Fast Approximate Bayesian Gaussian Process Regression

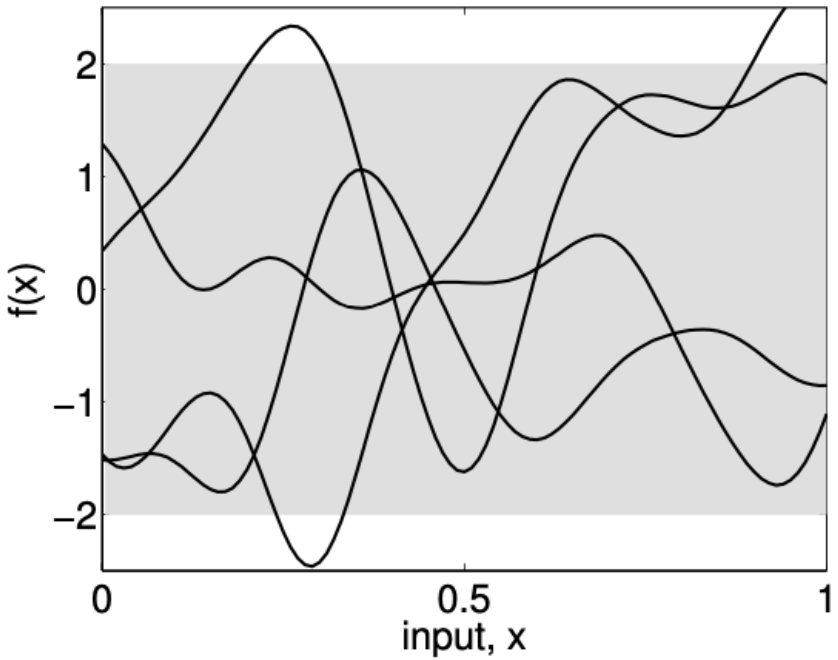
Kelly Moran

Statistical Sciences Group, Los Alamos National Laboratory

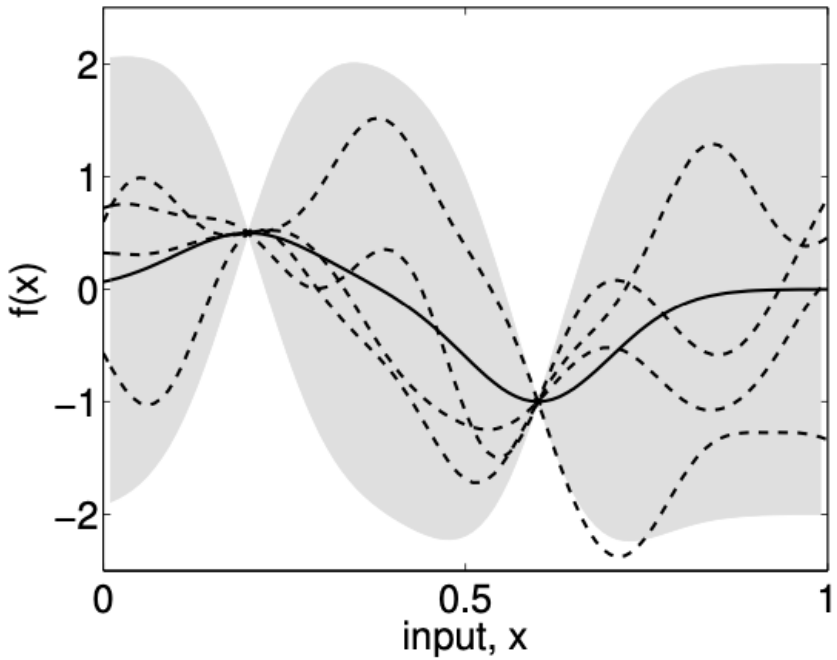
ISBA World Meeting
June 28, 2022

LA-UR-22-25861

Gaussian processes (GPs)



(a), prior

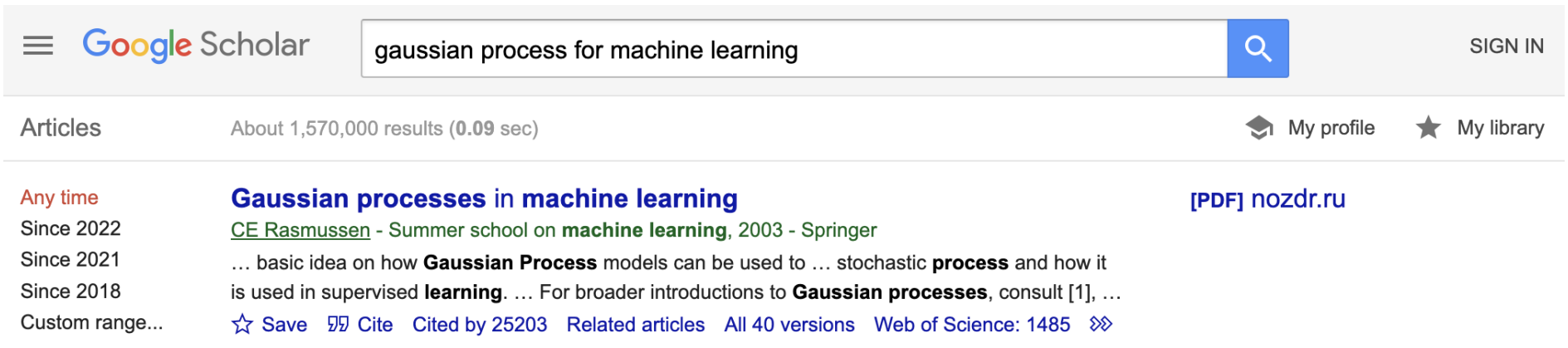


(b), posterior

Source: Gaussian Processes for Machine Learning by Rasmussen and Williams (2005).

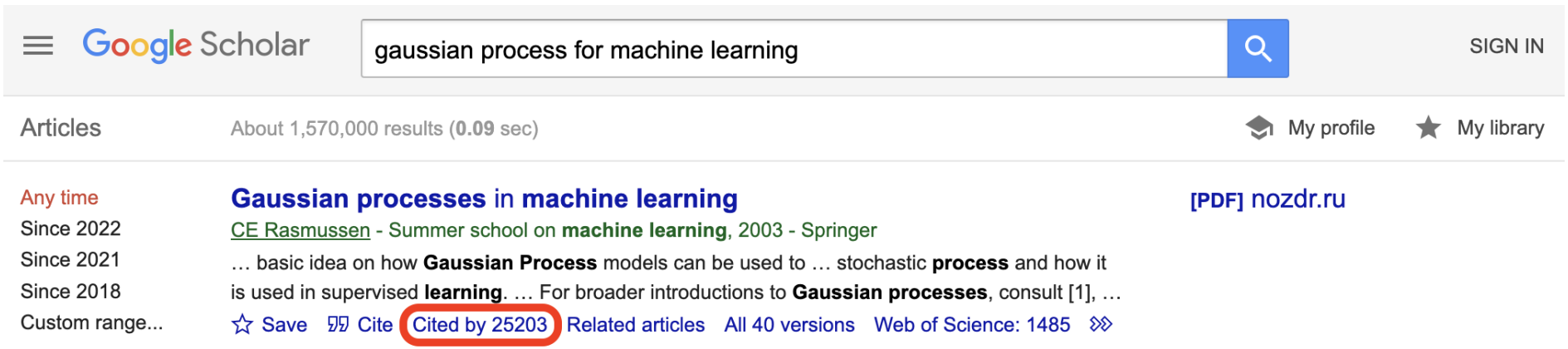
Gaussian processes (GPs) are useful tools

- Give a reliable estimate of their own uncertainty.
- Inherit benefits of the underlying Gaussian distribution. E.g., marginals, conditionals, and derivatives are all still Gaussian.
- Enable us to encode assumptions we may have about a data set, including periodicity, smoothness (or lack thereof), wiggliness, etc.
- Not just for regression with normally distributed error structure, can be used for regression with count data, classification, as a building block in other models, etc.



Gaussian processes (GPs) are useful tools

- Give a reliable estimate of their own uncertainty.
- Inherit benefits of the underlying Gaussian distribution. E.g., marginals, conditionals, and derivatives are all still Gaussian.
- Enable us to encode assumptions we may have about a data set, including periodicity, smoothness (or lack thereof), wiggliness, etc.
- Not just for regression with normally distributed error structure, can be used for regression with count data, classification, as a building block in other models, etc.



Gaussian process (GP) regression

$$y = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathrm{N}(0, \tau^{-1})$$

Inputs $x_i \in \mathbb{R}^d, \quad i = 1, \ldots, n$

Here $f \sim GP(\mu, k)$, encodes dependence.

Covariance function/kernel k is $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and is such that $k(x_i, x_j)$ is the covariance between $f(x_i)$ and $f(x_j)$.

