



# Scalable Gaussian Process Inference with Stan

Till Hoffmann 

Harvard T.H. Chan  
School of Public Health

Jukka-Pekka Onnela 

Harvard T.H. Chan  
School of Public Health

---

## Abstract

Gaussian processes (GPs) are sophisticated distributions to model functional data. Whilst theoretically appealing, they are computationally cumbersome except for small datasets. We implement two methods for scaling GP inference in **Stan**: First, a general sparse approximation using a directed acyclic dependency graph. Second, a fast, exact method for regularly spaced data modeled by GPs with stationary kernels using the fast Fourier transform. Based on benchmark experiments, we offer guidance for practitioners to decide between different methods and parameterizations. We consider two real-world examples to illustrate the package. The implementation follows **Stan**'s design and exposes performant inference through a familiar interface. Full posterior inference for ten thousand data points is feasible on a laptop in less than 20 seconds.

*Keywords:* Gaussian process, Fourier transform, sparse approximation, Stan, Python.

---

## 1. Introduction

Gaussian processes (GPs) are flexible non-parametric models for functions with applications in time series analysis (Roberts, Osborne, Ebden, Reece, Gibson, and Aigrain 2013), geospatial statistics (Krig 1951), robotics (Deisenroth, Fox, and Rasmussen 2015), and beyond. More formally, a GP is a distribution over functions  $f(x)$  such that any finite set of  $n$  values  $\mathbf{f} = f(\mathbf{x})$  evaluated at  $\mathbf{x} = \{x_1, \dots, x_n\}$  follows a multivariate normal distribution (Rasmussen and Williams 2006). The distribution is thus fully specified by its mean  $\mu(x) = \mathbb{E}(f(x))$  and covariance (kernel)  $k(x, x') = \text{cov}(f(x), f(x'))$ . A range of problem-specific kernels have been developed, such as squared exponential and Matérn kernels to model local correlations and sinusoidal kernels to capture periodic signals (Duvenaud 2014). In general, evaluating the likelihood requires inverting the covariance matrix  $\mathbf{K}$  obtained by evaluating the kernel  $k$  at all pairs of observed locations  $\mathbf{x}$ . Unfortunately, the computational cost of inverting  $\mathbf{K}$  scales as  $\mathcal{O}(n^3)$ , making GPs prohibitively expensive save for relatively small datasets.

Diverse schemes have been developed to approximate the likelihood for larger datasets, such as low-rank approximations of the covariance matrix (Hensman, Fusi, and Lawrence 2013), nearest-neighbor approximations (Wu, Pleiss, and Cunningham 2022), and Fourier methods (Hensman, Durrande, and Solin 2017). Implementations of these methods are available in Python (Sheffield Machine Learning Group 2012; Ambikasaran, Foreman-Mackey, Greengard, Hogg, and O'Neil 2015) (including the machine learning frameworks **torch** (Gardner, Pleiss, Bindel, Weinberger, and Wilson 2018) and **tensorflow** (Matthews, van der Wilk, Nickson, Fujii, Boukouvalas, León-Villagrà, Ghahramani, and Hensman 2017)), R (Gramacy 2016), Julia

(Fairbrother, Nemeth, Rischard, Brea, and Pinder 2022), and MatLab (Vanhatalo, Riihimäki, Hartikainen, Jylänki, Tolvanen, and Vehtari 2013). Yet, despite its popularity, a library for scalable GP inference is lacking for Stan (Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, and Riddell 2017).

Here, we discuss the implementation of two scalable approaches in Stan which can be easily integrated using Stan’s `#include` directive. First, in section 2, we present an implementation of GPs on directed acyclic graphs which can encode structured dependencies between observations and generalizes nearest-neighbor approximations. Second, in section 3, we use Fourier methods to evaluate the GP likelihood exactly for observations on a regular grid in one and two dimensions. Both implementations are designed to dovetail with Stan’s design philosophy, facilitating their integration into larger models. We consider a benchmark problem and discuss the importance of different parameterizations for performant inference in section 4. Furthermore, we demonstrate the utility of both approaches with two examples: Inferring the density of trees in a 50 ha plot in Panama (Condit, Perez, Aguilar, Lao, Foster, and Hubbell 2019) and predicting passenger numbers on the London Underground network. We summarize our contributions in section 6 and discuss how the package can be employed to build more complex models.

As Stan does not have a package repository, we have published the library as a Python package **gptools-stan** on PyPI. The package includes the Stan library code and provides utility functions to integrate with the popular Stan interface **cmdstanpy**. The library can also be obtained directly from <https://github.com/onnella-lab/gptools>. Extensive technical documentation and examples are available at <https://gptools-stan.readthedocs.org>.

## 2. Gaussian processes with structured dependencies

The joint distribution of observations  $\mathbf{f}$  may be expressed as the product of conditional distributions (which are available in closed form for multivariate normal distributions (Gelman, Carlin, S, Dunson, Vehtari, and Rubin 2013, appendix A1))

$$p(\mathbf{f}) = p(f_1) \prod_{j=2}^n p(f_j \mid f_{j-1}, \dots, f_1). \quad (1)$$

The conditional structure in eq. (1) can be encoded by a directed acyclic graph (DAG) whose nodes represent observations such that a directed edge exists from a node  $j$  to each of its predecessors  $\mathcal{P}_j = \{j-1, \dots, 1\}$ , where the ordering is arbitrary. If two observations do not depend on one another, the corresponding edge can be removed from the DAG to reduce the computational cost. In particular, evaluating each factor of eq. (1) requires inverting a matrix with size equal to the number of predecessors of the corresponding node—a substantial saving if the graph is sparse. For example, nearest-neighbor methods, a special case, reduce the asymptotic runtime to  $\mathcal{O}(nq^3)$  by retaining only edges from each node to at most  $q$  of its nearest predecessors. This approach can yield excellent approximations provided that the neighborhoods are large enough and that the kernel only models local correlations (Wu *et al.* 2022). For example, nearest-neighbor methods are not suitable for periodic kernels but can be approximated by structured dependencies if the period is known.

We implemented a custom distribution in Stan such that a GP with squared exponential kernel on a DAG embedded in a  $p$ -dimensional space can be specified as

```
f ~ gp_graph_exp_quad_cov(loc, x, sigma, length_scale, edges);
```

where `vector[n]` `loc` is the prior mean, `array [n] vector[p]` `x` is an array of node locations in  $p$  dimensions, `real` `sigma` is the marginal scale of the GP, and `real` `length_scale` is the correlation length. The graph is encoded by the edge list `array [,] int edges`, a two-dimensional array of integer node labels comprising predecessors and successors in the first and second row, respectively. For example,  $\{\{1, 2, 3\}, \{2, 3, 4\}\}$  represents the line graph of four nodes. Following Stan’s convention, node labels start at one. Similar distributions are provided for the `matern32` and `matern52` kernels.

