Supplementary Material

High-Dimensional Gaussian Process Inference with Derivatives

A. Linear Algebra

Kronecker products play an important role in the derivations so here list a few properties that will be useful, see (Van Loan, 2000) for more. The Kronecker product for a matrix $A \in \mathbb{R}^{M\times N}$ and $B \in \mathbb{R}^{P\times Q}$ is a block matrix $(A \otimes B) \in \mathbb{R}^{MP\times NQ}$ with block $[i, j] = A_{ij} \cdot B$. We will also require the "perfect shuffle" matrix S and the column-stacking operation of a matrix $\mathrm{vec}(\cdot)$ (Van Loan, 2000).

Properties For matrices of appropriate sizes (these will be valid for the derivations).

- $(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$
- $(A \otimes B)(C \otimes D) = (AC \otimes BD)$
- $S_{NQ}\,\mathrm{vec}(X) = \mathrm{vec}(X^\top)$ for $X \in \mathbb{R}^{Q\times N}$
- $(A \otimes B)\,\mathrm{vec}(X) = \mathrm{vec}(BXA^\top)$ for $A \in \mathbb{R}^{M\times N}$, $B \in \mathbb{R}^{P\times Q}$ and $X \in \mathbb{R}^{Q\times N}$.

The final property is particularly prevalent the in derivations so we introduce the shorthand

$$(A \otimes B)\,\mathrm{vec}(X) \to BXA^\top$$

to denote the "unvectorized" result. If the vectorization operation is applied to the result then the flattened correct result is obtained.

Notation The derivations contain several several matrices that we here list to give an overview. The input dimension is D and there are N observations.

- $X \in \mathbb{R}^{D\times N}$: All evaluation points stacked into a matrix.
- $\nabla K \nabla' \in \mathbb{R}^{DN\times DN}$: Kernel gram matrix for the derivatives with decompositions $\nabla K \nabla' = B + UCU^\top$.
- $G \in \mathbb{R}^{D\times N}$: All gradients stacked into a matrix. $\mathrm{vec}(G)$ r.h.s. of $\nabla K \nabla'\,\mathrm{vec}(Z) = \mathrm{vec}(G)$.
- $Z \in \mathbb{R}^{D\times N}$: the solution to $\nabla K \nabla'\,\mathrm{vec}(Z) = \mathrm{vec}(G)$, (Riesz representers).
- $B \in \mathbb{R}^{DN\times DN}$: Kronecker product of $K' \otimes \Lambda$
- $C \in \mathbb{R}^{N^2\times N^2}$: Symmetric matrix defined as $C = \mathrm{diag}(\mathrm{vec}(K''))S_{NN} = S_{NN}\,\mathrm{diag}(\mathrm{vec}(K''))$.
  - $C\,\mathrm{vec}(M) \to K'' \odot M^\top$.
  - $C^{-1}\,\mathrm{vec}(M) \to M^\top \oslash K''$.
  - $\odot$ and $\oslash$ correspond to the elementwise multiplication and division respectively.
- $U \in \mathbb{R}^{ND\times N^2}$: Tall and thin Kronecker product used in $\nabla K \nabla' = B + UCU^\top$.
  - For dot product kernels $U = (I \otimes \Lambda(X - \boldsymbol{c}))$.
  - For stationary kernels $U = (I \otimes \Lambda X)L$.
- $L \in \mathbb{R}^{N^2\times N^2}$: Sparse operator required for U in stationary kernels

  - $[L^\top \operatorname{vec}(M)]_{ab} \to M_{aa} - M_{ab}$
  - $[L \operatorname{vec}(M)]_{ab} \to \operatorname{diag}(\sum_a M_{ab}) - M_{ab}$
- $Q \in \mathbb{R}^{N \times N}$: Solution to $(C^{-1} + U^\top B^{-1} U) \operatorname{vec}(Q) = U^\top B^{-1} \operatorname{vec}(G)$

B. Kernel derivatives

Conditioning a GP on gradient observations requires the derivative of the kernel w.r.t. its arguments. Here we derive these terms for kernels with inner products and stationary kernels. We use the notation $\partial_b{}^j$ as shorthand for $\partial / \partial \boldsymbol{x}_b{}^j$ and use k' to refer to the derivative w.r.t. the scalar argument r. The notation mirrors that of Sec. 2.

B.1. General kernels

If we write a general kernel $k(\boldsymbol{x}_a, \boldsymbol{x}_b) = k(r(\boldsymbol{x}_a, \boldsymbol{x}_b))$ then the general form of each component for gradient inference will take the following form.

$$\begin{aligned} k(\boldsymbol{x}_a, \boldsymbol{x}_b) &= k(r(\boldsymbol{x}_a, \boldsymbol{x}_b)) \\ \partial_b{}^j k(r) &= k'(r) \partial_b{}^j r \\ \partial_a{}^i k(r) &= k'(r) \partial_a{}^i r \\ \partial_a{}^i \partial_b{}^j k(r) &= k'_{ab}(r) \cdot \partial_a{}^i \partial_b{}^j r + k''_{ab}(r) \cdot (\partial_a{}^i r)(\partial_b{}^j r) \end{aligned} \tag{18}$$

