Delft University of Technology
Faculty of Applied Sciences
Faculty of Electrical Engineering, Mathematics and Computer Science
Department of Mathematical Physics

Exoplanet surface mapping using scattered light curves

by

Samuel Stuger

to obtain the degree of

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE
TECHNISCHE NATUURKUNDE

Amsterdam, Netherlands
August 2021

TU Delft Delft University of Technology

BSc thesis TECHNISCHE WISKUNDE & TECHNISCHE NATUURKUNDE

"Exoplanet surface mapping using scattered light curves"

Samuel Stuger

Delft University of Technology

Supervisors

Dr. Paul M. Visser

Dr. Aurèle J. L. Adam

Committee members

Dr. Akira Endo

Dr. Klaas P. Hart

Dr. Daphne M. Stam

ABSTRACT

Characterizing the surfaces of Earth-like exoplanets will be an essential challenge in investigating their habitability and the possible existence of life on the surfaces of these planets. Using the next generation of telescopes, we will be able to observe the reflected starlight of exoplanets. However, due to the vast distance, exoplanets will only appear as single pixels. Nonetheless, the varying brightness of such a pixel, due to the the annual rotation of the planet around their star and daily rotation around its axis, can provide information about the planetary surface.

In this thesis, we provide a method for reconstructing a planet's surface map from its reflected light curve. We are going to derive an equation for the reflective light-curve under the assumption that the surface map is characterized by four different surface types (ocean, vegetation, sand and snow), is stationary (no clouds), and that all reflection is diffuse (Lambertian). We will show that the transformation is a linear function of the surface map and we will work out the transformation for arbitrary observer inclination and axial tilt. Using this knowledge, we create mock light curve data of self-generated planets. Afterwards, the transformation is inverted using the Moore-Penrose pseudo-inverse and the mock data will be used to demonstrate the surface map recovery for edge-on and face-on observations of planets with different axial tilts. Furthermore, we also provide a method for recovering the planet's axial tilt from its reflected light curve.

Even when a realistic amount of photon shot noise is added to the light curve, we are able to retrieve the planet's surface map and axial tilt fairly well, especially when the planet's tilt has larger axial tilt.

CONTENTS

LIST OF SYMBOLS

e_{max}	Longest edge of tetrahedron
h_{new}	Height value of added vertex
ρ	Radius of exoplanet
R	Radius of orbit
I_0	Power output of star
ω	Orbital angular velocity
Ω	Spin angular velocity
$\hat{\mathbf{e}}_\mathbf{o}$	Unit vector from star to observer
$\hat{\mathbf{e}}_\mathbf{r}$	Unit vector from planet to star
$\hat{\mathbf{e}}_\mathbf{s}$	Unit vector from center of planet to planetary surface
$\hat{\mathbf{n}}$	Spin-axis of planet
$\mathfrak{D}$	Domain on exoplanet surface that is simultaneously visible and illuminated
R_y	Elemental rotation matrix about y-axis
R_z	Elemental rotation matrix about z-axis
$\hat{\mathbf{e}}'_\mathbf{s}$	Rotated unit vector from center of planet to planetary surface
ϕ	Longitude of the exoplanet
θ	Colatitude of the exoplanet
α	Equinox angle of the exoplanet
β	Obliquity of the exoplanet
(α, β)	Spin angular velocity
γ	Observer inclination
N_{ave}	Average number of photons that reach observer
$\dot{N}_{star}$	Average rate of photons emitted by star
d_{JWST}	Diameter of main telescope of James Webb Space Telescope
d_o	Distance to exoplanet
t_{int}	Integration time of telescope
f	Reflected light curve
$\mathbf{f}$	Sampled light curve
a	Albedo map
$\mathbf{a}$	Albedo map vector in pixel basis
T	Transformation matrix mapping albedo map to light curve
T^+	Moore-Penrose pseudo-inverse
σ_{max}	Largest singular value
$SVCR$	Singular Value Cutoff Ratio

1 INTRODUCTION

Since scientists and philosophers have started to question geocentrism in the sixteenth century, some of the most broad-minded people have suspected that planets might not be exclusive to our solar system. The first person to put forward this blasphemous idea of exoplanets was Giordano Bruno, an Italian philosopher, mathematician, poet and cosmological theorist, who proposed that the stars were remote suns surrounded by their own planets. However, he was way ahead of his time and, due to political and religious reasons, his ideas fell on deaf ears with most of his contemporaries. This started to change after Isaac Newton presented his *Philosophiæ Naturalis Principia Mathematica*, where he also mentioned the existence of distant planets, and belief in the existence of exoplanets started to grow.

Unfortunately, in spite of the growing support for the idea of distant planetary systems, there was no way to actually observe them. Until in 1995, almost four hundred year after Giordano Bruno's death, the first exoplanet named Dimidium was discovered by astronomers Mayor and Queloz [1995] through its tiny gravitational pull on its star, 51 Pegasi.

Currently, more than 4000 exoplanets have been confirmed and more than 3500 candidates are waiting for verification [1]. Some of these exoplanets lie in the "habitable zone", close enough to their star that, assuming the presence of an Earth-like atmosphere, water on the planet's surface would not freeze or boil off [Dressing and Charbonneau, 2015; Tuomi et al., 2014]. These Earth-like planets could even contain continents and watery oceans, which would make them promising candidates for extraterrestrial life.

Since discovering extraterrestrial life is one of the ultimate goals in the study of exoplanets, astronomers not only seek to locate new exoplanets, but also examine their surface environment and meteorology [Kawahara and Fujii, 2011]. However, examining exoplanet surfaces by direct imaging is still impossible due to the enormous distance to other planetary systems. Even with the next generation of telescopes, which NASA and ESA are currently designing, exoplanets will appear only as single pixels to us, much like Earth in the Pale Blue Dot picture taken by Voyager 1 as it left the solar system (Figure 1.1).

As the planet orbits around its star, the starlit portion of the exoplanet surface that is visible to us changes much like the phases of the moon, resulting in an annual change in the pixel's brightness. Furthermore, if the exoplanet surface is not homogeneous (e.g. due to the presence of continents and oceans), daily oscillations in the pixel's brightness will also be visible as the exoplanet rotates about its axis and different parts of the surface with

1 https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PS

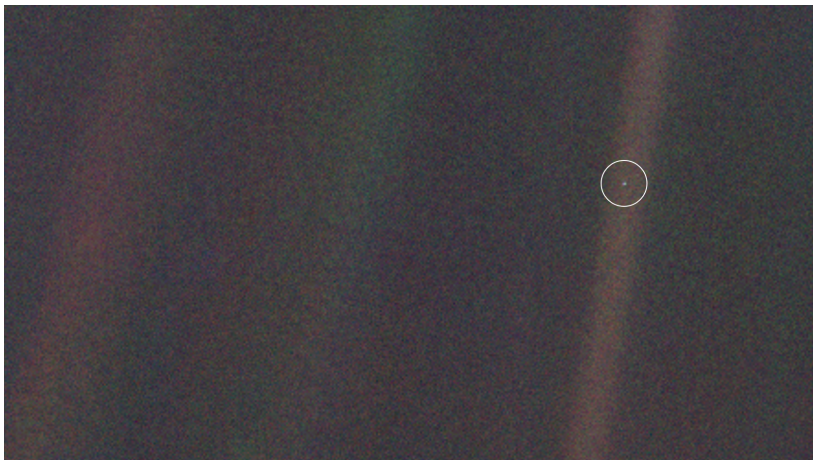


Figure 1.1: The Pale Blue Dot picture of Earth, taken by Voyager 1 as it left the solar system. Like Earth in this picture, a directly observed exoplanet will appear as a single pixel. In this thesis, we will use the pixel's varying brightness to retrieve the planet's surface map.[2]

different reflectivities (albedos) are illuminated by the starlight. This double-periodic light signal is called the reflected light curve of the exoplanet.

