

NOTE: this is a very rough draft of this manuscript — read at your own risk!

exoplanet: A framework for gradient-based inference with time domain astronomical data sets

DANIEL FOREMAN-MACKEY,¹ IAN CZEKALA,^{2,*} ERIC AGOL,^{3,4,†} AND
RODRIGO LUGER¹

¹*Center for Computational Astrophysics, Flatiron Institute, 162 5th Ave, New York, NY 10010*

²*Department of Astronomy, 501 Campbell Hall, University of California, Berkeley, CA 94720*

³*Department of Astronomy, University of Washington, Seattle, WA 98195*

⁴*Virtual Planetary Laboratory, University of Washington, Seattle, WA 98195*

ABSTRACT

As larger and more precise datasets continue to be collected for the discovery and characterization of exoplanets, we must also develop computationally efficient and rigorous methods for making inferences based on these data. The efficiency and robustness of numerical methods for probabilistic inference can be improved by using derivatives of the model with respect to the parameters. We derive methods for exploiting these gradient-based methods when fitting exoplanet data based on radial velocity, astrometry, and transits. Alongside the paper, we release an open source, well-tested, and computationally efficient implementation of these methods. These methods can substantially reduce the computational cost of fitting large datasets and systems with more than one exoplanet.

Keywords: methods: data analysis — methods: statistical

1. OUTLINE

1. Introduction

- the datasets available and forthcoming
- existing tools for exoplanet fitting
- motivation for gradients

2. Automatic differentiation and inference frameworks

3. Hamiltonian Monte Carlo and variants

4. Custom gradients required for exoplanet datasets

* NASA Hubble Fellowship Program Sagan Fellow

† Guggenheim Fellow

- Radial velocities and Kepler’s equation
 - Astrometry
 - Transits
5. Implementation details and benchmarks
 6. Examples
 7. Discussion

2. INTRODUCTION

The astronomical community has been investing in time domain survey astronomy and—thanks to this investment—large, homogeneous, public time-domain datasets are becoming common, and the questions that the community is asking are becoming more ambitious. One application of these surveys is the discovery and characterization of stellar systems including multiple exoplanets and stars. Sophisticated statistical methods have been developed for these applications and deployed at large scale, but the computational cost and scaling of these methods limit their application to restricted datasets or models with fewer data points or parameters. This paper describes a new class of methods that have not previously been applied to time series analysis in astrophysics. The key benefit of these methods is that they improve the computational scaling of robust probabilistic inference as a function of the number of parameters in the model; this improved efficiency can reduce runtime for inference in multi-planet systems by orders of magnitude.

The crucial difference between the methods discussed in this paper compared to previous work is that we derive computationally efficient first derivatives of the model with respect to its parameters. Some work has been done to derive some of the relevant derivatives for specific sub problems (CITE) often with the goal of Fisher Information analysis to predict the performance of future surveys, but there is not a general framework that can propagate derivatives for (the majority of) the applications that exoplanet astronomers require. To achieve this goal, we build on methods from applied mathematics and computer science, and work within a framework developed for machine learning. The key concept is *automatic differentiation* that allows efficient and (as the name suggests) automatic propagation of derivatives through complicated computer code. Automatic differentiation has been widely applied across the sciences and it is the fundamental development that enabled the application of machine learning methods like deep learning. There have been some applications within astrophysics, but its general application has been somewhat limited for both historical and practical reasons. One reason for this is that models in astrophysics are often physically motivated and their calculation often involves special functions or numerical methods that have been developed and implemented in special purpose frameworks that can’t be easily be differentiated or otherwise combined with existing automatic differentiation interfaces. There has been progress made towards general

purpose differentiation of physical systems (CITE examples of ODE solvers with variation), but there is still a significant amount of overhead when applying these methods for astrophysics applications.

This paper doesn’t solve this problem, but it does describe the derivation and implementation of a large number of the physical models used in exoplanet characterization. While the examples in this paper are (mostly) focused on the characterization of exoplanetary systems, it is also applicable to other time domain applications out of the box. Furthermore, a pedagogical introduction to the relevant ideas is included to help enable other applications in astrophysics.

This paper is divided into three overarching “chapters”. The first presents an introduction to the ideas and algorithms that enable automatic differentiation and gradient-based inference. Then the second chapter presents the derivation of the core models (and their derivatives) that are used by the **exoplanet** project. Finally, in the third chapter, we demonstrate the application of these methods on several problems in exoplanet astrophysics.

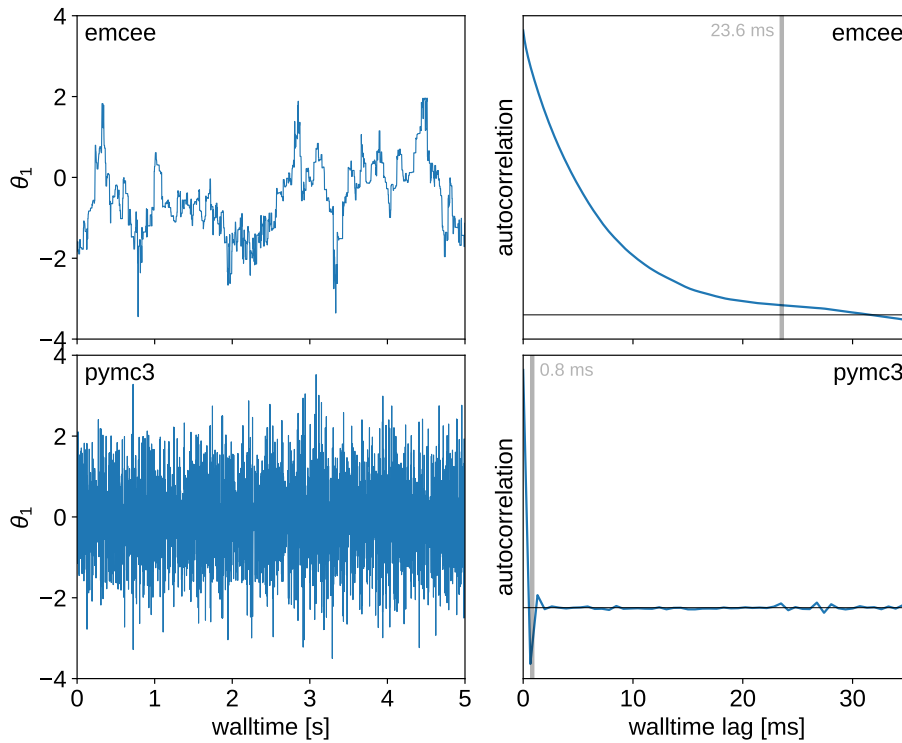



Figure 1. This is a figure. 

3. exoplanet IN CONTEXT

The **exoplanet** project lives in a vibrant ecosystem of software developed for inference with time domain data sets. In particular, there are several open source packages that have been developed for fitting observations of exoplanets including, for example,

EXOFASTv2 (Eastman et al. 2019), juliet (Espinoza et al. 2019), allesfitter (Günther & Daylan 2019), Exo-Striker (Trifonov 2019), pyaneti (Barragán et al. 2019), and RadVel (Fulton et al. 2018). While many of the features of these packages overlap with `exoplanet` and, in fact, the development and design of `exoplanet` was deeply inspired by this ecosystem, both the scope and goals of the `exoplanet` project are fundamentally different. First and foremost, `exoplanet` does not—and won’t ever—include a push button interface to fit any data set. Instead, `exoplanet` is designed as a *framework* that provides the functions that are necessary to build a pipeline. We imagine that future projects like those listed above—or, perhaps, future versions of those projects—will be built with `exoplanet` (or something like it) as the computational backend.

With this contrast in mind, it might make more sense to compare `exoplanet` to projects like `batman` (Kreidberg 2015), `ellc` (Maxted 2016), `starry` (Luger et al. 2019), or `celerite` (Foreman-Mackey et al. 2017), that work behind-the-scenes with many of these .

4. AUTOMATIC DIFFERENTIATION

The main limitation of gradient-based inference methods is that, in order to use them, you must *compute the gradients*! The fundamental quantity of interest is the first derivative (gradient) of the log-likelihood function (or some other goodness-of-fit function) with respect to the parameters of the model. In all but the simplest cases, symbolically deriving this gradient function is tedious and error-prone. Instead, it is generally preferable to use a method (called automatic differentiation) that can automatically compute the exact gradients of the model *to machine precision* at compile time. We would like to emphasize that automatic differentiation does not compute numerical derivatives like finite difference methods, but it does not compute symbolic derivatives either. Instead, it efficiently evaluates the numerical value of the derivative of a piece of code for some given inputs by implicitly applying the chain rule to each operation in the code.

