A&A, 699, A42 (2025)
https://doi.org/10.1051/0004-6361/202554518
© The Authors 2025

Astronomy & Astrophysics

gallifrey: JAX-based Gaussian process structure learning for astronomical time series

C. Boettner[1,2,⋆]

[1] Kapteyn Astronomical Institute, University of Groningen, Landleven 12 (Kapteynborg, 5419), 9747 AD Groningen, The Netherlands
[2] GELIFES Institute, University of Groningen, Nijenborgh 7, 9747 AG Groningen, The Netherlands

Received 13 March 2025 / Accepted 21 May 2025

ABSTRACT

Context. Gaussian processes (GPs) have become a common tool in astronomy for analysing time series data, particularly in exoplanet science and stellar astrophysics. However, choosing the appropriate covariance structure for a GP model remains a challenge in many situations, limiting model flexibility and performance.
Aims. This work provides an introduction to recent advances in GP structure learning methods, which enable the automated discovery of optimal GP kernels directly from the data, with the aim of making these methods more accessible to the astronomical community.
Methods. We present gallifrey, a JAX-based Python package that implements a sequential Monte Carlo algorithm for Bayesian kernel structure learning. This approach defines a prior distribution over kernel structures and hyperparameters, and efficiently samples the GP posterior distribution using a novel involutive Markov chain Monte Carlo procedure.
Results. We applied gallifrey to common astronomical time series tasks, including stellar variability modelling, exoplanet transit modelling, and transmission spectroscopy. We show that this methodology can accurately interpolate and extrapolate stellar variability, recover transit parameters with robust uncertainties, and derive transmission spectra by effectively separating the background from the transit signal. When compared with traditional fixed-kernel approaches, we show that structure learning has advantages in terms of accuracy and uncertainty estimation.
Conclusions. Structure learning can enhance the performance of GP regression for astronomical time series modelling. We discuss a road map for algorithmic improvements in terms of scalability to larger datasets, so that the methods presented here can be applied to future stellar and exoplanet missions such as PLATO.

Key words. asteroseismology – methods: data analysis – methods: statistical – techniques: photometric – techniques: spectroscopic – planets and satellites: detection

1. Introduction

Exoplanet science and stellar astrophysics are increasingly reliant on the analysis of high-cadence, high-precision time series data from dedicated surveys. Missions like *Kepler* (Borucki et al. 2010) and TESS (Ricker et al. 2015), and the forthcoming PLATO mission (Rauer et al. 2014), generate large quantities of data, requiring robust statistical methodologies for signal extraction and noise characterisation. Gaussian processes (GPs) have, in this context, emerged as a powerful and versatile tool for a wide variety of applications, including transit searches (Crossfield et al. 2016), transit modelling (Gibson et al. 2012; Gibson 2014; Barros et al. 2020), systematics modelling (Foreman-Mackey et al. 2015; Aigrain et al. 2015, 2016), low-resolution space- and ground-based spectroscopy (Evans et al. 2013, 2015, 2017; Ahrer et al. 2022, 2023), stellar variability modelling (Angus et al. 2018; Gillen et al. 2020; Luger et al. 2021; Nicholson & Aigrain 2022), and radial velocity modelling (Aigrain et al. 2012; Haywood et al. 2014). GPs are particularly useful in situations involving correlated noise, where neglecting the correlation can lead to an overestimation of the signal-to-noise ratio and, in turn, to false positive detections (Pont et al. 2006). Thanks to this wide applicability, a variety of software packages have been created for GP modelling, including george (Ambikasaran et al. 2015), tinygp (Foreman-Mackey et al. 2024), and celerite (Foreman-Mackey et al. 2017a), as well as packages that directly incorporate GPs into astronomical modelling, such as juliet and (Espinoza et al. 2019) exoplanet (Foreman-Mackey et al. 2021). For a comprehensive review of GP use in astronomy, see Aigrain & Foreman-Mackey (2023).

A key practical challenge in applying GPs, however, lies in selecting an appropriate kernel function capable of adequately representing the data. Traditional approaches, relying on manual selection from a limited set of kernels and subsequent hyperparameter optimisation, can be subjective and may not fully exploit the potential of GP methods for complex time series modelling. With recent advances in machine learning, there has been significant interest in learning the appropriate covariance structure directly from the data instead.

A fruitful approach is to define a flexible symbolic language over possible kernel structures and combinations, and then to search this space for appropriate kernel structures (Duvenaud et al. 2013; Abdessalem et al. 2017). In one of the early works in this direction, Duvenaud et al. (2013) employed a greedy-search algorithm to find the best-fitting kernel structure in an iterative fashion. This was followed by Saad et al. (2019), who introduced a fully Bayesian framework by defining a prior over possible kernel structures and sampling them using Markov chain Monte Carlo (MCMC) methods. This, in turn, was improved upon by

[⋆] Corresponding author: boettner@astro.rug.nl

Saad et al. (2023), who refined the approach of Saad et al. (2019) by introducing a novel sequential Monte Carlo (SMC) algorithm (Del Moral et al. 2006; Chopin & Papaspiliopoulos 2020), enhancing both performance and speed compared to the pure MCMC version.

The main goal of this work is to make these advances accessible to the exoplanet community. To this end, we introduce `gallifrey`, a Python package specifically designed to facilitate GP time series structure learning. `gallifrey` implements an SMC algorithm based on Saad et al. (2023), incorporating a Bayesian prior over kernel structures and leveraging the computational efficiency of the JAX framework (Bradbury et al. 2018) for both enhanced performance and automatic differentiation capabilities. To facilitate the usage of this framework, we provide the open-source code[1] as well as in-depth documentation, which provides detailed tutorials to create all the figures presented in this work[2].

The paper is structured as follows: in Sect. 2 we provide a short review of GPs and GP regression, introduce the theoretical framework for structure learning, and detail the SMC algorithm for efficient sampling of the GP posterior. In Sect. 3 we introduce the `gallifrey` Python package and discuss its design choices. In Sect. 4 we demonstrate applications of this methodology to a range of stellar variability, transit modelling, and transmission spectroscopy cases. Finally, in Sect. 5 we discuss use cases of the package and outline a road map for future improvements.

2. Methods

2.1. A brief review of Gaussian processes

Gaussian processes are a versatile tool in parameter-independent statistical modelling, and can be understood as a generalisation of the Gaussian probability distribution. By defining a distribution over functions, rather than individual variables, they can be thought of as an infinite-dimensional analogue of a Gaussian distribution. In the following, we give a brief summary of the key equations for GPs and GP regression. This description closely follows the seminal textbook by Rasmussen & Williams (2006).

2.1.1. Definition

A one-dimensional, zero-mean, continuous-time GP is defined through a covariance function $k(x, x')$ as

$$f(x) \sim \mathcal{GP}(0, k(x, x')), \tag{1}$$

where the covariance function is defined as $k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$[3]. A GP is characterised by the fact that for any finite set of sample points $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, the probability distribution of the corresponding function values $\boldsymbol{f}$ is given by a multivariate Gaussian:

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{0}, K(\boldsymbol{x}, \boldsymbol{x})), \tag{2}$$

where K is the covariance matrix constructed by evaluating the kernel function k at points $\boldsymbol{x}$. For a given kernel, a GP spans a space of functions, with associated probabilities for these functions under the GP. In a regression task, the GP can be treated as a prior over these functions, rather than parameters, and can be used for non-parametric inference.

2.1.2. Gaussian process regression

In Bayesian regression of parametric models, priors are typically placed over parameters $\boldsymbol{\theta}$ (e.g. a and b in $a \cdot x + b \cdot x^2$), and a likelihood function $\mathcal{L}$ represents the probability of observing data $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\} = \{x_i, y_i\}_{i=1}^N$ given the model and specific parameter values. The posterior distribution of these parameters is then derived using Bayes's theorem,

$$P(\boldsymbol{\theta}|\mathcal{D}) \;\propto\; \mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}), \tag{3}$$

where $P(\boldsymbol{\theta})$ is the prior over the parameters $\boldsymbol{\theta}$. In contrast, GP regression can be understood as defining the prior directly over functions f. Given a likelihood function and data, the posterior distribution over functions is given by

$$P(f|\mathcal{D}) \;\propto\; \mathcal{L}(\mathcal{D}|f) \cdot P(f), \tag{4}$$

where the prior $P(f)$ is defined by the GP given by Eq. (1). As in the parametric case, the choice of likelihood function is flexible and should be selected to appropriately model the noise in the data. In particular, it is not restricted to a Gaussian likelihood, despite the prior being a GP. However, a number of key relations become analytically tractable when a Gaussian likelihood is assumed.

For a Gaussian likelihood with variance σ_n^2, the joint distribution of observed values $\boldsymbol{y}$ at $\boldsymbol{x}$, and unobserved latent values $\boldsymbol{f}^*$ at new points $\boldsymbol{x}^*$ is given by

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}^* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\text{n}^2 \boldsymbol{I} & K(\boldsymbol{x}, \boldsymbol{x}^*) \\ K(\boldsymbol{x}^*, \boldsymbol{x}) & K(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix}\right). \tag{5}$$

By marginalising over $\boldsymbol{y}$, we obtain the predictive distribution for the latent function values $\boldsymbol{f}^*$ at the locations $\boldsymbol{x}^*$, which is also given by a Gaussian,

$$P\left(\boldsymbol{f}^*|\boldsymbol{x}^*, \boldsymbol{x}, \boldsymbol{y}\right) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{6}$$

where

$$\boldsymbol{\mu} = K\left(\boldsymbol{x}^*, \boldsymbol{x}\right)\left[K(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\text{n}^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y}, \tag{7}$$

and

$$\boldsymbol{\Sigma} = K\left(\boldsymbol{x}^*, \boldsymbol{x}^*\right) - K\left(\boldsymbol{x}^*, \boldsymbol{x}\right)\left[K(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\text{n}^2 \boldsymbol{I}\right]^{-1} K\left(\boldsymbol{x}, \boldsymbol{x}^*\right). \tag{8}$$

In most applications, σ_n^2 will correspond to the known measurement uncertainties of the observations, although it can also be treated as a learnable parameter. The predictive distribution for new observations $\boldsymbol{y}^*$ at locations $\boldsymbol{x}^*$ is given by

$$P\left(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{x}, \boldsymbol{y}\right) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma} + \sigma_\text{n}^2 \boldsymbol{I}). \tag{9}$$

For a set of residuals $\boldsymbol{r} = (\boldsymbol{y}^* - \boldsymbol{\mu})$, and a covariance matrix $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} + \sigma_\text{n}^2 \boldsymbol{I}$, we can define the whitened residuals

$$\boldsymbol{z} = \boldsymbol{U}\boldsymbol{r}, \tag{10}$$

where $\boldsymbol{U}$ is defined by the Cholesky decomposition $\tilde{\boldsymbol{\Sigma}}^{-1} = \boldsymbol{U}^\top \boldsymbol{U}$. From Eq. (9), follows that the distribution of the whitened residuals $\boldsymbol{z}$ is given by the standard Gaussian $\mathcal{N}(0, \boldsymbol{I})$. Furthermore, in the case of a Gaussian likelihood, the marginal likelihood (i.e. Bayesian model evidence) $P(\mathcal{D}) = \int P(\mathcal{D}|f)P(f)\, \mathrm{d}f$ is given by

$$\begin{aligned} \log P(\mathcal{D}) = &- \frac{1}{2}\boldsymbol{y}^\top \left(K(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\text{n}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \\ &- \frac{1}{2} \log\left|K(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\text{n}^2 \boldsymbol{I}\right| - \frac{N}{2} \log 2\pi. \end{aligned} \tag{11}$$

[1] https://github.com/ChrisBoettner/gallifrey

[2] https://chrisboettner.github.io/gallifrey

[3] One can also specify a mean function $m(x) = \mathbb{E}[f(x)]$, but in many applications, the mean function is assumed to be zero for simplicity.

