Draft version September 27, 2024
Typeset using LaTeX **modern** style in AASTeX631

jaxoplanet: Hardware-Accelerated Computation of Stellar Light Curves

Lionel J. Garcia,[1] Soichiro Hattori,[1] and Daniel Foreman-Mackey[1]

[1] *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 circulation 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 and their mutual occultations. 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. LIGHT CURVE MODELS

Inspired by the idea of Pál (2012), Luger et al. (2019) and Agol et al. (2020) present a general framework to compute the light curves of spherical bodies with non-uniform surface intensities. This framework, known as *starry*, allows for precise and efficient computation of the light curves of stars and exoplanets, and has been successfully applied to a wide variety of science cases. In this section, we describe the mathematical formalism behind the *starry* model.

1.1. *Spherical harmonics*

The surface of a spherical body can be naturally described by spherical harmonics: a set of special functions defined on the sphere that can serve as orthonormal basis to represent any spherical surface map. If $\tilde{\mathbf{y}}$ denotes the spherical harmonics basis of degree l_{max} (defined in Luger et al. 2019, section X) and $\mathbf{y}$ a vector that contains the coefficients describing the surface in this basis, the intensity I of the map projected onto the sky-projected (x, y) plane of the observer is

$$I(x,y) = \tilde{\mathbf{y}}^\mathsf{T}(x,y)\,\mathbf{y}. \tag{1}$$

In order to orient this surface, for example with a given inclination, obliquity and phase around its rotation axis, we define a rotation matrix $\mathbf{R}$ that acts on the spherical harmonics coefficients $\mathbf{y}$, such that the intensity of the map projected onto the sky of the observer is

$$I(x,y) = \tilde{\mathbf{y}}^\mathsf{T}(x,y)\,\mathbf{R}\,\mathbf{y}. \tag{2}$$

1.2. *Rotation light curve*

Using this expression, the integrated intensity of the rotated map is

$$F = \iint I(x,y)\,\mathrm{d}S = \iint \tilde{\mathbf{y}}^\mathsf{T}\mathbf{R}\mathbf{y}\,\mathrm{d}S \tag{3}$$

In order to simplify this integral, Luger et al. 2019 introduce a change of basis from the spherical harmonics basis $\tilde{\mathbf{y}}$ to a simple polynomial basis $\tilde{\mathbf{p}}$ characterized by the change of basis matrix $\mathbf{A_1}$. In this new basis, $\mathbf{y}$ can be expressed as

$$\mathbf{p} = \mathbf{A_1}\mathbf{y}, \tag{4}$$

such that

$$I(x,y) = \tilde{\mathbf{p}}^\mathsf{T}(x,y)\,\mathbf{A_1}\mathbf{R}\mathbf{y}. \tag{5}$$

Hence,

$$F = \iint \tilde{\mathbf{p}}^\mathsf{T}\mathbf{A_1}\mathbf{R}\mathbf{y}\,\mathrm{d}S = \mathbf{r}^\mathsf{T}\mathbf{A_1}\mathbf{R}\mathbf{y}, \tag{6}$$

with

$$\mathbf{r}^{\mathsf{T}} \equiv \iint \tilde{\mathbf{p}}^{\top} \,\mathrm{d}S, \tag{7}$$

since $\mathbf{A_1}$, $\mathbf{R}$ and $\mathbf{y}$ are independent of the coordinates on the sphere.

As $\tilde{\mathbf{p}}$ is simply made of polynomials of the coordinates x, y and z, the elements of the *rotation vector* $\mathbf{r}$ can be computed analytically and only once for a given l_{max} (see Luger et al. 2019, Eq. 20).

1.3. *Occultation light curve*

The same formalism can be used to compute the integrated intensity of a surface map occulted by another spherical body. In this case, the intensity of the map must be integrated only over the unocculted surface S of the disk, such that

$$F = \iint_S I(x,y)\,\mathrm{d}S = \left(\iint_S \tilde{\mathbf{p}}^{\mathsf{T}} \,\mathrm{d}S\right) \mathbf{A_1}\mathbf{R}\mathbf{y}. \tag{8}$$

To simplify this expression, Pál 2012 note that any two-dimensional integral of the unocculted disk can be reduced to a one-dimensional integral over the contour C of the occulting body using Green's theorem[1]. This can be done if the integrand of the integral over S can be described by a function f such that

$$f(x,y) = \frac{\delta G_y}{\delta x} - \frac{\delta G_x}{\delta y}, \tag{9}$$

where G is a vector field defined and differentiable over S.

By applying a last change of basis matrix $\mathbf{A_2}$ to the polynomial basis $\tilde{\mathbf{p}}$, Luger et al. 2019 transforms $\tilde{\mathbf{p}}$ to the *Green's basis* $\tilde{\boldsymbol{g}}$ that satisfies this property, meaning that the surface integral of each of its components $\tilde{g_n}$ can be transformed to the line integral

$$s_n \equiv \iint_S \tilde{g_n}(x,y)\,\mathrm{d}S = \oint_C \mathbf{G}_n(x,y)\,\mathrm{d}\mathbf{r}, \tag{10}$$

with $\mathbf{G}_n$ given in Luger et al. 2019, Equation 34. These functions require the occulting body to be aligned with the y-axis of the reference frame, which can be achieved by applying a final rotation matrix $\mathbf{R}'$ to the surface map.

Using these expressions, the integrated intensity of the occulted map follows the idiomatic linear expression

$$F = \mathbf{s}^{\mathsf{T}}\mathbf{A}\,\mathbf{R}'\,\mathbf{R}\,\mathbf{y}, \tag{11}$$

[1] Also known as Stokes' theorem in three-dimensional vector calculus.

with $\mathbf{A} = \mathbf{A_2}\mathbf{A_1}$ and the integrals s_n arranged in the *solution vector* $\mathbf{s}^\mathsf{T}$.

1.4. *Limb-darkened surfaces*

We notice that a simpler situation is encountered when the surface of the spherical body can be described by a radial intensity profile $I(r)$, such as a limb-darkened star. In this case, the vector of spherical harmonics coefficients $\mathbf{y}$ is very sparse (because all $m \neq 0$ components are zero) and the surface map doesn't need to be rotated. Contemporary to Luger et al. (2019), Agol et al. (2020) explore this case and provide simple expressions for the integrated intensity of a star whose surface can be described by a polynomial limb darkening law occulted by an opaque body. In this case:

1. The surface is simply described by $\mathbf{u}$, a vector of polynomial limb darkening coefficients.
2. The Green's basis takes a simpler form (see Agol et al. 2020, Eq. X).
3. Matrices involved in the computation of the integrated flux have l_{max} columns, versus $(l_{max}+1)^2$ in the general case, reducing drastically the computational cost of the model.

