

## Hardware-Accelerated Computation of Stellar Light Curves

LIONEL J. GARCIA,<sup>1</sup> SOICHIRO HATTORI,<sup>1</sup> AND DANIEL FOREMAN-MACKEY<sup>1</sup>

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

### ABSTRACT

The measurement of stellar fluxes over time serves a wide variety of science cases. Transit and occultation light curves, for example, are used to characterize the orbital parameters of stellar systems, and infer the atmospheric properties of their transiting bodies. Phase curves, on the other hand, can be used to study the thermal emission of exoplanets, and the atmospheric properties of stars. In order to infer these properties, a fast, accurate and robust model of stellar light curves is needed, one that accounts for the non-uniform surface intensity of spherical bodies. The analytical framework known as **starry** satisfies these requirements, and was successfully applied over the years following its development. However, as datasets and model complexity grow, the tractable inference of these parameters becomes increasingly challenging, and often motivates biased approximations. In this paper, we present a new implementation of the **starry** framework in **jax**, a high-performance machine-learning library designed for large-scale computations. We describe the changes made to the original version of **starry** and evaluate the performance of its new implementation that we release in the open-source Python package **jaxoplanet**. Through this implementation, we provide the community with a differentiable model of stellar light curves, and enable the use of powerful machine-learning tools available in the **jax** ecosystem.

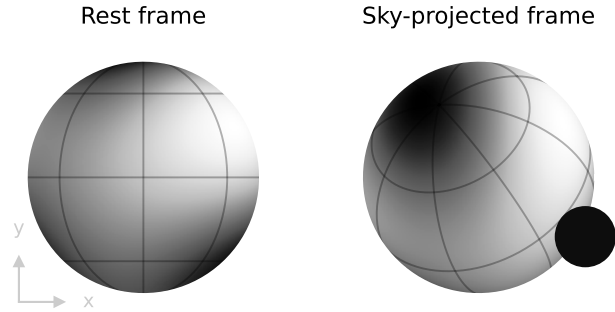
### INTRODUCTION

## 1. STARRY

## 2. OPTIMIZATION

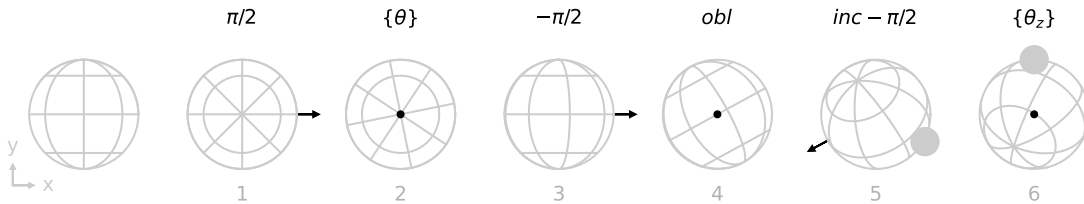
2.1. *Limb-darkening multiplicative maps*2.2. *Spherical harmonics rotations*

With *starry*, every light curve computation at a given time  $t$  involves a rotation of the spherical harmonics basis, from the rest-frame of the star (Figure 1, left) to its sky-projected orientation (Figure 1, right), with a final rotation relative to the position of the occulting body (see Figure 2. from Luger et al. 2019).



**Figure 1.** Rotation of the spherical harmonics basis to place the star in the sky-projected frame of the observer.

The rotation of spherical harmonics involves the computation of Wigner-D-matrices, commonly obtained through robust recursion relations in both degree  $l$  and order  $m$ , but leading to costly computations. Hence, knowing how to decompose the full spherical harmonics rotation using pre-computed rotations is essential to achieve optimal performance in the *starry* framework. Figure 2 shows the 6 elementary rotations currently used in the *starry* implementation.



**Figure 2.** Consecutive rotations of the spherical harmonics basis in the *starry* implementation. Each schematic shows a sphere rotated from a previous orientation around an axis displayed as a black vector and an angle shown on top.