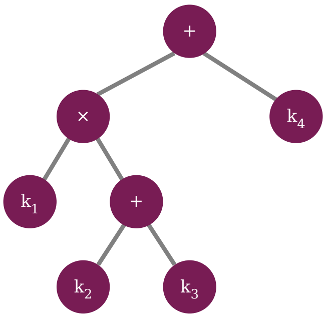


Fig. 1. Example of a combinatorial kernel and its binary tree representation. The kernel, k, is described by a non-linear combination of four elementary kernels, $k = k_1 \cdot (k_2 + k_3) + k_4$. This combination can be visualised as a binary tree, where the leaf nodes correspond to the elementary kernels (atoms), and the internal nodes represent operations.

2.2. Kernel structure learning

While GPs offer closed-form solutions for both the predictive distribution and the marginal likelihood, the challenge lies in determining the appropriate kernel function for a given task. In practise, one often resorts to selecting a kernel from a standard set of pre-defined kernels, and optimising their hyperparameters based on the available data, or sampling them using methods such as MCMC. We provide a list of common kernels and hyperparameter optimisation techniques in Appendix A.

Still, there is often no clear recipe for the choice of kernel. While physical arguments are sometimes a helpful guide (e.g. Foreman-Mackey et al. 2017b argue the Matérn family of kernels is useful for modelling stellar variability), or expected properties of the function space (e.g. smoothness, stationarity, etc.) provide direction, kernel selection is usually performed manually, which is time-consuming and prone to biases.

In contrast, structure learning aims to learn the appropriate kernel structure, alongside its hyperparameters, directly from the data. Several approaches have been developed to address this challenge. A basic insight is that combinations of kernel functions, through addition and multiplication, also yield valid kernel functions. This allows for the systematic exploration of kernel structures by compositionally combining elementary kernels from a predefined library. The structure of the resulting kernel can be visualised as a full binary tree (Fig. 1), where the leaves represent elementary kernels (hereafter referred to as atoms), and the internal nodes represent operations (addition or multiplication). The task of structure learning then becomes identifying a suitable tree structure for the problem at hand.

One approach, introduced by Duvenaud et al. (2013) and further developed by Kim & Teh (2018), uses an iterative, greedy strategy. This process typically begins with a simple base kernel. A GP is then fitted to the data using this initial kernel, and the goodness-of-fit is evaluated. If the fit is deemed suboptimal, a new, more complex kernel structure is explored by either adding an elementary kernel to, or multiplying an elementary kernel with, the existing structure. In the context of the binary tree representation, this corresponds to expanding the tree by adding new leaves or modifying existing ones. The quality of fit for all kernel structures within a given layer of complexity is assessed using an objective function. The structures exhibiting the best fit are then propagated forwards to the next layer, iteratively increasing complexity, until no further improvement in the objective is observed. Common objectives used in this context include the Akaike information criterion AIC $= 2k - 2\ln(\hat{L})$ (Akaike 1998), and the Bayesian information criterion BIC $= k \ln N - 2\ln(\hat{L})$ (Schwarz 1978), where N is the number of parameters, k is the number of hyperparameters in the model and $\hat{L}$ is the likelihood. A detailed implementation of this procedure can this process can be found in Algorithm 2 of Kim & Teh (2018).

While this iterative approach comes with the advantage of easy interpretability and mimics, in a sense, the manual kernel construction process a researcher would undertake, it has a number of drawbacks. Namely, the optimisation can get stuck in local optima during the search process, the optimisation can be unstable, and the procedure is prone to overfitting, especially when the data are sparse.

A more robust alternative is to treat kernel structure learning within a fully Bayesian framework. Rather than specifying a single kernel, we can construct a prior distribution over different kernel structures and their associated hyperparameters. The posterior distribution for the latent function f, considering the kernel structure prior, is then given by

$$P(f|\mathcal{D}) \propto \mathcal{L}(\mathcal{D}|f) \cdot P(f|k, \boldsymbol{\eta}) \cdot P(k, \boldsymbol{\eta}), \tag{12}$$

where $P(k, \boldsymbol{\eta})$ represents the prior distribution over kernel structures k and their hyperparameters $\boldsymbol{\eta}$. The binary tree representation of kernel structures naturally suggests a recipe for defining the kernel structure prior. For instance, each node type (addition, multiplication, atom) and each atom type can be associated with a prior probability. The prior probability for a specific tree structure can be derived by multiplying the probabilities of its constituent nodes and atoms. From this perspective, larger, more complex tree structures, composed of more operations and atoms, inherently have a lower prior probability than simpler trees, which naturally helps mitigate overfitting. We can then, in principle, use Eq. (12) to sample latent functions and calculate posterior probabilities.

This Bayesian framework comes with some computational challenges. Specifically, the kernel structure prior is defined over a discrete and combinatorial space, which makes standard MCMC algorithms, designed for continuous parameter spaces, inapplicable. Furthermore, the problem is trans-dimensional, as the dimensionality of the continuous hyperparameter vector $\boldsymbol{\eta}$ increases with the complexity of the kernel structure. We address these challenges by employing a custom SMC-based sampling algorithm, as detailed by Saad et al. (2023), implemented in `gallifrey` (see Sect. 3).

2.3. Sequential Monte Carlo

The posterior in Eq. (12) can be sampled using any sampling algorithm that is adapted to the peculiarities of this specific problem formulation (i.e. a partially discrete, partially continuous trans-dimensional parameter space). In particular, one could directly use an adapted MCMC algorithm, as was done by Saad et al. (2019). However, with the specific focus on time series modelling, SMC sampling offers some distinct advantages. SMC is closely related to MCMC, and can be understood as a population-based approach to Monte Carlo sampling.

At its heart, an SMC algorithm approximates a target probability distribution by iteratively refining an ensemble of weighted samples, referred to as particles. The goal of SMC is to sample from a sequence of distributions, $\{\pi_t\}_{t=1}^{T}$, where π_t gradually approaches the desired target posterior distribution $P(f|\mathcal{D})$. We began by sampling an initial set of particles from a simpler distribution π_0, usually the prior. Subsequently, these particles were evolved through a series of steps to represent $\pi_1, \pi_2, \ldots, \pi_T \approx P(f|\mathcal{D})$.

The evolution of particles in SMC involves three steps, which are repeated for each transition $\pi_{t-1} \to \pi_t$ (for $t = 1, 2, \dots, T$):

1. Re-weighting: In each iteration, the particles are assigned weights that reflect how well they align with the next distribution in the sequence, relative to the previous one. Particles that match the new distribution more closely receive higher weights, effectively adjusting their importance in the ensemble. Given a set of N particles $\{x_i^{(t-1)}\}_{i=1}^N$ approximating the distribution π_{t-1} with normalised weights $\{w_i^{(t-1)}\}_{i=1}^N$, the updated un-normalised weights are calculated as

$$w_i^{*(t)} = w_i^{(t-1)} \frac{\tilde{\pi}_t\left(x_i^{(t-1)}\right)}{\tilde{\pi}_{t-1}\left(x_i^{(t-1)}\right)}, \tag{13}$$

   where $\tilde{\pi}_t$ and $\tilde{\pi}_{t-1}$ are the un-normalised target densities for distributions π_t and π_{t-1}, respectively. From the un-normalised weights $w_i^{*(t)}$, the normalised weights $w_i^{(t)} = w_i^{*(t)} / \sum_i w_i^{*(t)}$ are calculated.
2. Resampling: After re-weighting, some particles can have very low weights, barely contributing to the approximation, while others have disproportionately high weights, leading to sample degeneracy. Resampling addresses this by eliminating particles with low weights and duplicating those with high weights. The simplest resampling technique is multinomial resampling, where a new set of N particles are drawn from the ensemble $\{x_i^{(t-1)}\}_{i=1}^N$, where the probability of drawing the i-th particle is equal to the weight $w_i^{(t)}$ (with replacement). After the resampling, all particle weights are set to $w_i^{(t)} = 1/N$. Particles with higher weights in the previous step are more likely to be selected multiple times, effectively concentrating the samples in regions of higher probability mass.
3. Rejuvenation: To increase particle diversity after resampling (which introduces duplicates), a rejuvenation step is applied. Typically, this involves applying a number of MCMC steps to move the particles to new locations. This step increases particle diversity and enhances exploration of the sample space, leading to an improved posterior estimate.

In the context of time series structure learning, the sequence of distributions $\{\pi_t\}_{t=1}^T$ can be constructed using data annealing (e.g. Karamanis & Seljak 2025). Starting with the prior distribution π_0 over model parameters (kernel structures and hyperparameters), each step t introduces a new batch of observational data. The distributions π_t then approximates the posterior distribution given the data observed up to step t. We discuss a number of common convergence tests for SMC in Appendix B.