### 3. Gaussian processes in Fourier space

If the observation points  $\mathbf{x}$  form a regular grid and the kernel is stationary, i.e.,  $k(x, x') = k(x - x')$ , we can use the fast Fourier transform (FFT) to evaluate the likelihood exactly in  $\mathcal{O}(n \log n)$  time (Rasmussen and Williams 2006, appendix B). For exposition, we consider the continuous Fourier transform here and provide details for the discrete Fourier transform in appendix A. As the Fourier transform is a linear operator and  $f$  is (infinite-dimensional) multivariate normal, the Fourier coefficients

$$\tilde{f}(\xi) = \int_{-\infty}^{\infty} dx \exp(-2\pi i \xi x) f(x)$$

are also multivariate normal. Assuming  $\mu(x) = 0$  for simplicity, the mean of Fourier coefficients is zero and their expected complex-conjugate product at two different frequencies  $\xi$  and  $\xi'$  is

$$\begin{aligned} \mathbb{E}(\tilde{f}(\xi) \overline{\tilde{f}(\xi')}) &= \int_{-\infty}^{\infty} dx dx' \exp(-2\pi i (x\xi - x'\xi')) k(x - x') \\ &= \int_{-\infty}^{\infty} dx' \exp(-2\pi i x'(\xi - \xi')) \int_{-\infty}^{\infty} d\Delta \exp(-2\pi i \Delta \xi) k(\Delta), \end{aligned}$$

where we changed variables to  $x = \Delta + x'$  and factorized the integrals in the second line. The first integral is a representation of the Dirac delta function  $\delta(\xi - \xi')$ , and the second integral is the Fourier transform of the kernel. Fourier coefficients are thus uncorrelated, and, subject to careful bookkeeping, we can evaluate the GP likelihood exactly. In other words, the Fourier transform diagonalizes stationary kernels.

In practice, we need to consider observation points  $\mathbf{x}$  on a regular grid to reap the computational benefits of the Fourier approach by employing the FFT (Press, Teukolsky, Vetterling, and Flannery 2007). This may seem overly restrictive. But, in many settings, data naturally form a regular grid, e.g., financial time series with fixed sampling interval (Hoffmann, Peel, Lambiotte, and Jones 2020), resampled or binned time series (Flaxman, Wilson, Neill, Nickisch, and Smola 2015), or rasterized images (Tipping and Bishop 2002). Contrary to most real-world scenarios, the FFT assumes periodic boundary conditions. This peculiarity can be overcome by padding the domain, as discussed in more detail in section 5.2. We implemented a custom distribution in Stan such that a GP on a grid can be specified as

```
f ~ gp_rfft(loc, cov_rfft);
```

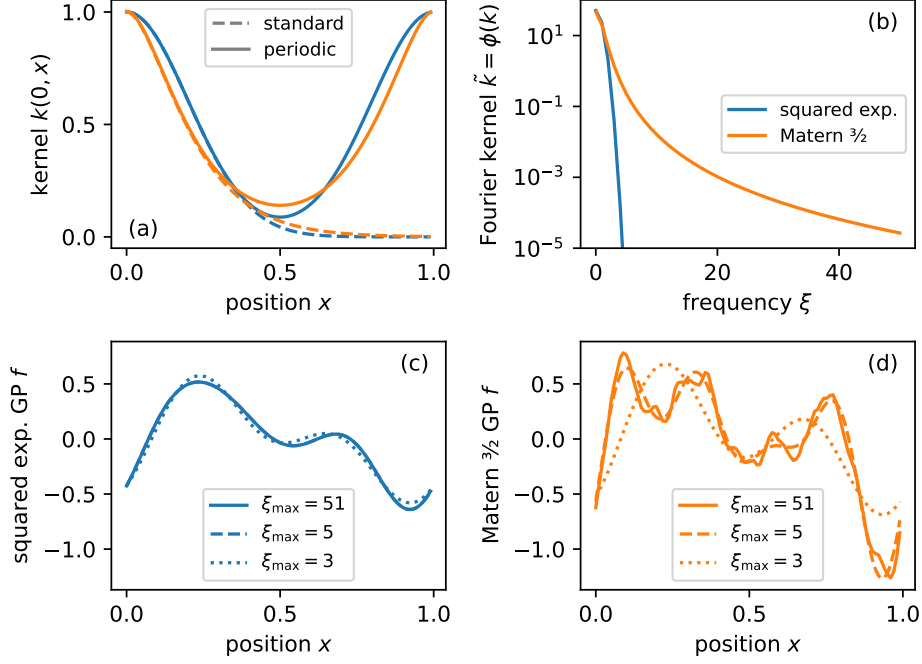


Figure 1: *Gaussian processes can often be captured by a small number of Fourier modes.* Panel (a) shows the periodic and non-periodic (standard) versions of the squared exponential (blue) and Matérn  $3/2$  (orange) kernels with  $\sigma = 1$  and  $\ell = 0.2$ . The two versions are hardly distinguishable for  $x < \ell$ . Padding may need to be introduced if non-periodic signals are modeled with periodic kernels. The power spectrum of the two periodic kernels is shown in panel (b). The squared exponential kernel has negligible power for all but the first few frequencies, explaining why it is often considered too smooth to “represent natural phenomena” (Handcock and Stein 1993). Panels (c) and (d) show realizations of GPs with squared exponential and Matérn  $3/2$  kernels, respectively. Different line styles correspond to approximations with a different number of Fourier modes. For the squared exponential kernel, the number of modes can be reduced by an order of magnitude without substantially affecting realizations. The Matérn kernel requires more modes due to its heavy-tailed power spectrum.

where `vector[n] loc` is the prior mean and `vector[n %/% 2 + 1] cov_rfft` is the real FFT (RFFT) of the kernel evaluated on the grid (`%/%` denotes integer division in `Stan`). The size of `cov_rfft` is smaller than  $n$  because the negative frequency terms are redundant and can be omitted for real signals (Press *et al.* 2007, chapter 12.3). Fortunately, the RFFT of common kernels, such as the squared exponential kernel and Matérn kernels, can be evaluated directly in the Fourier domain (Rasmussen and Williams 2006, chapter 4), as shown in fig. 1 (see appendix B for details). Evaluating the kernel in the Fourier domain also obviates the need for small “nugget” variance or “jitter” typically required for numerical stability (Neal 1997). We thus only need to evaluate one Fourier transform, that of the signal, to evaluate the likelihood. The library provides the following functions to evaluate Fourier-domain kernels

```
gp_periodic_exp_quad_cov_rfft(n, sigma, length_scale, period)
gp_periodic_matern_cov_rfft(n, nu, sigma, length_scale, period)
```

where `n` is the number of grid points, `period` is the size of the domain, and `nu` is the smoothness parameter of the Matérn kernel. `sigma` and `length_scale` have the same meaning as in section 2. Equivalent functions, which we discuss further in section 5.2, are provided for two-dimensional grids. We implemented the Fourier-domain kernels by naively discretizing frequencies. This approach works well if the number of grid points is large and the correlation length is small compared with the size of the domain. More sophisticated methods may be required otherwise (Borovitskiy, Terenin, Mostowsky, and Deisenroth 2020).