Some authors (e.g. [Cowan and Agol, 2008; Farr et al., 2018; Fan et al., 2019; Aizawa et al., 2020; Kawahara and Masuda, 2020; Cowan and Fujii, 2021]) have started solving the problem of recovering surface maps of exoplanets by creating mock observations of the Earth's reflected light curve using the albedo data of our planets that has been measured by NASA satellites (Figure 1.2). They have shown that by measuring the changes in the intensity of the reflected light curve, it is possible to retrieve both the planet's axial tilt and a rough surface map, using daily variations for longitudinal resolution and annual variations for latitudinal resolution.

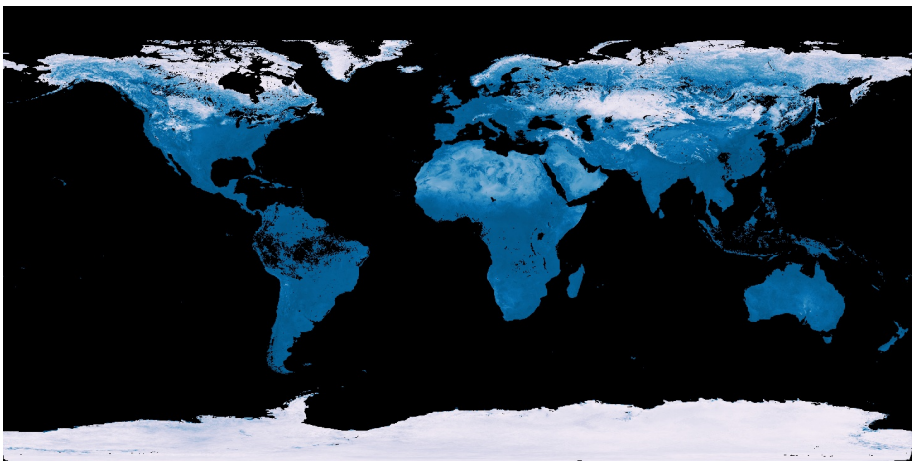


Figure 1.2: Albedo map of Earth as recorded by NASA in February 2017.[3]

In this thesis, we attempt to expand on this method of recovering exoplanet surfaces, named spin-orbit tomography (SOT) by Kawahara and Fujii [2010]. However, instead of using Earth's albedo data to produce reflected light curve measurements, we will first generate surface maps of our own planets. Afterwards, we will try to retrieve both the surface map (albedo

map) and the axial tilt of the planet for several spin-axis configurations and observer inclinations. This will be done in order to get a better understanding of the retrieval capabilities of spin-orbit tomography, as the next generation of satellites will be in high demand.

In Chapter 2 of this thesis, the method of tetrahedral subdivision will be used to generate Earth-like planetary surfaces. Next, in Section 3.2, an analytical expression for the reflected light curve that arises from a planet's albedo map will be derived. In Section 3.3, this analytical formula will be translated into a numerical linear transformation, mapping a planets albedo map onto its reflected light curve. Lastly, in Chapter 4, all the knowledge acquired in the previous chapters will be used to retrieve the surface maps of exoplanets and the problem of unknown axial tilt will be tackled.

2 THE GENERATION OF ALBEDO MAPS

The objective of this thesis is the retrieval of exoplanet surfaces based on their reflected light curves from their host star. Due to the current absence of light curve data of exoplanets and other planets and moons in our Solar System, artificial planets and their albedo maps will be generated. Therefore, in this chapter the concept of Bond albedo and the generation of planetary surface and albedo maps will be discussed.

2.1 THE CONCEPT OF ALBEDO

The Bond albedo of a surface is a measure of the diffuse reflection (Lambertian) of the surface. It is defined as the ratio of the intensity of the total stellar radiation that is reflected by the surface to the intensity of the total stellar radiation that reaches the surface. Bond albedo is therefore measured on a scale from 0, corresponding to total absorption of all incident radiation, to 1, total reflection of all incident radiation. The albedo map of an exoplanet, a, then maps any point on the surface of the exoplanet to the interval [0, 1], according the Bond albedo of that point. Typical Bond albedos of planets in the Solar System range from 0.088, for Mercury , to 0.76, for the cloud-covered Venus [Mallama, 2017; Haus et al., 2016]. Terrestrial Bond albedos range from 0.06, for the ocean, to 0.80, for freshly fallen snow [NSIDC, 2020]. For the to-be-generated albedo maps a distinction will be made between four types of surfaces: ocean, vegetation, sand and snow. These surface types were chosen since they are all prevalent surface types on Earth, and for their indicative ability of planetary habitability and life or their commonness amongst other terrestrial planets in our Solar System. Their typical albedos can be viewed in Table 2.1.

Surface Type	Typical Albedo	Motivation
Ocean	0.06	Indicator of habitability/life
Vegetation	0.15	Indicator of habitability/life
Sand	0.40	Common surface type
Snow	0.80	Common surface type

Table 2.1: Typical Bond albedos of four common terrestrial surface types [NSIDC, 2020; Betts and Ball, 1997; Tetzlaff, 1983].

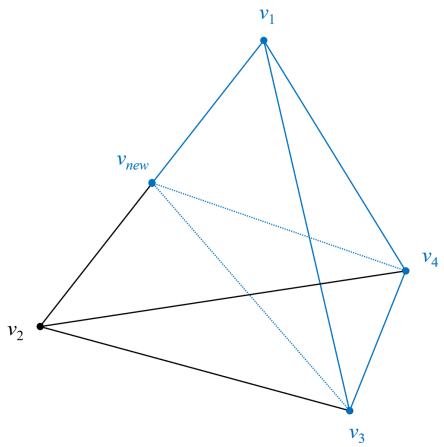


Figure 2.1: Cutting a tetrahedron into two smaller tetrahedra by adding a vertex on one of the edges.

2.2 THE GENERATION OF ALBEDO MAPS

For the generation of albedo maps of 'imaginary' exoplanets, first altitude maps of these planets will be created. These altitude maps map any pixel on the planetary surface to a certain height value. Then, these altitude maps will be converted into albedo maps by dividing the surface into the four aforementioned surface types based on elevation. The altitude maps will be generated using the method of tetrahedral subdivision, as described by Mogensen [2010].

2.2.1 Altitude map generation by tetrahedral subdivision

The method of tetrahedral subdivision is a method that is normally used for generating pseudo-random surface maps for games and art. The method ultimately constructs a 2D grid of values that assigns a height value to each point on the grid, thereby creating an altitude map of the surface. By placing the set of points on a sphere, pseudo-random altitude maps of planets can be created.

