



# Physics of Eclipsing Binaries. V. General Framework for Solving the Inverse Problem

Kyle E. Conroy<sup>1</sup> , Angela Kochoska<sup>1</sup> , Daniel Hey<sup>2,3</sup> , Herbert Pablo<sup>4</sup> , Kelly M. Hambleton<sup>1</sup> , David Jones<sup>5,6</sup> ,  
Joseph Giammarco<sup>7</sup> , Michael Abdul-Masih<sup>8</sup> , and Andrej Prša<sup>1</sup>

<sup>1</sup> Department of Astrophysics and Planetary Science, Villanova University, 800 East Lancaster Avenue, Villanova, PA 19085, USA; [kyle.conroy@villanova.edu](mailto:kyle.conroy@villanova.edu)

<sup>2</sup> School of Physics, Sydney Institute for Astronomy (SIfA) The University of Sydney, NSW 2006, Australia

<sup>3</sup> Stellar Astrophysics Centre, Department of Physics and Astronomy Aarhus University, DK-8000 Aarhus C, Denmark

<sup>4</sup> American Association of Variable Star Observers, 49 Bay State Road, Cambridge, MA 02138, USA

<sup>5</sup> Instituto de Astrofísica de Canarias, E-38205 La Laguna, Tenerife, Spain

<sup>6</sup> Departamento de Astrofísica, Universidad de La Laguna, E-38206 La Laguna, Tenerife, Spain

<sup>7</sup> Eastern University, Dept. of Astronomy and Physics, 1300 Eagle Rd, St. Davids, PA 19087, USA

<sup>8</sup> Institute of Astronomy, KU Leuven, Celestijnenlaan 200 D, B-3001, Leuven, Belgium

Received 2020 June 30; revised 2020 August 16; accepted 2020 September 1; published 2020 October 9

## Abstract

PHOEBE 2 is a Python package for modeling the observables of eclipsing star systems, but until now it has focused entirely on the forward model—that is, generating a synthetic model given fixed values of a large number of parameters describing the system and the observations. The inverse problem, obtaining orbital and stellar parameters given observational data, is more complicated and computationally expensive as it requires generating a large set of forward models to determine which set of parameters and uncertainties best represents the available observational data. The process of determining the best solution and also of obtaining reliable and robust uncertainties on those parameters often requires the use of multiple algorithms, including both optimizers and samplers. Furthermore, the forward model of PHOEBE has been designed to be as physically robust as possible, but it is computationally expensive compared to other codes. It is useful, therefore, to use whichever code is most efficient given the reasonable assumptions for a specific system, but learning the intricacies of multiple codes presents a barrier to doing this in practice. Here we present release 2.3 of PHOEBE (publicly available from <http://phoebe-project.org>), which introduces a general framework for defining and handling distributions on parameters and utilizing multiple different estimation, optimization, and sampling algorithms. The presented framework supports multiple forward models, including the robust model built into PHOEBE itself.

*Unified Astronomy Thesaurus concepts:* [Eclipsing binary stars \(444\)](#)

## 1. Introduction

Eclipsing binary systems provide the ability to determine absolute fundamental parameters of stellar components, calibrate stellar evolution models, measure the orientation and dynamical changes in orbits, and derive distances without the need for calibration. For benchmark-grade eclipsing binaries, a precision in fundamental stellar parameters (masses and radii) of a few percent can be obtained (Torres et al. 2010). All these scientific goals can only be achieved through solving the inverse problem: models are created to simulate synthetic observables given a set of input parameters, and the input parameters can then be varied to determine which values result in synthetic observables that best represent the observations. The parameter space, however, is complex and often not smooth, making this task expensive and time consuming.

PHOEBE is one of many modeling codes that operate by discretizing the gravitationally distorted surface of each stellar component, populating the discretized grid with local quantities from atmosphere tables, and integrating over visible elements at any given time in the orbit to synthesize model observables. Until now, all releases since the 2.0 release (Prša et al. 2016; Horvat et al. 2018; Jones et al. 2020) have been focused on adding advanced physics and precision to the forward model and leaving the inverse problem to the user.

Because of the complexity of the parameter space, the different combinations of observations available, and the wide range of system morphologies and relevant physics effects, each individual system often requires a manual treatment in

order to obtain the most accurate and precise model and uncertainties possible. Furthermore, it is often necessary to utilize multiple algorithms in order to increase the chance of determining the optimal solution and produce reliable and realistic uncertainties on the modeled parameters. Having to “wrap” several algorithms, each with its own interface, around the forward model can present a steep learning curve and effort overhead for solving these systems. To address this, PHOEBE now includes a common interface to interact with a variety of algorithms, some off-the-shelf and others custom built, that are useful for solving the inverse problem.

To complicate matters, the available forward-model codes each have their own interfaces and often different parameterizations, making it even more difficult to compare the forward models or choose the code most appropriate for a given scenario. PHOEBE 2 has been designed and developed with precision, robustness, and flexibility as the main priorities, but this comes at the cost of computational efficiency. In many cases, other codes may be just as capable of modeling a given system but at a fraction of the time cost. Learning the intricacies of each of these codes presents yet another learning curve and overhead. PHOEBE now incorporates several other publicly available codes as the forward model, converting parameterization and units for both inputs and outputs as necessary. This brings several large benefits, including the abilities to easily compare the synthetic models created by different codes, choose the code most appropriate and efficient for a specific system, and transition to more expensive but precise models throughout solving the inverse problem.

Here we outline a general framework for solving the inverse problem with PHOEBE. We divide the available solver algorithms incorporated in this PHOEBE release into three categories: “estimators,” “optimizers,” and “samplers.” Section 2 details the “estimators” now available in PHOEBE that estimate system parameters from observations alone, without the need for a forward model. These estimators are designed to cheaply determine the parameter space of a particular system given any of the data available. Section 3 then discusses the merit function from the available forward models. This merit function is then used for both “optimizers” (Section 4) and “samplers” (Section 5) to find the optimal solution and determine robust posteriors (Section 6). Section 7 then presents the new user interface introduced in this release, allowing easy access to most of the inverse-problem functionality from either a web browser or a dedicated desktop client. See the Appendix for descriptions of the symbols used for system parameters throughout the paper.

This paper is accompanied by the 2.3 release of PHOEBE, available from <http://phoebe-project.org> and <https://github.com/phoebe-project/phoebe2>. Scripts to reproduce all figures in this paper are available at <http://phoebe-project.org/publications/2020Conroy+>.

## 2. Estimators

It is often useful and computationally prudent to estimate the values of as many parameters as possible without having to compute multiple physical models. The 2.3 release of PHOEBE incorporates several algorithms that act directly on the observations themselves, with the goal of providing fast initial solutions for a subset of the relevant system parameters without significant user supervision, including the following:

1. RV periodogram, LC periodogram: computes the periodogram for light curves or radial-velocity curves, respectively, and proposes an orbital period,  $P_{\text{orb}}$ ;
2. RV geometry: fits a simple Keplerian orbit to propose values for  $t_{0,\text{supconj}}$ ,  $e$ ,  $\omega_0$ ,  $v_\gamma$ , and either  $q$  and the projected orbital semimajor axis,  $a_{\text{orb}} \sin i$ , or the projected per-component semimajor axes,  $a_{\text{comp},i} \sin i$ ;
3. LC geometry: estimates phases of eclipse minima, ingress, and egress and uses that information to propose values for  $t_{0,\text{supconj}}$ ,  $e$ , and  $\omega_0$ ; and
4. EBAI: uses a trained artificial neural network on the input light curves to propose values for  $t_{0,\text{supconj}}$ ,  $T_{\text{eff},2}/T_{\text{eff},1}$ ,  $(R_{\text{equiv},1} + R_{\text{equiv},2})/a_{\text{orb}}$ ,  $e \sin \omega_0$ ,  $e \cos \omega_0$ , and  $i$ .

For all estimators that work in phase space (as opposed to time space), the input data can optionally be binned to maintain the efficiency of the estimators on large data sets.

### 2.1. Periodograms

The most common observational data sets used when modeling eclipsing star systems are light curves and radial-velocity curves. For the majority of eclipsing systems (those that do not exhibit significant time-dependent changes), these data can be phase folded on the orbital period of the system. If the period is well known and the orbital parameters do not change between successive orbits, then the system can be sampled in phase space instead.

PHOEBE 2.3 contains wrappers around two common periodograms—box least squares (BLS, for light curves) and Lomb Scargle (for light curves or radial velocities)—as

implemented by `astropy.timeseries`<sup>9</sup> (Astropy Collaboration et al. 2013). The wrapper takes any number of light curves or radial velocities as input, in addition to several of the advanced options from `astropy`, including automatic or manual sampling in period/frequency, the option to change the objective function for BLS, and the ability to set the proposed eclipse durations for BLS. If running on multiple light curves, each light curve is normalized by dividing by either the median or maximum flux value before sending to the periodogram algorithm. Radial velocities across all requested data sets are combined and then normalized by the absolute maximum value for the primary and secondary star independently, with the secondary then mirrored. After running the wrapper, the periodogram itself is returned and can be plotted, and the peak period is proposed for adoption. Especially for near-circular systems with similar eclipses, these algorithms often find a strong signal at aliases of the true orbital period, so an easy interface is provided for adopting any factor of the proposed peak period.

### 2.2. Radial-velocity Geometry

Analytical radial velocities (RVs) are used as an estimator for the eccentricity ( $e$ ), argument of periastron ( $\omega$ ), systemic velocity ( $v_\gamma$ ), and time of superior conjunction ( $t_{0,\text{supconj}}$ ), as well as the mass ratio ( $q$ ) and projected orbital semimajor axis ( $a_{\text{orb}} \sin i$ ) in the case where RVs are provided for both components or the single-component projected semimajor axis ( $a_{\text{comp},i} \sin i$ ) in the case where RVs are only provided for either one of the stellar components. All estimates are based on the per-component radial-velocity equations as a function of the true anomaly ( $\vartheta$ ; Prša 2018):

$$\begin{aligned} \text{RV}_1(\vartheta) &= \frac{2\pi a_1 \sin i}{P_{\text{orb}} \sqrt{1-e^2}} [e \cos \omega + \cos(\omega + \vartheta)] + v_\gamma \\ &= \frac{2\pi q a \sin i}{P_{\text{orb}}(1+q) \sqrt{1-e^2}} [e \cos \omega + \cos(\omega + \vartheta)] + v_\gamma, \quad (1) \\ \text{RV}_2(\vartheta) &= -\frac{2\pi a_2 \sin i}{P_{\text{orb}} \sqrt{1-e^2}} [e \cos \omega + \cos(\omega + \vartheta)] + v_\gamma \\ &= -\frac{2\pi a \sin i}{P_{\text{orb}}(1+q) \sqrt{1-e^2}} [e \cos \omega + \cos(\omega + \vartheta)] + v_\gamma. \quad (2) \end{aligned}$$

We assume that the period ( $P_{\text{orb}}$ ) is known and use the phase-folded radial-velocity curves as input. The data are smoothed with a low-pass Savitzky–Golay filter to remove high-frequency noise.

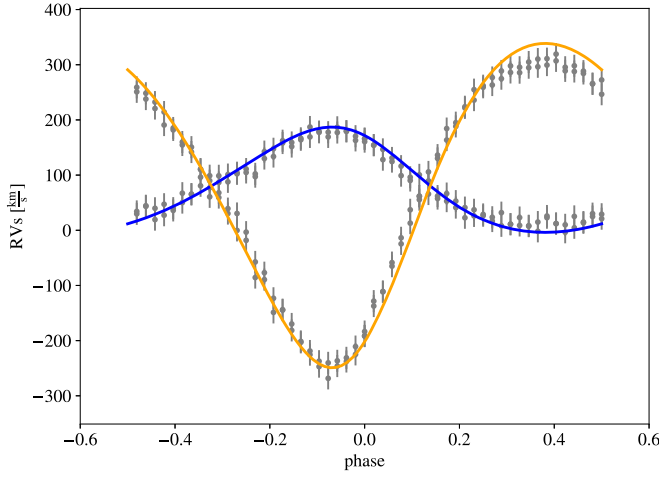
The first two parameters that are estimated in parallel are the mass ratio and the systemic velocity. From Equation (1) we can derive analytical expressions that hold at any point in the orbit for both parameters:

$$v_\gamma = \frac{\text{RV}_1(\vartheta) + q \text{RV}_2(\vartheta)}{1+q}, \quad (3)$$

$$q = \frac{\text{RV}_1(\vartheta) - v_\gamma}{-\text{RV}_2(\vartheta) + v_\gamma}. \quad (4)$$

As we can see,  $v_\gamma$  and  $q$  appear in both expressions, which means we cannot uniquely determine them independently of

<sup>9</sup> Requires `astropy` v3.2 or later.



