Journal of Machine Learning Research 21 (2020) 1-42 Submitted 7/18; Revised 10/19; Published 2/20

Conjugate Gradients for Kernel Machines

Simon Bartels SBARTELS@TUE.MPG.DE
Philipp Hennig PH@TUE.MPG.DE
Max Planck Institute for Intelligent Systems and University of Tübingen
Maria-von-Linden-Str. 6, Tübingen, GERMANY

Editor: Mohammad Emtiyaz Khan

Abstract

Regularized least-squares (kernel-ridge / Gaussian process) regression is a fundamental algorithm of statistics and machine learning. Because generic algorithms for the exact solution have cubic complexity in the number of datapoints, large datasets require to resort to approximations. In this work, the computation of the least-squares prediction is itself treated as a probabilistic inference problem. We propose a structured Gaussian regression model on the kernel function that uses projections of the kernel matrix to obtain a low-rank approximation of the kernel and the matrix. A central result is an enhanced way to use the method of conjugate gradients for the specific setting of least-squares regression as encountered in machine learning.

Keywords: Gaussian processes, kernel methods, low-rank approximation, conjugate gradients, probabilistic numerics

1. Introduction

Regularized least-squares is one of the fundamental algorithms in statistics and machine learning. Due to its importance it is known under a variety of names such as kernel ridge regression (Hoerl and Kennard, 1970), spline regression (*e.g.* Wahba (1990)), Kriging (*e.g.* Matheron (1973)) and Gaussian process (GP) regression (*e.g.* Rasmussen and Williams (2006)). The common principle is the estimation of a regression function from a reproducing kernel Hilbert space (RKHS) $f : \mathbb{X} \to \mathbb{R}$ over some domain $\mathbb{X}$ that minimizes the regularized loss (Rasmussen and Williams, 2006, Eq. (6.19))

$$\mathcal{L}(f) = \frac{1}{2}\|f\|_k^2 + \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2,$$

where $(\boldsymbol{x}_i, y_i) \in \mathbb{X} \times \mathbb{R}$, $i = 1, \ldots, N$ are observations, $\sigma^2 \in \mathbb{R}^+$ is a regularization parameter, k is the corresponding kernel and $\|\cdot\|_k$ is the RKHS norm of f.

The minimizer of this loss has a closed-form solution that coincides with the posterior mean of the Gaussian process $p(f \,|\, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{GP}(f; \bar{f}, \bar{c})$ under a zero-mean prior $p(f) = \mathcal{GP}(f; 0, k)$ and likelihood $p(\boldsymbol{y} \,|\, \boldsymbol{f}(\boldsymbol{X})) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}(\boldsymbol{X}), \sigma^2\boldsymbol{I})$ (Kimeldorf and Wahba, 1970; Wahba, 1990; Rasmussen and Williams, 2006):

$$\bar{f}(\boldsymbol{x}_*) = \boldsymbol{k}_*^\intercal(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}, \tag{1}$$

$$\bar{c}(\boldsymbol{x}_*, \boldsymbol{x}_{**}) = k(\boldsymbol{x}_*, \boldsymbol{x}_{**}) - \boldsymbol{k}_*^\intercal(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_{**} \tag{2}$$

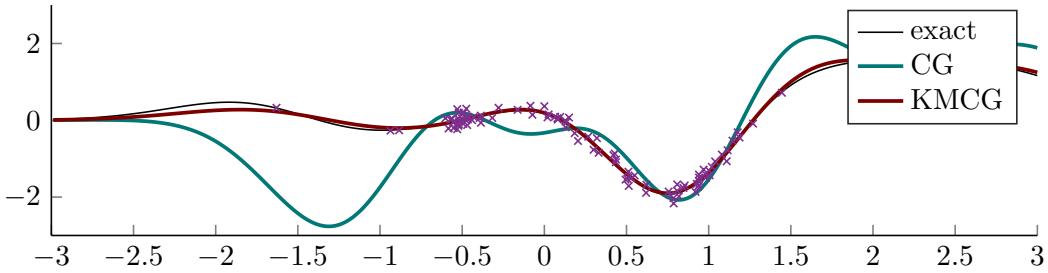


Figure 1: Our algorithm KMCG in comparison to CG on a toy setup. The dataset consists of one hundred data-points where the targets are a draw from a zero-mean Gaussian process with squared exponential kernel (Eq. (18) with $\mathbf{\Lambda} = 0.25$ and $\theta_f = 2$). The thin, black line is the posterior mean of that Gaussian process (Eq. 1). The light-green line is the mean prediction produced by conjugate gradients after $P = 7$ steps and the dark-red line is the mean prediction of KMCG (where the number of inducing inputs $M = N$).

where $\boldsymbol{K}_{ij} = k(x_i, x_j)$, and $\boldsymbol{k}_{*,i} = k(\boldsymbol{x}_*, \boldsymbol{x}_i)$.

For datasets up to about $N \sim 5 \cdot 10^4$ observations, the standard approach to solve Equations (1) and (2) is to compute a Cholesky decomposition (Benoit, 1924) of $\boldsymbol{K} + \sigma^2 \boldsymbol{I}$ at a cubic cost $\mathcal{O}(N^3)$. For larger datasets, a number of approximate algorithms have been proposed that yield an approximation $\hat{f}$ to $\bar{f}$ in linear time (Zhu et al., 1998; Csató and Opper, 2002; Snelson and Ghahramani, 2007; Walder et al., 2008; Rahimi and Recht, 2009; Titsias, 2009b; Lázaro-Gredilla et al., 2010; Yan and Qi, 2010; Le et al., 2013; Solin and Särkkä, 2014; Wilson and Nickisch, 2015; Hensman et al., 2018). Comparative empirical studies like those of Chalupka et al. (2013) or Quiñonero-Candela and Rasmussen (2005) indicate that some of these methods can provide good approximations in a reasonable amount of time, although there is no conclusive 'best practice' among these choices.

Not included in the list above are iterative linear solvers, such as the method of conjugate gradients (CG) (Hestenes and Stiefel, 1952). These algorithms construct an approximate solution to systems of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ using repeated matrix-vector multiplications (MVMs). In general, each MVM with $\boldsymbol{K}$ has quadratic costs $\mathcal{O}(N^2)$ which is one reason why the machine learning community prefers the methods above.

Furthermore, a linear solver needs to run again for new test inputs when computing the posterior uncertainty (Eq. 2) and Gaussian process regression often requires the evaluation of the log marginal likelihood:

$$\ln p(\boldsymbol{y}) = -\frac{1}{2}\boldsymbol{y}^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} + \frac{1}{2} \ln |2\pi (\boldsymbol{K} + \sigma^2 \boldsymbol{I})|^{-1}. \tag{3}$$

Conjugate gradients can be used to estimate $|\boldsymbol{K}|$ (Filippone and Engler, 2015), yet also requiring several runs.

Below, we present a way of using CG specifically tailored to Equations (1) to (3) which we dub *kernel machine conjugate gradients* (KMCG). Our approach follows the notion of probabilistic numerics (PN) (Hennig et al., 2015) which phrases approximation as inference. A common idea of PN formulations is to replace a deterministic yet intractable operation by Bayesian inference where, by design, prior and likelihood admit analytic estimation of the intractable solution. In our case, the 'intractable' operation is the inversion of very large

matrices (*i.e.* of size $N \times N$ such that N^3 is intractable), and the design criterion for the prior is that the posterior mean over the matrix has to admit efficient inversion, which we achieve through the matrix inversion lemma. Instead of providing an approximation solely to the vector $(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y}$, our approach uses the MVMs performed by CG to learn an approximation directly to the function k.

The following section proposes a model-template that can be used to learn low-rank approximations to kernel functions. The subsequent section shows how conjugate gradients can be applied into that template. A discussion on how our approach relates to existing work is presented thereafter, in Section 3.4.

2. Model

To approximate Equations (1) to (3), we will approximate the kernel and, to this end, present a probabilistic estimation rule for k. The idea is to treat the kernel as unknown and to choose prior and likelihood such that the posterior mean k_M is efficient to evaluate and yields a kernel of finite rank. Substituting for this finite-rank kernel in Equations (1) to (3) then allows to compute these expressions faster. The following sections describe finite-rank kernel, our prior, possible likelihoods and resulting posteriors. Fig. 2 shows a schematic summary of this section.

2.1. Finite-rank Kernel

An M-rank approximation to a kernel is a factorization of the form

$$k(\boldsymbol{x}, \boldsymbol{z}) \approx \boldsymbol{\phi}(\boldsymbol{x})^* \boldsymbol{\Sigma}^{-1} \boldsymbol{\phi}(\boldsymbol{z})$$

where $\boldsymbol{\phi}(\boldsymbol{x}) : \mathbb{X} \to \mathbb{C}^M$, $\boldsymbol{\phi}^*$ denotes the conjugate transpose, and $\boldsymbol{\Sigma}$ is an $M \times M$ Hermitian and positive definite matrix.

Given such an expansion one can use the matrix-inversion, and matrix-determinant lemmata to approximate Equations (1) to (3) with the expressions below

$$\overline{f}(\boldsymbol{x}_*) \approx \boldsymbol{\phi}(\boldsymbol{x}_*)^* \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^* + \sigma^2 \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\Phi}\boldsymbol{y} \tag{4}$$

$$\overline{c}(\boldsymbol{x}_*, \boldsymbol{z}_*) \approx \sigma^2 \boldsymbol{\phi}(\boldsymbol{x}_*)^* \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^* + \sigma^2 \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\phi}(\boldsymbol{z}_*) \tag{5}$$

$$\ln p(\boldsymbol{y}) \approx -\frac{1}{2}\boldsymbol{y}^\intercal \boldsymbol{\Phi}^* \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^* + \sigma^2 \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\Phi}\boldsymbol{y} - \frac{1}{2} \ln \left| \left(\frac{1}{\sigma^2} \boldsymbol{\Phi}\boldsymbol{\Phi}^* + \boldsymbol{\Sigma} \right) \right| - \frac{N}{2} \ln(2\pi\sigma^2) \tag{6}$$

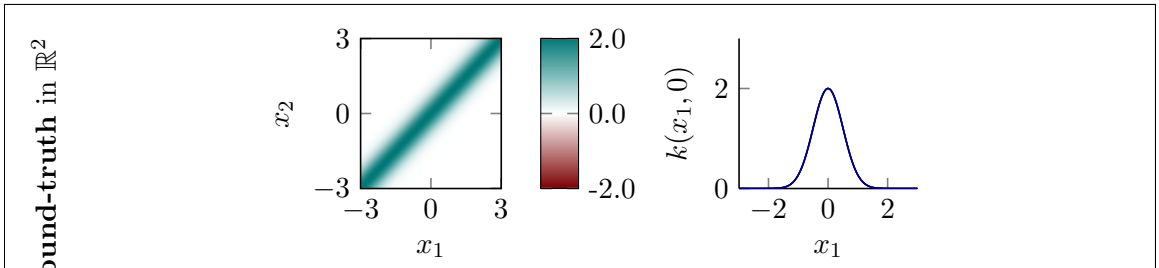
where $\boldsymbol{\phi}(\boldsymbol{x}_*)_j = \phi_j(\boldsymbol{x}_*)$ and $\boldsymbol{\Phi}_{ij} = \phi_i(\boldsymbol{X}_j)$. Typically $M \ll N$ and therefore the computational costs to evaluate Equations (4) to (6) reduce from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$, *i.e.* linear in N. The dominant factor is the matrix-matrix product $\boldsymbol{\Phi}\boldsymbol{\Phi}^*$.

An example for a finite-rank kernel that will become important later, is the *Subset of Regressors* (SoR) approximation (Quiñonero-Candela and Rasmussen, 2005)

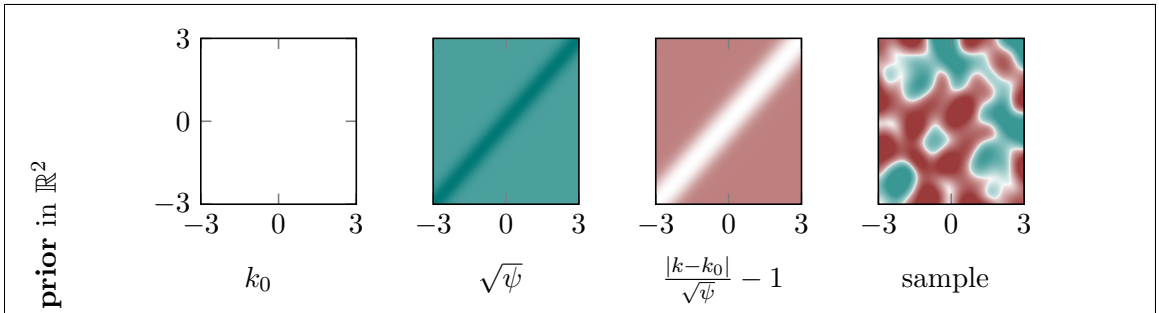
$$k_{SoR}(\boldsymbol{x}, \boldsymbol{z}) = k(\boldsymbol{x}, \boldsymbol{X}_U) k(\boldsymbol{X}_U, \boldsymbol{X}_U)^{-1} k(\boldsymbol{X}_U, \boldsymbol{z}) \tag{7}$$

where $\boldsymbol{X}_U$ is a set of M so called inducing inputs. The method proposed in this work (KMCG) is related to SoR. Readers familiar with SoR will be aware of the associated flaws, and methods to remedy them (Quiñonero-Candela and Rasmussen, 2005; Titsias, 2009b).

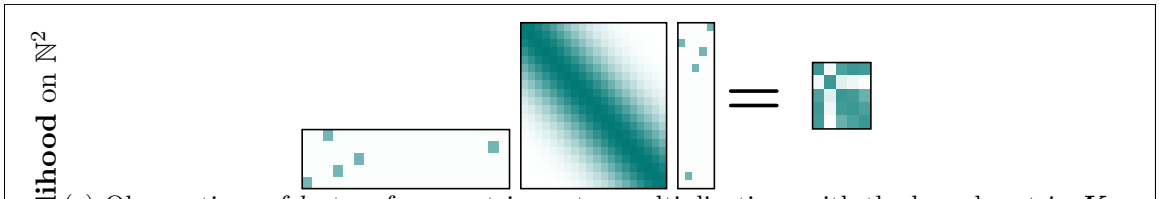
Figure 2: Schematic summary of our proposed kernel approximation method.



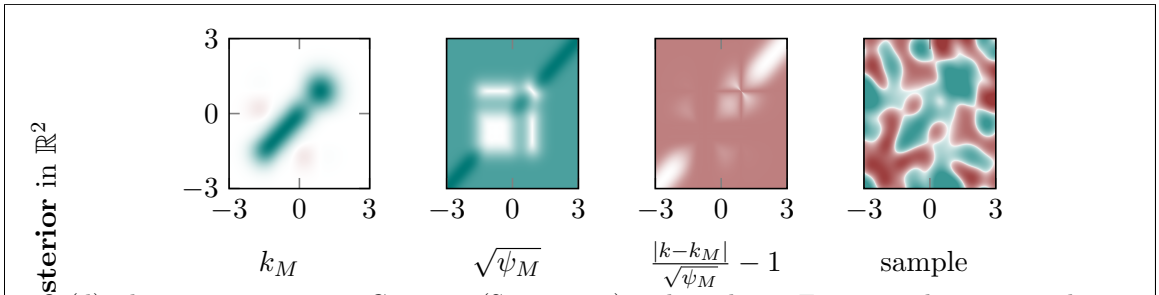
(a) The kernel k, here a squared exponential (Eq. 18) is assumed to be an unknown function.



(b) Section 2.2 describes a Kronecker-structured Gaussian process prior over the kernel. Above pictures show from left-to-right: prior mean (zero), prior standard deviation, the absolute error divided by the standard deviation minus one and a sample from this prior.



(c) Observations of k stem from matrix-vector multiplications with the kernel matrix $\boldsymbol{K}$ (Section 2.3), sketched using random columns of the identity matrix.



(d) The posterior is again Gaussian (Section 2.4) and similar to Figure 2a the pictures show from left-to-right: mean, standard deviation, relative error and a sample. By design, the posterior mean k_M is an approximation of finite rank which allows to efficiently solve the original least-squares problem (Section 2.1).

The *Deterministic Training Conditional (DTC)* approximation alleviates this issue by using the exact kernel for the prior uncertainty over the test inputs (Quiñonero-Candela and Rasmussen, 2005). In effect this is a substitution of Eq. (5) for Eq. (8) below.

$$\overline{c}(\boldsymbol{x}_*, \boldsymbol{z}_*) \approx k(\boldsymbol{x}_*, \boldsymbol{x}_{**}) - \boldsymbol{\phi}(\boldsymbol{x}_*)^* \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^* + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\phi}(\boldsymbol{z}_*) \tag{8}$$

We will apply the same substitution for our method KMCG. Another approach to this problem is taken by the FITC method (Quiñonero-Candela and Rasmussen, 2005). FITC can be obtained from SoR by substituting the approximate diagonal elements $k_{SoR}(\boldsymbol{x}, \boldsymbol{x})$ for their exact counterpart $k(\boldsymbol{x}, \boldsymbol{x})$. For our method, we found that this correction slightly worsens performance.

2.2. Prior

Consider a Gaussian process prior over bivariate functions

$$k \sim \mathcal{GP}(k_0, \gamma\psi) \tag{9}$$

where $\psi : \mathbb{X}^2 \times \mathbb{X}^2 \to \mathbb{R}$ is a covariance function over kernels and $\gamma \in \mathbb{R}^+$ is a scaling parameter. Since the posterior mean is meant to be a substitution for the exact kernel, this is an exchange of one least-squares problem for another. Without further assumptions, calculating the posterior over k is more expensive than computing the equations of interest (Equations 1 to 3).

Efficient inference is rendered possible by imposing the following structure on ψ

$$\psi(k(\boldsymbol{a}, \boldsymbol{b}), k(\boldsymbol{c}, \boldsymbol{d})) := \frac{1}{2} w(\boldsymbol{a}, \boldsymbol{c}) w(\boldsymbol{b}, \boldsymbol{d}) + \frac{1}{2} w(\boldsymbol{a}, \boldsymbol{d}) w(\boldsymbol{b}, \boldsymbol{c}) \tag{10}$$

for $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{X}$ and where w is a covariance function on the domain $\mathbb{X}$. Consider the first addend. It states that the similarity between $k(\boldsymbol{a}, \boldsymbol{b})$ and $k(\boldsymbol{c}, \boldsymbol{d})$ depends on the similarity of $\boldsymbol{a}$ and $\boldsymbol{c}$, and $\boldsymbol{b}$ and $\boldsymbol{d}$–a natural assumption for kernel matrices.

The second addend is a symmetrization of the first. Observe that each addend is a product kernel of two pairs of inputs and recall that a product kernel produces Kronecker product matrices. The sum of the two products leads to covariance matrices that have a *symmetric* Kronecker product form, *i.e.* $\forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{N \times N} : \psi(\boldsymbol{A}, \boldsymbol{B}) = \boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{N^2 \times N^2}$ (see Appendix A). This will allow a sufficiently efficient evaluation of the posterior. Fig. 2 visualizes the variance and shows samples from this prior for the toy setup from Fig. 1.

This choice of prior offers a trade-off between efficient tractable inference and the desire to encode as much prior structural information about the kernel as possible. One desirable property to encode is symmetry, and indeed, matrix-valued functions sampled from this prior distribution are symmetric (*c.f.* Fig. 2 for examples, Appendix A.1 for formal proof). Kernel functions are also positive definite. Unfortunately, since the positive definite cone is not a linear sub-space of the vector-space of real matrices, this property can not be encoded in a Gaussian prior, in closed form.[1] However, it *is* possible to guarantee positive-definiteness of the posterior mean point estimate through the specific choice of prior parameters $k_0 = 0$ (proof in Proposition 11, p. 27). For this reason, we adopt this choice for the remainder.