The method of tetrahedral subdivision for generating surface maps, starts by placing a tetrahedron around a spherically distributed set of points and assigning a height value to each of the vertices of the tetrahedron. The height value of a point is then computed using the following algorithm. First, an extra vertex is added to the longest edge. Hereafter, edges are created between the added vertex and the two nonadjacent vertices, cutting the tetrahedron into two smaller tetrahedra (Figure 2.1). Subsequently, a pseudo-random height value is computed for the new vertex, based on vertices of the cut edge. Next, the tetrahedron containing the point is selected and the algorithm is repeated. When the desired resolution is reached, the height value of the point is taken to be the average height value of the vertices of the final tetrahedron. A more detailed description of the process can be found in Algorithm 2.1 and examples of a generated planets are shown in Figure 2.2 and Figure 2.3.

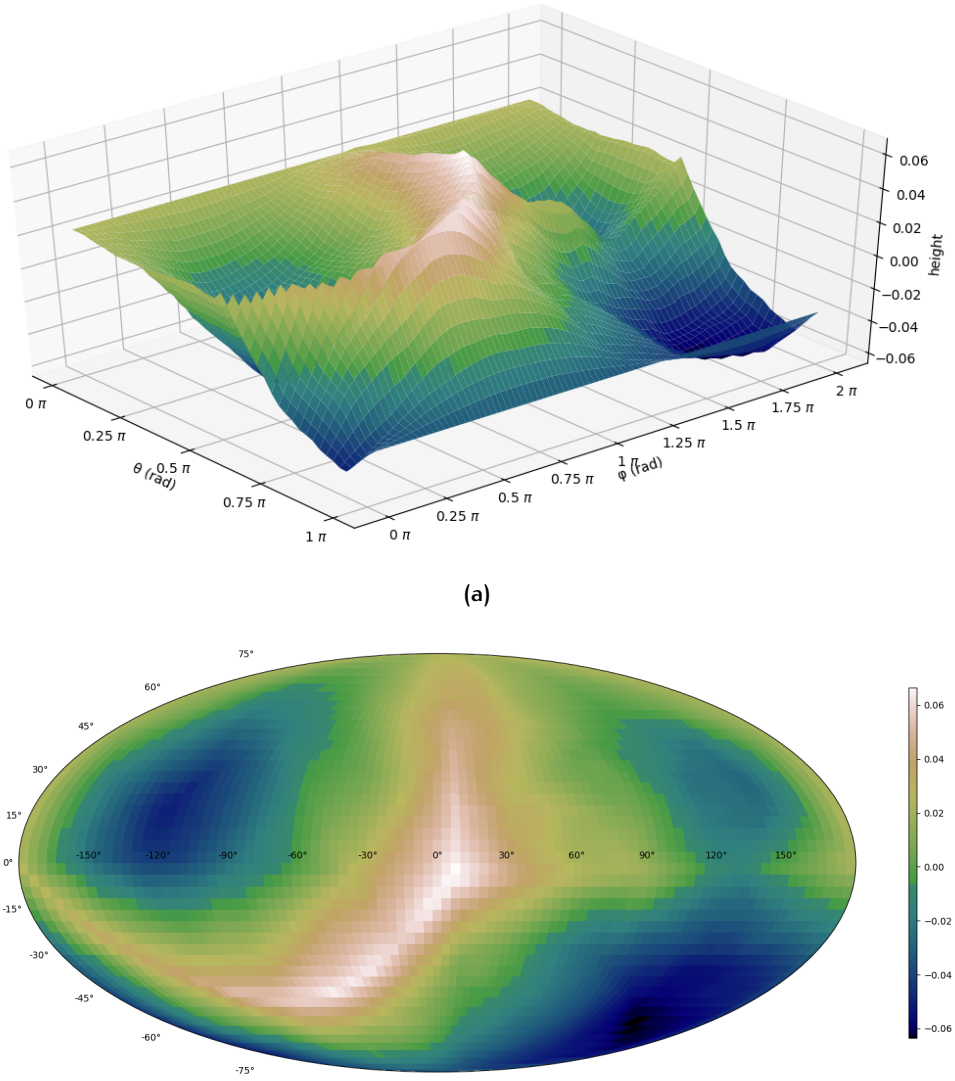


Figure 2.2: An example of a surface map generated using the method of tetrahedral subdivision as described by [Mogensen, 2010]. (a) A 3D-projection of the altitude-grid. (b) A Mollweide projection of the grid on a sphere.

Algorithm 2.1: TETRAHEDRAL SUBDIVISION(p, v, s, h)

Input: A point on the unit-sphere p, array of coordinates of the vertices of a starting tetrahedron v, an array containing their seeds s, and an array containing their altitudes h

1 **Repeat:**

1. Find longest edge e_{max}
2. Rearrange vertices so that e_{max} is between v_1 and v_2
3. Create new vertex v_{new} with:
   - $v_{new} = (v_1 + v_2)/2$
   - $s_{new} = (s_1 + s_2)/2$
   - $h_{new} = (h_1 + h_2)/2 + 0.01 \times random(s_{new}) \times e_{max}{}^{\frac{3}{2}}$

   Where *random* produces a pseudo-random number between -1 and 1
4. If p lies inside the convex hull of $\{v_1, v_{new}, v_3, v_4\}$, set $v_2 = v_{new}$, $s_2 = s_{new}$ and $h_2 = h_{new}$. Otherwise, set $v_1 = v_{new}$, $s_1 = s_{new}$ and $h_1 = h_{new}$

2 **Stop if:** e_{max} is small enough

Output: $(h_1 + h_2 + h_3 + h_4)/4$: The height value of p

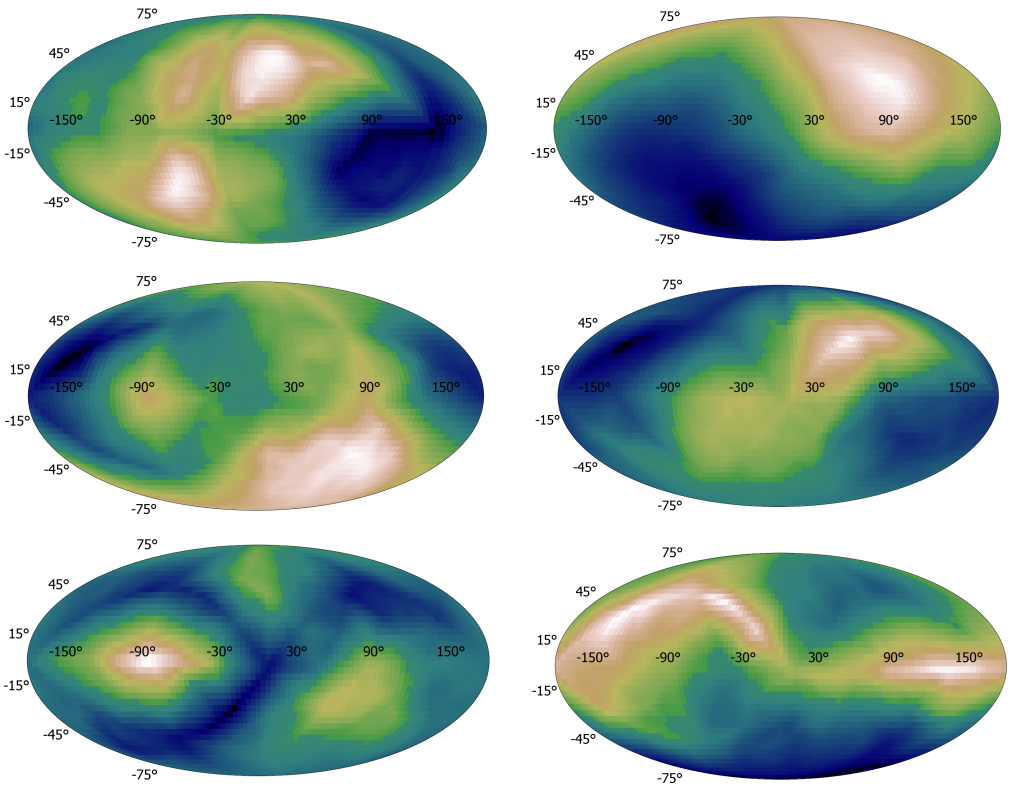


Figure 2.3: Several examples of surface maps generated using the method of tetrahedral subdivision as described by [Mogensen, 2010].