## 4. Benchmark and the importance of parameterizations

We consider a simple benchmark problem to study the performance of different methods and compare them with standard Gaussian process inference which inverts the kernel. The model comprises a one-dimensional zero-mean Gaussian process prior with squared exponential kernel and an independent normal observation model with variance  $\kappa^2$ , i.e.,

$$\begin{aligned} \mathbf{f} &\sim \text{MultivariateNormal}(0, \mathbf{K}) \\ \mathbf{y} &\sim \text{Normal}(\mathbf{f}, \kappa^2). \end{aligned} \tag{2}$$

We used a marginal kernel scale  $\sigma = 1$  and unit correlation length  $\ell = 1$  to evaluate the covariance matrix  $K$  on an integer grid, i.e.  $\mathbf{x} = \{0, \dots, n-1\}$ . Employing the **cmdstanpy** interface, we drew 100 posterior samples from a single chain after 100 warmup samples. Warmup samples are used to adapt the sampler for efficient exploration of the posterior. Default values were used for all other parameters. We considered different dataset sizes between  $n = 2^4$  and  $n = 2^{14}$  and allocated a maximum computational budget of one minute for all  $n$ , i.e., the sampler was terminated if it did not complete after 60 seconds.

The mean runtime as a function of dataset size is shown in panels (a) and (b) of fig. 2 for small ( $\kappa = 0.1$ ) and large ( $\kappa = 10$ ) noise scales as solid lines, respectively. As expected, the runtime of the standard approach grows rapidly as  $n$  increases. We observed an empirical runtime scaling of  $n^{\approx 2.5}$  for the standard approach, not dissimilar from the expected asymptotic scaling of  $\mathcal{O}(n^3)$ . Exploring models with more than a few hundred data points is prohibitively expensive—even for this simple setup. For the graph-based approach, we used the five nearest predecessors ( $q = 5$ ) to construct a dependency graph. The method is comparatively slow for

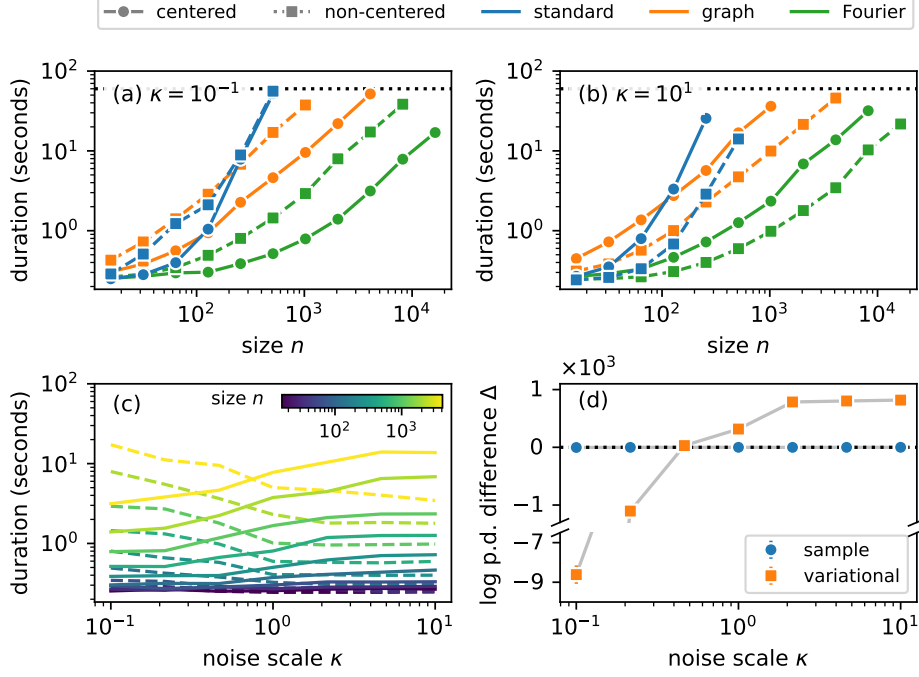


Figure 2: *Different approaches, parameterizations, as well as the informativeness of the data substantially affect runtimes.* Centered parameterizations are preferable when the data are strong (small  $\kappa$ ), and non-centered parameterizations are superior when the data are weak (large  $\kappa$ ), as shown in panels (a) and (b), respectively. Independent of parameterization and dataset size, Fourier methods offer the best performance when observations are regularly spaced. For datasets exceeding a few hundred observations, graph methods are faster than the standard approach which requires inversion of the covariance matrix. The dotted horizontal line represents the maximum computational budget of 60 s. As shown in panel (c) for the Fourier approach, the runtime of the centered and non-centered parameterizations increases and decreases, respectively, as the data become less informative. While different parameterizations are primarily a performance concern for drawing posterior samples, they have important consequences for variational posterior approximations. Panel (d) shows the difference  $\Delta$  in log posterior density (p.d.) between the non-centered and centered parameterization on 20% held-out data for  $n = 1,024$  using the Fourier approach. If the model is fit with a variational mean-field approximation, the non-centered parameterization offers higher log posterior scores than the centered parameterization when the data are weak and vice versa. Bootstrapped standard errors are smaller than the size of markers in panel (d).

small datasets but outperforms the standard approach as  $n$  grows. The Fourier approach has the best performance irrespective of dataset size but is limited to observations on a grid.

The model in eq. (2) employs the natural *centered* parameterization (Papaspiliopoulos, Roberts, and Sköld 2007), i.e., each observation  $y_i$  is independent given the corresponding latent  $f_i$ . This parameterization works well if the data are informative (small  $\kappa$ ) because each observation  $y_i$  constrains the corresponding latent parameter  $f_i$ . The elements of  $\mathbf{f}$  are thus relatively uncorrelated under the posterior, and the Hamiltonian sampler can explore the distribution efficiently (Homan and Gelman 2014).

However, if the data are weak (large  $\kappa$ ), they cannot independently constrain each element of  $\mathbf{f}$  and the GP prior dominates the posterior. The resulting correlation among elements of  $\mathbf{f}$  frustrates the sampler, especially if the correlation length is large. We can overcome this challenge by employing a *non-centered* parameterization such that the parameters of the model are uncorrelated under the prior (Papaspiliopoulos *et al.* 2007). Here, we reparameterize the model in terms of a white noise vector  $\mathbf{z}$  of the same size as  $\mathbf{f}$  and obtain realizations of the GP  $\mathbf{f} = \phi^{-1}(\mathbf{z}, \boldsymbol{\mu}, \mathbf{K})$  using an inverse transform  $\phi^{-1}$  which must be selected carefully to ensure  $\mathbf{f}$  follows the desired distribution. We chose the inverse transform for consistency with the FFT: The forward transform maps to the Fourier domain, and the inverse transform maps to real space. The reparameterized model is

$$\begin{aligned}\mathbf{z} &\sim \text{Normal}(0, 1) \\ \mathbf{f} &= \phi^{-1}(\mathbf{z}, 0, \mathbf{K}) \\ \mathbf{y} &\sim \text{Normal}(\mathbf{f}, \kappa^2).\end{aligned}\tag{3}$$

We implemented the following transforms for the graph-based and Fourier approaches.

```
f = gp_inv_graph_exp_quad_cov(z, loc, x, sigma, length_scale, edges);
f = gp_inv_rfft(z, loc, cov_rfft);
```

where `vector[n]`  $\mathbf{z}$  are the non-centered white noise parameters and all other parameters are as described previously. For the standard method, we implemented the non-centered parameterization as  $\mathbf{f} = \mathbf{L}\mathbf{z}$  (Papaspiliopoulos *et al.* 2007), where  $L$  is the Cholesky decomposition of the covariance matrix  $\mathbf{K}$ .

