z}_i \in \mathbb{R}^n$ with $\mathbf{z}_i = W\hat{\mathbf{z}}_i + c_i \mathbf{z}_0$,*

$$(W K_G W^T + \sigma^2 I)\mathbf{z}_i = W\hat{\mathbf{z}}_{i+1} + c_{i+1}\mathbf{z}_0,$$

where $\hat{\mathbf{z}}_{i+1} = (K_G W^T W + \sigma^2 I)\hat{\mathbf{z}}_i + c_i K_G W^T \mathbf{z}_0$ and $c_{i+1} = \sigma^2 \cdot c_i$. Call this operation a factorized update *and denote it as $(\hat{\mathbf{z}}_{i+1}, c_{i+1}) = \mathcal{A}(\hat{\mathbf{z}}_i, c_i)$. If the vector $K_G W^T \mathbf{z}_0$ is precomputed in $\mathcal{O}(n + m \log m)$ time, each subsequent factorized update takes $\mathcal{O}(m \log m)$ time.*

Proof. We have:

$$\begin{aligned}(W K_G W^T + \sigma^2 I)\mathbf{z}_i &= (W K_G W^T + \sigma^2 I)W\hat{\mathbf{z}}_i + c_i (W K_G W^T + \sigma^2 I) z_0 \\ &= W(K_G W^T W + \sigma^2 I)\hat{\mathbf{z}}_i + c_i W(K_G W^T z_0) + c_i \cdot \sigma^2 \cdot z_0 \\ &= W\left[(K_G W^T W + \sigma^2 I)\hat{\mathbf{z}}_i + c_i K_G W^T z_0\right] + c_i \cdot \sigma^2 \cdot z_0.\end{aligned}$$

which completes the derivation of the update.

As discussed in Sec. 3.1, it takes $\mathcal{O}(n + m \log m)$ time to precompute $K_G W^T \mathbf{z}_0$, after which: it takes $\mathcal{O}(m \log m)$ time to compute $(K_G W^T W + \sigma^2 I)\hat{\mathbf{z}}_i$, it takes $\mathcal{O}(m)$ time to add in $c_i K_G W^T \mathbf{z}_0$, and it takes $\mathcal{O}(1)$ time to update c_i, for a total of $\mathcal{O}(m \log m)$ time for each update.

□

Claim 4. *For any $\mathbf{z}_i, \mathbf{y}_i \in \mathbb{R}^n$ with $\mathbf{z}_i = W\hat{\mathbf{z}}_i + c_i \mathbf{z}_0$ and $\mathbf{y}_i = W\hat{\mathbf{y}}_i + d_i \mathbf{y}_0$,*

$$\mathbf{z}_i^T \mathbf{y}_i = \hat{\mathbf{z}}_i^T W^T W \hat{\mathbf{y}}_i + d_i \hat{\mathbf{z}}_i^T W^T \mathbf{y}_0 + c_i \hat{\mathbf{y}}_i^T W^T \mathbf{z}_0 + c_i d_i \mathbf{y}_0^T \mathbf{z}_0.$$

We denote the above operation by $\langle(\hat{\mathbf{z}}_i, c_i), (\hat{\mathbf{y}}_i, d_i)\rangle$.

Proof. We have:

$$\mathbf{z}_i^T \mathbf{y}_i = (W\hat{\mathbf{z}}_i + c_i \mathbf{z}_0)^T (W\hat{\mathbf{y}}_i + d_i \mathbf{y}_0) = \hat{\mathbf{z}}_i^T W^T W \hat{\mathbf{y}}_i + d_i \hat{\mathbf{z}}_i^T W^T \mathbf{y}_0 + c_i \hat{\mathbf{y}}_i^T W^T \mathbf{z}_0 + c_i d_i \mathbf{y}_0^T \mathbf{z}_0,$$