Sequential Monte Carlo has a number of advantages over MCMC for the GP time series structure learning context. The resampling step concentrates more computational effort in high probability regions, thereby increasing the quality of the approximation. Furthermore, GPs are known to scale very unfavourably, exhibiting a $O(N^3)$ scaling behaviour in the number of data points. By sequentially introducing additional data points, the inference can be sped up significantly.

3. The `gallifrey` package

To make GP time series structure learning methods approachable for astronomical applications, especially in the context of exoplanet light curve modelling, we developed the `gallifrey` Python package. This package implements the SMC algorithm with a Bayesian prior over kernel structures, as originally described by Saad et al. (2023) and inspired by the `AutoGP.jl` Julia package. The package is built using the JAX framework (Bradbury et al. 2018) to ensure computational efficiency and scalability. JAX offers native parallelisation capabilities, which can be leverages for an inherently parallel algorithm like SMC. Furthermore, JAX's automatic differentiation simplifies the implementation of gradient-based sampling methods like Hamiltonian Monte Carlo (HMC), which is used for particle rejuvenation.

In this section, we give a general overview of the inner workings of `gallifrey` and some of its design choices. For a more in-depth description of the algorithmic details, see Saad et al. (2023).

3.1. Prior over kernel structures

The kernel structure prior is defined over a space of possible kernel functions, constructed through a grammar of kernel composition. The package employs a library of atomic kernels, $\mathcal{K} = \{k_1, k_2, \dots, k_n\}$, and a set of operators, $O = \{O_1, O_2, \dots, O_m\}$. These components are defined within the `GPConfig` class, allowing for user customisation and extension. Natively, `gallifrey` supports addition and summations operators. A list of implemented atomic kernels can be found in Table A.1.

Kernel structures are sampled recursively. At each node in the kernel tree, a choice is made between selecting an atomic kernel or an operator, based on predefined probabilities p_{kernel} and p_{operator} for each component. The probability of sampling a specific kernel structure K is thus determined by the product of the probabilities of selecting each component in its construction:

$$P(K) = \prod_{\text{node} \,\in\, K} P(\text{component at node}). \tag{14}$$

This prescription naturally favours simple kernel structures over more complicated ones. The kernel construction process is constrained by a maximum tree depth, $D_{\max}$, to ensure computational tractability and prevent overfitting. The prior probabilities p_{kernel} and p_{operator} act as hyperparameters, allowing users to guide the search towards kernel structures of desired complexity.

3.2. Priors for kernel hyperparameters

Each atomic kernel $k \in \mathcal{K}$ is parameterised by a set of hyperparameters, $\boldsymbol{\eta}_k$. To complete the Bayesian model specification, we defined hierarchical priors over these hyperparameters, i.e. $P(k, \boldsymbol{\eta}) = P(\boldsymbol{\eta}|k)P(k)$. In `gallifrey`, we employed weakly informative priors, typically log-normal distributions for positive hyperparameters (e.g., length scales and variances) and logit-normal distributions for bounded hyperparameters (e.g. powers and sigmoid parameters). The variance of the likelihood function σ_n^2 can be specified to a fixed value, if it describes for example observational noise. Alternatively, σ_n^2 can be sampled, in which case the prior is described by an inverse-gamma distribution.

3.2.1. Sequential Monte Carlo inference engine

`gallifrey` employs SMC to perform Bayesian inference over the space of kernel structures and their hyperparameters. The SMC algorithm iteratively refines a population of particles, each representing a candidate kernel structure and parameter set, as more data are sequentially incorporated.

Re-weighting: As new data points are added, particle weights are updated based on their marginal log-likelihood. The weight update for particle i at SMC step t is proportional to the ratio of marginal likelihoods:

$$w_i^{(t)} \propto w_i^{(t-1)} \frac{P(\mathbf{y}_t|\mathbf{x}_t, K_i, \boldsymbol{\eta}_i)}{P(\mathbf{y}_{t-1}|\mathbf{x}_{t-1}, K_i, \boldsymbol{\eta}_i)}, \tag{15}$$

where $\mathbf{X}_t$ and $\mathbf{y}_t$ represent the data up to time t, and K_i and $\boldsymbol{\eta}_i$ are the kernel structure and hyperparameters for particle i. The marginal log-likelihood is given by Eq. (11).

Resampling: To avoid particle degeneracy, a resampling step is performed when the normalised effective sample size (ESS),

$$\text{ESS} = \frac{1}{\sum_{i=1}^{N} \cdot \left(w_i^{(t)}\right)^2}, \tag{16}$$

falls below a predefined threshold (by default $1/2 \cdot N$, where N is the number of particles). `gallifrey` utilises stratified resampling, implemented through the `BlackJAX` package, by default, which reduced the variance of the resulting sample compared to simple multinomial resampling.

Rejuvenation: After the resampling, a number of rejuvenation moves are performed. The rejuvenation involves a hybrid MCMC strategy:

1. Structure MCMC move: A new kernel structure, K_i', is proposed for each particle, i, by modifying the binary tree structure describing the kernel. These moves are accepted or rejected based on a Metropolis-Hastings criterion, considering the kernel prior $P(K)$ and the marginal likelihood $P(\mathbf{y}_t|\mathbf{X}_t, K, \boldsymbol{\theta})$.
2. Parameter HMC move: If the structure move is accepted, the new continuous hyperparameters $\boldsymbol{\eta}_i'$ of the new kernel structure K_i', and optionally a new noise variance σ_n^2, are proposed by applying a number of HMC steps in the continuous parameter space. The HMC algorithm is implemented using the Python sampling package `BlackJAX` (Cabezas et al. 2024).

Once all SMC rounds are complete, the algorithm returns a final sample of particles $\{x_i^T\}_{i=1}^N$ with weights $\{w_i^T\}_{i=1}^N$, which constitute a posterior sample of the GP and can be used for inference. In particular, since each particle corresponds to a GP, with kernel k_i, hyperparameter $\boldsymbol{\eta}_i$, and a predictive distribution given by Eq. (9), the predictive distribution of the whole ensemble is given by a Gaussian mixture model (GMM),

$$P(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} w_i \cdot P(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{x}, \boldsymbol{y}, k_i, \boldsymbol{\eta}_i), \tag{17}$$

where the weights w_i are precisely the importance weights $\{w_i^T\}_{i=1}^N$ returned by the SMC sampler.

4. Example applications

To demonstrate the usefulness of GP structure learning methods, we present a number of example applications in this section.

4.1. Stellar variability

We started the exploration by modelling the stellar variability of the star TIC 10863087, a variable M dwarf star observed by TESS. This target was previously used as a case study in the exoplanet modelling package `exoplanet` (Foreman-Mackey et al. 2021). The star has a mass of approximately 0.4 $M_\odot$ and a rotation period of roughly 0.83 days, rendering it a good test case for evaluating `gallifrey`'s capacity for interpolation and extrapolation of time series data. We retrieved an 8-day observation period from TESS using the `lightkurve` Python package (Lightkurve Collaboration 2018). Following post-processing steps involving outlier removal and thinning, we obtained a time series comprising 814 data points, shown in Fig. 2.

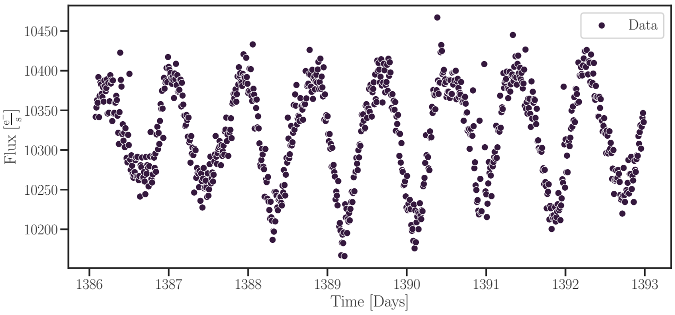


Fig. 2. Example time series of stellar variability. The figure shows a seven-day window of the star TIC 10863087 observed with TESS, after removing outliers and thinning.

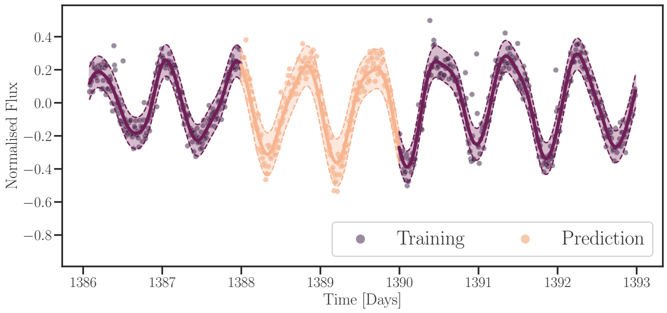


Fig. 3. Demonstration of time series interpolation with `gallifrey`. The figure shows the same data as Fig. 2. The purple data points have been used to train and condition the GP. The yellow region shows the prediction. The central line shows the mean prediction, and the confidence bands correspond to one standard deviation.

To assess `gallifrey`'s interpolation capabilities, we masked a two-day window within the time series and train a GP on the remaining data. We employed standard `gallifrey` settings, including a maximum tree depth of 3, and the linear, period, and radial basis function (RBF) kernels as atomic kernels. The training process utilises the SMC sampler with 64 particles and a linear annealing schedule, where approximately 5% of data points are added in each round. We performed 75 structure MCMC moves and 10 parameter HMC moves per step. The resulting interpolation is shown in Fig. 3, showing a good match to the data. A key advantage of the SMC method is the availability of an entire ensemble of solutions, allowing us to examine individual predictions and the variety of learned kernel structures. A sample of particles including their learned kernel structures are shown in Fig. 5. We observe that `gallifrey` identifies kernel structures that can become quite complex, while maintaining realistic uncertainties.

In addition to interpolation, we evaluated `gallifrey`'s forecasting capabilities. In this experiment, we included the first four

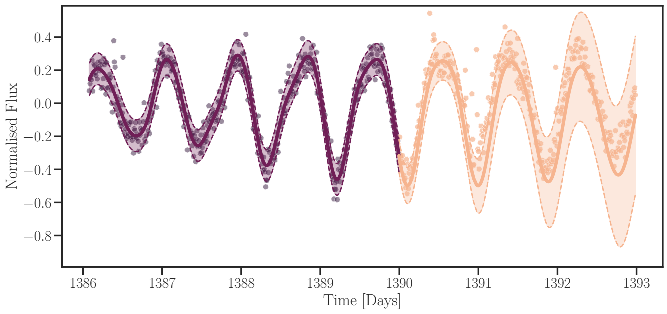


Fig. 4. Same as Fig. 3 but for forecasting rather than interpolation.

days of data for training and attempted to forecast the subsequent three days, maintaining the same setup as before. The resulting forecast is presented in Fig. 4. The forecast demonstrates good accuracy for up to one day. While deviations emerge afterwards, the uncertainty also increases, ensuring that the observed data remains well within the predicted uncertainty bounds.