As shown in panels (a) and (b) of fig. 2, the non-centered parameterization (dashed lines) is more performant than the centered parametrization (solid lines) if the noise scale is large and vice versa. Panel (c) further illustrates the importance of choosing the right parameterization: The runtime differs by up to a factor of five as we vary the noise scale  $\kappa$ . While parameterization is primarily a performance concern for Hamiltonian Monte Carlo samplers, it can have a substantial impact on the predictive ability of models if variational mean-field inference is used. Variational approximations of the posterior tend to assign low probability mass to regions of the parameter space where the full posterior has low mass (Bishop 2006, chapter 10.1). Consequently, variational approximations are too narrow if the posterior is highly correlated, and we expect predictions to be overconfident. To test this hypothesis, we sampled synthetic data with  $n = 1,024$  data points from the prior predictive distribution and fitted the models in eqs. (2) and (3) to 80% of each synthetic dataset variationally. We evaluated the predictive ability of the fitted models by evaluating the log posterior density on the 20% held out GP realizations, i.e.,  $\log p(\mathbf{f}_{\text{test}} \mid \mathbf{y}_{\text{train}})$ . To compare the parameterizations, we approximated the



log posterior difference  $\Delta = \log p(\mathbf{f}_{\text{test}} = \phi^{-1}(\mathbf{z}_{\text{test}}) \mid \mathbf{y}_{\text{train}}) - \log p(\mathbf{f}_{\text{test}} \mid \mathbf{y}_{\text{train}})$  using a Gaussian kernel density estimator (Bishop 2006, chapter 2.5.1). As shown in panel (d) of fig. 2, the non-centered parameterization makes better predictions than the centered parameterization when the noise scale is large and vice versa. We also fitted the two parameterizations by drawing posterior samples using Stan’s Hamiltonian sampler and evaluated the log posterior difference. Different parameterizations did not affect the predictive performance but had a significant impact on runtime.

The higher-frequency terms of smooth GPs have low power, as shown in panel (b) of fig. 1. We can further improve performance of the non-centered parameterization by discarding all but the first few low-frequency terms. This approach is particularly effective for the squared exponential kernel because the power decays rapidly. For the example shown in panel (c), a GP using only the first five Fourier modes is indistinguishable from the GP considering all 51 modes, reducing the dimensionality of the parameter space by an order of magnitude. GPs with Matérn kernels typically require more Fourier modes because the power spectrum has a relatively heavy tail.

## 5. Illustrations

### 5.1. Passengers on the London Underground network

Millions of people use the London Underground network, commonly referred to as the “Tube”, to travel across the city each day (Transport for London 2019). The number of passengers using each station is affected by various factors, including how connected it is and which zone the station is in (the network comprises nine transport zones). In addition to these fixed effects, we also expect passenger numbers to be affected by smooth spatial effects, e.g., due to variability in population density. The spatial effect can naturally be modeled as a GP with structured dependencies induced by the transport network itself. We collected network data from the Transport for London open data API (Transport for London 2022) and obtained the average daily number of entries and exits at each station in 2019 (Transport for London 2019), as shown in panel (a) of fig. 3. The model includes fixed effects for each degree and zone, mildly regularized by half-t priors with two degrees of freedom. We truncated the degree and zone of each station at five and six, respectively, because only few stations exceed these values. The overall number of passengers is captured by a scalar  $\mu$ , and we use the GP to explain any residual effects. A squared exponential kernel was employed for the covariance, and we used a half-t prior for the marginal scale. The correlation length of the kernel is not identifiable if it is smaller than the smallest distance between stations (0.16 km) or larger than the extent of the transportation network (62 km) (Trangucci, Betancourt, and Vehtari 2016). We thus used a log-uniform prior on the interval [0.32 km, 31 km] to suppress extreme length scales. All distances were evaluated in the Ordnance Survey National Grid projection (epsg:27700). A non-centered parameterization was used because the residual effects are not strongly identified by the data after controlling for zone and degree. We used a log-normal observation model (rather than a model for count data) because passenger data are heavy-tailed and reported as daily averages.

```
functions {
  #include gptools/util.stan
```