giving the claim. □

Algorithm 3 Efficiently Factorized Conjugate Gradient (EFCG)

```
1: procedure EFCG($K_G, W, \mathbf{b}, \sigma, \mathbf{x}_0, \epsilon$)
2:     New Iterates:
3:         $\hat{\mathbf{v}}_k = \left[K_G W^T W + \sigma^2 I\right] \hat{\mathbf{r}}_k, \hat{\mathbf{u}}_k = W^T W \hat{\mathbf{r}}_k$
4:         $\hat{\mathbf{z}}_k = \left[K_G W^T W + \sigma^2 I\right] \hat{\mathbf{p}}_k,\ \hat{\mathbf{s}}_k = W^T W \hat{\mathbf{p}}_k$
5:     $\mathbf{r_0} = \mathbf{b} - \tilde{K}\mathbf{x}_0,\ \hat{\mathbf{r}}_\mathbf{0} = \mathbf{0},\ c_0^r = 1$
6:     $\hat{\mathbf{p}}_0 = \mathbf{0},\ c_0^p = 1,\ \hat{\mathbf{x}}_0 = \mathbf{0},\ c_0^x = 0$
7:     $\hat{\mathbf{v}}_0 = \mathbf{0}, \hat{\mathbf{u}}_0 = \mathbf{0}$
8:     $\hat{\mathbf{z}}_0 = \mathbf{0}, \hat{\mathbf{s}}_0 = \mathbf{0}$
9:     for $k = 0$ to maxiter do
10:        $\alpha_k = \frac{\hat{\mathbf{u}}_k^T \hat{\mathbf{r}}_k + c_k^r c_k^r \|\mathbf{r}_0\| + 2c_i^r (\hat{\mathbf{r}}_i^T W^T \mathbf{r}_0)}{\hat{\mathbf{s}}_k^T \hat{\mathbf{z}}_k + c_k^z c_k^p \|\mathbf{r}_0\| + c_k^p (\hat{\mathbf{z}}_k^T W^T \mathbf{r}_0) + c_k^z (\hat{\mathbf{r}}_k^T W^T \mathbf{r}_0)}$
11:        $(\hat{\mathbf{x}}_{k+1}, c_{k+1}^x) = (\hat{\mathbf{x}}_k, c_k^x) + \alpha_k \cdot (\hat{\mathbf{p}}_k, c_k^p)$
12:        $(\hat{\mathbf{r}}_{k+1}, c_{k+1}^r) = (\hat{\mathbf{r}}_k, c_k^r) - \alpha_k \cdot (\hat{\mathbf{z}}_k + c_k^p \cdot (K_G W^T \mathbf{r}_0), \sigma^2 c_k^p)$
13:        $[\hat{\mathbf{v}}_{k+1}, \hat{\mathbf{u}}_{k+1}] = \mathcal{B}(\hat{\mathbf{r}}_{k+1})$
14:        if $\hat{\mathbf{u}}_{k+1}^T \hat{\mathbf{r}}_{k+1} + c_{k+1}^r c_{k+1}^r \|\mathbf{r}_0\| + 2c_{k+1}^r (\hat{\mathbf{r}}_{k+1}^T W^T \mathbf{r}_0) \le \epsilon$ exit loop
15:        $\beta_k = \frac{\hat{\mathbf{u}}_{k+1}^T \hat{\mathbf{r}}_{k+1} + c_{k+1}^r c_{k+1}^r + 2c_{k+1}^r (\hat{\mathbf{r}}_{k+1}^T W^T \mathbf{r}_0)}{\hat{\mathbf{u}}_k^T \hat{\mathbf{r}}_k + c_k^r c_k^r \|\mathbf{r}\|_0 + 2c_k^r (\hat{\mathbf{r}}_k^T W^T \mathbf{r}_0)}$
16:        $(\hat{\mathbf{p}}_{k+1}, c_{k+1}^p) = (\hat{\mathbf{r}}_{k+1}, c_{k+1}^r) + \beta_k \cdot (\hat{\mathbf{p}}_k, c_k^p)$
17:        $\hat{\mathbf{s}}_{k+1} = \hat{\mathbf{u}}_{k+1} + \beta_k \cdot \hat{\mathbf{s}}_k$
18:        $\hat{\mathbf{z}}_{k+1} = \hat{\mathbf{v}}_{k+1} + \beta_k \cdot \hat{\mathbf{z}}_k$
19:    return $\mathbf{x}_{k+1} = W\hat{\mathbf{x}}_{k+1} + c_{k+1}^x \cdot \mathbf{r}_0 + \mathbf{x}_0$
```