The basic idea behind automatic differentiation is that code is always written as the composition of basic functions and, if we know how to differentiate the subcomponents, then we can apply the chain rule to differentiate the larger function. The realization that this method could be applied to neural network models was the key discoveries that launched the field of “deep learning”, where a specific type of automatic differentiation is used and called “backpropagation” (CITE). In many ways, motivated by this application, there has been substantial research and investment into making these methods general and computationally efficient. Many open source libraries have been released that ease the use of these methods (e.g. TensorFlow, PyTorch, Stan, ceres, Eigen, etc. CITATIONS). Despite the existence of these projects, automatic differentiation has not been widely adopted in the astronomical literature because there is some learning curve associated with porting existing code bases to a new language or framework. Furthermore, many projects in astronomy involve fitting

Forward pass	Forward tangent pass	Reverse pass
$u_1 = x = \mathbf{2}$	$\dot{u}_1 = \mathbf{1}$	
$u_2 = y = -\mathbf{1}$	$\dot{u}_2 = \mathbf{0}$	
$u_3 = u_1^2 = \mathbf{4}$	$\dot{u}_3 = 2 u_1 \dot{u}_1 = \mathbf{4}$	
$u_4 = 2 u_2 = -\mathbf{2}$	$\dot{u}_4 = 2 \dot{u}_2 = \mathbf{0}$	
$u_5 = u_3 + u_4 = \mathbf{2}$	$\dot{u}_5 = \dot{u}_3 + \dot{u}_4 = \mathbf{4}$	
$u_6 = \sqrt{u_5} = \mathbf{1.414}$	$\dot{u}_6 = \dot{u}_5 / 2 \sqrt{u_5} = \mathbf{1.414}$	
$u_7 = \sin u_1 = \mathbf{0.909}$	$\dot{u}_7 = \dot{u}_1 \cos u_1 = -\mathbf{0.416}$	
$u_8 = u_1 u_7 = \mathbf{1.819}$	$\dot{u}_8 = u_1 \dot{u}_7 + \dot{u}_1 u_7 = \mathbf{0.077}$	$\overline{u}_1 = \overline{u}_8 u_7 = -\mathbf{0.909}, \overline{u}_7 = \overline{u}_8 u_1 = -\mathbf{2}$
$u_9 = u_6 - u_8 = -\mathbf{0.404}$	$\dot{u}_9 = \dot{u}_6 - \dot{u}_8 = \mathbf{1.337}$	$\overline{u}_6 = \overline{u}_9 = \mathbf{1}, \overline{u}_8 = -\overline{u}_9 = -\mathbf{1}$
		$\overline{u}_9 = \mathbf{1}$

realistic, physically motivated models that involve numerical solutions to differential equations or special functions that are not natively supported by the popular automatic differentiation frameworks.

In this paper, we demonstrate how to incorporate exoplanet-specific functions into these frameworks and derive the functions needed to characterize exoplanets using gradient-based methods applied to radial velocity, astrometry, and transit datasets. Similar derivations would certainly be tractable for microlensing, direct imaging, and other exoplanet characterization methods, but these are left to a future paper.

4.1. An introduction to automatic differentiation

First of all, we want to emphasize again that automatic differentiation is distinct from both numerical and symbolic differentiation. It is a method for computing the *value* of the derivative of a piece of code, without symbolically differentiating the operations.

Example:

$$f(x) = u(x) v(x) \quad (1)$$

$$\frac{df(x)}{dx} = \frac{du(x)}{dx} v(x) + u(x) \frac{dv(x)}{dx} \quad (2)$$

Instead, define $x \rightarrow x + \epsilon \delta x$ where $\epsilon^2 = 0$. Therefore,

$$u(x + \epsilon \delta x) = u(x) + \frac{du(x)}{dx} \epsilon \delta x \quad (3)$$

This equality is exact because of our definition that $\epsilon^2 = 0$. Therefore,

$$f(x) + \epsilon \delta f = u(x + \epsilon \delta x) v(x + \epsilon \delta x) \quad (4)$$

$$= \left[u(x) + \frac{du(x)}{dx} \epsilon \delta x \right] \left[v(x) + \frac{dv(x)}{dx} \epsilon \delta x \right] \quad (5)$$

$$= u(x) v(x) + \epsilon \left[\frac{du(x)}{dx} v(x) + u(x) \frac{dv(x)}{dx} \right] \delta x \quad (6)$$

The basic idea is that, at its roots, a computer program is the composition of simpler operations. If the derivatives of the simplest operations are known, the automatic differentiation library can automatically apply the chain rule to each step of the calculation to accumulate the derivative of the full program.

4.2. Choice of modeling framework

At present, there are many model-building libraries available in every programming language, and it is not clear which should be preferred. Since it is a popular language for code development in astrophysics, we focus on libraries with interfaces implemented in the `Python` programming language. The two most popular libraries for model building in `Python` are the well-supported and actively-developed packages `TensorFlow` (Abadi et al. 2016) and `PyTorch` (Paszke et al. 2017). Each of these libraries come with extensions that enable probabilistic modeling and gradient-based inference: `tensorflow-probability` (Dillon et al. 2017) and `Pyro` (Bingham et al. 2018). Both of these libraries (`TensorFlow` and `PyTorch`) and their inference engines are built for high performance machine learning and, while many of those features are also useful for our purposes, some of the inference goals in astrophysics are somewhat different. For example, the primary posterior inference method implemented by these packages is variational inference *DFM: (CITE)*. *DFM: We discuss the application of variational inference later in this paper*, but the primary inference method used in astrophysics is Markov chain Monte Carlo (MCMC) and the implementations of gradient-based MCMC (HMC, NUTS etc) within these packages is not as fully featured as the state-of-the-art MCMC inference packages like `PyMC3` (Salvatier et al. 2016) and `Stan` (Carpenter et al. 2015; Carpenter et al. 2017). So, for the purposes of this paper, we restrict our focus to `PyMC3` and its modeling engine `Theano` (Theano Development Team 2016).

`Theano` is a model building framework for `Python` that, like `TensorFlow`, provides an interface for defining a computational graph that can be used to efficiently compute a model and its derivatives. It was originally developed for deep learning, but it was adopted by the `PyMC3` project and extended to support probabilistic modeling and gradient-based inference methods like Hamiltonian Monte Carlo, no U-turn sampling, and variational inference. In our experiments, the combination of `Theano` and `PyMC3` provided the best balance of modern inference features, ease of use, and computational efficiency of all the existing `Python`-first modeling frameworks. The `Stan` project provides a more mature and feature-rich set of inference methods, but it is more difficult to build and debug the complicated physical models that are required by exoplanet applications using the `Stan` modeling language. Furthermore, as discussed below, we must provide custom astronomy-specific operations as part of this library and, while it is possible to extend the `Stan` math library, we found it to be much easier to develop, test, and release the necessary features using `PyMC3` and `Theano`.

5. INFERENCE METHODS

Gradient-based methods can be applied to both optimization and inference tasks. For example, the task of finding the maximum likelihood or maximum a posteriori parameters can be significantly accelerated using the gradient of the objective with respect to the model parameters. While computing the gradient will be somewhat more expensive than computing only the value of the function, it is generally significantly more efficient than using finite difference methods. Furthermore, since the gradients computed using automatic differentiation is exact, unlike finite difference which will suffer from numerical issues unless the step size is carefully tuned, the lack of numerical noise generally reduces the number of steps the optimizer must take before reaching the optimum.

5.1. *Optimization*

One standard inference procedure is maximum likelihood or maximum a posteriori inference by optimization. In this case, the objective (either the likelihood or posterior function) is maximized (numerically) with respect to the input parameters. Operations like this are commonly performed within the astronomical literature (CITE) and gradient-free methods are commonly used for this purpose (LM, NM). These methods can work reasonably well in low dimensions when the objective is well behaved, but this performance generally degrades rapidly with the number of parameters. In other scientific fields, gradient-based optimization is the industry standard (CITE), and these methods have been demonstrated to scale to large parameter spaces. In astronomy, it is now common to use the **BFGS** optimization routine (CITE) combined with finite difference estimates of the derivatives. These methods can be applied to larger problems, but the computational cost of evaluating the model scales with the number of dimensions and the results are generally sensitive to the choice of step size in the finite difference. Furthermore, the numerical noise introduced by the finite difference calculations reduces the performance of the optimization routine and increases the number of model evaluations needed to reach the optimum.