Using the data annealing schedule inherent in SMC, we can visualise the evolution of the learned structure learning procedure. Figure 6 visualises predictions after consecutive SMC rounds. Initially, with limited data, the prediction is highly uncertain, and the periodic pattern is not captured, leading to a predominantly flat forecast. As more data are incorporated, the prediction progressively improves, and periodic terms become increasingly dominant in the learned kernel structure.

4.2. Transit fitting

As seen in the previous section, the structure learning approach works well for time series interpolation, and this example demonstrates how to leverage these capabilities for transit parameter inference. For light curve fitting, GPs are frequently used as background models, and it is common to either jointly fit the light curve model $M(\boldsymbol{p})$, which is dependent on transit parameters $\boldsymbol{p}$, with the GP (effectively treating the transit model as a deterministic mean function in Eq. (1)), or to utilise the GP for detrending. In this detrending approach, the GP is fit to the data, and the GP mean prediction is subtracted from the data. Often, the transit region is masked out to prevent the GP from fitting the transit signal itself.

When employing learned kernels, simultaneous fitting of the light curve and the GP can introduce a significant risk of overfitting, especially if the GP's features operate on comparable timescales to the transit duration. On the other hand, simply subtracting the estimated GP mean leads to an underestimation of uncertainties and disregards correlations in the noise structure. To mitigate these issues, a hybrid approach can effectively separate the background and transit modelling steps in a two-stage process.

First, we divided the observational dataset $\mathcal{D} = \{\boldsymbol{t}, \boldsymbol{y}\}$ into two disjoint sets: $\mathcal{D}_{\text{trans}} = \{\boldsymbol{t}_{\text{trans}}, \boldsymbol{y}_{\text{trans}}\}$, which contains the data points within the transit window, and $\mathcal{D}_{\text{bg}} = \{\boldsymbol{t}_{\text{bg}}, \boldsymbol{y}_{\text{bg}}\}$, which contains the out-of-transit (background/training) data. We then applied the kernel structure learning procedure, from Sect. 2, exclusively to the background dataset $\mathcal{D}_{\text{bg}}$. This allowed us to determine the optimal ensemble of kernel structures and their associated hyperparameters that best describe the noise and background model, without risking overfitting on the transit signal.

Table 1. Parameters of the simulated planetary system around 16 Cyg B and retrieved transit parameters.

Parameter	True value	Retrieved value
Period (days)	10	–
Impact parameter	0	–
Limb darkening coefficients	[0.25, 0.75]	–
Radius ratio $R_p/R_\star$	0.046013	0.0465 ± 0.0006
Transit duration (days)	0.01728	0.01730 ± 0.00015
Time of mid-transit (days)	49.499602	49.49946 ± 0.00007

Notes. The retrieved values correspond to the mean and standard deviation of the MCMC chain. Parameters without retrieved values were fixed to their true value.

To estimate the transit parameters $\boldsymbol{p}$, we formulated the posterior distribution for $\boldsymbol{p}$ as

$$P(\boldsymbol{p}|\mathcal{D}_{\text{trans}}, \mathcal{D}_{\text{bg}}) \propto P(\boldsymbol{y}_{\text{trans}}|\boldsymbol{t}_{\text{trans}}, \mathcal{D}_{\text{bg}}, \boldsymbol{p}) \cdot P(\boldsymbol{p}), \tag{18}$$

where $P(\boldsymbol{p})$ is the prior distribution over the transit parameters. The likelihood term, $P(\boldsymbol{y}_{\text{trans}}|\boldsymbol{t}_{\text{trans}}, \mathcal{D}_{\text{bg}}, \boldsymbol{p})$, is derived from the predictive distribution of the GP. Specifically, we assume that the observed in-transit data $\boldsymbol{y}_{\text{trans}}$ can be modelled as the sum of a transit model $M(\boldsymbol{t}_{\text{trans}}, \boldsymbol{p})$ and the background function, which is described by the GP. Therefore, the likelihood is given by

$$P(\boldsymbol{y}_{\text{trans}}|\boldsymbol{t}_{\text{trans}}, \mathcal{D}_{\text{bg}}, \boldsymbol{p}) = P(\boldsymbol{y}_{\text{trans}} - M(\boldsymbol{t}_{\text{trans}}, \boldsymbol{p})|\boldsymbol{t}_{\text{trans}}, \mathcal{D}_{\text{bg}}). \tag{19}$$

This likelihood is evaluated using the predictive mixture distribution obtained from the background GP model, given by Eq. (17). This approach leverages the full information on the time series structure learned by the SMC algorithm. By considering the learned covariance structure between data points, this approach also yields a more robust treatment of correlated noise compared to simple detrending methods.

4.2.1. Fitting a transit for a PLATO-like light curve

To demonstrate this approach, we employed a synthetic light curve for the star 16 Cyg B (KIC 12069449), a main-sequence star with $M = 1.04\ M_\odot$, generated using the `Plato Solar-Like Lightcurve Simulator` (Samadi et al. 2019). We simulated an 80-day observation period and introduced a transit signal corresponding to a planet with a radius of 0.5 R_{Jup} (Fig. 7a). The parameters of the planet and star system are summarised in Table 1.

For transit parameter estimation, we selected a time window encompassing a single transit event (Fig. 7b), masked out the transit region, and trained `gallifrey` to learn the background model using the out-of-transit data using an ensemble of 64 particles. Afterwards, we sampled the transit parameter posteriors using the predictive mixture distribution of the learned background model as the likelihood, as described above. We employed the JAX-based light curve modelling package `jaxoplanet` (Hattori et al. 2024) and the sampling package `BlackJAX` (Cabezas et al. 2024). For simplicity, we fixed the planet period, impact parameter, and limb darkening coefficients, and assumed flat priors over the remaining parameters.

The resulting posterior distribution for the transit parameters is shown in Fig. 7c. The planet-to-star radius radio and transit

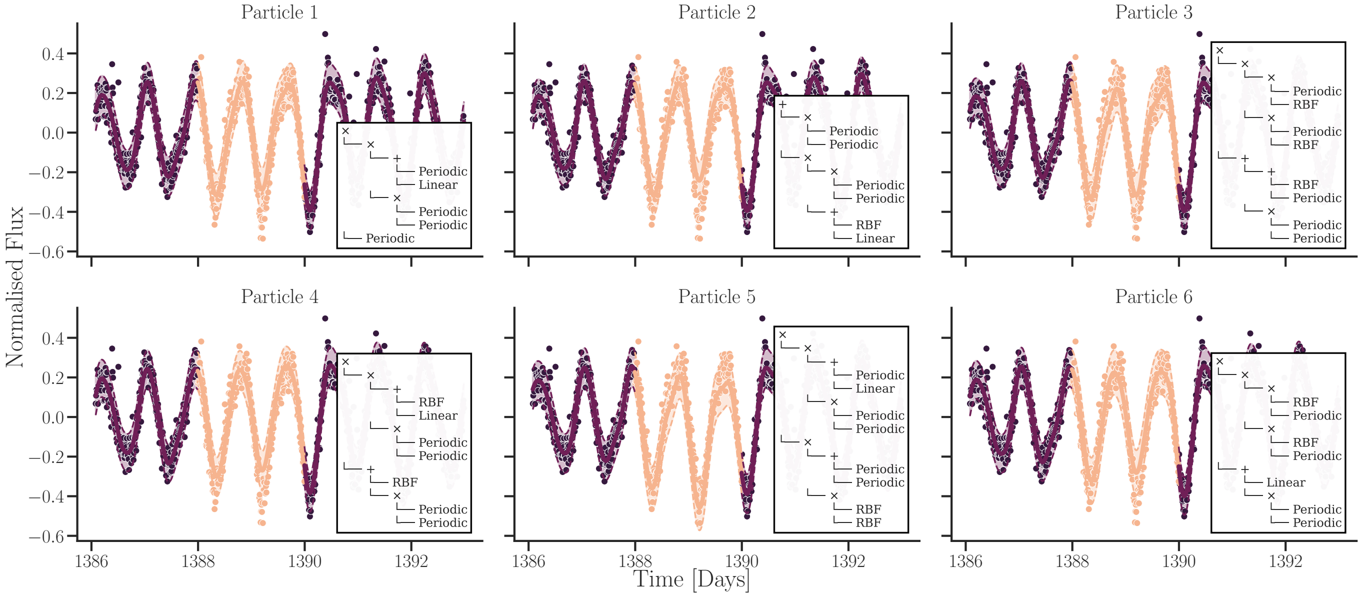


Fig. 5. Sample of predictions for individual particles from the final SMC ensemble used for Fig. 3, including their respective kernel structures. Particles exhibit a variety of kernel structures and produce varying predictions, which leads to a more robust overall prediction when combined. Note that while the depth of every kernel displayed here corresponds to the maximum tree depth, $D_{\max} = 3$, the learned kernels can end up shallower if simpler kernel structures are preferred.