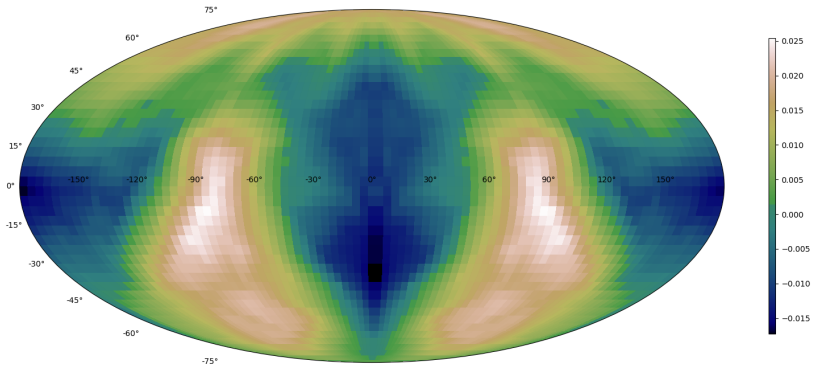


Figure 2.4: An example of a composite map that is created by mirroring a planet around the prime meridian. In Section 3.3.1 this mirrored map will in be used to better illustrate the daily variation of reflected light curves.

Besides computational efficiency, the main advantage of the method of tetrahedral subdivision is that, unlike most other surface map generators, there are no discontinuities or artefacts created when projecting the map onto a sphere. Furthermore, discontinuities between nearby points are greatly reduced, since h_{new} scales with ${e_{max}}^{\frac{3}{2}}$ and by the time that two nearby points lie in different tetrahedra, e_{max} is way smaller than 1. This results in a more realistic, smoother surface. However, a direct negative consequence of this smoother surface is that the created planets are generally simple planets housing only a single continent. Nevertheless, surface maps that consist of multiple continents can be created in a multitude of ways. For example, by mirroring maps around a meridian or by adding parts of different surfaces together. However, this can create unwanted discontinuities in the surface map. An example of a composite map, which will later be used to better illustrate some properties of reflected light curves, is shown in Figure 2.4.

2.2.2 From altitude map to albedo map

After an altitude map of a planet is generated, its albedo map is created by dividing the surface pixels into the aforementioned surface types based on the pixel's relative elevation. All the pixels below average height are assigned the Bond albedo value corresponding to ocean, while the other pixels are assigned the albedo value corresponding to either vegetation, sand or snow, in ascending order of elevation. Figure 2.5 shows the albedo maps of the generated planets shown in figure Figure 2.3.

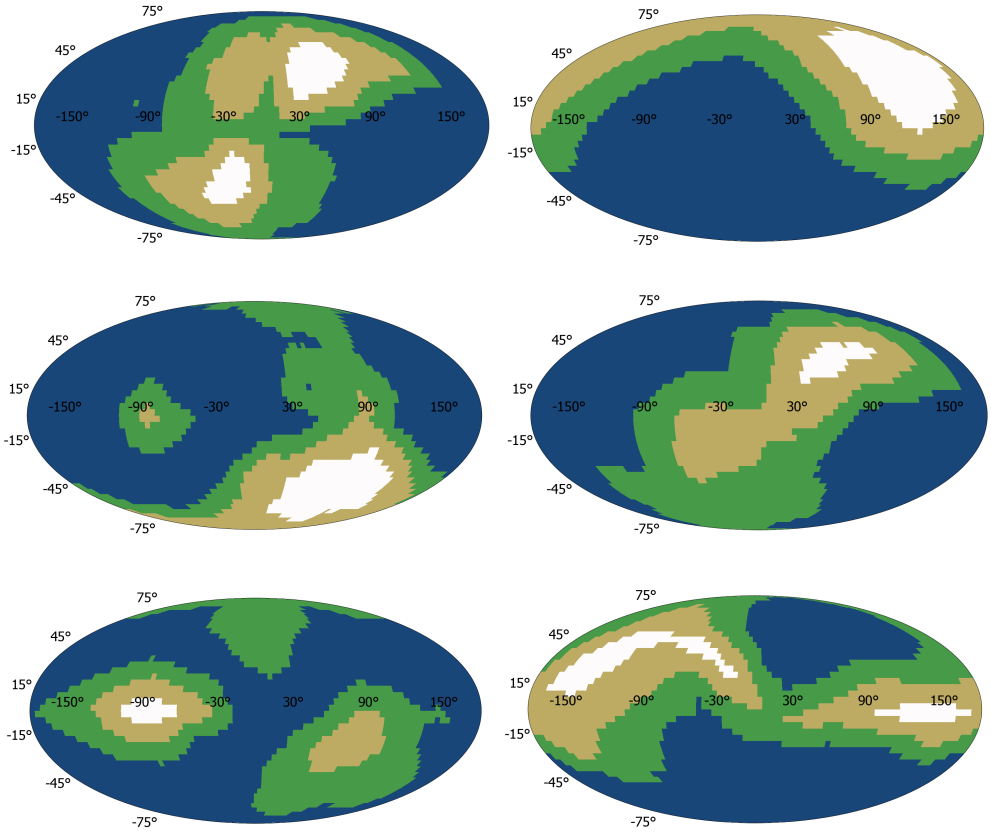


Figure 2.5: Several examples of albedo maps of the generated planets illustrated in Figure 2.3 by dividing the surface pixels in four surface types: water (blue), vegetation (green), sand (yellow) and snow (white) with Bond albedos as in Table 2.1. The division is based on the height value of the pixels.

3 REFLECTED LIGHT-CURVE OF A PLANET

In this chapter an expression for how the reflected light curve of an exoplanet arises from the albedo map of the exoplanet will be derived. This expression will be used to compute and analyze multiple reflected light curves of the planets generated in Chapter 2. In the following chapter, this knowledge shall be used to construct the albedo map of a generated exoplanet using its reflected light curve.

3.1 THE GENERAL CASE

The general case that shall be explored in this thesis, which can be viewed in Figure 3.1, is a planetary system, where a lone, moonless planet with radius ρ moves around its star in a circular orbit in the xy-plane with radius R and angular frequency ω. Additionally, the planet rotates around its own axis $\hat{\mathbf{n}}$ with angular frequency Ω. Furthermore, the following vectors need to be introduced:

$$\hat{\mathbf{e}}_\mathbf{r} = \begin{bmatrix} -\cos\omega t \\ -\sin\omega t \\ 0 \end{bmatrix} \quad \hat{\mathbf{e}}_\mathbf{s} = \begin{bmatrix} \cos\phi\sin\theta \\ \sin\phi\sin\theta \\ \cos\theta \end{bmatrix} \quad \hat{\mathbf{e}}_\mathbf{o} = \begin{bmatrix} \sin\gamma \\ 0 \\ \cos\gamma \end{bmatrix} \quad \hat{\mathbf{n}} = \begin{bmatrix} \cos\alpha\sin\beta \\ \sin\alpha\sin\beta \\ \cos\beta \end{bmatrix} \tag{3.1}$$

where the vectors $\hat{\mathbf{e}}_\mathbf{r}$ and $\hat{\mathbf{e}}_\mathbf{o}$ point from centre of the planet towards the host star,the observer, respectively. The vector $\hat{\mathbf{e}}_\mathbf{s}$ is a vector normal to the surface, defined by fixed spherical coordinates ϕ and θ. Furthermore, the axial tilt (α, β) of the exoplanet specifies the orientation of the spin-axis of the planet $\hat{\mathbf{n}}$. and the observer inclination γ defines the viewing angle of the observer [1]. A detailed schematic illustration of the general case can be found in Figure 3.1.