In this figure, rotations 1 to 3 correspond to a rotation of the star around its rotation axis, and rotations 3 to 5 place the star on a sky-projected frame. Finally, rotation

6 aligns the spherical harmonics map to the occulting body on the  $y$  axis in order to apply Green’s theorem in the appropriate basis (Figure 2 from Luger et al. 2019). The first thing to notice is that rotations 1 to 3 could be simply reduced to a rotation of angle  $\theta$  about the  $y$ -axis, turning steps 1 to 3 into one. However, rotations around the pole, i.e. with a rotation axis along  $z$ , have simpler expressions that can be implemented separately. In addition, pre-computing these matrices at lower cost is particularly important for steps 2 and 6, involving a potentially large set of angles  $\{\theta\}$  and  $\{\theta_z\}$ , defined over times  $\{t\}$ , for which to compute the full light curve. This explains the current decomposition of the complete rotation in six separate steps.

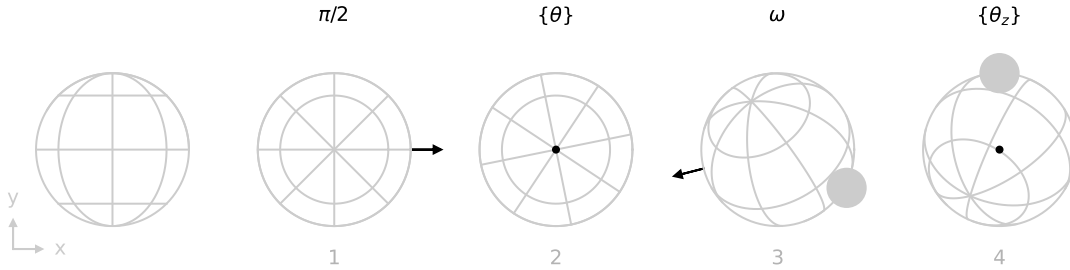
In `jaxoplanet`, we compute the Wigner-D-matrices by employing the Risbo recursion relations (Risbo 1996) implemented in `JAX` as part of the `s2fft` Python package (Price & McEwen 2023). In addition, we merge rotations 3, 4 and 5 (see Figure 3) into a single compound rotation of axis

$$\mathbf{v} = \frac{1}{\sqrt{1 - \cos^2\left(\frac{inc}{2}\right) \cos^2\left(\frac{obl}{2}\right)}} \begin{pmatrix} \sin\left(\frac{inc}{2}\right) \cos\left(\frac{obl}{2}\right) \\ \sin\left(\frac{inc}{2}\right) \sin\left(\frac{obl}{2}\right) \\ -\cos\left(\frac{inc}{2}\right) \sin\left(\frac{obl}{2}\right) \end{pmatrix}, \quad (1)$$

and angle

$$\omega = 2 \cos^{-1} \left( \cos\left(\frac{inc}{2}\right) \cos\left(\frac{obl}{2}\right) \right). \quad (2)$$

This way, the complete rotation reduces to the four separate steps shown in Figure 3.



**Figure 3.** Consecutive rotations of the spherical harmonics basis in the `jaxoplanet` implementation. Each schematic shows a sphere rotated from a previous orientation around an axis displayed as a black vector and an angle shown on top.

### 2.3. Computation of the solution vectors

### 2.4. Autodiff?

## 3. PERFORMANCE

## 4. CASE STUDY: ...

## REFERENCES

- |   |   |
|---|---|
| <p>Luger, R., Agol, E., Foreman-Mackey, D., et al. 2019, <i>The Astronomical Journal</i>, 157, 64, doi: <a href="https://doi.org/10.3847/1538-3881/aae8e5">10.3847/1538-3881/aae8e5</a></p> | <p>Price, M. A., &amp; McEwen, J. D. 2023, <i>Journal of Computational Physics</i>, submitted</p> |
|---|---|

Risbo, T. 1996, Journal of Geodesy, 70,  
383, doi: [10.1007/BF01090814](https://doi.org/10.1007/BF01090814)