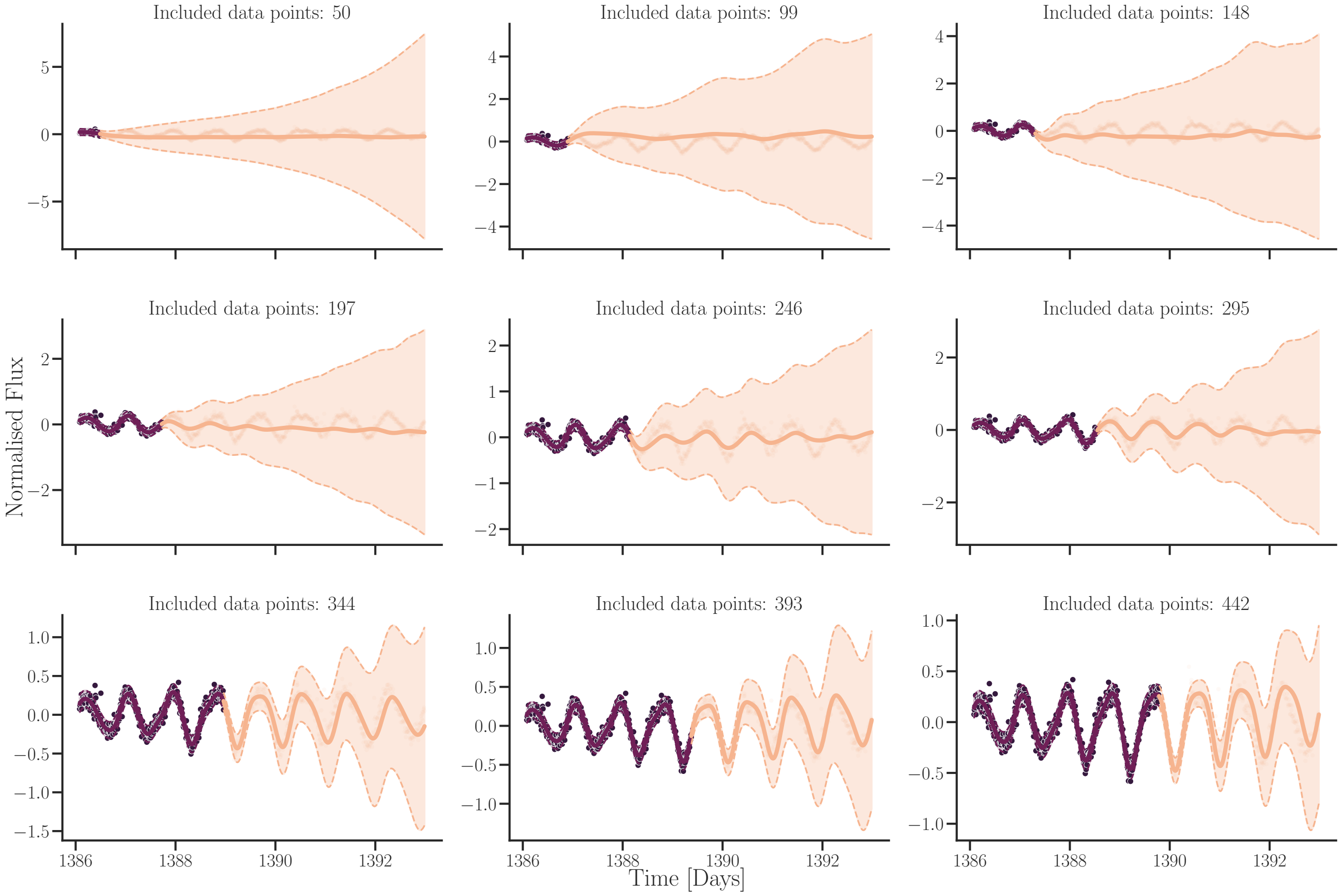
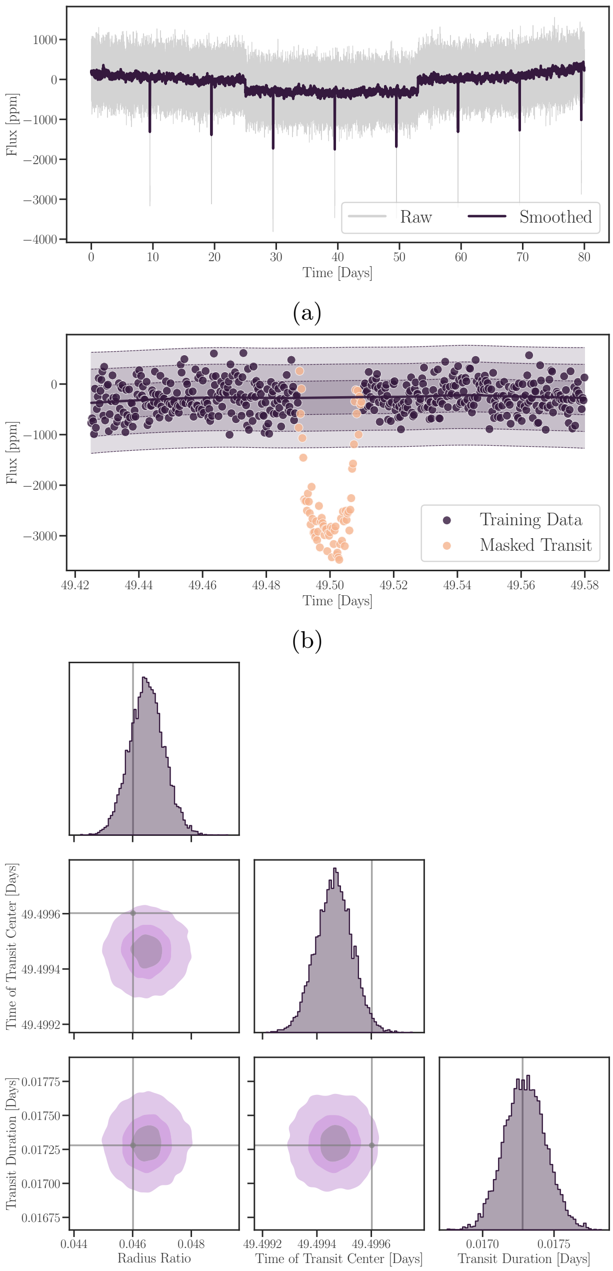


Fig. 6. Snapshots of SMC rounds during the structure learning process for producing the forecast shown in Fig. 4. Purple data points have already been seen during the annealing schedule, while yellow regions are predictions. The algorithm learns more intricate patterns and reduces prediction uncertainty as more data points are added.



(a)

(b)

(c)

Fig. 7. Example of a transit parameter fit for a PLATO-like light curve. (*a*) Simulated 80-day light curve for 16 Cyg B using the `PLATO Solar-like Light-curve Simulator`. The grey curve shows the original light curve, and the purple shows a running mean average. (*b*) Zoom in on a single transit event. The purple data points are used to train and condition the GP, while the yellow data correspond to the masked transit. Also shown is the mean GP prediction, and the one, two, and three standard deviation confidence bands. (*c*) Posterior distribution for the transit parameter, ***p***, using the GP predictive distribution as likelihood. The grey lines represent the true values.

duration is recovered within one standard deviation, while the time of mid-transit is recovered within three standard deviations.

4.2.2. Comparison with a simple RBF kernel

The previous example demonstrated the effectiveness of the method on realistic transit data, making it suitable for current and upcoming transit missions like PLATO. However, the stellar background in that case was relatively simple, and a less complex kernel would likely have sufficed. In this example, we aim to illustrate the difference between a structure learning and using a simple pre-defined kernel, when dealing with stellar variability and correlated noise.

Table 2. Transit parameters of the simulated system in Sect. 4.2.2.

Parameter	True value	RBF kernel	Learned kernel
Period	10	–	–
Time of mid-transit	0.0	–	–
Transit duration	0.2	–	–
Impact parameter	0	–	–
Radius ratio	0.1	0.110 ± 0.009	0.100 ± 0.004
u_1	0.1	<0.398	<0.441
u_2	0.3	<0.613	<0.660

Notes. In the RBF Kernel and Learned Kernel columns, the value for the radius ratio corresponds to the mean and standard deviation of the posterior MCMC chain. The upper limits for u_1 and u_2 correspond to the 95th percentile. Parameters without retrieved values were fixed to their true value. The period, time of mid-transit and transit duration are given in arbitrary units.

We created a synthetic stellar background data with periodic, sinusoidal oscillations, and a non-linear trend. We also added correlated (red) noise, rather than white noise. This type of noise can be encountered, for example in ground-based observations, where the dynamical state of the atmosphere contributes to the observational noise.

Specifically, we employed an autoregressive model of order 1, AR(1), as the background noise model. An AR(1) process is defined by

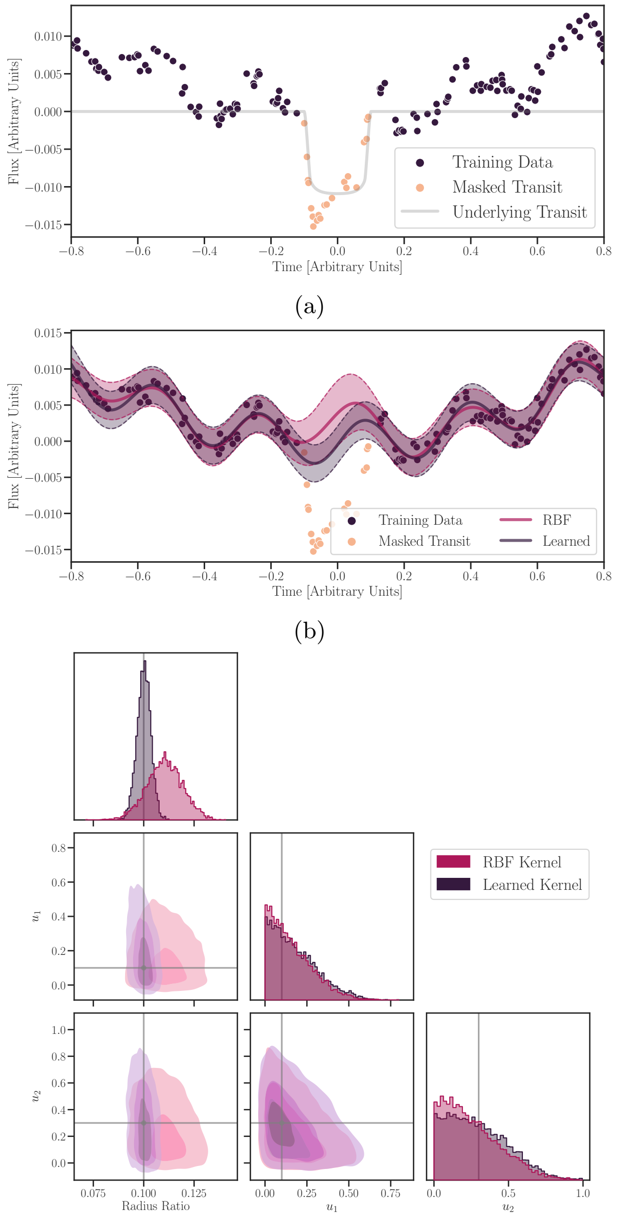
$$\epsilon_t = \phi \cdot \epsilon_{t-1} + n_t, \tag{20}$$

where ϵ_t is the noise at time t, ϕ is the autoregressive coefficient (with $0 \leq \phi < 1$), and n_t is white noise with variance σ_w^2. This model introduces a dependence of the noise at time t on the noise at the previous time step, $t-1$. The AR(1) process is the discrete-time analogue of the Ornstein–Uhlenbeck process, which has a covariance function given by $k(x, x') = \sigma^2 \exp(-|x - x'|/l)$, where l is a length scale parameter related to ϕ. Previous work (e.g. Pont et al. 2006) has highlighted that neglecting this type of correlated noise can lead to biased estimates of transit parameters.