B.2 Efficiently Factorized Conjugate Gradient Algorithms

We now present Efficiently factorized conjugate gradient (EFCG) (Algorithm 3), which improves on our basic Factorized CG algorithm (Algorithm 2) by avoiding any extra multiplication with W and $W^T W$. The central idea of EFCG is to exploit the fact that each time we perform a matrix-vector multiplication with the GSGP operator $K_G W^T W + \sigma^2 I$, we also must perform one with $W^T W$. We can save the result of this multiplication to avoid repeated work. In particular, this lets us avoid extra MVM costs associated with $W^T W$ present in factorized inner products steps of Algorithm 2. Similar to Algorithm 2, Algorithm 3 also maintains iterates CG (Algorithm 1) exactly in the same compressed form $\mathbf{x}_k = W\hat{\mathbf{x}}_k + c_k^x \mathbf{r}_0 + \mathbf{x}_0$, $\mathbf{r}_k = W\hat{\mathbf{r}}_k + c_k^r \mathbf{r}_0$, and $\mathbf{p}_k = W\hat{\mathbf{p}}_k + c_k^p \mathbf{r}_0$.

We let $[\mathbf{v}, \mathbf{u}] = \mathcal{B}(\mathbf{x})$ denote the operation that returns $\mathbf{v} = \left[K_G W^T W + \sigma^2 I\right] \mathbf{x}$ and $\mathbf{u} = W^T W \mathbf{x}$. Since $\mathbf{u}$ must be computed as in intermediate step in computing $\mathbf{v}$, this operation has the same cost as a standard matrix-vector-multiplicaiton with $\left[K_G W^T W + \sigma^2 I\right]$. Notice that Algorithm 3 performs just one $\mathcal{B}(\mathbf{x})$ operation per iteration, requiring a single matrix vector multiplication with each of K_G and $W^T W$ per iteration. Both $K_G W^T \mathbf{r}_0$ and $W^T \mathbf{r}_0$ are precomputed.

In addition to $\mathcal{B}(\mathbf{x})$ operation, the superior efficiency of the EFCG Algorithm 3 over CG and Factorized CG can mainly be attributed to following facts:

- It uses four new iterates: $\hat{\mathbf{v}}_k = \left[K_G W^T W + \sigma^2 I\right] \hat{\mathbf{r}}_k$, $\hat{\mathbf{u}}_k = W^T W \hat{\mathbf{r}}_k$, $\hat{\mathbf{z}}_k = \left[K_G W^T W + \sigma^2 I\right] \hat{\mathbf{p}}_k$ and $\hat{\mathbf{s}}_k = W^T W \hat{\mathbf{p}}_k$. Given these iterates all factorized inner products can be computed without any extra multiplication with $W^T W$.

- In case, the initial solution $\mathbf{x}_0 = \mathbf{0}$, which is the most common choice in practice for CG, $\tilde{K}\mathbf{x}_0$ multiplication can be avoided. Also, observe that in SKI mean and covariance approximation (Definition 1), we only need $W^T \mathbf{x}_{k+1}$ which is equal to $W^T W \hat{\mathbf{x}}_{k+1} + W^T \mathbf{r}_0 + W^T \mathbf{x}_0$. Since, $W^T \mathbf{r}_0$ is pre-computed and $W^T \mathbf{x}_0 = 0$, no extra multiplication with W or $W^T W$ is required other than computing $K_G W^T \mathbf{r}_0$ and $W^T \mathbf{r}_0$.

In Algorithm 4, we present a further simplified variant on EFCG for the case when initial residual $\mathbf{r}_0$ is in the span of W, to directly compute $W^T\mathbf{x}_{k+1}$, where $\mathbf{x}_{k+1}$ is the final solution returned by the EFCG algorithm. Observe SKI mean and covariance expressions of SKI definition (i.e., Definition 1), we always need to post-process $\mathbf{x}_{k+1}$ as $W^T\mathbf{x}_{k+1}$ to estimate them. Unlike EFCG, simplified EFCG (i.e., Algorithm 4) maintains a compressed form for $\mathbf{r}_k$ using $\hat{\mathbf{r}}_k$ (as $\mathbf{r}_k = W\hat{\mathbf{r}}_k$) and doesn't maintain $\mathbf{p}_k$. In addition to that, Algorithm 4 also maintains another iterate $\hat{\mathbf{x}}^d_{k+1}$ such that $W^T\mathbf{x}_{k+1} = \hat{\mathbf{x}}^d_{k+1} + W^T\mathbf{x}_0$.