In this paper, we describe the computations needed to compute the required gradients using backpropagation and these derivatives can be used to substantially improve the performance of optimization applied to exoplanet datasets.

DFM: Make an example.

For example, we computed the number of function evaluations and total run time needed to find an estimate of the maximum likelihood parameters for a simulated radial velocity data set and compared the precision of this estimate to the *DFM: true* maximum. In this case, finite difference required *DFM: how many?* as many more function evaluations.

Similarly, we compare the performance of finite difference and exoplanet for optimizing the parameters for a transiting exoplanet. In this case... *DFM: finish this.*

5.2. *Sampling*

A popular method for evaluating posterior integrals is to use Markov chain Monte Carlo (MCMC). These methods can be substantially accelerated using gradients of the log probability with respect to the input parameters using Langevin or Hamiltonian methods.

DFM: Talk about Langevin methods? CITE things

5.3. Variational inference

6. ORBITAL CONVENTIONS

A general introduction to Keplerian orbits is given by [Murray & Correia \(2010\)](#); here we describe the implementation choices specific to the *exoplanet* codebase. We follow a set of internally consistent orbital conventions that also attempts to respect as many of the established conventions of the stellar and exoplanetary fields as possible. The location of a body on a Keplerian orbit with respect to the center of mass is given in the perifocal plane by

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (7)$$

By construction, the orbit of the body in the perifocal frame is constrained entirely to the $x - y$ plane (see Figure 2). The range of orbital convention choices mainly stems from how the location of the body in the perifocal frame is converted to the observer frame,

$$\mathbf{R} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \quad (8)$$

We choose to orient the $\hat{\mathbf{X}}$ unit vector in the north direction, $\hat{\mathbf{Y}}$ in the east direction, and $\hat{\mathbf{Z}}$ towards the observer. Under this convention, the inclination of an orbit is defined by the dot-product of the orbital angular momentum vector with $\hat{\mathbf{Z}}$, conforming to the common astrometric convention that a zero inclination ($i = 0^\circ$) orbit is face-on, with the test body orbiting in a counter-clockwise manner around the primary body.

In the stellar and exoplanetary fields, there is less agreement on which segment of the line of nodes is defined as the *ascending* node. We choose to define the ascending node as the point where the orbiting body crosses the plane of the sky moving *away* from the observer (i.e., crossing from $Z > 0$ to $Z < 0$). This convention has historically been the choice of the visual binary field; the opposite convention occasionally appears in exoplanet and planetary studies.

To implement the transformation from perifocal frame to observer frame, we consider the rotation matrices

$$\mathbf{P}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (9)$$

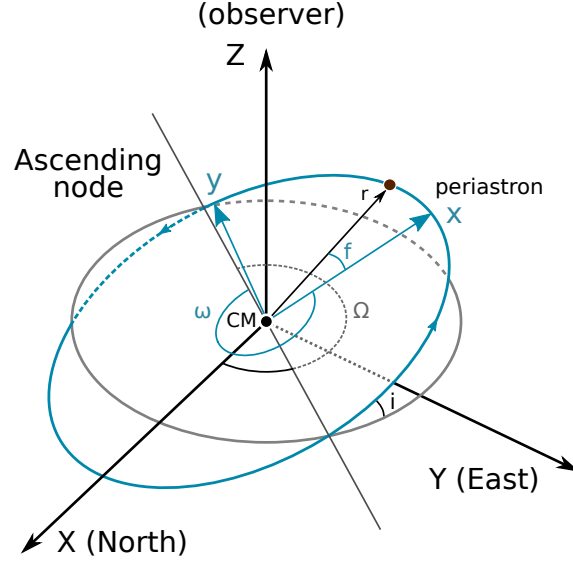



Figure 2. The conventions used to orient the perifocal frame (x, y, z) relative to the observer frame (X, Y, Z) . 

$$\mathbf{P}_z(\phi) = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (10)$$

which result in a *clockwise* rotation of the axes, as defined using the right hand rule.¹

To convert a position \mathbf{r} in the perifocal frame to the observer frame \mathbf{R} (referencing Figure 2), three rotations are required: (i) a rotation about the z -axis through an angle ω so that the x -axis coincides with the line of nodes at the ascending node (ii) a rotation about the x -axis through an angle $(-i)$ so that the two planes are coincident and finally (iii) a rotation about the z -axis through an angle Ω . Applying these rotation matrices yields

$$\mathbf{R} = \mathbf{P}_z(\Omega)\mathbf{P}_x(-i)\mathbf{P}_z(\omega)\mathbf{r}. \quad (11)$$

As a function of true anomaly f , the position in the observer frame is given by

$$\begin{aligned} X &= r[\cos \Omega(\cos \omega \cos f - \sin \omega \sin f) - \sin \Omega \cos i(\sin \omega \cos f + \cos \omega \sin f)] \\ Y &= r[\sin \Omega(\cos \omega \cos f - \sin \omega \sin f) + \cos \Omega \cos i(\sin \omega \cos f + \cos \omega \sin f)] \\ Z &= -r \sin i(\sin \omega \cos f + \cos \omega \sin f). \end{aligned} \quad (12)$$

¹ This means when we look down the z -axis, for a positive angle ϕ , it would be as if the x and y axes rotated clockwise. In order to find out what defines counter-clockwise when considering the other rotations, we look to the right hand rule and cross products of the axes unit vectors. Since $\hat{x} \times \hat{y} = \hat{z}$, when looking down the z axis the direction of the x -axis towards the y -axis defines counter-clockwise. Similarly, we have $\hat{y} \times \hat{z} = \hat{x}$, and $\hat{z} \times \hat{x} = \hat{y}$.

Simplifying the equations using the sum angle identities, we find

$$\begin{aligned} X &= r(\cos \Omega \cos(\omega + f) - \sin(\Omega) \sin(\omega + f) \cos(i)) \\ Y &= r(\sin \Omega \cos(\omega + f) + \cos(\Omega) \sin(\omega + f) \cos(i)) \\ Z &= -r \sin(\omega + f) \sin(i). \end{aligned} \tag{13}$$

7. PARAMETERIZING AN ORBIT

At its heart, `exoplanet` models Keplerian orbits. There is some support for relaxing this assumption (REFERENCE N-BODY W REBOUND), but it is important that we clearly present how to represent a Keplerian orbit before generalizing. In `exoplanet` these orbits are specified with respect to a single central body (generally the most massive body) and then a system of non-interacting bodies orbiting the central. This is a good parameterization for exoplanetary systems and binary stars, but it is sometimes not sufficient for systems with multiple massive bodies where the interactions will be important to the dynamics.

The central body is defined by any two of radius R_\star , mass M_\star , or density ρ_\star . The third physical parameter can always be computed using the other two so `exoplanet` will throw an error if three are given (even if they are numerically self-consistent).

The orbits are then defined by:

- the period P or semi-major axis a of the orbit,
- the time of conjunction t_0 or time of periastron passage t_p ,
- the planet’s radius R_{pl} and mass M_{pl} ,
- the eccentricity e , argument of periastron ω and the longitude of ascending node Ω , and
- the inclination of the orbital plane i or the impact parameter b .

A `KeplerianOrbit` object in `exoplanet` will always be fully specified. For example, if the orbit is defined using the period P , the semi-major axis a will be computed using Kepler’s third law.

8. PARAMETERIZING TRANSITING PLANETS

It is common practice to parameterize a transiting planet using period P , time of conjunction t_0 , and impact parameter b . If parameters are well constrained then specific choices about priors won’t be important, but the eccentricity and argument of periastron are rarely well constrained by the transit alone, so it can be important to consider the details of the parameterization.

From geometry, we can derive that expected prior distribution for $\cos i$ is uniform from -1 to 1 . This doesn’t directly translate to a uniform prior on b . Instead, if we want to sample in b we need to take into account the Jacobian in the change of

variables

$$p(\cos i) = \left| \frac{db}{d \cos i} \right| p(b) \quad . \quad (14)$$

Since,

$$b = \frac{a}{R_\star} \cos i \left[\frac{1 - e^2}{1 + e \sin \omega} \right] \quad , \quad (15)$$

the relevant Jacobian is

$$\frac{db}{d \cos i} = \frac{R_\star}{a} \left[\frac{1 + e \sin \omega}{1 - e^2} \right] \quad . \quad (16)$$

Similarly, in order for a transit to occur, the impact parameter must satisfy

$$|b| < 1 + R_{\text{pl}}/R_\star \quad . \quad (17)$$

In other words, the conditional prior probability for the impact parameter (conditioned on the assumption that the planet transits and the other orbital parameters) is

$$p(b | a, e, \omega, R_{\text{pl}}, R_\star) = \frac{R_\star}{a} \left[\frac{1 + e \sin \omega}{1 - e^2} \right] \begin{cases} 1/[2(1 + R_{\text{pl}}/R_\star)] & \text{if } |b| < 1 + R_{\text{pl}}/R_\star \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

Similarly, the prior on the planet's radius should also depend on the detectability of the signal, but this will only be important for small planets with radius error bars that span the detection threshold which won't be common.