We added a transit signal to this synthetic data using `jaxoplanet`, the transit parameters are given in Table 2. The resulting light curve is shown in Fig. 8a.

We then applied the same two-stage approach as in Sect. 4.2. We masked the transit region and trained two separate GP models on the background data, D_{bg}. On the one hand, we used `gallifrey`'s structure learning with a maximum tree depth $D_{\max} = 3$, and a kernel library consisting of the periodic, RBF, and linear kernels as atomic kernels. For comparison, we also trained a model with a maximum tree depth of $D_{\max} = 0$ and only the RBF kernel as the base kernel. This mirrors typical applications of GPs, where a single, pre-defined kernel is chosen. The RBF kernel is a common default choice in such scenarios.

Figure 8b shows the learned GPs for both methods. The models perform similar outside the transit region, where training data



(a)

(b)

(c)

Fig. 8. Transit parameter fit for the synthetic light curve with noticeable stellar variability and correlated noise in Sect. 4.2.2. (*a*) Synthetic data with correlated noise generated using an AR(1) process, with an added transit signal. The grey line corresponds to the underlying transit light curve. (*b*) GP prediction for the learned kernel structure (purple) and the RBF kernel (magenta). The central line corresponds to the mean prediction with a confidence band of one standard deviation. (*c*) Posterior distribution for the transit parameter, *p*, using the learned kernel and RBF kernel to construct the predictive distribution.

are available. However, the predictions within the masked transit region differ significantly. The simple RBF kernel predicts a larger background value compared to the structure-learned model. This difference will directly impact the subsequent transit parameter estimation.

We sampled the transit parameter posteriors using the same approach as in the previous section. We fixed the planetary period, time of mid-transit, transit duration,and impact parameter to their true value, and fitted the planet-to-star radius ratio R_p/R_* and the two limb darkening coefficients, u_1 and u_2. We again applied flat priors, where the limb darkening coefficients were constrained to lie between 0 and 1.

Figure 8c shows the posterior distributions for both models. In both cases, the limb-darkening parameters cannot be meaningfully constrained and have similar distributions. However, the kernel learning method is able to constrain the radius ratio more precisely and more accurately. The mean value for the learned kernel is 0.100 ± 0.004, which matches the true value to three significant figures. In contrast, the RBF kernel method estimate is about one standard deviation away from the true value, while the standard deviation itself is more than twice as large for this kernel than for the learned kernel.

4.3. Transmission spectroscopy of HATS-46 b

As a final example, we used `gallifrey` to obtain the transmission spectrum of HATS-46 b, a hot Jupiter with a Jupiter-like radius ($R_p = 0.95\ R_{\mathrm{Jupiter}}$) and a mass of $M_p = 0.16\ M_{\mathrm{Jupiter}}$. This planet has been observed using the EFOSC2 instrument on the ground-based New Technology Telescope, and the data have been analysed by Ahrer et al. (2023).

In the original work, background modelling was performed using a variety of methods, including GPs with different kernels. Their analysis encompassed the white light curve and 25 spectroscopic light curves, and was performed in a two-step process. First, they fit the transit parameters of the white light curve and fixed the shared parameters $\boldsymbol{p}_{\mathrm{shared}}$ (stellar radius, inclination, and time of mid-transit) to their best-fit values. Subsequently, they sample the individual parameters $\boldsymbol{p}_{\mathrm{individual}}$ (radius ratio, and limb-darkening coefficients, u_1 and u_2) for each spectroscopic light curve independently. This done to maintain consistency across the light curves, as different values of the stellar radius, for example, would lead to shifts in the derived transit depths relative to each other.

We used an alternative approach, building on the analysis techniques presented in the previous sections. We first fitted GP background models to all 26 light curves (the white light curve and 25 spectroscopic light curves), masking the transit regions during the fitting process. This provides, for each light curve, a GP model trained on the out-of-transit data.

From these trained GP models, we obtain 26 predictive distributions for the transit regions, one for each light curve. and then construct a transit model for each light curve, using the corresponding predictive distribution as the background model. The likelihood for all parameters (shared and individual) is obtained by multiplying the individual likelihoods. For a single light curve *i*, the likelihood is based on the GMM predictive distribution. The full likelihood is the product of all individual likelihoods,

$$\mathcal{L}(\boldsymbol{p}_{\mathrm{shared}}, \{\boldsymbol{p}_{\mathrm{ind},i}\}) = \prod_{i=1}^{26} \mathcal{L}_i(\boldsymbol{p}_{\mathrm{shared}}, \boldsymbol{p}_{\mathrm{ind},i}). \tag{21}$$

Our final model has 4 shared parameters: stellar radius ($R_\star$), inclination (i), stellar mass ($M_\star$), and time of mid-transit (T_0). It also has $26 \times 3 = 78$ individual parameters: the radius ratio ($(R_p/R_\star)_i$), and the quadratic limb-darkening coefficients ($u_{1,i}$ and $u_{2,i}$) for each of the 26 light curves. This gives a total of 82 parameters. We fixed the planet's orbital period to the value reported by Ahrer et al. (2023), $T = 4.7423749$ days.

A key advantage of our approach is that we disentangle the GP fitting and the transit parameter fitting. Traditionally, fitting both the GP hyperparameters and the transit parameters simultaneously would be computationally infeasible. However, because we have already obtained a sample of GPs (and constructed their

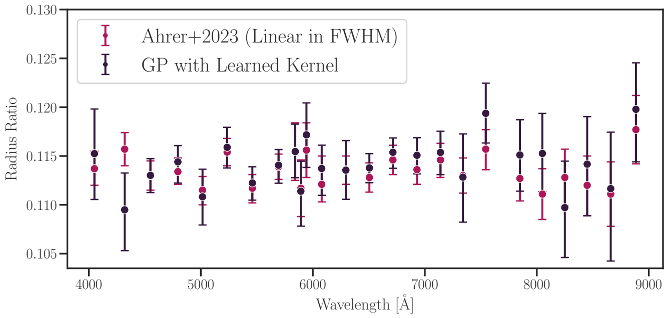


Fig. 9. Transmission spectrum of HATS-46 b using gallifrey's learned GP model (purple) and the reference spectrum obtained by Ahrer et al. (2023, magenta), using their fiducial detrending method.

predictive distributions), we no longer need to sample the GP hyperparameters during the transit parameter estimation. This drastically reduces the computational cost.

Finally, we used efficient HMC sampling, specifically the No-U-Turn Sampler (NUTS; Hoffman & Gelman 2011) implemented in BlackJAX, to sample the 82 transit parameters simultaneously. This allows us to fully propagate the uncertainties in the shared parameters into the final transmission spectrum.

The resulting transmission spectrum, derived from this joint analysis, is shown in Fig. 9, along with the spectrum from Ahrer et al. (2023) for comparison. Overall, the two spectra are consistent with one another, while our uncertainties are, on average, larger. This is expected because we included the uncertainties in the shared parameters (rather than fixing them), we incorporated different GP kernel structures, and we also sampled the second limb-darkening coefficient (u_2), which was fixed in the original analysis. Despite these differences, we obtain a consistent and robust transmission spectrum because we accounted for all sources of uncertainty in a principled Bayesian manner. Figure 10 shows the individual transit light curves and their corresponding fits, demonstrating the agreement between our model and the observed data. The model captures both the transit signal and the stellar variability, as represented by the GP predictive distributions.

5. Discussion

We have demonstrated the potential of GP structure learning as a powerful tool for time series modelling tasks, particularly within the context of stellar activity and exoplanet transit modelling. We show that structure learning can lead to more robust, consistent, and in some cases more accurate transit parameter retrievals. We have implemented this structure learning approach in a self-consistent Bayesian manner, by constructing a prior over kernel structure that allows for the exploration of a space of plausible GP models. To ensure robust and efficient posterior sampling of this space, we used SMC methods, which benefit from native parallelisability and can create ensemble forecasts over different samples from the GP space. SMC also accelerates the sampling process, since it allows for the sequential addition of data points using a data annealing scheme, which partially offsets the inherently inefficient $O(N^3)$ scaling of GP regression. The combination of the structure learning method and SMC model ensembles enables the creation of robust predictive distributions, which allow us to decouple the background modelling task from the transit parameter inference. This results in a faster and more uncertainty-aware parameter retrieval while avoiding the risk of overfitting.

To encourage the wider adoption of these techniques, we have developed gallifrey, a user-friendly Python package that encapsulates the described algorithms. gallifrey implements the structure learning methodology, originally described by Saad et al. (2023), and is built on the JAX library (Bradbury et al. 2018) and its surrounding ecosystem. This ensures computational efficiency and automatic differentiability, which is useful for gradient-based sampling methods in high-dimensional parameter spaces.

We have shown the applicability of this approach to both space-based transit light curves, such as those from TESS or the upcoming PLATO mission, and ground-based spectroscopic observations. To demonstrate the advantages of disentangling the GP background and transit modelling, we give an example of obtaining the transmission spectrum of the hot Jupiter HATS-46 b from the spectroscopic light curves. By independently modelling the background in each light curve using flexible GPs trained on out-of-transit data, we generated predictive distributions for the in-transit background, which can be used as likelihood functions for the transit parameter retrieval. This allows for joint sampling of the transit parameters for all light curves, leading to more accurate and robust uncertainty estimates for the transmission spectrum.