**Figure 1.** Analytic Keplerian radial-velocity model fitted to a synthetic set of observations with estimated values for eccentricity ( $e$ ), argument of periastron ( $\omega_0$ ), mass ratio ( $q$ ), projected semimajor axis ( $a_{\text{orb}} \sin i$ ), and the time of superior conjunction ( $t_{0,\text{supconj}}$ ) from only the phased radial-velocity observations.

each other. For that purpose we compute them iteratively, starting from a crude estimate of  $q$  as the ratio of the primary and secondary RVs. To maximize the efficiency, we only estimate  $q$  and  $v_\gamma$  in the RV points that correspond to the maximum primary RV. If only one RV is available,  $q$  cannot be reliably estimated, and in this case only  $v_\gamma$  is estimated as the midpoint of the available RV.

With  $q$  and  $v_\gamma$  estimated, we can compute the projected semimajor axis,  $a \sin i$ . We first estimate the semiamplitudes from the available observations in each RV:

$$K_i = 0.5[\max(\text{RV}_i - v_\gamma) - \min(\text{RV}_i - v_\gamma)] \quad (5)$$

where  $i = 1, 2$  refers to the primary and secondary component, respectively. The corresponding projected semimajor axes are then

$$a_i \sin i = \frac{K_i P_{\text{orb}}}{2\pi} \sqrt{1 - e^2}. \quad (6)$$

Initially, we assume  $e = 0$ . If both RVs are available,  $a \sin i = a_1 \sin i + a_2 \sin i$ .

Once  $q$ ,  $v_\gamma$ , and  $a \sin i$  are estimated, we fix them in the analytical radial velocities and fit for eccentricity and argument of periastron with a least-squares algorithm. We compute several solutions by iterating over a small grid of starting points with the combinations  $e_0 = [0, 0.4]$  and  $\omega_0 = [0, \pi/2, \pi]$ , which ensures the entire feasible parameter space is covered, and we iterate over the least-squares solution until the values converge within a tolerance or the maximum number of iterations is reached. After each iteration, the value of  $a \sin i$  is recomputed with the updated eccentricity  $e$ , and the solution is used as an initial point in the next iteration.

Finally, the phase of superior conjunction is estimated as the phase at which  $v = v_\gamma$  and then converted to the time of superior conjunction,  $t_{0,\text{supconj}}$ .

An example of the performance of the estimator is given in Figure 1.

### 2.3. Light-curve Geometry

PHOEBE 2.3 introduces an estimator based on fitting a two-Gaussian model to a phased light curve to evaluate the eccentricity and argument of periastron, as the two best-constrained parameters from light-curve geometry. For a detailed description of the model, refer to Mowlavi et al. (2017). Here, we will focus on its implementation in PHOEBE.

The two-Gaussian model is a fast analytical model that fits a composite function of a constant cosine term, two Gaussians, or any combination of these to the observed data, resulting in seven different models. The simplest model consists only of a constant term ( $C$ ), while the most complex one consists of a constant, two Gaussian functions, and a cosine term. Each Gaussian is characterized by its central position ( $\mu$ ), amplitude ( $A$ ), and rms width ( $\sigma$ ). The cosine term is only characterized by its amplitude ( $A_{\text{ell}}$ ) as the model assumes its period is equal to half the light-curve period (0.5 in the case of a phase-folded light curve), and its trough coincides with the position of either the primary or secondary eclipse.

The two-Gaussian model is sensitive to the initial values of the model parameters. Thus, to ensure its convergence, PHOEBE first estimates the eclipse positions, widths, and depths using a simple algorithm that searches for the minimum of the light curve and isolates data in its vicinity that cross the median flux in phase ( $\varphi$ ) space. All seven models are then fitted with a least-squares optimizer, and the best fit is chosen as the one with the highest Bayesian information criterion (BIC; Schwarz 1978). We determine the eclipse parameters following the prescriptions in Mowlavi et al. (2017), where the positions of the eclipses coincide with the central positions of the Gaussians ( $\varphi_i = \mu_i$ ), the widths of the eclipses are a factor of the Gaussian rms widths ( $w_i = 5.6\sigma_i$ ), and the depths are computed as the constant term minus the flux at eclipse positions ( $d_i = C - \text{flux}(\varphi_i)$ ). Figure 2 shows an example case with the best fit model and the resulting eclipse phases.

The estimator computes the time of superior conjunction, eccentricity, and argument of periastron as follows:

1. Time of superior conjunction:

$$t_{0,\text{supconj}} = t_{0,\text{supconj,orig}} + \varphi_1 P_{\text{orb}}. \quad (7)$$

2. Eccentricity:

$$e = \left[ \sin^2\left(\frac{\psi - \pi}{2}\right) + \left(\frac{w_2 - w_1}{w_2 + w_1}\right)^2 \cos^2\left(\frac{\psi - \pi}{2}\right) \right]^{1/2}. \quad (8)$$

3. Argument of periastron:

$$\omega_1 = \arcsin\left(\frac{1}{e} \frac{w_2 - w_1}{w_2 + w_1}\right) \quad (9)$$

$$\omega_2 = \arccos\left(\frac{\sqrt{1 - e^2}}{2e \tan(\psi - \pi)}\right) \quad (10)$$

$$\omega = \begin{cases} \omega_2, & \text{if } \omega_1 > 0 \\ 2\pi - \omega_2, & \text{if } \omega_1 < 0 \end{cases} \quad (11)$$

where  $\psi = \pi + 2 \arctan \frac{e \cos \omega}{\sqrt{1 - e^2}}$  and is computed iteratively through solving  $2\pi \Delta\Phi = \psi - \sin \psi$ , where  $\Delta\Phi$  is the separation between the two eclipses in phase space.

In addition, the phases of ingress and egress and eclipse widths and depths are computed and exposed to the user. These are then used to also propose an optional phase mask based on

the eclipse edges and a padding of 30% of the eclipse width (see Section 3.4.1 and Figure 5).

#### 2.4. Artificial Neural Network (EBAI)

EBAI is a back-propagating artificial neural network specifically implemented to recover fundamental parameters from phased light curves of eclipsing binary systems (Prša et al. 2008, 2019). For the implementation within PHOEBE 2.3, we retrain the network on detached systems using the two-Gaussian model (see Section 2.3), using 201 input units, 40 hidden units, 5 output units, learning rate parameter  $\eta = 0.2$ , 33,235 exemplars, and  $4 \times 10^6$  iterations.

To use EBAI as an estimator, the user can pass any number of light-curve data sets, which are normalized and fitted with a two-Gaussian model, automatically introducing a phase shift if necessary to ensure the eclipse is at phase zero as required by the training set. This analytic representation is then normalized to unity, sampled at the same 201 phase points as the training set, and passed to the two matrix transformations determined from the EBAI training (Figure 3). The results received directly from EBAI are then converted as necessary to the proposed values for  $T_{\text{eff},2}/T_{\text{eff},1}$ ,  $(R_{\text{equiv},1} + R_{\text{equiv},2})/a_{\text{orb}}$ ,  $e \sin \omega_0$ ,  $e \cos \omega_0$ , and  $i$ , in addition to  $t_{0,\text{supconj}}$  computed from the applied phase shift.

### 3. The Merit Function

Any optimization (see Section 4) or sampling (Section 5) that requires the forward model to be computed requires a definition of the merit function to compare the synthetic model to the observations. The components of the merit function (Inprobability) and terms that have been adopted within PHOEBE are explained below:

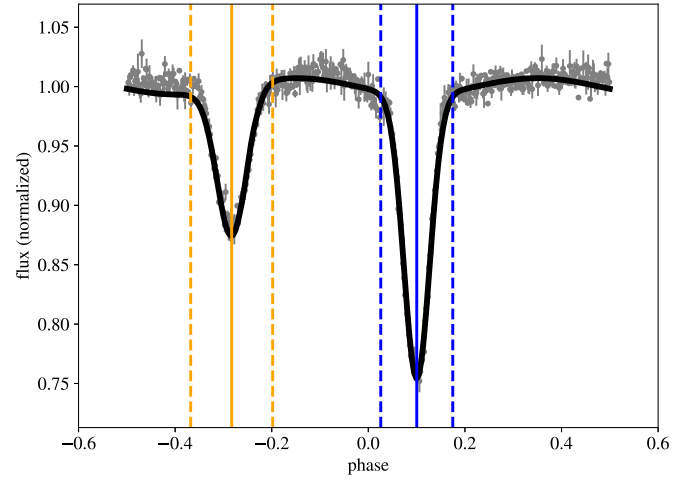
1. **Inpriors:** the log priors term is defined as the log probability of drawing the current face values,  $p$ , from the provided prior probability distribution functions,  $\pi$ . When priors are not provided or applicable, this term will be constant and so will not contribute to the optimization and sampling. Priors are discussed in more detail in Section 3.2:

$$\text{Inpriors} = \sum_{\text{priors}} \ln(P(p|\Pi)). \quad (12)$$

2. **Residuals:** observations ( $y_o$ ) – synthetic model ( $y_m$ ), whether those are fluxes or radial velocities, after any phase masking (see Section 3.4.1). By default, the synthetic model is computed at the exact times of the (masked) observations. However, the user can also compute the model at a list of custom times. In that case, the synthetic model is linearly interpolated to the times (or phases) of the observations, and then residuals are determined. It is the responsibility of the user to ensure the model is sufficiently sampled so that linear interpolation does not cause any issues. See Section 3.4 for details about the forward model itself, times of the synthetic model, phase masking, and interpolation:

$$\text{residuals} = y_o(t_{\text{masked}}) - y_m(t_{\text{masked}}). \quad (13)$$

3. **chi2 ( $\chi^2$ ):**  $\chi^2$  is defined here to be the sum of squares of the residuals of all data points across all enabled “data sets” over the squares of the provided per-point uncertainties (if provided, although note that some algorithms require uncertainties to be provided), plus an additional term to



**Figure 2.** Best of seven trial two-Gaussian models used to determine the phases of eclipse minima, ingress, and egress (blue and orange vertical lines represent the primary and secondary eclipses, respectively), as well as the input binned synthetic observations. These are then used to estimate values for eccentricity ( $e$ ), argument of periastron ( $\omega_0$ ), and the time of superior conjunction ( $t_{0,\text{supconj}}$ ).

handle uncertainty underestimation from the provided observation uncertainties,  $\sigma_o$ . This uncertainty underestimation term ( $\sigma_{\text{inf}}$ ) can be optimized or sampled in the case where the provided uncertainties are believed to be underestimated. By default,  $\sigma_{\text{inf}} = -\text{inf}$ , which cancels the additional term in the denominator. See Section 3.5 for more about this “noise nuisance” parameter:

$$\chi^2 = \sum_{\text{data sets}} \frac{(y_o - y_m)^2}{\sigma^2} + \ln(\sigma^2), \quad (14)$$

where

$$\sigma^2 = \sigma_o^2 + y_m^2 e^{(2\sigma_{\text{inf}})}. \quad (15)$$

4. **Inlikelihood:** the log likelihood is, by definition,

$$\text{Inlikelihood} = -0.5\chi^2. \quad (16)$$

5. **Inprobability:** the log probability is the default merit function used within PHOEBE (note that, technically, the negative log probability is used for optimizers that minimize a merit function as the optimal model is found by maximizing the log probability). When priors are not provided or applicable, this essentially becomes analogous to  $\chi^2$ . The log probability is defined as the sum of the log priors and the log likelihood:

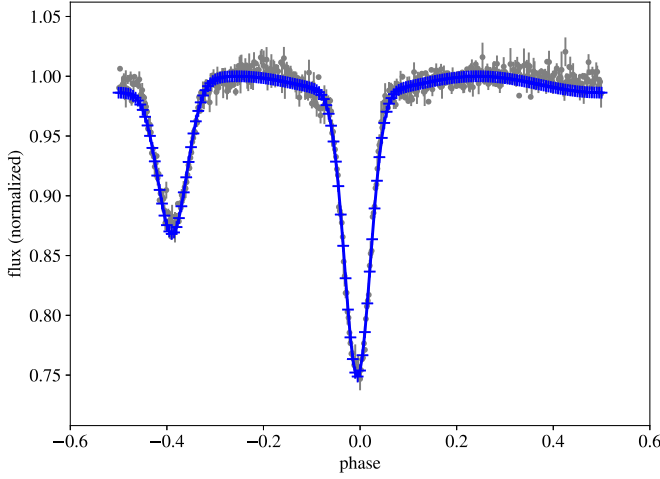
$$\text{Inprobability} = \text{Inpriors} + \text{Inlikelihood}. \quad (17)$$

6. **Inposteriors:** the log posteriors, similar to the log priors, describe the probability of drawing the current face values,  $p$ , given the resulting posterior probability distribution functions. These posteriors (see Section 6) are often determined from sampling the log probability with an algorithm such as Markov Chain Monte Carlo (MCMC; see Section 5.1):

$$\text{Inposteriors} = \sum_{\text{posteriors}} \ln(P(p|\text{posterior pdf})). \quad (18)$$

Unless overridden with a custom merit function, all optimizers and samplers within PHOEBE use the log probability as defined





**Figure 3.** Analytic two-Gaussian representation of the automatically binned phased light curve sent to EBAI to estimate  $t_{0,\text{supconj}}$ ,  $T_{\text{eff},\text{ratio}}$ ,  $R_{\text{equiv},\text{fractional sum}}$ ,  $e \sin \omega_0$ ,  $e \cos \omega_0$ , and  $i$ .

above. Additionally, if the user would like to implement their own optimizer or sampler outside of PHOEBE, all of the quantities above are exposed to the user for any given forward model.