For these reasons, and because of its wide application, our implementation treats radially-symmetric maps separately, leading to highly optimize computations of limb-darkened light curves.

2. IMPLEMENTATION DETAILS

2.1. *Limb darkened maps*

Limb darkened maps are obtained by multiplying a base map, devoid of limb darkening, with a pure limb darkened map. If $\mathbf{U}$ is the change of basis matrix from the polynomial limb darkening coefficients $\mathbf{u}$ to the spherical harmonics basis, the pure limb darkened map is

$$\mathbf{y}_u = \mathbf{U}\mathbf{u}. \tag{12}$$

The multiplication of the base map represented by $\mathbf{y}$ with the limb darkening map represented by $\mathbf{y}_u$ can be seen as the multiplication of two polynomial functions, only with special coefficients (called Clebsch–Gordan coefficients) inherent to the properties of spherical harmonics. To simplify this operation, we transform both maps into the polynomial basis, so that their product is the actual product of two polynomials.

2.2. *Limb darkening normalization*

In starry version 0.2.2 (Luger et al. 2019), the normalization of a limb-darkened light curve was done by assuming that adding limb darkening to a non-uniform surface does not modify its total luminosity. Hence, the total flux received from a star during its complete rotation should be the same with or without limb darkening. In older versions of starry ($\leq$0.2.2), this was done by actually computing the light curve of the star with and without limb darkening, and using the ratio of the two to normalize the limb-darkened light curve. As discussed in the starry documentation[2] this approach is wrong and leads to an unphysical normalization.

Instead, one can assume that the total flux of the star should be similar for all observers, independently of the surface map of the star. By noticing that all components of the spherical harmonics basis integrate to 0 (except the constant component $Y_{0,0}$) we deduce that the mean flux of the star must be coming from the surface feature seen by any observer, no matter his viewpoint, which corresponds to a purely limb-darkened surface (with no other source of non-uniformity).

The flux of this purely limb-darkened surface is

$$f_u = \mathbf{r}^\mathsf{T}\mathbf{A_1}\mathbf{y}_u \tag{13}$$

with $\mathbf{y}_u$ defined in Equation 12. Hence, we normalize limb-darkened maps by π/f_u, so that the total flux for a uniform map with no limb darkening is unchanged.

2.3. *Spherical harmonics rotations*

[2] https://starry.readthedocs.io/en/latest/notebooks/LDNormalization/

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).

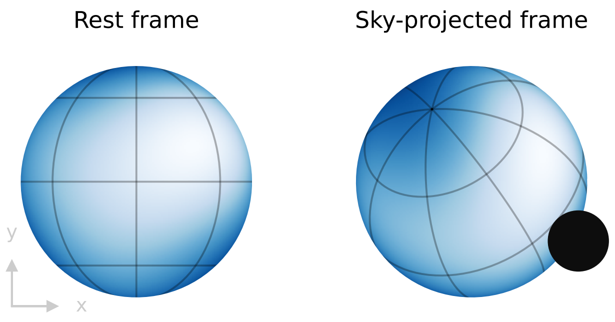

Figure 1. Rotation of the spherical harmonics basis to place the star in the sky-projected frame of the observer.

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.

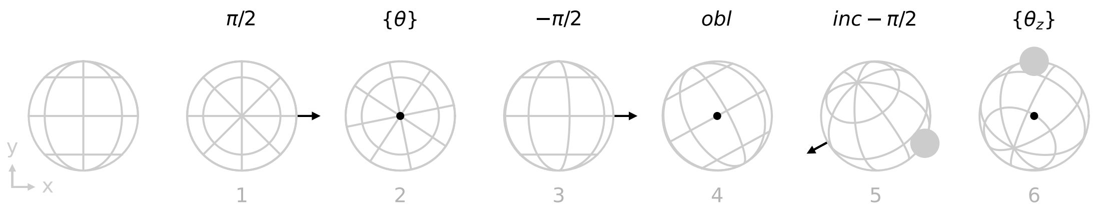

Figure 2. Consecutive rotations of the spherical harmonics basis in the starry implementation. Each schematic shows a sphere rotated from a previous orientation around an axis displayed as a black vector and an angle shown on top.

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 θ 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

$$\mathbf{v} = \frac{1}{\sqrt{1-\cos^2\left(\frac{inc}{2}\right)\cos^2\left(\frac{obl}{2}\right)}}\begin{pmatrix} \sin\left(\frac{inc}{2}\right)\cos\left(\frac{obl}{2}\right) \\ \sin\left(\frac{inc}{2}\right)\sin\left(\frac{obl}{2}\right) \\ -\cos\left(\frac{inc}{2}\right)\sin\left(\frac{obl}{2}\right) \end{pmatrix}, \tag{14}$$

and angle

$$\omega = 2\cos^{-1}\left(\cos\left(\frac{inc}{2}\right)\cos\left(\frac{obl}{2}\right)\right). \tag{15}$$

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

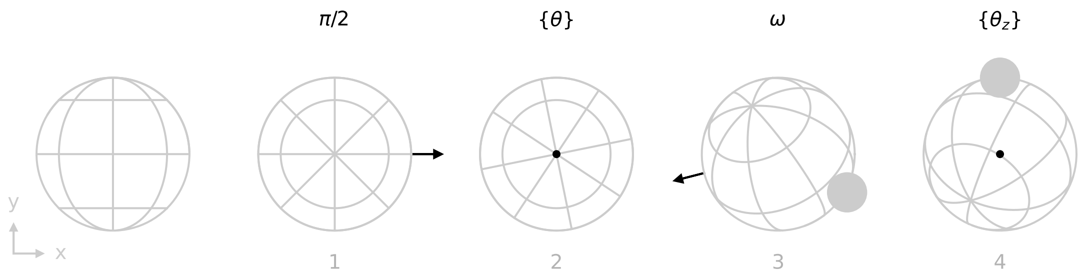

Figure 3. Consecutive rotations of the spherical harmonics basis in the jaxoplanet implementation. Each schematic shows a sphere rotated from a previous orientation around an axis displayed as a black vector and an angle shown on top.