Algorithm 4 Simplified EFCG – Initial residual (i.e., $\mathbf{r}_0$) is in span of W

```
1:  procedure EFCG($K_G, W, \hat{\mathbf{r}}_0, \sigma, \epsilon$)
2:      New Iterates:
3:              $\hat{\mathbf{v}}_k = \left[K_G W^T W + \sigma^2 I\right]\hat{\mathbf{r}}_k, \hat{\mathbf{u}}_k = W^T W \hat{\mathbf{r}}_k$
4:              $\hat{\mathbf{z}}_k = \left[K_G W^T W + \sigma^2 I\right]\hat{\mathbf{p}}_k,\ \hat{\mathbf{s}}_k = W^T W \hat{\mathbf{p}}_k$
5:      $\hat{\mathbf{x}}^d_0 = \mathbf{0}$
6:      $\hat{\mathbf{v}}_0, \hat{\mathbf{u}}_0 = \mathcal{B}(\hat{\mathbf{r}}_0)$
7:      $\hat{\mathbf{z}}_0 = \hat{\mathbf{v}}_0, \hat{\mathbf{s}}_0 = \hat{\mathbf{u}}_0$
8:      for $k = 1$ to maxiter do
9:          $\alpha_k = \frac{\hat{\mathbf{u}}_k^T \hat{\mathbf{r}}_k}{\hat{\mathbf{s}}_k^T \hat{\mathbf{z}}_k}$
10:         $\hat{\mathbf{x}}^d_{k+1} = \hat{\mathbf{x}}^d_k + \alpha_k \cdot \hat{\mathbf{s}}_k$
11:         $\hat{\mathbf{r}}_{k+1} = \hat{\mathbf{r}}_k - \alpha_k \cdot \hat{\mathbf{z}}_k$
12:         $\hat{\mathbf{v}}_{k+1}, \hat{\mathbf{u}}_{k+1} = \mathcal{B}(\hat{\mathbf{r}}_{k+1})$
13:         if $\hat{\mathbf{u}}_{k+1}^T \hat{\mathbf{r}}_{k+1} \le \epsilon$ exit loop
14:         $\beta_k = \frac{\hat{\mathbf{u}}_{k+1}^T \hat{\mathbf{r}}_{k+1}}{\hat{\mathbf{u}}_k^T \hat{\mathbf{r}}_k}$
15:         $\hat{\mathbf{s}}_{k+1} = \hat{\mathbf{u}}_{k+1} + \beta_k \cdot \hat{\mathbf{s}}_k$
16:         $\hat{\mathbf{z}}_{k+1} = \hat{\mathbf{v}}_{k+1} + \beta_k \cdot \hat{\mathbf{z}}_k$
17:     return $\hat{\mathbf{x}}^d_{k+1}$
```

The requirement of initial residual $\mathbf{r}_0$ to be in the span of W can be met in two ways. For example, for SKI posterior mean inference, we set $\mathbf{x}_0 = \frac{1}{\sigma^2}\cdot\mathbf{y}$ implying $\mathbf{r}_0 = \mathbf{y} - \left[WK_GW^T + \sigma^2 I\right]\frac{1}{\sigma^2}\cdot\mathbf{y} = -\frac{1}{\sigma^2}\cdot WK_GW^T\mathbf{y}$, which lies in the span of W. Consequently, we initiate Algorithm 4 with $\hat{\mathbf{r}}_0 = -\frac{1}{\sigma^2}\cdot K_GW^T\mathbf{y} \in \mathbb{R}^{m\times 1}$ and following the invariance of simplified EFCG algorithm, compute transformed solutions as $W^T\mathbf{x}_{k+1} = \hat{\mathbf{x}}^d_{k+1} + \frac{1}{\sigma^2}\cdot W^T y$. Notice that simplified EFCG (unlike EFCG) does not require pre-computations of terms $K_GW^T\mathbf{r}_0$ and $W^T\mathbf{r}_0$. In fact, simplified ECFG requires only multiplication with W^T, i.e., to obtain W^Ty which is sufficient for both, initialization of $\mathbf{r}_0$ and also to compute the transformation of final solution $W^T\mathbf{x}_{k+1}$ (which is equal to $W^T\left(\tilde{K}_X + \sigma^2 I\right)^{-1}\mathbf{y}$). As a result, simplified EFCG can compute SKI posterior mean using only sufficient statistics (W^TW and $W^T\mathbf{y}$).