### 3.1. Parameterization

Depending on the available observations (single light curve, multiple light curves, single-lined radial velocities, double-lined radial velocities, and so on), previously known information, and the orthogonality (or lack thereof) of the parameter space, it is advantageous to be able to choose a specific parameterization. For example, due to the parameterization of the Roche model, PHOEBE 1 (based on Wilson–Devinney) parameterizes the orbit by the orbital period ( $P_{\text{orb}}$ ), mass ratio ( $q$ ), and semimajor axis ( $a_{\text{orb}}$ ; along with eccentricity  $e$  and orientation parameters such as inclination and argument of periastron). The individual stellar masses ( $M_1$  and  $M_2$ ) can be computed from these parameters and Kepler’s third law, but cannot directly be fixed or adjusted.

PHOEBE now provides a flexible framework for scenarios like these. For example, by default, PHOEBE adopts a parameterization similar to the legacy version but also exposes read-only parameters for the masses that are derived by Kepler’s third law:

$$M_1 = \frac{4\pi a^3}{GP^2} \frac{1}{1+q}, \quad M_2 = \frac{4\pi a^3}{GP^2} (1+q). \quad (19)$$

This system of five parameters ( $M_1$ ,  $M_2$ ,  $P$ ,  $a$ ,  $q \equiv M_2/M_1$ ) can be manipulated from the Python interface such that any two of the five parameters are “constrained” as read-only, based on Kepler’s third law. Setting or fitting the individual masses is then possible by choosing two other parameters to be read-only instead.

There are many other similar parameterization choices in a binary model in which one parameterization may be preferred over another because of the orthogonality within the local parameter space or the availability of information known in advance. Table 1 shows a list of the applicable constraints included in the 2.3 release of PHOEBE.

**Table 1**  
Built-in Constraints in PHOEBE

Default Free Parameter(s)	Other Inputs	Default Read-only Parameter(s)
$P_{\text{orb}}, q, a_{\text{orb}}$		$M_1, M_2$
$P_{\text{orb}}$		$f_{\text{orb}}$ (frequency)
$e, \omega_0$		$e \sin \omega_0, e \cos \omega_0$
$a_{\text{orb}}, i_{\text{orb}}$		$a_{\text{orb}} \sin i_{\text{orb}}$
$t_{0,\text{supconj}}$	$P_{\text{orb}}, \dot{P}_{\text{orb}}, e, \omega_0, \dot{\omega}, t_0$	$t_{0,\text{perpass}}, t_{0,\text{ref}}$
$t_{0,\text{perpass}}$	$P_{\text{orb}}, \dot{P}_{\text{orb}}, t_0$	$M$ (mean anomaly)
$F$ (synchronicity)	$P_{\text{orb}}$	$P_{\text{rot}}$
$P_{\text{rot}}$		$f_{\text{rot}}$ (frequency)
$M, R_{\text{equiv}}$		$\log g$
$a_{\text{orb}}$	$q$	$a_{\text{comp}}$
$a_{\text{orb}}$	$i_{\text{orb}}, q$	$a_{\text{comp}} \sin i_{\text{orb}}$
Pitch, $i_{\text{orb}}$		$i_{\text{comp}}$
Yaw, $\Omega_{\text{orb}}$ (asc. node)		$\Omega_{\text{comp}}$
$A_v, R_v$		$E(B - V)$
Contacts: $R_{\text{equiv}}$	$q$	$FF$ (fillout factor), $\Omega$ (equipotential)
Optional Constraints		
$T_{\text{eff},1}, T_{\text{eff},2}$		$T_{\text{eff},2}/T_{\text{eff},1}$
$R_{\text{equiv},1}, R_{\text{equiv},2}, a_{\text{orb}}$		$R_{\text{equiv},2}/R_{\text{equiv},1},$ $(R_{\text{equiv},1} + R_{\text{equiv},2})/a_{\text{orb}}$
Semidetached: $R_{\text{equiv}}$		$R_{\text{equiv},\text{crit}}$
$\pi$ (parallax)		$d$ (distance)

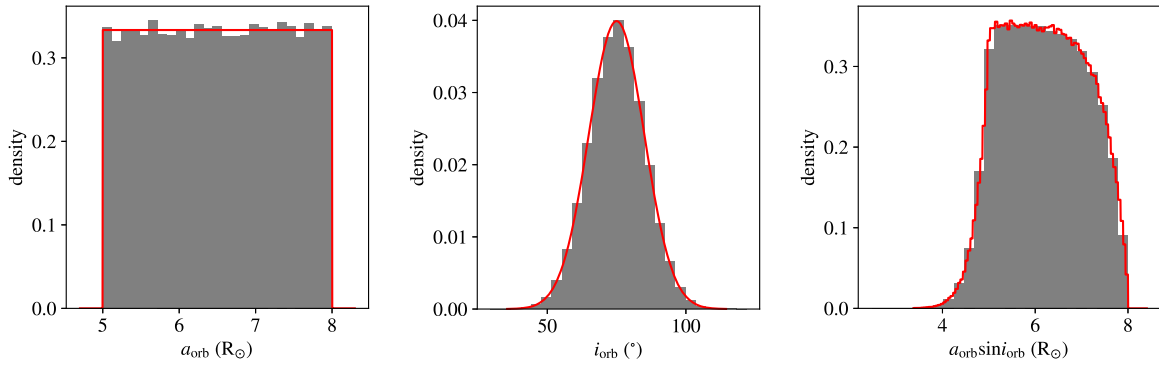
**Note.** Parameters in the top half of the table are included in systems by default, whereas the bottom parameters can optionally be added. The parameters in the right column are, by default, read-only and derived from the parameters in the two left columns. The read-only status of any parameter in the right column can be replaced with one in the left column (as long as any given parameter is “constrained” by a single expression), allowing for a large combination of parameterization choices. Parameters in the middle column are included in the expression itself, but are not currently available to be chosen as the read-only parameter.

### 3.2. Priors

PHOEBE allows creating and attaching distributions (as DISTL<sup>10</sup> distribution objects) to any parameter represented by a float value in the system. DISTL is a Python package that handles defining, converting, storing, and manipulating various probability distributions. The version currently packaged with PHOEBE supports the following univariate distribution types: uniform (boxcar), Gaussian, delta, histogram, and samples (which generates a kernel density estimation, KDE, under the hood around an input set of samples to allow drawing from the same distribution as the samples while not drawing from the set directly). Additionally, the following multivariate distribution types are supported: multivariate Gaussians, multivariate histograms, and multivariate samples that each contain the known covariances between parameters.

Once defined and attached to a parameter, these distributions can then be plotted directly or propagated through the constraint logic discussed in Section 3.1 (see Figure 4). Any distribution attached to a “free” parameter can have its value randomly drawn and set for the respective distribution. Any distribution (whether attached to an adjustable or “read-only” parameter) can provide the probability of drawing the current face value of the parameter. When passed to an optimization or

<sup>10</sup> <https://github.com/kecnry/distl>



**Figure 4.** Uniform distribution placed on the semimajor axis (left), and a Gaussian distribution placed on the inclination (middle). These distributions can then be propagated through the logic discussed in Section 3.1 to get the resulting distribution on  $a \sin i$  (right).

sampling algorithm that supports priors, these probabilities are then included in the merit function (see Equations (12) and (17)).

This flexibility allows for numerous useful scenarios. For example, it is possible to set a prior distribution on  $a \sin i$  (e.g., known from the literature) while marginalizing over a defined range in both  $a$  and  $i$ . This also allows for setting a prior on any observationally constrained parameter that is not directly a data set observable, including  $\log g$  or mass from spectra reduction, for instance. As will be discussed in Section 6, it is also possible to sample with one parameterization ( $e$  and  $\omega_0$ , for example) but then expose the posteriors and uncertainties in another ( $e \sin \omega_0$  and  $e \cos \omega_0$ ).

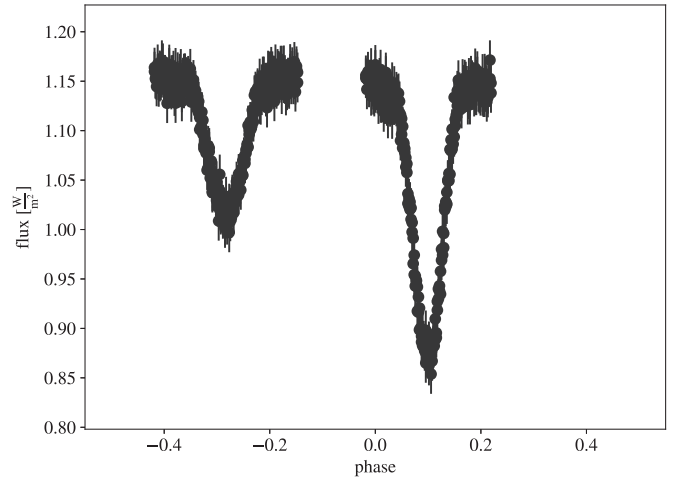
### 3.3. Angle Wrapping

All parameters and distributions that correspond to angles (argument of periastron, longitude of ascending node, and so on) include automatic angle wrapping (with the exception of inclination, which is limited to the range between zero and  $\pi$  to avoid ambiguity with the longitude of the ascending node). For example, if setting a value, either manually or within an optimizer or sampler, outside the  $[0, 2\pi]$  range, the value will automatically be mapped back onto the range  $[0, 2\pi]$ . This is particularly useful if the true value is near the wrapping point, as the optimizer or sampler can explore the parameter space continuously.

This can cause some issues within optimizing or sampling when the merit function is insensitive to one of the parameters. For example, for a circular system, the argument of periastron has no effect on the observables. If allowed to wrap continuously, the sampler will continue to randomly walk indefinitely. To avoid this, the per-wrapped angles are limited to within  $\pi$  from the central value of the initializing distribution (see Section 5), essentially placing an uninformative prior and preventing the sampler from wandering across the wrapped space.

### 3.4. Forward Model

The “forward model” consists of the synthetic model for a fixed set of input parameters, which is then compared to the observations through the merit function, as described in detail in Section 3.

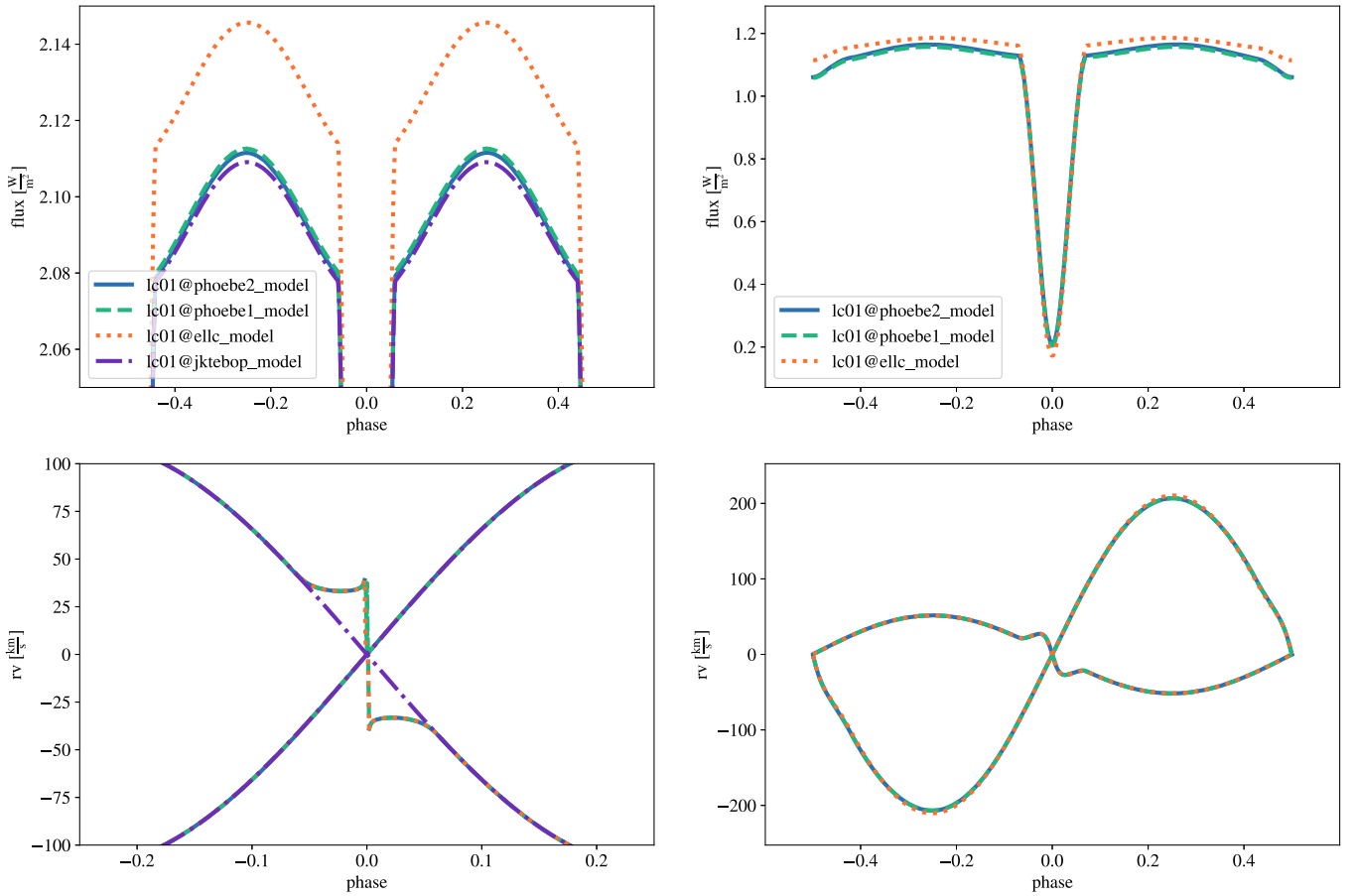


**Figure 5.** The same light-curve data set as in Figures 2 and 3 masked to the phases of ingress and egress (plus an additional 30% padding) as proposed by the light-curve geometry estimator (see Section 2.3).