2.4. *Computation of the solution vectors*

We described in section 1.3 how the integrated intensity of a spherical body occulted by another can be computed using the solution vector $\mathbf{s}^\mathsf{T}$. Pál (2012) emphasizes that, knowing $\mathbf{G}_n$, each component s_n of the solution vector defined in Equation 10 can be decomposed into

$$s_n = \mathcal{Q}(\mathbf{G}_n) - \mathcal{P}(\mathbf{G}_n) \tag{16}$$

where $\mathcal{Q}(\mathbf{G}_n)$ and $\mathcal{P}(\mathbf{G}_n)$ are line integrals over the contours of the occulting body and the primary's surface, as shown in Figure 4.

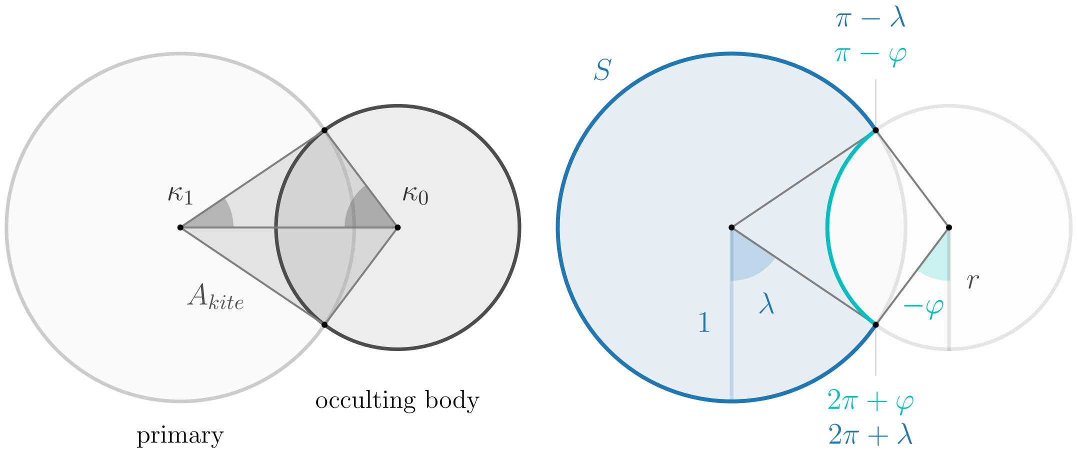

Figure 4. Geometric configuration and angles definitions for the two-body occultation. On the left figure, the larger light grey disk corresponds to the primary body being occulted by the darker secondary body. On the right figure, the area S on which the intensity is integrated is shown in light blue. By choosing an appropriate transform, the integration over S can be reduced to a line integral over the contour of the occulting body, which can be decomposed into the contour of the occulting body within the area of the primary body (in cyan) and the contour of the primary body everywhere else (in blue).

In the general case, the $\mathcal{Q}$ integral takes a simple analytical form and can be computed using the recurrence relations from Luger et al. (2019) (Equation D29). The case of a limb-darkened surface is even simpler as $\mathcal{Q}(\mathbf{G}_n) = 0$ for all $n > 0$. On the other hand, the $\mathcal{P}$ integral is harder to evaluate and requires the use of recurrence relations and truncated series expansions, with a scheme that depends on the geometric configuration of the system (see Luger et al. 2019, Section D.2).

In order to translate the starry implementation to JAX, we evaluate the $\mathcal{P}$ integrals numerically using Gauss-Legendre integrations, requiring the evaluation of the integrated functions only over a fixed number of points.

2.4.1. *Polynomial limb darkening*

For a surface described by a polynomial limb darkening law of order n, the $\mathcal{P}$ integral is given in Agol et al. (2020) by

$$\mathcal{P}(\mathbf{G}_n) = \begin{cases} \pi - \kappa_1 - r^2\kappa_0 + A_{kite} & n = 0 \\ 2s_0 + 4\pi\eta - 2\pi & n = 2 \\ 2r(4br)^{\frac{n}{2}} \displaystyle\int_{-\kappa_0/2}^{\kappa_0/2} (k^2 - \sin^2\varphi')^{\frac{n}{2}} (r - b + 2b\sin^2\varphi')\,\mathrm{d}\varphi' & \mathrm{n} \geq 3, \end{cases}$$

where κ_0, κ_1 and A_{kite} are the angles and area shown in Figure 4 and computed following Agol et al. (2020) (Equations 31 and 32). For $n = 2$, η is defined in Agol et al. (2020) (Equation 53) and depends on the elliptic parameter

$$k = \frac{1 - r^2 - b^2 + 2br}{4br}. \tag{17}$$

For $n = 1$, we use Equation 31, 32 and 34 from Luger et al. (2019) to write

$$\mathcal{P}(\mathbf{G}_1) = \int_{-\kappa_0/2}^{\kappa_0/2} r(r - b\cos 2\varphi')\,\frac{1 - z^3}{3(1 - z^2)}\mathrm{d}\varphi' \quad \text{and} \quad \mathcal{Q}(\mathbf{G}_1) = \frac{2}{3}(\pi - \kappa_1), \tag{18}$$

where $1 - z^2 = b + r - 2\cos\varphi'$ [3].

2.4.2. *Non-uniform surface*

In the general case, for a star with a non-radially symmetric surface intensity, n denotes the index of the spherical harmonics component such that

$$n = l^2 + l + m, \tag{19}$$

where l is the degree of the spherical harmonic and m its order. For each component, the expression of $\mathcal{P}(\mathbf{G}_n)$ depends on μ and ν defined as

$$\begin{aligned} \mu &\equiv l - m, \\ \nu &\equiv l + m. \end{aligned} \tag{20}$$

[3] note that φ' is different from the angle φ shown in Figure 4 and was introduced through the change of variable $\varphi' = \frac{1}{2}(\varphi - \frac{3\pi}{2})$.