1. *e.g.* Hennig (2015) discusses this problem and possible solutions.

There are other properties of certain kernels that would be desirable to encode, but which are not feasible within the chosen framework without sacrificing fast computability. For example, stationarity of the kernel can not be represented by a prior with Kronecker structure in the covariance since $\boldsymbol{a}$ and $\boldsymbol{b}$ (and symmetrically $\boldsymbol{c}$ and $\boldsymbol{d}$) do not appear together as arguments to w.

The question remains how to choose w. Recall that w should reflect the similarity between $k(\boldsymbol{a}, \boldsymbol{b})$ and $k(\boldsymbol{c}, \boldsymbol{d})$ which depends on the similarity of $\boldsymbol{a}$ and $\boldsymbol{c}$, and $\boldsymbol{b}$ and $\boldsymbol{d}$. To measure the relationship between inputs is exactly the purpose of the kernel k and we therefore set

$$w := k$$

for the remainder. Even if k fails to capture similarity between inputs, as choice for w it still captures the similarity between the kernel values. Furthermore, samples from the approximate kernel will be a function of w and lastly, this choice is convenient computationally as expressions simplify.

2.3. Likelihood

Having specified a prior over k, we will now be concerned with how to obtain observations. Iterative solvers like conjugate gradients proceed by collecting a sequence of *linear* projections of the (kernel) matrix to be inverted, in the form of matrix-vector products. In fact, this general structure also describes the setting of non-adaptive approaches like inducing point methods, which can be interpreted as collecting multiplications of the kernel matrix with a set of *pre-specified* and *sparse* vectors (namely the unit selection vectors $e_{\boldsymbol{x}_{u_i}}$). We can use these matrix-vector products for learning a low-rank version of the kernel by introducing the linear operator

$$\boldsymbol{T}_{\boldsymbol{p}} : (\mathbb{X} \times \mathbb{X})^{\mathbb{R}} \to \mathbb{R}^{P^2}, k \mapsto \operatorname{vec}\left(\left[\iint k(\boldsymbol{x}, \boldsymbol{z}) p_i(\boldsymbol{x}) p_j(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z} \right]_{ij} \right) \tag{11}$$

where $i, j = 1...P$, $\boldsymbol{p} = [p_1, ..., p_P]$ are densities or distributions and $\operatorname{vec}(\boldsymbol{A})$ is a column vector created by stacking the rows of $\boldsymbol{A}$.

Example 1 (Matrix-vector multiplication) *Define* $\boldsymbol{T}_{\boldsymbol{p}}$ *with*

$$p_i(\boldsymbol{x}) := \sum_{j=1}^{M} s_{ij} \delta(\boldsymbol{x} - \boldsymbol{x}_{u_j}). \tag{12}$$

Then the evaluation of $\boldsymbol{T}_{\boldsymbol{p}} k$ *reduces to a matrix vector product, that is* $\operatorname{mat}(\boldsymbol{T}_{\boldsymbol{p}} k) = \boldsymbol{S}^{\intercal} k(\boldsymbol{X}_U, \boldsymbol{X}_U) \boldsymbol{S}$ *where* $\boldsymbol{S}_{ij} = s_{ij}$, $\boldsymbol{X}_U = [\boldsymbol{x}_{u_1}, ..., \boldsymbol{x}_{u_M}]$ *and* $\operatorname{mat}(\,)$ *transforms a* P^2 *vector into a* $P \times P$ *matrix, s.t.* $\operatorname{mat}(\operatorname{vec}(\boldsymbol{A})) = \boldsymbol{A}$.

The $\boldsymbol{x}_{u_j}$ can be datapoints or arbitrary elements of the domain $\mathbb{X}$. The choice $\boldsymbol{S}_{ij} := \delta_{ij}$ leads to the *Subset of Regressors* approximation (Proposition 1, p. 7).

Example 2 (Integrals with Eigenfunctions) *Let ϕ_i $i = 1, ..., P$ be orthogonal Eigenfunctions of k with respect to a density ν on $\mathbb{X}$,* i.e.

$$\int k(\boldsymbol{x}, \boldsymbol{z})\phi_i(\boldsymbol{z})\nu(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} = \lambda_i \phi_i(\boldsymbol{x})$$
$$\int \phi_i(\boldsymbol{z})\phi_j(\boldsymbol{z})\nu(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} = \delta_{ij}$$

where $\lambda_i \in \mathbb{R}$ and δ_{ij} is the Kronecker indicator function (compare Rasmussen and Williams (2006, p. 96)). Then for

$$p_i(\boldsymbol{x}) := \phi_i(\boldsymbol{x})\nu(\boldsymbol{x})$$

the observations $[\mathrm{mat}\,(\boldsymbol{T}_{\boldsymbol{p}}k)]_{ij} = \delta_{ij}\lambda_i$ *are spectral values of the kernel.*

In essence, this example shows another possibility to express prior knowledge over the kernel.

This likelihood leads to the *Projected Bayes Regressor* (Trecate et al., 1999) (Proposition 2, p. 8), which is a historical, deterministic precursor to the more widely known random Fourier feature expansion of Rahimi and Recht (2008).

2.4. Posterior and Subsumed Approximation Methods

The observation operator $\boldsymbol{T}_{\boldsymbol{p}}$ is a linear projection, and hence transforms the Gaussian prior into an also Gaussian posterior. Given the prior (Eq. 9) and any likelihood of the previous section, the posterior is Gaussian with:

$$\begin{aligned} p(k \mid \boldsymbol{Y}, \boldsymbol{T}_{\boldsymbol{p}}) &= \mathcal{N}(k_M, w_M) \\ k_M &= k_0 + (\boldsymbol{T}_{\boldsymbol{p}}\psi)^\intercal (\boldsymbol{T}_{\boldsymbol{p}}(\boldsymbol{T}_{\boldsymbol{p}}\psi)^\intercal)^{-1}(\mathrm{vec}\,(\boldsymbol{Y}) - \boldsymbol{T}_{\boldsymbol{p}}k_0) \\ \psi_M &= \psi - (\boldsymbol{T}_{\boldsymbol{p}}\psi)^\intercal (\boldsymbol{T}_{\boldsymbol{p}}(\boldsymbol{T}_{\boldsymbol{p}}\psi)^\intercal)^{-1}\boldsymbol{T}_{\boldsymbol{p}}\psi \end{aligned} \tag{13}$$

The concrete posterior depends on the choice of $\boldsymbol{T}_{\boldsymbol{p}}$. The following propositions presents approximation methods that have a view as GP inference with low-rank kernel and how they arise in our framework.

Proposition 1 (Subset of Regressors) *Consider the prior of Eq.* (9) *with $k_0 := 0$ and $w := k$ and the likelihood defined in Example 1 with $s_{ij} = \delta_{ij}$. Then the posterior mean k_M is equivalent to that of SoR:*

$$k_M(\boldsymbol{x}, \boldsymbol{z}) = k_{SoR} = k(\boldsymbol{x}, \boldsymbol{X}_U)k(\boldsymbol{X}_U, \boldsymbol{X}_U)^{-1}k(\boldsymbol{X}_U, \boldsymbol{z})$$

where $\boldsymbol{X}_U$ are inducing inputs, not necessarily part of $\boldsymbol{X}$.

The proof is part of Appendix B. An example of this posterior distribution is shown in Figure 2. The related method, *Fully Independent Conditional* (FIC) has a very similar kernel, $k_{FIC} = k_{SoR}(\boldsymbol{x}, \boldsymbol{z}) + \delta(\boldsymbol{x} - \boldsymbol{z})(k(\boldsymbol{x}, \boldsymbol{x}) - k_{SoR}(\boldsymbol{x}, \boldsymbol{x}))$. The structure indicates that this kernel should fit as well into our framework. One option that comes to mind, is to model the diagonal elements as certain, using a prior with mean $k_0(k(\boldsymbol{a}, \boldsymbol{b})) := \delta(\boldsymbol{a} - \boldsymbol{b})k(\boldsymbol{a}, \boldsymbol{b})$ and

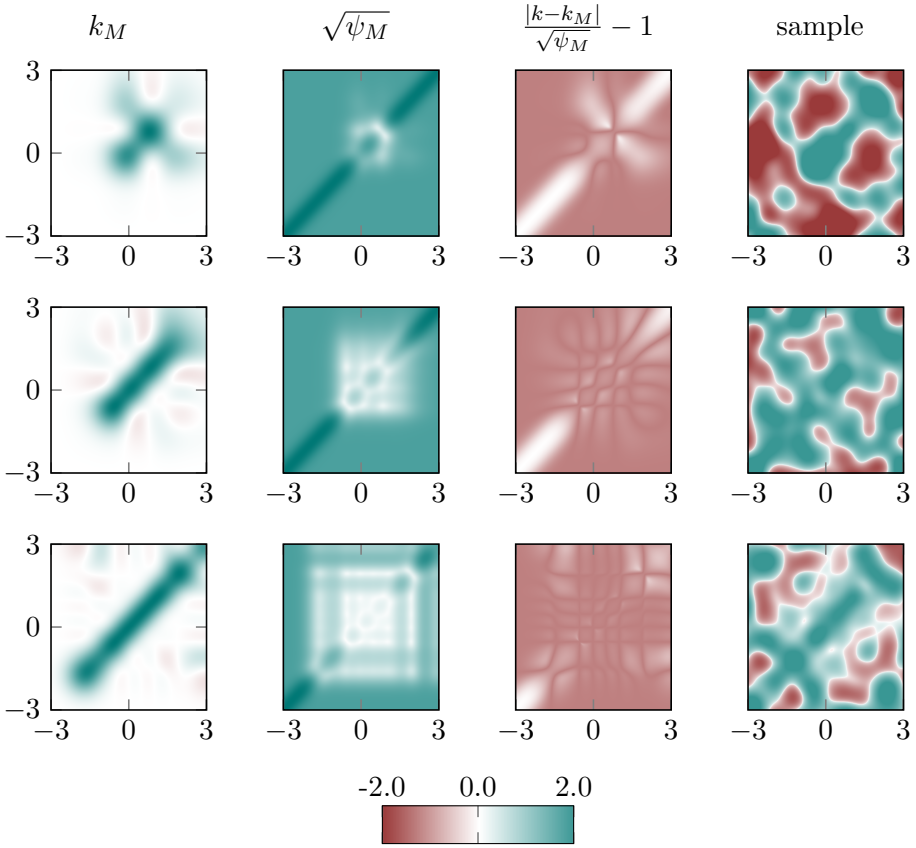


Figure 3: progression of the posterior (Eq. 14) for KMCG on the toy example from Figure 1 for $P = 2, 4$ and 8 conjugate gradients steps. The columns show from left to right: mean, standard deviation, standardized error (white refers to perfect calibration, green to overconfidence and red to underconfidence) and a sample.

covariance function $\psi'(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{c}, \boldsymbol{d}) := (1 - \delta(\boldsymbol{a} - \boldsymbol{b}))\psi(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{c}, \boldsymbol{d})(1 - \delta(\boldsymbol{c} - \boldsymbol{d}))$. The posterior mean, however, is in general not the *FIC* kernel, as for off-diagonal elements, the prediction differs to due to the certainty over the diagonal elements. Furthermore, the modification of the covariance function annuls the convenient algebraic properties of the associated covariance matrices and hence, this prior is dismissed as potential *FIC* competitor.

Another strategy could be to add the diagonal elements as observations. However, this is not possible with the operator as defined in Eq. (11) as it requires the mapping to a finite-dimensional vector. Also restricting the observation to test- and training points does not lead to *FIC*. It remains an open question whether *FIC* fits into our proposed kernel approximation scheme.[2]

Proposition 2 (Projected Bayes Regressor) *Consider the prior of Eq.* (9) *with* $k_0 := 0$ *and* $w := k$ *and the likelihood defined in Example 2. Let* λ_1 *to* λ_P *be the largest eigenvalues of the kernel* k *(w.r.t to the Mercer expansion) and assume the inputs* $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ *are independent*

2. We found that replacing heuristically the approximate diagonal for the exact diagonal does not improve performance.

and identical draws from ν. Then the posterior kernel k_M leads to the Projected Bayes Regressor *(Trecate et al., 1999).*

The proof is part of Appendix C.

3. Conjugate Gradients for Kernel Machines

The previous section introduced a probabilistic estimation rule for the kernel k. This section presents another data-collection approach using conjugate gradients that leads to a new approximation algorithm: *kernel machine conjugate gradients* (KMCG).

The interest to use conjugate gradients for kernel machines goes back to more than 25 years (Skilling, 1993) and is still continuing (Davies, 2015; Filippone and Engler, 2015). Albeit quadratic costs per step, CG has advantages over many of the approximation methods referenced in the introduction. CG has only one parameter, the desired precision, which is more natural than *e.g.* the number of inducing inputs for inducing point methods (Quiñonero-Candela and Rasmussen, 2005). This means, the computational budget of CG is not fixed in advance but varies as necessary for the problem at hand.

3.1. Conjugate Gradients

Conjugate gradients (Algorithm 1) is an iterative solver for linear equation systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a real, symmetric and positive definite matrix (Hestenes and Stiefel, 1952). In theory, CG returns the exact solution $\boldsymbol{x}$ after N steps. In practice, CG is used as approximate solver and can provide good approximations to $\boldsymbol{x}$ in significantly less than N steps.

The costs of running CG are dominated by a matrix-vector multiplication in each step which in general has complexity $\mathcal{O}(N^2)$. The number of necessary steps depends on the eigenvalues $\lambda_1 > ... > \lambda_N$ of $\boldsymbol{A}$. The following summary of the properties of CG is an excerpt from Nocedal and Wright (1999, Chapter 5.1). We use the notation $\|\boldsymbol{x}\|^2_{\boldsymbol{\Lambda}} := \boldsymbol{x}^\intercal\boldsymbol{\Lambda}\boldsymbol{x}$ for any symmetric and positive definite matrix $\boldsymbol{\Lambda}$. The $\boldsymbol{A}$-error of CG decreases in each step with

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}\|^2_{\boldsymbol{A}} \leq \left(\frac{\lambda_{N-k} - \lambda_1}{\lambda_{N-k} + \lambda_1}\right)^2 ||\boldsymbol{x}_0 - \boldsymbol{x}||^2_{\boldsymbol{A}}$$

and one can show that if $\boldsymbol{A}$ has at most r distinct eigenvalues, then CG terminates after r steps with the exact solution. Thus, conjugate gradients is particularly advantageous if the eigenvalues of $\boldsymbol{A}$ are clustered or decay rapidly.

3.2. Kernel-machine Conjugate Gradients

Our approach is to run conjugate gradients for P steps on a kernel matrix of size M and to treat the matrix multiplications ($\boldsymbol{z}_i$ in Algorithm 1) as observations in the model presented in Section 2. Formally, the likelihood is defined similar to the SoR likelihood (Example 1) albeit scaled.

Definition 3 (Conjugate-gradients likelihood) *Choose a subset $\boldsymbol{X}_M$ of size M from $\boldsymbol{X}$ and denote as $\boldsymbol{y}_M \in \mathbb{R}^M$ the vector that contains the corresponding entries of $\boldsymbol{y}$. Run*

Algorithm 1 Conjugate Gradients

procedure ConjugateGradients($\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{x}_0$, ε)
  $\boldsymbol{r}_0 \leftarrow \boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}$ ▷ The initial residual ...
  $\boldsymbol{s}_0 \leftarrow -\boldsymbol{r}_0$ ▷ ... is the first search direction.
  $i \leftarrow 0$
  while $||\boldsymbol{r}_i||_2 > \varepsilon$ **do**
    $\boldsymbol{z}_i \leftarrow \boldsymbol{A}\boldsymbol{s}_i$ ▷ the most expensive step: $\mathcal{O}(N^2)$ matrix-multiplication
    $\alpha_i \leftarrow \frac{\boldsymbol{r}_i^\intercal \boldsymbol{r}_i}{\boldsymbol{s}_i^\intercal \boldsymbol{z}_i}$ ▷ optimal linesearch along $\boldsymbol{s}_i$ for $\phi(\boldsymbol{x}) := \boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{x}^\intercal \boldsymbol{b}$
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \alpha_i \boldsymbol{s}_i$ ▷ update to the solution
    $\boldsymbol{r}_{i+1} \leftarrow \boldsymbol{r}_i + \alpha_i \boldsymbol{z}_i$ ▷ analogue update to the residual
    $\boldsymbol{s}_{i+1} \leftarrow -\boldsymbol{r}_{i+1} + \frac{\boldsymbol{r}_{i+1}^\intercal \boldsymbol{r}_{i+1}}{\boldsymbol{r}_i^\intercal \boldsymbol{r}_i} \boldsymbol{s}_i$ ▷ Gram-Schmidt applied to the new residual
    $i \leftarrow i + 1$
  end while
  return $\boldsymbol{x}_i$
end procedure

conjugate gradients (Algorithm 1 on p. 10) with $\boldsymbol{x}_0 := \boldsymbol{0}$, $\boldsymbol{A} = k(\boldsymbol{X}_M, \boldsymbol{X}_M)$, $\boldsymbol{b} = \boldsymbol{y}_M$ *and* $\varepsilon := 0.01||\boldsymbol{b}||_2$. *In Equation* (11) *set*

$$p_i(\boldsymbol{x}) := \sum_{j=1}^{M} s_j \delta(\boldsymbol{x} - \boldsymbol{x}_j)$$

where s_j *is the* j*-th entry of vector* $\boldsymbol{s}_i$ *in iteration* i *of Algorithm 1.*

Remark 4 *KMCG uses only the CG search directions* $\boldsymbol{s}_1, ..., \boldsymbol{s}_P$ *and* not *the solution* $\hat{\boldsymbol{x}}$. *Other search directions (*e.g. *from the Lanczos process) could also be used*[3].

Using this likelihood, the resulting approximate kernel (Eq. (13)) and approximate Equations are (Proposition 9, p. 26):

$$\hat{k}_M(\boldsymbol{x}_*, \boldsymbol{x}_{**}) = k(\boldsymbol{x}_*, \boldsymbol{X}_M)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal k(\boldsymbol{X}_M, \boldsymbol{x}_{**}) \tag{14}$$

$$\hat{f}(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{X}_M)\boldsymbol{S}(\boldsymbol{R}^\intercal \boldsymbol{R} + \sigma^2 \boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S})^{-1}\boldsymbol{R}^\intercal \boldsymbol{y} \tag{15}$$

$$\begin{aligned}\hat{c}(\boldsymbol{x}_*, \boldsymbol{x}_{**}) = {} & k(\boldsymbol{x}_*, \boldsymbol{x}_{**}) - k(\boldsymbol{x}_*, \boldsymbol{X}_M)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal k(\boldsymbol{X}_M, \boldsymbol{x}_{**}) \\ & + \sigma^2 k(\boldsymbol{x}_*, \boldsymbol{X}_M)\boldsymbol{S}\left(\boldsymbol{R}^\intercal \boldsymbol{R} + \sigma^2 \boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S}\right)^{-1}\boldsymbol{S}^\intercal k(\boldsymbol{X}_M, \boldsymbol{x}_{**})\end{aligned}$$

$$\begin{aligned}\ln \hat{Z} = {} & \frac{1}{2\sigma^2}(\boldsymbol{y}^\intercal \boldsymbol{y} - \boldsymbol{y}^\intercal \boldsymbol{R}(\boldsymbol{R}^\intercal \boldsymbol{R} + \sigma^2 \boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S})^{-1}\boldsymbol{R}^\intercal \boldsymbol{y}) \\ & + \frac{1}{2}\ln|\boldsymbol{R}^\intercal \boldsymbol{R} + \sigma^2 \boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S}| - \frac{1}{2}|\boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S}| \\ & + \frac{1}{2}(N - P)\ln \sigma^2 + \frac{1}{2}N \ln 2\pi\end{aligned} \tag{16}$$

where $\boldsymbol{S} := \begin{bmatrix}\boldsymbol{s}_1 & ... & \boldsymbol{s}_P\end{bmatrix}$, $\boldsymbol{R} := k(\boldsymbol{X}_M, \boldsymbol{X})\boldsymbol{S}$ and P is the number of CG-iterations. We call this approximation *kernel machine conjugate gradients* (KMCG).