As discussed by *DFM: CITE exofast*, there is a strong degeneracy between e , b and ω that is introduced by the fact that the duration is the main observable that constrains these three quantities. While it is possible to change variables to fit in duration instead of e and analytically marginalize over the branches of the solution for e , both the bookkeeping and reparameterization is difficult to get right. In particular, NUTS requires that all parameters have infinite support, but the allowed range of the duration parameter depends on the branch and the other parameters. Furthermore, there is a funnel situation since at some values of ω , all values of e are identical.

9. THE CUSTOM OPERATIONS PROVIDED BY EXOPLANET

9.1. A solver for Kepler's equation

A key assumption that is often made when fitting exoplanets is that the planets are orbiting the central star on independent bound Keplerian orbits (CITE). To compute the coordinates of a planet on its orbit, we must solve Kepler's equation

$$M = E - e \sin E \quad (19)$$

for the eccentric anomaly E at fixed eccentricity e and mean anomaly M . This equation must be solved numerically and there is a rich literature discussing methods for this implementation. We have found the method from CITE to provide the right balance of numerical stability (with a relative accuracy of about 10^{-20} for all values of M and e) and computational efficiency.

As well as solving Kepler's equation, we must also efficiently evaluate the gradients (or, more precisely, *back-propagate* the gradients) of E with respect to e and M . After solving for E , this can be computed efficiently and robustly using implicit differentiation. For numerical methods like this, it is often possible to compute the gradients without applying automatic differentiation to the implementation directly and that is generally preferred for the purposes of numerical stability. In this case, the relevant gradients are

$$\frac{dE}{de} = \frac{\sin E}{1 - e \cos E} \quad (20)$$

$$\frac{dE}{dM} = \frac{1}{1 - e \cos E} \quad (21)$$

In practice, we have found it more efficient to fuse the calculations of the eccentric anomaly E and the true anomaly f working only with the sine and cosines of these quantities. To compute the true anomaly for a given eccentric anomaly, the relation is

$$\tan \frac{f}{2} = \sqrt{\frac{1+e}{1-e}} \tan \frac{E}{2} \quad (22)$$

and we can compute this efficiently using the following trigonometric identities

$$\tan \frac{E}{2} = \frac{\sin E}{1 + \cos E} \quad (23)$$

$$\sin f = \frac{2 \tan f/2}{1 + \tan^2 f/2} \quad (24)$$

$$\cos f = \frac{1 - \tan^2 f/2}{1 + \tan^2 f/2} \quad (25)$$

with the special case $\sin f = 0$ and $\cos f = -1$ when $\cos E = -1$.

Then, the relevant derivatives can be evaluated using the results ($\sin f$ and $\cos f$) of the forward solve:

$$\frac{df}{dM} = \frac{(1 + e \cos f)^2}{(1 - e^2)^{3/2}} \quad (26)$$

$$\frac{df}{de} = \frac{(2 + e \cos f) \sin f}{1 - e^2} \quad (27)$$

DFM: Add a plot showing the relative cost of the forward and reverse evaluation.

9.2. Transit light curves

Another custom operation included as part of `exoplanet` is for calculating model light curves for the transits (and occultations) including limb-darkening and exposure time integration. This core functionality is provided by the code `starry` (Luger et al. 2019) *DFM: cite limbdark*. Briefly, `starry` represents the star using a spherical harmonic representation of its surface brightness and then uses this representation to analytically integrate the surface of the star to compute the model light curve and the derivatives with respect to the physical parameters. This algorithm is a generalization of the popular quadratic limb darkening model (Mandel & Agol 2002) to a much more flexible class of stellar surface maps *DFM: cite other limb darkening models*. For our purposes, `starry`’s key feature is the fact that it not only computes the model light curve (as has been done previously *DFM: cite*), but also the derivatives of this model with respect to the parameters of the system (the orbital parameters and the spherical harmonic coefficients). In the case of pure limb darkening, these derivatives are calculated analytically following *DFM: cite limbdark* while, in the more general case, the derivatives are accumulated using automatic differentiation in the C++ code (using Eigen *DFM: cite Eigen*). *DFM: Make a plot of the gradient light curves*. This means that `starry` can be included within the backpropagation framework needed by `exoplanet`.

From an interface perspective, `starry` is included as a set of custom Theano operations that compute the matrices in the `starry` equation as a function of the sky coordinates of the bodies in the system and the spherical harmonic coefficients. The linear algebra is performed directly in Theano. This means that the gradients are efficiently backpropagated using the linear algebra routines provided by Theano.

Exposure time integration—In many cases, the finite exposure time integration of transit light curves must be included in the model light curve in order to get correct constraints on the physical properties of the system. `exoplanet` includes several routines for integrating the model. First, it includes an oversampling routine as recommended by *DFM: cite kipping*, including generalizations to higher order integration schemes. This has the benefit that it is simple to use and can generally be applied to any model light curve. However, this method also introduces significant memory overhead because each data point requires many (at least tens) of model evaluations and these must be stored in memory for the purposes of backpropagation.

Because of this shortcoming, `exoplanet` also implements an adaptive exposure time integration routine directly in C++ that can be used for Keplerian orbits. In this case, the time integral is performed using an adaptive Simpson’s rule *DFM: cite* directly in the backend, reducing the computational cost and memory overhead. This routine works by recursively subdividing the exposure interval until a parabolic approximation to the flux in the subexposure matches the computed flux to a specified tolerance. The derivative of this integral is accumulated using during the forward pass.

The second aspect of exposure time integration incorporated into **exoplanet** applies to the Gaussian Process kernel. The standard **celerite** kernel assumes that each exposure time is infinitesimal in duration. This is an adequate approximation if the kernel varies slowly and smoothly across an exposure time; however, the finite exposure time modifies the kernel significantly when it is comparable to, or longer than, any of the timescales occurring in the components of the kernel. In appendix C we present novel expressions for the time-integrated **celerite** kernel, which retains the necessary properties for fast computation when the exposures times do not overlap.

Performance—As demonstrated in the papers describing this algorithm (Luger et al. 2019) *DFM: cite limbdark*, the performance of the algorithms implemented in **starry** is competitive with the other standard light curve implementations *DFM: cite*. Figure *DFM: whatever* demonstrates this in the context of **exoplanet**. *DFM: Make a runtime figure and explain it.*

9.3. Scalable Gaussian Processes for time series

exoplanet also includes an implementation of scalable Gaussian Processes (GPs) using the **celerite** algorithm (Foreman-Mackey et al. 2017). To make this algorithm compatible with the **Theano** automatic differentiation framework, we also derived the backpropagation functions for **celerite** (Foreman-Mackey 2018). We implemented these functions in C++ as a custom **Theano** operation so that they can be seamlessly integrated into a **PyMC3** probabilistic model.

celerite provides an algorithm for performing exact inference with GP models that scales linearly with the number of data points, instead of the typical cubic scaling. This scaling is achieved by exploiting structure in the covariance matrix when the covariance function is restricted to be a specific stationary form, with one-dimensional inputs. The observations can, however, be un-evenly sampled and heteroschedastic without any loss of accuracy or computational performance. As discussed by Foreman-Mackey et al. (2017), this covariance function can be interpreted as the covariance generated by a mixture of stochastically-driven, damped simple harmonic oscillators, which makes this a good model for the variability of stars in the time domain. **exoplanet** provides an interface for constructing covariance functions that satisfy the constraints of the **celerite** algorithm and for solving the relevant linear algebra.

This operation is useful for many **exoplanet** applications where the datasets are large enough that traditional GP modeling is intractable. For example, the time series for a typical short cadence target from the **TESS** mission has a light curve with about 20,000 data points. Evaluating a GP model on these data using high-performance general linear algebra libraries would take *DFM: how long?*.