#### 3.4.1. Phase Masking, Exposure Times, Undersampling, and Interpolation

By default, PHOEBE will compute the forward model at the time stamps of the observations. For convenience, PHOEBE also supports excluding specific phases of the data to improve computational efficiency. For example, during initial exploration and optimizations of basic geometric and orbital parameters, it can be useful to exclude the out-of-eclipse section of light curves (see Figure 5). These data should then be reintroduced into the merit function for further optimization and determination of posteriors from all available observations. To allow for this, PHOEBE implements a very basic “phase masking” that can be enabled and disabled. Note, however, that any outlier removal or more complex masking (in time space, for example) will currently need to be done outside of PHOEBE. When phase masking is being used within optimizing or sampling, PHOEBE will exclude those times from both the forward model and the residuals in the merit function.

Additionally, it is often useful to “downsample” the observations to save on computation time or, when not optimizing or sampling, to “oversample” for plotting purposes or to predict how the model behaves outside the limits of observations. To accomplish all of these use cases, PHOEBE allows overriding the computation times of each data set to a custom array of times. When optimizing or sampling, PHOEBE will, by default, automatically take the shorter of these two arrays (the masked observation times or the requested computation times). Within the



**Figure 6.** Comparison of several supported backends run through PHOEBE on the same system (detached system on the left and semidetached system on the right). Although not identical, this is useful for comparing the assumptions between different models and for conveniently taking advantage of models that are most efficient for a particular science case or system. Slight vertical scaling differences are likely because of the assumptions in each code as well as the assumptions in the flux scaling described in Section 3.4.4.

merit function, the synthetic model is then interpolated onto the times of the masked observations. In cases where the system is not time dependent, PHOEBE will allow for “extrapolating” outside the time bounds of the synthetic model by interpolating in phase space, but it will not allow doing so when the system is time dependent (for example, if any time derivative is nonzero, including the presence of apsidal motion, or at least one of the components has spots and is asynchronously rotating).

Lastly, PHOEBE supports oversampling the model over the exposure time for each requested synthetic time. The resulting synthetic flux at the midtime is then exposed as the mean of the oversampled fluxes. Long-cadence Kepler data, for example, has a 29.4 minute exposure time, which results in phase smoothing of the photometry. Setting the appropriate exposure time and enabling oversampling allow the model to reproduce this effect.

#### 3.4.2. Limb Darkening

PHOEBE 2.0 introduced automatic interpolated limb darkening by default (see Prša et al. 2016 and Section 5.2.3), in addition to manual coefficients for linear, logarithmic, quadratic, square root, and power laws. PHOEBE 2.2 (Jones et al. 2020) then included the ability to automatically query coefficients for any of the built-in laws, on a per-surface-element basis.

Optimizing or sampling in PHOEBE 2.3 supports both interpolated and “lookup” limb darkening but does not currently support fitting manually set coefficients. Interpolated limb darkening and per-element lookup are only supported within the PHOEBE backend itself; for other backends (see Section 3.4.3), the coefficients are queried from the atmosphere tables in PHOEBE based on the mean stellar parameters.

#### 3.4.3. Alternate Computation Backends

PHOEBE has been designed to be as robust and accurate as possible (for specific implementation details, see Prša et al. 2016; Horvat et al. 2018; Jones et al. 2020), but this does come at the cost of computational efficiency. Any inefficiency is magnified when a large number of forward-model instances needs to be generated by an optimizer or sampler. Depending on the specific system being studied, substantial time can be saved by “disabling” certain effects and making simplifying assumptions. PHOEBE does support some of these assumptions; irradiation is expensive and can be disabled when negligible or spherical stars can replace Roche-distorted stars, for instance. Other codes may be better suited or more optimized for a given system. In some cases, higher-order effects that require the use of PHOEBE can be ignored while searching the whole parameter space and optimizing orbital parameters, for example, but may still be necessary to include

**Table 2**  
Comparison of Available Physics and Features in the Forward Model “Backends” as Wrapped by PHOEBE

	PHOEBE 2	PHOEBE 1 (“legacy”)	ellc	jktebop
Supported versions	2.3+	1.0	1.8.2+	v40+
LCs	Yes	Yes	Yes	Yes
LCs (absolute fluxes)	Yes	Yes	Rescaled	Rescaled
RVs (dynamical)	Yes	Yes	Yes	Yes
RVs (with R-M)	Yes	Yes	Yes (no irradi.)	No
Spectral line profiles	Forward model	No	No	No
Orbits	Forward model	No	No	No
Access to underlying meshes	Forward model	Forward model	No	No
Detached systems	Yes	Yes	Yes	Yes
Semidetached systems	Yes	Yes	Yes	No
Contact systems	Yes	Yes	No	No
(Native) parameterization	Period, mass ratio, semimajor axis, synchronicities, equivalent radii, $T_{\text{eff}}$ , eccentricity ( $e$ ), arg. of per. ( $\omega_0$ )	Period, mass ratio, semimajor axis, synchronicities, equipotentials, $T_{\text{eff}}$ , eccentricity ( $e$ ), arg. of per. ( $\omega_0$ )	Period, mass ratio, synchronicities, fractional radii, SB ratio, $\sqrt{e} \cos \omega_0$ , $\sqrt{e} \sin \omega_0$	Period, mass ratio, sum of fractional radii, ratio of radii, SB ratio, $e \cos \omega_0$ , $e \sin \omega_0$ (or $e$ & $\omega_0$ )
Surface distortion	Roche, rotating star, sphere	Roche	Roche, sphere, roche_v, poly1p5, poly3p0, love	Biaxial spheroid for ellipsoidal and irradiation contributions, sphere for eclipse shapes
Asynchronous rotation	Yes	Yes	Yes	No
Atmospheres	Blackbody, Castelli–Kurucz, phoenix	Blackbody (planckint), Castelli–Kurucz (atmx)	None	None
Number of supported passbands	~30	~30	0	0
Irradiation	Horvat/Lambert, Wilson	Wilson	Lambert	Biaxial spheroid
Limb darkening	Interpolated, lin, log, quad, sqrt, power	lin, log, sqrt	lin, log, quad, sqrt, power	lin, log, quad, sqrt
Gravity brightening/darkening	Yes	Yes	Yes	Yes
Spots	Circular	Circular	Circular	No
Rømer delay	Yes (optional)	No	Yes	No
Apsidal motion	Yes	Yes	Yes	No
Period time derivative	Yes	Yes	No	No
Spin–orbit misalignment	Yes	No	Limited	No
Extinction/reddening	Yes	No	No	No

**Note.** Note that some additional capabilities may be available in the native versions of the codes but are excluded here if they are not easily mapped from PHOEBE’s parameterization.



when attempting to get precise estimates for the final reported values.

For this reason, and to enable comparing codes against each other, PHOEBE includes wrappers to the forward-model component of several other public codes, including PHOEBE legacy (based on Wilson–Devinney; Prša & Zwitter 2005; Wilson & Devinney 1971; Wilson 1979, 2008; Wilson & Van Hamme 2014), ELLC (Maxted 2016), and JKTEBOP (Southworth et al. 2004, 2007, 2009; Southworth 2011). These forward models are designed to be “drop-in” replacements to the forward model provided by PHOEBE, with a minimal number of additional options that are specific to that code. Figure 6 shows a comparison of several forward models computed through PHOEBE on the same set of parameters. There are some notable discrepancies between the models caused by the different assumptions in each code and limitations on the ability to translate parameterizations and outputs of each code. However, this interface provides the ability to conduct these comparisons and choose the most appropriate model for any given system. Table 2 shows an overview of the implemented features, limitations, and assumptions of each of the backends currently available.

These wrappers translate from the parameterization used within PHOEBE to the parameterization used within the requested forward model, called the external code, and translate the results into the correct units used by PHOEBE. It is important to note that this translation may not always be exact and does add some time cost: a native optimization and sampling wrapped around the individual codes will always perform faster. However, regardless of the chosen forward model, all backends can also benefit from some of the functionality built into the PHOEBE frontend, including flexible parameterization via “constraints” (see Section 3.1), phase masking, exposure times, and time interpolation (see Section 3.4.1), limb-darkening coefficient lookups (see Section 3.4.2), flux scaling via passband luminosities and third light in light curves (some natively; see Section 3.4.4), systemic velocities and per-component radial-velocity offset (see Section 3.4.5), noise nuisance to handle observational uncertainty underestimation (see Section 3.5), Gaussian processes (see Section 3.6), and all of the optimization and sampling methods implemented in PHOEBE. If using any of these backends or disabling physical effects within the PHOEBE backend, it is always good practice to occasionally compare the resulting model with one computed without these assumptions.

Whenever publishing work that makes use of these alternate backends, we encourage citing the external code as appropriate.

#### 3.4.4. Flux Scaling (Passband Luminosity, Distance, and Third Light)

The surface fluxes (formally computed at a distance of 1 m from the equivalent point source) calculated directly by the PHOEBE 2 backend are defined as (see Prša 2018)

$$F_{\text{pb}|w} = \mathcal{P}_{\text{int}|w} \sum_{\text{stars}} S_{\text{rel}} \sum_{\text{elems}} \langle I_{\mu, \text{abs}|w} \rangle \mathcal{V} \mu \Delta A, \quad (20)$$

where  $|w$  denotes the flux weighting scheme (energy or photon counts),  $\mathcal{P}_{\text{int}|w}$  is the suitably weighted integral of the passband transmission function, “stars” are the components defined in the system,  $S_{\text{rel}}$  is the per-star scaling factor that translates between absolute and relative units,  $\langle I_{\mu, \text{abs}|w} \rangle$  are

passband-averaged, per-surface-element specific emergent intensities interpolated from the model atmosphere tables,  $\mathcal{V}$  is the surface element visibility function, and  $\mu \Delta A$  are projected surface areas of each element. For components where a passband luminosity is provided by the user,  $S_{\text{rel}}$  is determined as

$$S_{\text{rel}} = \frac{L_{\text{pb, rel}|w}}{L_{\text{pb, abs}|w}}. \quad (21)$$

Note that these absolute and relative luminosities are defined at time  $t_0$  and exclude any extrinsic or aspect-dependent effects (distance, irradiation, spots, pulsations, boosting, and so on). Absolute luminosities are determined from absolute normal intensities from the tables:

$$L_{\text{pb, abs}|w} = \pi \mathcal{P}_{\text{int}|w} \sum_{\text{elems}} \langle I_{\perp, \text{abs}|w} \rangle \mathcal{D}_{\text{int}} \Delta A, \quad (22)$$

where  $\langle I_{\perp, \text{abs}|w} \rangle$  are the passband-averaged absolute emergent normal intensities interpolated from the model atmosphere tables,  $\mathcal{D}_{\text{int}}$  are the integrals of the limb-darkening functions over all angles, and  $\Delta A$  are the surface areas of each element (Prša 2018).