3. In exploratory experiments, we found conjugate gradients search directions to perform slightly better than Lanczos search directions.

Algorithm 2 Kernel Machine Conjugate Gradients

procedure KMCG(k, $\boldsymbol{X}$, $\boldsymbol{y}$, σ^2, ε)
  ▷ We assume (w.l.o.g.) that the inducing inputs are a subset of $\boldsymbol{X}$, denoted by $\boldsymbol{X}_M$.
  ▷ Let $\boldsymbol{y}_M$ the be corresponding entries of $\boldsymbol{y}$.
  Conjugate Gradients($k(\boldsymbol{X}_M, \boldsymbol{X}_M), \boldsymbol{y}_M, \varepsilon$) ▷ ignore solution $\hat{\boldsymbol{x}}$
  $\boldsymbol{S} \leftarrow [\boldsymbol{s}_1, ..., \boldsymbol{s}_P]$ ▷ collect CG search directions
  $\boldsymbol{Z} \leftarrow [\boldsymbol{z}_1, ..., \boldsymbol{z}_P]$ ▷ $\boldsymbol{Z} = \boldsymbol{K}_M \boldsymbol{S}$
  if $M < N$ **then**
    $\boldsymbol{R} \leftarrow k(\boldsymbol{X}, \boldsymbol{X}_M)\boldsymbol{S}$
  else
    $\boldsymbol{R} \leftarrow \boldsymbol{Z}$ ▷ When $\boldsymbol{X}_M = \boldsymbol{X}$ above matrix multiplication is not necessary.
  end if
  $\boldsymbol{L}_1 \leftarrow \mathrm{chol}(\boldsymbol{S}^\intercal \boldsymbol{Z})$ ▷ precompute required Choleskies
  $\boldsymbol{L}_2 \leftarrow \mathrm{chol}(\sigma^2 \boldsymbol{S}^\intercal \boldsymbol{Z} + \boldsymbol{R}^\intercal \boldsymbol{R})$
  evaluate Eqs. (15) to (16)
end procedure

3.3. Properties

Figure 3 shows how the approximation to the kernel progresses for the toy example from Figure 1. Computing the Cholesky of $\boldsymbol{R}^\intercal \boldsymbol{R} + \sigma^2 \boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S}$ costs $\mathcal{O}(NMP)$. After that, evaluating the mean prediction is possible in $\mathcal{O}(M)$ and the variance in $\mathcal{O}(MP)$.

In case $P = M$, KMCG reduces to SoR since all occurrences of $\boldsymbol{S}$ in Equation (14) cancel and what remains is the SoR kernel (Equation 7). If $\boldsymbol{K}_M$ has a favorable distribution of eigenvalues such that conjugate gradients terminates in less than M steps (*c.f.* Section 3.1), KMCG can be used to speed up SoR.[4] In practice, this kind of advantage can only be expected to be beneficial when realized in low-level code. The level of efficiency of existing low-level linear algebra routines makes it challenging to evaluate this area.

Recall that the computational complexity of CG for the solution of Eq. (9) in P iterations is $\mathcal{O}(N^2 P)$, that of inducing point methods with M inducing inputs is $\mathcal{O}(NM^2)$, and KMCG running for P iterations on M inducing points has complexity $\mathcal{O}(NMP)$. The subsequent evaluation section is dedicated to the case $M = N$, *i.e.* using the whole data set which places KMCG in direct competition to plain conjugate gradients.

3.3.1. Relationship to the Nadaraya-Watson estimator

Taking only one step ($P = 1$) implies $\boldsymbol{S} = \boldsymbol{y}_M$ and Equation (15) takes the following form

$$\hat{f}(\boldsymbol{x}_*) = \alpha \sum_{m=1}^{M} k(\boldsymbol{x}_m, \boldsymbol{x}_*) y_m$$

where $\alpha = \frac{\boldsymbol{y}_M^\intercal \boldsymbol{K}_M \boldsymbol{y}_M}{\sigma^2 \boldsymbol{y}_M^\intercal \boldsymbol{K}_M \boldsymbol{y}_M + \boldsymbol{y}_M^\intercal \boldsymbol{K}_M \boldsymbol{K}_M \boldsymbol{y}_M}$. The equation bears resemblance to the Nadaraya-Watson estimator (Bishop, 2006, p. 301f): a sum over all training targets weighted by the

4. The same applies to related methods such as *DTC* (Quiñonero-Candela and Rasmussen, 2005) and Titsias' method (Titsias, 2009b).

similarity of the corresponding input to the test input. However, the scaling-factor α is different.

Since conjugate gradients solves the linear system for the mean prediction, it is to be expected that this might incur a trade-off to the approximation of the variance. See Section 4 for an empirical evaluation of the quality of the variance estimate.

3.3.2. Uncertainty

In addition to the posterior mean k_M, the Gaussian formulation of the approximation problem also provides a posterior variance ψ_M. It is a natural question to which degree this object can be interpreted as a notion of uncertainty or, more specifically, as an estimate of the square error $(k - k_M)^2$.

This section provides an analysis of this covariance for KMCG, showing it to be an outer bound on the true error. Figure 3 visualizes this for the toy data set from Figure 1.

Proposition 5 (relative error bound) *The relative size of estimation error and error estimate is bounded from above by 2.*

$$\frac{(k(\boldsymbol{x},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z}))^2}{\psi_M(k(\boldsymbol{x},\boldsymbol{z}), k(\boldsymbol{x},\boldsymbol{z}))} \leq 2$$

Proof To simplify notation, define $\boldsymbol{k}_x^\intercal := k(\boldsymbol{x}, \boldsymbol{X})$ and $\boldsymbol{G} := \boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{K} \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal$.

For KMCG posterior mean and variance evalute to (Appendix B):

$$\begin{aligned}
k_M(\boldsymbol{x},\boldsymbol{z}) &= \boldsymbol{k}_x^\intercal \boldsymbol{G} \boldsymbol{k}_z, \\
\psi_M(k(\boldsymbol{x},\boldsymbol{z}), k(\boldsymbol{x},\boldsymbol{z})) &= 1/2 \left(k(\boldsymbol{x},\boldsymbol{x})k(\boldsymbol{z},\boldsymbol{z}) + k(\boldsymbol{x},\boldsymbol{z})^2 - \boldsymbol{k}_x^\intercal \boldsymbol{G} \boldsymbol{k}_x \boldsymbol{k}_z^\intercal \boldsymbol{G} \boldsymbol{k}_z - (\boldsymbol{k}_x^\intercal \boldsymbol{G} \boldsymbol{k}_z)^2 \right) \\
&= 1/2 \left(k(\boldsymbol{x},\boldsymbol{x})k(\boldsymbol{z},\boldsymbol{z}) + k(\boldsymbol{x},\boldsymbol{z})^2 - k_M(\boldsymbol{x},\boldsymbol{x})k_M(\boldsymbol{z},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z})^2 \right).
\end{aligned}$$

As a variance $\psi_M(k(\boldsymbol{x},\boldsymbol{x}), k(\boldsymbol{x},\boldsymbol{x}))$ is always larger than 0 which implies $k(\boldsymbol{x},\boldsymbol{x}) \geq k_M(\boldsymbol{x},\boldsymbol{x})$ for all $\boldsymbol{x}$. This allows to lower bound $\psi_M(k(\boldsymbol{x},\boldsymbol{z}), k(\boldsymbol{x},\boldsymbol{z}))$ by $\frac{1}{2}k(\boldsymbol{x},\boldsymbol{z})^2 - \frac{1}{2}k_M(\boldsymbol{x},\boldsymbol{z})^2$ leading to

$$\begin{aligned}
\frac{(k(\boldsymbol{x},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z}))^2}{\psi_M(k(\boldsymbol{x},\boldsymbol{z}), k(\boldsymbol{x},\boldsymbol{z}))} &\leq 2 \frac{(k(\boldsymbol{x},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z})^2}{k(\boldsymbol{x},\boldsymbol{z})^2 - k_M(\boldsymbol{x},\boldsymbol{z})^2} \\
&= 2 \frac{(k(\boldsymbol{x},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z})^2}{(k(\boldsymbol{x},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z}))(k(\boldsymbol{x},\boldsymbol{z}) + k_M(\boldsymbol{x},\boldsymbol{z}))} \\
&= 2 \frac{|k(\boldsymbol{x},\boldsymbol{z}) - k_M(\boldsymbol{x},\boldsymbol{z})|}{k(\boldsymbol{x},\boldsymbol{z}) + k_M(\boldsymbol{x},\boldsymbol{z})} \\
&\leq 2.
\end{aligned}$$

■

3.4. Related Work

In terms of using conjugate gradients for kernel machines there is related work by Filippone and Engler (2015). Their algorithm ULISSE is aimed at the estimation of the marginal likelihood $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ where $\boldsymbol{\theta}$ are hyper-parameters of the kernel k. They use a randomized conjugate gradients to estimate gradients of the log-marginal likelihood (Eq. (3)) which in combination with Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) allows to sample from $p(\boldsymbol{\theta} \mid \boldsymbol{y})$. Our work is complementary to ULISSE. While running CG the matrix multiplications the inference perspective in Section 2 can be used to build a low-rank approximation of the kernel matrix which can serve as preconditioner for the next SGLD step.

Using the Kronecker product for efficient inference has been explored before for example in the KISS-GP framework (Wilson and Nickisch, 2015). The difference to this work is that Wilson and Nickisch (2015) factorize the kernel matrix $\boldsymbol{K}$ into a Kronecker-product where here it is the covariance matrix of the prior $\psi(\boldsymbol{K}, \boldsymbol{K})$ over the kernel that has Kronecker structure (cf. Eq. 9). A synergy between their and our approach is hard to imagine. However, the follow-up work by Pleiss et al. (2018) uses Lanczos iteration to build a low-rank approximation of a kernel matrix $\boldsymbol{C}$ for the variance prediction. Presumably, one could use instead KMCG.

4. Empirical Comparison of Conjugate Gradients and Kernel Machine Conjugate Gradients

This section elaborates the conceptual differences between CG and KMCG and then compares both algorithms with numerical experiments. Consider Equation (1), restated below for convenience.

$$\bar{f}(\boldsymbol{x}_*) = \boldsymbol{k}_*^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} \tag{1}$$

CG computes an approximation to $(\boldsymbol{K} + \sigma^2 I)^{-1}\boldsymbol{y}$ and uses the exact $\boldsymbol{k}_*$. In contrast, KMCG computes an approximation to k and substitutes $\boldsymbol{k}_*$ as well. That the systematic replacement of the kernel can be of importance has been noted before by Rasmussen and Williams (2006, p. 177) when comparing SoR and the Nyström method (Williams and Seeger, 2001). The SoR method approximates k with the kernel in Equation (7). In contrast Nyström uses the exact $\boldsymbol{k}_*$ such that the predictive variance (Eq. 2) can become negative. They further observed that for large M, Nyström and SoR have a similar performance, yet for small M Nyström performs poorly. We made the same observations for CG and KMCG in the following comparison on common regression problems.

Classical conjugate gradients is used to solve the equations $(\boldsymbol{K} + \sigma^2 \boldsymbol{I})\boldsymbol{\alpha} = \boldsymbol{y}$. In contrast, since the goal of KMCG is to learn an approximation to the kernel, the algorithm runs conjugate gradients on $\boldsymbol{K}\boldsymbol{\alpha} = \boldsymbol{y}$, *i.e.* without noise term. Both methods were evaluated in terms of the average relative error

$$\varepsilon_f := \frac{1}{n_*} \sum_{k=1}^{n_*} \left| \frac{\bar{f}(\boldsymbol{x}_{*,k}) - \hat{f}(\boldsymbol{x}_{*,k})}{\bar{f}(\boldsymbol{x}_{*,k})} \right|, \tag{17}$$

where $\boldsymbol{x}_{*,k}$ is a test input not part of the training set.

The text book version of conjugate gradients in Algorithm 1 is known to be numerically unstable[5] (Golub and Van Loan, 2013, p. 635) and there exist different strategies to cope with this problem. We refer the interested reader to Golub and Van Loan (2013, p. 562f) and the references therein. To explore the potential of our method, we bypass this implementation issue using the slowest[6] yet most stable solution: complete reorthogonalization[7] (Golub and Van Loan, 2013, p. 564) and the explicit projection-method formulation (Saad, 2003, p. 135 Eq. (5.7)) to compute $\boldsymbol{\alpha}$.

Therefore the following comparison will be conceptually, *i.e.* over the number of conjugate gradient steps. For completeness, Appendix E.1 contains results how KMCG performs in wall-clock time. Often the baseline methods converge faster since block-matrix multiplication is faster than looped matrix-vector multiplication.

Baseline methods are the *Fully Independent Training Conditional* (FITC) approximation (Quiñonero-Candela and Rasmussen, 2005) and the *Variational Free Energy* (VFE) method (Titsias, 2009a), with inducing inputs randomly selected from the data set as recommended by Chalupka et al. (2013). The baseline runs were repeated 10 times and besides the average, each figure shows also the progressive minimum and maximum over all runs, to take into account for more elaborate inducing input selection schemes.

In all our experiments, we used two popular stationary kernel functions: automatic relevance determination (ARD) Squared Exponential and ARD Matérn 5/2 (Rasmussen and Williams, 2006, p. 83f, p. 106),

$$k_{SE}(d(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Lambda})) = \theta_f \exp\left(-\frac{1}{2}d^2\right) \tag{18}$$

$$k_{52}(d(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Lambda})) = \theta_f \left(1+\sqrt{5}d+\frac{5}{3}d^2\right)\exp\left(-\sqrt{5}d\right) \tag{19}$$

where $d = d(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Lambda}) = ||\boldsymbol{x}-\boldsymbol{z}||_{\boldsymbol{\Lambda}}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix. All experiments were executed with Matlab R2019a on an Intel i7 CPU with 32 Gigabytes of RAM running Ubuntu 18.04.

4.1. Common Regression data sets

The data sets chosen are small such that computation of the exact GP is still feasible. The origin and purpose of each data set can be found in Appendix D. Each data set has been shuffled and split into two sets, using one for training and the other for testing. For each data set, we optimized the kernel parameters running Carl Rasmussen's `minimize` function[8] for 100 optimization-steps, where initially all kernel hyper-parameters are set to 1.

Fig. 4 shows how the average relative error develops for the described setup[9]. The number of inducing inputs M was set to $M = \sqrt{NP}$ such that $\mathcal{O}$-notation costs are equivalent to KMCG: Since KMCG uses multiplications with $\boldsymbol{K}$ for observations, the costs per CG-step

5. see additional results in Appendix E.3
6. Computing the exact solution is actually faster.
7. We experimented with selective reorthogonalization (Simon, 1984) but found it in our experiments to be slower than full reorthogonalization.
8. This method is part of the GPML toolbox (Rasmussen and Nickisch, 2010), see `http://www.gaussianprocess.org/gpml/code/matlab/doc`.
9. Since the Matérn kernel experiments look very similar, these results are in Appendix E.2

are $\mathcal{O}(N^2P)$. The upper x-axis displays the number of conjugate gradients steps, the lower x-axis, the number of inducing inputs.

During early iterations the performance of CG is not as reliable as KMCG and the latter also improves more consistently. In comparison to the baselines, KMCG often provides a worse approximation to start with but exhibits a faster convergence rate.

In contrast to plain conjugate gradients, KMCG naturally provides estimates for variance (Eq. 2) and evidence (Eq. 3). Define the average relative errors ε_{var} and ε_{ev} analogously to Equation (17), respectively. Figure 6 and 5 show the average relative error of these estimates in comparison to the baselines. For all data sets one can observe that the approximation quality of KMCG for the evidence (Eq. (3)) is improving at first and then worsening. KMCG is better at approximating the quadratic form than the determinant. Therefore, the approximation often 'overshoots'.

The baselines clearly outperform KMCG in these experiments. A possible explanation is that the baselines provide a better overall-approximation to the kernel matrix: After P CG-steps, the KMCG kernel is of rank P whereas using M inducing inputs, the VFE kernel is of rank M (so is the FITC kernel, putting the diagonal correction aside). Since $M = \sqrt{NP}$, the baselines can afford more inducing inputs M than KMCG can afford CG-steps P.

Overall, when it comes to real-time, the baselines are preferable over KMCG. The picture changes when matrix-multiplication is less expensive than $\mathcal{O}(N^2)$ which is investigated in the next section.

4.2. Grid-structured data sets

In the previous section the baselines are the preferable estimators over KMCG. This changes when matrix-multiplication costs less than $\mathcal{O}(N^2)$. For example when the kernel is a product kernel (such as squared exponential) and the data set has grid-structure, the cost for matrix-multiplication is almost linearly in the number of data-points (Wilson et al., 2014) such that the number of CG-steps KMCG can take, matches the number of baseline inducing inputs.

4.2.1. Artificial data sets

The data sets considered in the following are artificial multi-dimensional grids.[10] For the training set, along each axis G points are equally spaced in $[-G/4, G/4]$ distorted by Gaussian noise $\mathcal{N}(0, 10^{-3})$. One hundred test inputs are uniformly distributed over the $[-G/4, G/4]$ cube. Targets are drawn from a Gaussian process with squared exponential kernel (length scales and amplitude equal to 1). The number of inducing inputs had to be capped at 500 due to memory limitations.

Figure 7 shows how the approximation error to mean, variance and likelihood term evolves, zoomed in on the first 100 steps. For reference, Fig. 7 also shows a 10×10 data set to give an idea how each method would evolve when investing more computational power would be feasible. In the appendix, Figure 10 shows the same comparison over time for the whole 500 steps, stopping KMCG when it becomes slower than the baselines.

10. Computing the exact solution is feasible exploiting the Kronecker structure of the kernel matrix which we use to evaluate the quality of the approximation methods. However, we may imagine datapoints missing, s.t. matrix-vector multiplication is fast but computing the exact solution is not.