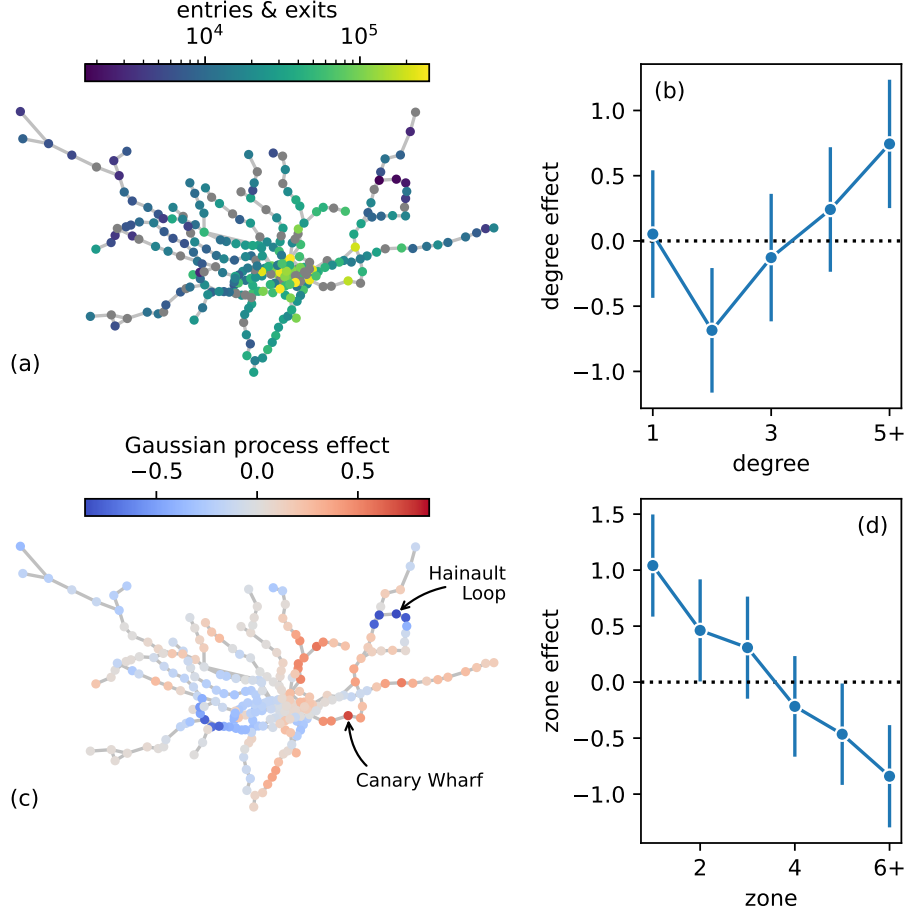


Figure 3: A Gaussian process on the London Underground network identifies idiosyncrasies of transport use. Panel (a) shows the daily average number of entries and exits for each of the 267 stations in 2019 together with the transport network. Grey nodes denote held out data. Panels (b) and (d) show the effect of degree and zone on passenger numbers, respectively. Central stations with larger degree tend to have more passengers. Termini with degree one have uncharacteristically many passengers because they serve as interchanges for longer-distance trains. Panel (c) shows the GP which explains residual variations in passenger volumes after controlling for zone and degree. On the one hand, stations in the Hainault loop have relatively few passengers because they are served infrequently compared with nearby stations. On the other hand, Canary Wharf has surprisingly many entries and exits because it is a busy financial center in London.

```

#include gptools/graph.stan
}

data {
  int num_stations, num_edges, num_zones, num_degrees;
  array [num_stations] vector[2] station_locations;
  array [num_stations] int passengers;
  array [2, num_edges] int edge_index;
  matrix[num_stations, num_zones] one_hot_zones;
  matrix[num_stations, num_degrees] one_hot_degrees;
}

parameters {
  vector[num_stations] z;
  real loc;
  real<lower=0> sigma, kappa;
  real<lower=log(0.32), upper=log(31)> log_length_scale;
  vector[num_zones] zone_effect;
  vector[num_degrees] degree_effect;
}

transformed parameters {
  real length_scale = exp(log_length_scale);
  vector[num_stations] f = gp_inv_graph_exp_quad_cov(
    z, zeros_vector(num_stations), station_locations, sigma,
    length_scale, edge_index);
  vector[num_stations] log_mean = loc + f + one_hot_zones
    * zone_effect + one_hot_degrees * degree_effect;
}

model {
  z ~ std_normal();
  sigma ~ student_t(2, 0, 1);
  zone_effect ~ student_t(2, 0, 1);
  degree_effect ~ student_t(2, 0, 1);
  kappa ~ student_t(2, 0, 1);
  for (i in 1:num_stations) {
    if (passengers[i] > 0) {
      log(passengers[i]) ~ normal(log_mean[i], kappa);
    }
  }
}

```

We fitted the model to 80% of the passenger data withholding 20% of the stations uniformly at random for later evaluation. Held-out data are encoded as -1 in the Stan model. Panel (b) shows the effect of degree on passenger numbers on the log scale. They tend to increase with the degree of a station as they offer passengers a variety of travel options. Termini with degree

one are an exception: Their passenger numbers are uncharacteristically large because they serve as stepping stones to longer-distance travel beyond the Tube network. Unsurprisingly, central stations in zones one to three tend to have more passengers than stations in the suburbs (zones four and above), as shown in panel (d). The GP captures any residuals that cannot be explained by the degree of the station or the zone it is located in, as shown in panel (c). For example, on the one hand, Canary Wharf has the largest residual effect. It is one of London’s financial centers, and the station serves tens of thousands of commuters each day despite being a station without an interchange. On the other hand, stations in the north of the Hainault loop have the largest negative residual effect because the stations are served by only three trains an hour (Transport for London 2020). Passengers divert to nearby stations that are served by twelve trains an hour (Transport for London 2020). Comparing the model with a model without GP effects using the log posterior predictive distribution on held out data, we observe no significant difference after bootstrapping errors. Nevertheless, this example illustrates how our package can be used to easily construct GPs with structured dependencies.

## 5.2. Density of *T. panamensis* on a 50 ha plot in Panama

To illustrate the use of Fourier methods, we consider the density of *T. panamensis* trees during the 2015 census of the 50 ha Barro Colorado plot in Panama (Condit *et al.* 2019). The plot is divided into quadrants of 20 m side length. As shown in panel (a) of fig. 4, the data comprise the frequency of trees within each quadrant, i.e., a matrix of count data with shape (25, 50). The observed tree frequency counts  $\mathbf{y}$  were modeled by a negative-binomial distribution to account for possible overdispersion. We used a latent GP to model the log-mean of this distribution and capture the tree density. We employed a Matérn kernel with smoothness parameter  $\nu = 3/2$ , half-t prior for the marginal scale and overdispersion parameter, and log-uniform prior for the correlation length as in section 5.1. Because the quadrants are regularly spaced, the likelihood can be evaluated exactly using Fourier methods, as discussed in section 3. However, unlike the FFT, trees are not subject to periodic boundary conditions. To mitigate this issue and reduce correlation between opposing sides of the plot, we padded the matrix with ten additional quadrants in each dimension (corresponding to 200 m) resulting in a matrix with shape (35, 60). Despite increasing the number of latent variables by almost 70%, the method is faster than the standard approach which inverts the covariance matrix.

```
functions {
  #include gptools/util.stan
  #include gptools/fft.stan
}

data {
  int num_rows, num_cols, num_rows_padded, num_cols_padded;
  array [num_rows, num_cols] int frequency;
}

parameters {
  matrix[num_rows_padded, num_cols_padded] z;
  real loc;
```