Since version 2.2 (Jones et al. 2020), PHOEBE has supported multiple modes for handling passband luminosities and scaling of light-curve fluxes:

1. Absolute: Intensities are used and integrated directly as retrieved from atmosphere and passband tables, resulting in absolute fluxes. In this case, the scaling factor,  $S_{\text{rel}}$ , in Equations (20) and (21) is, by definition, unity.
2. Data set-scaled: If observations are provided, “*data set-scaled*” allows one to automatically scale the resulting absolute fluxes to the data using a least-squares algorithm. This is particularly useful for optimizing parameters for a normalized light curve without having to worry about correlations with the luminosity. Although convenient, when determining uncertainties from sampling (Sections 5 and 6), it is advised to instead marginalize over the passband luminosities in order to avoid underestimated uncertainties.
3. Component-coupled: A passband luminosity ( $L_{\text{pb, rel}|w}$ ) for one of the components is provided by the user, and intensities are internally scaled such that this prescribed intrinsic luminosity at time  $t_0$  is achieved (Equation (21)). This same scaling factor is then used for both components in Equation (20).
4. Data set-coupled: If multiband photometry is available, “*data set-coupled*” allows use of the same scaling factor across multiple light curves to preserve any color information, adopting  $S_{\text{rel}}$  for both components in one data set from those defined in another data set, and therefore allowing  $L_{\text{pb, rel}|w}$  to be computed from Equation (21).
5. Decoupled: Decoupling allows for setting the desired luminosities of stars independently, ignoring any scaling information from the atmosphere and passband tables. Here the scaling factors for each star in Equations (20) and (21) are allowed to be independent of each other.

For most “alternate backends” (see Section 3.4.3 and Table 2), where scaling the intensities or passing passband

luminosities directly is not possible, the returned fluxes or magnitudes need to be rescaled into the units used by PHOEBE by estimating (using the same assumptions as above) the total passband flux from the individual passband luminosities:

$$F_{\text{alt, backend, scaled}} = F_{\text{alt, backend}} \sum_{\text{stars}} \frac{L_{\text{pb, rel}|w}}{4\pi}. \quad (23)$$

When using the “*decoupled*” mode, the desired luminosities  $L_{\text{pb, rel}|w}$  per star can be used directly to estimate this flux scaling without the need for computing absolute luminosities. But for any other mode, one or more relative passband luminosities must be computed internally (via Equation (21)) by first determining the absolute luminosities of each component. PHOEBE Legacy follows a logic similar to PHOEBE 2 and takes passband luminosities as input; the respective  $L_{\text{pb, rel}|w}$  are passed directly and the returned fluxes do not need to be rescaled with Equation (23). However, unless using “*decoupled*,” “*component-coupled*” (which is supported natively by PHOEBE Legacy), or “*data set-scaled*,” the absolute luminosities still need to be estimated to handle the appropriate coupling of the relative luminosities. In any of these cases, two methods are introduced in PHOEBE 2.3 to estimate these absolute luminosities:

1. PHOEBE meshes at  $t_0$ : A PHOEBE mesh is created at time  $t_0$  using the Roche model for surface distortion and is then populated with the appropriate intensities from the given atmosphere and passband tables and integrated over the entire surface to calculate the intrinsic passband luminosity (Equation (22)). As building the mesh is one of the more computationally expensive steps in PHOEBE, estimating luminosities in this way can largely negate the speed benefits of using these “alternate backends,” but it is more accurate for any significant surface distortion.
2. Stefan–Boltzmann approximation: The mean stellar values for  $R_{\text{equiv}}$ ,  $T_{\text{eff}}$ ,  $\log g$ , and abundance are used to query the selected atmosphere and passband tables for the “mean”  $\langle I_{\perp, \text{abs}|w} \rangle$  and the integral of the limb-darkening model  $\mathcal{D}_{\text{int}}$ . The absolute passband luminosity for each star is then approximated as

$$L_{\text{pb, abs}|w} = 4\pi R_{\text{equiv}}^2 \langle I_{\perp, \text{abs}|w} (T_{\text{eff}}, \log g, \text{abun}) \rangle \mathcal{D}_{\text{int}} \times (T_{\text{eff}}, \log g, \text{abun}, \text{ld}) \mathcal{P}_{\text{int}|w}. \quad (24)$$

This essentially treats each star as a uniform sphere for the purposes of computing the absolute luminosity (and therefore the scaling factor). For cases where distortion is minimal or the exact scaling is not important, using this approximation will be significantly faster than requiring a mesh to be built.

It is important to note that, with either method, Equation (23) makes assumptions in estimating passband flux levels from these luminosities, so the scaled fluxes cannot be exact. Furthermore, for backends that accept surface brightness ratio instead of absolute effective temperatures, these are estimated as the ratio of the calculated passband luminosities over the square of their respective equivalent radii. And lastly, the normalizations natively used by these codes differ slightly, both from each other and from PHOEBE. ELLC normalizes the exposed light curves by the integrated flux over the irradiated surfaces of each component (Maxted 2016), whereas JKTEBOP normalizes to the magnitude of the system out-of-eclipse at quadrature (J. Southworth 2020, private communication). The

effect of these different treatments on PHOEBE’s rescaling can be seen in Figure 6. Despite these drawbacks, this implementation allows for simple support of passband-level effects that may not be supported natively by these other codes and allow for an easier drop-in replacement.

To be consistent, the fluxes returned from all backends do not include contributions from third light and distance (they are disabled even for codes that would otherwise natively support these effects to ensure consistency). In the case where third light is provided in fractional (instead of flux) units, we first convert to flux units from the scaled luminosities of each star:

$$l_{3\text{flux}} = \frac{l_{3\text{frac}}}{1 - l_{3\text{frac}}} \sum_{\text{stars}} \frac{L_{\text{pb, rel}|w}}{4\pi} \quad (25)$$

where “stars” are again the components in the system, not including the source of the extraneous third light. This uses the same uniform and spherical approximation as was used for the luminosity-to-flux translation in Equation (23) and also requires absolute luminosities to be estimated through either of the two methods mentioned above whenever not using the “*decoupled*” mode.

If using the “*data set-scaled*” mode, the fluxes are instead scaled to the observations—before the inclusion of third light and so acting directly on absolute fluxes (i.e.,  $L_{\text{pb, rel}|w} \equiv L_{\text{pb, abs}|w}$ ) as passband luminosity scaling and distance are arbitrary in this case—by determining a flux-scale factor,  $S_{\text{ds}}$ , via least squares:

$$F_{\text{synthetic, data set-scaled}} = S_{\text{ds}} F_{\text{backend}} + l_{3\text{flux}}. \quad (26)$$

For any case where  $l_{3\text{frac}}$  is provided instead, the conversion to  $l_{3\text{flux}}$  will also include this same scale factor:

$$F_{\text{synthetic, data set-scaled}} = S_{\text{ds}} \left( F_{\text{backend}} + \frac{l_{3\text{frac}}}{1 - l_{3\text{frac}}} \sum_{\text{stars}} \frac{L_{\text{pb, abs}|w}}{4\pi} \right). \quad (27)$$

PHOEBE 2.3 also introduces support for using “*data set-scaled*” on multiple light curves that are coupled together using the “*data set-coupled*” mode. In this case, Equations (26) and (27) are used, as appropriate, but fitting for a single  $S_{\text{ds}}$  for all of the coupled data sets simultaneously.

In all other (non-“*data set-scaled*”) modes, distance and third light are then included on top of the returned fluxes from the backend:

$$F_{\text{synthetic}} = \frac{F_{\text{backend}}}{d^2} + l_{3\text{flux}} \quad (28)$$

where  $F_{\text{backend}}$  is either adopted from PHOEBE (Equation (20)) or any alternate backend after rescaling, if necessary (Equation (23)).

#### 3.4.5. Systemic Velocity and Per-component Radial-velocity Offsets

Similar to distance and third light being included on top of the respective back-end fluxes, radial velocities directly from any backend exclude any offsets (even for backends that do natively support systemic velocities). The final synthetic radial velocities are then defined as

$$\text{RV}_{\text{synthetic}} = \text{RV}_{\text{backend}} + \text{RV}_{\gamma} + \text{RV}_{\text{offset}} \quad (29)$$

where  $\text{RV}_{\gamma}$  is the constant barycentric systemic velocity added to all RVs (for all components across all data sets), whereas

$RV_{\text{offset}}$  allows for per-data set and per-component offsets to the velocities.

These per-component offsets are particularly useful for hot stars, where offsets in spectral lines are commonplace because they are being created in different levels of the atmospheres (Underhill & Hill 1994; Harmanec et al. 2002; Shenar et al. 2018). The magnitude of this offset can vary between the two components in the system but also between radial-velocity data sets that were reduced from different spectral lines. By building this into the model itself, these offsets can be marginalized over, and any correlations (with the mass ratio or systemic velocity  $RV_{\gamma}$ , for example) can be properly accounted for in the posteriors.

### 3.5. Noise Nuisance

Determining posteriors and uncertainties of physical parameters of the system depends strongly on the observational uncertainties. Unfortunately, these are often known to be underestimated, so it is important that any such underestimation does not propagate through to a biased solution or underestimated model uncertainties.

In the cases where the quoted observational uncertainties may be underestimated by a constant factor (such as in Kepler data; see Jenkins 2017), we can introduce the uncertainty scaling factor directly into observational uncertainties in the definition of the merit function (see also Equations (14) and (15)):

$$\sigma^2 = \sigma_o^2 + y_m^2 e^{(2\sigma_{\text{inf}})}. \quad (30)$$

This can be particularly useful as a “nuisance parameter” while sampling to determine posteriors (see Sections 5 and 6). By marginalizing over this underestimation factor, any degeneracies between the factor itself and the physical model parameters can be encoded in the resulting posteriors.<sup>11</sup>

We refer the reader to Hogg et al. (2010), which provides a practical overview to handling observational uncertainties in several different situations.

### 3.6. Gaussian Processes

A Gaussian process (GP) is a random process in which each data point is drawn from the assigned random variable where the joint distribution of all variables is Gaussian. GPs allow modeling of the noise correlations (such as serial correlation or heteroscedasticity) on top of the astrophysical signal described by the noise-free forward model, and thus replace the need to fit functions (such as polynomials) to the data for the purpose of detrending. Within PHOEBE, GPs are applied to the forward model and, as such, are available for any of the supported computation backends with any choice of the kernel (which describes the covariances between adjacent points). PHOEBE implements GPs via CELERITE, a Python implementation of GPs that aims to be computationally efficient and scalable to a large number of data points (Foreman-Mackey et al. 2017). PHOEBE 2.3 supports any combination of Matern 3/2 and Simple Harmonic Oscillator kernels, but intentionally excludes the white noise term (“jitter”) natively supported by CELERITE to avoid conflicting with the noise nuisance parameter separately implemented within PHOEBE (Section 3.5).

After the original forward model is computed and interpolated onto the observation times (if necessary), the residuals between the forward model and observations are determined. These residuals and the input parameters for the kernels are then passed to CELERITE, which then provides the GP component of the model. The forward model and GP contribution are then added, resulting in the final model (which in turn is used to compute the residuals or merit function for optimizers or samplers). Note that GPs require the final forward model to be exposed at the exact times of the observations, although the physical forward model is still computed at the requested times or phases, as discussed in Section 3.4.1.

Figure 7 showcases an example of including synthetic observations with additional superimposed trends and noise. Here the inclusion of GPs accounts for the residuals between the observations and the synthetic forward model.

While GPs would ideally account only for the noise component (stochastic *and* correlated), their high degree of freedom can lead to overfitting the data and accounting for some of the signal. This can manifest in excess GP spectral power at the orbital period, and, in consequence, leads to degenerate solutions and affects parameter posteriors. It is thus important to properly marginalize (see Section 5) over GP parameters to avoid, or at least quantify, the level of overfitting.

### 3.7. Parallelization

The PHOEBE backend itself is parallelized at the per-time level via a Message Passing Interface (MPI) implementation. Here the initial setup is done by a single processor, which then sends individual time stamps to the remaining processors to compute the requested synthetic observables at those times. The main processor then compiles and orders the results and exposes the model. The wrappers around other backends generally support parallelization at the per-data set level, allowing each processor to compute the observables for all times, but for a single data set. However, within optimizers or samplers that support that capability, the parallelization within the forward models is disabled in favor of parallelizing at the per-model level via either MPI or multiprocessing. As this has less overhead, it is often more efficient.

It should be noted that for MCMC (see Section 5.1), for example, this loses its advantages if there are more processors available than requested walkers. Whenever this is the case, PHOEBE automatically switches to run MCMC in serial with each iteration parallelized per time.