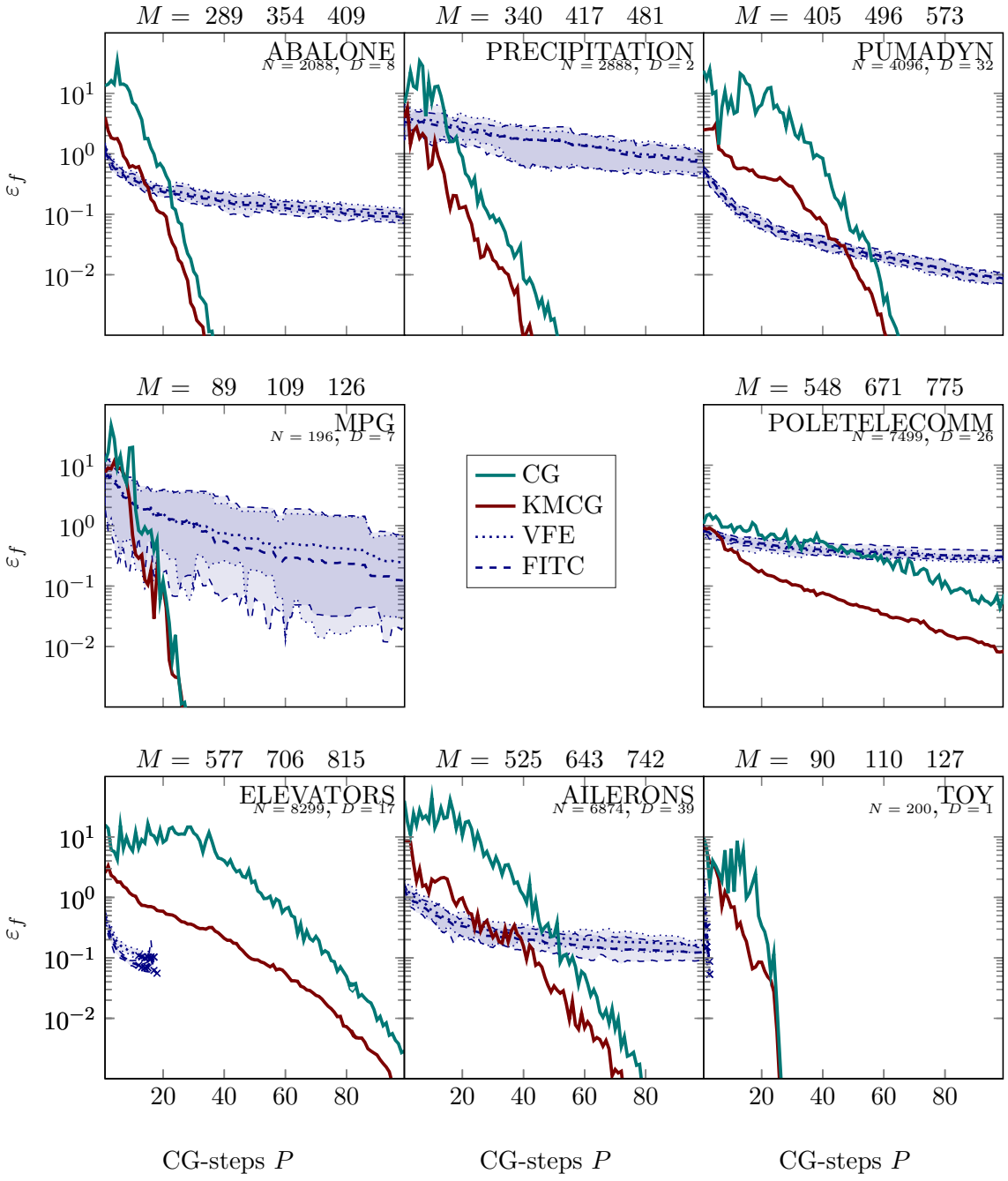


Figure 4: progression of the relative error ε_f as a function of the number of iterations of CG and KMCG for different data sets using the squared-exponential kernel (Eq. 18). The bottom axis is the number of CG-steps, which is the same for all plots. Therefore, this axis is visible only in the last row. The top axis denotes the number of inducing inputs used by the baseline methods. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

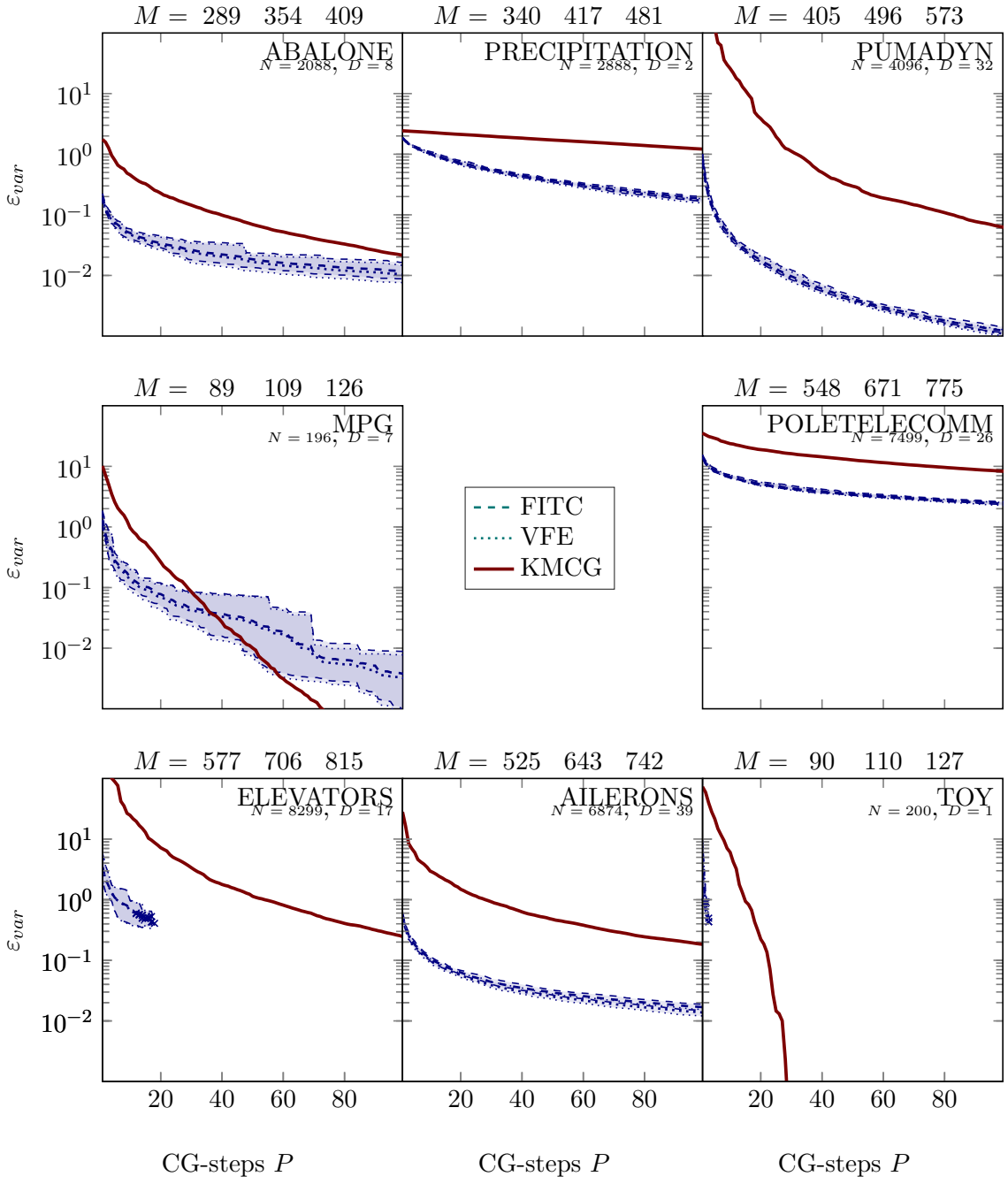


Figure 5: progression of the relative error of the variance ε_{var} as a function of the number of iterations of KMCG and baseline for different data sets using the squared-exponential kernel (Eq. 18). The bottom axis is the number of CG-steps, which is the same for all plots. Therefore, this axis is visible only in the last row. The top axis denotes the number of inducing inputs used by the baseline methods. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

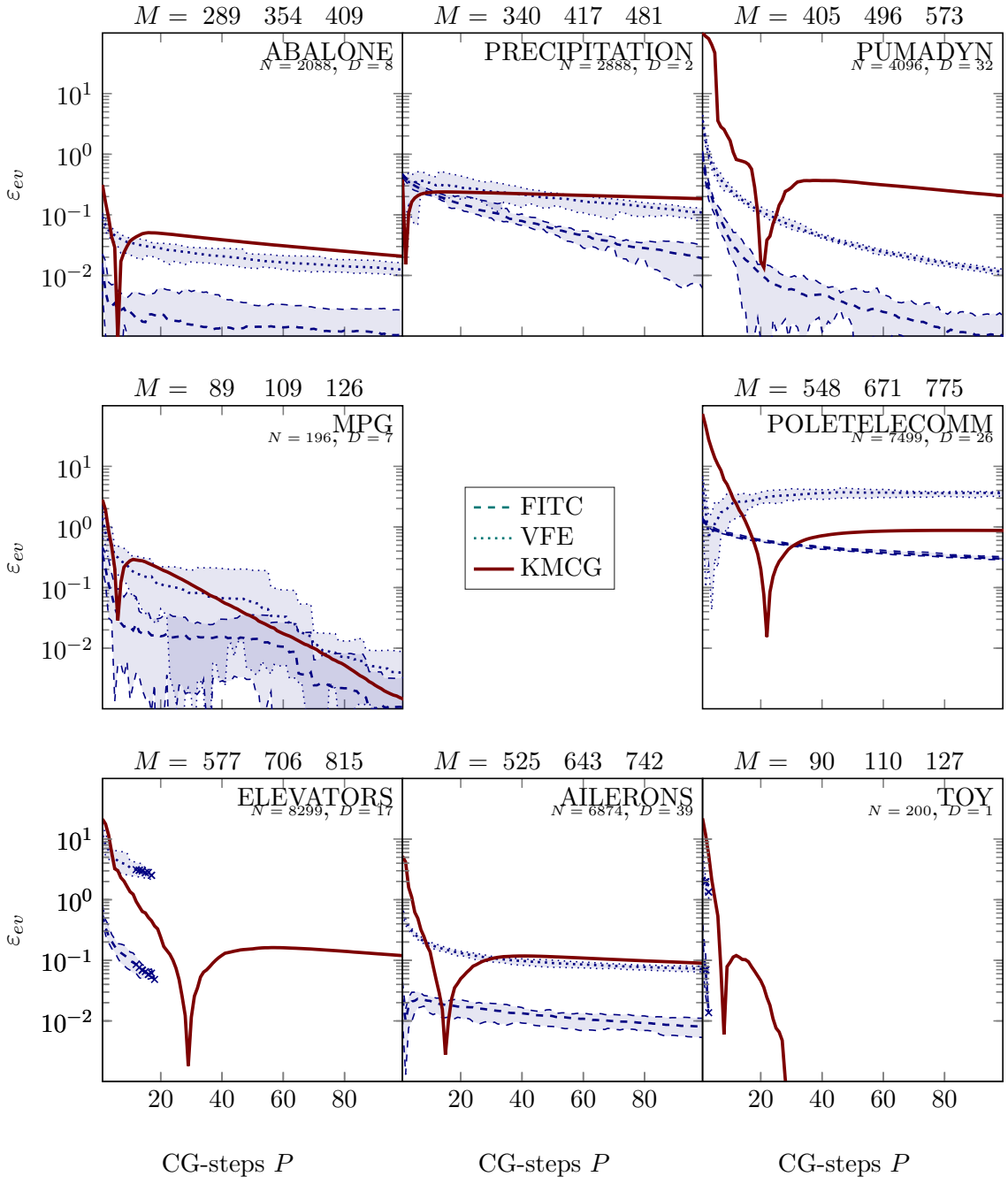


Figure 6: progression of the relative error of the evidence ε_{ev} as a function of the number of iterations of baseline and KMCG for different data sets using the squared-exponential kernel (Eq. 18). The bottom axis is the number of CG-steps, which is the same for all plots. Therefore, this axis is visible only in the last row. The top axis denotes the number of inducing inputs used by the baseline methods. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run. The small spikes in the plots where KMCG appears to be close to the solution correspond to changes of the estimate from too small to too large.

On these data sets KMCG dominates the baseline methods. After already one hundred CG-steps KMCG provides a useful approximation to the posterior mean whereas the baselines hardly show any progress. For the variance, the same computational effort is not enough. Though the baselines find better solutions, all methods essentially fail to arrive at a satisfactory solution of a relative error below one. The issue is that all methods overestimate the posterior variance by two orders of magnitude. The picture is similar for the evidence, albeit the approximations are closer to the truth and KMCG performs slightly better on average.

4.2.2. Natural Sound Modeling

For a real-world example of a grid-structured data set, we repeat the Natural Sound Modeling experiment considered by Turner (2010); Wilson and Nickisch (2015) and Dong et al. (2017). Given the intensity of a sound signal recorded over time, the objective is to recover the signal in missing regions. All inputs (*i.e.* including missing) are equidistant and hence the kernel matrix (over all inputs) is Toeplitz for stationary kernel. The kernel matrix over the given inputs is not Toeplitz, which forbids to use this structure for exact inference. Nevertheless matrix-vector-multiplication can be performed in linear time.

We use the squared-exponential kernel with the hyper-parameters used by Dong et al. (2017). Since the exact posterior is infeasible to compute, we report only the standardized mean squared-error:

$$SMSE := \frac{1}{\mathbb{V}[\boldsymbol{y}]} \sum_{j=1}^{N_*} (\boldsymbol{y}_{*,j} - \hat{f}(\boldsymbol{x}_{*,j}))^2.$$

To conform with the original experiment, we added for each baseline method a run the inducing inputs where chosen to be on a regular grid. The result of this run correspond to the minimum. Figure 8 confirms the observations from the previous section that KMCG arrives at satisfactory solutions faster than baseline, if matrix-vector multiplication is not an issue.

5. Conclusion

We have presented a new approximate inference method for kernel machines that showed how linear solvers can be used in combination with low-rank kernel approximations. The approach is based on a probabilistic numerics viewpoint: the kernel k is treated as a latent quantity and a linear solver is used for collecting observations of k. By design, the resulting approximate kernel is of low rank and is plugged into the nonparametric least-squares problem. The approach is not restricted to least-squares problems but applicable in any scenario where the bottleneck is the inversion of a large kernel matrix, as *e.g.* GP classification.

Our *kernel machine conjugate gradients* (KMCG), consistently outperforms plain conjugate gradients in numerical experiments. This does not change the fact that standard dense kernel least-squares problems are often more efficiently solved by inducing point methods. However, as demonstrated in Section 4.2, in the settings which allow fast multiplication with the kernel matrix, the new algorithm can improve upon the state of the art.

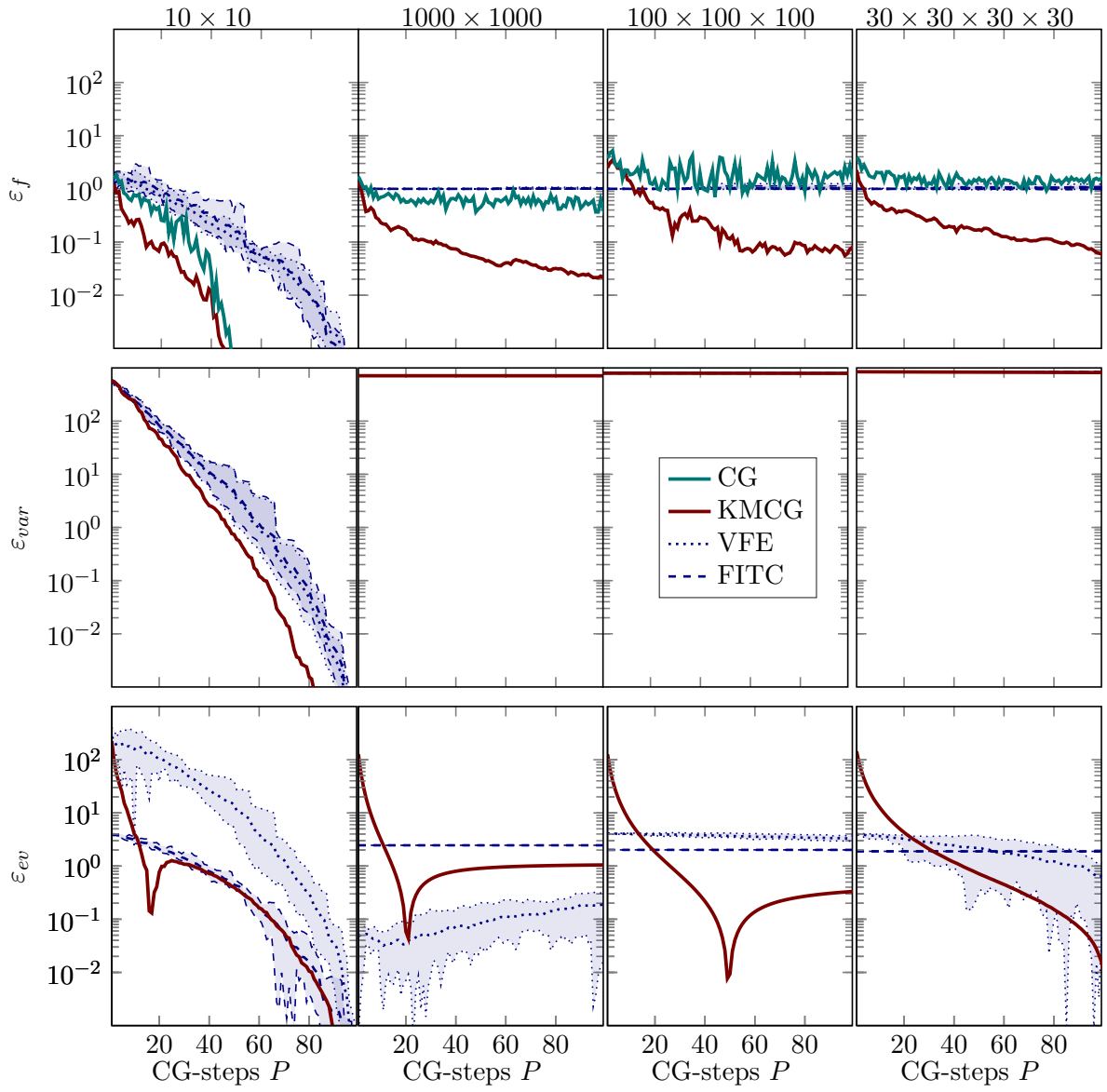


Figure 7: comparison of baseline and KMCG on grid-structured data sets using the squared exponential kernel (Eq. 18). The shaded area visualizes minimum and maximum over all baseline runs.

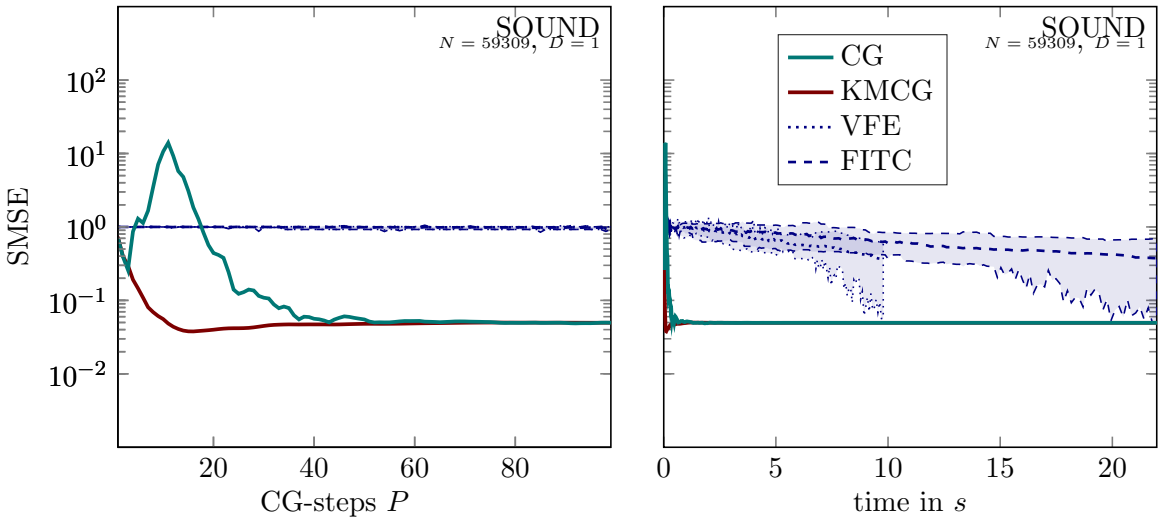


Figure 8: comparison of KMCG and CG on the SOUND data set, using the squared-exponential kernel (Eq. 18). **Left panel:** progression of the standardized mean-squared error (SMSE) over the first 100 iterations of CG and KMCG. **Right panel:** the same comparison with unlimited steps in wall-clock time, cut-off when the slowest baseline (FITC with 500 inducing inputs) finishes. The shaded area visualizes minimum and maximum over all baseline runs.