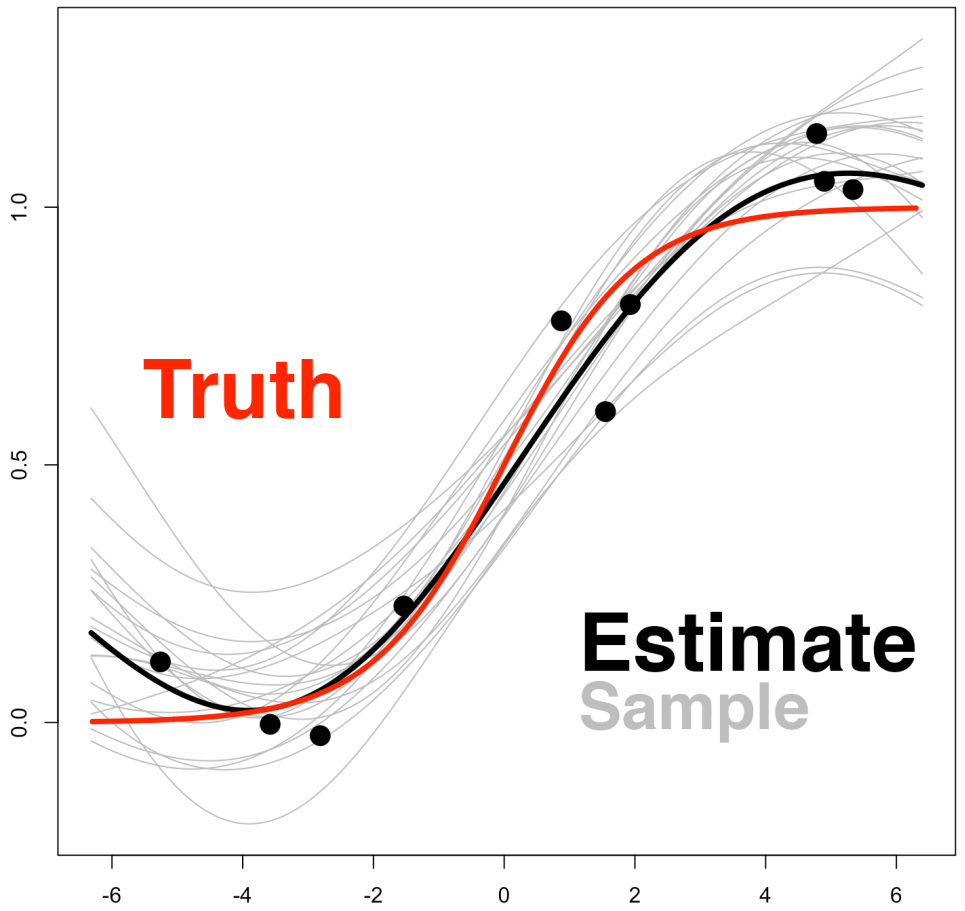
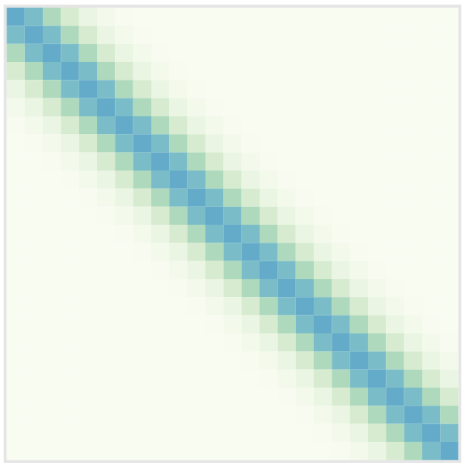


Figure: Samples from GP regression based on some observed data (black points).

Different covariance kernels

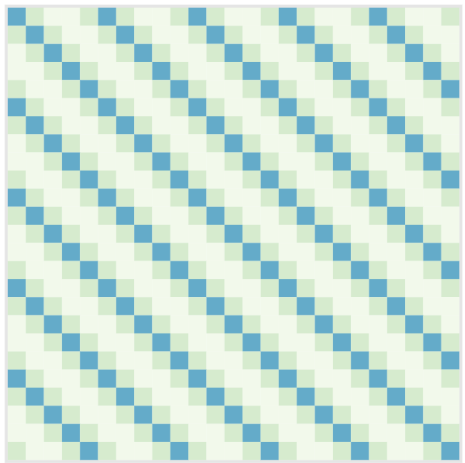
RBF KERNEL

$$\sigma^2 \exp\left(-\frac{||t-t'||^2}{2l^2}\right)$$



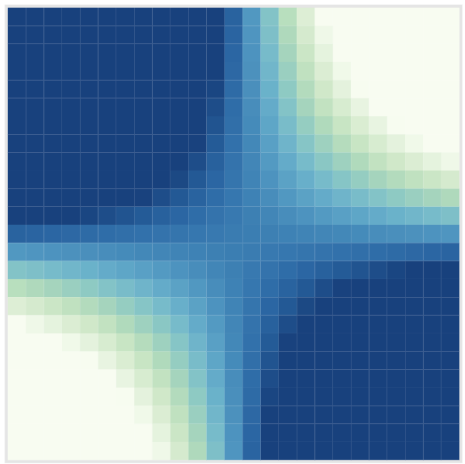
PERIODIC

$$\sigma^2 \exp\left(-\frac{2\sin^2(\pi|t-t'|/p)}{l^2}\right)$$



LINEAR

$$\sigma_b^2 + \sigma^2(t-c)(t'-c)$$



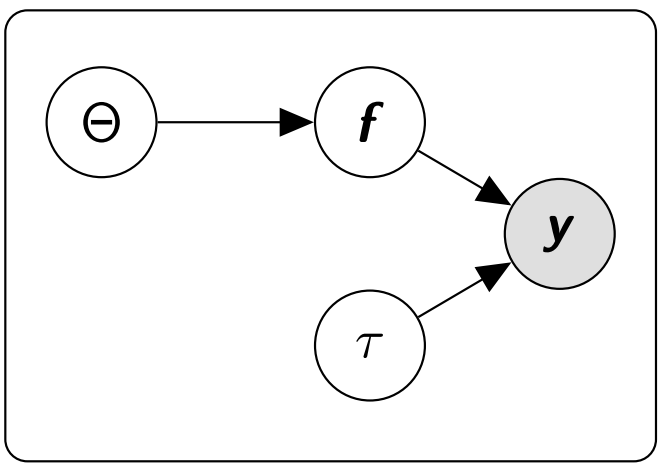
Source: `https://distill.pub/2019/visual-exploration-gaussian-processes/`.

Bayesian Gaussian process regression

- Rather than fixing the precision τ or kernel hyperparameters Θ, assign prior distributions $\pi(\cdot)$ for them.

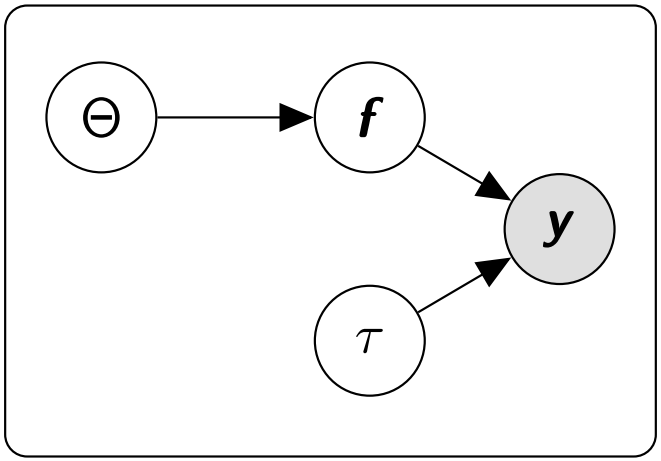
- The posterior distribution is:
  $\pi(\boldsymbol{f}, \Theta, \tau | \boldsymbol{y}) \propto \pi(\boldsymbol{y}|...)\ \pi(\boldsymbol{f}|\Theta)\ \pi(\Theta)\ \pi(\tau)$

- Can represent the model as a directed acyclic graph (DAG).

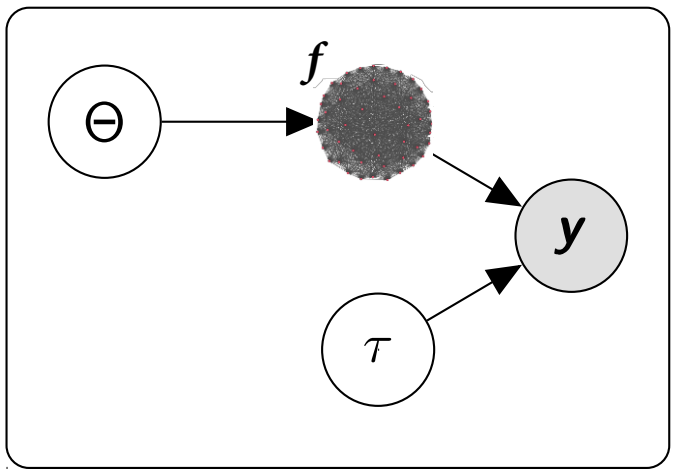


Why GP regression is expensive (DAG view)

Bayesian GP regression shown as a DAG:



But $\boldsymbol{f}$ is actually quite complex...



Why GP regression is expensive (equation view)

In equation view, the log-likelihood function is:

$$\ell_n(\Theta, \tau) = -\frac{n}{2} \log(2\pi) - \log\big[\det(K + \tau^{-1} I_n)\big] - \frac{1}{2} \boldsymbol{y}^T (K + \tau^{-1} I_n)^{-1} \boldsymbol{y},$$

and the GP function posterior is $\boldsymbol{f} | \boldsymbol{y}, X, \Theta, \tau \sim \mathrm{N}(\mu_f, \Sigma_f)$, where calculating the posterior mean μ_f and covariance Σ_f requires linear solves and matrix-matrix products.