The $\mathcal{P}$ integral is then given by Equation D32 from Luger et al. (2019) as

$$\mathcal{P}(\mathbf{G}_n) = \begin{cases} 2(2r)^{l+2} \int\limits_{-\kappa/2}^{\kappa/2} (s_{\varphi'}^2 - s_{\varphi'}^4)^{\frac{\mu+4}{4}} (\delta + s_{\varphi'}^2)^{\frac{\nu}{2}} \,\mathrm{d}\varphi' & \frac{\mu}{2} \text{ even} \\ \mathcal{F} \int\limits_{-\kappa/2}^{\kappa/2} (s_{\varphi'}^2 - s_{\varphi'}^4)^{\frac{l-2}{2}} (k^2 - s_{\varphi'}^2)^{\frac{3}{2}} (1 - 2s_{\varphi'}^2) \,\mathrm{d}\varphi' & \mu = 1,\, l \text{ even} \\ \mathcal{F} \int\limits_{-\kappa/2}^{\kappa/2} (s_{\varphi'}^2 - s_{\varphi'}^4)^{\frac{l-3}{2}} (\delta + s_{\varphi'}^2)(k^2 - s_{\varphi'}^2)^{\frac{3}{2}} (1 - 2s_{\varphi'}^2) \,\mathrm{d}\varphi' & \mu = 1,\, l \neq 1,\, l \text{ odd} \\ 2\mathcal{F} \int\limits_{-\kappa/2}^{\kappa/2} (s_{\varphi'}^2 - s_{\varphi'}^4)^{\frac{\mu-1}{4}} (\delta + s_{\varphi'}^2)^{\frac{\nu-1}{2}} (k^2 - s_{\varphi'}^2)^{\frac{3}{2}} \,\mathrm{d}\varphi' & \frac{\mu-1}{2} \text{ even},\, l \neq 1 \\ \int\limits_{-\kappa/2}^{\kappa/2} r(r - b\cos 2\varphi')\, \frac{1 - z^3}{3(1 - z^2)} \mathrm{d}\varphi' & \mu = 1,\, l = 1 \\ 0 & \text{otherwise,} \end{cases}$$

where $s_{\varphi'} = \sin\varphi'$, $\kappa = \varphi + \frac{\pi}{2}$ (see Luger et al. 2019, Equation D31) and δ is defined for numerical stability as

$$\delta = \frac{b - r}{2r}. \tag{21}$$

Note how, unlike in the starry implementation, the term $\mu = 1$, $l = 1$ is derived exactly like in the polynomial limb darkening case, instead of analytically which would require the evaluation of elliptic integrals.

Because of its numerical integration, the precision of the $\mathcal{P}$ integral is limited and depends on the order q of the Gauss-Legendre quadrature, which corresponds to the number of point for which the integrand is evaluated. To minimize this number, we note that all integrands are symmetrical functions, except for $\mu = 1$ and $l = 1$. Hence, all integrals can be computed over the interval $\left[0, \frac{\kappa}{2}\right]$, instead of $\left[-\frac{\kappa}{2}, \frac{\kappa}{2}\right]$, and multiplied by 2. We also note that all terms for which $\frac{\mu}{2}$ is even reach machine precision for an integration order as low as $q = 20$, that we hold fix. With these optimizations, we benchmark the precision and speed of our final implementation for the solution vector in section 3.

3. PERFORMANCE

Although *starry* occultation light curves have analytical expressions, their practical implementation relies on numerical approximations chosen to balance precision and computation time. This is the case, for example, of the solution vector $\mathbf{s}$, computed using numerical series expansions in starry (Luger et al. 2019, section D.2.3) and numerical integration in jaxoplanet (see **??**). Other expressions, such as the Wigner-D matrices used in the rotation of the spherical harmonics basis, are computed using reccurence relations prone to the accumulation of numerical errors and instability.

In this section, we evaluate the accuracy and speed of the jaxoplanet implementation and compare it with the C++ implementation of starry.

3.1. *Precision*

We evaluate the precision of the starry and jaxoplanet terms against quantities computed at arbitrary precision, using a ground truth version of the code implemented with the `mpmath` Python library[4]. The precision of the overall flux f can only be understood by considering the precision of the terms involved in **??**: the solution vector $\mathbf{s}$ ($\mathbf{r}$ if no occultation), the basis matrices $\mathbf{A}$ and $\mathbf{B}$, and the Wigner-D rotation tensor $\mathbf{R}$. Although we show in Figure 13 that the precision of the starry and jaxoplanet implentations differ for many of these terms, we focus this section on the solution vector $\mathbf{s}$ and the overall flux f.

Assuming that we perform the numerical integration of the solution vector $\mathbf{s}$ at a relatively high order ($n = 500$; see **??**), Figure 5 shows that our reimplentation of starry reaches relative errors lower than 1 part per billion for $\mathbf{s}$ and the overall flux f, considering both a small ($r = 0.01$) and a large ($r = 100$) occultor transiting the star accross numerically-sensitive values of impact parameters b. For reference, Figure 14 shows comparable errors on the same quantities computed with the C++ implentation of starry. In addition, Figure 6 shows the evolution of the error on f for increasing degrees of the spherical harmonics, both for starry and jaxoplanet. With these results, we validate the precision of jaxoplanet ans show it is on per with the legacy C++ starry implentation[5] described in Luger et al. (2019).

[4] https://mpmath.org/
[5] version 1.2.0 (Luger et al. 2021)