3.2 ANALYTIC EXPRESSION FOR THE REFLECTED LIGHT-CURVE

We are interested in the features of the planet's surface, but as explained in the introduction, we do not have the resolution to resolve features on the surface by direct imaging. Only the total light intensity that the planet reflects towards us is measurable, this physical quantity is called the reflected light curve $f(t)$ of the exoplanet . The reflected light curve that reaches the

1 There are two cases that will primarily be studied: edge-on ($\gamma = \frac{\pi}{2}$) and face-on ($\gamma = 0$). For edge-on observation, the observer is in the orbital plane, while for face-on observation the observer is perpendicular to the orbital plane.

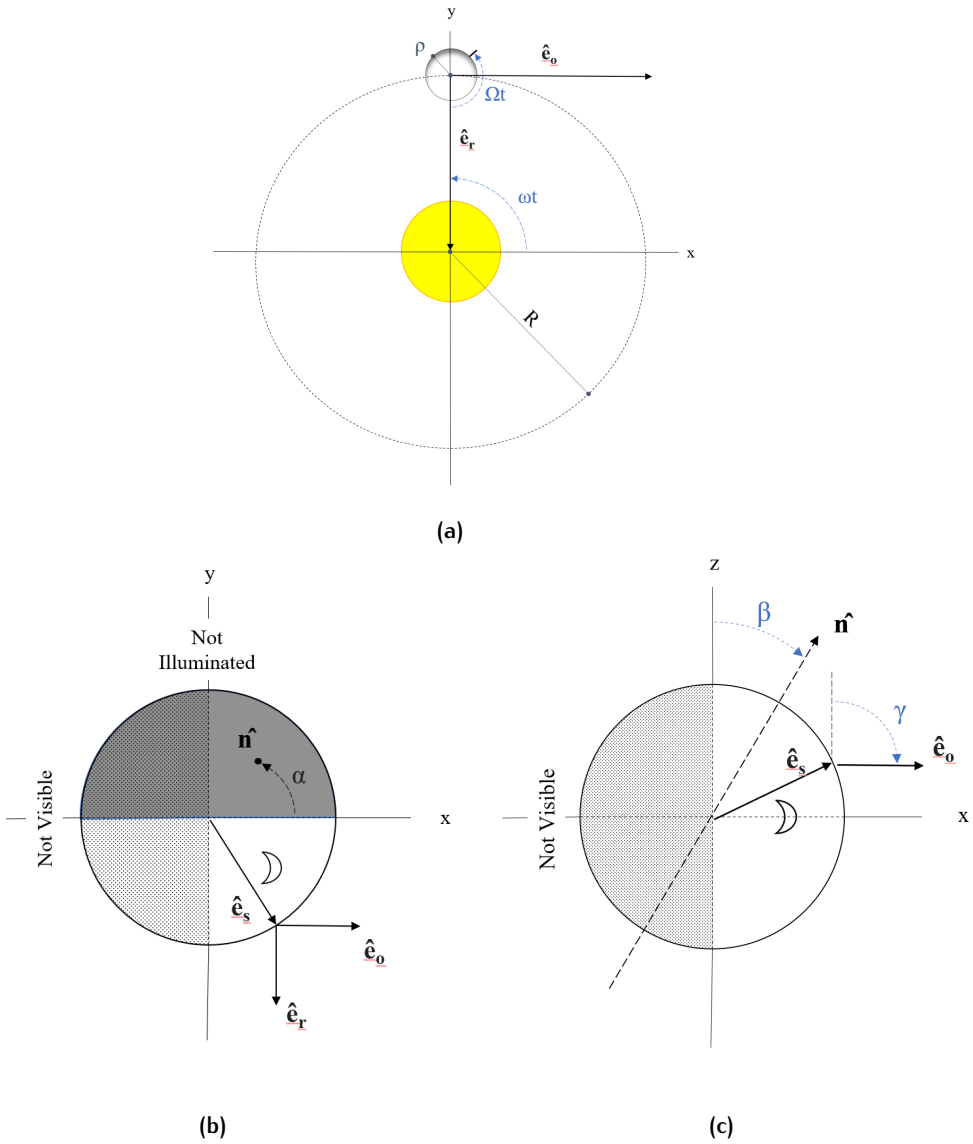


Figure 3.1: Schematic illustration of our planetary system. The planet is moving in a circular orbit around its host star with angular frequency ω, starting on the positive x-axis, while spinning around its own axis with angular frequency Ω. The unit vectors $\hat{\mathbf{e}}_\mathbf{r}$ and $\hat{\mathbf{e}}_\mathbf{o}$ point from the center of the planet towards the host star and the observer, respectively. In addition, the unit vector $\hat{\mathbf{e}}_\mathbf{s}$ is a vector normal to the planet surface. As mentioned in Section 3.2, the illuminated pixels satisfy $\langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle \geq 0$, and visible pixels satisfy $\langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle \geq 0$. The spin axis $\hat{\mathbf{n}}$ is specified by the equinox angle α, the angle of between the projection of $\hat{\mathbf{n}}$ onto the xy-plane and $\hat{\mathbf{x}}$, and the obliquity β, the angle between $\hat{\mathbf{n}}$ and $\hat{\mathbf{z}}$. (a) xy-projection of the planetary system. (b) xy-projection of the planet. (c) xz-projection of the planet.

observer is the product of three terms, each describing a stage in the journey of the light waves, from planetary incidence to observation:

$$\begin{aligned} f(t) = &\text{ fraction of total stellar power that reaches the surface} \\ &\times \text{ Bond albedo of the surface} \\ &\times \text{ fraction of reflected light that reaches the observer} \end{aligned} \tag{3.2}$$

In order to derive an analytical expression for the reflected light-curve, we need to know which domain on the planetary surface contributes to the light-curve. In other words, which part of the surface reflects light from the host star to the observer. First, note that precisely one hemisphere of the surface is visible to us, namely the hemisphere containing the points on the surface where $\langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle \geq 0$. Moreover, also one hemisphere of the planetary surface is illuminated by the star, namely the hemisphere containing the points on the surface where $\langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle \geq 0$. The intersection of these two hemispheres is exactly the part of the surface that contributes to the reflected light-curve, and is called the observable domain, denoted by $\mathfrak{D}$:

$$\mathfrak{D} = \{\mathbf{s} : \langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle \geq 0 \wedge \langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle \geq 0\} \tag{3.3}$$