Sampling from the GP prior requires

$$\text{Cholesky}(K) \cdot \boldsymbol{z} \text{ or } K^{\frac{1}{2}} \boldsymbol{z}, \quad \boldsymbol{z} \sim N(0_n, I_n),$$

and sampling from the posterior requires

$$\mu_f + \text{Cholesky}(\Sigma_f) \cdot \boldsymbol{z} \text{ or } \mu_f + \Sigma_f^{\frac{1}{2}} \boldsymbol{z}, \quad \boldsymbol{z} \sim N(0_n, I_n).$$

Why GP regression is expensive (equation view)

In equation view, the log-likelihood function is:

$$\ell_n(\Theta, \tau) = -\frac{n}{2}\log(2\pi) - \log\left[\det(K + \tau^{-1} I_n)\right] - \frac{1}{2}\, \boldsymbol{y}^T (K + \tau^{-1} I_n)^{-1} \boldsymbol{y},$$

and the GP function posterior is $\boldsymbol{f} | \boldsymbol{y}, X, \Theta, \tau \sim \mathrm{N}(\mu_f, \Sigma_f)$, where calculating the posterior mean μ_f and covariance Σ_f requires linear solves and matrix-matrix products

Sampling from the GP prior requires

$$\text{Cholesky}(K) \cdot \boldsymbol{z} \text{ or } K^{\frac{1}{2}} \boldsymbol{z}, \quad \boldsymbol{z} \sim N(0_n, I_n),$$

and sampling from the posterior requires

$$\mu_f + \text{Cholesky}(\Sigma_f) \cdot \boldsymbol{z} \text{ or } \mu_f + \Sigma_f^{\frac{1}{2}} \boldsymbol{z}, \quad \boldsymbol{z} \sim N(0_n, I_n).$$

Scalable GP regression

- Fast sparse Gaussian process methods: The informative vector machine (Herbrich, Lawrence, and Seeger, 2003)
- A unifying view of sparse approximate Gaussian process regression (Quinonero-Candela and Rasmussen, 2005)
- Gaussian predictive process models for large spatial data sets (Banerjee et al. 2008)
- Bayesian treed Gaussian process models with an application to computer modeling (Gramacy and Lee, 2008)
- Variational learning of inducing variables in sparse Gaussian processes (Titsias, 2009)
- On fixed-domain asymptotics and covariance tapering in Gaussian random field models (Wang and Loh 2011)
- A full scale approximation of covariance functions for large spatial data sets (Sang and Huang 2012)
- Twenty years of mixture of experts (Yuksel, Wilson, and Gader, 2012)
- Efficient Gaussian process regression for large datasets (Banerjee, Dunson, and Tokdar 2012)
- Covariance tapering for prediction of large spatial data sets in transformed random fields (Hirano and Yajima, 2013)
- Massively Parallel Approximate Gaussian Process Regression (Gramacy, Niemi, and Weiss, 2014)
- Distributed variational inference in sparse Gaussian process regression and latent variable models (Gal et al., 2014)
- Gaussian process models with parallelization and GPU acceleration (Dai et al., 2014)
- Distributed Gaussian processes (Deisenroth and Ng, 2015)
- Local Gaussian Process Approximation for Large Computer Experiments (Gramacy and Apley, 2015)
- Fast Direct Methods for Gaussian Processes (Ambikasaran et al., 2015)
- Kernel interpolation for scalable structured Gaussian processes (KISS-GP) (Wilson and Nickisch, 2015)
- Multi-level restricted maximum likelihood covariance estimation and kriging for large non-gridded spatial datasets (Castrillon-Candás, Genton, and Yokota, 2016)
- Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets (Datta et al. 2016)
- Nonseparable dynamic nearest neighbor Gaussian process models for large spatio-temporal data with an application to particulate matter analysis (Datta et al., 2016)
- Multi-resolution approximations of Gaussian processes for large spatial datasets (Katzfuss and Gong, 2018)
- Generalized robust Bayesian committee machine for large-scale Gaussian process regression (Liu et al., 2018)
- Nested Kriging predictions for datasets with a large number of observations (Rulliere et al., 2018)
- Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes (Finley et al. 2019)
- Likelihood Approximation With Hierarchical Matrices For Large Spatial Datasets (Litvinenko et al., 2019)
- On random subsampling of Gaussian process regression: A graphon-based analysis (Hayashi, Imaizumi, and Yoshida, 2019)
- And many, many, many more not listed here (see review papers such as "When Gaussian process meets big data: A review of scalable GPs")

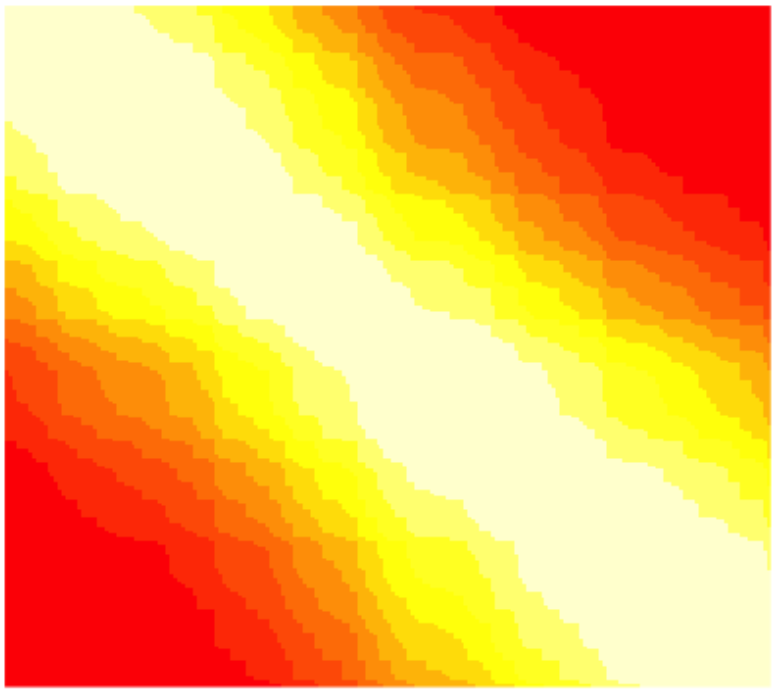
Where our approach fits in

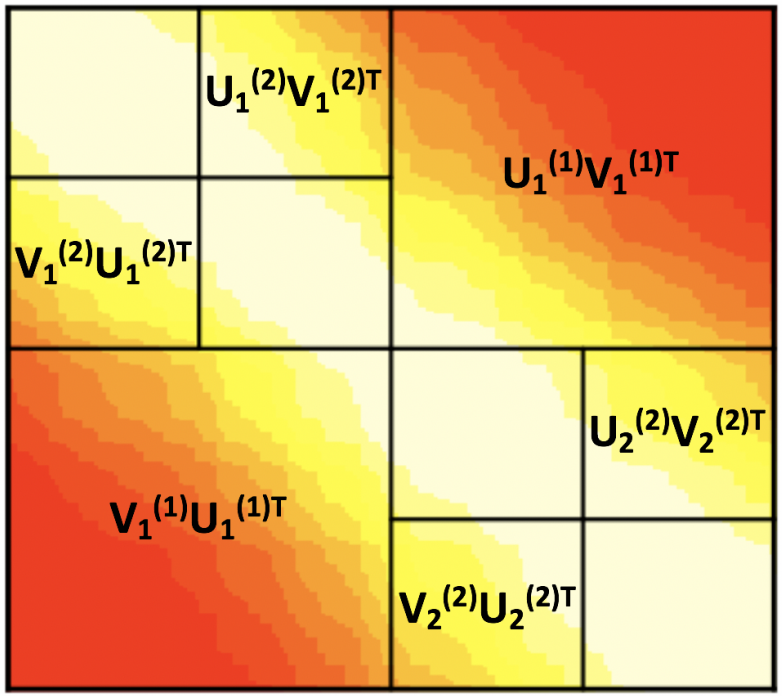
The philosophy: Rather than changing the model, approximate the covariance matrix in a way that allows you to speed up the operations necessary for fully Bayesian inference.

What we develop: A novel sampling algorithm for posterior draws of $f|y$ with a provable bound on the divergence between the true and approximate posterior, novel use of $\mathcal{H}$-matrices in the Bayesian context, and a tensor product formulation for higher-dimensional input spaces. Published in JRSS-B in 2022: See `https://doi.org/10.1111/rssb.12494`.

Where we're going: Explore alternative ways of approximating K (or k). Develop and release modular **R** code that can be incorporated into existing samplers in place of exact GPs.

Example: Approximating K using an $\mathcal{H}$-matrix





Using an approximation based on the Hierarchical Off-Diagonal Low Rank (HODLR) decomposition, a type of hierarchical matrix, you can calculate the Cholesky, perform solves, and get determinants at $\mathcal{O}(n \log n)$ cost.

$\mathcal{H}$-matrices and GPs

Prior to our work, $\mathcal{H}$-matrices had not yet been extended to fully Bayesian GP regression models. Here's why.

Say you want to sample from n-dimensional $\mathrm{N}(\mathbf{0}, C)$.

1. Get Cholesky factorization $C = LL^T$.
2. Sample n univariate standard normal random variables; call these samples $\boldsymbol{z}$, so $\boldsymbol{z} \sim \mathrm{N}(\mathbf{0}, I_n)$.
3. Then $L\boldsymbol{z}$ has the desired distribution.

The GP posterior doesn't fit neatly in this format:

$$\begin{aligned} \boldsymbol{f}|\boldsymbol{y}, X, \Theta, \tau &\sim \mathrm{N}(\mu_f, \Sigma_f), \\ \Sigma_f &= K - K(K + \tau^{-1}\mathbf{I}_n)^{-1}K \\ &= (K^{-1} + \tau\mathbf{I}_n)^{-1} \\ &= K(\tau K + \mathbf{I}_n)^{-1} \end{aligned}$$

Sampling algorithm

Algorithm 1: Recall τ is the noise precision. Let ϵ denote the tolerated difference max norm. Sampling proceeds as follows:

1. If $\tau < 1$, set $\epsilon^* = \tau\epsilon$, else set $\epsilon^* = \epsilon$.
2. Create $\mathcal{H}$ matrix decomposition of K with tolerance ϵ^*/τ and $\tau K + I$ with tolerance ϵ^*, $\tilde{K}$ and $\tilde{M}$ respectively (assembly, cost $\mathcal{O}(n \log^2 n)$).
3. Get W s.t. $\tilde{K} = WW^T$ (symmetric factorization, cost $\mathcal{O}(n \log^2 n)$).
4. Sample $\boldsymbol{a}, \boldsymbol{b} \sim \mathrm{N}(0, I_n)$.
5. Let $Z = \sqrt{\tau}\tilde{K}a + Wb$ (matrix vector products, cost $\mathcal{O}(n \log n)$).
6. Let $Z^* = \tilde{M}^{-1}Z$ (solve, cost $\mathcal{O}(n \log n)$).
7. Let $R = \tau\tilde{K}\tilde{M}^{-1}\boldsymbol{y}$ (solve then mat-vec product, cost $\mathcal{O}(n \log n)$).
8. Finally, let $Z^{**} = R + Z^*$, which is the approximated sample from $p(\boldsymbol{f}|\boldsymbol{y}, X, \Theta)$ (vector addition, cost $\mathcal{O}(n)$).

