FLATIRON INSTITUTE

exoplanet: A toolkit for scalable inference for exoplanetary systems using transits, radial velocities, & astrometry

Daniel Foreman-Mackey,[1] Ian Czekala,[2, *] and Eric Agol[3, 4, †]

[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. In this paper, I derive methods for exploiting these gradient-based methods when fitting exoplanet data based on radial velocity, astrometry, and transits. Alongside the paper, I 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

   - Radial velocities and Kepler's equation

* NASA Hubble Fellowship Program Sagan Fellow
† Guggenheim Fellow

2. INTRODUCTION

Datasets are getting larger and more precise and this leads to an interest in more ambitious questions about these systems. The existing inference methods are no longer up to the task when the datasets are large and when the number of parameters is large. Many probabilistic inferences in astronomy have been limited by the existing tools that can't scale e.g. emcee. However, in other fields such as ML, methods have been developed that can scale to datasets too large to fit into memory and millions of parameters. The key ingredient in all of these methods is that it must be possible to efficiently evaluate the derivative of the model with respect to the physical parameters. This can often be intractable for applications astrophysics because the models generally include a physically motivated component that can't be trivially differentiated. However, we can take advantage of the substantial development that has been invested in automating this process.

This paper presents an example of how these tools and methods can be exploited in the specific astronomical application of fitting exoplanet datasets. We go through the derivation and implementation of the custom functions and their gradients that are required for gradient-based characterization of exoplanets.

These methods and their implementation are not an all-in-one package designed to do the fitting. Instead, it provided the framework needed to use these tools within pipelines.

3. 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 the fact that we are not talking *numerical* derivatives like finite difference methods, and there is no approximation being made.

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. This
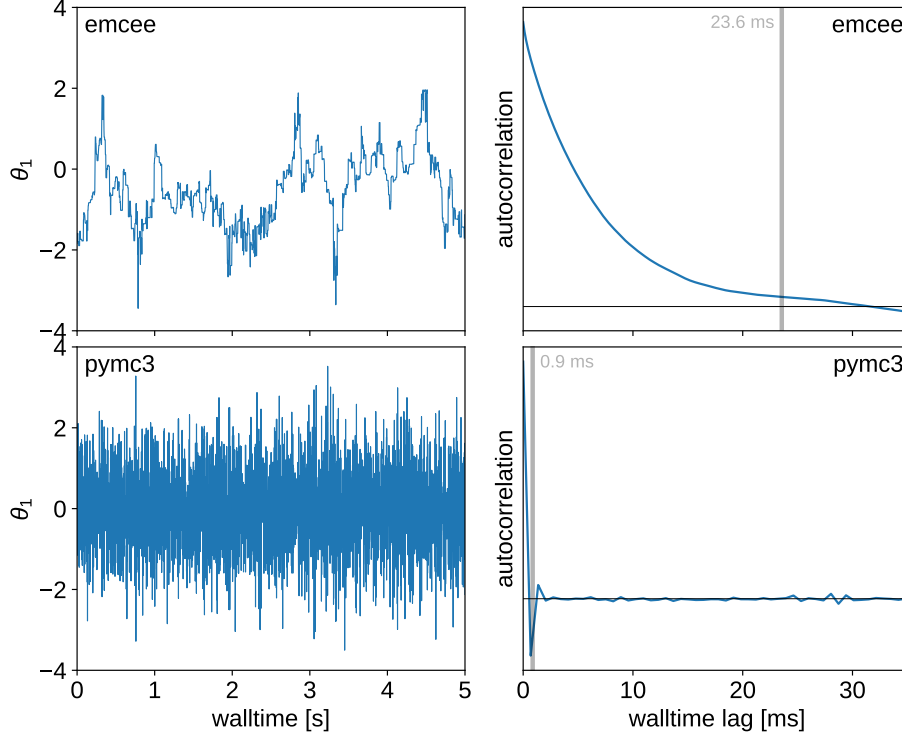
Figure 1. This is a figure.

realization was one of the key discoveries that launched the field of machine learning called "deep learning", where automatic differentiation was given the name "back-propagation" (CITE). Since then, there has been a substantial research base that has made these methods general and computationally efficient, and from this, many open source libraries have been released that ease the use of these methods (e.g. Tensor-Flow, 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 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.

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.

4. ORBITAL CONVENTIONS

We choose to follow a set of orbital conventions that respects as many of the established conventions while also hewing to sensible choices of right-hand rule.

The orbit is specified in the perifocal plane x, y, z, technically in the x, y plane, where $z = 0$. Then, the choice is to decide how to rotate this coordinate frame into the observed frame, which is specified by X, Y, Z.