We thus use the convention of ordering the entries in the Gram matrix $\nabla K \nabla'$ first according to the N data points $\boldsymbol{x}_{1:N}$, and then according to dimension, i.e.,

$$\nabla K \nabla' = \begin{pmatrix} \nabla k(\boldsymbol{x}_1, \boldsymbol{x}_1) \nabla' & \dots & \nabla k(\boldsymbol{x}_1, \boldsymbol{x}_N) \nabla' \\ \vdots & \ddots & \vdots \\ \nabla k(\boldsymbol{x}_N, \boldsymbol{x}_1) \nabla' & \dots & \nabla k(\boldsymbol{x}_N, \boldsymbol{x}_N) \nabla' \end{pmatrix}, \tag{19}$$

where each block has the size $D \times D$. We highlight this ordering as it deviates from the conventional way found in the literature. Each element of the a, b^{th} block take the form $\partial_a{}^i \partial_b{}^j k(r)$ specified in Eq. (18), where no assumption on the structure of the kernel has been done at this point. The first term decomposes into a Kronecker product for the kernels we consider, because indices a, b and i, j separate. This term can thus be efficiently inverted. The second term is what usually makes closed-form gradient inference intractable which will be further explored below for dot product kernels and stationary kernels.

B.2. Dot Product Kernels

For dot product kernels we define the function r as

$$r(\boldsymbol{x}_a, \boldsymbol{x}_b) = (\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda (\boldsymbol{x}_b - \boldsymbol{c}). \tag{20}$$

See Sec. B.2.1 for examples of dot product kernels.

The relevant terms of Eq. (18) are:

$$\begin{aligned} \partial_a{}^i r(\boldsymbol{x}_a, \boldsymbol{x}_b) &= [\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^i \\ \partial_b{}^j r(\boldsymbol{x}_a, \boldsymbol{x}_b) &= [\Lambda(\boldsymbol{x}_a - \boldsymbol{c})]^j \\ \partial_a{}^i \partial_b^j r(\boldsymbol{x}_a, \boldsymbol{x}_b) &= \Lambda^{ij} \end{aligned}$$

From this we see the Gram matrix of Eq. (18) will look like:

$$\begin{aligned} \partial_a{}^i \partial_b{}^j k(r) &= k'_{ab}(r) \cdot \Lambda^{ij} + k''_{ab}(r) \cdot [\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^i [(\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda]^j \\ &= [K \otimes \Lambda]_{ab}^{ij} + [(I \otimes \Lambda \tilde{X}) \underbrace{(S_{NN} \operatorname{diag}(\operatorname{vec}(K'')))}_{C} (I \otimes \tilde{X} \Lambda)^\top]_{ab}^{ij} \end{aligned} \tag{21}$$

The first term is of Kronecker structure which is easy to invert using properties of Kronecker products. The second consists of rank-1 corrections block-wise multiplied with the scalar value k''_{ab}. The input indices are flipped for the term i.e., b appears as a row index and a as column. This shuffling is what makes the structure of the gradient Gram matrix difficult, but it can be resolved with the Kronecker transposed product. To derive the structure of the second term we start by defining the matrix $\tilde{X} \in \mathbb{R}^{D\times N}$, $\tilde{X} = X - \boldsymbol{c}$. We can then form the following outer product to get the structure:

$$
\begin{aligned}
[\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^i \left[(\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda^\top\right]^j &= [\Lambda\tilde{X}_b]^i[(\Lambda\tilde{X}_b)^\top]^j \\
&= \sum_{m,n}^{N} [\Lambda\tilde{X}_n]^i[\Lambda\tilde{X}_m]^j \delta_{am}\delta_{bn} \\
&= \sum_{n,n'}^{N}\sum_{m,m'}^{N} [\Lambda\tilde{X}_n]^i[\Lambda\tilde{X}_m]^j \delta_{am'}\delta_{bn'}\delta_{mm'}\delta_{nn'} \\
&= \sum_{n,n'}^{N}\sum_{m,m'}^{N} \left(\delta_{am'}\cdot[\Lambda\tilde{X}_n]^i\right) \underbrace{(\delta_{mm'}\delta_{nn'})}_{S_{NN}} \left(\delta_{bn'}\cdot[\Lambda\tilde{X}_m]^j\right) \\
&= \sum_{n,n'}^{N}\sum_{m,m'}^{N} [I\otimes\Lambda\tilde{X}]^i_{a,m'n}\,[S_{NN}]_{m'n,n'm}\,[I\otimes(\Lambda\tilde{X})^\top]^j_{n'm,b} \\
&= \left[(I\otimes\Lambda\tilde{X})S_{NN}(I\otimes\tilde{X}\Lambda)^\top\right]^{ij}_{ab}
\end{aligned}
$$

To get the right scalar value for each block outer product one has to write the term like below.

$$
\underbrace{(I\otimes\Lambda\tilde{X})}_{U}\underbrace{(S_{NN}\,\mathrm{diag}(\mathrm{vec}(K'')))}_{C}\underbrace{(I\otimes\Lambda\tilde{X})^\top}_{U^\top} \tag{22}
$$

with $C_{m'n,n'm} = K''_{mn}\delta_{mm'}\delta_{nn'}$ a symmetric $N^2\times N^2$ matrix.