A second way to meet the requirement of the initial residual $\mathbf{r}_0$ being in the span of W, is by setting $\mathbf{x}_0 = \mathbf{0}$ when $\hat{\mathbf{y}}$ is provided such that $\mathbf{y} = W\hat{\mathbf{y}}$. This is the case, e.g., in posterior covariance approximation. The final transformed solution $W^T\mathbf{x}_{k+1}$ in this setting reduces to $\hat{\mathbf{x}}^d_{k+1}$. Notice for this setting also, we need only W^TW and do not require the matrix W.

Consquently, simplified EFCG can compute both SKI posterior mean and covariance function using onlyt he sufficient statistics (W^TW and $W^T\mathbf{y}$) and without even realizing W matrix in memory.

B.3 Factorized Lanczos Algorithms

The Lanczos algorithm can be utilized to factorize a symmetric matrix $A \in \mathbb{R}^{n\times n}$ as QTQ^T such that $T \in \mathbb{R}^{n\times n}$ is a symmetric tridiagonal matrix and $Q \in \mathbb{R}^{n\times n}$ is orthonormal matrix. Previously, it has been used with k iterations (i.e. Algorithm 5) to compute low-rank and fast approximations of SKI covariance matrix and log-likelihood of the data. For further details, we refer readers to (Pleiss et al., 2018; Dong et al., 2017).

Similar to Factorized CG, we derive factorized Lanczos algorithm (FLA) using factorized inner products and matrix vector multiplication, as described in Algorithm 6. We maintain all iterates in $\mathbb{R}^{m\times 1}$ similar to Factorized CG, in particular, $Q_{:,i} \in \mathbb{R}^{n\times 1}$ vectors are maintained in compressed form such that $Q_{:,i} = W\hat{Q}_{:,i} + d_i \cdot \mathbf{b}$. The

Algorithm 5 Lanczos Algorithm (LA)

```
1: procedure LA($K_G, W, \mathbf{b}, \sigma, k$)
2:     $\mathbf{q}_0 = \mathbf{0}, \mathbf{q}_1 = \mathbf{b}, \beta_1 = 0$
3:     $Q_{:,1} = \mathbf{q}_1$
4:     for $i = 1$ to $k$ do
5:         $\mathbf{q}_{i+1} = \tilde{K}\mathbf{q}_i - \beta_i \cdot \mathbf{q}_{i-1}$
6:         $\alpha_i = \mathbf{q}_i^T \mathbf{q}_{i+1}$
7:         $T_{i,i} = \alpha_i$
8:         if $i == k$ then exit loop
9:         $\mathbf{q}_{i+1} = \mathbf{q}_i - \alpha_i \cdot \mathbf{q}_i$
10:        $\mathbf{q}_{i+1} = \mathbf{q}_{i+1} - [Q_{:,1}, ..., Q_{:,i}] \left( [Q_{:,1}, ..., Q_{:,i}]^T \mathbf{q}_{i+1} \right)$
11:        $\beta_{i+1} = \|\mathbf{q}_{i+1}\|$
12:        $T_{i,i+1} = T_{i,i+1} = \beta_{i+1}$
13:        $\mathbf{q}_{i+1} = \frac{1}{\beta_{i+1}} \cdot \mathbf{q}_{i+1}$
14:        $Q_{:,i+1} = \mathbf{q}_{i+1}$
15:    return $Q$, $T$
```