Acknowledgments

The authors thank Alexandra Gessner, Hans Kersting, Agustinus Kristiadi, Frederik Kunstner, Filip de Roos and Matthias Werner for helpful discussions and proof-reading. The authors gratefully acknowledge financial support by the Emmy Noether Programme of the German Research Union (DFG, Grant HE 7114/3-1); by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg.

Appendix A. Properties of the Symmetric Kronecker Product

The Kronecker product and its symmetric version have been studied, among others, by Loan (2000) and Magnus and Neudecker (1980). The definitions used in this work slightly differ from the authors above and instead follow Hennig (2015). The Kronecker product for two arbitrary matrices $\boldsymbol{A} \in \mathbb{R}^{N_1 \times N_2}$, $\boldsymbol{B} \in \mathbb{R}^{N_3 \times N_4}$ is defined as

$$[\boldsymbol{A} \otimes \boldsymbol{B}]_{ij,kl} \coloneqq \boldsymbol{A}_{ik}\boldsymbol{B}_{jl}$$

where $i \in \{1, ..., N_1\}$, $j \in \{1, ..., N_3\}$, $k \in \{1, ..., N_2\}$ and $l \in \{1, ..., N_4\}$, and ij is not a product but a double-index. The following identities about Kronecker products and the vectorization operator can be found in Hennig and Kiefel (2013), and are restated here for the convenience of the reader:

$$(\boldsymbol{A} \otimes \boldsymbol{B})\,\mathrm{vec}\,(\boldsymbol{C}) = \mathrm{vec}\,(\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^\intercal) \tag{K1}$$
$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C}) \otimes (\boldsymbol{B}\boldsymbol{D}) \tag{K2}$$
$$(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1} \tag{K3}$$
$$(\boldsymbol{A} \otimes \boldsymbol{B})^{\top} = \boldsymbol{A}^\intercal \otimes \boldsymbol{B}^\intercal \tag{K4}$$
$$(\boldsymbol{A} + \boldsymbol{B}) \otimes \boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{C} + \boldsymbol{B} \otimes \boldsymbol{C} \tag{K5}$$

where[11] $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D} \in \mathbb{R}^{N \times N}$, and $\boldsymbol{A}$ and $\boldsymbol{B}$ are assumed to be invertible.

An appealing property of Kronecker-structured matrices is their interaction with vectorized matrices. For a square matrix $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{a}_1 & \dots & \boldsymbol{a}_N\end{bmatrix}^\intercal \in \mathbb{R}^{N \times N}$, the *vectorization operator* $\mathrm{vec}(\,) : \mathbb{R}^{N \times N} \to \mathbb{R}^{N^2}$ stacks the rows[12] of $\boldsymbol{A}$ into one vector:

$$\mathrm{vec}\,(\boldsymbol{A}) = \begin{bmatrix}\boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_N\end{bmatrix}, \quad \text{with} \quad [\mathrm{vec}\,(\boldsymbol{A})]_{(ij)} = [\boldsymbol{A}]_{ij}$$

11. The conditions can be more general but for ease of exposition, we assume all matrices are square and of equal size.
12. Stacking the columns is equivalently possible and common. It is associated with a permutation in the definition of the Kronecker product, but the resulting inferences are equivalent.

and mat ( ) transforms an N^2 vector into an $N \times N$ matrix, s.t. $\text{mat}\,(\text{vec}\,(\boldsymbol{A})) = \boldsymbol{A}$. A vector product of vectorized matrices corresponds to the trace of their product:

$$\text{vec}\,(\boldsymbol{A})^\intercal \,\text{vec}\,(\boldsymbol{B}) = \text{tr}\,[\boldsymbol{A}\boldsymbol{B}^\intercal]\,. \tag{V1}$$

Proof

$$\begin{aligned}
\text{tr}\,[\boldsymbol{A}\boldsymbol{B}^\intercal] &= \sum_i [\boldsymbol{A}\boldsymbol{B}^\intercal]_{ii} \\
&= \sum_{i,j} \boldsymbol{A}_{ij}\boldsymbol{B}^\intercal_{ji} \\
&= \sum_{i,j} \boldsymbol{A}_{ij}\boldsymbol{B}_{ij} \\
&= \text{vec}\,(\boldsymbol{A})^\intercal \,\text{vec}\,(\boldsymbol{B})
\end{aligned}$$

■

The *symmetric* Kronecker product for two square matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{N\times N}$ of equal size is defined as

$$\boldsymbol{A}\otimes\boldsymbol{B} := \boldsymbol{\Gamma}_N(\boldsymbol{A}\otimes\boldsymbol{B})\boldsymbol{\Gamma}_N$$

where $[\boldsymbol{\Gamma}_N]_{ij,kl} := {}^1\!/_2\delta_{ik}\delta_{jl} + {}^1\!/_2\delta_{il}\delta_{jk}$ satisfies

$$\boldsymbol{\Gamma}\,\text{vec}\,(\boldsymbol{C}) = {}^1\!/_2\,\text{vec}\,(\boldsymbol{C}) + {}^1\!/_2\,\text{vec}\,(\boldsymbol{C}^\intercal)$$

for all square-matrices $\boldsymbol{C} \in \mathbb{R}^{N\times N}$.

Equivalently, one can write

$$(\boldsymbol{A}\otimes\boldsymbol{B})_{ij,kl} = \frac{1}{4}\left(\boldsymbol{A}_{ik}\boldsymbol{B}_{jl} + \boldsymbol{A}_{il}\boldsymbol{B}_{jk} + \boldsymbol{B}_{ik}\boldsymbol{A}_{jl} + \boldsymbol{B}_{il}\boldsymbol{A}_{jk}\right).$$

The symmetric Kronecker product inherits some of the desirable properties of the Kronecker product. Some of the following identities can, again, be found in Hennig (2015), some are due to Loan (2000) and Magnus and Neudecker (1980) and some are novel. The proof gives exact credit.

Proposition 6 *Let $\boldsymbol{V}, \boldsymbol{W} \in \mathbb{R}^{N\times N}$ be square matrices and $\boldsymbol{A}^\intercal, \boldsymbol{B} \in \mathbb{R}^{N\times M}$ be rectangular.*

$$\boldsymbol{W}\otimes\boldsymbol{W} = \boldsymbol{\Gamma}_N(\boldsymbol{W}\otimes\boldsymbol{W}) \tag{SK1}$$

$$\boldsymbol{\Gamma}_M(\boldsymbol{A}\otimes\boldsymbol{A}) = (\boldsymbol{A}\otimes\boldsymbol{A})\boldsymbol{\Gamma}_N \tag{SK2}$$

$$\boldsymbol{V}\otimes\boldsymbol{W} = \boldsymbol{W}\otimes\boldsymbol{V} \tag{SK3}$$

$$(\boldsymbol{A}\otimes\boldsymbol{A})(\boldsymbol{W}\otimes\boldsymbol{W})(\boldsymbol{B}\otimes\boldsymbol{B}) = (\boldsymbol{A}\boldsymbol{W}\boldsymbol{B})\otimes(\boldsymbol{A}\boldsymbol{W}\boldsymbol{B}) \tag{SK4}$$

$$\boldsymbol{W}\otimes\boldsymbol{W} - \boldsymbol{V}\otimes\boldsymbol{V} = (\boldsymbol{W}+\boldsymbol{V})\otimes(\boldsymbol{W}-\boldsymbol{V}) \tag{SK5}$$

$$(\boldsymbol{W}\otimes\boldsymbol{W})^{-1} = (\boldsymbol{W}^{-1}\otimes\boldsymbol{W}^{-1}). \tag{SK6}$$

The interpretation of Eq. (SK6) *requires some care: symmetric Kronecker product matrices are rank deficient. Eq.* (SK6) *is to be read in the sense that for symmetric $\boldsymbol{Y} \in \mathbb{R}^{N\times N}$,* i.e. *$\boldsymbol{Y} = \boldsymbol{Y}^\intercal$, $\boldsymbol{X} := \text{mat}\left((\boldsymbol{W}^{-1}\otimes\boldsymbol{W}^{-1})\,\text{vec}\,(\boldsymbol{Y})\right)$ satisfies $\text{vec}\,(\boldsymbol{Y}) = (\boldsymbol{W}\otimes\boldsymbol{W})\,\text{vec}\,(\boldsymbol{X})$ and $\boldsymbol{X}$ is the unique symmetric solution.*

Proof The proofs for Eqs. (SK1) and (SK2) can be found in Magnus and Neudecker (1999)[p. 46-50]. In the notation of Magnus and Neudecker (1999) $\boldsymbol{\Gamma} = N_n = D_n D_n^+$ and $K = 2\boldsymbol{\Gamma} - 2\boldsymbol{I}$. Eq. (SK1) is Theorem 13 (a). Eq. (SK2) follows from Theorem 9 (a).

To show $(\boldsymbol{W} \otimes\!\!\otimes \boldsymbol{V}) = (\boldsymbol{V} \otimes\!\!\otimes \boldsymbol{W})$, let $\boldsymbol{X} \in \mathbb{R}^{N \times N}$ be an arbitrary matrix.

$$\begin{aligned}
(\boldsymbol{V} \otimes\!\!\otimes \boldsymbol{W}) \operatorname{vec}(\boldsymbol{X}) &= \boldsymbol{\Gamma}(\boldsymbol{V} \otimes \boldsymbol{W})\boldsymbol{\Gamma} \operatorname{vec}(\boldsymbol{X}) \\
&= \frac{1}{2}\boldsymbol{\Gamma}(\boldsymbol{V} \otimes \boldsymbol{W}) \operatorname{vec}(\boldsymbol{X} + \boldsymbol{X}^\intercal) \\
&= \frac{1}{2}\boldsymbol{\Gamma} \operatorname{vec}(\boldsymbol{V}(\boldsymbol{X} + \boldsymbol{X}^\intercal)\boldsymbol{W}^\intercal) \\
&= \frac{1}{4} \operatorname{vec}(\boldsymbol{V}(\boldsymbol{X} + \boldsymbol{X}^\intercal)\boldsymbol{W}^\intercal + \boldsymbol{W}(\boldsymbol{X} + \boldsymbol{X}^\intercal)\boldsymbol{V}^\intercal) \\
&= \frac{1}{2}\boldsymbol{\Gamma} \operatorname{vec}(\boldsymbol{W}(\boldsymbol{X} + \boldsymbol{X}^\intercal)\boldsymbol{V}^\intercal) \\
&= \frac{1}{2}\boldsymbol{\Gamma}(\boldsymbol{W} \otimes \boldsymbol{V}) \operatorname{vec}(\boldsymbol{X} + \boldsymbol{X}^\intercal) \\
&= \boldsymbol{\Gamma}(\boldsymbol{W} \otimes \boldsymbol{V})\boldsymbol{\Gamma} \operatorname{vec}(\boldsymbol{X}) \\
&= (\boldsymbol{W} \otimes\!\!\otimes \boldsymbol{V}) \operatorname{vec}(\boldsymbol{X})
\end{aligned}$$

To show Eq. (SK4), use (SK2).

$$\begin{aligned}
(\boldsymbol{A} \otimes \boldsymbol{A})(\boldsymbol{W} \otimes\!\!\otimes \boldsymbol{W})(\boldsymbol{B} \otimes \boldsymbol{B}) &= (\boldsymbol{A} \otimes \boldsymbol{A})\boldsymbol{\Gamma}(\boldsymbol{W} \otimes \boldsymbol{W})\boldsymbol{\Gamma}(\boldsymbol{B} \otimes \boldsymbol{B}) \\
&= \boldsymbol{\Gamma}(\boldsymbol{A} \otimes \boldsymbol{A})(\boldsymbol{W} \otimes \boldsymbol{W})(\boldsymbol{B} \otimes \boldsymbol{B})\boldsymbol{\Gamma} \\
&= \boldsymbol{\Gamma}(\boldsymbol{A}\boldsymbol{W}\boldsymbol{B} \otimes \boldsymbol{A}\boldsymbol{W}\boldsymbol{B})\boldsymbol{\Gamma} \\
&= \boldsymbol{A}\boldsymbol{W}\boldsymbol{B} \otimes\!\!\otimes \boldsymbol{A}\boldsymbol{W}\boldsymbol{B}
\end{aligned}$$

The proof of Eq. (SK5) uses (SK3).

$$\begin{aligned}
(\boldsymbol{A} + \boldsymbol{B}) \otimes\!\!\otimes (\boldsymbol{A} - \boldsymbol{B}) &= \boldsymbol{\Gamma}(\boldsymbol{A} + \boldsymbol{B}) \otimes (\boldsymbol{A} - \boldsymbol{B})\boldsymbol{\Gamma} \\
&= \boldsymbol{\Gamma}(\boldsymbol{A} \otimes \boldsymbol{A} - \boldsymbol{A} \otimes \boldsymbol{B} + \boldsymbol{B} \otimes \boldsymbol{A} - \boldsymbol{B} \otimes \boldsymbol{B})\boldsymbol{\Gamma} \\
&= \boldsymbol{A} \otimes\!\!\otimes \boldsymbol{A} - \boldsymbol{A} \otimes\!\!\otimes \boldsymbol{B} + \boldsymbol{B} \otimes\!\!\otimes \boldsymbol{A} - \boldsymbol{B} \otimes\!\!\otimes \boldsymbol{B} \\
&= \boldsymbol{A} \otimes\!\!\otimes \boldsymbol{A} - \boldsymbol{B} \otimes\!\!\otimes \boldsymbol{A} + \boldsymbol{B} \otimes\!\!\otimes \boldsymbol{A} - \boldsymbol{B} \otimes\!\!\otimes \boldsymbol{B} \\
&= \boldsymbol{A} \otimes\!\!\otimes \boldsymbol{A} - \boldsymbol{B} \otimes\!\!\otimes \boldsymbol{B}
\end{aligned}$$

It remains to prove Eq. (SK6). Assume $\boldsymbol{Z}$ satisfies $(\boldsymbol{W} \otimes\!\!\otimes \boldsymbol{W}) \operatorname{vec}(\boldsymbol{Z}) = \operatorname{vec}(\boldsymbol{Y})$ and $\boldsymbol{Z} = \boldsymbol{Z}^\intercal$. Then,

$$\begin{aligned}
\operatorname{vec}(\boldsymbol{Y}) &= (\boldsymbol{W} \otimes\!\!\otimes \boldsymbol{W}) \operatorname{vec}(\boldsymbol{Z}) \\
&= (\boldsymbol{W} \otimes \boldsymbol{W})\boldsymbol{\Gamma}_N \operatorname{vec}(\boldsymbol{Z}) \quad \text{using Eq. (SK1) and Eq. (SK2)} \\
&= (\boldsymbol{W} \otimes \boldsymbol{W}) \operatorname{vec}(\boldsymbol{Z}) \quad \text{since } \boldsymbol{Z} = \boldsymbol{Z}^\intercal
\end{aligned}$$

and hence, $\boldsymbol{Z} = (\boldsymbol{W} \otimes \boldsymbol{W})^{-1} \operatorname{vec}(\boldsymbol{Y})$. Using Eq. (K3) and again Eq. (SK1),

$$\boldsymbol{Z} = (\boldsymbol{W}^{-1} \otimes\!\!\otimes \boldsymbol{W}^{-1}) \operatorname{vec}(\boldsymbol{Y})$$

which is the definition of $\boldsymbol{X}$. ■

A.1. Sampling from a Gaussian with Symmetric Kronecker Covariance matrix

To sample matrices from the KMCG posterior (Eq. 13) the following proposition will be useful.

Proposition 7 *Let $\boldsymbol{W}, \boldsymbol{W}_M \in \mathbb{R}^{N\times N}$ be symmetric and positive semi-definite matrices s.t. $\boldsymbol{W} - \boldsymbol{W}_M$ is symmetric positive-semidefinite as well. Further let $\operatorname{vec}(\boldsymbol{Y}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{W}\otimes\boldsymbol{W} - \boldsymbol{W}_M\otimes\boldsymbol{W}_M)$, denote with $\boldsymbol{L}_+$ the Cholesky of $\boldsymbol{W} + \boldsymbol{W}_M$, with $\boldsymbol{L}_-$ the Cholesky of $\boldsymbol{W} - \boldsymbol{W}_M$ and let $\operatorname{vec}(\boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{N^2})$, then $\boldsymbol{\Gamma}(\boldsymbol{L}_1 \otimes \boldsymbol{L}_2)\operatorname{vec}(\boldsymbol{X})$ and $\operatorname{vec}(\boldsymbol{Y})$ have the same distribution.*

Remark: This shows that $\boldsymbol{Y}$ is symmetric due to the $\boldsymbol{\Gamma}$-operator.

Proof As $\operatorname{vec}(\boldsymbol{X})$ is standard normal, $\boldsymbol{\Gamma}(\boldsymbol{L}_+ \otimes \boldsymbol{L}_-)\operatorname{vec}(\boldsymbol{X})$ is distributed Gaussian with mean $\boldsymbol{0}$ and covariance matrix $\boldsymbol{\Gamma}(\boldsymbol{L}_+ \otimes \boldsymbol{L}_-)(\boldsymbol{\Gamma}(\boldsymbol{L}_+ \otimes \boldsymbol{L}_-))^\intercal$.

$$\begin{aligned}\boldsymbol{\Gamma}(\boldsymbol{L}_+ \otimes \boldsymbol{L}_-)\left[\boldsymbol{\Gamma}(\boldsymbol{L}_+ \otimes \boldsymbol{L}_-)\right]^\intercal &= \boldsymbol{\Gamma}(\boldsymbol{L}_+ \otimes \boldsymbol{L}_-)(\boldsymbol{L}_+^\intercal \otimes \boldsymbol{L}_-^\intercal)\boldsymbol{\Gamma} \\ &= (\boldsymbol{L}_+\boldsymbol{L}_+^\intercal)\otimes(\boldsymbol{L}_-\boldsymbol{L}_-^\intercal) \\ &= (\boldsymbol{W} + \boldsymbol{W}_M)\otimes(\boldsymbol{W} - \boldsymbol{W}_M)\end{aligned}$$

According to Equation (SK5): $(\boldsymbol{W} + \boldsymbol{W}_M)\otimes(\boldsymbol{W} - \boldsymbol{W}_M) = \boldsymbol{W}\otimes\boldsymbol{W} - \boldsymbol{W}_M\otimes\boldsymbol{W}_M$. ■

Appendix B. Inducing Input Methods

This section contains the proof of Proposition 1, introduced on page 7, and, for the readers convenience, restated below, along with the referenced equations.

Proposition 8 (Subset of Regressors) *Consider the prior of Eq. (9) with $k_0 := 0$ and $w := k$ and the likelihood defined in Example 1 with $s_{ij} = \delta_{ij}$. Then the posterior mean k_M is equivalent to that of SoR:*

$$k_M(\boldsymbol{x}, \boldsymbol{z}) = k_{SoR} = k(\boldsymbol{x}, \boldsymbol{X}_U)k(\boldsymbol{X}_U, \boldsymbol{X}_U)^{-1}k(\boldsymbol{X}_U, \boldsymbol{z})$$

where $\boldsymbol{X}_U$ are inducing inputs, not necessarily part of $\boldsymbol{X}$.