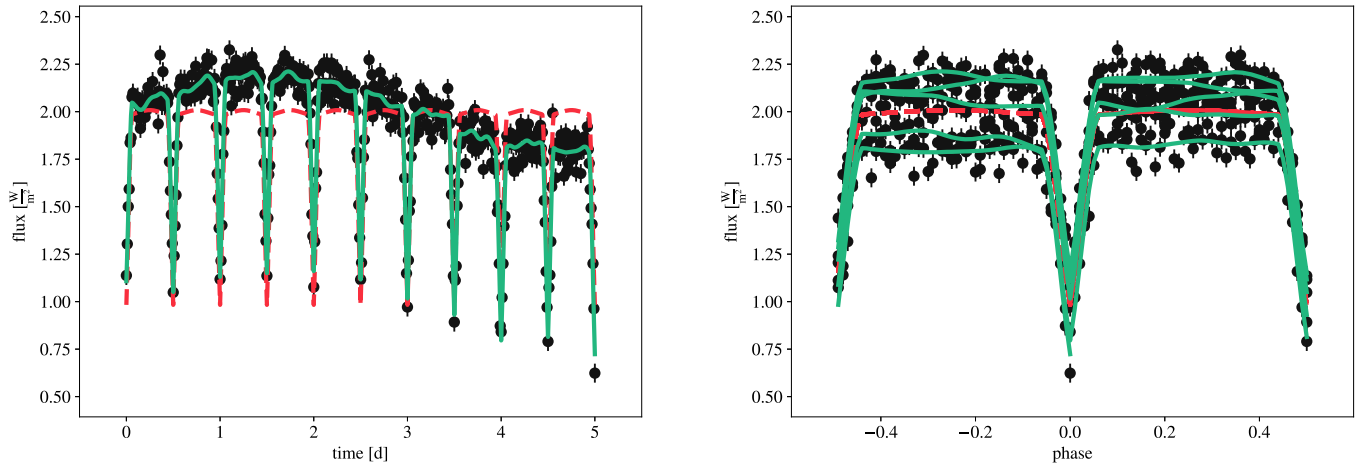
In addition, if not using MPI, PHOEBE will make use of a multiprocessing pool across all available processors for any backend that supports multiprocessing (currently emcee and dynesty).

## 4. Optimizers

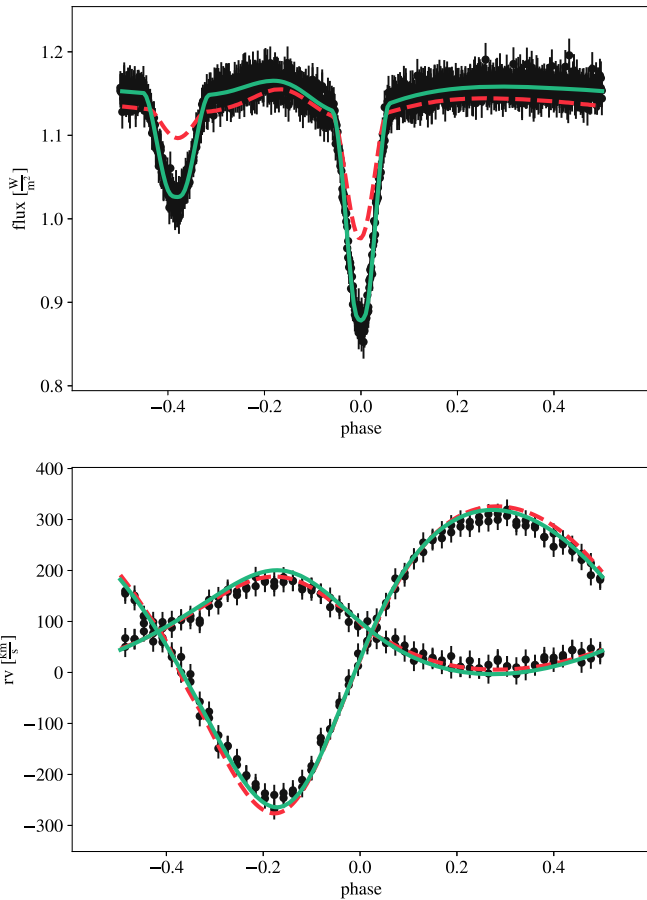
PHOEBE 2.3 includes wrappers around several optimization algorithms from `scipy.optimize`<sup>12</sup> (Virtanen et al. 2020), including Nelder–Mead (Gao & Han 2012), Powell, and conjugate gradient. These optimizers can be quite efficient at improving a model fit once already in the vicinity of the

<sup>11</sup> See the example for fitting a line to data with underestimated uncertainties in the EMCEE online documentation at <https://emcee.readthedocs.io/en/v3.0.0/tutorials/line/>.

<sup>12</sup> PHOEBE 2.3 requires `scipy` 1.2+.



**Figure 7.** Forward model both without (red dashed lines) and with (green solid lines) the Gaussian processes. Two Gaussian process kernels (a Matern 3/2 kernel and a simple harmonic oscillation kernel) are shown in time space (left) and phase space (right).



**Figure 8.** Model improvement after adopting proposals from estimators (dashed red lines) and after running Nelder–Mead (solid green lines).

optimal solution, found via estimators or a full global parameter space search, for example.

PHOEBE allows for choosing which parameters will be adjusted by the optimizer, as well as the ability to define priors, if desired. The respective algorithm is then called by passing the negative log likelihood (see Section 3) to the appropriate SCIPY algorithm. The optimization results in the maximum a priori (MAP) solution if priors are defined, or the maximum-likelihood

estimation otherwise. It is important to note that, irrespective of whether the user provides priors or not, the parameter limits, wrapping limits, and any failed models still act as uninformative priors that penalize the merit function.

As with estimators (Section 2), PHOEBE then exposes the proposed values for each of the adjusted parameters along with the value of the merit function before and after optimization as well as any diagnostic values returned by SCIPY, allowing the user to decide which, if any, of the proposed values to adopt. Figure 8 shows the results of a Nelder–Mead optimization on several parameters starting at the proposed values from estimators. After just a few hundred iterations, the residuals and  $\chi^2$  show that the model has improved significantly.

## 5. Samplers

Samplers are not designed to find the global solution; rather, their purpose is to explore the shape of the local parameter space and provide robust posteriors and uncertainties that expose the underlying degeneracies between various parameters.

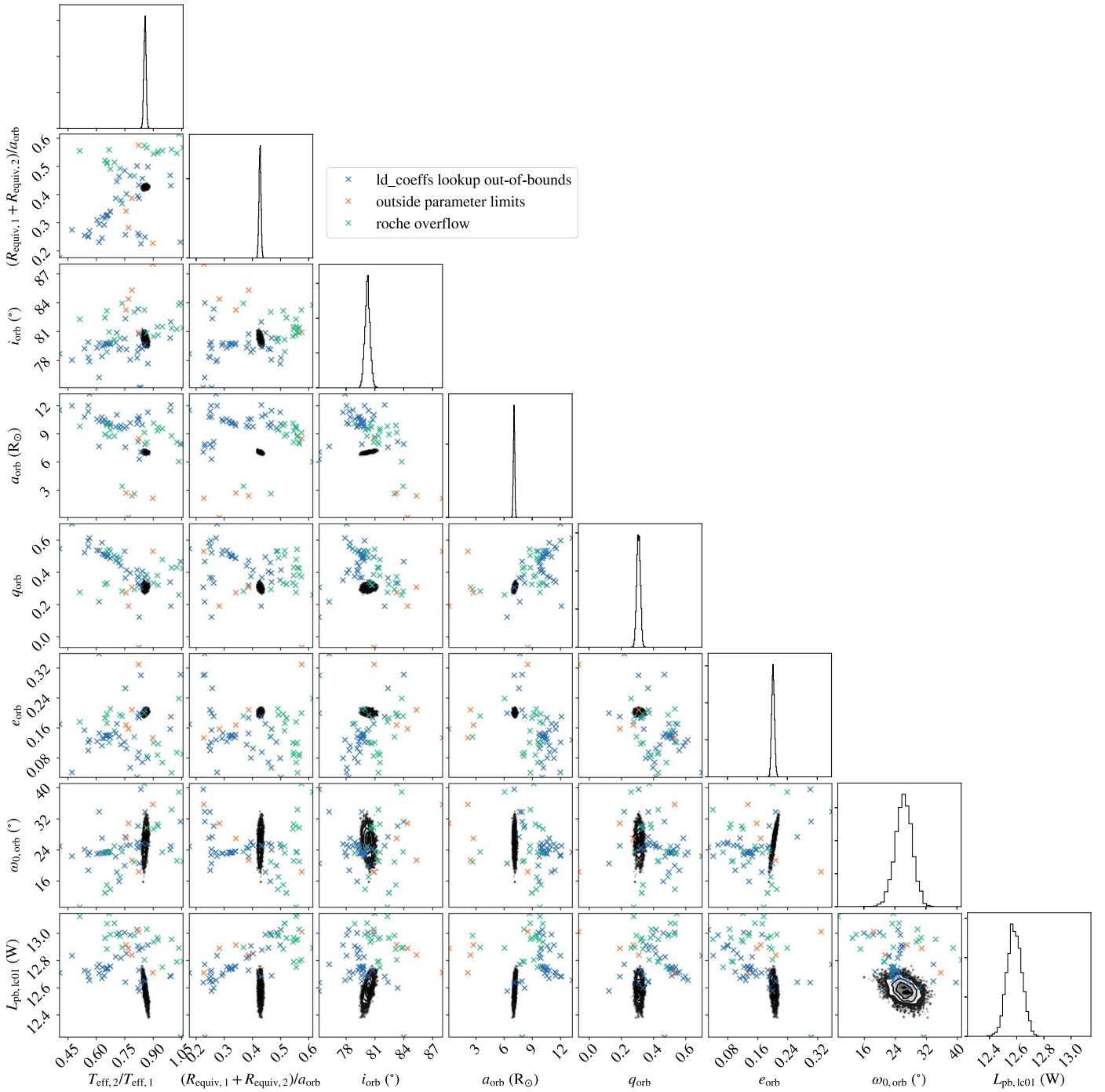
### 5.1. Markov Chain Monte Carlo (*emcee*)

MCMC is a Bayesian method of sampling the parameter space in order to estimate the posterior density function—including any correlations between the sampled parameters—by using the log probability as the merit function (see Section 3). Affine-invariant MCMC algorithms (Goodman & Weare 2010) sample this same space based on the *ensemble* of workers and generally have a shorter burn-in period and require less user tuning (Foreman-Mackey et al. 2013), and they are therefore a good candidate to use whenever the forward model is computationally expensive and the probability space may not be orthogonal.

The EMCEE<sup>13</sup> (Foreman-Mackey et al. 2013, 2019) Python package is an affine-invariant MCMC Python implementation that has been widely adopted by the field. PHOEBE includes a wrapper around EMCEE, exposing most of the basic functionality along with the convenience of easily adding or removing sampled parameters and using complex distributions for priors and the initial sample, while handling the merit function and

<sup>13</sup> PHOEBE 2.3 has been tested with EMCEE v3.0 (and does not support earlier versions).





**Figure 9.** Example corner plot showing the posteriors from an emcee run along with the positions in the parameter space of “failed/rejected samples.” The legend shows that some of the values proposed by emcee were rejected due to Roche overflow or a failure to look up appropriate limb-darkening coefficients, for example. This plot is particularly useful for diagnosing any issues, especially if the posteriors seem to be pushing against a boundary.

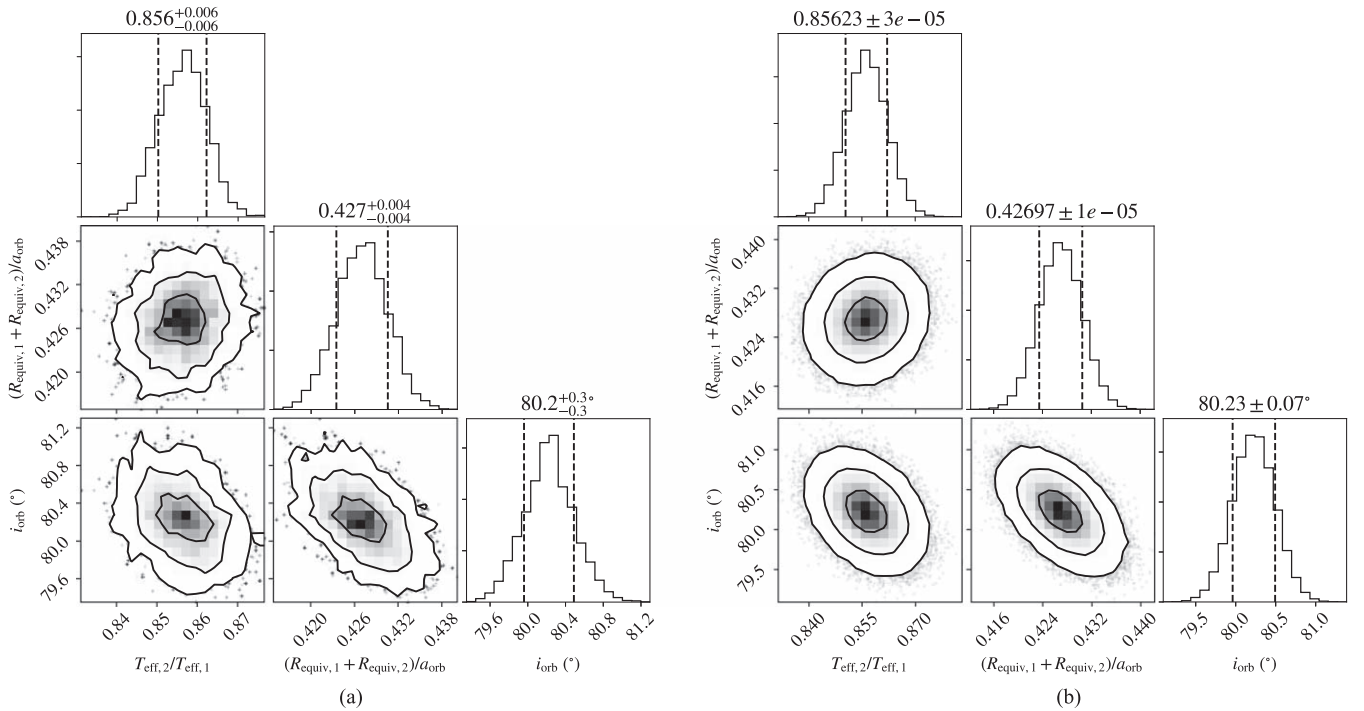
parallelization issues. However, as with any algorithm, it is still important to understand EMCEE itself, how best to adjust the inputs, and how to interpret the results. Foreman-Mackey et al. (2013) provide a good overview and discussion, and the online documentation for EMCEE (<https://emcee.readthedocs.io>) provides several helpful examples.