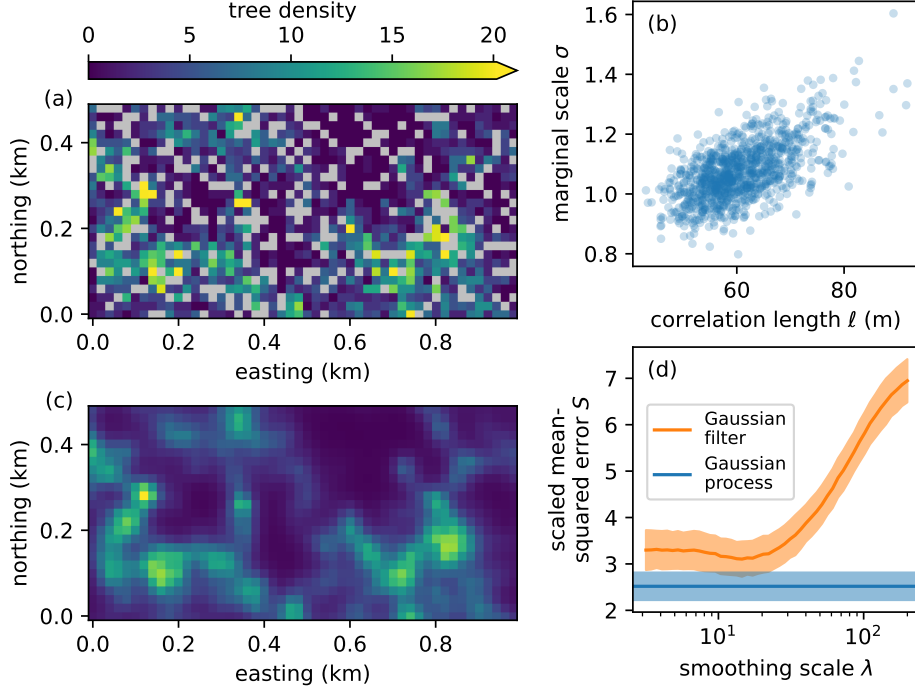


Figure 4: A Gaussian process based on the two-dimensional Fourier transform can accurately predict the frequency of trees. Panel (a) shows the number of *T. panamensis* trees per 20 m quadrant collected during the 2015 census of the 50 ha Barro Colorado plot in Panama (Condit *et al.* 2019) as a heat map. Gray quadrants indicate the 20% held-out test data. Panel (c) shows the posterior median of the expected tree frequency, recovering the held out data and smoothing the empirical frequencies. Posterior samples of the correlation length and marginal scale of the Matérn kernel are shown in panel (b). Correlation length samples are well below the padding (200 m) introduced to mitigate the effect of periodic boundary conditions. Panel (d) shows the scaled mean squared error with bootstrapped errors on the held-out data under the Gaussian process model and a Gaussian filter with variable smoothing scale.

```

real<lower=0> sigma, kappa;
real<lower=log(2), upper=log(28)> log_length_scale;
}

transformed parameters {
  real<lower=0> length_scale = exp(log_length_scale);
  matrix[num_rows_padded, num_cols_padded %% 2 + 1] rfft2_cov =
    gp_periodic_matern_cov_rfft2(1.5, num_rows_padded, num_cols_padded,
      sigma, [length_scale, length_scale]',
      [num_rows_padded, num_cols_padded]');
  matrix[num_rows_padded, num_cols_padded] f = gp_inv_rfft2(
    z, rep_matrix(loc, num_rows_padded, num_cols_padded), rfft2_cov);
}

model {
  to_vector(z) ~ std_normal();
  loc ~ student_t(2, 0, 1);
  sigma ~ student_t(2, 0, 1);
  kappa ~ student_t(2, 0, 1);
  for (i in 1:num_rows) {
    for (j in 1:num_cols) {
      if (frequency[i, j] >= 0) {
        frequency[i, j] ~ neg_binomial_2(exp(f[i, j]), 1 / kappa);
      }
    }
  }
}

```

We fitted the model to 80% of the quadrants chosen uniformly at random, withholding the remainder for evaluation. Despite the noisy, masked observations, the model was able to learn a smooth estimate of the density of trees, as shown in panel (c). The posterior median of the correlation length of 60 m was well below the padding of 200 m introduced to attenuate the effect of periodic boundary conditions, as shown in panel (b). We used a scaled mean-squared error (SMSE) to evaluate the model on the held out data and compare it with the simpler approach of smoothing the data with a two-dimensional Gaussian filter. The SMSE is

$$S(\mathbf{y}, \hat{\mathbf{y}} = \exp \hat{\mathbf{f}}) = \frac{1}{m} \sum_{j=1}^m \frac{(y_i - \exp \hat{f}_i)^2}{\max(y_i, 1)},$$

where the sum is over  $m$  test points and  $\hat{\mathbf{f}}$  is the posterior median of the latent GP. We divided each term by the observed count  $y_i$  (or one if the count was zero) to ensure the measure was not dominated by large counts because the sampling variance of a Poisson count process (without overdispersion) is equal to its mean.

A simple method to estimate the number of trees in held-out quadrants is to apply a Gaussian filter to the data and compare the two methods. The Gaussian filter estimate is

$$\hat{\mathbf{y}}_\lambda = \frac{\mathbf{g}_\lambda * (\mathbf{b} \circ \mathbf{y})}{\mathbf{g}_\lambda * \mathbf{b}},$$

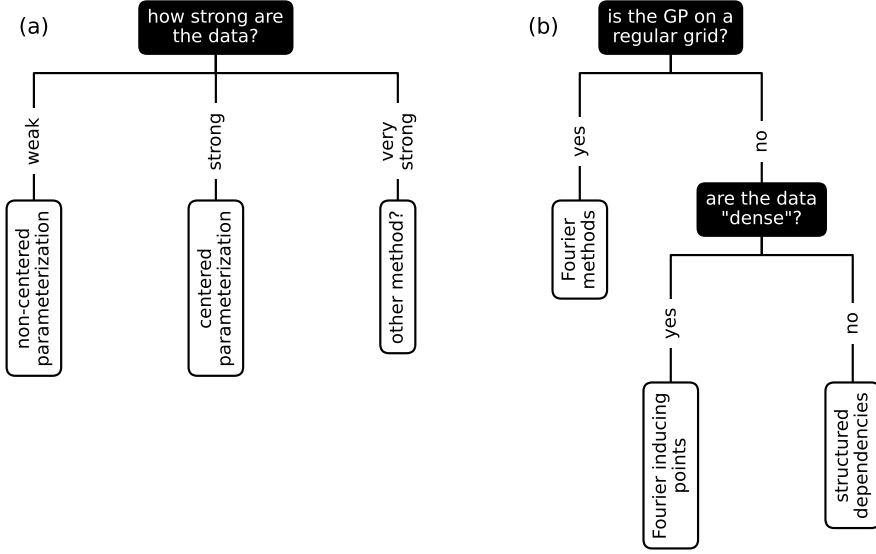


Figure 5: *The optimal approach and parameterization depends on the data.* Panel (a) assists in with the choice of parameterization (or considering alternative methods if the data are very strong). Choosing an appropriate method or approximation is illustrated in panel (b) depending on the structure of the data.

where  $*$  denotes convolution,  $\circ$  denotes the elementwise product,  $\mathbf{g}_\lambda$  is a Gaussian filter with smoothing scale  $\lambda$ , and  $\mathbf{b}$  is the binary mask indicating which data are available for training. Gaussian filters “blur” the data locally such that adjacent elements of the smoothed signal can inform one another (Lindeberg 1990). For large  $\lambda$ , estimates are approximated by the sample mean, and, for small  $\lambda$ , they are dominated by local noise. Panel (d) of fig. 4 shows the SMSE for the Gaussian filter as a function of smoothing scale and the SMSE achieved by the GP model. The latter achieves a lower SMSE than the former for all smoothing scales, illustrating the utility of GPs for modeling spatial effects. Unlike in section 5.1, it was not possible to use the posterior predictive distribution for evaluation because the Gaussian filter is not a generative model.

## 6. Discussion

We implemented two popular approaches for scaling GPs to larger datasets in Stan: The sparse approximation with structured dependencies discussed in section 2 and the exact Fourier approach in section 3 which is applicable to data on a grid. For centered parameterizations, the likelihood can be evaluated or approximated directly. For non-centered parameterizations, we sample standard normal random variables  $\mathbf{z}$  and use the inverse transform to obtain a GP sample  $\mathbf{f} = \phi^{-1}(\mathbf{z})$ .