The mentioned equations are

$$k \sim \mathcal{GP}(k_0, \gamma\psi), \tag{9}$$

$$\psi(k(\boldsymbol{a}, \boldsymbol{b}), k(\boldsymbol{c}, \boldsymbol{d})) := \frac{1}{2}w(\boldsymbol{a}, \boldsymbol{c})w(\boldsymbol{b}, \boldsymbol{d}) + \frac{1}{2}w(\boldsymbol{a}, \boldsymbol{d})w(\boldsymbol{b}, \boldsymbol{c}), \tag{10}$$

$$\boldsymbol{T}_{\boldsymbol{p}} : (\mathbb{X}\times\mathbb{X})^{\mathbb{R}} \to \mathbb{R}^{P^2}, k \mapsto \operatorname{vec}\left(\left[\iint k(\boldsymbol{x}, \boldsymbol{z})p_i(\boldsymbol{x})p_j(\boldsymbol{z})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{z}\right]_{ij}\right), \tag{11}$$

$$\text{and } p_i(\boldsymbol{x}) := \sum_{j=1}^{M} s_{ij}\delta(\boldsymbol{x} - \boldsymbol{x}_{u_j}). \tag{12}$$

Proposition 1 follows from the more general Proposition 9, below.

Proposition 9 *Consider the prior of Eq.* (9) *(without the restriction $w = k$) and the likelihood defined in Example 1. The posterior over k is $p(k \mid \boldsymbol{Y} = \boldsymbol{T_p} k) = \mathcal{N}(k; k_M, \psi_M)$ with posterior mean*

$$\begin{aligned} k_M(\boldsymbol{a}, \boldsymbol{b}) = k_0(\boldsymbol{a}, \boldsymbol{b}) &+ w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{b}) \\ &- w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal k_0(\boldsymbol{X}_U, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{b}) \end{aligned} \tag{20}$$

and posterior variance

$$\begin{aligned} \psi_M(k(\boldsymbol{a},\boldsymbol{b}), k(\boldsymbol{c}, \boldsymbol{d})) = &\frac{1}{2} w(\boldsymbol{a}, \boldsymbol{c}) w(\boldsymbol{b}, \boldsymbol{d}) + \frac{1}{2} w(\boldsymbol{a}, \boldsymbol{d}) w(\boldsymbol{b}, \boldsymbol{c}) \\ &- \frac{1}{2} w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{c}) w(\boldsymbol{b}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{d}) \\ &- \frac{1}{2} w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{d}) w(\boldsymbol{b}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{c}) \end{aligned} \tag{21}$$

where $\boldsymbol{W}_M = w(\boldsymbol{X}_U, \boldsymbol{X}_U)$.

Proof The proof is tedious linear algebra. If prior and likelihood are Gaussian, so is the posterior with mean

$$k_M(\boldsymbol{a}, \boldsymbol{b}) = k_0(\boldsymbol{a}, \boldsymbol{b}) - (\boldsymbol{T_p}\psi(k(\boldsymbol{a}, \boldsymbol{b}), \cdot))^\intercal (\boldsymbol{T_p}(\boldsymbol{T_p} w)^\intercal)^{-1} \operatorname{vec}\left(\boldsymbol{Y} - \boldsymbol{S}^\intercal k_0(\boldsymbol{X}_U, \boldsymbol{X}_U)\boldsymbol{S}\right),$$

and variance

$$\psi_M(k(\boldsymbol{a}, \boldsymbol{b}), k(\boldsymbol{c}, \boldsymbol{d})) = \psi((\boldsymbol{a}, \boldsymbol{b}), (\boldsymbol{c}, \boldsymbol{d})) - (\boldsymbol{T_p}\psi((\boldsymbol{a}, \boldsymbol{b}), (\cdot, \cdot)))^\intercal (\boldsymbol{T_p}(\boldsymbol{T_p}\psi)^\intercal)^{-1} \boldsymbol{T_p}\psi(\boldsymbol{c}, \boldsymbol{d}), (\cdot, \cdot)).$$

With Lemma 10 and Eq. (SK6), we can write

$$\begin{aligned} &(\boldsymbol{T_p}\psi(k(\boldsymbol{a}, \boldsymbol{b}), \cdot))^\intercal (\boldsymbol{T_p}(\boldsymbol{T_p} w)^\intercal)^{-1} \\ &= \frac{1}{2} \operatorname{vec}\left(\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{a}) w(\boldsymbol{b}, \boldsymbol{X}_U) + w(\boldsymbol{X}_U, \boldsymbol{b}) w(\boldsymbol{a}, \boldsymbol{X}_U))\boldsymbol{S}\right)^\intercal \left((\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1} \otimes (\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\right) \\ &= \frac{1}{2} \operatorname{vec}\left((\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{a}) w(\boldsymbol{b}, \boldsymbol{X}_U) + w(\boldsymbol{X}_U, \boldsymbol{b}) w(\boldsymbol{a}, \boldsymbol{X}_U))\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\right)^\intercal \end{aligned}$$

and, using Eq. (V1), obtain for Eq. (20):

$$\begin{aligned} k_M(\boldsymbol{a}, \boldsymbol{b}) &= k_0(\boldsymbol{a}, \boldsymbol{b}) \\ &\quad + \frac{1}{2} \operatorname{tr}\left[(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{a}) w(\boldsymbol{b}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}(\boldsymbol{Y} - k_0(\boldsymbol{X}_U, \boldsymbol{X}_U))\right] \\ &\quad + \frac{1}{2} \operatorname{tr}\left[(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{b}) w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}(\boldsymbol{Y} - k_0(\boldsymbol{X}_U, \boldsymbol{X}_U))\right] \\ &\quad = k_0(\boldsymbol{a}, \boldsymbol{b}) + w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{b}) \\ &\quad - w(\boldsymbol{a}, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal k_0(\boldsymbol{X}_U, \boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U, \boldsymbol{b}) \end{aligned}$$

The derivation for Eq. (21) follows analogously.

■

Lemma 10 *Let $\boldsymbol{T_p}$ be defined by Eq.* (12)*.*

$$\boldsymbol{T_p} w(k(\boldsymbol{a},\boldsymbol{b}),\cdot) = \frac{1}{2}\operatorname{vec}\left(\boldsymbol{S}^\intercal\left(w(\boldsymbol{X}_U,\boldsymbol{a})w(\boldsymbol{b},\boldsymbol{X}_U)+w(\boldsymbol{X}_U,\boldsymbol{b})w(\boldsymbol{a},\boldsymbol{X}_U)\right)\boldsymbol{S}\right) \tag{22}$$

$$\boldsymbol{T_p}(\boldsymbol{T_p} w(\cdot,\cdot))^\intercal = (\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S}) \circledast (\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S}) \tag{23}$$

Proof Denote with mat ( ) the complement of the vectorization operator, *i.e.* $\operatorname{mat}(\operatorname{vec}(\boldsymbol{A})) = \boldsymbol{A}$. Define the matrix $\boldsymbol{S} \in \mathbb{R}^{N\times M}$ as $\boldsymbol{S}_{ij} = s_{ij}$ and denote with $\boldsymbol{S}_l$ the l-th column of $\boldsymbol{S}$. Also recall that by Eq. (10) $\psi(k(\boldsymbol{a},\boldsymbol{b}),k(\boldsymbol{x},\boldsymbol{z})) = \frac{1}{2}(w(\boldsymbol{a},\boldsymbol{x})w(\boldsymbol{b},\boldsymbol{z})+w(\boldsymbol{a},\boldsymbol{z})w(\boldsymbol{b},\boldsymbol{x}))$.

$$\begin{aligned}
&[\operatorname{mat}(\boldsymbol{T_p}[\psi(k(\boldsymbol{a},\boldsymbol{b}),k(\cdot,\cdot))])]_{ij} \\
&= \iint \psi(k(\boldsymbol{a},\boldsymbol{b}),k(\boldsymbol{x},\boldsymbol{z}))\left(\sum_{l=1}^M s_{il}\delta(\boldsymbol{x}-\boldsymbol{u}_l)\right)\left(\sum_{l=1}^M s_{jl}\delta(\boldsymbol{z}-\boldsymbol{u}_l)\right)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{z} \\
&= \iint \frac{1}{2}(w(\boldsymbol{a},\boldsymbol{x})w(\boldsymbol{b},\boldsymbol{z})+w(\boldsymbol{a},\boldsymbol{z})w(\boldsymbol{b},\boldsymbol{x}))\left(\sum_{l=1}^M s_{il}\delta(\boldsymbol{x}-\boldsymbol{u}_l)\right)\left(\sum_{l=1}^M s_{jl}\delta(\boldsymbol{z}-\boldsymbol{u}_l)\right)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{z} \\
&= \frac{1}{2}\sum_{m=1}^M\sum_{l=1}^M \boldsymbol{S}_{im}\boldsymbol{S}_{jl}(w(\boldsymbol{a},\boldsymbol{u}_m)w(\boldsymbol{b},\boldsymbol{u}_l)+w(\boldsymbol{a},\boldsymbol{u}_l)w(\boldsymbol{b},\boldsymbol{u}_m)) \\
&= \frac{1}{2}\left[\boldsymbol{S}^\intercal w(\boldsymbol{X}_U,\boldsymbol{a})w(\boldsymbol{b},\boldsymbol{X}_U)\boldsymbol{S}+\boldsymbol{S}^\intercal w(\boldsymbol{X}_U,\boldsymbol{b})w(\boldsymbol{a},\boldsymbol{X}_U)\boldsymbol{S}\right]_{ij} \\
&= \frac{1}{2}\left[\boldsymbol{S}^\intercal(w(\boldsymbol{X}_U,\boldsymbol{a})w(\boldsymbol{b},\boldsymbol{X}_U)+w(\boldsymbol{X}_U,\boldsymbol{b})w(\boldsymbol{a},\boldsymbol{X}_U))\boldsymbol{S}\right]_{ij}
\end{aligned}$$

which shows Eq. (22)

$$\begin{aligned}
&= [\operatorname{mat}((\boldsymbol{S}^\intercal\otimes\boldsymbol{S}^\intercal)\boldsymbol{\Gamma}\operatorname{vec}(w(\boldsymbol{X}_U,\boldsymbol{a})w(\boldsymbol{b},\boldsymbol{X}_U)))]_{ij} \\
&= [\operatorname{mat}((\boldsymbol{S}^\intercal\otimes\boldsymbol{S}^\intercal)\boldsymbol{\Gamma}(w(\boldsymbol{X}_U,\boldsymbol{a})\otimes w(\boldsymbol{X}_U,\boldsymbol{b})))]_{ij}
\end{aligned}$$

Repeating above derivations shows the second statement, Eq. (23):

$$\begin{aligned}
\boldsymbol{T_p}(\boldsymbol{T_p}\psi)^\intercal &= (\boldsymbol{S}^\intercal\otimes\boldsymbol{S}^\intercal)\boldsymbol{\Gamma}(w(\boldsymbol{X}_U,\boldsymbol{X}_U)\otimes w(\boldsymbol{X}_U,\boldsymbol{X}_U))\boldsymbol{\Gamma}(\boldsymbol{S}\otimes\boldsymbol{S}) \\
&= (\boldsymbol{S}\otimes\boldsymbol{S})^\intercal(w(\boldsymbol{X}_U,\boldsymbol{X}_U)\circledast w(\boldsymbol{X}_U,\boldsymbol{X}_U))(\boldsymbol{S}\otimes\boldsymbol{S}) \\
&= (\boldsymbol{S}^\intercal\boldsymbol{W}_M\boldsymbol{S})\circledast(\boldsymbol{S}^\intercal\boldsymbol{W}_M\boldsymbol{S}) \qquad \text{Equation (SK4)}
\end{aligned}$$

■

Proposition 11 *If $k_0 = 0$, $\boldsymbol{S}$ has rank M, and k and w are positive definite kernel functions then the posterior mean in Eq.* (20) *is symmetric and positive semi-definite.*

Proof

With $k_0 = 0$ the expression for k_M from Proposition 9 simplifies to

$$k_M(\boldsymbol{x},\boldsymbol{z}) = w(\boldsymbol{x},\boldsymbol{X}_U)\boldsymbol{S}(\boldsymbol{S}^\intercal\boldsymbol{W}_M\boldsymbol{S})^{-1}\boldsymbol{S}^\intercal\boldsymbol{K}_M\boldsymbol{S}(\boldsymbol{S}^\intercal\boldsymbol{W}_M\boldsymbol{S})^{-1}\boldsymbol{S}^\intercal w(\boldsymbol{X}_U,\boldsymbol{z}).$$

The function k_M is symmetric since k is symmetric. The bivariate function k_M is said to be positive (semi-)definite iff for all $n \in \mathbb{N}$ and for all $\boldsymbol{Z} \in \mathbb{X}$, $k_M(\boldsymbol{Z}, \boldsymbol{Z})$ is a positive (semi-)definite matrix. Since $k(\boldsymbol{X}_U, \boldsymbol{X}_U)$ is a symmetric and positive definite (s.p.d.) matrix, so is $\boldsymbol{S}^\intercal k(\boldsymbol{X}_U, \boldsymbol{X}_U)\boldsymbol{S}$ for arbitrary $\boldsymbol{S}$. The same argument holds for $\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S}$. Since $\boldsymbol{S}$ is rank M, $(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1}$ exists and the inverse of an s.p.d. matrix is s.p.d. as well. Therefore $\boldsymbol{S}(\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1} \boldsymbol{S}^\intercal \boldsymbol{K}_M \boldsymbol{S} (\boldsymbol{S}^\intercal \boldsymbol{W}_M \boldsymbol{S})^{-1} \boldsymbol{S}$ is symmetric and positive semi-definite. This completes the proof.

■

Appendix C. Projected Bayes Regressor

This section contains the proof of Proposition 2 restated below.

Proposition 12 (Projected Bayes Regressor) *Consider the prior of Eq.* (9) *with* $k_0 := 0$ *and* $w := k$ *and the likelihood defined in Example 2. Let* λ_1 *to* λ_P *be the largest eigenvalues of the kernel* k *(w.r.t to the Mercer expansion) and assume the inputs* $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ *are independent and identical draws from* ν*. Then the posterior kernel* k_M *leads to the* Projected Bayes Regressor *(Trecate et al., 1999).*

Proof Given Lemma 13 below, all that remains is to substitute k_M in Eq. (1) which evaluates to

$$\phi(\boldsymbol{x}_*)^\intercal (\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal + \sigma^2 \boldsymbol{\Lambda}^{-1}) \boldsymbol{\Phi}^\intercal \boldsymbol{y}. \tag{24}$$

Comparing $b(\boldsymbol{x})$ in Definition 1 in Trecate et al. (1999) and Eq. (24) one observes that both are equivalent. ■

Lemma 13 *Let* ϕ_i $i = 1, ..., P$ *be orthogonal Eigenfunctions of* k *with respect to a density* ν *on* $\mathbb{X}$*,* i.e.

$$\int k(\boldsymbol{x}, \boldsymbol{z}) \phi_i(\boldsymbol{z}) \nu(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} = \lambda_i \phi_i(\boldsymbol{x})$$
$$\int \phi_i(\boldsymbol{z}) \phi_j(\boldsymbol{z}) \nu(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} = \delta_{ij}$$

where $\lambda_i \in \mathbb{R}$ *and* δ_{ij} *is the Kronecker delta. Under the prior of Eq.* (9) *with* $k_0 := 0$ *and* $w := k$ *and the likelihood defined in Example 2 with* $p_i(\boldsymbol{x}) = \phi_i(\boldsymbol{x})\nu(\boldsymbol{x})$*, the approximate kernel (Eq.* (13)*) evaluates to*

$$k_M(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{M} \lambda_j \phi_j(\boldsymbol{x}) \phi_j(\boldsymbol{z}) = \boldsymbol{\phi}(\boldsymbol{x})^\intercal \boldsymbol{\Lambda} \boldsymbol{\phi}(\boldsymbol{z})$$

where $[\boldsymbol{\phi}(\boldsymbol{x})]_i = \phi_i(\boldsymbol{x})$ *and* $\boldsymbol{\Lambda}_{ij} := \delta_{ij} \lambda i$*.*

Proof With a zero prior-mean, the posterior over k (Eq. (13)) simplifies to

$$k_M(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{T_p}\psi(k(\boldsymbol{x}, \boldsymbol{z}), \cdot)^\intercal (\boldsymbol{T_p}(\boldsymbol{T_p}\psi)^\intercal)^{-1} \boldsymbol{T_p} k.$$

Differing from the proof of Proposition 9 the observation operator $\boldsymbol{T_p}$ (Eq. 11) is of the form:

$$\begin{aligned}
[\mathrm{mat}\,(\boldsymbol{T_p}k)]_{ij} &= \iint k(\boldsymbol{x}, \boldsymbol{z})\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{z})\ \nu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}\boldsymbol{z}) \\
&= \lambda_i \int \phi_i(\boldsymbol{z})\phi_j(\boldsymbol{z})\ \nu(\mathrm{d}\boldsymbol{z}) \\
&= \lambda_i \delta_{ij} \\
&= \boldsymbol{\Lambda}_{ij}.
\end{aligned}$$

The observation operator $\boldsymbol{T_p}$ applied to the covariance function w evaluates to:

$$\begin{aligned}
[\mathrm{mat}\,(\boldsymbol{T_p}\psi(k(\boldsymbol{a}, \boldsymbol{b}), k(\cdot, \cdot\cdot)))]_{ij} &= \left[\mathrm{mat}\left(\boldsymbol{T_p}\left[\frac{1}{2}k(\boldsymbol{a}, \cdot)k(\boldsymbol{b}, \cdot\cdot) + \frac{1}{2}k(\boldsymbol{a}, \cdot\cdot)k(\boldsymbol{b}, \cdot)\right]\right)\right]_{ij} \\
&= \frac{1}{2}\iint k(\boldsymbol{a}, \boldsymbol{x})k(\boldsymbol{b}, \boldsymbol{z})\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{z})\nu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}\boldsymbol{z}) \\
&+ \frac{1}{2}\iint k(\boldsymbol{a}, \boldsymbol{x})k(\boldsymbol{b}, \boldsymbol{z})\phi_j(\boldsymbol{x})\phi_i(\boldsymbol{z})\nu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}\boldsymbol{z}) \\
&= \frac{1}{2}\lambda_i\lambda_j(\phi_i(\boldsymbol{a})\phi_j(\boldsymbol{b}) + \phi_i(\boldsymbol{b})\phi_j(\boldsymbol{a})) \qquad (25) \\
&= \frac{1}{2}[\boldsymbol{\Lambda}(\boldsymbol{\phi}(\boldsymbol{a})\boldsymbol{\phi}(\boldsymbol{b})^\intercal + \boldsymbol{\phi}(\boldsymbol{b})\boldsymbol{\phi}(\boldsymbol{a})^\intercal)\boldsymbol{\Lambda}]_{ij}.
\end{aligned}$$