When preparing an EMCEE run through PHOEBE, the user can define the distributions to be used for the initial sample (which defines which parameters will be included in the samples) in any preferred parameterization and, optionally, distributions for the priors. If priors are not provided, uninformative distributions are

still adopted based on the limits of the individual parameters and angle-wrapping limits (see Section 3.3). Furthermore, if any individual model fails for any reason, the merit function (see Section 3) will return  $-\infty$ , essentially acting as a prior on the allowed parameter space that is due to unphysical systems or limitations of the backend itself. The causes for each of these rejected samples can optionally be exposed to the user, as shown in Figure 9.

Although the initial sample and the priors can be the same set of distributions, it is generally good practice to start EMCEE in an  $N$ -dimensional hyperball (i.e., via Gaussian distributions on





**Figure 10.** Example corner plot of the (burn-in and thinning applied) posteriors derived directly from emcee results (left). These can optionally be translated into a multivariate Gaussian distribution (right), which, when appropriate, represents the same information with only the means and covariance matrix. This is particularly useful for including the means and covariances in a publication or reusing the posteriors as priors.

multiple parameters) around the best-known solution from optimization (see Section 4). Alternatively, it is possible to sample directly from the priors themselves (although this is generally less efficient and more susceptible to getting stuck in local solutions; see Foreman-Mackey et al. 2013 for a discussion on this topic). Priors, on the other hand, can either be informative—uncertainties from an external analysis or from the literature—or uninformative—conservative uniform distributions to prevent the chains from “wandering” into parameter space that is clearly incorrect or unphysical, and they are included in the merit function as described in Section 3.

In addition to distributions, the user can set options for the number of processors, number of walkers, and number of iterations. After the run is complete (or at intermediate steps, as requested by the user), the user can view the progress of the chains and the log probability versus iteration and adjust the burn-in and thinning parameters as necessary. If the run is completed but has not yet converged, PHOEBE allows for continuing an EMCEE run from the existing chains.

In addition to the full chains, the acceptance fractions and autocorrelation times are exposed to the user. As a rule of thumb, Foreman-Mackey et al. (2013) suggest that acceptance fractions all be between 0.2 and 0.5, and enough iterations are done to cover roughly 10 autocorrelation times. PHOEBE then automatically determines defaults for thinning and burn-in based on the autocorrelation times of the chains:

$$\text{burnin} = F_{\text{burnin}} \max(\tau_{\text{autocorr}}) \quad (31)$$

and

$$\text{thin} = F_{\text{thin}} \min(\tau_{\text{autocorr}}) \quad (32)$$

where  $F_{\text{burnin}}$  and  $F_{\text{thin}}$  are user-defined scaling factors that default to 2 and 0.5, respectively, and  $\tau_{\text{autocorr}}$  are the estimated autocorrelation times, per parameter, as exposed by EMCEE.

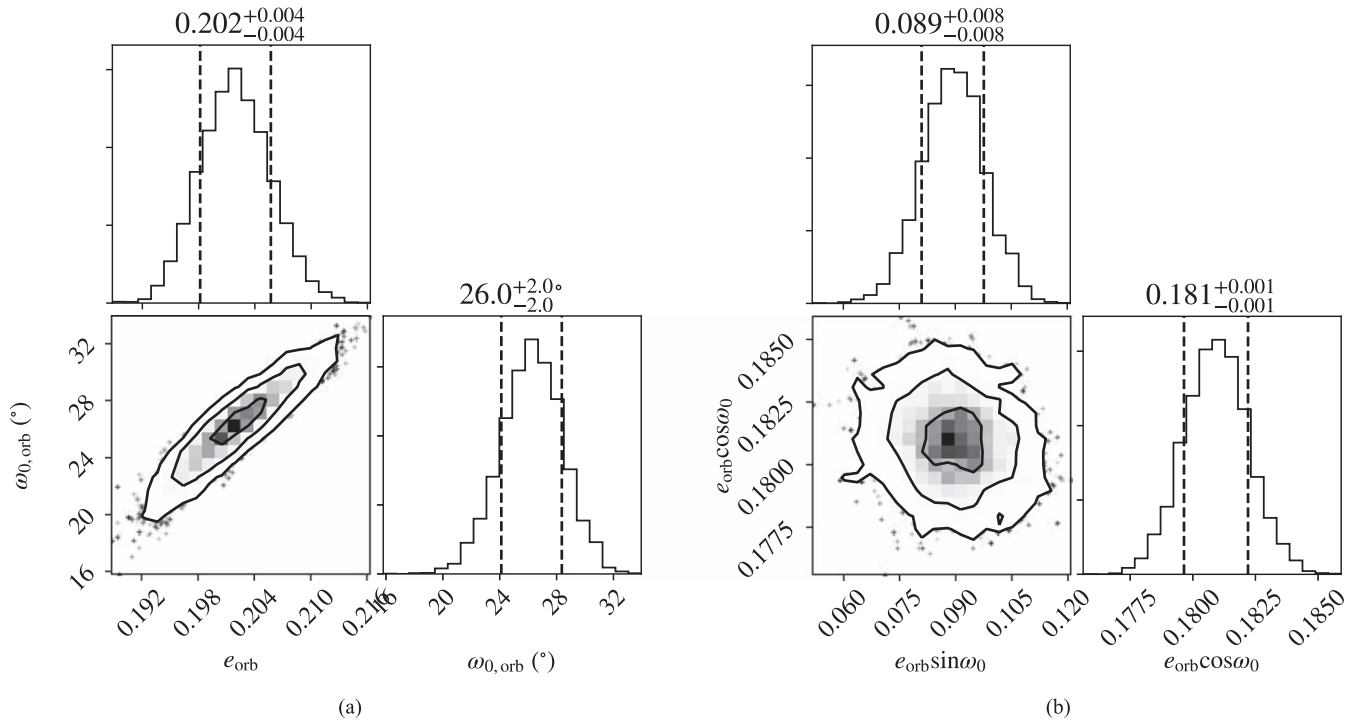
Once the chains sufficiently cover the parameter space and are considered to be converged, the user can then adjust the thinning, burn-in, or a cutoff in log probability to apply to the chains returned by EMCEE and generate a multivariate distribution of the resulting posteriors (see more in Section 6).

## 5.2. Dynamic Nested Sampling (Dynesty)

Nested sampling is another Bayesian method for sampling the parameter space. Unlike MCMC, which samples from an initial distribution, walks the parameter space, and is “penalized” by the priors in the merit function, nested sampling continually samples directly from the priors themselves, which are broken into slices (see Speagle 2020 for an in-depth comparison and discussion). Nested sampling, unlike MCMC, is capable of exploring and exposing multimodal posteriors. However, with wide uninformative priors, nested sampling can become prohibitively expensive. *Dynamic* nested sampling somewhat alleviates this expense by dynamically changing the number of samples per iteration as the algorithm converges to the final posterior. Additionally, unlike MCMC, which is allowed to wander outside the initial sampling distributions as long as the priors allow it, dynamic nested sampling will never consider any solution outside the original parameter space defined by the priors.

The DYNesty<sup>14</sup> code is a dynamic nested sampling Python package. As with EMCEE, it is very useful to first understand the details and intricacies of dynamic nested sampling and the DYNesty code. Speagle (2020) provides a detailed discussion, and the online documentation (<https://dynesty.readthedocs.io>) gives numerous examples.

<sup>14</sup> PHOEBE 2.3 has been tested with DYNesty v1.0.



**Figure 11.** Example corner plot of the eccentricity and argument of periastron posteriors from an emcee run (left). Although the distributions are correct, these are by nature both non-Gaussian and highly correlated, making it difficult to parameterize the results for inclusion in a publication, for example. These posteriors can be propagated through the respective constraints (see Section 3.1) to instead expose the posteriors in  $e \sin \omega$  and  $e \cos \omega$ , which are more orthogonal and can often conveniently be converted to Gaussian posteriors.

Since DYNesty samples directly from the priors, PHOEBE excludes the priors from the log likelihood. Instead, PHOEBE makes use of DISTL (see Section 3.2) to transform the defined priors into prior transforms that map a random value between 0 and 1 onto the drawn values of each individual parameter and passes these to DYNesty. It is important to note that this transformation requires ignoring any possible covariances between parameters in the priors (i.e., if priors were provided as multivariate distributions, they would be first flattened into univariate distributions before being passed to DYNesty).

After calling DYNesty from within PHOEBE, the user has access to all of the output arrays from DYNesty itself to access or plot any diagnostic figures. The samples from DYNesty can then be transformed into posterior distributions by accounting for their respective weights.

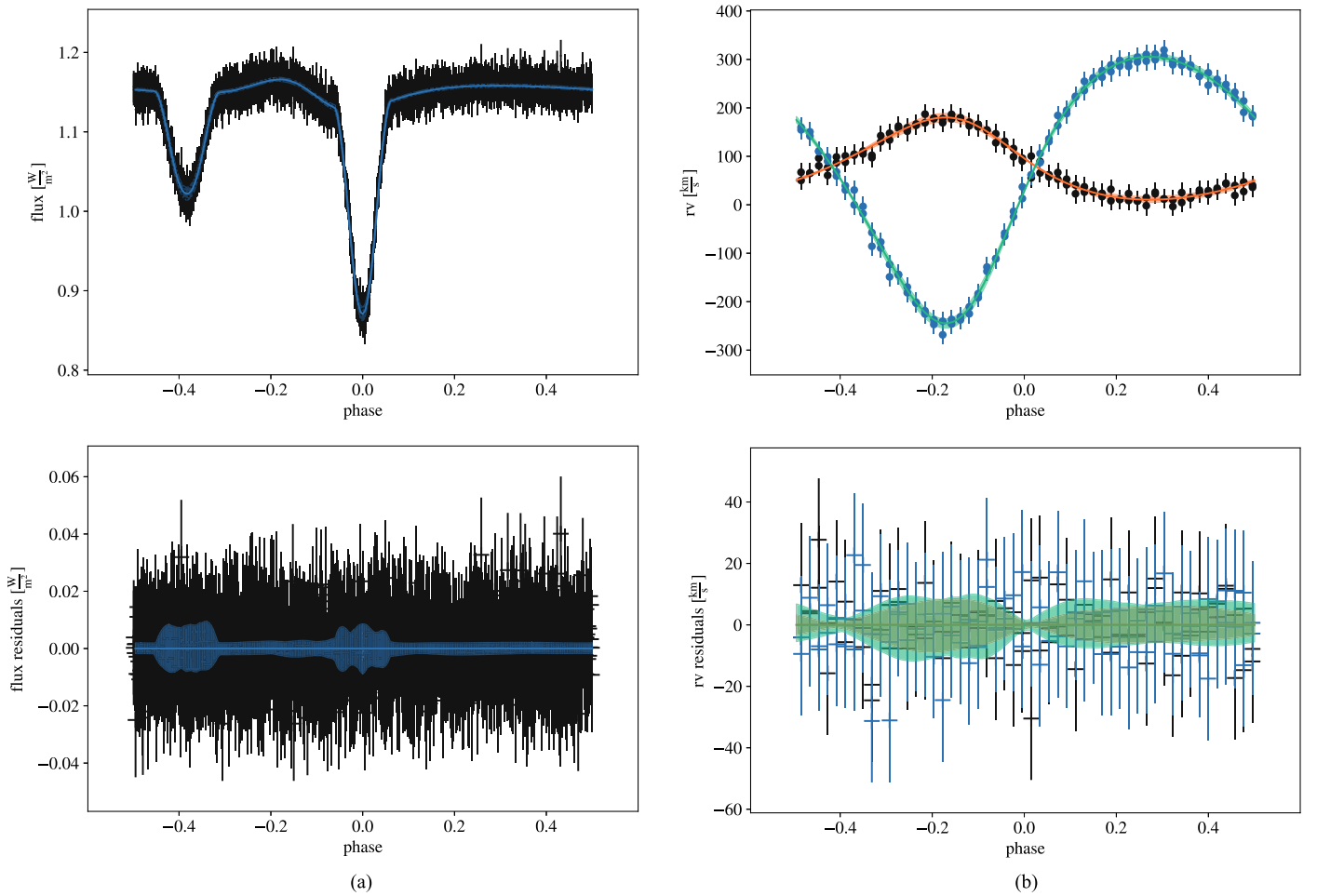
## 6. Posteriors and Uncertainties

The results from any sampler can be accessed as a posterior distribution, representing not only the uncertainties on each of the sampled parameter values but also the correlations (or covariances) between all of the sampled parameters. When these distributions are sufficiently Gaussian, the posteriors can easily be converted into a multivariate Gaussian distribution that represents the posteriors as just the means and a covariance matrix. Figure 10 shows an example corner plot of the posteriors of several parameters obtained from EMCEE samples and their conversion into a multivariate Gaussian.