Sampling algorithm

Algorithm 1: Recall τ is the noise precision. Let ϵ denote the tolerated difference max norm. Sampling proceeds as follows:

1. If $\tau < 1$, set $\epsilon^* = \tau\epsilon$, else set $\epsilon^* = \epsilon$.
2. Create $\mathcal{H}$ matrix decompositi… $^*/\tau$ and $\tau K + I$ with tolerance ϵ^*, $\tilde{K}$ a… t $\mathcal{O}(n \log^2 n)$).
3. Get W s.t. $\tilde{\nu}$ … $(n \log^2 n)$).
4. Sample $\boldsymbol{a}$, …
5. Let $Z = \sqrt{\tau}$… products, cost $\mathcal{O}(n \log n)$).
6. Let $Z^* = \tilde{M}^{-1}$… $\mathcal{O}(n \log n)$).
7. Let $R = \tau \tilde{K} \tilde{M}^{-1}$… (solve then mat-vec product, cost $\mathcal{O}(n \log n)$).
8. Finally, let $Z^{**} = R + Z^*$, which is the approximated sample from $p(\boldsymbol{f}|\boldsymbol{y}, X, \Theta)$ (vector addition, cost $\mathcal{O}(n)$).

TL;DR

Approximation fidelity

Theorem

Let $\boldsymbol{p} \sim N(\mu_f, \Sigma_f)$ *where* $\mu_f = \tau \Sigma_f \boldsymbol{y}$ *and* Σ_f *is an* $n \times n$ *positive definite matrix, with* $\Sigma_f = K(\tau K + I)^{-1}$ *for* K *the* $n \times n$ *realization of some symmetric covariance kernel,* $\boldsymbol{y}$ *is a length-n vector, and* τ *is a constant. Define* $M = (\tau K + I)$ *such that* $\Sigma_f = KM^{-1}$. *Then there exists matrices* $\tilde{K}, \tilde{M} \in \mathcal{H}$ *with* $||K - \tilde{K}||_{\max} \leq \varepsilon$ *and* $||M - \tilde{M}||_{\max} \leq \varepsilon$ *such that for* $\tilde{\Sigma}_f = \tilde{K}\tilde{M}^{-1}$, $\tilde{\mu}_f = \tau \tilde{\Sigma}_f \boldsymbol{y}$, *and* $\boldsymbol{q} \sim N(\tilde{\mu}_f, \tilde{\Sigma}_f)$

$$\mathcal{D}_{KL}(\mathcal{P}||\mathcal{Q}) = E_{\mathcal{P}}\left[log\left(\frac{\mathcal{P}}{\mathcal{Q}}\right)\right] \leq c_1 n^2 \varepsilon + c_2 n^{5/2} \varepsilon + c_3 n^3 \varepsilon^2, \qquad (1)$$

with $\lim_{\varepsilon \to 0} \mathcal{D}_{KL}(\mathcal{P}||\mathcal{Q}) = 0$, *where the density functions of* $\boldsymbol{p}$ *and* $\boldsymbol{q}$ *are denoted by* $\mathcal{P}$ *and* $\mathcal{Q}$, *respectively. The constants* c_1, c_2, *and* c_3 *are dependent on the conditioning of* K *and* M.

Approximation fidelity

Theorem

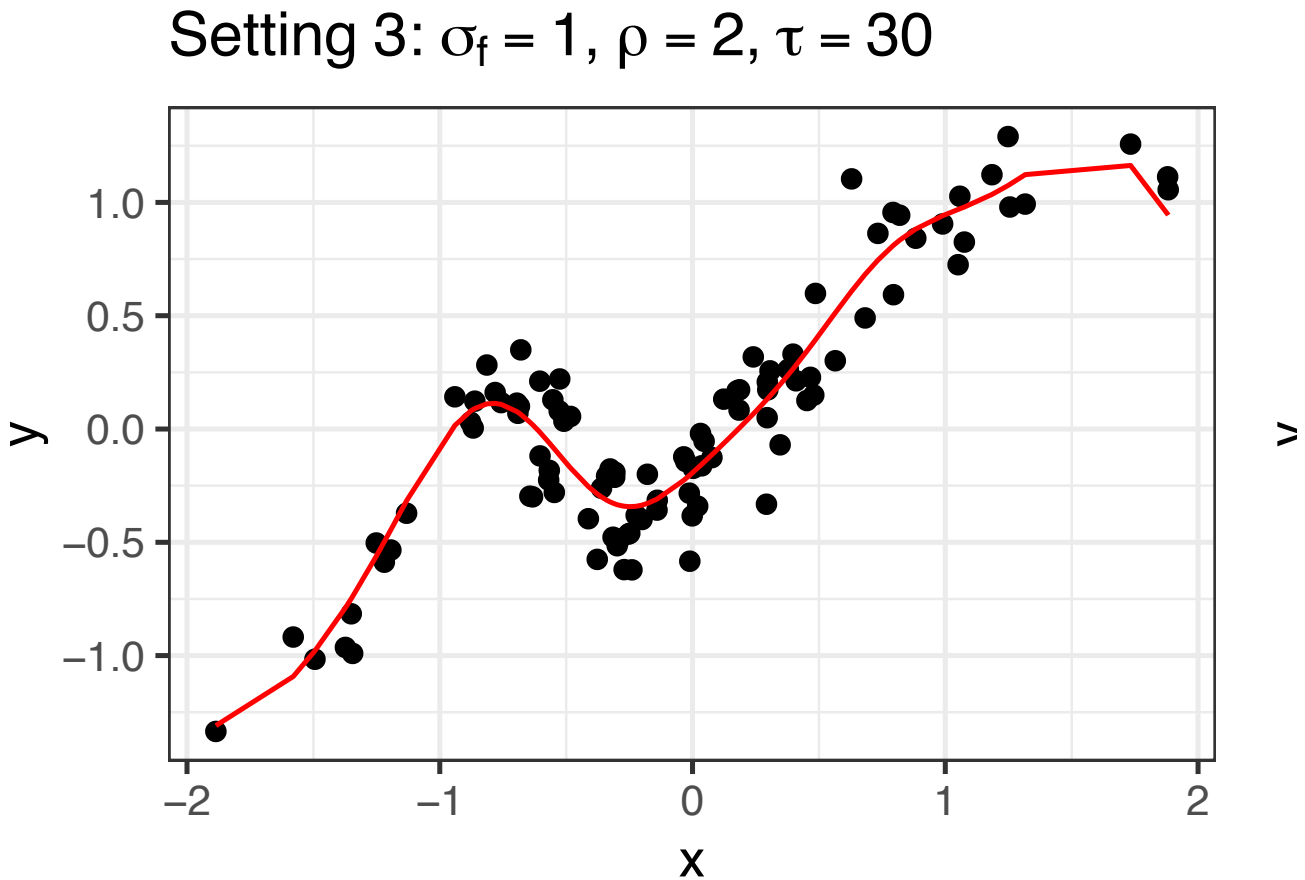
Let $\boldsymbol{p} \sim N(\mu_f, \Sigma_f)$ *where* $\mu_f = \tau \Sigma_f \boldsymbol{y}$ *and* Σ_f *is an* $n \times n$ *positive definite matrix, with* $\Sigma_f = K(\tau K + I)^{-1}$ *for* K *the* $n \times n$ *[illegible]zation of some symmetric covariance kernel,* $\boldsymbol{y}$ *is a [illegible] and* τ *is a constant. Define* $M = (\tau K + I)$ *such th[illegible]re exists matrices* $\tilde{K}, \tilde{M} \in \mathcal{H}$ *with* $||K - \tilde{K}$ *[illegible]such that for* $\tilde{\Sigma}_f = \tilde{K}\tilde{M}^{-1}$, $\hat{\mu}$ *[illegible]*

$$\mathcal{D}_{KL}(\mathcal{P}||\mathcal{Q}) \leq \varepsilon + c_2 n^{5/2}\varepsilon + c_3 n^3 \varepsilon^2, \qquad (1)$$