Algorithm 6 Factorized Lanczos Algorithm (FLA)

```
1: procedure FLA($K_G, W, \mathbf{b}, \sigma, k$)
2:     $\hat{\mathbf{q}}_0 = \mathbf{0}, c_0^q = 0, \hat{\mathbf{q}}_1 = \mathbf{0}, c_1^q = 1, \beta_1 = 0$
3:     $\hat{Q}_{:,1} = \hat{\mathbf{q}}_1, \mathbf{\Lambda} = \mathbf{0} \in \mathbb{R}^{k\times 1}, \mathbf{d} = \mathbf{0} \in \mathbb{R}^{k\times 1}$
4:     for $i = 1$ to $k$ do
5:         $(\hat{\mathbf{q}}_{i+1}, c_{i+1}^q) = (\hat{\mathbf{q}}_i, c_i^q) - \beta_i \cdot \mathcal{A}(\hat{\mathbf{q}}_{i-1}, c_{i-1}^q)$
6:         $\alpha_i = \langle (\hat{\mathbf{q}}_i, c_i^q), (\hat{\mathbf{q}}_{i+1}, c_{i+1}^q) \rangle$
7:         $T_{i,i} = \alpha_i; \quad \mathbf{d}_i = c_i^q$
8:         if $i == k$ then exit loop
9:         $(\hat{\mathbf{q}}_{i+1}, c_{i+1}^q) = (\hat{\mathbf{q}}_i, c_i^q) - \alpha_i \cdot (\hat{\mathbf{q}}_i, c_i^q)$
10:        $\mathbf{\Lambda}_j = \langle (\hat{Q}_{:,j}, c_j^q), (\hat{\mathbf{q}}_{i+1}, c_{i+1}^q) \rangle; \quad \forall j \in \{1, ..., i\}$
11:        $\hat{\mathbf{q}}_{i+1} = \hat{\mathbf{q}}_{i+1} - \hat{Q}_{:,1:i}\mathbf{\Lambda}_{1:i}; \quad c_{i+1}^q = c_{i+1}^q - \mathbf{d}_{1:i}^T \mathbf{\Lambda}_{1:i}$
12:        $\beta_{i+1} = \sqrt{\langle (\hat{\mathbf{q}}_{i+1}, c_{i+1}^q), (\hat{\mathbf{q}}_{i+1}, c_{i+1}^q) \rangle}$
13:        $T_{i,i+1} = T_{i,i+1} = \beta_{i+1}$
14:        $\hat{\mathbf{q}}_{i+1} = \frac{1}{\beta_{i+1}} \cdot \hat{\mathbf{q}}_{i+1}; \quad c_{i+1}^q = \frac{c_{i+1}^q}{\beta_{i+1}}$
15:        $\hat{Q}_{:,i+1} = \hat{\mathbf{q}}_{i+1}$
16:    return $\hat{Q}$, $T$, $\mathbf{d}$
```

$T \in \mathbb{R}^{k \times k}$ matrix of Lanczos algorithm is retained as it is in FLA. Next, in a manner similar to EFCG, we derive efficient factorized algorithm (EFLA) as shown in Algorithm 7. Specifically, EFLA relies on two new iterates: $\hat{P}_{:,i} = W^T W \hat{Q}_{:,i}$ and $\hat{\mathbf{s}}_i = \left[K_G W^T W + \sigma I\right] \hat{\mathbf{q}}_i$ and maintains iterates of Lanczos algorithm as $Q_{:,i} = W\hat{Q}_{:,i} + d_i \cdot \mathbf{b}$, similar to FLA. Notice that EFLA only requires one $\mathcal{B}(\mathbf{x})$ operation per loop thereby avoiding any extra MVMs with W and $W^T W$, except one time pre-computations of $K_G W^T \mathbf{b}$ and $W^T \mathbf{b}$.