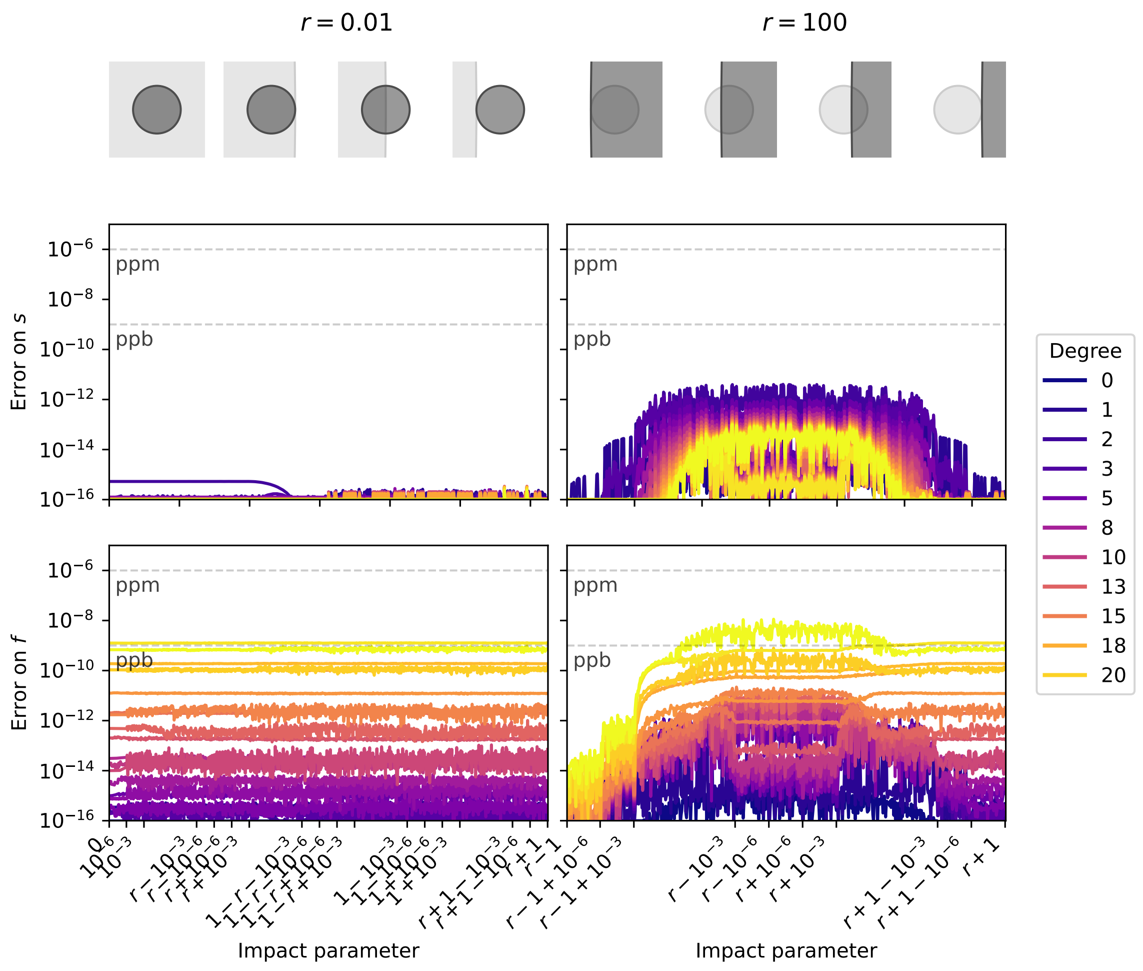

Figure 5. Error of the solution vector **s** and the overall flux f for all terms of the spherical harmonics basis up to degree $l = 20$. Errors are computed against arbitrary-precision computations of **s** and f for a small ($r = 0.01$) and a large ($r = 100$) occultor transiting the star accross numerically-sensitive values of impact parameters b. For each (l, m) basis component all coefficients are set to zero except $y_{l,m} = 1$ and errors are scaled to the highest values of **s** (respectively f) over b. Then, the plotted errors for each degree l are the maximum of the scaled errors over all $m \in [-l, l]$ components over b. The disks at the top of the figure show the transit configurations of the occultor (light dark) and the star (light gray) for different values of b.

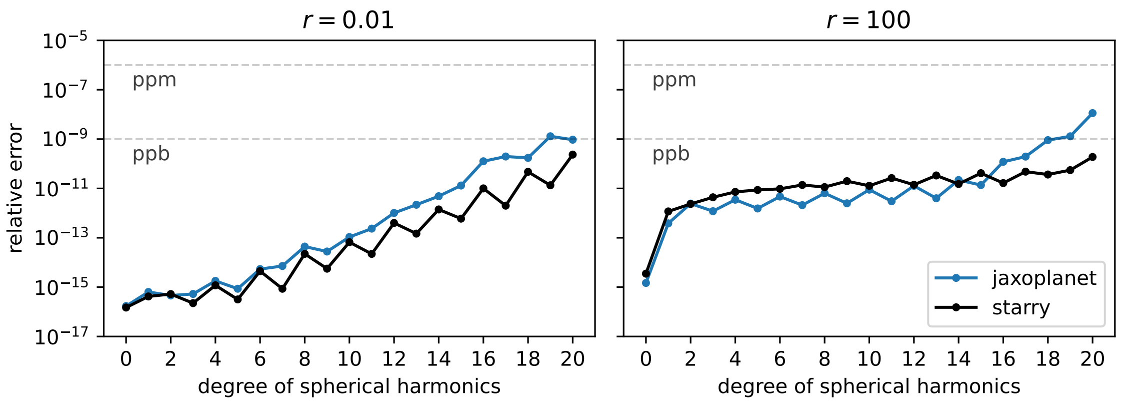

Figure 6. Maximum errors in the overall flux f for all degrees of the spherical harmonics components. Errors shown in this figure correspond to the maximum values of the scaled errors shown in Figure 5, maxima taken over the range of values b.

For reference, Figure 7 shows the relative error in the flux of a limb-darkened star occulted by an opaque body for increasing orders of a polynomial limb darkening law. In this figure, we also report the error of the flux computed using the exoplanet Python package (limited to linear and quadratic limb darkening).

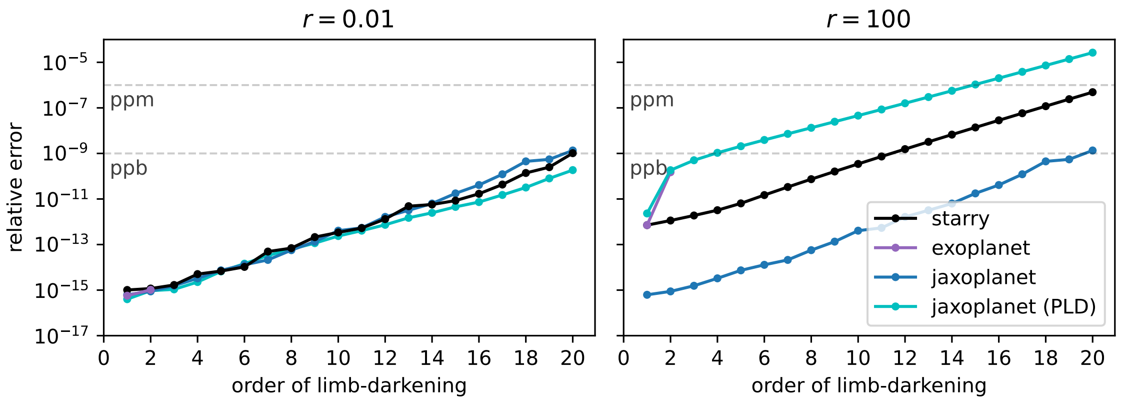

Figure 7. Maximum errors in the overall flux f of a star occulted by an opaque body for increasing orders of its polynomial limb darkening law. Errors shown in this figure correspond to the maximum values of the scaled errors shown in Figure 5, taken over the range of values b. For each order n all limb darkening coefficients are set to zero except the n-th coefficient $u_n = 1$.