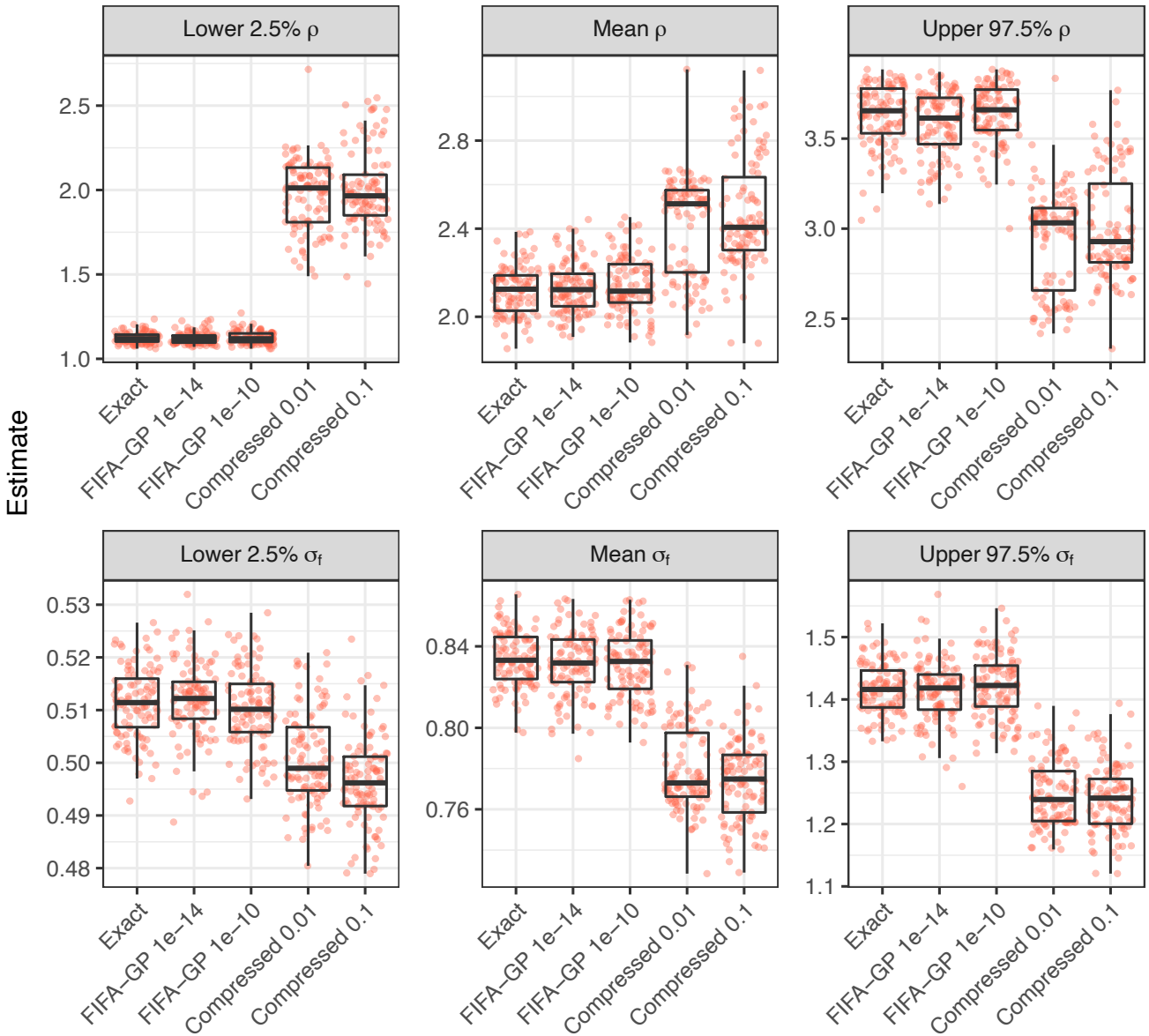
with $\lim_{\varepsilon \to 0} \mathcal{D}_{KL}(\mathcal{P}||\mathcal{Q}) = 0$, *where the density functions of* $\boldsymbol{p}$ *and* $\boldsymbol{q}$ *are denoted by* $\mathcal{P}$ *and* $\mathcal{Q}$, *respectively. The constants* c_1, c_2, *and* c_3 *are dependent on the conditioning of* K *and* M.

TL;DR

Small n simulation, 1D input (example data set)

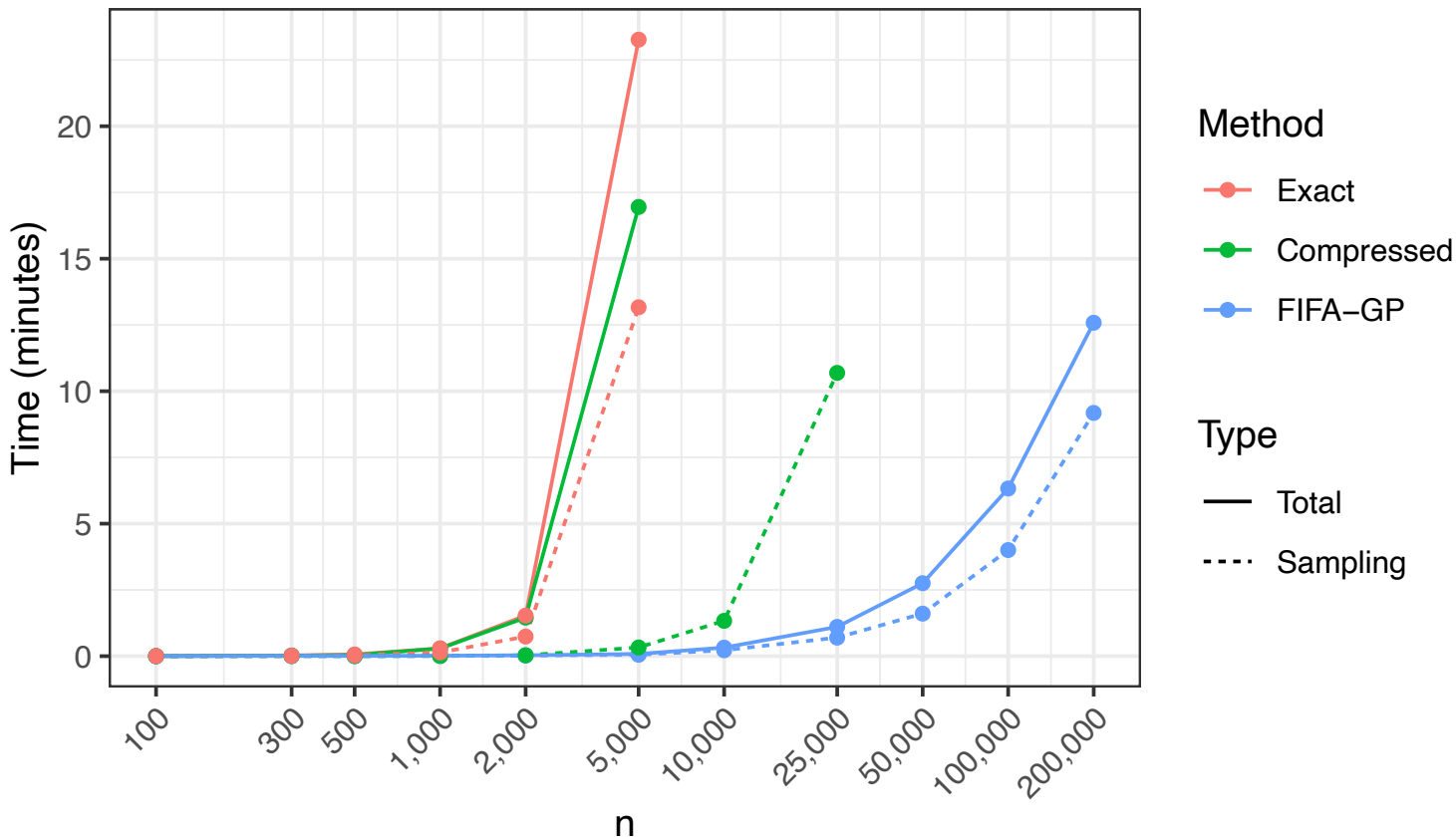


Small n simulation, 1D input (example results, more in paper)



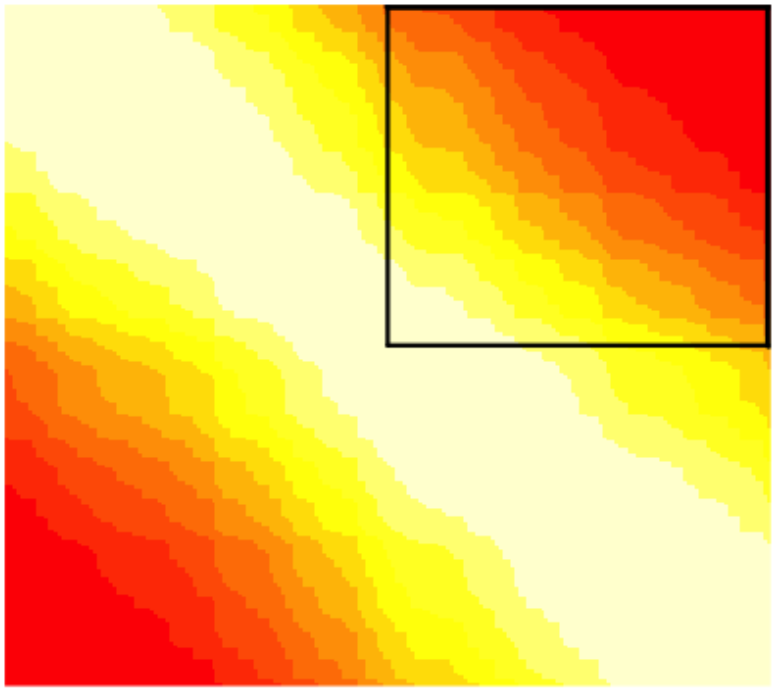
Large n simulation, 1D input (timing for 100 samples)

Observations have mean $f(x) = \sin 2x + \frac{1}{8}e^x$ and precision $\tau = 2$.

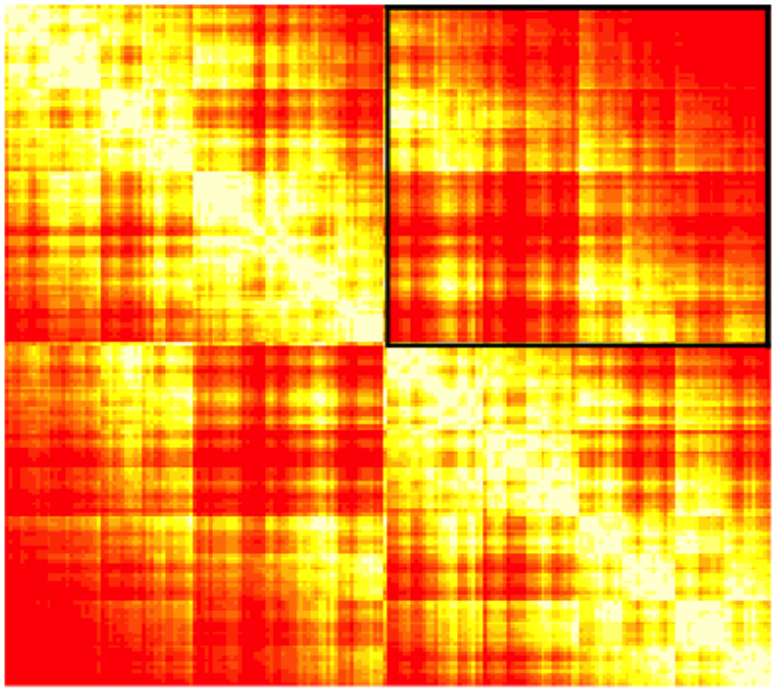


MSE and MSPE for FIFA-GP are similar to the exact GP and compressed GP, and superior to the NNGP.