10. DISTRIBUTIONS

exoplanet also comes equipped with several custom “distributions” that can be used to simplify modeling for **exoplanet** specific applications, but some also have broader applicability.

10.1. General distributions

Angles—There are several angles that must be included in exoplanet applications. For example, the argument of periastron and the phase of the orbit. This can be difficult to sample using most MCMC algorithms because of the periodic boundary conditions of the parameter space. Therefore, for angles the argument of periastron ω that have a related amplitude (eccentricity e in this case), it is common practice to fit for e and ω together using

$$h = \sqrt{e} \sin \omega \quad \text{and} \quad k = \sqrt{e} \cos \omega \quad (28)$$

as the parameters with the constraint $h^2 + k^2 < 1$. It has been demonstrated (CITE) that this still imposes a uniform prior on both e and ω . This is not a good parameterization, however, when using gradient based inference schemes because these do not do a good job of handling hard constraints $0 \leq e < 1$. Instead, for angular parameters, we sample in

$$n_1 \sim \mathcal{N}(0, 1) \quad \text{and} \quad n_2 \sim \mathcal{N}(0, 1) \quad (29)$$

where $\mathcal{N}(0, 1)$ is the standard normal and compute the angle as $\omega = \arctan2(n_1, n_2)$. This also produces a uniform prior on ω , but there are no longer any hard boundaries in the problem, at the cost of one extra parameter.

One issue that arises with this parameterization is that the joint posterior in n_1 and n_2 can become “funnel shaped” when the angle is well constrained. This dynamic range can cause serious performance issues for gradient-based samplers (CITE). To deal with this, we regularize the density as

$$\log p(n_1, n_2) = -\frac{n_1^2 + n_2^2}{2} + \alpha \log(n_1^2 + n_2^2) + \text{const} \quad (30)$$

where α is a parameter that controls the strength of the regularization. We find that a default value of $\alpha = 10$ gives good performance in most cases.

Unit vector—Another generally useful distribution is the distribution over unit vectors. We follow (CITE Stan) and include a custom distribution for PyMC3 that provides support for unit vectors by reparameterizing as $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and computing the unit vector as $\mathbf{u} = \mathbf{n} / \|\mathbf{n}\|_2$.

10.2. Quadratic limb darkening parameters

`exoplanet` has support for limb darkening profiles of arbitrary order as described in LIMB DARK, but a commonly used model is the quadratic limb darkening model so `exoplanet` includes some features to make this easier. The algorithm for efficiently evaluating the light curve for a transit of a quadratically limb darkened star was first presented by Mandel & Agol (2002), but the light curve model in `exoplanet` is

evaluated following Agol et al. including the performance and numerical stability improvements described in that paper. The form of the limb darkening law is

$$I(\mu)/I(1) = 1 - u_1(1 - \mu) - u_2(1 - \mu)^2 \quad (31)$$

where $\mu = \sqrt{1 - b^2}$ and b is the distance from the center of the star in units where the stellar radius is one. For this limb darkening profile to be everywhere positive and monotonically decreasing as a function of b , the parameters u_1 and u_2 must satisfy the following relations

$$u_1 + u_2 < 1 \quad (32)$$

$$u_1 > 0 \quad (33)$$

$$u_1 + 2u_2 > 0 \quad (34)$$

It has been demonstrated (by Kipping 2013) that these constraints can be enforced by reparameterizing as

$$q_1 = (u_1 + u_2)^2 \quad (35)$$

$$q_2 = \frac{u_1}{2(u_1 + u_2)} \quad (36)$$

with the simpler constraints

$$0 < q_1 < 1 \quad (37)$$

$$0 < q_2 < 1 \quad (38)$$

Kipping (2013) argues that this is an “uninformative” parameterization because the log determinant of the Jacobian of the transformation is zero, *DFM: discuss details of this*: but it’s worth noting that the marginal priors $p(u_1)$ and $p(u_2)$ are not uniform.

In *exoplanet*, we make one further change of variables to

$$\tilde{q} = \log(q) - \log(1 - q) \quad (39)$$

where \tilde{q} has support across the full real line. This is not an uninformative change of variables so we must take this into account by adding the log determinant of the Jacobian to the log probability. For this change of variables, the correction is

$$\log\left(\frac{dq}{d\tilde{q}}\right) = \log q + \log(1 - q) \quad (40)$$

10.3. Radius and impact parameter

A parameterization has been proposed to parameterize the radius ratio r and impact parameter b to be sampled uniformly within the trapezoid $r_{\min} < r < r_{\max}$ and $0 < b < 1 + r$ (CITE). This parameterization is included within *exoplanet*, but implies a non-uniform marginal for radius ratio (*DFM: make plot*).

The marginal density for the Espinoza distribution is

$$p(r, b) = \begin{cases} 1/A & \text{if } r_{\min} \leq r < r_{\max} \text{ and } b \leq 1 + r \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

where

$$A = \int_{r_{\min}}^{r_{\max}} \int_0^{1+r} db dr \quad (42)$$

$$= \int_{r_{\min}}^{r_{\max}} (1 + r) dr \quad (43)$$

$$= (r_{\max} - r_{\min}) + \frac{1}{2} (r_{\max}^2 - r_{\min}^2) \quad (44)$$

therefore

$$p(r) = \frac{1}{A} \int_0^{1+r} db \quad (45)$$

$$= \begin{cases} (1 + r)/A & \text{if } r_{\min} \leq r < r_{\max} \\ 0 & \text{otherwise} \end{cases} \quad (46)$$

This means that the inference will be biased towards measuring large radius planets. Instead, we argue that the less informative prior would be uniform in log radius or log radius ratio. Therefore, we recommend reparameterizing as

$$r \sim \text{LogUniform}(r_{\min}, r_{\max}) \quad (47)$$

$$b \sim \text{Uniform}(0, 1 + r) \quad (48)$$

which will introduce less bias when inferring radius ratios. This can be efficiently sampled using `exoplanet`.

11. EXAMPLES

12. DISCUSSION

This research was partially conducted during the Exostar19 program at the Kavli Institute for Theoretical Physics at UC Santa Barbara, which was supported in part by the National Science Foundation under Grant No. NSF PHY-1748958.

Conversations: Astro data group, Ruth Angus, Tom Barclay, Megan Bedell, Jonathan (Yoni) Brande, Tim Brandt, Trevor David, Greg Gilbert, Ed Gillen, Tyler Gordon, Christina Hedges, Dan Hey, Vedad Hodzic, Pierre Maxted, Hugh Osborn, Erik Petigura, Rich Teague, Vincent Van Eylen, and Alexa Villaume.

APPENDIX

A. SOLVING KEPLER'S EQUATION

A crucial component for all of the models in this paper is an accurate and high-performance solver for Kepler's equation

$$M = E - e \sin E \quad . \quad (A1)$$

We use the method presented by [Nijenhuis \(1991\)](#). Unlike most of the methods used in astrophysics, this algorithm *is not iterative*. This can lead to much better compiler optimization. Furthermore, this algorithm doesn’t require the evaluation of trigonometric functions directly. As well as improving performance, this also increases the accuracy of the results because it avoids catastrophic cancellations when evaluating $E - \sin E$ and $1 - \cos E$ by evaluating these differences directly using a series expansion.

The procedure is as follows:

1. Initial estimates are selected for the parameters by partitioning the parameter space into 4 regions and applying a set of heuristics specific to each part of parameter space.
2. This estimate is refined using either a single iteration of Halley’s second order root finding algorithm (with the sine function with its series expansion) or a variant of the cubic approximation from [Mikkola \(1987\)](#).
3. Finally, the result is updated by a single step of a high-order generalized Newton method where the order of the method controls the target accuracy of the method. In the **exoplanet** implementation, we use a 4th order update.

The details of the implementation are exactly ported from the implementation in [Nijenhuis \(1991\)](#) so we won’t reproduce that here, but we did an experiment to compare the performance and accuracy of this method to some implementations of Kepler solvers commonly used for exoplanet fitting. Specifically we compare to the implementations from the **batman** transit fitting framework ([Kreidberg 2015](#)) and the **RadVel** radial velocity fitting library ([Fulton et al. 2018](#)). In both cases, the solver is implemented in C and exposed to Python using the Python C-API directly or Cython ([Behnel et al. 2011](#)) respectively. Similarly, the implementation in **exoplanet** is written in C++ with Python bindings exposed by Theano ([Theano Development Team 2016](#)).