Algorithm 7 Efficiently Factorized Lanczos algorithm (EFLA)

```
1: procedure EFLA($K_G, W, \mathbf{b}, \sigma, k$)
2:     New Iterates:
3:         $\hat{P}_{:,i} = W^T W \hat{Q}_{:,i}$ and $\hat{\mathbf{s}}_i = \left[K_G W^T W + \sigma I\right] \hat{\mathbf{q}}_i$
4:     $\hat{\mathbf{q}}_0 = \mathbf{0}, c_0^q = 0, \hat{\mathbf{q}}_1 = \mathbf{0}, c_1^q = 1, \beta_1 = 0$
5:     $\hat{Q}_{:,1:k} = [\mathbf{0}, ..., \mathbf{0}] \in \mathbb{R}^{m \times k}, \hat{P}_{:,1:k} = [\mathbf{0}, ..., \mathbf{0}] \in \mathbb{R}^{m \times k}$
6:     $\hat{Q}_{:,1} = \hat{\mathbf{q}}_1, \mathbf{\Lambda} = \mathbf{0} \in \mathbb{R}^{k \times 1}, \hat{\mathbf{s}}_i = \mathbf{0} \in \mathbb{R}^{m \times 1}, \mathbf{d} = \mathbf{0} \in \mathbb{R}^{k \times 1}$
7:     for $i = 1$ to $k$ do
8:         $\hat{\mathbf{q}}_{i+1} = \hat{\mathbf{s}}_i + c_i^q \cdot K_G W^T \mathbf{b} - \beta_i \cdot \hat{\mathbf{q}}_{i-1}$
9:         $c_{i+1}^q = \sigma^2 c_i^q - \beta_i c_{i-1}^q$
10:        $\alpha_i = \hat{P}_{:,i}^T \hat{\mathbf{q}}_{i+1} + c_i^q c_{i+1}^q + \left(c_i^q \cdot \hat{\mathbf{q}}_{i+1} + c_{i+1}^q \cdot \hat{\mathbf{q}}_i\right)^T W^T \mathbf{b}$
11:        $T_{i,i} = \alpha_i; \quad \mathbf{d}_i = c_i$
12:        if $i == k$ then exit loop
13:        $(\hat{\mathbf{q}}_{i+1}, c_{i+1}^q) = (\hat{\mathbf{q}}_i, c_i^q) - \alpha_i \cdot (\hat{\mathbf{q}}_i, c_i^q)$
14:        $\mathbf{\Lambda}_{1:i} = \hat{P}_{:,1:i}^T \hat{\mathbf{q}}_{i+1} + \left(c_{i+1}^q + \hat{\mathbf{q}}_{i+1}^T W^T \mathbf{b}\right) \cdot \mathbf{c}_{1:i} + c_{i+1}^q \cdot \hat{Q}_{:,1:i}^T W^T \mathbf{b}$
15:        $\hat{\mathbf{q}}_{i+1} = \hat{\mathbf{q}}_{i+1} - \hat{Q}_{:,1:i} \mathbf{\Lambda}_{1:i}; \quad c_{i+1}^q = c_{i+1}^q - \mathbf{d}_{1:i}^T \mathbf{\Lambda}_{1:i}$
16:        $\hat{\mathbf{s}}_{i+1}, \hat{P}_{:,i+1} = \mathcal{B}(\hat{\mathbf{q}}_{i+1})$
17:        $\beta_{i+1} = \sqrt{\hat{P}_{:,i+1}^T \hat{\mathbf{q}}_{i+1} + c_{i+1}^q c_{i+1}^q + 2 c_{i+1}^q \hat{\mathbf{q}}_{i+1}^T W^T \mathbf{b}}$
18:        $T_{i,i+1} = T_{i,i+1} = \beta_{i+1}$
19:        $\hat{\mathbf{q}}_{i+1} = \frac{1}{\beta_{i+1}} \cdot \hat{\mathbf{q}}_{i+1}; \quad c_{i+1}^q = \frac{c_{i+1}^q}{\beta_{i+1}}$
20:        $\hat{\mathbf{s}}_{i+1} = \frac{1}{\beta_{i+1}} \cdot \hat{\mathbf{s}}_{i+1}; \quad \hat{P}_{:,i+1} = \frac{1}{\beta_{i+1}} \cdot \hat{P}_{:,i+1}$
21:        $\hat{Q}_{:,i+1} = \hat{\mathbf{q}}_{i+1}$
22:    return $\hat{Q}$, $T$, $\mathbf{d}$
```

C Experiments – Omitted Details and Additional Results

C.1 Hardware and hyper-parameters details

We run all of our experiments on Intel Xeon Gold 6240 CPU @ 2.60GHz with 10 GB of RAM. In all experimental settings, our kernels are squared exponential kernels wrapped within a scale kernel (Wilson and Nickisch, 2015; Dong et al., 2017). Therefore, our hyper-parameters are σ, length-scales and output-scale as also presented in Table 2. Length-scales are specific to each dimension for multi-dimensional datasets. For sine and sound datasets, we have utilized GPytorch to optimize hyper-parameters. For precipitation and radar datasets, we have considered previously optimized parameters in (Dong et al., 2017) and in (Angell and Sheldon, 2018), respectively.