Given different parameterizations and approaches, which should be used in practice? As discussed in section 4 and shown in panel (a) of fig. 5, a non-centered parameterization is appropriate if the data are weak, and a centered parameterization is preferable if the data

are strong. Choosing the right parameterization ensures parameters are relatively uncorrelated under the posterior distribution which accelerates inference. For variational mean-field approximations, choosing the right parameterization is even more important: It affects the quality of the approximation, as discussed in section 4. Most variational approaches use a centered parameterization (Hensman *et al.* 2013; Wu *et al.* 2022), and their approximations may be improved by considering non-centered parameterizations. If the data are very strong, the benefits of GPs may be outweighed by their complexity because the likelihood dominates the GP prior. If in doubt, we suggest using a non-centered parameterization, as we have done in section 5, because GPs are typically employed when the data are not sufficiently informative for simpler approaches to succeed.

One the one hand, if data are spaced irregularly, sparse approximations discussed in section 2 are appropriate, especially if the density of observation points is inhomogeneous. On the other hand, if the observation points form a regular grid, Fourier methods discussed in section 3 can evaluate the exact likelihood rapidly and should be preferred, as shown in panel (b). Performance can be further improved by employing a non-centered parameterization and discarding high-frequency terms. Padding may be required to attenuate the effect of periodic boundary conditions inherent to the fast Fourier transform. Fourier methods may also be appropriate if the density of observation points is relatively homogeneous. In particular, we may consider a latent GP  $\mathbf{g}$  on a grid and use it to predict the GP of interest  $\mathbf{f}$  at each observation point, i.e.,

$$p(\mathbf{f} \mid \mathbf{g}) = \prod_{j=1}^n p(f_j \mid \mathbf{g}).$$

This method reduces the computational cost because elements of  $\mathbf{f}$  are conditionally independent given  $\mathbf{g}$  at the regularly spaced “inducing points” (Hensman *et al.* 2013).

We hope that our library and the illustrations in section 5 will accelerate the development of models employing GPs in Stan. Integrating GP approximations with Stan’s ecosystem, rather than developing a bespoke GP library, will allow practitioners to leverage the framework’s flexibility and the shared knowledge of the engaged Stan community.

## Computational details

The results in this paper were obtained using Python 3.10.0, **cmdstanpy** 1.0.8, and **cmdstan** 2.30.1. All experiments were run on a single core of a 2020 MacBook Pro with an Apple Silicon M1 chip and 16 GB of RAM.

## References

- Ambikasaran S, Foreman-Mackey D, Greengard L, Hogg DW, O’Neil M (2015). “Fast Direct Methods for Gaussian Processes.” *IEEE Trans. Pattern Anal. Mach. Intell.*, **38**, 252–265. doi:10.1109/TPAMI.2015.2448083.
- Bishop CM (2006). *Pattern recognition and machine learning*. Springer.
- Borovitskiy V, Terenin A, Mostowsky P, Deisenroth M (2020). “Matérn Gaussian Processes on Riemannian Manifolds.” In *Adv. Neural. Inf. Process. Syst.*, volume 33, pp. 12426–12437.



- Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). “Stan: A Probabilistic Programming Language.” *J. Stat. Softw.*, **76**(1), 1–32. doi:[10.18637/jss.v076.i01](https://doi.org/10.18637/jss.v076.i01).
- Condit R, Perez R, Aguilar S, Lao S, Foster R, Hubbell S (2019). “Complete data from the Barro Colorado 50-ha plot: 423,617 trees, 35 years.” Dryad Data Platform. doi:[10.15146/5xcp-0d46](https://doi.org/10.15146/5xcp-0d46).
- Deisenroth MP, Fox D, Rasmussen CE (2015). “Gaussian Processes for Data-Efficient Learning in Robotics and Control.” *IEEE Trans. Pattern Anal. Mach. Intell.*, **37**(2), 408–423. doi:[10.1109/TPAMI.2013.218](https://doi.org/10.1109/TPAMI.2013.218).
- Duvenaud DK (2014). *Automatic Model Construction with Gaussian Processes*. Ph.D. thesis, University of Cambridge.
- Fairbrother J, Nemeth C, Rischard M, Brea J, Pinder T (2022). “**GaussianProcesses.jl**: A Nonparametric Bayes Package for the Julia Language.” *J. Stat. Softw.*, **102**, 1–36. doi:[10.18637/jss.v102.i01](https://doi.org/10.18637/jss.v102.i01).
- Flaxman S, Wilson A, Neill D, Nickisch H, Smola A (2015). “Fast Kronecker Inference in Gaussian Processes with non-Gaussian Likelihoods.” In *Int. Conf. Mach. Learn.*, volume 37, pp. 607–616.
- Gardner JR, Pleiss G, Bindel D, Weinberger KQ, Wilson AG (2018). “**GPpyTorch**: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration.” In *Adv. Neural. Inf. Process. Syst.*, volume 31.
- Gelman A, Carlin JB, S SH, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian Data Analysis*. Chapman & Hall/CRC.
- Gramacy RB (2016). “**laGP**: Large-Scale Spatial Modeling via Local Approximate Gaussian Processes in R.” *J. Stat. Softw.*, **72**(1), 1–46. doi:[10.18637/jss.v072.i01](https://doi.org/10.18637/jss.v072.i01).
- Handcock MS, Stein ML (1993). “A Bayesian Analysis of Kriging.” *Technometrics*, **35**(4), 403–410. doi:[10.1080/00401706.1993.10485354](https://doi.org/10.1080/00401706.1993.10485354).
- Hensman J, Durrande N, Solin A (2017). “Variational Fourier Features for Gaussian Processes.” *J. Mach. Learn. Res.*, **18**(1), 5537–5588. doi:[10.5555/3122009.3242008](https://doi.org/10.5555/3122009.3242008).
- Hensman J, Fusi N, Lawrence ND (2013). “Gaussian Processes for Big Data.” In *Uncertainty Artif. Intell.*, volume 29, pp. 282–290. doi:[10.5555/3023638.3023667](https://doi.org/10.5555/3023638.3023667).
- Hoffmann T, Peel L, Lambiotte R, Jones NS (2020). “Community detection in networks without observing edges.” *Sci. Adv.*, **6**(4), eaav1478. doi:[10.1126/sciadv.aav1478](https://doi.org/10.1126/sciadv.aav1478).
- Homan MD, Gelman A (2014). “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” *J. Mach. Learn. Res.*, **15**(1), 1593–1623. doi:[10.5555/2627435.2638586](https://doi.org/10.5555/2627435.2638586).
- Krige DG (1951). “A statistical approach to some basic mine valuation problems on the Witwatersrand.” *J. South Afr. Inst. Min. Metall.*, **52**(6), 119–139. doi:[10.10520/AJA0038223X\4792](https://doi.org/10.10520/AJA0038223X\4792).

