

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

DANIEL FOREMAN-MACKEY<sup>1</sup>

<sup>1</sup>*Center for Computational Astrophysics, Flatiron Institute, New York, NY*

## 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
- Astrometry
- Transits

### 5. Implementation details and benchmarks

### 6. Examples

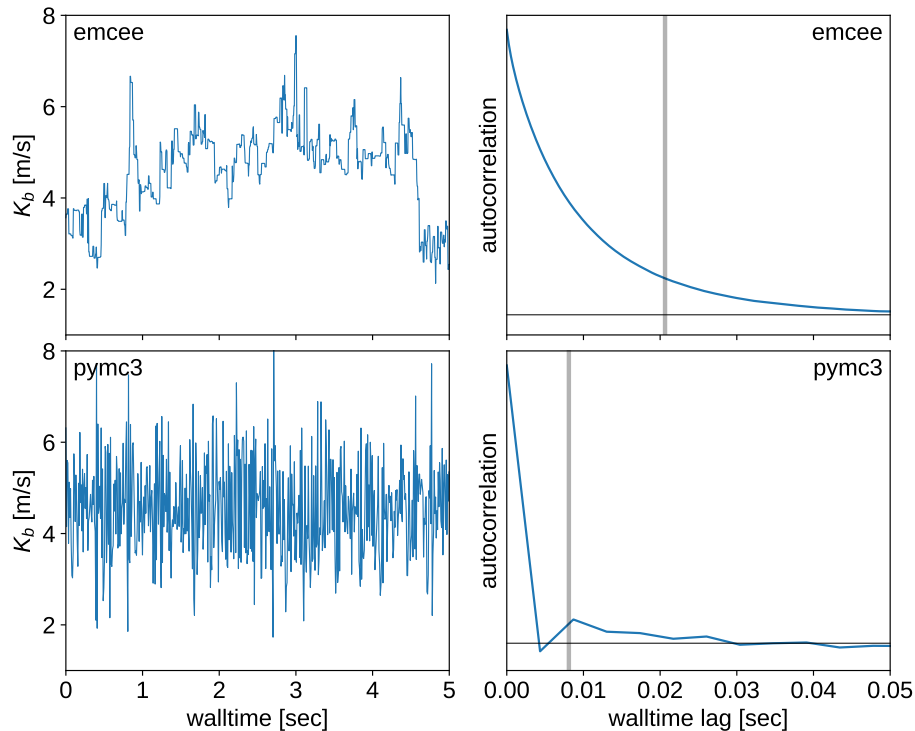
### 7. Discussion

## 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.



**Figure 1.** This is a figure. 

### 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 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. 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 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.

At present, there are many model-building libraries available in every programming language, and it is not clear which should be preferred. For this paper, we focus on implementations within TensorFlow (Abadi et al. 2016) because it is accessible using Python, the most widely used programming language in astronomy (CITE?) and because the code is well-supported and stable. This being said, the basic ideas behind each library are similar and the core implementation in C++ can be minimally modified to support other frameworks.

## 4. THE CUSTOM OPERATIONS PROVIDED BY EXOPLANET

### 4.1. 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

$$M = E - e \sin E \quad (1)$$

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

$$\frac{dE}{de} = \frac{\sin E}{1 - e \cos E} \quad (2)$$

$$\frac{dE}{dM} = \frac{1}{1 - e \cos E} \quad (3)$$

*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  $M$ .

#### 4.2. Limb-darkened light curves

#### 4.3. Scalable Gaussian Processes for time series

### 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

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

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:

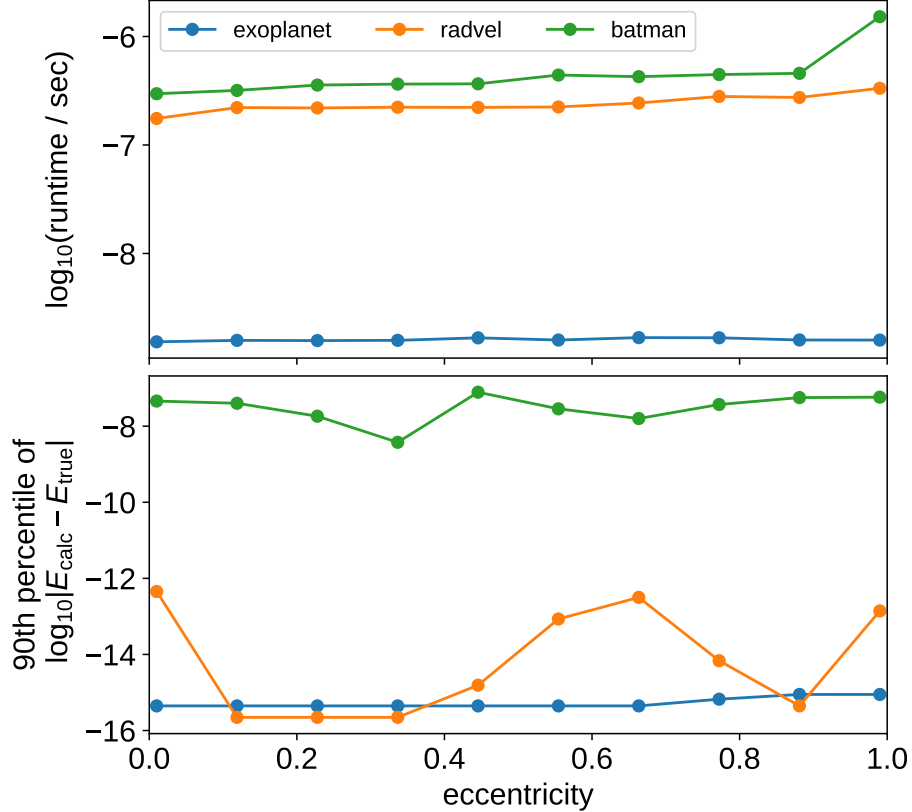
1. Initial estimates are selected for the parameters by partitioning the parameter space into 4 regions and applying a set of heuristics specific to each part of parameter space.
2. This estimate is refined using either a single iteration of Halley’s second order root finding algorithm (with the sine 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 5000 true eccentric anomalies  $E_{\text{true}}$  uniformly distributed between 0 and  $2\pi$ . 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  $E_{\text{calc}}$ . The top panel of Figure 2 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 2 shows the 90th percentile of the distribution of absolute errors when comparing  $E_{\text{true}}$  and  $E_{\text{calc}}$  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<sup>1</sup>.



**Figure 2.** 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 5000 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](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

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

<sup>1</sup> [https://github.com/lkreidberg/batman/blob/70f2c0c609124bdf4f17041bf09d5426f3c93334/c\\_src/\\_rsky.c#L45](https://github.com/lkreidberg/batman/blob/70f2c0c609124bdf4f17041bf09d5426f3c93334/c_src/_rsky.c#L45)

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  $z_2$

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

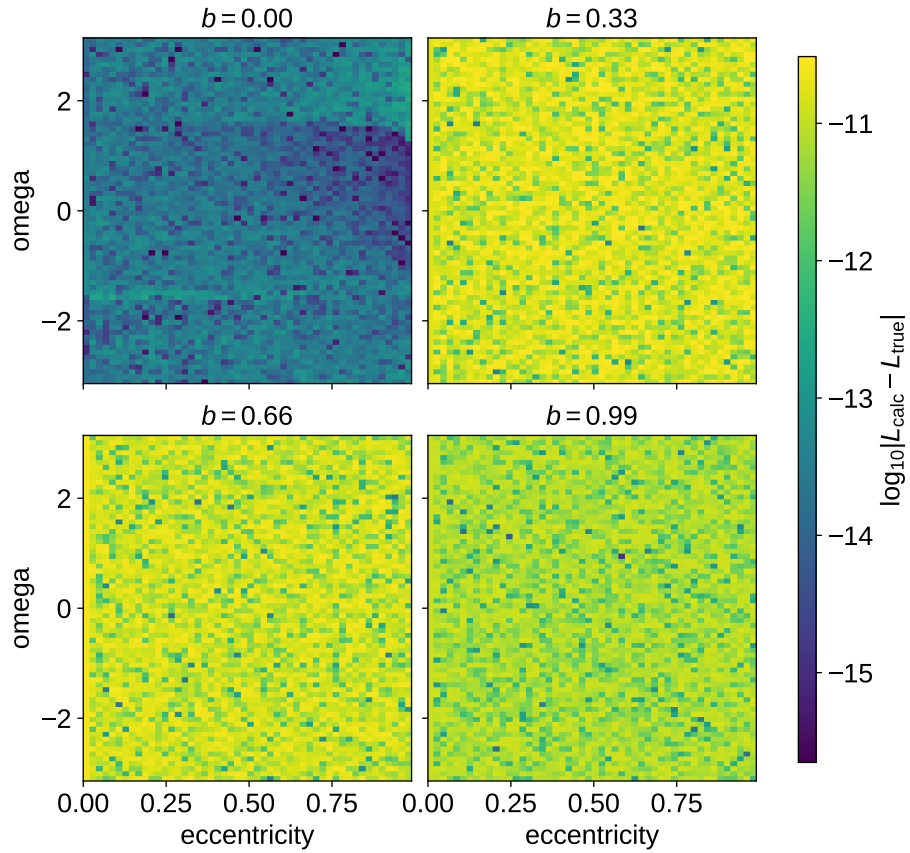
where


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

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

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

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 3.** This is a figure. 



## REFERENCES

- Abadi, M., et al. 2016, in OSDI, Vol. 16, 265–283
- Behnel, S., et al. 2011, Computing in Science Engineering, 13, 31
- 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
- Nijenhuis, A. 1991, Celestial Mechanics and Dynamical Astronomy, 51, 319
- Theano Development Team. 2016, arXiv e-prints, abs/1605.02688.  
<http://arxiv.org/abs/1605.02688>