B.2.1. Examples for inner product kernels

Kernel	$k(r)$	$k'(r)$	$k''(r)$
Polynomial(p)	$\frac{r^p}{p(p-1)}$	$\frac{r^{p-1}}{(p-1)}$	r^{p-2}
Polynomial(2)	$\frac{r^2}{2}$	r	1
Exponential/Taylor	$\exp(r)$	$\exp(r)$	$\exp(r)$

Table 1. Examples for inner product kernels where $r = (\boldsymbol{x}_a - \boldsymbol{c})^\top\Lambda(\boldsymbol{x}_b - \boldsymbol{c})$.

B.3. Stationary kernels

For a stationary kernel we define

$$
r(\boldsymbol{x}_a, \boldsymbol{x}_b) = (\boldsymbol{x}_a - \boldsymbol{x}_b)^\top\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b).
$$

Note here the discrepancy to conventional notation and do *not* think of r as a radius or Mahalonobis distance here (but rather its square). Then we have the following identities:

$$
\begin{aligned}
\partial_a{}^i r(\boldsymbol{x}_a, \boldsymbol{x}_b) &= 2\cdot[\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^i \\
\partial_b{}^j r(\boldsymbol{x}_a, \boldsymbol{x}_b) &= -2\cdot[\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^j \\
\partial_a{}^i\partial_b{}^j r(\boldsymbol{x}_a, \boldsymbol{x}_b) &= -4\cdot\Lambda^{ij}.
\end{aligned}
$$

The Gram matrix will have the general structure:

$$\partial_a{}^i \partial_b{}^j k(r) = -2k'_{ab}(r) \cdot \Lambda_{jl} - 4k''_{ab}(r) \cdot [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^i [(\boldsymbol{x}_a - \boldsymbol{x}_b)^\top \Lambda]^j. \tag{23}$$

Usually the factors 2 and 4 disappear due to scalar values of $k'(r)$ and $k''(r)$, see Sec. B.3.1.

Writing the second term in matrix form is a bit more intricate than Eq. (22), but taking the same approach we get

$$\begin{aligned}
&[\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^i [(\boldsymbol{x}_a - \boldsymbol{x}_b)^\top \Lambda]^j = [\Lambda \boldsymbol{x}_a]^i [\boldsymbol{x}_a^\top \Lambda]^j - [\Lambda \boldsymbol{x}_b]^i [\boldsymbol{x}_a^\top \Lambda]^j - [\Lambda \boldsymbol{x}_a]^i [\boldsymbol{x}_b^\top \Lambda]^j + [\Lambda \boldsymbol{x}_b]^i [\boldsymbol{x}_b^\top \Lambda]^j \\
&= \sum_{mn} \delta_{am} \delta_{bn} \left([\Lambda \boldsymbol{x}_m]^i [\boldsymbol{x}_m^\top \Lambda]^j - [\Lambda \boldsymbol{x}_n]^i [\boldsymbol{x}_m^\top \Lambda]^j - [\Lambda \boldsymbol{x}_m]^i [\boldsymbol{x}_n^\top \Lambda]^j + [\Lambda \boldsymbol{x}_n]^i [\boldsymbol{x}_n^\top \Lambda]^j \right) \\
&= \sum_{mn} \left(\delta_{am} \left([\Lambda \boldsymbol{x}_m]^i - [\boldsymbol{x}_n^\top \Lambda]^i \right) \right) \left(\delta_{bn} \left([\boldsymbol{x}_m^\top \Lambda]^j - [\boldsymbol{x}_n^\top \Lambda]^j \right) \right) \\
&= \sum_{mnpp'} \left(\delta_{am} \left(\delta_{pm} [\Lambda \boldsymbol{x}_p]^i - \delta_{pn} [\boldsymbol{x}_p^\top \Lambda]^i \right) \right) \left(\delta_{bn} \left(\delta_{p'm} [\boldsymbol{x}_{p'}^\top \Lambda]^j - \delta_{p'n} [\boldsymbol{x}_{p'}^\top \Lambda]^j \right) \right) \\
&= \sum_{mnoo'pp'} \left([\Lambda \boldsymbol{x}_p]^i \delta_{ao} \delta_{mo} (\delta_{pm} - \delta_{pn}) \right) \left([\boldsymbol{x}_n^\top \Lambda]^j \delta_{bo'} \delta_{no'} (\delta_{p'm} - \delta_{p'n}) \right) \\
&= \sum_{mn} \sum_{op} \underbrace{\delta_{ao} [\Lambda \boldsymbol{x}_p]^i}_{U_{ai,op}} \underbrace{\delta_{om} (\delta_{pm} - \delta_{pn})}_{L_{op,mn}} \sum_{o'p'} \underbrace{\delta_{o'n} (\delta_{p'm} - \delta_{p'n})}_{L_{mn,o'p'}} \underbrace{\delta_{o'b} [\boldsymbol{x}_n^\top \Lambda]^j}_{U_{o'p',bj}}
\end{aligned} \tag{24}$$

For dot product kernels we used $U = (I \otimes \Lambda(X - \boldsymbol{c}))$, for stationary kernels we instead use $U = (I \otimes \Lambda X)L$. The second term of the Gram matrix is formed by $UCU^\top$ in the same way as Eq. (22). U is however no longer a Kronecker product which makes the algorithmic details more involved. It is therefore more convenient to use the UL representation where L is a sparse linear operator. $U^\top \operatorname{vec}(g) = L^\top \operatorname{vec}(X^\top \Lambda g)_{mn} = \operatorname{vec}(X^\top \Lambda g_{mn} - X^\top \Lambda g_{mm})$