As these distributions are DISTL distribution objects, they can easily be manipulated into any of the supported distribution types, saved to a file, plotted, have symmetric or asymmetric uncertainties exposed to include in published results, or used

for further sampling. For example, posteriors from one sampling run can be adopted and used as either the initial sampling distribution or priors for another sampling run. This can be useful in optimizing and determining posteriors for one data set, but maintaining that information in the log likelihood for another data set or when new observations become available. As with distributions used for priors (Section 3.2), these posteriors can be propagated through the “constraints” and exposed in any desired parameterization (Section 3.1). For example, Figure 11 shows the EMCEE samples for eccentricity and argument of periastron, which can then be propagated to  $e \sin \omega_0$  and  $e \cos \omega_0$ , which are easier to represent as a multivariate Gaussian and report in published scientific results. This allows for independent choices for the combination of parameters for sampling, priors, and posteriors, giving the full flexibility needed based on the available observations and known information.

It is important to note that the degeneracies and uncertainties determined by any sampler are computed under the assumption that all remaining parameters are held fixed at their face values with no uncertainties, leading to uncertainty underestimation or possibly even an incorrect solution. For example, if the value of the orbital period is believed to be well known and therefore left fixed during sampling, the resulting posteriors—and therefore uncertainties—on all other sampled parameters will exclude any degeneracies with respect to the orbital period, and will therefore likely be underestimated. Likewise, in the case where the assumed fixed value is incorrect, the remaining sampled parameters may be incorrect as well. In effect, all parameters that are not sampled are treated as if they have a delta function for a prior and are known to infinite precision. In a perfect world, all parameters would be sampled or



**Figure 12.** Any distribution (in this case the posteriors from the emcee run) can be propagated through the forward model. Here the  $3\sigma$  uncertainties on the light curve (left) and radial-velocity curves (right) and their residuals (bottom) are shown, depicting that the spread in the observations, given their uncertainties, is well represented by the spread in the model caused by the uncertainties in the posteriors.

marginalized over, but in practice that is not feasible with tens or even hundreds of possible parameters.

Lastly, as with any distribution, PHOEBE now allows propagation of posteriors through the forward model itself. By running the forward model over a number of samples from the posterior distributions, PHOEBE can compute and expose these individual models or the median model and  $1\sigma$ ,  $3\sigma$ , or  $5\sigma$  spreads in flux or radial velocity. Figure 12 shows an example of the  $3\sigma$  spread in the synthetic model when drawn from the posteriors from an EMCEE run.

## 7. User Interface

The PHOEBE 2.3 release also introduces a user interface, capable of all of the functionality of both the forward model and inverse problem described above, including basic plotting functionality, and it is available for download at <http://phoebe-project.org/clients>. The user interface is designed with a server–client model, splitting the code base into three distinct roles: the Python package itself, the user interface client, and a lightweight web server that can run on either the same machine or remotely from the client and is responsible for running commands sent from the user interface client via the Python package. This design allows for the flexibility of installing the server on a remote machine with more resources and either

installing the user interface locally or accessing a hosted version through the web browser.<sup>15</sup>

With or without the user interface, PHOEBE also allows the user to conveniently export the forward-model or inverse-problem jobs into a standalone script, which can then be run on a high-performance cluster, and the resulting file can be imported locally to view the progress or final results. This allows for running all interactive and visual components of the fitting process on a local machine while offloading computationally expensive tasks to a remote machine that may not be capable of hosting a web server or plotting graphics.

## 8. Conclusion

The 2.3 release of PHOEBE builds on past releases and introduces a general framework for data fitting running several common algorithms to address the inverse problem, and it has an interface for defining and dealing with complex distributions. It also introduces the ability to run the forward model through several other publicly available codes in addition to PHOEBE and a web-based user interface.

Although far from a black-box automated fitting pipeline, this common interface aims to ease the learning curve and

<sup>15</sup> A publicly available version is currently hosted at <http://ui.phoebe-project.org>.

effort required to employ multiple algorithms and codes while attempting to obtain both accurate and precise parameter values and uncertainties for eclipsing binary systems. As additional algorithms and forward models continue to be adopted by the field, we plan to incorporate them into PHOEBE within this same framework.

PHOEBE is an open-source project under the GPL3 license and is hosted at <http://phoebe-project.org> and <https://github.com/phoebe-project/phoebe2>. Contributions and feedback are welcome.

The development of PHOEBE is possible through the NSF AAG grants 1517474 and 1909109 and NASA 17-ADAP17-68, which we gratefully acknowledge.

We thank John Southworth and Pierre Maxted for their permission and discussions regarding their codes, JKTEBOP and ELLC.

D.J. acknowledges support from the State Research Agency (AEI) of the Spanish Ministry of Science, Innovation and Universities (MCIU) and the European Regional Development Fund (FEDER) under grant AYA2017-83383-P. D.J. also acknowledges support under grant P/308614 financed by funds transferred from the Spanish Ministry of Science, Innovation and Universities, charged to the General State Budgets and with funds transferred from the General Budgets of the Autonomous Community of the Canary Islands by the Ministry of Economy, Industry, Trade and Knowledge.

K.H. gratefully acknowledges support from NASA ADAP grant 18-ADAP18-228.

D.R.H. gratefully acknowledges the support of the Australian Government Research Training Program.

M.A. acknowledges support from the FWO-Odysseus program under project G0F8H6N.










*Software:* PHOEBE (Prša et al. 2016; Horvat et al. 2018; Jones et al. 2020), distl (<https://github.com/kecnry/distl>), EBAI (Prša et al. 2008, 2019), ellc (Maxted 2016), jktebop (Southworth et al. 2004, 2007, 2009; Southworth 2011), celerite (Foreman-Mackey et al. 2017), emcee (Foreman-Mackey et al. 2013, 2019), dynesty (Speagle 2020), schwimmbad (Price-Whelan & Foreman-Mackey 2017), numpy (Oliphant 2006), scipy (Virtanen et al. 2020), astropy (Astropy Collaboration et al. 2013), matplotlib (Hunter 2007), corner (Foreman-Mackey 2016).

## Appendix Description of Symbols

1.  $a_{\text{orb}}$ : orbital semimajor axis.
2.  $a_{\text{orb}} \sin i$ : orbital semimajor axis projected along the line of sight.
3.  $a_{\text{comp}} \sin i$ : component semimajor axis projected along the line of sight.
4.  $A_v$ ,  $R_v$ ,  $E(B - V)$ : extinction values.
5.  $e$ : orbital eccentricity.
6.  $f_{\text{orb}}$ ,  $f_{\text{rot}}$ : orbital and rotational frequencies, respectively.
7.  $i$ : orbital inclination.
8.  $P_{\text{orb}}$ ,  $\dot{P}_{\text{orb}}$ : orbital period at time  $t_0$  and its time derivative, respectively.
9.  $q$ : mass ratio defined as  $M_2/M_1$ .
10.  $R_{\text{equiv}}$ : stellar equivalent (volumetric) radius.
11.  $t_{0,\text{supconj}}$ : reference time of superior conjunction.

12.  $T_{\text{eff}}$ : stellar effective temperature.
13.  $v_\gamma$ : systemic velocity.
14.  $\omega_0$ ,  $\dot{\omega}$ : argument of periastron at time  $t_0$  and its time derivative, respectively.
15.  $\Omega_{\text{orb}}$ : longitude of the ascending node.

## ORCID iDs

Kyle E. Conroy  <https://orcid.org/0000-0002-5442-8550>  
 Angela Kochoska  <https://orcid.org/0000-0002-9739-8371>  
 Daniel Hey  <https://orcid.org/0000-0003-3244-5357>  
 Herbert Pablo  <https://orcid.org/0000-0002-1355-5860>  
 Kelly M. Hambleton  <https://orcid.org/0000-0001-5473-856X>  
 David Jones  <https://orcid.org/0000-0003-3947-5946>  
 Joseph Giammarco  <https://orcid.org/0000-0003-4560-7925>  
 Michael Abdul-Masih  <https://orcid.org/0000-0001-6566-7568>  
 Andrej Prša  <https://orcid.org/0000-0002-1913-0281>

## References

- Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, *A&A*, **558**, A33
- Foreman-Mackey, D. 2016, *JOSS*, **1**, 24
- Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. 2017, *AJ*, **154**, 220
- Foreman-Mackey, D., Farr, W., Sinha, M., et al. 2019, *JOSS*, **4**, 1864
- Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, *PASP*, **125**, 306
- Gao, F., & Han, L. 2012, *Comput. Optimization Appl.*, **259**
- Goodman, J., & Weare, J. 2010, *Commun. Appl. Math. Comput. Sci.*, **5**, 65
- Harmanec, P., Božić, H., Percy, J. R., et al. 2002, *A&A*, **387**, 580
- Hogg, D. W., Bovy, J., & Lang, D. 2010, arXiv:1008.4686
- Horvat, M., Conroy, K. E., Pablo, H., et al. 2018, *ApJS*, **237**, 26
- Hunter, J. D. 2007, *CSE*, **9**, 90
- Jenkins, J. M. 2017, Kepler Data Processing Handbook: Overview of the Science Operations Center, Kepler Science Document KSCI-19081-002, (Baltimore, MD: STScI)
- Jones, D., Conroy, K. E., Horvat, M., et al. 2020, *ApJS*, **247**, 63
- Maxted, P. F. L. 2016, *A&A*, **591**, A111
- Mowlavi, N., Lecoœur-Taïbi, I., Holl, B., et al. 2017, *A&A*, **606**, A92
- Oliphant, T. E. 2006, A Guide to NumPy, Vol. 1
- Price-Whelan, A. M., & Foreman-Mackey, D. 2017, *JOSS*, **2**, 357
- Prša, A. 2018, Modeling and Analysis of Eclipsing Binary Stars: The Theory and Design Principles of PHOEBE (Beograd: Institute of Physics)
- Prša, A., Conroy, K. E., Horvat, M., et al. 2016, *ApJS*, **227**, 29
- Prša, A., Guinan, E. F., Devinney, E. J., et al. 2008, *ApJ*, **687**, 542
- Prša, A., Guinan, E. F., Devinney, E. J., et al. 2019, EBAI: Eclipsing Binaries with Artificial Intelligence, Astrophysics Source Code Library, ascl:1908.018
- Prša, A., & Zwitter, T. 2005, *ApJ*, **628**, 426
- Schwarz, G. 1978, *AnSta*, **6**, 461
- Shenar, T., Hainich, R., Todt, H., et al. 2018, *A&A*, **616**, A103
- Southworth, J. 2011, *MNRAS*, **417**, 2166
- Southworth, J., Bruntt, H., & Buzasi, D. L. 2007, *A&A*, **467**, 1215
- Southworth, J., Hinse, T. C., Dominik, M., et al. 2009, *ApJ*, **707**, 167
- Southworth, J., Maxted, P. F. L., & Smalley, B. 2004, *MNRAS*, **351**, 1277
- Speagle, J. S. 2020, *MNRAS*, **493**, 3132
- Torres, G., Andersen, J., & Giménez, A. 2010, *A&ARv*, **18**, 67
- Underhill, A. B., & Hill, G. M. 1994, *ApJ*, **432**, 770
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, *Nature Methods*, **17**, 261
- Wilson, R. E. 1979, *ApJ*, **234**, 1054
- Wilson, R. E. 2008, *ApJ*, **672**, 575
- Wilson, R. E., & Devinney, E. J. 1971, *ApJ*, **166**, 605
- Wilson, R. E., & Van Hamme, W. 2014, *ApJ*, **780**, 151