Dataset	σ	Length-scale	Output-scale
Sine	0.074	0.312	1.439
Sound	0.009	10.895	0.002
Radar	50.000	[0.250, 0.250, 200]	3.500
Precipitation	3.990	[3.094, 2.030, 0.189]	2.786

Table 2: Hyper-parameters used for all datasets. Length-scale is of size d of each dataset.

C.2 Results: Synthetic sine dataset

Figure 6 depicts the number of iteration and pre-processing time taken by GSGP and SKI for synthetic sine dataset wrt number of sample, for the setting on which Figure 3 reports the results. The number of iterations for GSGP and SKI are always close and possibly differ only due to finite precision.

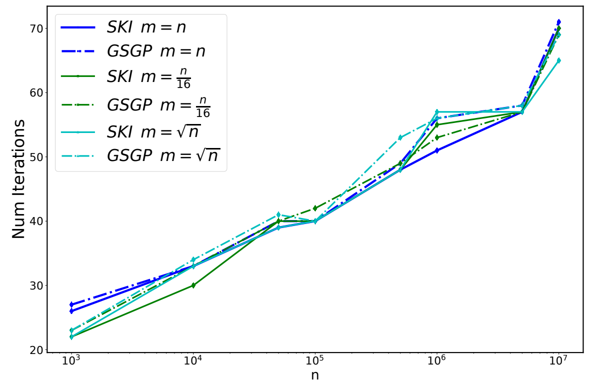

Figure 6: Number of iterations taken by SKI and GSGP on synthetic dataset. Results are averaged over 8 trials.

C.3 Results: Precipitation dataset

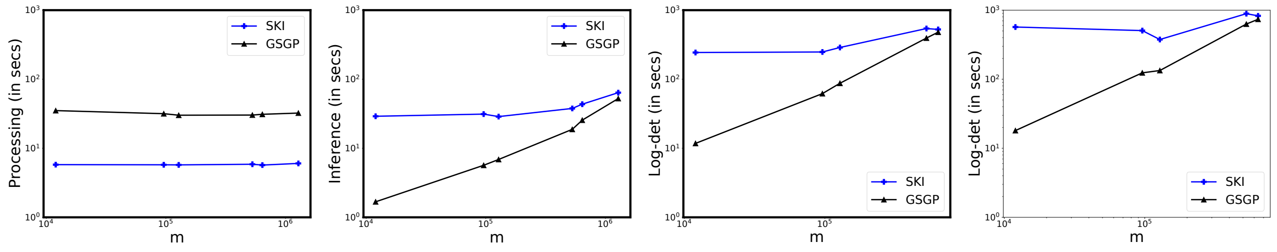

Figure 7: Inference time vs. m for SKI inference tasks on the precipitation data set. From left to right: pre-processing, mean inference, log-determinant for $tol = 0.1$ and for $tol = 0.01$ and using 30 random vectors.

Figure 7 shows running time vs. m for GP inference tasks on precipitation data set of $n = 528\text{K}$ (Dong et al., 2017). We consider $m \in \{12\text{K}, 96\text{K}, 128\text{K}, 528\text{K}, 640\text{K}\}$. This is a situation where even for $m > n$, GSGP is faster compared to SKI. Pre-processing is up to 6x slower for GSGP due to the need to compute $W^T W$. To perform only one mean inference, the overall time of GSGP and SKI *including* pre-processing is similar as some of the per-iteration gains are offset by pre-processing. However, for the log-determinant computation task (as part of the log-likelihood computation), *several* more iterations of linear solvers are required as also demonstrated by log-det computation in Figure 7. It is worth noting that pre-processing of GSGP is required only once which can be performed initially for the log-det computation and later be utilized for the posterior mean and covariance inference. Therefore, overall, GSGP is more effective than SKI for all inference tasks.