B.3.1. Examples for stationary kernels

Kernel	$k(r)$	$k'(r)$	$k''(r)$
Squared exponential	$\mathrm{e}^{-r/2}$	$-\frac{1}{2}k(r)$	$\frac{1}{4}k(r)$
Matérn $\nu = 1/2$	$\mathrm{e}^{-\sqrt{r}}$	$-\frac{k(r)}{2\sqrt{r}}$	$\frac{1}{4r^{3/2}}\left(\sqrt{r}+1\right)k(r)$
Matérn $\nu = 3/2$	$(1+\sqrt{3r})\,\mathrm{e}^{-\sqrt{3r}}$	$\frac{\sqrt{3}}{2\sqrt{r}}\left(\mathrm{e}^{-\sqrt{3r}} - k(r)\right)$	$\frac{\sqrt{3}}{2\sqrt{r}}\left(\frac{k(r)}{2r} - k'(r) - \mathrm{e}^{-\sqrt{3r}}\,\frac{1+\sqrt{3r}}{2r}\right)$
Matérn $\nu = 5/2$	$\left(1+\sqrt{5r}+\frac{5r}{3}\right)\mathrm{e}^{-\sqrt{5r}}$	$\left(\frac{\sqrt{5}}{2\sqrt{r}}+\frac{5}{3}\right)\mathrm{e}^{-\sqrt{5r}} - \frac{\sqrt{5}}{2\sqrt{r}}k(r)$	$\frac{\sqrt{5}}{2\sqrt{r}}\left(\frac{k(r)}{2r} - k'(r) - \mathrm{e}^{-\sqrt{5r}}\left(\frac{1+\sqrt{5r}}{2r}+\frac{5}{3}\right)\right)$
Rational quadratic	$\left(1+\frac{r}{2\alpha}\right)^{-\alpha}$	$-\frac{1}{2}\left(1+\frac{r}{2\alpha}\right)^{-\alpha-1}$	$\frac{\alpha+1}{4\alpha}\left(1+\frac{r}{2\alpha}\right)^{-\alpha-2}$

Table 2. Examples for stationary kernels where $r = (\boldsymbol{x}_a - \boldsymbol{x}_b)^\top \Lambda (\boldsymbol{x}_a - \boldsymbol{x}_b)$.

Table 2 contains the kernels we considered. For reasons of space, we derive the general expressions for the Matérn family with half integer smoothness parameter $\nu = p + \frac{1}{2}$ for $p \in \mathbb{N}$ here, which reads

$$k_{p+1/2}(r) = \exp\left(-\sqrt{2\nu r}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left(\sqrt{8\nu r}\right)^{p-i},$$

and has the monstrous derivatives

$$\begin{aligned}
k'_{p+1/2}(r) &= -\sqrt{\frac{\nu}{2r}} k_{p+1/2} + \exp\left(-\sqrt{2\nu r}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p-1} \frac{(p+i)!}{i!(p-i-1)!} \left(\sqrt{8\nu r}\right)^{p-i-1} \sqrt{\frac{2\nu}{r}} \\
k''_{p+1/2}(r) &= \left(\sqrt{\frac{\nu}{8r^3}} + \frac{\nu}{2r}\right) k_{p+1/2} - \left(\sqrt{\frac{\nu}{2r^3}} + \frac{2\nu}{r}\right) \exp\left(-\sqrt{2\nu r}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p-1} \frac{(p+i)!}{i!(p-i-1)!} \left(\sqrt{8\nu r}\right)^{p-i-1} \\
&\quad + \sqrt{\frac{2\nu}{r}} \exp\left(-\sqrt{2\nu r}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p-2} \frac{(p+i)!}{i!(p-i-2)!} \left(\sqrt{8\nu r}\right)^{p-i-2}.
\end{aligned}$$

The general form of the Matérn kernels also falls into the category of stationary kernels, as do the spectral mixture kernels (Wilson & Adams).

C. Decomposition Benefits

In Appendix B we showed that $\nabla K \nabla'$ can be written as $B + UCU^\top$, see Appendix A for summary. In Sec. 2.3 we discussed some benefits of the decomposition that we here explain more in detail.

C.1. Woodbury vector for $N < D$

The decomposition is particularly interesting when the number of observations N is small. In this setting we can employ the matrix inversion lemma, Eq. (6) restated here for convenience

$$(B + UCU^\top)^{-1} = B^{-1} - B^{-1}U\left(C^{-1} + U^\top B^{-1} U\right)^{-1} U^\top B^{-1}.$$

If the size of C is smaller than B and B^{-1} is "cheap", then the r.h.s. above is computationally beneficial. The involved matrices are all comparatively large, but by using the important properties of Kronecker products (Appendix A) it is possible to significantly lower the requirements. Here we outline the required operations for a dot product kernel with $\tilde{X} = X - \boldsymbol{c}$. The operations for stationary kernels are similar but require the additional application of L for each operation involving U.