A set of unitary polynomial limb darkening coefficients of order 20 leads to spherical harmonics coefficients as large as 10^6. This explains the larger errors observed in Figure 7 (error versus limb darkening order) compared to the ones shown in Figure 6 (error versus degree of the spherical harmonics). We not that the errors scale with the values of u.

As described in **??**, we compute *starry* light curves using the Gauss Legendre quadrature to approximate the $\mathcal{P}$ integral involved in the expression of the solution vector $\mathbf{s}$. Hence, the precision of the computed flux f depends on the order of the Legendre polynomial which defines the number of points used to approximate $\mathcal{P}$. For this reason, users must carefully set the order parameter to reach the precision required for their application, as shown in Figure 8.

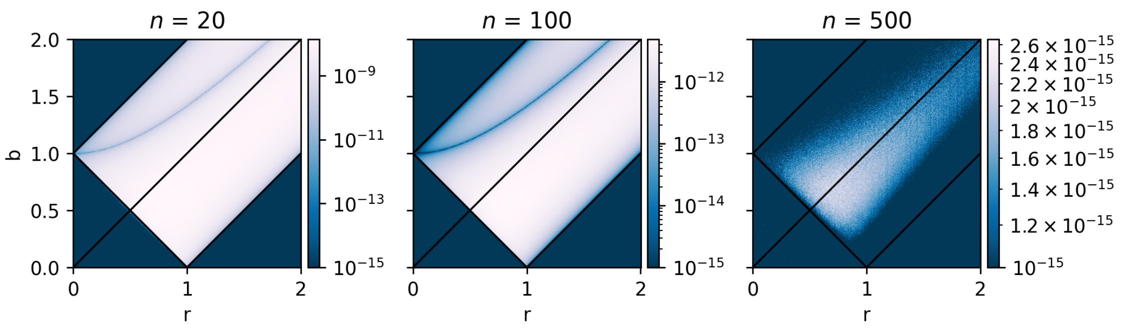

Figure 8. relative numerical error in the flux f of a linearly limb-darkened star (degree l=1) transited by an opaque companion over the (r, b) parameter space. The error is evaluated for different orders n of the Gauss-Legendre quadrature and computed against starry for speed and convenience. Note that Figure 5 was produced using $n = 500$ (right plot) for which $\mathbf{s}$ reaches machine precision.

In Figure 9, we show how the error in the solution vector evolves with the order n of the Gauss-Legendre quadrature, and its impact on the error in the overall flux.

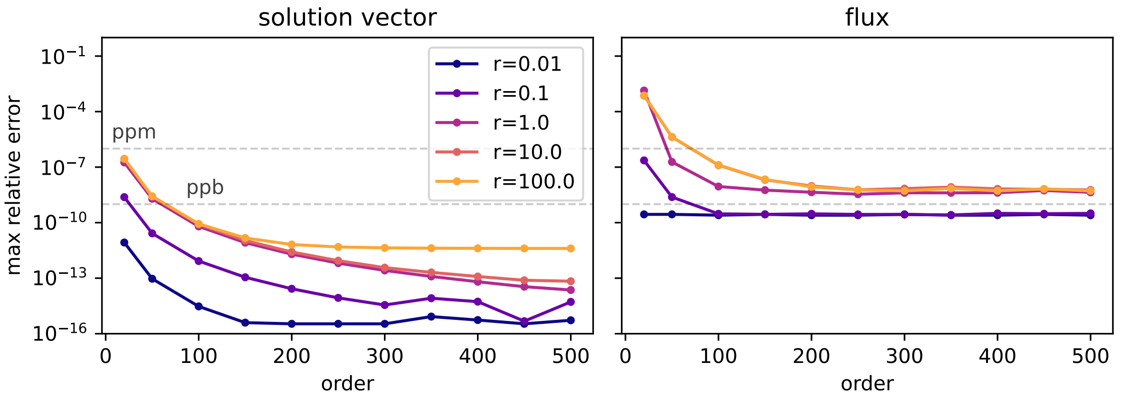

Figure 9. Maximum errors in the solution vector and overall flux f for different occultor radii and orders n of the Gauss-Legendre quadrature used to compute the P integral (see **??**). Errors shown in this figure correspond to the maximum values of the scaled errors shown in Figure 5, maxima taken over the range of values b. As in Figure 5 the maximum degree of the spherical harmonics basis is set to $l = 20$.

The minimum error is always achieved for higher values of n, which comes with higher computational cost (see section 3.2). Hence users must carefully set the order parameter to balance precision and computational speed. As the minimum error reached also depends on the degree of the spherical harmonics basis used, Figure 10 shows the errors on the flux for each degree of the spherical harmonics basis, for an occultor with relative radii $r = 0.01$, $r = 1$ and $r = 100$.

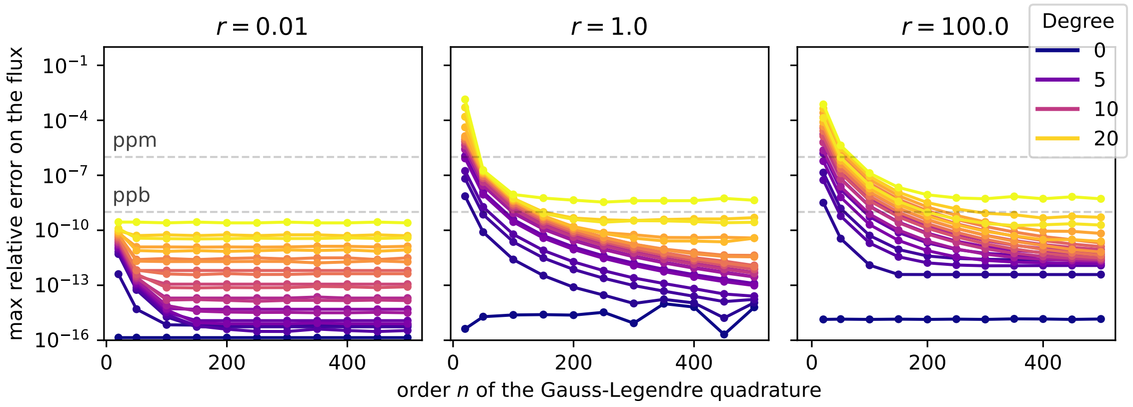

Figure 10. Maximum errors in the overall flux f for all degrees of the spherical harmonics components for different occultor radii and orders n of the Gauss-Legendre quadrature used to compute the P integral (see **??**). Errors shown in this figure correspond to the maximum values of the scaled errors shown in Figure 5, maxima taken over the range of values b.

