The Journal of Open Source Software

starry_process: Interpretable Gaussian processes for stellar light curves

Rodrigo Luger[1, 2], **Daniel Foreman-Mackey**[1], **and Christina Hedges**[3, 4]

1 Center for Computational Astrophysics, Flatiron Institute, New York, NY **2** Virtual Planetary Laboratory, University of Washington, Seattle, WA **3** Bay Area Environmental Research Institute, P.O. Box 25, Moffett Field, CA 94035, USA **4** NASA Ames Research Center, Moffett Field, CA

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Statement of need

Mapping the surfaces of stars using time series measurements is a fundamental problem in modern time-domain stellar astrophysics. This inverse problem is ill-posed and computationally intractable, but in the associated AAS Journals publication submitted in parallel to this paper, we derive an interpretable effective Gaussian Process (GP) model for this problem that enables robust probabilistic characterization of stellar surfaces using photometric time series observations. Implementation of this model requires the efficient evaluation of a set of special functions and recursion relations that are not readily available in existing probabilistic programming frameworks. The `starry_process` package provides the necessary elements to perform this analysis with existing and forthcoming astronomical datasets.

Summary

We implement our interpretable GP in the open-source, user-friendly `Python` package `starry_process`, which can be installed via `pip` or from source on GitHub. The code is thoroughly unit-tested and well documented, with examples on how to use the GP in custom inference problems. As discussed in the associated AAS Journals publication, users can choose, among other options, whether or not to marginalize over inclination and whether or not to model a normalized process. Users can also choose the spherical harmonic degree of the expansion, although it is recommended to use $l_{\max} = 15$ (see below). Users may compute the mean vector and covariance matrix in either the spherical harmonic basis or the flux basis, or they may sample from it or use it to compute marginal likelihoods. Arbitrary order limb darkening is implemented following Agol et al. (2020).

The code was designed to maximize the speed and numerical stability of the computation. Although most of the computations of the expectation integrals involve many layers of nested sums over all spherical harmonic coefficients, these may be expressed as high-dimensional tensor products, which can be evaluated efficiently on modern hardware. Many of the expressions can also be either pre-computed or computed recursively. To maximize the speed of the algorithm, the code is implemented in hybrid `C++/Python` using the just-in-time compilation capability of the `Theano` package (Theano Development Team, 2016). Since all equations derived here have closed form expressions, these can be differentiated in a straightforward and numerically stable manner, enabling the computation of backpropagated gradients within `Theano`. As such, `starry_process` is designed to work out-of-the box with `Theano`-based inference tools such as `PyMC3` for NUTS/HMC or ADVI sampling (Salvatier et al., 2016).

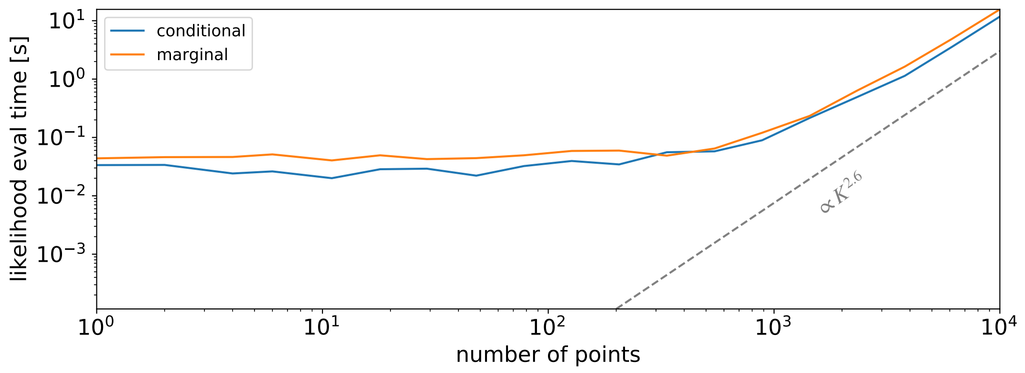

Figure 1: Evaluation time in seconds for a single log-likelihood evaluation as a function of the number of points K in each light curve when conditioning on a value of the inclination (blue) and when marginalizing over the inclination (orange). At $l_{\max} = 15$, computation of the covariance matrix of the GP takes about 20ms on a 2018 MacBook Pro. The dashed line shows the asymptotic scaling of the algorithm, which is due to the Cholesky factorization and solve operations.

Figure 1 shows the computational scaling of the `Python` implementation of the algorithm for the case where we condition the GP on a specific value of the inclination (blue) and the case where we marginalize over inclination (orange). Both curves show the time in seconds to compute the likelihood (averaged over many trials) as a function of the number of points K in a single light curve. For $K \lesssim 100$, the computation time is constant at $10 - 20$ ms for both algorithms. This is the approximate time (on a typical modern laptop) taken to compute the GP covariance matrix given a set of hyperparameters $\boldsymbol{\theta}_\bullet$. For larger values of K, the cost approaches a scaling of $K^{2.6}$, which is dominated by the factorization of the covariance matrix and the solve operation to compute the likelihood. The likelihood marginalized over inclination is only slightly slower to compute, thanks to the tricks discussed in the associated publication in the AAS Journals.