1. $T = U^\top B^{-1} \operatorname{vec}(G) \to \tilde{X}^\top G (K')^{-1}$.
   - $T \in \mathbb{R}^{N \times N}$
2. Solve: $\left(C^{-1} + U^\top B^{-1} U\right) \operatorname{vec}(Q) = \operatorname{vec}(T)$: $\left(C^{-1} + (K')^{-1} \otimes (\tilde{X}^\top \Lambda \tilde{X})\right) \operatorname{vec}(Q) = \operatorname{vec}(T)$.
   - $Q \in \mathbb{R}^{N \times N}$
3. $\operatorname{vec}(Z) = B^{-1} \operatorname{vec}(G) - B^{-1} U \operatorname{vec}(Q)$: $Z = \Lambda^{-1} G (K')^{-1} - XQ(K')^{-1}$.
   - $Z \in \mathbb{R}^{D \times N}$

Special case Step 2 in the above procedure is the source of the $\mathcal{O}((N^2)^3)$ scaling in computations. For the situation outlined in Sec. 4.2 it is possible to solve the linear system analytically. A multiplication with the linear system in step 2 for the second order polynomial kernel is performed as

$$(C^{-1} + (\tilde{X}\Lambda\tilde{X})^{-1} \otimes (\tilde{X}^\top \Lambda \tilde{X}))\operatorname{vec}(V) \to V^\top + (\tilde{X}^\top \Lambda \tilde{X}) V (\tilde{X}\Lambda\tilde{X})^{-1}. \tag{25}$$

For the outlined situation in Sec. 4.2 the r.h.s. $T = (\tilde{X}^\top A \tilde{X})(\tilde{X}^\top \Lambda \tilde{X})^{-1}$.

The solution to the linear system is

$$Q = \frac{1}{2}(\tilde{X}^\top \Lambda \tilde{X})^{-1}(\tilde{X}^\top A \tilde{X}).$$

This is easily verified by inserting the value for Q in Eq. (25)

$$\begin{aligned}
Q^\top + (\tilde{X}^\top \Lambda \tilde{X}) Q (\tilde{X}^\top \Lambda \tilde{X})^{-1} &= \frac{1}{2}(\tilde{X}^\top A \tilde{X})(\tilde{X}^\top \Lambda \tilde{X})^{-1} + (\tilde{X}^\top \Lambda \tilde{X})[\frac{1}{2}(\tilde{X}^\top \Lambda \tilde{X})^{-1}(\tilde{X}^\top A \tilde{X})](\tilde{X}^\top \Lambda \tilde{X})^{-1} \\
&= \frac{1}{2}(\tilde{X}^\top A \tilde{X})(\tilde{X}^\top \Lambda \tilde{X})^{-1} + \frac{1}{2}(\tilde{X}^\top A \tilde{X})(\tilde{X}^\top \Lambda \tilde{X})^{-1} \\
&= (\tilde{X}^\top A \tilde{X})(\tilde{X}^\top \Lambda \tilde{X})^{-1} = T
\end{aligned}$$

C.2. Benefits for general N

The derived Kronecker structure of the Gram matrix $\nabla K \nabla'$ in Eq. (2) highlights an important speedup of multiplication. Multiplying a vectorized matrix V of same shape as G with the Gram matrix is obtained by the following computations

$$\nabla K \nabla' \operatorname{vec}(V) = \Lambda V K' + \Lambda X (K'' \odot V^\top \Lambda X),$$

A full algorithm for multiplication with the Gram matrix is available in Alg. 2, with modification for stationary kernels written in red. The advantage of defining such a routine is that the Gram matrix never needs to be built, which reduces the memory requirement from $\mathcal{O}((DN)^2)$ to $\mathcal{O}(DN + N^2)$.

Algorithm 2 $\nabla K \nabla'$-MVM

Require: x_0
Input: $(V \in \mathbb{R}^{D\times N}, K' \in \mathbb{R}^{N\times N}, K'' \in \mathbb{R}^{N\times N}, \tilde{X} \in \mathbb{R}^{D\times N})$
$M = \tilde{X}^\top \Lambda V$
$\boldsymbol{m} = \mathrm{diag}(M)$ {Multiplication with $L^\top$}
$M = M - \boldsymbol{m}^\top$
$M = K'' \odot M^\top$
$\boldsymbol{m} = \sum_a M_{ab}$ {Multiplication with L}
$M = \boldsymbol{m}^\top - M$
Return: $\Lambda V K' + \Lambda \tilde{X} M$

D. Gradient and Hessian inference

Once $Z \in \mathbb{R}^{D\times N}$ has been obtained from solving $\nabla K \nabla' \,\mathrm{vec}(Z) = \mathrm{vec}(G)$ it is possible to infer the gradient and Hessian at a new point $\boldsymbol{x}_a$. Note that a is now an index with a single value and b takes N values, so $K_{ab} = \boldsymbol{k}_{ab}$ is a row vector. Inferring the gradient and Hessian at a point $\boldsymbol{x}_a$ requires the following contractions

$$\bar{\boldsymbol{g}}(\boldsymbol{x}_a)^i = \sum_{bl} [\partial_a^i \partial_b^l k(r)]_{ab}^{il} Z_b^l, \tag{26}$$

and

$$\bar{H}(\boldsymbol{x}_a)^{ij} = \sum_{bl} [\partial_a^i \partial_a^j \partial_b^l k(r)]_{aab}^{ijl} Z_b^j. \tag{27}$$

D.1. Dot product kernels

Gradient For dot product kernels the gradient at a point $\boldsymbol{x}_a$ is readily available from Eq. (26) and Eq. (20) as

$$\boldsymbol{g}(\boldsymbol{x}_a) = \Lambda Z (\boldsymbol{k}'_{ab})^\top + \Lambda(X - \boldsymbol{c})((\boldsymbol{k}''_{ab})^\top \odot Z^\top(\boldsymbol{x}_a - \boldsymbol{c})).$$

A prior mean for the gradient was omitted.