- Lindeberg T (1990). “Scale-space for discrete signals.” *Trans. Pattern Anal. Mach. Intell.*, **12**(3), 234–254. doi:[10.1109/34.49051](https://doi.org/10.1109/34.49051).
- Matthews AGdG, van der Wilk M, Nickson T, Fujii K, Boukouvalas A, León-Villagrà P, Ghahramani Z, Hensman J (2017). “**GPflow**: A Gaussian process library using **TensorFlow**.” *J. Mach. Learn. Res.*, **18**(40), 1–6.
- Neal RM (1997). “Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification.” *Technical Report 9702*, University of Toronto.
- Papaspiliopoulos O, Roberts GO, Sköld M (2007). “A General Framework for the Parametrization of Hierarchical Models.” *Stat. Sci.*, **22**(1), 59–73. doi:[10.1214/088342307000000014](https://doi.org/10.1214/088342307000000014).
- Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.
- Rasmussen CE, Williams CKI (2006). *Gaussian Processes for Machine Learning*. MIT Press. doi:[10.7551/mitpress/3206.001.0001](https://doi.org/10.7551/mitpress/3206.001.0001).
- Roberts S, Osborne M, Ebden M, Reece S, Gibson N, Aigrain S (2013). “Gaussian processes for time-series modelling.” *Philos. Trans. R. Soc. A*, **371**(1984), 20110550. doi:[10.1098/rsta.2011.0550](https://doi.org/10.1098/rsta.2011.0550).
- Sheffield Machine Learning Group (2012). “**GPpy**: A Gaussian process framework in python.” <https://github.com/SheffieldML/GPy>.
- Tipping M, Bishop C (2002). “Bayesian Image Super-Resolution.” In *Adv. Neural. Inf. Process. Syst.*, volume 15.
- Trangucci R, Betancourt M, Vehtari A (2016). “Prior formulation for Gaussian process hyperparameters.” In *Practical Bayesian Nonparametrics Workshop, Adv. Neural. Inf. Process. Syst.*
- Transport for London (2019). “London Underground passenger counts data.” URL <http://crowding.data.tfl.gov.uk/>.
- Transport for London (2020). “Central Line Working Timetable.” URL <https://content.tfl.gov.uk/cen-wtt-70.pdf>.
- Transport for London (2022). “Transport for London Unified API.” URL <https://api.tfl.gov.uk>.
- Vanhatalo J, Riihimäki J, Hartikainen J, Jylänki P, Tolvanen V, Vehtari A (2013). “**GPstuff**: Bayesian Modeling with Gaussian Processes.” *J. Mach. Learn. Res.*, **14**, 1175–1179.
- Wu L, Pleiss G, Cunningham JP (2022). “Variational nearest neighbor Gaussian process.” In *Int. Conf. Mach. Learn.*, volume 162, pp. 24114–24130.

## A. Independence of discrete Fourier coefficients

In section 3, we considered the properties of the continuous Fourier transform of Gaussian processes. In practice, we always collect data at discrete points, and we consider the statistical properties of discrete Fourier coefficients under a Gaussian process prior here. The discrete Fourier transform of a signal  $\mathbf{f}$  is

$$\tilde{f}_\xi = \sum_{j=0}^{n-1} \exp\left(-\frac{2\pi i \xi j}{n}\right) f_j,$$

where  $\xi$  is the (discrete) frequency and  $n$  is the number of observations. The expected complex conjugate product at two frequencies  $\xi$  and  $\xi'$  is

$$\begin{aligned} \mathbb{E}\left(\tilde{f}_\xi \overline{\tilde{f}_{\xi'}}\right) &= \sum_{j=0}^{n-1} \sum_{j'=0}^{n-1} \exp\left(-\frac{2\pi i}{n}(j\xi - j'\xi')\right) k(j - j') \\ &= \sum_{j'=0}^{n-1} \exp\left(-\frac{2\pi i j'}{n}(\xi - \xi')\right) \sum_{\Delta=-j'}^{n-1-j'} \exp\left(-\frac{2\pi i \Delta l}{n}\right) k(\Delta), \end{aligned}$$

where we have used the same change of variables as in section 3. As the summands of the inner sum are  $n$ -periodic, we may shift the limits of summation to  $[0..n-1]$  without changing the sum. We obtain

$$\mathbb{E}\left(\tilde{f}_\xi \overline{\tilde{f}_{\xi'}}\right) = n\delta_{\xi\xi'} \tilde{k}_\xi,$$

where  $\delta$  is the Kronecker delta. Fourier coefficients of different frequencies are thus independent and have variance  $n\tilde{k}_\xi$ .

## B. Kernels in the real and Fourier domains

### B.1. Squared exponential kernel

The non-periodic squared exponential kernel is defined as

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right),$$

where  $\sigma$  is the marginal scale and  $\ell$  is the correlation length. Its discrete power spectrum on a periodic domain of size  $L$  is

$$\tilde{k}(\xi) = \frac{\sqrt{2\pi}n\sigma^2\ell}{L} \exp\left(-2\left(\frac{\pi\xi\ell}{L}\right)^2\right),$$

where  $n$  is the number of grid points.

## B.2. Matérn kernel

The non-periodic Matérn kernel is defined as

$$k_\nu(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \zeta^\nu K_\nu(\zeta),$$

$$\text{where } \zeta = \frac{\sqrt{2\nu} |x - x'|}{\ell}$$

is a rescaled distance,  $\nu$  is a smoothness parameter,  $\Gamma$  denotes the gamma function,  $|x - x'|$  is the Euclidean distance between  $x$  and  $x'$ , and  $K_\nu$  denotes the modified Bessel function of the second kind. For  $\nu = 3/2$  and  $\nu = 5/2$ , the kernel simplifies to

$$k_{3/2} = \sigma^2 \left( 1 + \frac{\sqrt{3} |x - x'|}{\ell} \right) \exp \left( -\frac{\sqrt{3} |x - x'|}{\ell} \right)$$

$$k_{5/2} = \sigma^2 \left( 1 + \frac{\sqrt{5} |x - x'|}{\ell} + \frac{5 |x - x'|^2}{3\ell^2} \right) \exp \left( -\frac{\sqrt{5} |x - x'|}{\ell} \right).$$

It reduces to the Laplace kernel for  $\nu = 1/2$ . Its discrete power spectrum on a periodic domain of size  $L$  is

$$\tilde{k}(\xi) = \sigma^2 \frac{n\ell}{L} \left( \frac{2\pi}{\nu} \right)^{p/2} \frac{\Gamma(\nu + \frac{p}{2})}{\Gamma(\nu)} \left( 1 + \frac{2(\pi\ell\xi)^2}{\nu L^2} \right)^{-(\nu+p/2)},$$

where  $p$  is the dimensionality of the space.

## Affiliation:

Till Hoffmann, Jukka-Pekka Onnela

T.H. Chan School of Public Health

E-mail: [thoffmann@hsph.harvard.edu](mailto:thoffmann@hsph.harvard.edu), [onnela@hsph.harvard.edu](mailto:onnela@hsph.harvard.edu)

URL: <https://tillahoffmann.github.io/>, <https://www.hsph.harvard.edu/onnela-lab/>