To test the implementation, we generated 100000 true eccentric anomalies E_{true} uniformly distributed between 0 and 2π . Then, for a range of eccentricities e , we computed the mean anomaly M using Equation (A1) and used each library to solve Equation (A1) for E_{calc} . The top panel of Figure 3 shows the average computational cost per solve of Equation (A1) for each library as a function of eccentricity. The implementation of the [Nijenhuis \(1991\)](#) algorithm in **exoplanet** is more than an order of magnitude more efficient than the other methods and this cost does not depend sensitively on the eccentricity of the orbit. It is worth noting that both **batman** and **exoplanet** feature “fused” solvers that return both the eccentric anomaly and the true anomaly whereas **RadVel** only computes the eccentric anomaly so there is a small amount of computational overhead introduced into both **exoplanet** and **batman** when compared to **RadVel**.

The bottom panel of Figure 3 shows the 90th percentile of the distribution of absolute errors when comparing E_{true} and E_{calc} for each method as a function of

eccentricity. Across all values of eccentricity, **exoplanet** computes the eccentric anomaly with an accuracy of better than 15 decimal places. In some cases, the **RadVel** solver is slightly better, but at some eccentricities, the error is more than an order of magnitude worse for **RadVel** than for **exoplanet**. In all cases, the accuracy of the implementation from **batman** is several orders of magnitude worse than the other implementations. This accuracy could be improved by decreasing the convergence tolerance in the iterative solver at the cost of longer run times, but the current value of 10^{-7} is currently hard coded².

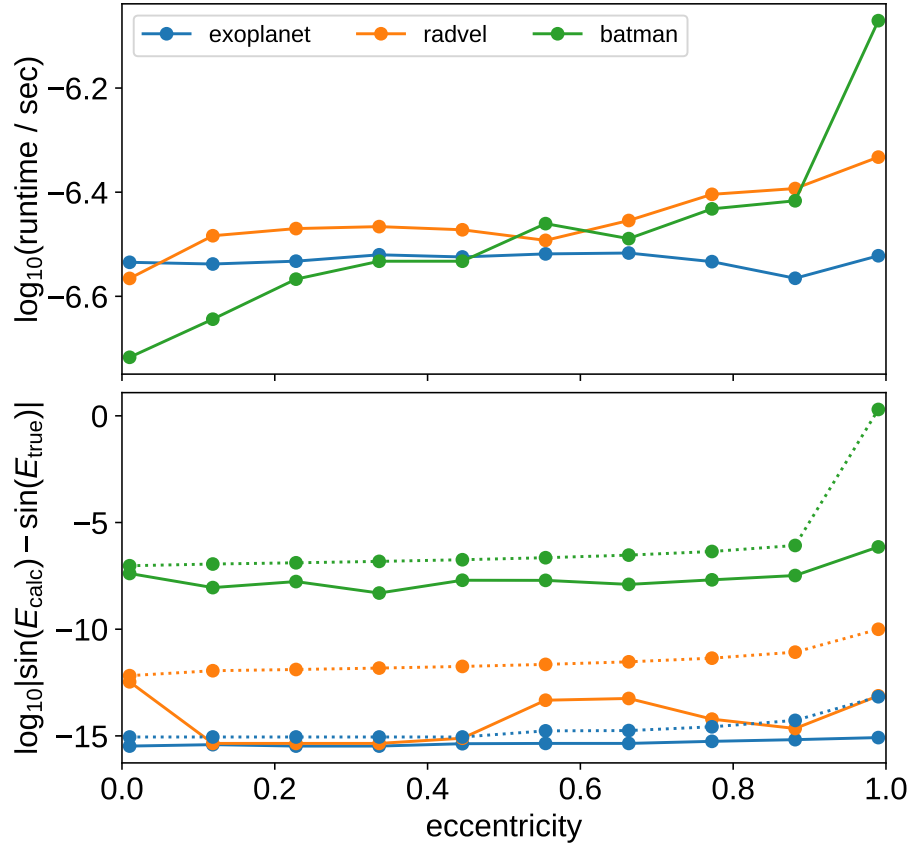


Figure 3. The performance of the Kepler solver used by **exoplanet** compared to the solvers from the **RadVel** (Fulton et al. 2018) and **batman** (Kreidberg 2015) libraries. *top*: The average wall time required to solve Kepler’s equation once for the eccentric anomaly E conditioned on eccentricity e and mean anomaly M . Smaller numbers correspond to faster solutions. *bottom*: The 90-th percentile of the absolute error of the computed eccentric anomaly for 100000 true values of E in the range $0 \leq E < 2\pi$. Smaller numbers correspond to more accurate solutions. https://github.com/lkreidberg/batman/blob/70f2c0c609124bdf4f17041bf09d5426f3c93334/c_src/_rsky.c#L45

² https://github.com/lkreidberg/batman/blob/70f2c0c609124bdf4f17041bf09d5426f3c93334/c_src/_rsky.c#L45

B. CONTACT POINTS

In the plane of the orbit, S_0 , the coordinates of the orbit satisfy the equation

$$(x_0 - a e)^2 + \frac{y_0^2}{1 - e^2} = a^2 \quad . \quad (\text{B2})$$

To rotate into the observing plane, we perform the following transformation

$$\mathbf{x}_2 = R_i R_\omega \mathbf{x}_0 \quad (\text{B3})$$

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{pmatrix} \begin{pmatrix} \cos \omega & \sin \omega & 0 \\ -\sin \omega & \cos \omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 0 \end{pmatrix} \quad (\text{B4})$$

In this space, the planet will transit whenever

$$z_2 < 0 \quad \text{and} \quad x_2^2 + y_2^2 < L^2 \quad (\text{B5})$$

where $L = R_\star + R_P$. The contact point therefore occurs when

$$\hat{x}_2^2 + \hat{y}_2^2 = L^2 \quad (\text{B6})$$

where the hat indicates that

Using the inverse of Equation (B3) to re-write Equation (B2) in terms of x_2 and y_2 , we find the following quadratic equation

$$A x_2^2 + B x_2 y_2 + C y_2^2 + D x_2 + E y_2 + F = 0 \quad (\text{B7})$$

with

$$A = (e^2 \cos^2 \omega - 1) \cos^2 i \quad (\text{B8})$$

$$B = 2 e^2 \cos i \sin \omega \cos \omega \quad (\text{B9})$$

$$C = e^2 \sin^2 \omega - 1 \quad (\text{B10})$$

$$D = 2 a e (1 - e^2) \cos^2 i \cos \omega \quad (\text{B11})$$

$$E = 2 a e (1 - e^2) \sin \omega \cos i \quad (\text{B12})$$

$$F = a^2 (e^2 - 1)^2 \cos^2 i \quad (\text{B13})$$

The pair of quadratic equations defined by Equation (B6) and Equation (B7) can be combined to give a quartic equation for x_2

$$a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3 + a_4 x_2^4 = 0 \quad (\text{B14})$$

where

$$a_0 = (CL^2 - EL + F)(CL^2 + EL + F) \quad (\text{B15})$$

$$a_1 = -2(BEL^2 - CDL^2 - DF) \quad (\text{B16})$$

$$a_2 = 2ACL^2 + 2AF - B^2L^2 - 2C^2L^2 - 2CF + D^2 + E^2 \quad (\text{B17})$$

$$a_3 = 2(AD + BE - CD) \quad (\text{B18})$$

$$a_4 = A^2 - 2AC + B^2 + C^2 \quad (\text{B19})$$

which can be solved symbolically (CITE Kipping) or numerically.

Edge-on orbits—For an edge-on orbit, $\cos i = 0$ and the contact points occur at

$$x_2 = \pm L \quad (\text{B20})$$

$$y_2 = 0 \quad , \quad (\text{B21})$$

but care must be taken when evaluating z_2 . To do this, we substitute Equation (B20) into Equation (B2) to get the following quadratic equation for z_2

$$b_0 + b_1 z_2 + b_2 z_2^2 = 0 \quad (\text{B22})$$

where

$$b_{0,\pm} = L^2 (e^2 \cos^2 \omega - 1) \mp 2 a e L \cos \omega (e^2 - 1) + a^2 (e^2 - 1)^2 \quad (\text{B23})$$

$$b_{1,\pm} = -2 a e \sin \omega (e^2 - 1) \pm 2 e^2 L \sin \omega \cos \omega \quad (\text{B24})$$

$$b_{2,\pm} = e^2 \sin^2(\omega) - 1 \quad . \quad (\text{B25})$$

There are 4 solutions to this system of which we are only interested in the ones where $z_2 < 0$ (the others are the contact points for the occultation).