Hessian The posterior mean of the Hessian in Eq. (27) first requires the third derivative of the kernel. Differentiating Eq. (20) again yields

$$\begin{aligned}\partial_a{}^i \partial_a{}^i \partial_b{}^l k(r) = {} & k''_{ab} \cdot \Lambda^{jl} \cdot [\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^i + k''_{ab} \cdot \Lambda^{il} \cdot [\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^j + \delta_{ab} k''_{ab} \cdot \Lambda^{ij} \cdot [\Lambda(\boldsymbol{x}_a - \boldsymbol{c})]^l \\ & + k'''_{ab} [\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^j [\Lambda(\boldsymbol{x}_a - \boldsymbol{c})]^l [\Lambda(\boldsymbol{x}_b - \boldsymbol{c})]^i\end{aligned}$$

To perform the contraction in Eq. (27) we first introduce $\tilde{X} = X - \boldsymbol{c}$ and perform the contraction over l which results in

$$\begin{aligned}\bar{H}(\boldsymbol{x}_a)^{ij} = {} & \sum_b k''_{ab} \cdot [\Lambda Z]_b^j \cdot [\Lambda \tilde{X}]_b^i + k''_{ab} \cdot [\Lambda Z]_b^i \cdot [\Lambda \tilde{X}]_b^j + \delta_{ab} \cdot \Lambda^{ij} \cdot k''_{ab} \cdot [\Lambda(\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda Z]_{ab} \\ & + k'''_{ab} \cdot [\Lambda \tilde{X}]_b^i \cdot [\Lambda \tilde{X}]_b^j \cdot [(\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda Z]_{ab}.\end{aligned}$$

The final contraction of b can easily be interpreted as standard matrix multiplication to arrive at the form

$$\bar{H}(\boldsymbol{x}_a) = \begin{bmatrix}\Lambda \tilde{X}, \Lambda Z\end{bmatrix} \begin{bmatrix} M & \hat{M} \\ \hat{M} & 0 \end{bmatrix} \begin{bmatrix} \tilde{X}^\top \Lambda \\ Z^\top \Lambda \end{bmatrix} + \Lambda \cdot \mathrm{Tr}(\breve{M}).$$

All these M-matrices are diagonal matrices with N elements

$$\begin{aligned} M_{bb} &= \boldsymbol{k}'''_{ab} \odot [(\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda Z]_{ab}, \\ \hat{M}_{bb} &= \boldsymbol{k}''_{ab} \\ \breve{M}_{bb} &= \delta_{ab} \cdot \boldsymbol{k}''_{ab} (\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda Z. \end{aligned}$$

The last expression including $\mathrm{Tr}(\breve{M})$ can be simplified to $k''_{aa}(\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda Z$ **if** $\boldsymbol{x}_a \in X$.

D.2. Stationary kernels

Gradient inference for stationary kernels looks similar to the dot product kernels but has some important differences. For the following derivations we introduce $\tilde{k}' = 2k'$, $\tilde{k}'' = 4k''$, $\tilde{k}''' = 8k''$ and $\tilde{X} = (\boldsymbol{x}_a - X)$. The posterior mean gradient at a point $\boldsymbol{x}_a$ for a stationary kernel is

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{x}_a) &= -\Lambda Z \tilde{\boldsymbol{k}}'_{ba} - \Lambda \tilde{X}(\tilde{\boldsymbol{k}}''_{ba} \odot \boldsymbol{m}_b), \\ \boldsymbol{m}_b &= (\sum_l Z_b^l \odot [\Lambda \tilde{X}]_b^l) \end{aligned} \tag{28}$$

Hessian The third derivative of stationary kernels required for the Hessian inference is

$$\begin{aligned} \partial_a{}^i \partial_a{}^i \partial_b{}^l k(r) = &-\tilde{k}''_{ab} \cdot \Lambda^{jl} \cdot [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^i - \tilde{k}''_{ab} \cdot \Lambda^{il} \cdot [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^j + \tilde{k}''_{ab} \cdot \Lambda^{ij} \cdot [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^l \\ &- \tilde{k}'''_{ab} [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^j [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^l [\Lambda(\boldsymbol{x}_a - \boldsymbol{x}_b)]^i, \end{aligned}$$

with $\boldsymbol{m}_b$ the same vector as in Eq. (28). The posterior mean is obtained in the same way as for the dot product, by Eq. (27)

$$\begin{aligned} \bar{H}(\boldsymbol{x}_a)^{ij} = &\sum_b -\tilde{k}''_{ab} \cdot [\Lambda Z]_b^j \cdot [\Lambda \tilde{X}]_b^i - \tilde{k}''_{ab} \cdot [\Lambda Z]_b^i \cdot [\Lambda \tilde{X}]_b^j + \Lambda^{ij} \cdot \tilde{k}''_{ab} \odot \boldsymbol{m}_b \\ &- (\tilde{k}'''_{ab} \odot \boldsymbol{m}_b) \cdot [\Lambda \tilde{X}]_b^i \cdot [\Lambda \tilde{X}]_b^j. \end{aligned}$$