Where does the $+Z$ axis point? Towards observer. X is north, Y is east.

What is the definition of the ascending node? Where the secondary is receding from the observer.

To do this, we consider the rotation matrices

$$
\boldsymbol{P}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{1}
$$

$$
\boldsymbol{P}_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}
$$

These rotation matrices result in a *clockwise* rotation of the axes, as defined using the right hand rule. 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, $\hat{x} \times \hat{y} = \hat{z}$, we know that 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 $\hat{z} \times \hat{x} = \hat{y}$.

One can see how we need to rotate by referencing Figure 2. The three rotations are (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. As one might notice, we break from some conventions (e.g., Murray & Correia 2010) in using a negative rotation for inclination, which is due to the change in how we label the ascending node. Applying these rotation matrices yields

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{P}_z(\Omega)\boldsymbol{P}_x(-i)\boldsymbol{P}_z(\omega) \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3}
$$

$$
\begin{aligned}
X &= r[\cos\Omega(\cos\omega\cos f - \sin\omega\sin f) - \sin\Omega\cos i(\sin\omega\cos f + \cos\omega\sin f)] \\
Y &= r[\sin\Omega(\cos\omega\cos f - \sin\omega\sin f) + \cos\Omega\cos i(\sin\omega\cos f + \cos\omega\sin f)] \\
Z &= -r\sin i(\sin\omega\cos f + \cos\omega\sin f).
\end{aligned} \tag{4}
$$

Using the sum angle identities to simplify the calculations, we find

$$
\begin{aligned}
X &= r(\cos\Omega\cos(\omega + f) - \sin(\Omega)\sin(\omega + f)\cos(i)) \\
Y &= r(\sin\Omega\cos(\omega + f) + \cos(\Omega)\sin(\omega + f)\cos(i)) \\
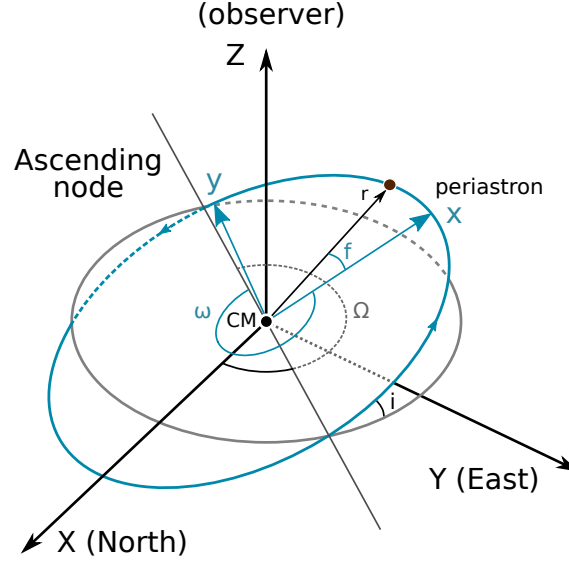Z &= -r\sin(\omega + f)\sin(i).
\end{aligned} \tag{5}
$$

Figure 2. The conventions used. ⟨/⟩

5. THE CUSTOM OPERATIONS PROVIDED BY EXOPLANET

5.1. *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

$$M \quad = \quad E - e \sin E \tag{6}$$

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

$$\frac{\mathrm{d}E}{\mathrm{d}e} \quad = \quad \frac{\sin E}{1 - e \cos E} \tag{7}$$

$$\frac{\mathrm{d}E}{\mathrm{d}M} \quad = \quad \frac{1}{1 - e \cos E} \quad . \tag{8}$$

Implementation details—We find that directly evaluating the true anomaly f when solving Kepler's equation is more efficient than evaluating it separately within Theano.

This means that we must also provide a method for evaluating the derivative of f with respect to e and M.

5.2. *Limb-darkened light curves*

5.3. *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?*.

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

$$M \quad = \quad E - e \sin E \quad . \tag{A1}$$

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 & Petigura 2017; 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