Many modern GP packages (e.g., Ambikasaran et al., 2015; Foreman-Mackey et al., 2017) have significantly better asymptotic scalings, but these are usually due to specific structure imposed on the kernel functions, such as the assumption of stationarity. Our kernel structure is determined by the physics (or perhaps more accurately, the geometry) of stellar surfaces, and its nonstationarity is a consequence of the normalization step in relative photometry. Moreover, and unlike the typical kernels used for GP regression, our kernel is a nontrivial function of the hyperparameters $\boldsymbol{\theta}_\bullet$, so its computation is necessarily more expensive. Nevertheless, the fact that our GP may be used for likelihood evaluation in a small fraction of a second for typical datasets ($K \sim 1,000$) makes it extremely useful for inference. Recall that we are implicitly marginalizing over all of the properties of *every spot* on the surface of the star.

In ensemble analyses, we must compute the likelihood of each of the M light curves conditioned on $\boldsymbol{\theta}_\bullet$. In practice, each star will have a different rotation period, different limb darkening coefficients, and different photometric uncertainty, meaning we must factorize M different covariance matrices. Fortunately, the spherical harmonic covariance, $\boldsymbol{\Sigma}_\mathbf{y}$ need only be computed once; this can then be linearly transformed into the flux basis for each light curve. As the evaluation of $\boldsymbol{\Sigma}_\mathbf{y}$ is the computationally-intensive step, the likelihood evaluation typically scales sub-linearly with M. Furthermore, it is possible to marginalize over the period and limb darkening coefficients, which would remove the scaling with M entirely if the photometric precision were the same for all light curves. Even if it is not, the algorithm can still be greatly sped up, although an implementation of this is deferred to the next paper.

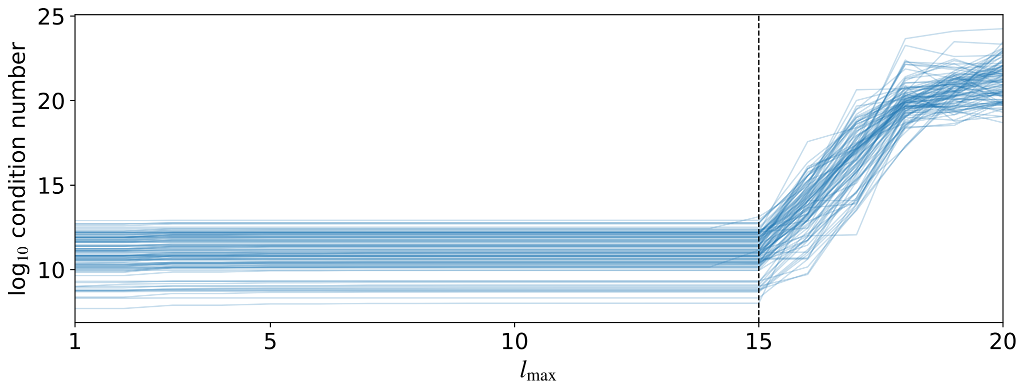

Figure 2: Log of the condition number of the covariance in the spherical harmonic basis, $\boldsymbol{\Sigma}_\mathbf{y}$, as a function of the spherical harmonic degree of the expansion, $l_{\max}$. Different lines correspond to different values of $\boldsymbol{\theta}_\bullet$ drawn from a uniform prior (see text for details). In the majority of the cases, the matrix becomes ill-conditioned above $l_{\max} = 15$

Our algorithm is also numerically stable over nearly all of the prior volume up to $l_{\max} = 15$. Figure 2 shows the log of the condition number of the covariance matrix in the spherical harmonic basis, $\boldsymbol{\Sigma}_\mathbf{y}$, as a function of the spherical harmonic degree of the expansion for 100 draws from a uniform prior over $r \in [10^\circ, 45^\circ]$, $\mu_\phi \in [0^\circ, 85^\circ]$, $\sigma_\phi \in [5^\circ, 40^\circ]$, and $n \in [1, 50]$. The condition number is nearly constant up to $l_{\max} = 15$ in almost all cases; above this value, the algorithm suddenly becomes unstable and the covariance is ill-conditioned. The instability occurs within the computation of the latitude and longitude moment integrals and is likely due to the large number of operations involving linear combinations of hypergeometric and gamma functions. While it may be possible to achieve stability at higher values of $l_{\max}$ via careful reparametrization of some of those equations, we find that $l_{\max} = 15$ is high enough for most practical purposes. We plan to revisit this point in future work.

Finally, instabilities can also occur if σ_ϕ is too small and/or n is too large. Values of $\sigma_\phi \lesssim 1^\circ$ lead to instabilities in the computation of the hypergeometric functions, while values of $n \gtrsim 50$ can sometimes cause the Cholesky factorization of the covariance to fail (although this can be mitigated by adding a small quantity to the diagonal to ensure positive-semidefinitess). In cases where the algorithm goes (very) unstable, the log-likelihood evaluation returns $-\infty$: in other words, they are silently rejected by an implicit prior. Fortunately, these cases are likely unphysical: in practice, there should always be some finite amount of variance in the latitudes of spots, and stars with more than 50 spots are likely too spotted for individual spots to be discernible in the first place. Instead, we are likely sensitive to *groups* of spots, which our GP is flexible enough to model.

Acknowledgements

References

Agol, E., Luger, R., & Foreman-Mackey, D. (2020). Analytic Planetary Transit Light Curves and Derivatives for Stars with Polynomial Limb Darkening. *159*(3), 123. https://doi.org/10.3847/1538-3881/ab4fee

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. (2015). Fast Direct Methods for Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*, 252. https://doi.org/10.1109/TPAMI.2015.2448083

Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. (2017). Fast and Scalable Gaussian Process Modeling with Applications to Astronomical Time Series. *154*(6), 220.

https://doi.org/10.3847/1538-3881/aa9332

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, *2*, e55.

Theano Development Team. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-Prints*, *abs/1605.02688*. http://arxiv.org/abs/1605.02688