The posterior mean can be written in standard matrix notation as

$$\bar{H}(\boldsymbol{x}_a) = \begin{bmatrix} \Lambda \tilde{X}, \Lambda Z \end{bmatrix} \begin{bmatrix} M & \hat{M} \\ \hat{M} & 0 \end{bmatrix} \begin{bmatrix} \tilde{X}^\top \Lambda \\ Z^\top \Lambda \end{bmatrix} + \Lambda \cdot \mathrm{Tr}(\breve{M}).$$

The diagonal matrices are this time given by

$$\begin{aligned} M_{bb} &= \tilde{\boldsymbol{k}}'''_{ab} \odot \boldsymbol{m}_b, \\ \hat{M} &= -\tilde{\boldsymbol{k}}''_{ab}, \\ \breve{M}_{bb} &= \tilde{\boldsymbol{k}}''_{ab} \odot \boldsymbol{m}_b. \end{aligned}$$

E. Further details about applications

E.1. Infering the optimizer

A GP with gradient observations learns a mapping $\boldsymbol{x} \to \nabla f(\boldsymbol{x})$. With efficient gradient inference we can also flip the inference and learn a mapping $\nabla f(\boldsymbol{x}) \to \boldsymbol{x}(\nabla f)$ and query what $\boldsymbol{x}(\nabla f(\boldsymbol{x}) = 0)$ for a new update. This is achieved by performing gradient inference but interchanging the input and output. The posterior mean for which $\boldsymbol{x}\ \nabla f(\boldsymbol{x}) = 0$ occurs is

$$\bar{\boldsymbol{x}}_* = \boldsymbol{x}_m + [\nabla K \nabla'(0, G)] \left[\nabla K \nabla'(G, G)\right]^{-1} \mathrm{vec}(X - \boldsymbol{x}_m).$$

E.2. Stationary linear solvers

For the special case of stationary linear solvers in linear algebra we have $f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_*)^\top A(\boldsymbol{x} - \boldsymbol{x}_*)$ and $\nabla f(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) = A(\boldsymbol{x} - \boldsymbol{x}_*)$ and we are interested in inferring $\boldsymbol{x}_*$.

For the polynomial(2) kernel if we use $\boldsymbol{c} = \boldsymbol{g}_m$ and prior mean $\boldsymbol{\mu} = \boldsymbol{x}_m$ inference is fast. First define $\tilde{X} = X - \boldsymbol{x}_m$ and $\tilde{G} = \boldsymbol{g} - \boldsymbol{g}_m$. Because $\tilde{G}^\top \tilde{X} = \tilde{X}^\top \tilde{G}$ we get the Z that solves $\nabla K \nabla' \mathrm{vec}(Z) = \mathrm{vec}(\tilde{X})$:

$$Z = \Lambda^{-1} \tilde{X} (\tilde{G}^\top \Lambda \tilde{G})^{-1} - \frac{1}{2} \tilde{G} (\tilde{G}^\top \Lambda \tilde{G})^{-1} \tilde{G}^\top \tilde{X} (\tilde{G}^\top \Lambda \tilde{G})^{-1} \tag{29}$$

Inferring at which the point $\hat{\boldsymbol{x}}_a$ a gradient $\boldsymbol{g}_a$ occurs is done by the following computation:

$$\begin{aligned}
\hat{\boldsymbol{x}}_a &= \boldsymbol{x}_m + \Lambda Z(\tilde{G}^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m)) + \Lambda \tilde{X}[Z^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m))] \\
&= \boldsymbol{x}_m + \tilde{X}(\tilde{G}^\top \Lambda \tilde{G})^{-1}(\tilde{G}^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m)) - \frac{1}{2}\Lambda \tilde{G}(\tilde{G}^\top \Lambda \tilde{G})^{-1}\tilde{G}^\top \tilde{X}(\tilde{G}^\top \Lambda \tilde{G})^{-1}(\tilde{G}^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m)) \\
&+ \Lambda \tilde{G}[(\tilde{G}^\top \Lambda \tilde{G})^{-1}\tilde{X}^\top(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m) - \frac{1}{2}(\tilde{G}^\top \Lambda \tilde{G})^{-1}\tilde{G}^\top \tilde{X}(\tilde{G}^\top \Lambda \tilde{G})^{-1}\tilde{G}^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m)] \\
&= \boldsymbol{x}_m + \tilde{X}(\tilde{G}^\top \Lambda \tilde{G})^{-1}\tilde{G}^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m) \\
&+ \Lambda \tilde{G}[(\tilde{G}^\top \Lambda \tilde{G})^{-1}\left(\tilde{X}^\top(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m) - \tilde{G}^\top \tilde{X}(\tilde{G}^\top \Lambda \tilde{G})^{-1}\tilde{G}^\top \Lambda(\tilde{\boldsymbol{g}}_a - \tilde{\boldsymbol{g}}_m)\right)]
\end{aligned}$$