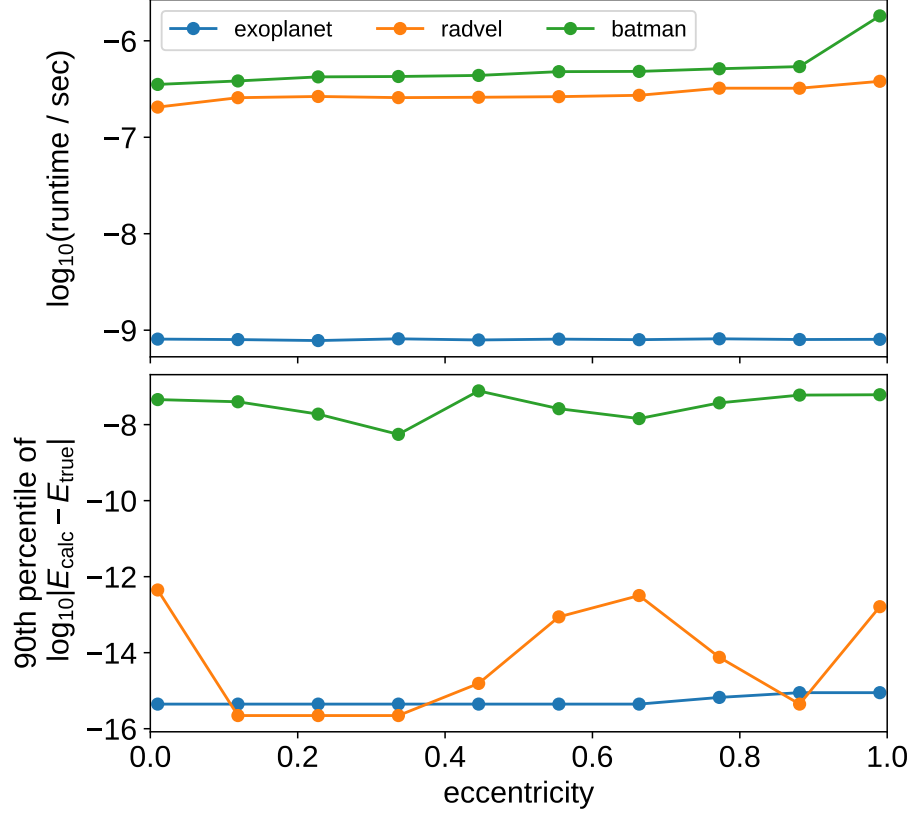iterative solver at the cost of longer run times, but the current value of 10^{-7} is currently hard coded[1].

Figure 3. The performance of the Kepler solver used by exoplanet compared to the solvers from the RadVel (Fulton & Petigura 2017; 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.

B. CONTACT POINTS

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

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

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

$$\boldsymbol{x}_2 \;=\; R_i\,R_\omega\,\boldsymbol{x}_0 \tag{B3}$$

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

[1] https://github.com/lkreidberg/batman/blob/70f2c0c609124bdf4f17041bf09d5426f3c93334/c__src/__rsky.c#L45

In this space, the planet will transit whenever

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

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

$$\hat{x_2}^2 + \hat{y_2}^2 = L^2 \tag{B6}$$

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

$$A\,x_2{}^2 + B\,x_2\,y_2 + C\,y_2{}^2 + D\,x_2 + E\,y_2 + F \;\;=\;\; 0 \tag{B7}$$

with

$$
\begin{aligned}
A &= \left(e^2\,\cos^2\omega - 1\right)\cos^2 i & \text{(B8)}\\
B &= 2\,e^2\,\cos i\,\sin\omega\,\cos\omega & \text{(B9)}\\
C &= e^2\,\sin^2\omega - 1 & \text{(B10)}\\
D &= 2\,a\,e\left(1 - e^2\right)\cos^2 i\,\cos\omega & \text{(B11)}\\
E &= 2\,a\,e\left(1 - e^2\right)\sin\omega\,\cos i & \text{(B12)}\\
F &= a^2\left(e^2 - 1\right)^2\cos^2 i & \text{(B13)}
\end{aligned}
$$

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

$$a_0 + a_1\,x_2 + a_2\,x_2{}^2 + a_3\,x_2{}^3 + a_4\,x_2{}^4 = 0 \tag{B14}$$

where

$$
\begin{aligned}
a_0 &= \left(CL^2 - EL + F\right)\left(CL^2 + EL + F\right) & \text{(B15)}\\
a_1 &= -2\left(BEL^2 - CDL^2 - DF\right) & \text{(B16)}\\
a_2 &= 2ACL^2 + 2AF - B^2L^2 - 2C^2L^2 - 2CF + D^2 + E^2 & \text{(B17)}\\
a_3 &= 2\left(AD + BE - CD\right) & \text{(B18)}\\
a_4 &= A^2 - 2AC + B^2 + C^2 & \text{(B19)}
\end{aligned}
$$

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

$$
\begin{aligned}
x_2 &= \pm L & \text{(B20)}\\
y_2 &= 0 \quad, & \text{(B21)}
\end{aligned}
$$