If the total stellar power is I_0, then at a distance R from the star the intensity has dropped to $I_0/4\pi R^2$. The fraction of the stellar power that is reflected by an infinitesimally small part of the surface $d^2\Omega_s$ is then equal to $a_s \langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle d^2\Omega_s$. Here a_s is the Bond albedo of the surface at $\hat{e}_s$. Furthermore, the fraction of the reflected light that reaches the observer is proportional $\langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle d^2\Omega_o/\pi$. Here $d^2\Omega_o$ is the solid angle of the observer. Multiplying these terms gives the contribution to the reflected light curve of a small part of the illuminated an visible surface:

$$\frac{I_0 \langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle \langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle a_s}{4\pi R^2} d^2\Omega_s d^2\Omega_o \tag{3.4}$$

The reflected light curve can then be found by integrating the above term over the observable domain $\mathfrak{D}$. However, astronomers often only measure the intensity of the light reflected by the planet relative to the total intensity of the star light that reaches them $I_0 d^2\Omega_o/4\pi$, thus it is customary to divide by this term [Winn et al., 2008]. Therefore, the reflected light curve can be computed using the following formula:

$$\begin{aligned} f(t) &= \frac{1}{\pi R^2} \iint_{\mathfrak{D}} \langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle \langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle a_s d^2\Omega_s \\ &= \frac{\rho^2}{\pi R^2} \iint_{\mathfrak{D}} \langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}_\mathbf{s} \rangle \langle \hat{\mathbf{e}}_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle a(\phi, \theta) \sin\theta d\phi d\theta \end{aligned} \tag{3.5}$$

3.3 NUMERICAL EXPRESSION FOR THE REFLECTED LIGHT-CURVE

In the previous section, we derived an an analytical expression for the reflected light, which allows us to compute its intensity of the reflected light curve at any time t assuming the Bond albedo of every point on the sphere is known. However, the maps generated in Chapter 2, provide the Bond albedo of only a finite number of points, say $N_\phi \times N_\theta$. Furthermore, real-life light curve measurements will also yield only a finite amount data, giving us the intensity of the reflected light curve at only a finite number of times, say N_t. Therefore, we shall derive a numerical analog to Equation 3.5.

The discrete counterpart of the reflected light curve $f(t)$ can be represented by a vector $\mathbf{f} \in \mathbb{R}^{N_t}$. Furthermore, by flattening the albedo maps generated in Chapter 2, they can be also be represented as vectors $\mathbf{a} \in \mathbb{R}^{N_\phi \times N_\theta}$. We now propose that the reflected light curve can now be calculated using the following linear transformation:

$$\begin{aligned} \mathbf{f} &= T\mathbf{a} \\ \mathbf{f} &= \begin{bmatrix} f(t_1) & . & . & . & f(N_t) \end{bmatrix}^T \\ \mathbf{a} &= \begin{bmatrix} a_{1,1} & . & . & a_{1,N_\theta} & a_{2,1} & . & . & a_{N_\phi,N_\theta} \end{bmatrix}^T \end{aligned} \tag{3.6}$$

In order to see what the entries of the transformation matrix T should be, we will work out what the light curve of a single pixel would be. As mentioned before, the coordinate system of the planet is defined in such a way that a given coordinate (ϕ_p, θ_p) points to the same pixel on the map for any time t_i. As a direct consequence of this choice, we have to rotate $\hat{\mathbf{e}}_\mathbf{s}$ in order for the vector to keep pointing to the same pixel in the inertial system (the non rotating coordinate system of the host star). This will be done by first rotating the vector Ωt_i radians around the z-axis to take care of daily rotation of the planet, and then rotating the vector with β° around the y-axis and by α° around the z-axis, because of the planet's axial tilt. This process, often referred to as a 3-2-3 rotation [see Goldstein, 1997, chapter 3], is illustrated in Figure 3.2. Using the elementary rotation matrices the normal vector pointing from the center of the planet to the pixel $\hat{\mathbf{e}}'_\mathbf{s}$ can be computed as:

$$\hat{\mathbf{e}}'_\mathbf{s} = R_z(\alpha) R_y(\beta) R_z(\Omega t_i) \hat{\mathbf{e}}_\mathbf{s} \tag{3.7}$$

Assuming that the pixel is in the observable domain at time t_i, the entry of T can be computed using the following equation:

$$T_{t_i,\phi_p,\theta_p} = \langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}'_\mathbf{s} \rangle \langle \hat{\mathbf{e}}'_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle \sin\theta_p \Delta\phi \Delta\theta \tag{3.8}$$

The next obstacle is dealing with the implicit expression of $\mathcal{D}$. We know that a pixel is in the observable domain when both $\langle \hat{\mathbf{e}}_\mathbf{r}, \hat{\mathbf{e}}'_\mathbf{s} \rangle \geq 0$ and $\langle \hat{\mathbf{e}}'_\mathbf{s}, \hat{\mathbf{e}}_\mathbf{o} \rangle \geq 0$. If the pixel is not visible or not illuminated, the pixel does not contribute to the reflected light curve and thus $T_{t_i,\phi,\theta}$ should to be zero. Otherwise, its entry should be as previously described in Equation 3.8. This can be

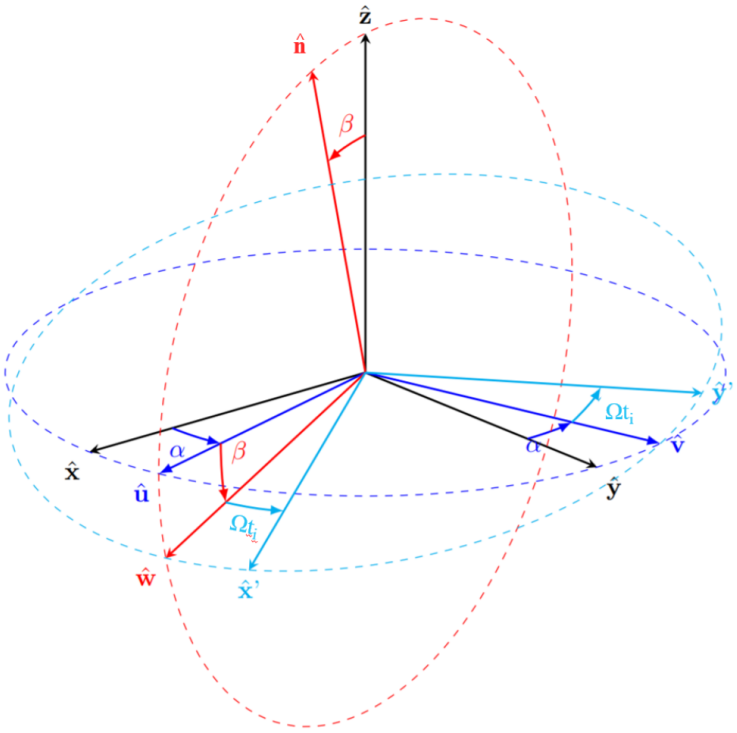


Figure 3.2: Schematic representation of the rotation of the planet with 3-2-3 Euler Angles $(\alpha, \beta, \Omega t_i)$. First rotating with Ωt_i around the z-axis, then rotating with β around the y-axis and lastly rotating with α around the z-axis yields the same result as directly rotating with Ωt_i around the spin axis $\hat{\mathbf{n}}$. Figure source code: [Dorian, 2017]

achieved by introducing the function $g^+ = (g + |g|)/2 = \max\{0, g\}$. By implementing this function to the two inner products, the entries of T can be found using:

$$T_{i,j} = \left[R_z(\alpha)\, R_y(\beta) R_z(\Omega t_i) \begin{pmatrix} \cos\phi_j \sin\theta_j \\ \sin\phi_j \sin\theta_j \\ \cos\theta_j \end{pmatrix} \cdot \begin{pmatrix} -\cos\omega t_i \\ -\sin\omega t_i \\ 0 \end{pmatrix} \right]^+ \left[R_z(\alpha) R_y(\beta) R_z(\Omega t_i) \begin{pmatrix} \cos\phi \sin\theta_j \\ \sin\phi_j \sin\theta_j \\ \cos\theta_j \end{pmatrix} \cdot \begin{pmatrix} \sin\gamma \\ 0 \\ \cos\gamma \end{pmatrix} \right]^+ \sin\theta \Delta\phi \Delta\theta \tag{3.9}$$

$$t_i = i\Delta t \qquad \phi_j = \left\lceil \frac{j}{N_\phi} \cdot \Delta\phi \right\rceil \qquad \theta_j = (j \mod N_\theta) \cdot \Delta\theta$$
$$\Delta\phi = \frac{2\pi}{N_\phi} \qquad \Delta\theta = \frac{\pi}{N_\theta}$$