F. Details about experiments

F.1. linear algebra

For the linear algebra task we generated the matrix A Eq. (14) in a manner beneficial for CG. The eigenvalues of A were generated according to

$$\lambda_i = \lambda_{min} + \frac{\lambda_{max} - \lambda_{min}}{N-1} \cdot \rho^{N-i} \cdot (N-i),$$

with $\lambda_{min} = 0.5$, $\lambda_{max} = 100$ yielding a condition number of $\kappa(A) = 200$ and $\rho = 0.6$ so approximately the 15 largest eigenvalues are larger than 1. In this setting CG is expected to converge in slightly more than 15 iterations. A relative tolerance in gradient norm of 10^{-5} was used as termination criterion due to numerical instabilities. The starting and solution points were sampled according to $\boldsymbol{x}_0 \sim \mathcal{N}(0, 5^2 \cdot I)$ and $\boldsymbol{x}_* \sim \mathcal{N}(-2 \cdot \mathbf{1}, I)$. The Hessian-based optimization used a fixed $\boldsymbol{c} = \mathbf{0}$ and $\boldsymbol{g}_c = A(\boldsymbol{c} - \boldsymbol{x}_*) = -A\boldsymbol{x}_* = -\boldsymbol{b}$ in the linear system interpretation $A\boldsymbol{x} = \boldsymbol{b}$. There a plenty of possibilities for how the algorithm can be implemented and this particular version was sensitive to the relative position of $\boldsymbol{c}$ and $\boldsymbol{x}_*$.

F.2. Nonlinear Optimization

We chose the test function (restated here for convenience)

$$f(\boldsymbol{x}) = \sum_{i=1}^{D-1} x_i^2 + 2 \cdot (x_{i+1} - x_i^2)^2$$

for the more challenging nonlinear experiments. It is a relaxed version of the famous Rosenbrock function, which was used to better control the magnitude of the gradients for the high-dimensional problem. This was important because the RBF kernels used for the optimization used a fixed Λ, which could lead to numerical issues if the magnitude of the steps and gradients drastically changed between iterations. The lengthscale of the isotropic kernels in the algorithms were $\Lambda = 9 \cdot I$ for GP-H and $\Lambda = 0.05 \cdot I$. There are too many options of extending the algorithm to go over in this manuscript, which is why the algorithm should be seen more as a proof-of-concept than radical new algorithm.

F.3. Hamiltonian Monte Carlo

We used the following unnormalized density as a target for the HMC experiment

$$f(x) = \exp\left(-\frac{1}{2}\left(x_1^2 + (a_0 x_1^2 + a_1 x_2 + a_2)^2 + \sum_{i=3}^{D} a_i x_i^2\right)\right) \tag{30}$$

and set the parameter vector to $\boldsymbol{a} = [2, -2, 2, \ldots, 2]^\top$. The distribution is thus Gaussian with variance $\frac{1}{2}$ in all components other than x_1 and x_2. Since we use an isotropic RBF kernel to model the potential energy (i.e., the negative logarithm of the above function), we randomly rotate the above function by applying sampled orthonormal matrices to the input vector.

Fig. 4 uses Eq. (30) directly, and thus the kernel is aligned with the problem. We choose a (squared) lengthscale of $0.4D$ where $D = 100$ from visual inspection of the typical scale of the "banana". HMC uses a step-size $\epsilon = 4 \cdot 10^{-3}/\lceil\sqrt[4]{D}\rceil$ and number of leapfrog steps $T = 32 \cdot \lceil\sqrt[4]{D}\rceil$, with the term $\lceil\sqrt[4]{D}\rceil$ being motivated by the analysis of how these parameters should change with increasing dimension (Neal et al., 2011). For all experiments we draw a standard normal vector as a

starting point and simulate D times with plain HMC for burn-in, before retaining samples in the case of HMC, or starting the training procedure for GPG-HMC. The training is performed as described in Sec. 5.3.

The rotated version of the above function used slightly different parameters for the RBF kernel, a squared lengthscale of $0.25D$ to stay on the conservative side about the target function. Also we halved the stepsize of the leapfrog integrator while leaving the number of steps taken unchanged. Otherwise, the acceptance rate also dropped significantly for both methods. All experiments used a mass parameter of $m = 1$.

Algorithm 3 summarizes the GPG-HMC method without the training procedure which leaves a lot of space for engineering. In fact, this is identical to standard HMC, except for the fact that instead of the true gradient ∇E the GP surrogate $\widehat{\nabla E}$ is used.

Algorithm 3 GPG-HMC

input $\boldsymbol{x}_0$, $E(\cdot)$, $\widehat{\nabla E}(\cdot)$, N, T, ϵ, m
output $\boldsymbol{X}$
  $\boldsymbol{x} = \boldsymbol{x}_0$; $\boldsymbol{X} = [\,]$
  for n = 1:N **do**
    $\boldsymbol{p} \sim \mathcal{N}(0, mI)$
    $H \leftarrow U(\boldsymbol{x}) + \frac{\boldsymbol{p}^\top \boldsymbol{p}}{2m}$
    $\boldsymbol{x}_{\text{new}}, \boldsymbol{p} \leftarrow \text{LEAPFROG}(\boldsymbol{x}, \boldsymbol{p}, \widehat{\nabla E}(\cdot), T, \epsilon)$
    $\Delta H \leftarrow E(\boldsymbol{x}) + \frac{\boldsymbol{p}^\top \boldsymbol{p}}{2m} - H$
    if $r \sim \text{UNIFORM}[0, 1] < \min(1, \mathrm{e}^{-\Delta H})$ **then**
      $\boldsymbol{x} \leftarrow \boldsymbol{x}_{\text{new}}$
    end if
    $\boldsymbol{X} \leftarrow [\boldsymbol{X}, \boldsymbol{x}]$
  end for