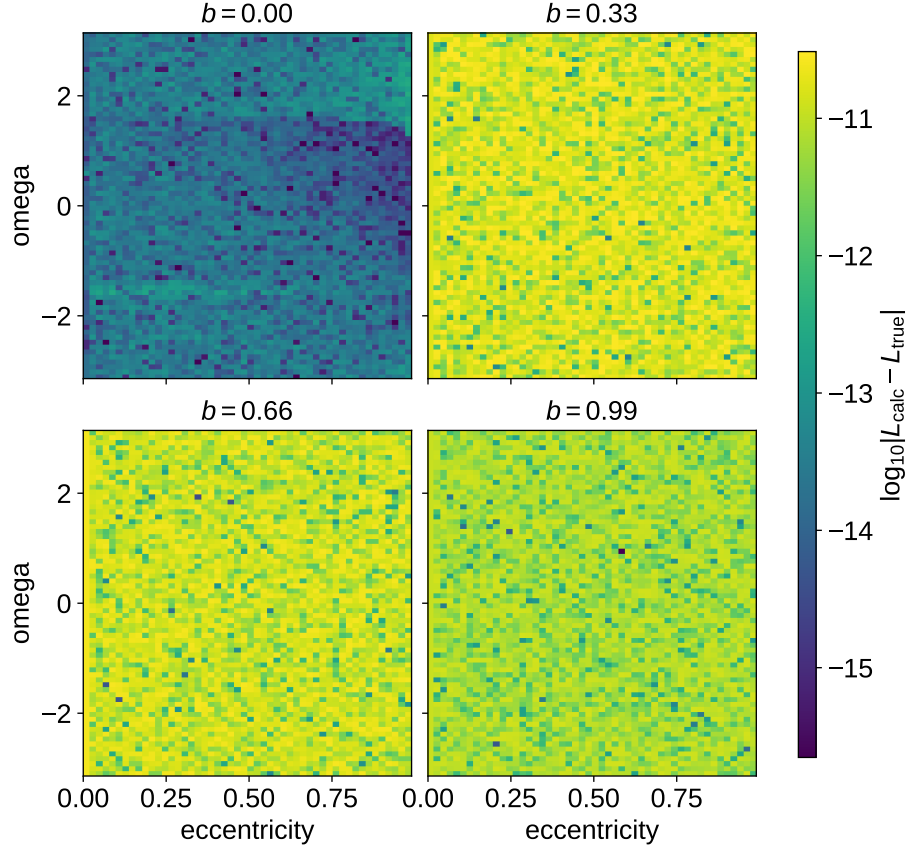



Figure 4. This is a figure. 

C. EXPOSURE TIME INTEGRATION FOR CELERITE MODELS

In this section we give expressions for the *exposure time*-integrated versions of the **celerite** kernel and its power spectrum. It turns out that the time-integrated **celerite** kernel remains expressible as a semi-separable matrix when the exposures do not overlap, but when exposures overlap for non-zero time lags, the kernel no longer is semi-separable.

For the **celerite** model, the general kernel function is expressed as

$$k(\tau) = a e^{-c\tau} \cos(d\tau) + b e^{-c\tau} \sin(d\tau) \quad , \quad (\text{C26})$$

where $\tau = |t_n - t_m|$ is the lag between times t_n and t_m , and the parameters of the model are a , b , c , and d . To find the exposure time-integrated kernel, we must evaluate the following integral

$$k_{\Delta}(\tau) = \frac{1}{\Delta^2} \int_{t_i - \Delta/2}^{t_i + \Delta/2} dt \int_{t_j - \Delta/2}^{t_j + \Delta/2} dt' k(|t - t'|) \quad (\text{C27})$$

where Δ is the length of the exposure.

When $\Delta \leq \tau$ the exposures do not overlap and the exposure time integrated model still falls within the class of **celerite** models:

$$k_{\Delta \leq \tau}(\tau) = A e^{-c\tau} \cos(d\tau) + B e^{-c\tau} \sin(d\tau) \quad , \quad \text{✍️} (\text{C28})$$

where c and d have not changed, but the amplitudes are given by the updated expressions:

$$A = \frac{2 C_1 [\cosh(c \Delta) \cos(d \Delta) - 1] - 2 C_2 \sinh(c \Delta) \sin(d \Delta)}{\Delta^2 (c^2 + d^2)^2}, \quad \text{✍️} (\text{C29})$$

and

$$B = \frac{2 C_2 [\cosh(c \Delta) \cos(d \Delta) - 1] + 2 C_1 \sinh(c \Delta) \sin(d \Delta)}{\Delta^2 (c^2 + d^2)^2}, \quad \text{✍️} (\text{C30})$$

with

$$C_1 = a c^2 - a d^2 + 2 b c d \quad \text{and} \quad C_2 = b c^2 - b d^2 - 2 a c d \quad . \quad (\text{C31})$$

For most real datasets, the constraint that $\Delta \leq \tau$ will be satisfied because, for a light curve from a single telescope and a uniform exposure time, the times of each cadence will be separated by greater than the exposure time. In this case, **celerite** can still be used to efficiently evaluate the GP model.

For the case of overlapping exposures $0 \leq \tau < \Delta$, the result is not as simple because of the absolute value in the definition of τ . In this case, we separate the

integral into three integrals that can be easily evaluated

$$\begin{aligned}
\Delta^2 k_{0 \leq \tau < \Delta}(\tau) &= \int_{t_j + \Delta/2}^{t_i + \Delta/2} dt \int_{t_j - \Delta/2}^{t_j + \Delta/2} dt' k(t - t') \\
&+ \int_{t_i - \Delta/2}^{t_j + \Delta/2} dt \int_{t_j - \Delta/2}^t dt' k(t - t') \\
&+ \int_{t_i - \Delta/2}^{t_j + \Delta/2} dt \int_t^{t_j + \Delta/2} dt' k(t' - t) \quad . \quad (C32)
\end{aligned}$$

Evaluating and simplifying the result gives

$$\begin{aligned}
k_{0 \leq \tau < \Delta}(\tau) &= k_{\Delta \leq \tau}(\tau) + 2 [(a c + b d) (c^2 + d^2) (\Delta - \tau) \\
&- C_1 \sinh(c \Delta - c \tau) \cos(d \Delta - d \tau) \\
&+ C_2 \cosh(c \Delta - c \tau) \sin(d \Delta - d \tau)] / [\Delta (c^2 + d^2)]^2 \quad \textcolor{blue}{\mathcal{C}} (C33)
\end{aligned}$$

where $k_{\Delta \leq \tau}(\tau)$ is the result from above where $\Delta \leq \tau$ (Equation C28).

The power spectrum of the time-integrated term derived above can be computed by taking the Fourier transform and, after some simplifications, it reduces to the expected result

$$\begin{aligned}
S_{\Delta}(\omega) &= S(\omega) \text{sinc}^2(\omega \Delta/2) \\
&= \sqrt{\frac{2}{\pi}} \left(\frac{(ac + bd)(c^2 + d^2) + (ac - bd)\omega^2}{\omega^4 + 2(c^2 - d^2)\omega^2 + (c^2 + d^2)^2} \right) \left(\frac{\sin(\frac{1}{2}\omega\Delta)}{\frac{1}{2}\omega\Delta} \right)^2 \quad (C34)
\end{aligned}$$

where $S(\omega)$ is the power spectrum of the original term and $\text{sinc}(x) = \sin(x)/x$ is the Fourier transform of a top hat representing the exposure. An interesting consequence of this form is that, since the sinc function goes to zero when $\omega \Delta = 2\pi n$ for integers $n \neq 0$, there is no power at integer multiples of the sampling frequency $f = \frac{2\pi}{\omega} = \Delta^{-1}$.

An important implementation note is that, while a non-integrated **celerite** term would generate a matrix with the amplitude a on the diagonal, the diagonal for the integrated term is not simply A because the lag on the diagonal is $\tau = 0 < \Delta$. This means that the diagonal entries in the covariance matrix for an time-integrated **celerite** model are

$$\begin{aligned}
k_{\Delta}(0) &= A + \frac{2}{[\Delta (c^2 + d^2)]^2} [(a c + b d) (c^2 + d^2) \Delta \\
&- C_1 \sinh(c \Delta) \cos(d \Delta) \\
&+ C_2 \cosh(c \Delta) \sin(d \Delta)] \quad (C35)
\end{aligned}$$

which can be simplified to

$$k_{\Delta}(0) = \frac{2 [(a c + b d) (c^2 + d^2) \Delta - C_1 + e^{-c \Delta} (C_1 \cos(d \Delta) + C_2 \sin(d \Delta))]}{\Delta^2 (c^2 + d^2)^2} \quad \textcolor{blue}{\mathcal{C}} (C36)$$

While, in general, this diagonal can be negative, for positive semi-definite kernels where $S(w) \geq 0$, the diagonal entry can also be computed as

$$k_{\Delta}(0) = \sqrt{2\pi} \int_{-\infty}^{\infty} S(\omega) \text{sinc}^2(\omega \Delta/2) d\omega \quad (\text{C37})$$

which will be everywhere non-negative since $S(\omega) \geq 0$ and $\text{sinc}^2(\omega \Delta/2) \geq 0$.