3.2. *Speed*

One of the advantage of the `JAX` library is its ability to perform hardware-accelerated computations on CPUs and GPUs. In this section, we evaluate the speed of the jaxoplanet implementation and compare it with the C++ implementation of starry. For quadratically limb-darkened light curve, we also compare our implementation with the one provided by the exoplanet Python package.

In Figure 11, we show the evolution of the computing time required to compute an occultation light-curve of a limb-darkened star and that of a non-uniform star described by spherical harmonics with a maximum degrees $l = 20$, depending on the order n of the Gauss-Legendre quadrature used to compute the $\mathcal{P}$ integral numerically (see **??**).

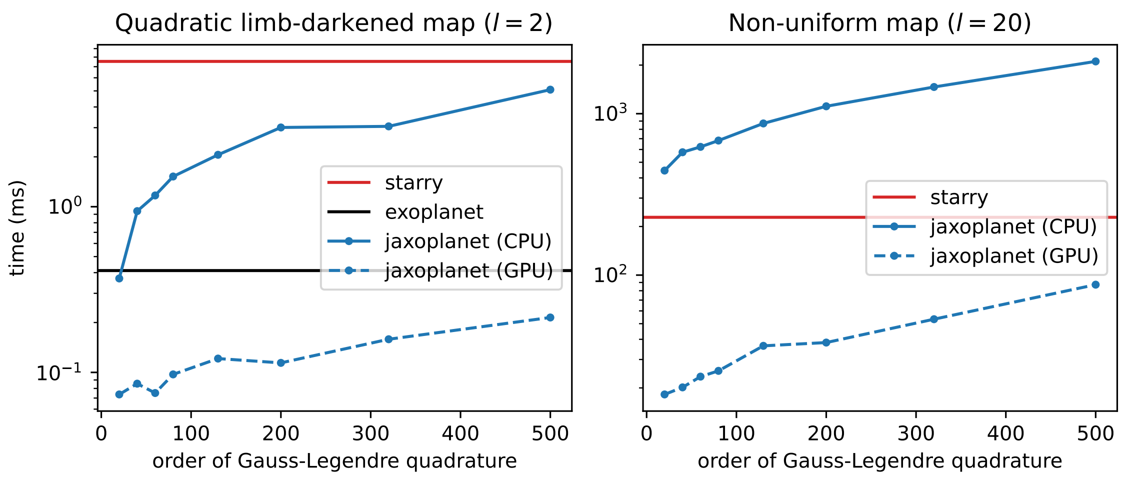

Figure 11. Computation time of the occultation light curve of a limb-darkened star (left) and a non-uniform star described by spherical harmonics up to order $l = 20$ (right) depending on the order n of the Gauss-Legendre quadrature used to compute the P integral numerically (see **??**). The processing time reported for exoplanet and starry is independent of n as these two codes provide closed-form solutions for the integral $\mathcal{P}$. Light curves are computed for 10 000 points in transit and an occultor radius $r = 0.1$.

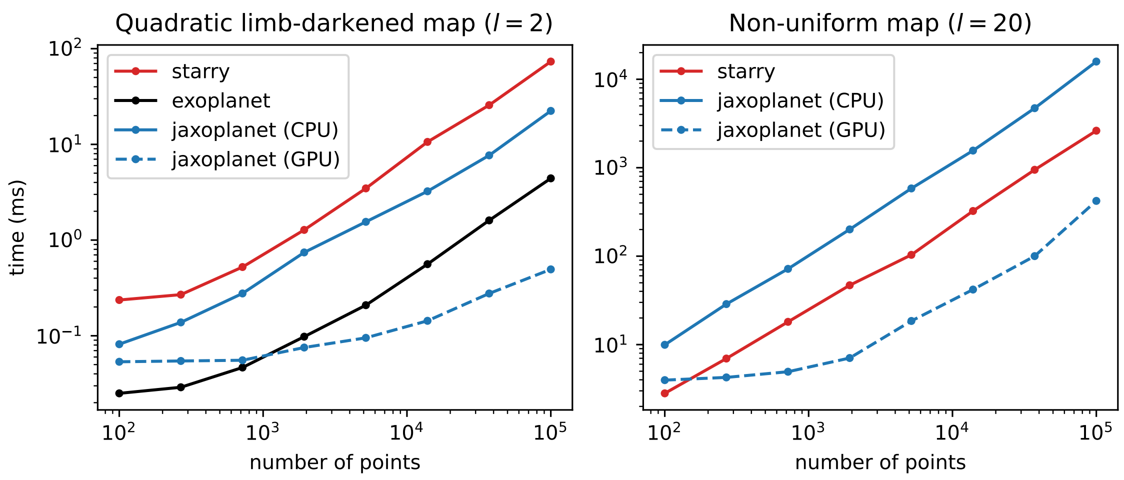

Figure 12. Computation time of the occultation light curve of a limb-darkened star (left) and a non-uniform star described by spherical harmonics up to order $l = 20$ (right) depending on the number of points in the light curve. Light curves are computed at order $n = 200$ and an occultor radius $r = 0.1$.

Figure 11, in combination with Figure 10, can be used to determine a value of n balancing precision and computation time.

Finally, Figure 12 shows the evaluation time of jaxoplanet as a function of the number of points in the light curve, compared to starry for an $l = 20$ surface map, and exoplanet for a quadratically limb-darkened star. As in the previous figure, we show that jaxoplanet is competitive against state-of-the-art codes on CPUs and offer a clear advantage on GPUs.