Dimension of input space



1D input requires rank $r = 8$ matrix to approximate to tolerance 10^{-12}.



2D input requires rank $r = 56$ matrix to approximate to tolerance 10^{-12}.

The curse of dimensionality

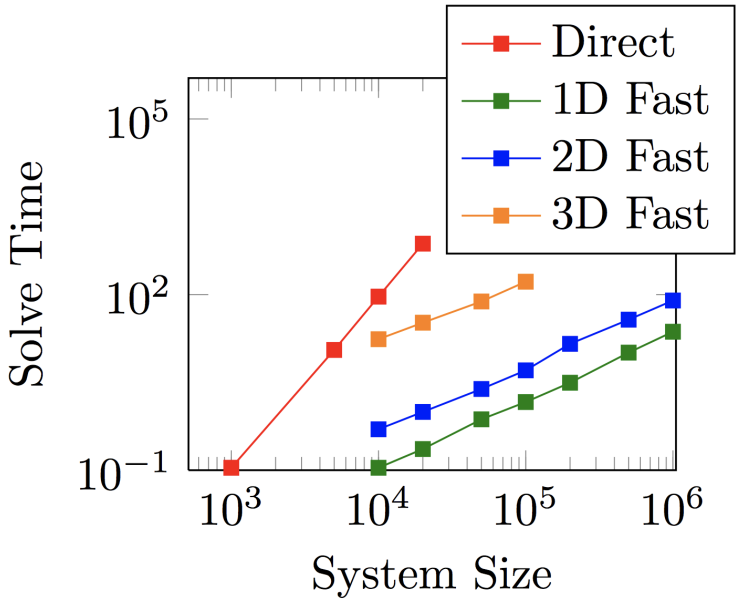


TABLE II

TIMINGS FOR GAUSSIAN COVARIANCE FUNCTIONS IN ONE, TWO, AND THREE DIMENSIONS. THE MATRIX ENTRIES ARE GIVEN AS $C_{ij} = 2\delta_{ij} + \exp\left(-||r_i - r_j||^2\right)$, WHERE r_i ARE RANDOM UNIFORMLY DISTRIBUTED POINTS IN THE INTERVAL $[-3, 3]^d$ $(d = 1, 2, 3)$.

	One-dimensional data					Two-dimensional data					Three-dimensional data				
n	Assembly	Factor	Solve	Det.	Error	Assembly	Factor	Solve	Det.	Error	Assembly	Factor	Solve	Det.	Error
$10,000$	0.12	0.11	0.008	0.01	10^{-13}	0.56	0.50	0.018	0.03	10^{-13}	15.4	17.3	0.113	0.91	10^{-12}
$20,000$	0.15	0.23	0.016	0.03	10^{-13}	1.16	0.99	0.028	0.05	10^{-13}	30.9	33.1	0.224	1.06	10^{-12}
$50,000$	0.47	0.71	0.036	0.12	10^{-12}	2.74	2.44	0.067	0.12	10^{-13}	75.5	76.3	0.434	1.68	10^{-11}
$100,000$	1.24	1.46	0.052	0.24	10^{-12}	5.43	5.08	0.165	0.23	10^{-12}	149	166	0.923	3.11	10^{-11}
$200,000$	2.14	3.12	0.121	0.39	10^{-13}	12.4	14.4	0.485	0.44	10^{-12}					
$500,000$	6.13	10.2	0.388	0.56	10^{-12}	31.7	37.3	1.33	1.17	10^{-12}					
$1,000,000$	14.1	23.2	0.834	1.52	10^{-12}	70.8	79.2	3.15	2.24	10^{-12}					

From "Fast Direct Methods for Gaussian Processes" by Ambikasaran et al. (2015).

Extending to higher dimensional inputs

- Our original solution was to model the d-dimensional surface as a tensor product of 1-dimensional GPs:

$$y_i = f_1(x_{1,i}) \otimes f_2(x_{2,i}) \otimes \ldots \otimes f_d(x_{d,i}) + e_i,$$
$$e_i \sim \mathrm{N}(0, \tau^{-1}), \quad f_h(\cdot) \sim GP(m_h(\cdot), k_h(\cdot,\cdot)),$$
$$i = 1, \ldots, n, \quad h = 1, \ldots, d.$$

- This approach can work well for some problems; see JRSS-B paper for a proof and details on simulated and non-synthetic performance.

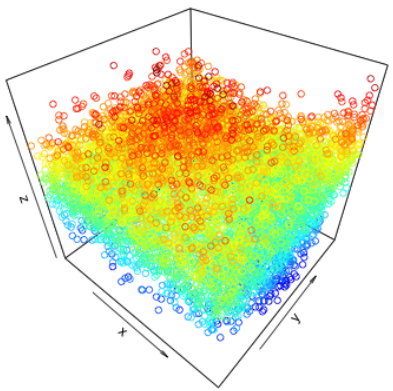
- But if we don't want to fiddle with tensor products, we could utilize a non-HODLR $\mathcal{H}$-matrix class that scales better with d (e.g. `H2Lib`, `HLIBCov`, or `SMASH`), or rethink the best stage at which to perform the approximation (i.e., perhaps approximate the kernel k rather than the realization of that kernel K).

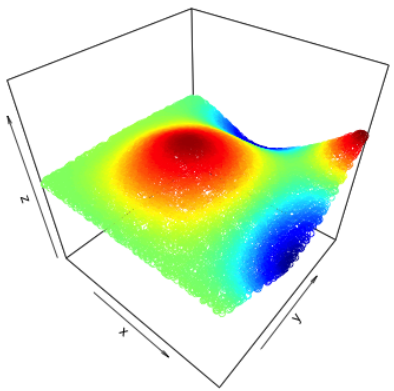
Tensor product simulation (data/surfaces)

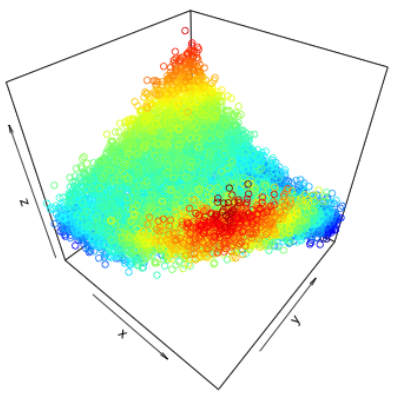
Simulate data from:

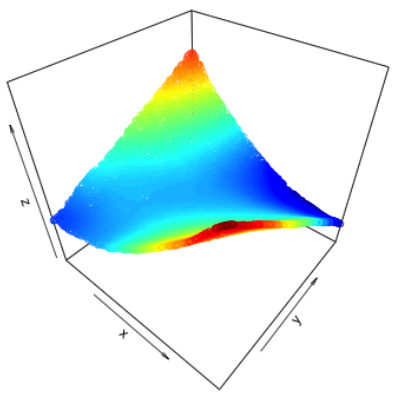
(Top) $g_1(x_1, x_2) = \sin(x_1)\sin(x_2)\sqrt{x_1 x_2}$