Applying $\boldsymbol{T_p}$ again, leads to

$$\begin{aligned}
[\boldsymbol{T_p}(\boldsymbol{T_p}\psi)^\intercal]_{ij,gh} &= \iint [\boldsymbol{T_p}\psi(k(\boldsymbol{x}, \boldsymbol{z}), k(\cdot, \cdot\cdot)]^\intercal_{gh}\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{z})\ \nu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}\boldsymbol{z}) \\
&\quad \text{using Equation (25)} \\
&= \frac{1}{2}\lambda_g\lambda_h \iint (\phi_g(\boldsymbol{x})\phi_h(\boldsymbol{z}) + \phi_g(\boldsymbol{z})\phi_h(\boldsymbol{x}))\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{z})\ \nu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}(\boldsymbol{z}) \\
&= \frac{1}{2}\lambda_g\lambda_h \int (\delta_{ig}\phi_h(\boldsymbol{z}) + \delta_{ih}\phi_g(\boldsymbol{z}))\phi_j(\boldsymbol{z})\ \nu(\mathrm{d}(\boldsymbol{z}) \\
&= \frac{1}{2}\lambda_g\lambda_h(\delta_{ig}\delta_{jh} + \delta_{ih}\delta_{jg}) \\
&= [\boldsymbol{\Lambda}\otimes\boldsymbol{\Lambda}]_{ij,gh}
\end{aligned}$$

where the last equation follows from the definition of the symmetric Kronecker product. This implies for the posterior mean over the kernel:

$$
\begin{aligned}
k_M(\boldsymbol{a}, \boldsymbol{b}) &= (\boldsymbol{T_p}\psi(k(\boldsymbol{a}, \boldsymbol{b}), \cdot)^\intercal (\boldsymbol{T_p}(\boldsymbol{T_p}\psi)^\intercal)^{-1} \boldsymbol{T_p} k \\
&= \frac{1}{2} \operatorname{vec}\left(\boldsymbol{\Lambda}(\boldsymbol{\phi}(\boldsymbol{a})\boldsymbol{\phi}(\boldsymbol{b})^\intercal + \boldsymbol{\phi}(\boldsymbol{b})\boldsymbol{\phi}(\boldsymbol{a})^\intercal)\boldsymbol{\Lambda}\right)^\intercal (\boldsymbol{\Lambda} \otimes \boldsymbol{\Lambda})^{-1} \operatorname{vec}(\boldsymbol{\Lambda}) \\
&= \frac{1}{2} \operatorname{vec}\left((\boldsymbol{\phi}(\boldsymbol{a})\boldsymbol{\phi}(\boldsymbol{b})^\intercal + \boldsymbol{\phi}(\boldsymbol{b})\boldsymbol{\phi}(\boldsymbol{a})^\intercal)\right)^\intercal \operatorname{vec}(\boldsymbol{\Lambda}) \\
&\text{applying Equation (V1):} \\
&= \frac{1}{2} \operatorname{tr}\left[\boldsymbol{\Lambda}(\boldsymbol{\phi}(\boldsymbol{a})\boldsymbol{\phi}(\boldsymbol{b})^\intercal + \boldsymbol{\phi}(\boldsymbol{b})\boldsymbol{\phi}(\boldsymbol{a})^\intercal)\right] \\
&= \boldsymbol{\phi}(\boldsymbol{a})^\intercal \boldsymbol{\Lambda} \boldsymbol{\phi}(\boldsymbol{b}).
\end{aligned}
$$

■

Appendix D. Benchmark data sets

Table 1 describes the purposes and origins of standard benchmark data sets used for Gaussian process regression. More information on PRECIPITATION can be found at `http://www.image.ucar.edu/Data/US.monthly.met/`. It appears that the data sets AILERONS, ELEVATORS and POLETELECOMM are no longer available under the link `https://www.dcc.fc.up.pt/~ltorgo/Regression/datasets.html`. However, all files are part of this submission.

Appendix E. Additional Experiments and Results

This section consists of figures showing the results of Section 4.1 for the Matérn kernel, real-time experiments and experiments with the textbook version of conjugate gradients.

E.1. Real-time Results

This section shows the same results as in Section 4.1 but over training-time instead of CG-steps. All figures have been trimmed to the slowest baseline method. Fig. 9 shows how the relative error ε_f develops over time for the squared exponential kernel and Fig. 10 shows the same for experiments over grid-structured data sets from Section 4.2. For the x-axis values we took the median of all measurements and fitted a quadratic function to these.

E.2. Matérn Kernel Results

The figures in this section show the results for the Matérn $5/2$ kernel (Eq. (19)) for the experiment setup described Section 4.1. Fig. 11 shows the results for the relative error ε_f, Fig. 12 and Fig. 13 the results for ε_{var} and ε_{ev}, respectively. Fig. 14 displays the relative error over time.

ABALONE Nash et al. (1994); Waugh (1995); Dua and Graff (2019)
age prediction of abalone from physical measurements
`https://archive.ics.uci.edu/ml/datasets/Abalone`

AILERONS Camachol (1998)
control action prediction on the ailerons of an F16 aircraft

ELEVATORS Camachol (1998)
control action prediction on the elevators of an F16 aircraft

MPG Quinlan (1993); Dua and Graff (2019)
fuel consumption prediction in miles per gallon for different attributes of cars
`https://archive.ics.uci.edu/ml/datasets/auto+mpg`

POLETELECOMM Weiss and Indurkhya (1995)
commercial telecommunication application–no further information

PRECIPITATION Vanhatalo and Vehtari (2008)
US annual precipitation prediction for the year 1995
`https://github.com/gpstuff-dev/gpstuff/blob/master/gp/demodata/USprec1.txt`

PUMADYN Snelson and Ghahramani (2006)
acceleration prediction one of the arm links given angles, positions and velocities of other links of a *Puma560* robot
`ftp://ftp.cs.toronto.edu/pub/neuron/delve/data/tarfiles/pumadyn-family/pumadyn-32nm.tar.gz`

SOUND Turner (2010); Wilson and Nickisch (2015)
sound intensity prediction of a signal recorded over time for missing regions
`https://github.com/kd383/GPML_SLD/blob/master/demo/sound/audio_data.mat`

TOY introduced in this work
targets are a draw from a zero-mean Gaussian process with squared exponential kernel (Eq. (18) with $\mathbf{\Lambda} = 0.25$ and $\theta_f = 2$), inputs stem in equal parts from a Gaussian mixture ($\mathcal{N}(0,1) + \mathcal{N}(1,0.1) + \mathcal{N}(-0.5,0.05)$) and the uniform distribution over $[0,1]$

Table 1: descriptions and sources for all data sets considered in this work.

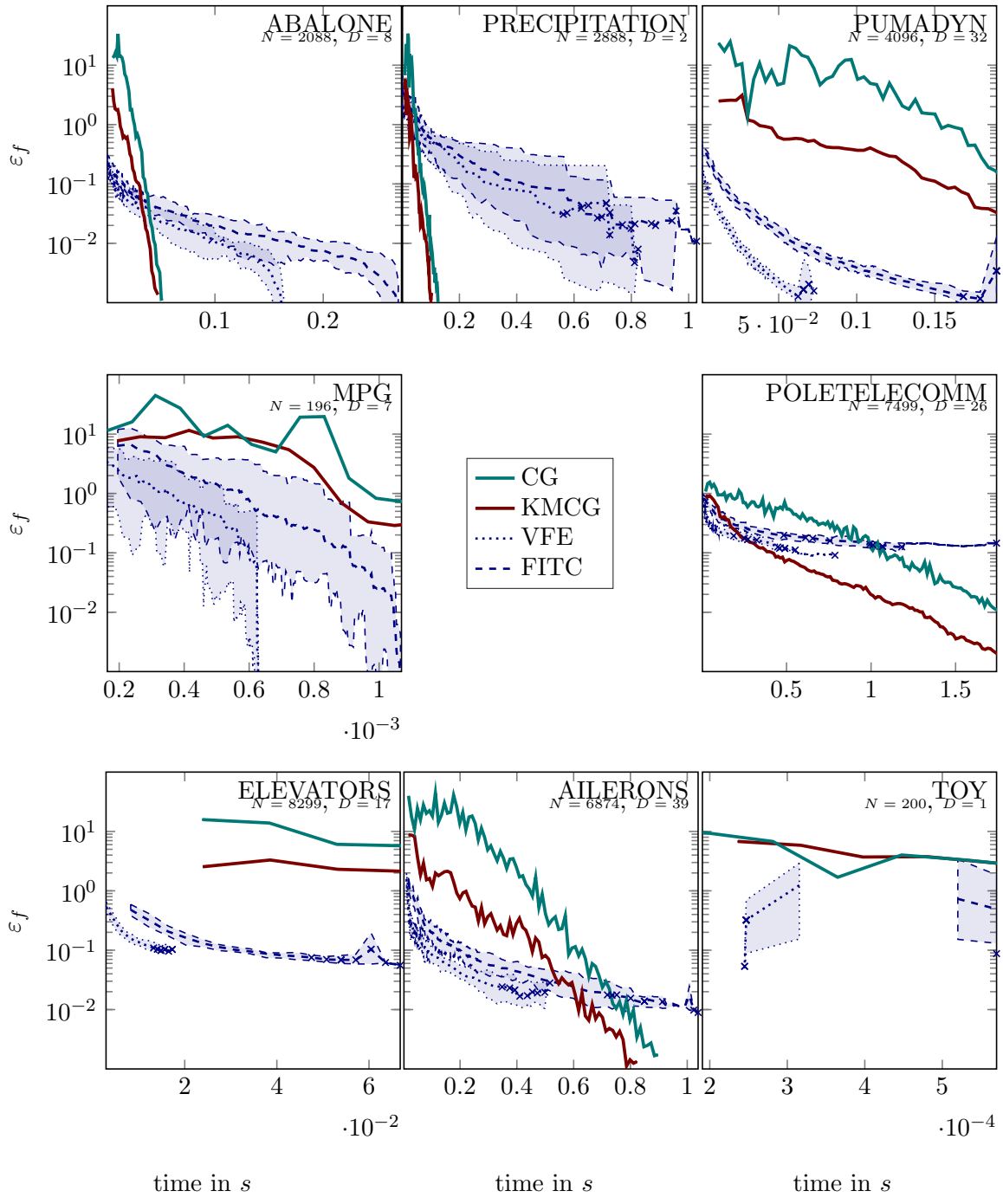


Figure 9: progression of the relative error ε_f over training time for different data sets using the squared-exponential kernel (Eq. 18). The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

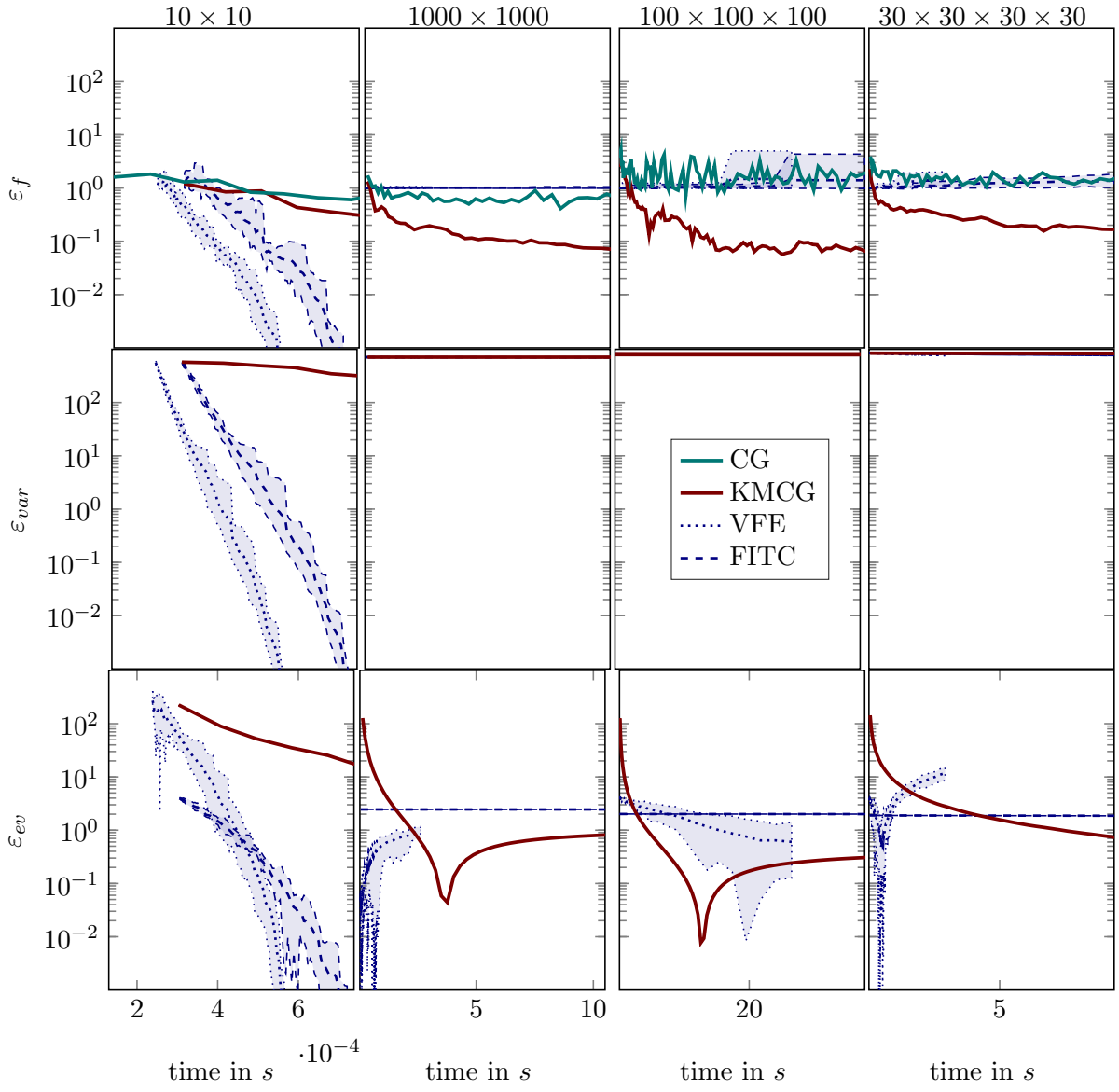


Figure 10: progression of the relative error ε_f over training time for different data sets using the squared-exponential kernel (Eq. 18). The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run. It may seem surprising that the runs on the $100 \times 100 \times 100$ data set take more than twice as long. By chance, the data set contains more extreme values in the kernel matrix, *i.e.* smaller than $1e^{-50}$. Multiplication with these elements takes more time.

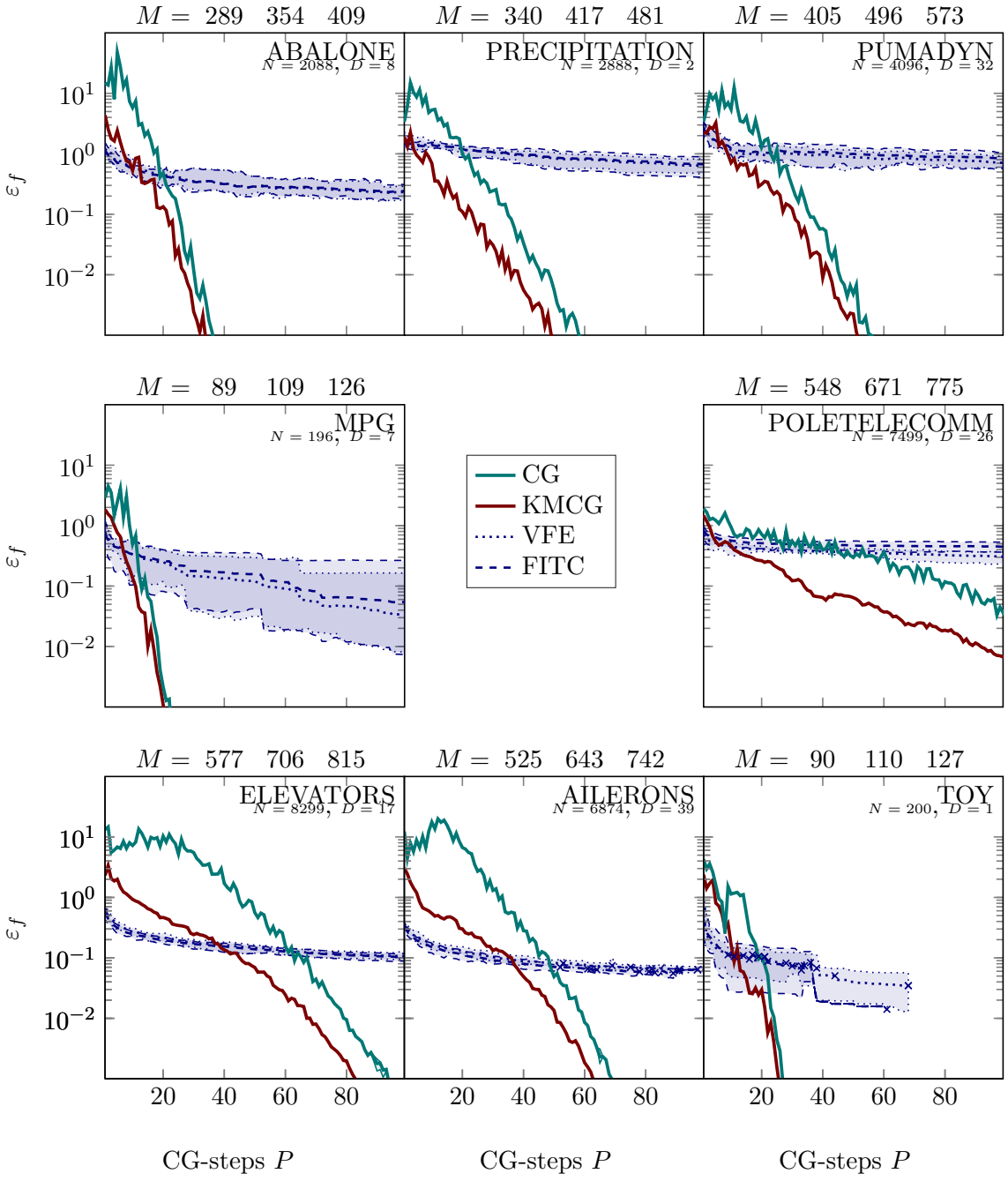


Figure 11: progression of the relative error ε_f as a function of the number of iterations of baseline and KMCG for different data sets using the Matérn kernel (Eq. 19). The bottom axis is the number of CG-steps, which is the same for all plots. Therefore, this axis is visible only in the last row. The top axis denotes the number of inducing inputs used by the baseline methods. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