4. CASE STUDIES: ...

4.1.

REFERENCES

Agol, E., Luger, R., & Foreman-Mackey, D. 2020, AJ, 159, 123, doi: 10.3847/1538-3881/ab4fee

Luger, R., Agol, E., Foreman-Mackey, D., et al. 2019, The Astronomical Journal, 157, 64, doi: 10.3847/1538-3881/aae8e5

—. 2021, rodluger/starry: v1.2.0, v1.2.0, Zenodo, doi: 10.5281/zenodo.5567781

Pál, A. 2012, MNRAS, 420, 1630, doi: 10.1111/j.1365-2966.2011.20151.x

Price, M. A., & McEwen, J. D. 2023, Journal of Computational Physics, submitted

Risbo, T. 1996, Journal of Geodesy, 70, 383, doi: 10.1007/BF01090814

APPENDIX

A. ERROR BUDGET

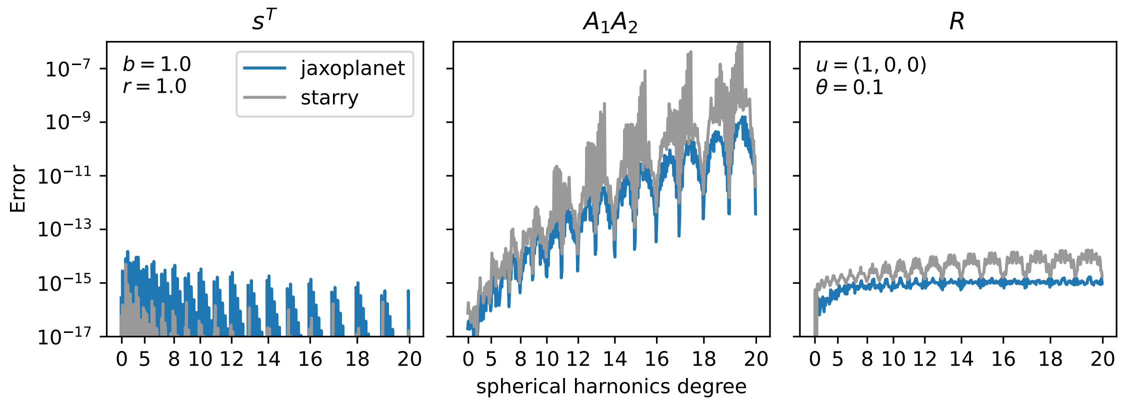

Figure 13.

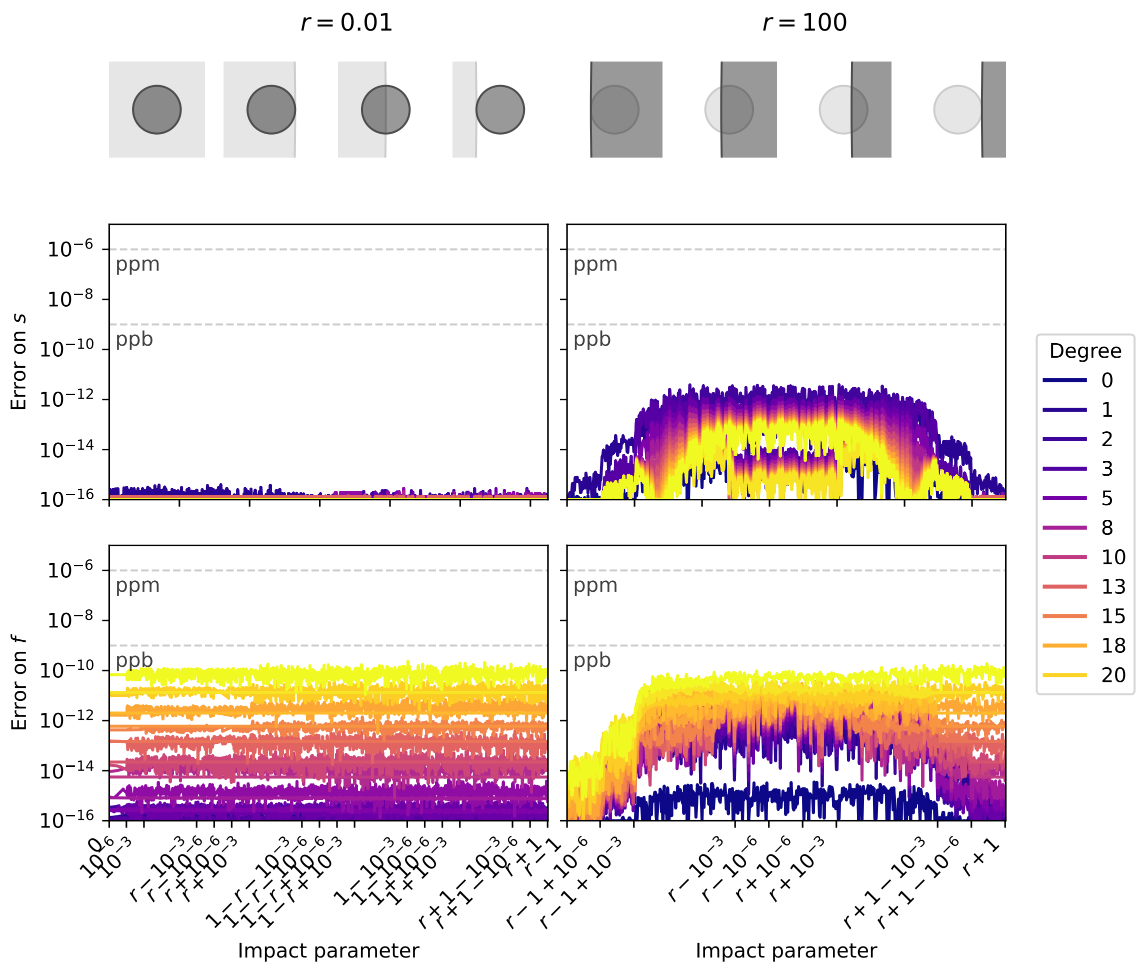

Figure 14. Same as Figure 5 but with *bvecs* and f computed with the C++ implementation of starry.

Table 1. Symbols used in this paper

Symbol	Definition	Reference
ω	Rotation angle of the combined rotation	Equation 15
n	order of the Gauss-Legendre approximation	
r	occultor radius in units of occulted body's radius	