(Bottom) $g_2(x_1, x_2) = x^2 - 2xy + 3y + 2$

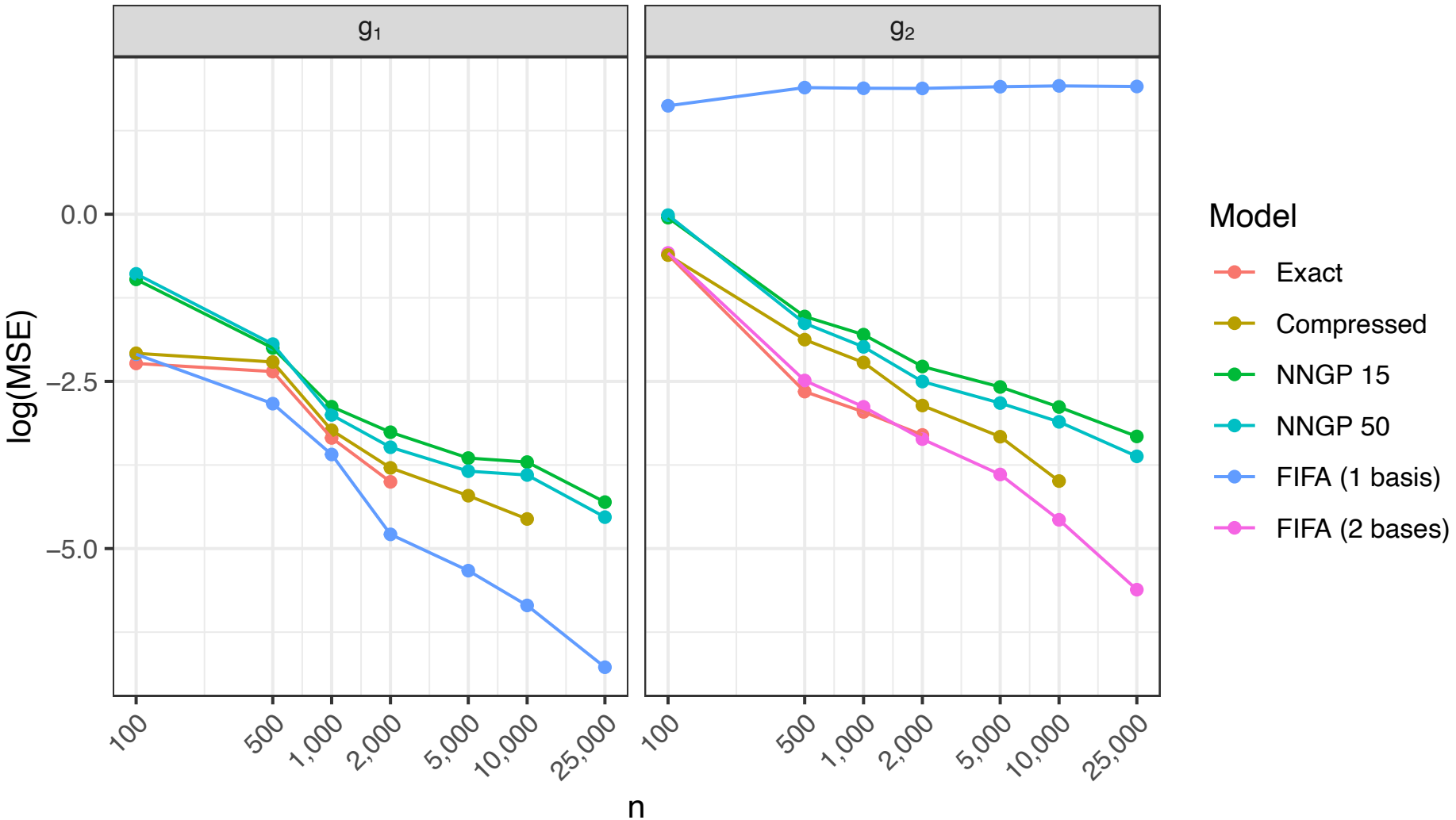




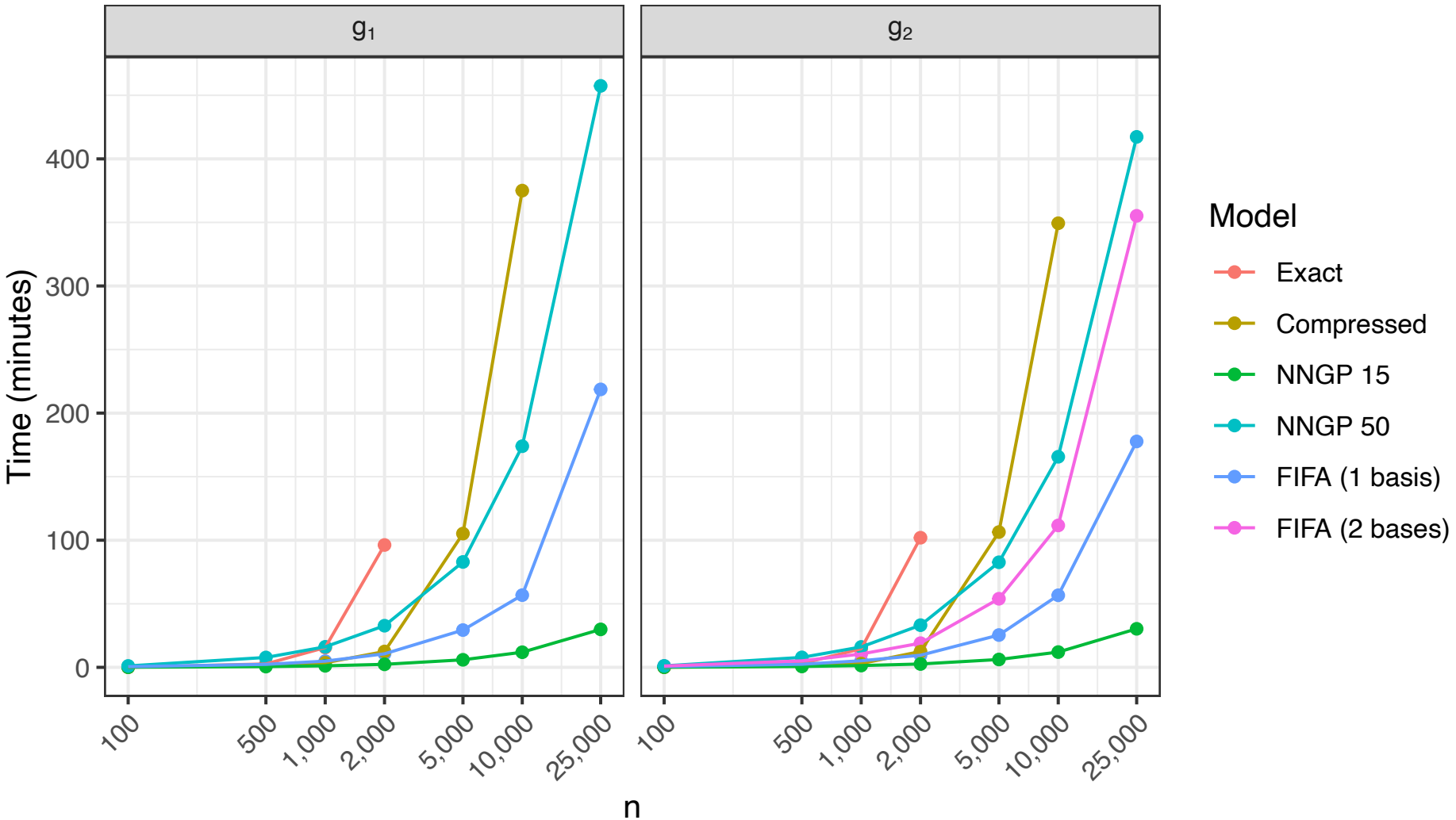




Tensor product simulation (MSE)



Tensor product simulation (timing for 7,000 iterations)



Illustrative real data examples

Atmospheric CO2 at Mauna Loa, HI:

- 358,253 observations, data at hourly level from 1974 to present.
- FIFA-GP **outperforms** the NNGP in MSPE for **75%** of hold-out years and consistently takes 1/3 the computation time (around 30 minutes for 7,000 samples).

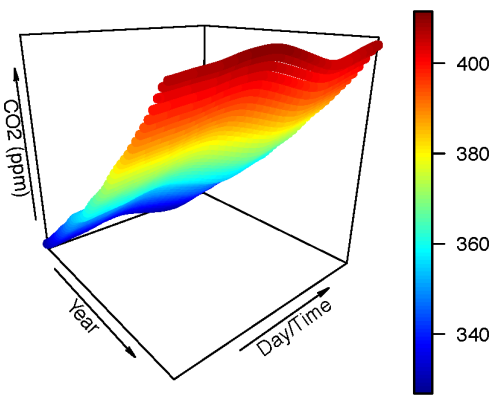


Figure: Model predicted atmospheric CO2 surface.

Particulate matter ($PM_{2.5}$) in USA:

- 4,977,391 observations, daily data for contiguous US since 1999.
- FIFA-GP yielded an **18% improvement** relative to the NNGP for hold-out predictive performance, and completed 7,000 samples in about **an hour**.

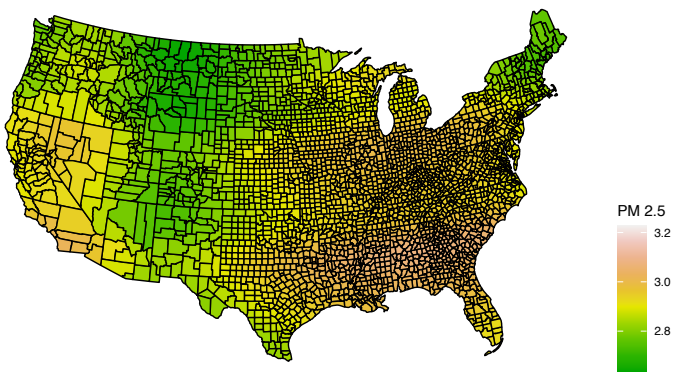
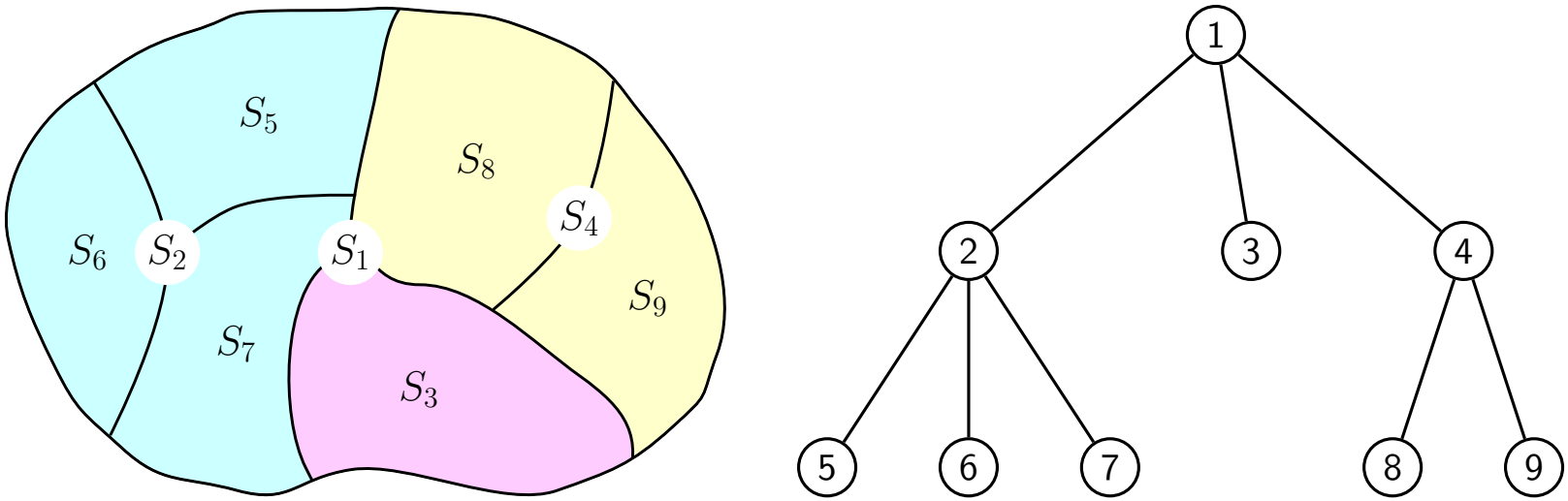


Figure: Model predicted 2019 $PM_{2.5}$ surface.