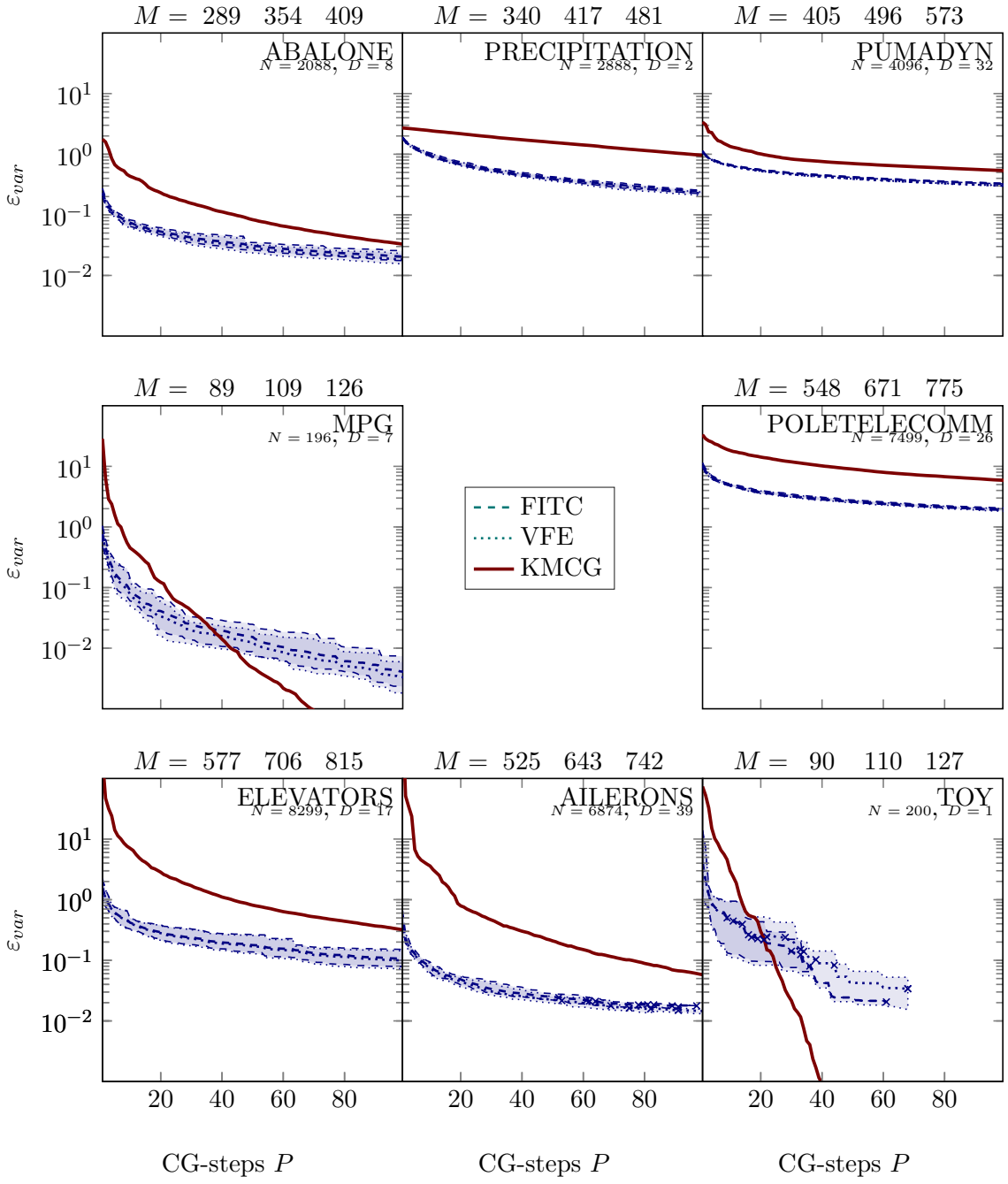


Figure 12: progression of the relative error of the variance ε_{var} as a function of the number of iterations of baseline and KMCG for different data sets using the Matérn kernel (Eq. 19). The bottom axis is the number of CG-steps, which is the same for all plots. Therefore, this axis is visible only in the last row. The top axis denotes the number of inducing inputs used by the baseline methods. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

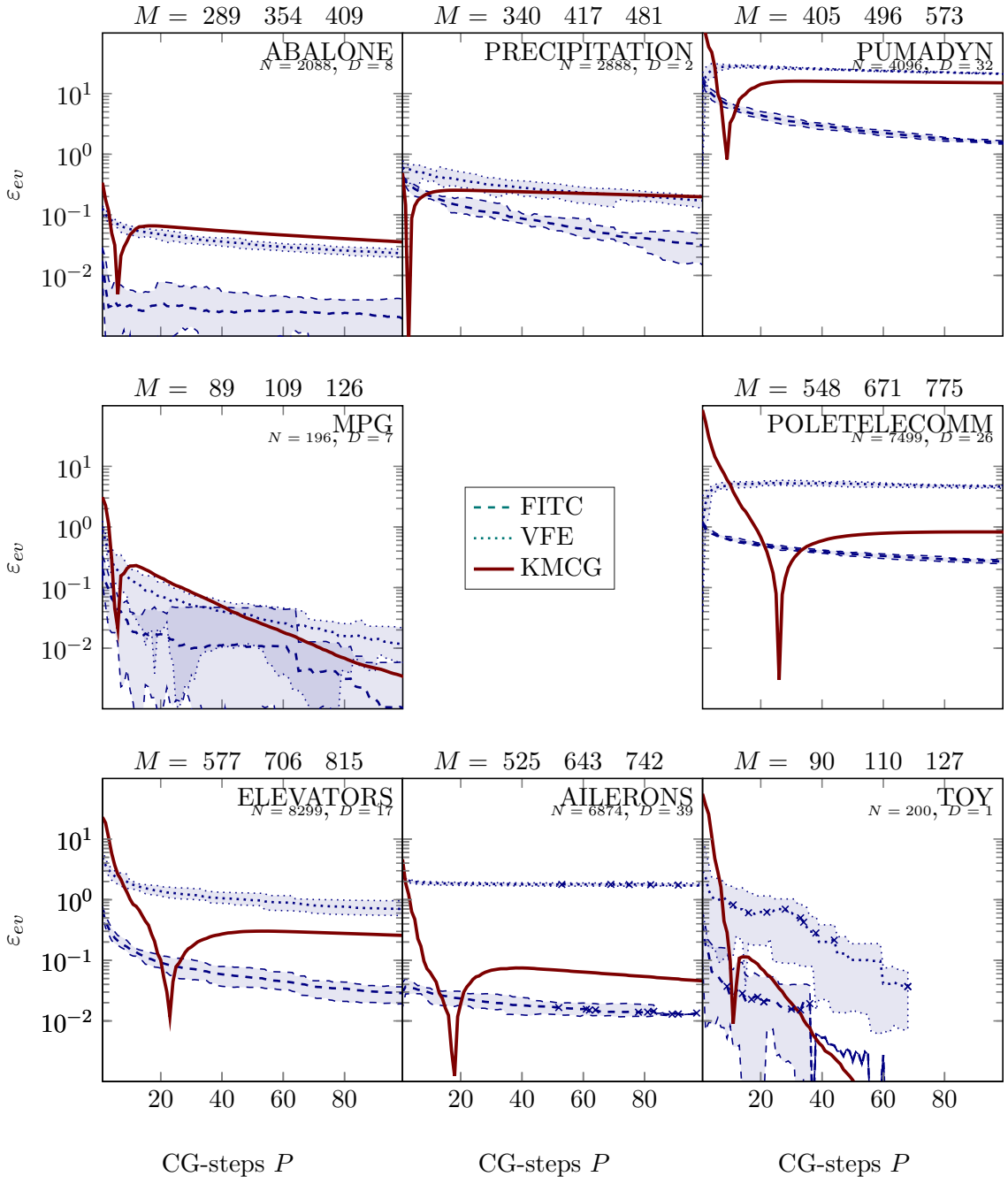


Figure 13: progression of the relative error of the evidence ε_{ev} as a function of the number of iterations of baseline and KMCG for different data sets using the Matérn kernel (Eq. 19). The bottom axis is the number of CG-steps, which is the same for all plots. Therefore, this axis is visible only in the last row. The top axis denotes the number of inducing inputs used by the baseline methods. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

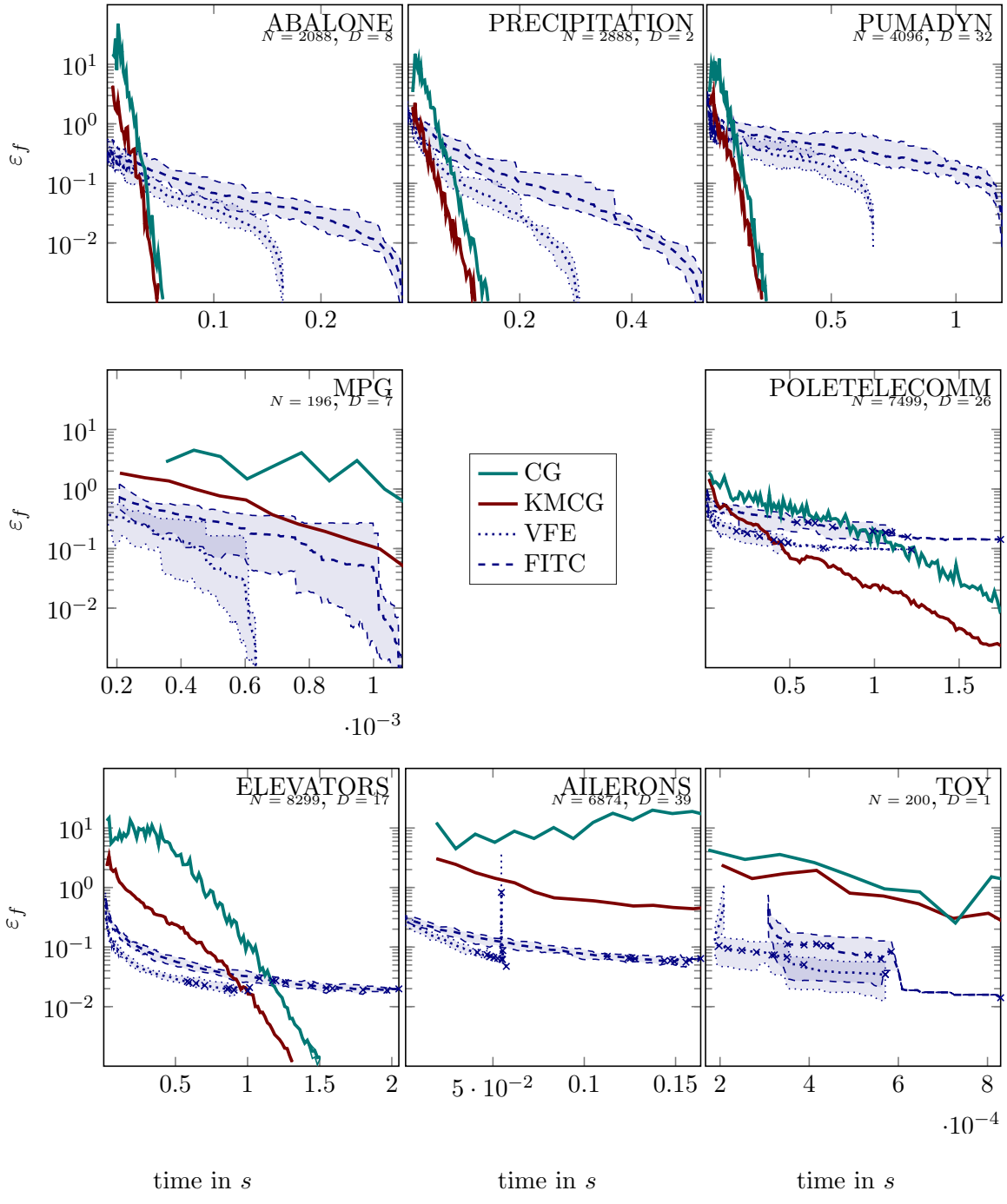


Figure 14: progression of the relative error ε_f over training time for different data sets using the Matérn kernel (Eq. 19). The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

E.3. Instability of Textbook Conjugate Gradients

The experiments in Section 4, where carried out by running conjugate gradients with full reorthogonalization. Fig. 15 demonstrates that for the problems under consideration, the textbook version of conjugate gradients is not sufficiently numerically stable.[13] With vanilla conjugate gradients in the background, KMCG can run only for a couple of steps before the Cholesky decomposition of $\boldsymbol{S}^\intercal \boldsymbol{K} \boldsymbol{S}$ fails.

References

Benoit. Note sûre une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés a un système d'équations linéaires en nombre inférieure a celui des inconnues. Application de la méthode a la résolution d'un système défini d'équations linéaires. (Procédé du Commandant Cholesky). *Bulletin Geodesique*, 7(1), 1924.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Rui Camachol. Inducing models of human control skills. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning*, 1998.

Krzysztof Chalupka, Williams, Christopher K. I., and Iain Murray. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14(1), 2013.

Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3), 2002.

Alexander Davies. *Effective implementation of Gaussian process regression for machine learning*. PhD thesis, University of Cambridge, 2015.

Kun Dong, David Eriksson, Hannes Nickisch, David Bindel, and Andrew G Wilson. Scalable log determinants for Gaussian process kernel learning. In *Advances in Neural Information Processing Systems*, 2017.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2019. URL `http://archive.ics.uci.edu/ml`.

Maurizio Filippone and Raphael Engler. Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE). In *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2015.

Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 4 edition, 2013.

13. Approved by the editor, this figure differs from the accepted revision, after correcting an error.

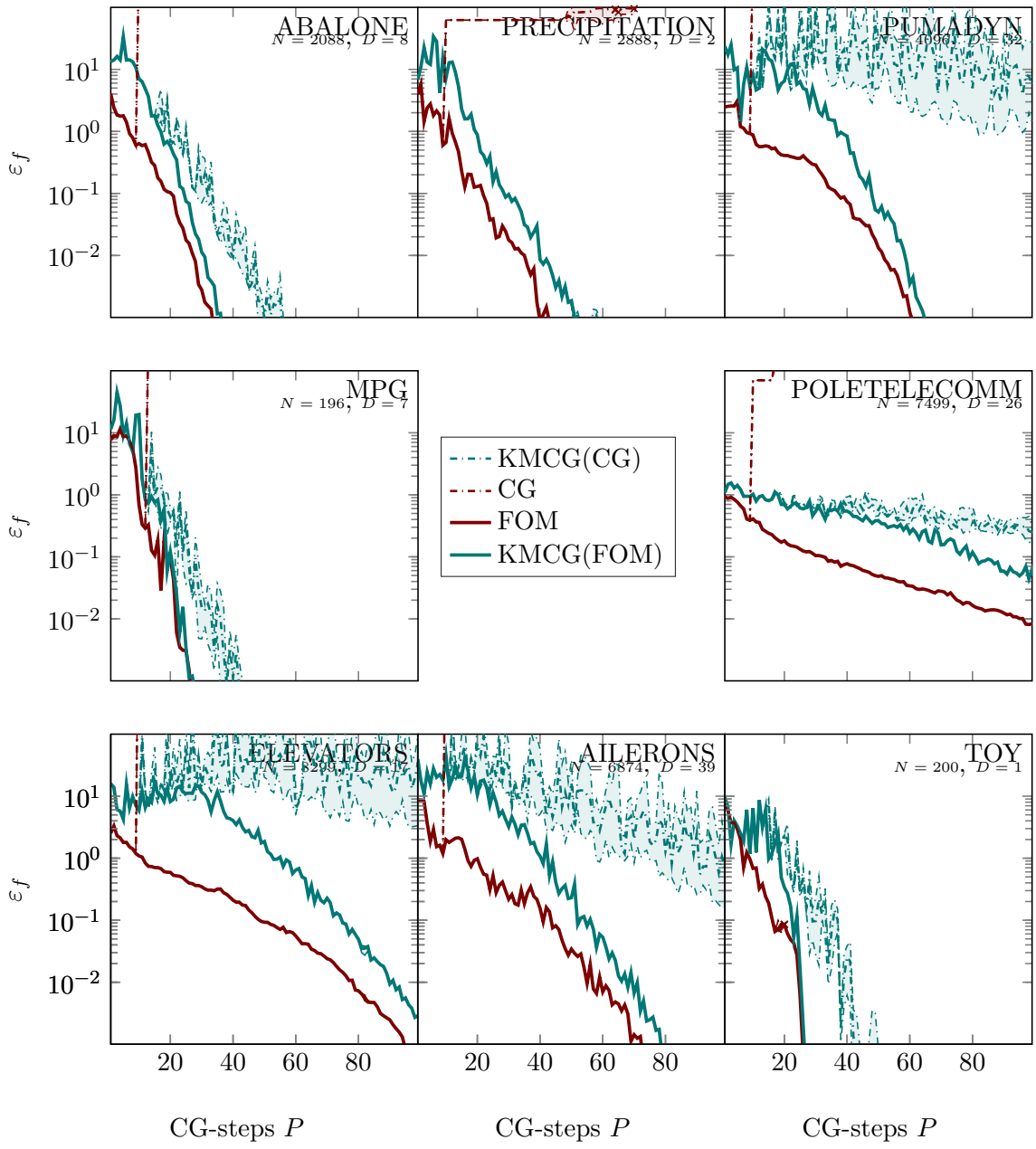


Figure 15: progression of the relative error ε_f over 100 CG-steps for different data sets using the squared exponential kernel (Eq. (18)), comparing CG and FOM. The shaded area visualizes minimum and maximum over all baseline runs. A cross denotes the end of a crashed run.

Philipp Hennig. Probabilistic interpretation of linear solvers. *SIAM Journal on Optimization*, 25(1), 2015.

Philipp Hennig and Martin Kiefel. Quasi-Newton methods – a new direction. *Journal of Machine Learning Research*, 14, March 2013.

Philipp Hennig, Michael A. Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2015.

James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151), 2018.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 1970.

George S. Kimeldorf and Grace Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 1970.

Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl E. Rasmussen, and Aníbal R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11, 2010.

Quoc Le, Tamas Sarlos, and Alexander Smola. Fastfood - computing Hilbert space expansions in loglinear time. In *Proceedings of the 28th International Conference on Machine Learning*, 2013.

Charles F. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1), 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

Jan R. Magnus and Heinz Neudecker. The elimination matrix: Some lemmas and applications. *SIAM Journal on Algebraic Discrete Methods*, 1(4), 1980. doi: 10.1137/0601049.

Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley, second edition, 1999.

Georges Matheron. The intrinsic random functions and their applications. *Advances in applied probability*, 1973.

Warwick J. Nash, Tracy L. Sellers, Simon R. Talbot, Andrew J. Cawthorn, and Wes B. Ford. The population biology of abalone (haliotis species) in Tasmania. 1, blacklip abalone (h. rubra) from the north coast and the islands of bass strait. Technical Report 48, Sea Fisheries Division, Marine Research Laboratories - Taroona, Department of Primary Industry and Fisheries, Tasmania, 1994. URL `https://trove.nla.gov.au/work/11326142`.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Verlag, 1999.

Geoff Pleiss, Jacob Gardner, Kilian Weinberger, and Andrew G. Wilson. Constant-time predictive distributions for Gaussian processes. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

John R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the 10th International Conference on International Conference on Machine Learning*, 1993.

Joaquin Quiñonero-Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6, 2005.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, 2008.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems 23*, 2009.

Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT, 2006.

Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11, 2010.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

Horst D. Simon. Analysis of the symmetric lanczos algorithm with reorthogonalization methods. *Linear Algebra and its Applications*, 61, 1984.

John Skilling. Bayesian numerical analysis. In Jr W. T. Grandy and P. W. Milonni, editors, *Physics and Probability: Essays in Honor of Edwin T. Jaynes*. 1993.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*. 2006.

Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression, 2014. URL `https://arxiv.org/abs/1401.5508v1`.

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, 2009a.

Michalis K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, 2009b.

Giancarlo F. Trecate, Christopher K. I. Williams, and Manfred Opper. Finite-dimensional approximation of Gaussian processes. In *Advances in Neural Information Processing Systems 2*, 1999.

Richard E. Turner. *Statistical Models for Natural Sounds*. PhD thesis, University College London, 2010.

Jarno Vanhatalo and Aki Vehtari. Modelling local and global phenomena with sparse Gaussian processes. In David McAllester and Petri Myllymäki, editors, *UAI 2008, Twenty-Fourth Conference on Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, 2008.

Grace Wahba. *Spline models for observational data*. Number 59 in CBMS-NSF Regional Conferences series in applied mathematics. SIAM, 1990.

Christian Walder, Kwang I. Kim, and Bernhard Schölkopf. Sparse multiscale Gaussian process regression. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Sam Waugh. *Extending and benchmarking Cascade-Correlation*. PhD thesis, University of Tasmania, 1995.

Sholom M. Weiss and Nitin Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3(1), 1995.

Max Welling and Yee W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 2001.

Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (kiss-gp). In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, 2015.

Andrew G. Wilson, Elad Gilboa, Arye Nehorai, and John P. Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems 27*. 2014.

Feng Yan and Yuan Qi. Sparse Gaussian process regression via l1 penalization. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Huaiyu Zhu, Christopher K. I. Williams, Richard J. Rohwer, and Michal Morciniec. Gaussian regression and optimal finite dimensional linear models. In C. M. Bishop, editor, *Neural Networks and Machine Learning*. 1998.