3.3.1 Application to generated planets

In the last section we derived a method to compute the reflected light-curve of an exoplanet if the Bond albedo of a discrete number of points on the planetary surface is known. In this section, we will show some examples of these

reflected light curves, starting with the edge-on observation of the reflected light curve of a homogeneous planet which can be seen in Figure 3.3.

When the orbital phase angle ωt is zero, the planet is positioned right between the observer and star, so the planet reflects no light to the observer. As the orbital phase angle increases, the intersection of the visible hemisphere and illuminated hemisphere grows and more light gets reflected to the observer. The maximum intensity is reached when the planet is directly behind the star and the visible and illuminated hemisphere overlap completely[2]. Hereafter the intensity of the light curve decreases again until the planet has finished its yearly period around the star. This process is comparable to the moon cycle. However, instead of seeing a change in the shape of the moon, a change in the intensity of the light curve can be seen.

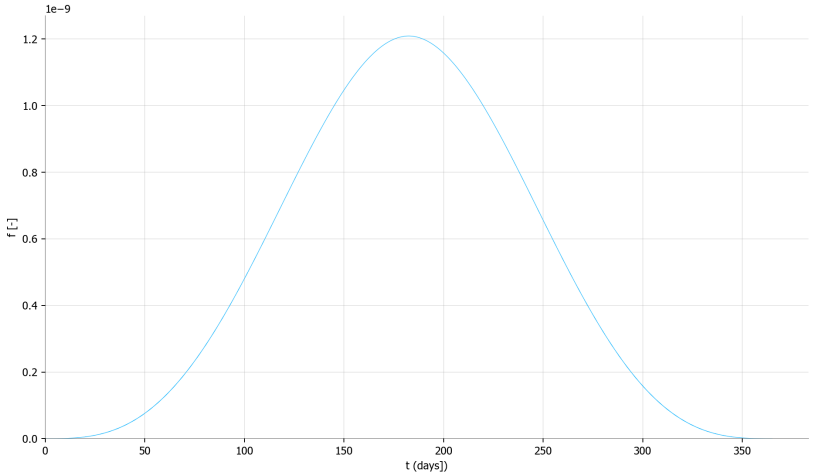


Figure 3.3: Graph of the reflected light curve obtained from a simulation of an edge-on observation of a homogeneous planet with axial tilt parameters $\alpha = 0°, \beta = 0°$. The light curve intensity reaches its minimum at orbital phase angle $\omega t = 0$ and maximum at $\omega t = \pi$

When the reflected light curve of a non-homogeneous planet is simulated, the daily variation of the light curve also becomes visible alongside the annual intensity variation that was shown in Figure 3.3. The reflected light curve of a non-homogeneous planet is shown in Figure 3.4, while Figure 3.5 displays the daily variation of the reflected light curve of the mirrored planet first shown in Figure 2.4. In Figure 3.5b the two peaks due to the high albedo of the two ice caps are clearly visible.

2 In real life, the planet would, of course, not be visible if it is positioned directly behind the star, but this will be overlooked in this report.

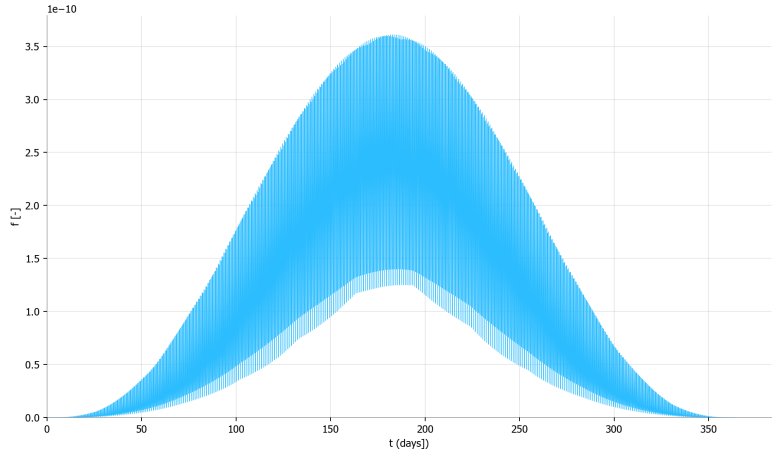
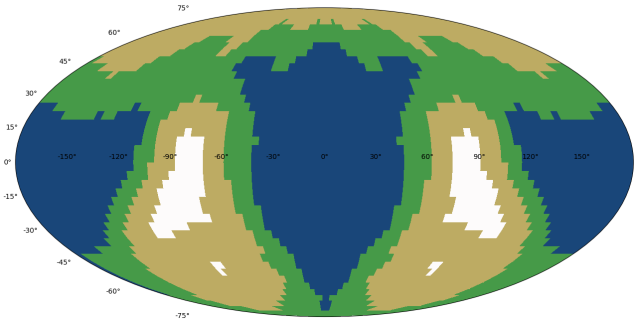
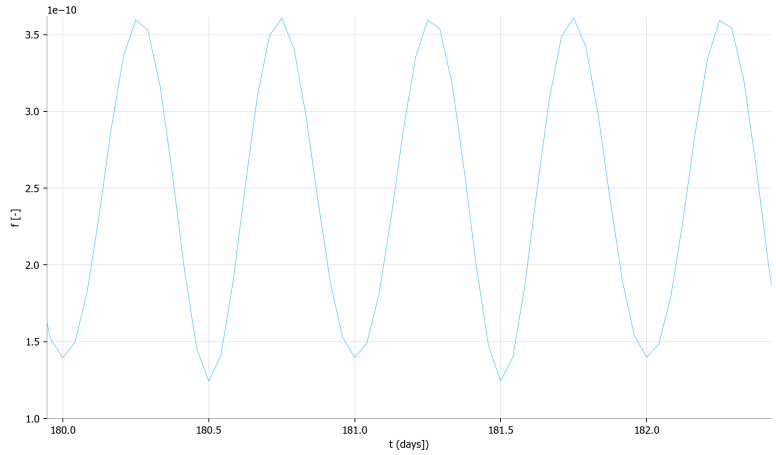


Figure 3.4: Graph of the reflected light curve obtained from a simulation of an edge-on observation of a non-homogeneous planet with axial tilt parameters $\alpha = 0°, \beta = 0°$.