Despite our focus on efficiency, the $O(N^3)$ scaling associated with standard GPs remains a fundamental limitation, particularly when dealing with large datasets. This scaling currently restricts the practical application of our method to individual transit windows within longer light curves, in order to keep computing time within reasonable bounds. To enable the modelling of long-duration light curves, for example those expected from the two-year single-field pointings of PLATO, more scalable techniques will need to be included in the structure learning methodology. Thankfully, there are several promising avenues for this. Hardware acceleration has been shown to enable scalable GP regression. Wang et al. (2019), for example, have scaled exact GPs to beyond one million data points using multi-GPU parallelisation. The JAX framework underpinning gallifrey inherently supports hardware acceleration via GPUs, and other GP implementations within the JAX ecosystem, such as tinygp (Foreman-Mackey et al. 2024), or the GPyTorch (Gardner et al. 2018) in the PyTorch ecosystem, have demonstrated order-of-magnitude speed increases through GPU utilisation. Beyond hardware acceleration, exact GPs that use quasi-separable kernels, such as those introduced by Foreman-Mackey et al. (2017a), scale linearly with data size $O(N)$ and have been shown to be a good model for stellar variability. This category of kernels is, in principle, compatible with the kernel structure learning techniques implemented in gallifrey. Integrating such kernels in the future would therefore potentially allow for first-principles, scalable GP structure learning for stellar background modelling. Beyond classical, exact GPs, approximate techniques such as sparse GPs (Leibfried et al. 2022) and Hilbert space GPs (Riutort-Mayol et al. 2022), or more recently deep kernel learning (Wilson et al. 2015; Ober et al. 2021), hold potential for even more scalable and adaptive applications.

In conclusion, GP structure learning presents a valuable avenue for exoplanet research, particularly in the context of transit modelling and retrieval. By providing a user-friendly Python implementation of state-of-the-art structure learning algorithms in gallifrey, we hope to make this methodology more accessible to researchers and help tackle the challenges of increasingly complex exoplanet datasets.

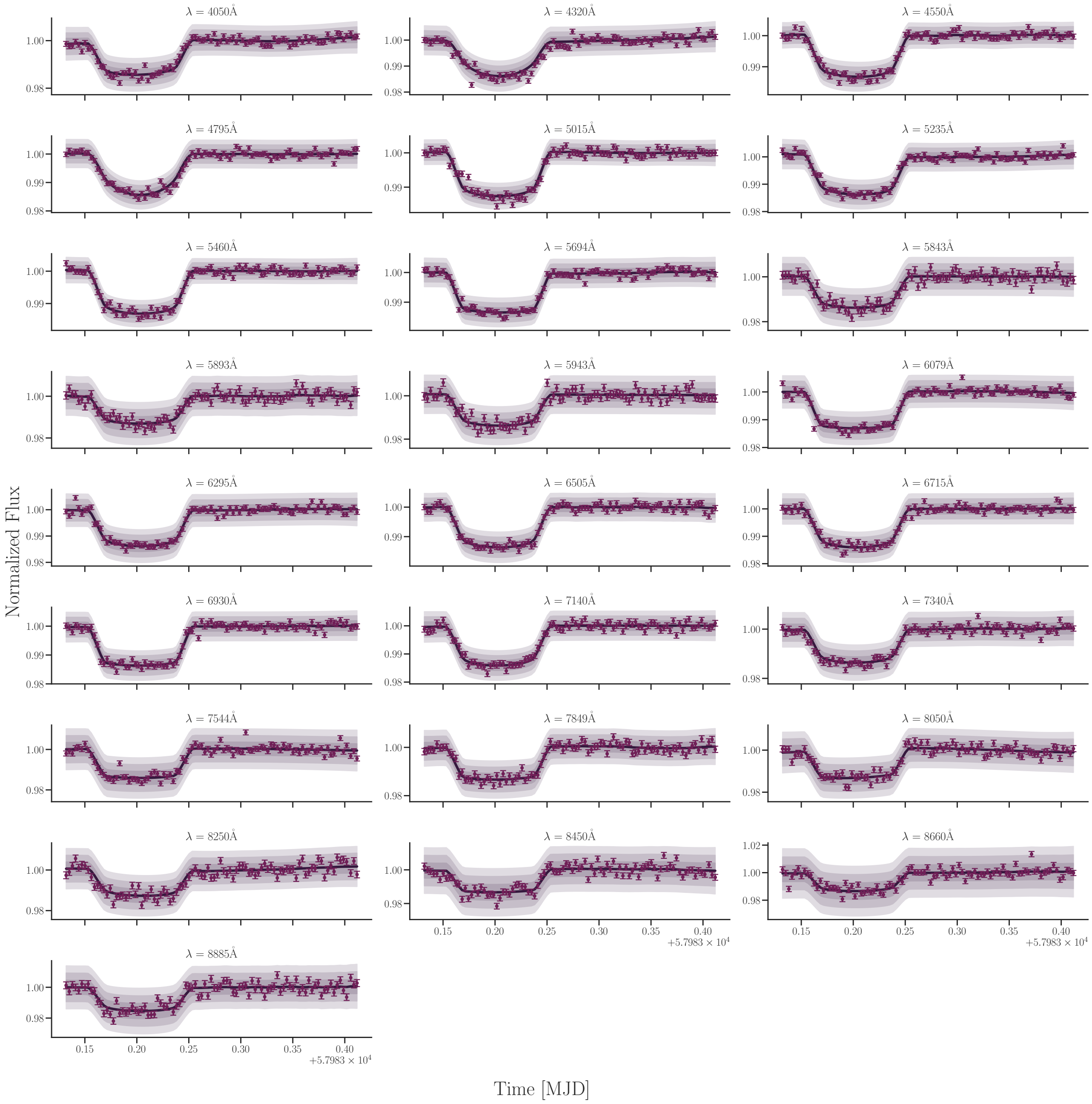


Fig. 10. Model light curves for the 25 spectroscopic HATS-46 b light curves. Shown are the mean GP predictions with one, two, and three standard deviation confidence bands. The transit models are included using the median transit parameter from the MCMC sampling of the joint posterior.

Data availability

The full source code and all data used in this work are available at https://github.com/ChrisBoettner/gallifrey. An in-depth documentation, as well as tutorials to recreate all figures in this work, can be found at https://chrisboettner.github.io/gallifrey. The spectroscopic light curves for the HATS-46 b spectrum can be found at https://cdsarc.cds.unistra.fr/viz-bin/cat/J/MNRAS/521/5636.

Acknowledgements. Chris Boettner thanks the Young Academy Groningen for their generous support through an interdisciplinary PhD fellowship. He also wants to thank his supervisor, Pratika Dayal, for kind and continued support, and further wants to thank Evie Ahrer for the valuable discussions and providing him light curve data to play around with. The gallifrey package is a Python re-implementation of the Julia package `AutoGP.jl` (Saad et al. 2023) and would have not been possible without it. Large parts of the implementation details are inspired by the fantastic packages `GPJax` (Pinder & Dodd 2022) and`tinygp` (Foreman-Mackey et al. 2024). Data was retrieved using the via `astroquery` (Ginsburg et al. 2019), `lightkurve` (Lightkurve Collaboration 2018) and `PSLS` (Samadi et al. 2019). Visualisations were made using `pandas` (The pandas development team 2023), `matplotlib` (Caswell et al. 2023) and `seaborn` (Waskom 2021).

References

Abdessalem, A. B., Dervilis, N., Wagg, D. J., & Worden, K. 2017, Front. Built Environ., 3

Ahrer, E., Wheatley, P. J., Kirk, J., et al. 2022, MNRAS, 510, 4857

Ahrer, E., Wheatley, P. J., Gandhi, S., et al. 2023, MNRAS, 521, 5636

Aigrain, S., & Foreman-Mackey, D. 2023, Annu. Rev. Astron. Astrophys., 61, 329

Aigrain, S., Pont, F., & Zucker, S. 2012, MNRAS, 419, 3147

Aigrain, S., Hodgkin, S. T., Irwin, M. J., Lewis, J. R., & Roberts, S. J. 2015, MNRAS, 447, 2880

Aigrain, S., Parviainen, H., & Pope, B. J. S. 2016, MNRAS, 459, 2408

Akaike, H. 1998, in Selected Papers of Hirotugu Akaike, eds. E. Parzen, K. Tanabe, & G. Kitagawa, Springer Series in Statistics (New York, NY: Springer), 199

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. 2015, IEEE Trans. Pattern Anal. Mach. Intell., 38, 252

Angus, R., Morton, T., Aigrain, S., Foreman-Mackey, D., & Rajpaul, V. 2018, MNRAS, 474, 2094

Bachoc, F. 2013, Computat. Statist. Data Anal., 66, 55

Barros, S. C. C., Demangeon, O., Díaz, R. F., et al. 2020, A&A, 634, A75

Borucki, W. J., Koch, D., Basri, G., et al. 2010, Science, 327, 977

Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, https://doi.org/10.5281/zenodo.10736936

Cabezas, A., Corenflos, A., Lao, J., et al. 2024, arXiv e-prints [arXiv:2402.10797]

Caswell, T. A., Elliott Sales de Andrade, Lee, A., et al. 2023, https://doi.org/10.5281/zenodo.592536

Chopin, N., & Papaspiliopoulos, O. 2020, An Introduction to Sequential Monte Carlo, Springer Series in Statistics (Cham: Springer International Publishing)

Crossfield, I. J. M., Ciardi, D. R., Petigura, E. A., et al. 2016, Astrophys. J. Suppl. Ser., 226, 7

Del Moral, P., Doucet, A., & Jasra, A. 2006, J. R. Stat. Soc. Ser. B Stat. Methodol., 68, 411

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., & Ghahramani, Z. 2013, arXiv e-prints [arXiv:1302.4922]

Espinoza, N., Kossakowski, D., & Brahm, R. 2019, MNRAS, 490, 2262

Evans, T. M., Pont, F., Sing, D. K., et al. 2013, ApJ, 772, L16

Evans, T. M., Aigrain, S., Gibson, N., et al. 2015, MNRAS, 451, 680

Evans, T. M., Sing, D. K., Kataria, T., et al. 2017, Nature, 548, 58

Foreman-Mackey, D., Montet, B. T., Hogg, D. W., et al. 2015, ApJ, 806, 215

Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. 2017a, Astrophys. Source Code Libr. [ascl:1709.008]

Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. 2017b, AJ, 154, 220

Foreman-Mackey, D., Savel, A., Luger, R., et al. 2021, https://doi.org/10.5281/zenodo.1998447

Foreman-Mackey, D., Yu, W., Yadav, S., et al. 2024, https://doi.org/10.5281/zenodo.10463641

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., & Wilson, A. G. 2018, in Adv. Neural Inf. Process. Syst.