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

$$b_0 + b_1\,z_2 + b_2\,z_2{}^2 \;\;=\;\; 0 \tag{B22}$$

where

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

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

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

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).
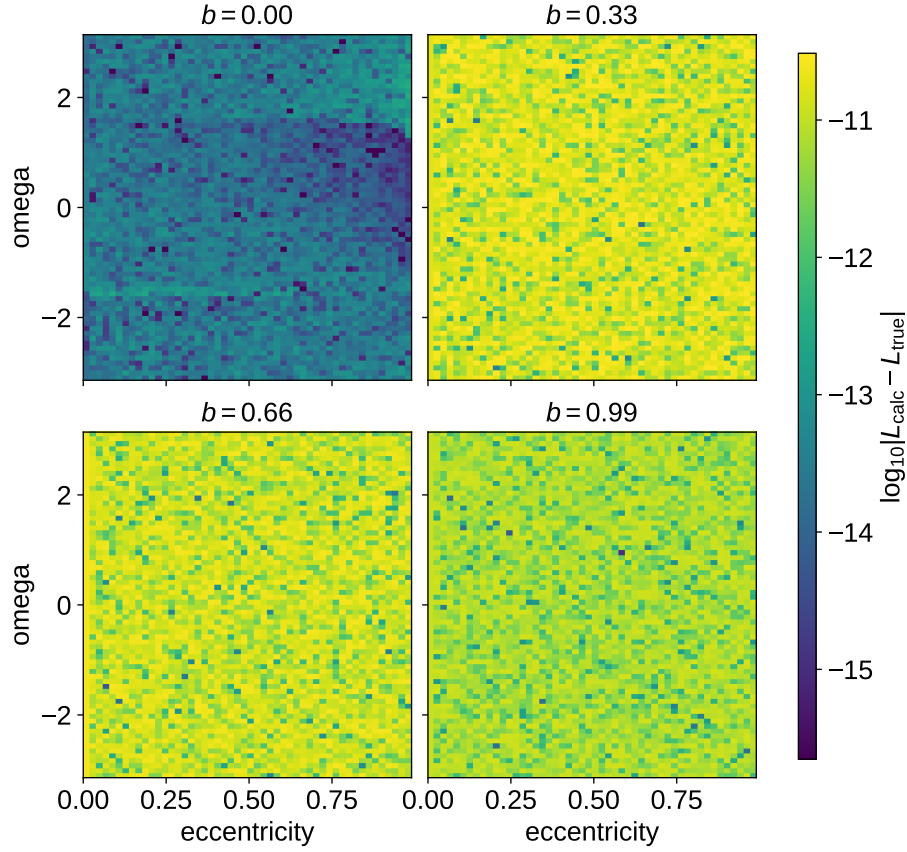

Figure 4. This is a figure.

C. EXPOSURE TIME INTEGRATION FOR CELERITE MODELS

Real term—For a real term:

$$k(\tau) = a\, \exp\left(-c\left(t_i - t_j\right)\right) \tag{C26}$$

the integrated kernel is

$$k_\Delta(\tau) = \frac{2}{(c\,\Delta)^2} \left[\cosh(c\,\Delta) - 1\right] k(\tau) \tag{C27}$$

where Δ is the integration time.

Complex term—The result is somewhat more complicated for a complex term:

$$k(\tau) = (a + i\,b)\,\exp\left(-(c + i\,d)\,(t_i - t_j)\right) \tag{C28}$$

In this case, the integrated kernel is

$$k_\Delta(\tau) = (A + i\,B)\,k(\tau) \tag{C29}$$

where

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

and

$$B = \frac{C_2\,[\cosh(c\,\Delta)\,\cos(d\,\Delta) - 1] + C_1\,\sinh(c\,\Delta)\,\sin(d\,\Delta)}{\Delta^2\,(c^2 + d^2)^2} \qquad \text{✐}\,(\text{C31})$$

where

$$C_1 = 2\,(a\,c^2 - a\,d^2 + 2\,b\,c\,d) \quad \text{and} \quad C_2 = 2\,(b\,c^2 - b\,d^2 - 2\,a\,c\,d) \tag{C32}$$

REFERENCES

Abadi, M., et al. 2016, in OSDI, Vol. 16, 265–283

Behnel, S., et al. 2011, Computing in Science Engineering, 13, 31

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

Foreman-Mackey, D. 2018, Research Notes of the American Astronomical Society, 2, 31

Foreman-Mackey, D., et al. 2017, AJ, 154, 220

Fulton, B., & Petigura, E. 2017, RadVel: Radial Velocity Fitting Toolkit, , , doi:10.5281/zenodo.580821. https://doi.org/10.5281/zenodo.580821

Fulton, B. J., et al. 2018, PASP, 130, 044504

Kreidberg, L. 2015, PASP, 127, 1161

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