Examples of these kernels and power spectra are shown in Figure 5 for the simple-harmonic oscillator kernel which specifies the coefficients $a = S_0 \omega_0 Q$, $b = a/\gamma$, $c = \omega_0/(2Q)$ and $d = c\gamma$ in terms of a characteristic frequency, ω_0 , a quality factor, $Q > \frac{1}{2}$, and a normalization factor, S_0 , where $\gamma = \sqrt{4Q^2 - 1}$; see [Foreman-Mackey et al. \(2017\)](#) for the expressions when $Q \leq \frac{1}{2}$. The kernels become smoother and lower amplitude for longer integration times, while the power spectra are suppressed at high frequency, and show a series of zeros at integer multiples of the sampling frequency, which is a consequence of aliasing at these frequencies.

For the case of $b = 0$ and $c = 0$ (called the “real” term by [Foreman-Mackey et al. 2017](#)), the integrated kernel simplifies to

$$k_{\Delta}(\tau) = \frac{2a e^{-c\tau}}{(c\Delta)^2} \begin{cases} e^{-c\Delta} - 1 + c\Delta & \tau = 0 \\ [c(\Delta - \tau) - \sinh(c\Delta - c\tau)] e^{c\tau} + \cosh(c\Delta) - 1 & 0 < \tau \leq \Delta \\ \cosh(c\Delta) - 1 & \Delta < \tau \end{cases} . \quad \text{✎ (C38)}$$

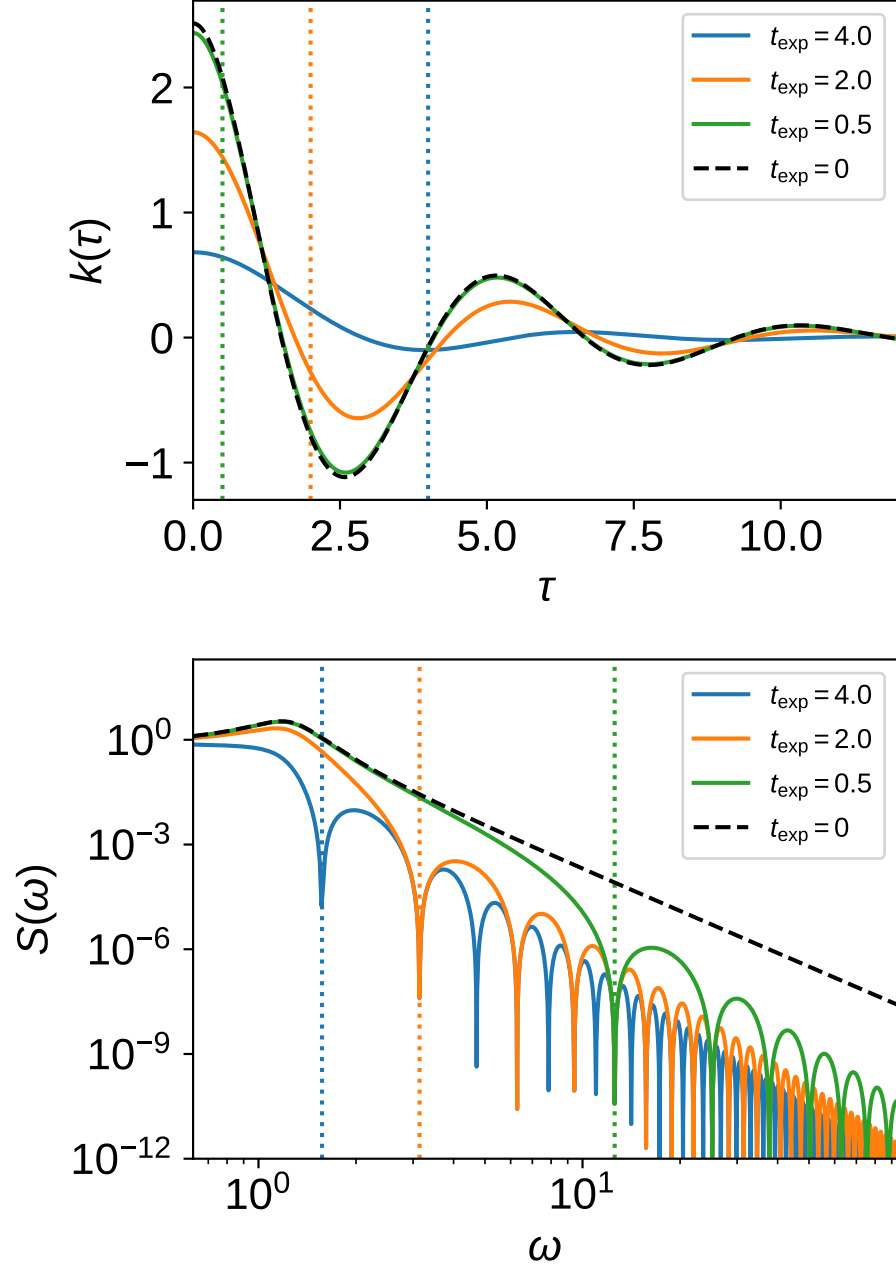


Figure 5. (Top) Time-integrated celerite kernels for $S_0 = 1, \omega_0 = 2\pi/5$, and $Q = 2$, with $\Delta = t_{\text{exp}} = (4, 2, \frac{1}{2})$ (blue, orange, green solid lines), as well as no time integration (dashed black line). Vertical dotted lines correspond to the sampling timescale. (Bottom) Corresponding power spectra. The vertical dotted lines indicate the sampling frequency in all three time-integrated cases. [☐](#)

REFERENCES

- | | |
|---|---|
| Abadi, M., et al. 2016, in OSDI, Vol. 16, 265–283 | Behnel, S., et al. 2011, Computing in Science Engineering, 13, 31 |
| Barragán, O., et al. 2019, MNRAS, 482, 1017 | Bingham, E., et al. 2018, Journal of Machine Learning Research |

- Carpenter, B., et al. 2015, ArXiv, 1509.07164
- Carpenter, B., et al. 2017, Journal of statistical software, 76
- Dillon, J. V., et al. 2017, arXiv e-prints, arXiv:1711.10604
- Eastman, J. D., et al. 2019, arXiv e-prints, arXiv:1907.09480
- Espinoza, N., et al. 2019, MNRAS, 490, 2262
- Foreman-Mackey, D. 2018, Research Notes of the American Astronomical Society, 2, 31
- Foreman-Mackey, D., et al. 2017, AJ, 154, 220
- Fulton, B. J., et al. 2018, PASP, 130, 044504
- Günther, M. N., & Daylan, T. 2019, `allesfitter`: Flexible star and exoplanet inference from photometry and radial velocity, Astrophysics Source Code Library, , , ascl:1903.003
- Kipping, D. M. 2013, MNRAS, 435, 2152
- Kreidberg, L. 2015, PASP, 127, 1161
- Luger, R., et al. 2019, AJ, 157, 64
- Mandel, K., & Agol, E. 2002, ApJL, 580, L171
- Maxted, P. F. L. 2016, A&A, 591, A111
- Mikkola, S. 1987, Celestial Mechanics, 40, 329
- Murray, C. D., & Correia, A. C. M. 2010, Keplerian Orbits and Dynamics of Exoplanets, ed. S. Seager, 15–23
- Nijenhuis, A. 1991, Celestial Mechanics and Dynamical Astronomy, 51, 319
- Paszke, A., et al. 2017, in NIPS-W
- Salvatier, J., et al. 2016, PeerJ Computer Science, 2, e55
- Theano Development Team. 2016, arXiv e-prints, abs/1605.02688.
<http://arxiv.org/abs/1605.02688>
- Trifonov, T. 2019, The Exo-Striker: Transit and radial velocity interactive fitting tool for orbital analysis and N-body simulations, , , ascl:1906.004