Approximate k instead of K

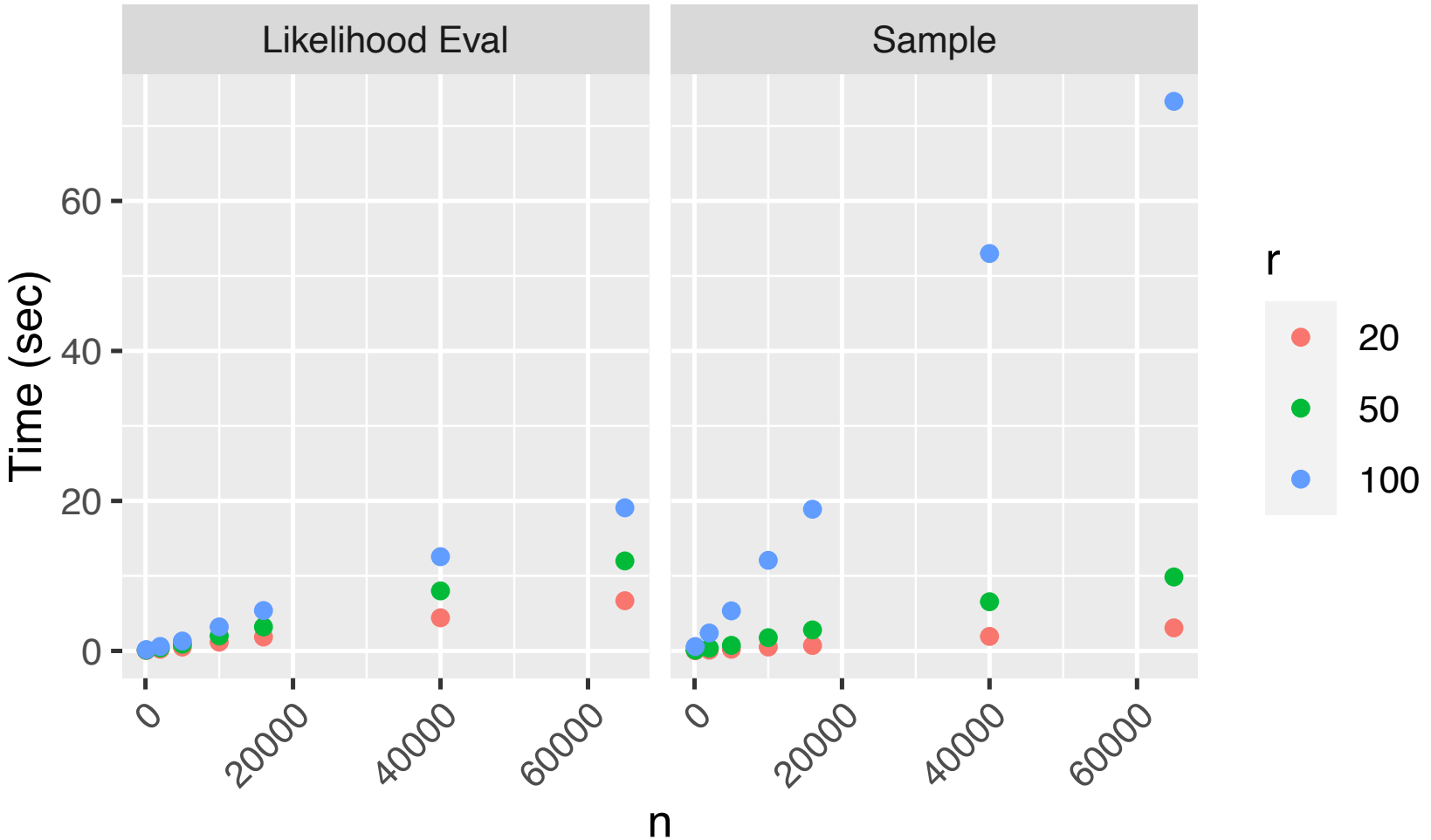
Hierarchical partitioning of domain $S =: S_1$



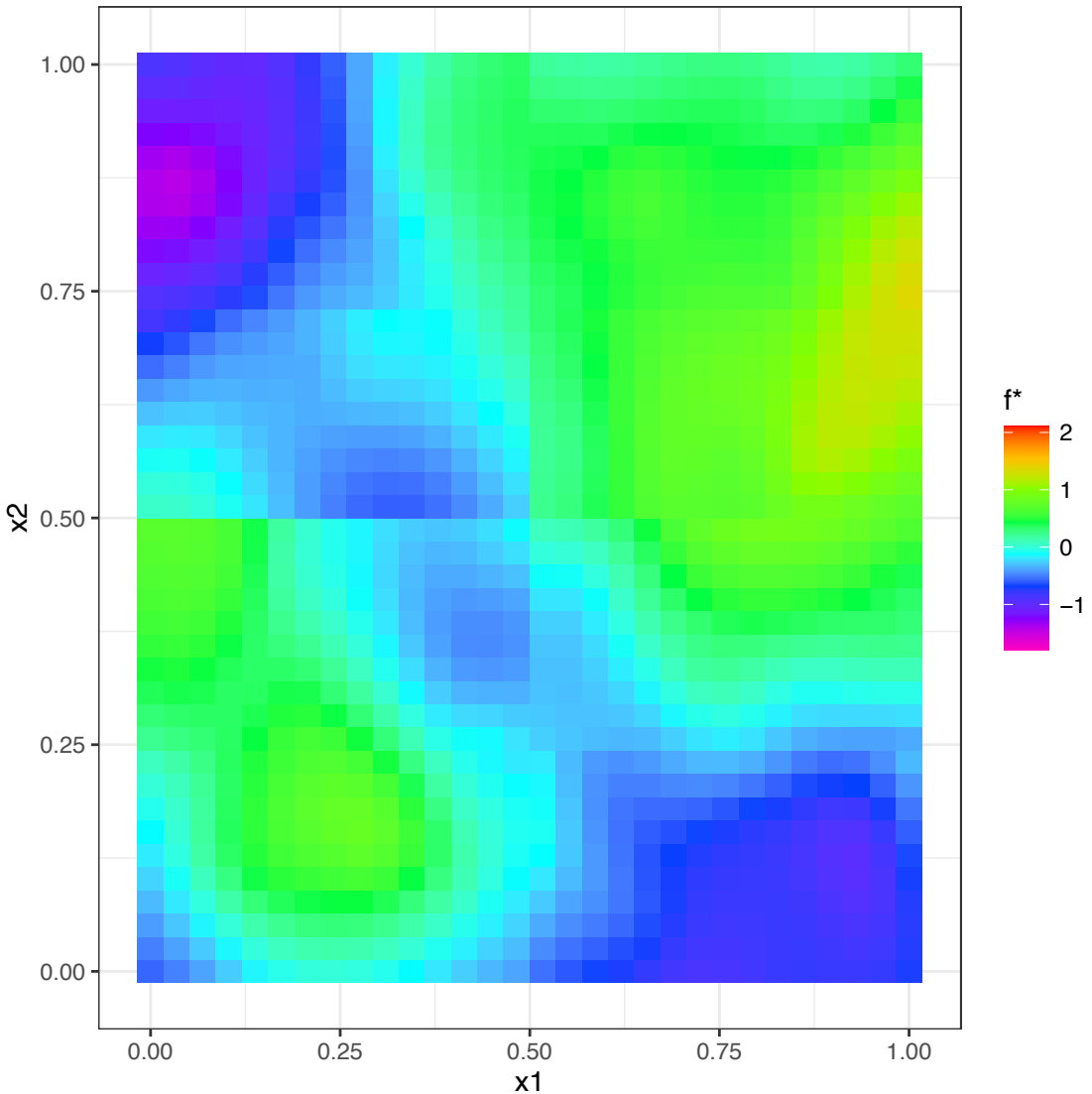
- For each nonleaf node i, find a set of landmark points $\underline{X}_i \subset S_i$
- Definition of k_{h} (informal example):
  - [Same leaf node] $\boldsymbol{x}, \boldsymbol{x}' \in S_8$: $k_{\mathsf{h}}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}')$
  - [One level higher] $\boldsymbol{x} \in S_8$, $\boldsymbol{x}' \in S_9$:
    $k_{\mathsf{h}}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \underline{X}_4) k(\underline{X}_4, \underline{X}_4)^{-1} k(\underline{X}_4, \boldsymbol{x}')$

Recursively Low-Rank Compressed Matrices (RLCMs) were used for kernel approximation in the frequentist GP context in "Linear-Cost Covariance Functions for Gaussian Random Fields" [Chen and Stein 2021].

Timing using different numbers of landmark points



Samples using few landmark points



Upcoming

- My co-author Matthew Wheeler along with Wesley Burr have an **R** package for fast Bayesian GP sampling using H-matrices at `https://github.com/wesleyburr/schnellerGP` (caveat: still very much in the development stage).

- Working on ideas for the potential improvements to sampling using RLCMs with few landmark points.

- If RLCM-based posterior sampling continues to seem promising, will develop an **R** package for Bayesian GP sampling leveraging RLCMs.

Questions?

Kelly R. Moran (krmoran@lanl.gov)

JRSS-B paper link: https://doi.org/10.1111/rssb.12494

This work was done in collaboration with Dr. Matthew Wheeler (NIEHS), and with guidance by Dr. Amy Herring (Duke). It was supported in part by the Department of Energy Computational Science Graduate Fellowship, DE-FG02-97ER25308, the National Institute of Environmental Health Sciences, 20200065DR, and the Laboratory Directed Research and Development program of Los Alamos National Laboratory.