Gibson, N. P. 2014, MNRAS, 445, 3401

Gibson, N. P., Aigrain, S., Roberts, S., et al. 2012, MNRAS, 419, 2683

Gillen, E., Briegal, J. T., Hodgkin, S. T., et al. 2020, MNRAS, 492, 1008

Ginsburg, A., Sipőcz, B. M., Brasseur, C. E., et al. 2019, AJ, 157, 98

Hattori, S., Garcia, L., Murray, C., et al. 2024, https://doi.org/10.5281/zenodo.10736936

Haywood, R. D., Collier Cameron, A., Queloz, D., et al. 2014, MNRAS, 443, 2517

Hoffman, M. D., & Gelman, A. 2011, arXiv e-prints [arXiv:1111.4246]

Karamanis, M., & Seljak, U. 2025, arXiv e-prints [arXiv:2407.20722]

Kim, H., & Teh, Y. W. 2018, arXiv e-prints [arXiv:1706.02524]

Leibfried, F., Dutordoir, V., John, S. T., & Durrande, N. 2022, arXiv e-prints [arXiv:2012.13962]

Lightkurve Collaboration (Cardoso, J. V. d. M., et al.) 2018, Lightkurve: Kepler and TESS Time Series Analysis in Python, Astrophysics Source Code Library

Luger, R., Foreman-Mackey, D., & Hedges, C. 2021, AJ, 162, 124

Nicholson, B. A., & Aigrain, S. 2022, MNRAS, 515, 5251

Ober, S. W., Rasmussen, C. E., & van der Wilk, M. 2021, arXiv e-prints [arXiv:2102.12108]

Pinder, T., & Dodd, D. 2022, JOSS, 7, 4455

Pont, F., Zucker, S., & Queloz, D. 2006, MNRAS, 373, 231

Rasmussen, C. E., & Williams, C. K. I. 2006, Gaussian Processes for Machine Learning (The MIT Press)

Rauer, H., Catala, C., Aerts, C., et al. 2014, Exp. Astron., 38, 249

Ricker, G. R., Winn, J. N., Vanderspek, R., et al. 2015, J. Astron. Telesc. Instrum. Syst., 1, 014003

Riutort-Mayol, G., Bürkner, P.-C., Andersen, M. R., Solin, A., & Vehtari, A. 2022, Stat. Comput., 33, 17

Saad, F. A., Cusumano-Towner, M. F., Schaechtle, U., Rinard, M. C., & Mansinghka, V. K. 2019, Proc. ACM Program. Lang., 3, 1

Saad, F. A., Patton, B. J., Hoffman, M. D., Saurous, R. A., & Mansinghka, V. K. 2023, arXiv e-prints [arXiv:2307.09607]

Samadi, R., Deru, A., Reese, D., et al. 2019, A&A, 624, A117

Schwarz, G. 1978, Ann. Stat., 6, 461

The pandas development team 2023, https://doi.org/10.5281/zenodo.3509134

Wang, K. A., Pleiss, G., Gardner, J. R., et al. 2019, arXiv e-prints [arXiv:1903.08114]

Waskom, M. 2021, JOSS, 6, 3021

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. 2015, arXiv e-prints [arXiv:1511.02222]

Appendix A: Kernel functions and hyperparameters

The choice of the kernel function $k(x, x')$ is crucial in GP regression, as it defines the space of possible functions considered by the prior.

To produce valid covariance matrices, as required for Eqs. 2 and 9, the kernel function must conform to certain properties. Specifically, a valid kernel needs to be symmetric, i.e. $k(x, x') = k(x', x)$, and the kernel matrix K should be positive semi-definite, fulfilling $\boldsymbol{v}^\top K \boldsymbol{v} \geq 0$ for any vector $\boldsymbol{v}$. Furthermore, kernels are called stationary if $k(x, x') = k(x - x')$, and isotropic if $k(x, x') = k(|x - x'|)$.

While any function fulfilling these conditions can serve as a kernel, there are several commonly used ones. The squared exponential or RBF kernel is often used to model smoothly varying stationary functions, whereas kernels from the Matérn family are used to model more rugged functions. Various periodic kernels exist to model periodic functions, and non-stationarity can be introduced using a linear kernel. We have compiled a list of commonly used kernels and their properties in Table A.1. All the listed kernels are implemented as atomic kernels in `gallifrey`, and can be used for structure learning. Importantly, any kernel can be scaled by some variance σ^2, and kernels can be combined arbitrarily to create new, more complex kernels.

All kernels $k(x, x'; \boldsymbol{\eta})$ in Table A.1 depend on hyperparameters $\boldsymbol{\eta}$, governing their specific properties. Common hyperparameters include a length scale l, which controls the smoothness of functions within the function space, and the period p for periodic kernels. In typical GP regression, the standard approach is to specify a kernel structure by choosing a combination of base kernels and then optimise the hyperparameters based on the dataset. Two common approaches for tuning hyperparameters are maximising the marginal likelihood $P(\mathcal{D}|k(\boldsymbol{\eta}))$, as given by Eq. (11) (where the dependence on the kernel and its hyperparameters is made explicit), or maximising the leave-one-out cross-validation (LOO-CV) predictive probability:

$$\log P_{\mathrm{LOOCV}}(\mathcal{D}|k(\boldsymbol{\eta})) = \sum_{i}^{N} \log P\left(y_i|\boldsymbol{x}_i, \boldsymbol{x}_{-i}, \boldsymbol{y}_{-i}; k(\boldsymbol{\eta})\right), \tag{A.1}$$

where $\{\boldsymbol{x}_{-i}, \boldsymbol{y}_{-i}\}$ represents the dataset with the i-th entry removed, and $P\left(y_i|\boldsymbol{x}_i, \boldsymbol{x}_{-i}, \boldsymbol{y}_{-i}\right)$ is given by Eq. (9). Numerical studies suggest that using the LOO-CV predictive probability leads to more accurate predictions in cases of mis-specified covariance functions (i.e. kernel structures), while the marginal likelihood is optimal when the kernel structure is well specified (Bachoc 2013).

Appendix B: Assessing the quality of the SMC posterior approximation

The SMC algorithm implemented in `gallifrey` approximates the posterior distribution over both kernel structures k and their hyperparameters $\boldsymbol{\eta}$, as defined in Eq. (12). Unlike MCMC, which targets convergence to a stationary distribution, SMC generates a sequence of weighted particle ensembles approximating a sequence of distributions $\{\pi_t\}_{t=1}^{T}$, culminating in a final weighted sample $\{(k_i, \boldsymbol{\eta}_i, w_T^{(i)})\}_{i=1}^{N}$ that approximates the target posterior $\pi_T \approx P(f|\mathcal{D})$. Assessing whether this final sample is a 'good' approximation is important for reliable inference. Key diagnostics include:

1. **Final weight distribution, effective sample size, and resampling history**: The ESS, given by Eq. (16), is a central diagnostic for SMC performance. It quantifies the degeneracy of the final particle weights $w_T^{(i)}$. A low ESS indicates that the approximation is effectively supported by only a few particles, likely indicating a poor representation of the true posterior. While resampling during the SMC run helps mitigate degeneracy, the distribution of the final weights and the final ESS should still be examined. A low final ESS suggests the final approximation is unreliable or unstable. Furthermore, `gallifrey` returns the resampling history after the SMC procedure. If particle degeneracy is no major issue, one would expect several rejuvenation steps between resampling steps. If the ESS is consistently low (triggering frequent resampling), increasing the number of particles (N), adjusting the annealing schedule, or increasing the number of rejuvenation steps may be necessary.
2. **Validation on unseen data**: In the transit fitting examples (Sects. 4.2 and 4.3), we masked the transit interval and used `gallifrey` for interpolation, modelling the background signal as if no transit occurred. If the length of the time series allows, it may be advisable to mask additional intervals (without transits), and evaluate the quality of the approximation on these intervals where the ground truth is known.
3. **Stability across runs**: If computational cost and time allows, a fundamental check could be running the `gallifrey` SMC procedure multiple times with different random seeds but identical settings. Stable results across different runs suggests a good approximation for the posterior.
4. **Number of particles**: In general, a larger number of particles N leads to a better approximation but increases computational cost. The 'correct' N is problem-dependent and must usually be determined empirically. In `gallifrey`, we used JAX's `pmap` function for efficient parallelisation. We therefore suggest using as many particles as there are physical devices (e.g. number of CPU cores) available, if possible. However, for the examples in Sect. 4, we find that as few as six particles yield stable results.

Table A.1: Common kernels in GP regression.

Name	Equation	# of Parameters	Parameter	Stationary
Periodic	$\sigma^2 \exp\left(-\frac{\sin^2(\pi\|x-x'\|/p)}{2l^2}\right)$	3	lengthscale : l variance : σ^2 period : p	Yes
Powered Exponential	$\sigma^2 \exp\left(-\left(\frac{\|x-x'\|}{l}\right)^\kappa\right)$	3	lengthscale : l variance : σ^2 power : κ	Yes
Squared Exponential / Radial Basis Function	$\sigma^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right)$	2	lengthscale : l variance : σ^2	Yes
Matérn $\left(\nu = \frac{1}{2}\right)$ / Ornstein-Uhlenbeck	$\sigma^2 \exp\left(-\frac{\|x-x'\|}{l}\right)$	2	lengthscale : l variance : σ^2	Yes
Matérn $\left(\nu = \frac{3}{2}\right)$	$\sigma^2 \left(1 + \frac{\sqrt{3}\|x-x'\|}{l}\right) \exp\left(-\frac{\sqrt{3}\|x-x'\|}{l}\right)$	2	lengthscale : l variance : σ^2	Yes
Matérn $\left(\nu = \frac{5}{2}\right)$	$\sigma^2 \left(1 + \frac{\sqrt{5}\|x-x'\|}{l} + \frac{5(x-x')^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}\|x-x'\|}{l}\right)$	2	lengthscale : l variance : σ^2	Yes
Rational Quadratic	$\sigma^2 \left(1 + \frac{(x-x')^2}{2\alpha l^2}\right)^{-\alpha}$	3	lengthscale : l variance : σ^2 shape : α	Yes
White	$\sigma^2 \delta(x - x')$	1	variance : σ^2	Yes
Linear	$\sigma^2 (x \cdot x')$	1	variance : σ^2	No
Linear with Shift	$b + \sigma^2((x - c) \cdot (x' - c))$	3	bias : b variance : σ^2 shift : c	No

Notes. Each of the listed kernels is implemented as an atomic kernel in `gallifrey`.