(a)



(b)

Figure 3.5: (a) Mollweide-projection of the planetary albedo map. (b) Graph of the daily variation of the reflected light curve obtained from a simulation of an edge-on observation of a non-homogeneous planet, mirrored with axial tilt parameters $\alpha = 0°, \beta = 0°$. The two daily peaks due to the two continental ice caps is clearly visible.

3.3.2 Artificial shot noise

Since the distance to even the closest exoplanets is at least tens of light-years, the intensity of the measured reflected light curves of these planets will be very low and light curve measurements will contain a considerable amount of noise. Thus, in order to get a better grasp of how well albedo maps can be retrieved using actual observations of reflected light curves, noise will be added to the light curves computed in Section 3.3. However, instead of the more commonly used Gaussian noise, the noise will be modelled using shot noise.

Shot noise will be inherently present in the measured reflected light curve data, as the rate that a star emits photons is not perfectly constant. Shot noise dictates that the number of photons observed by the telescope is drawn from a Poisson distribution with the average number of photons that reach the observer per hour N_{ave} as mean. Therefore, given N_{ave}, the probability that the k photons reach the observer in an hour is given by:

$$P(\text{k observed photons in an hour}) = e^{-N_{ave}} \frac{(N_{ave})^k}{k!} \tag{3.10}$$

Some probability mass functions of the Poisson distribution are shown in Figure 3.6.

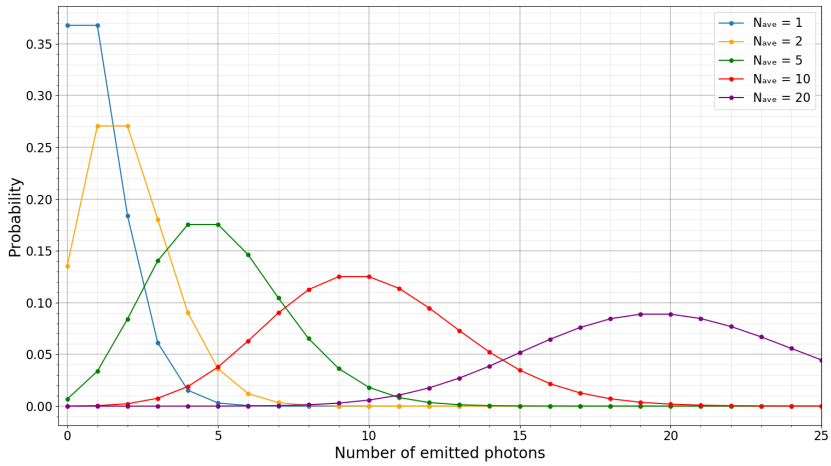


Figure 3.6: Graph showing the probability mass function of the Poisson distribution for different values of of the reflected light curve obtained from a simulation of an edge-on observation of a non-homogeneous planet with axial tilt parameters $\alpha = 0°, \beta = 0°$.

In order to get an understanding of the order of magnitude of N_{ave}, suppose we have a completely white, Earth-sized exoplanet orbiting, with an orbital radius of 1 AU, around a Sun-like star at 25 ly from the observer. Furthermore, we will assume that the light curve is measured with the to-be-launched James Webb Space Telescope, whose main telescope has a diameter of $d_{JWST} = 6.5$ m and can measure wavelengths from $\lambda = 600$ nm to 28.3μm. Lastly, the integration time of the telescope is taken to be 1 hour. Now N_{ave}

can be computed using the following equation. The derivation is analogous to the light curve intensity formula, Equation 3.2:

$$N_{ave} = \dot{N}_{star} \cdot t_{int} \cdot \left(\frac{d_{JWST}}{4d_o}\right)^2 \cdot \left(\frac{\rho}{2R}\right)^2 \cdot \frac{8}{3} \tag{3.11}$$

where $\dot{N}_{star}$ is the rate of photons being emitted by the star with a wavelength between $\lambda = 600$ nm and 28.3μm, calculated using the Stefan-Boltzmann law for a star with the same radius and surface temperature as the Sun. Furthermore, t_{int} is the integration time, d_o is the distance from the observer to the planetary system, and ρ and R are the radii of the planet and planetary orbit, respectively. The factor $\frac{8}{3}$ at the end of the equation is a correction factor from a homogeneously re-emitting sphere to a Lambertian sphere at full phase.

Entering the previously mentioned values into Equation 3.11, gives a $N_{ave} \approx 187$ photons. This results in a signal-to-noise ratio of $SNR_{max} = \sqrt{N_{ave}} \approx 13.7$. Of course, shot noise is not the only source of noise, as background starlight and instrumental noise will also pollute the signal. Therefore, the previously mentioned SNR should be treated as an upper limit.

4 MAPPING OF PLANETARY SURFACES

In the previous chapter, a linear transformation, mapping a planet's albedo map to its reflected light curve was derived, assuming that the orbital period, daily period, observer inclination and axial tilt were known. In this chapter we will try to recover the albedo map from the planet's reflected light curve by inverting this transformation. Furthermore, we will also try to recover the planet's axial tilt from the reflected light curve.

4.1 INVERTING TRANSFORMATION MATRIX

In Section 3.3, we saw that a linear transformation matrix T could be constructed, which can be used to express the planet's reflected light curve $\mathbf{f}$ as a function of the planet's albedo map $\mathbf{a}$. In matrix-vector form, the transformation was found to be $\mathbf{f} = T\mathbf{a}$. Thus, the next logical step in the retrieval of exo-planetary surfaces from their measured reflected light curves is finding the inverse of the linear transformation matrix T.

However, as T does not have to be a square matrix, an inverse does not always exist. Furthermore, even when T is chosen to be square, the inverse of T does not have to exist, since multiple albedo maps can give rise to the same reflected light curve [1]. Luckily, a generalized version of the inverse, the Moore-Penrose pseudo-inverse, does always exist and can be used to find the "best" fitting solution to Equation 3.6.

4.1.1 Moore-Penrose pseudo-inverse

Let $T : \mathbb{R}^{N_\phi \times N_\theta} \to \mathbb{R}^{N_t}$. Any albedo map vector $\mathbf{a}$ can be decomposed as the sum of a vector in $\text{Row}(T)$, say $\mathbf{a}_r$, and a vector in $\text{Null}(T)$, say $\mathbf{a}_n$. Similarly, any reflected light curve vector $\mathbf{f}$ can be decomposed as the sum of a vector in $\text{Row}\left(T^T\right) = \text{Col}(T)$, say $\mathbf{f}_c$, and a vector in $\text{Null}\left(T^T\right)$ [2]. Figure 4.1 shows how T transforms both $\mathbf{a}$ and $\mathbf{a}_r$ to $\mathbf{f}_c$, while $\mathbf{a}_n$ is mapped to $\mathbf{0}$. This means that as long as $\text{Null}(T)$ is not empty (i.e. T is not full rank), given a reflected light curve $\mathbf{f}$, we cannot determine a unique albedo map $\mathbf{a}$ such that $\mathbf{f} = T\mathbf{a}$.

1 For example, consider the face-on observation of two planets with zero axial tilt with exactly the same albedo map on the Northern hemisphere, but different albedo maps on the Southern hemisphere. Since the Southern hemispheres are never visible to the observer, they will never contribute to the light curve and both light curves will be identical.

2 Since the mock light curve data in this thesis is generated using Equation 3.6, $\mathbf{f}$ will always be in $\text{Col}(T)$. However, if shot noise is added to the light curve, this does not have to be the case.

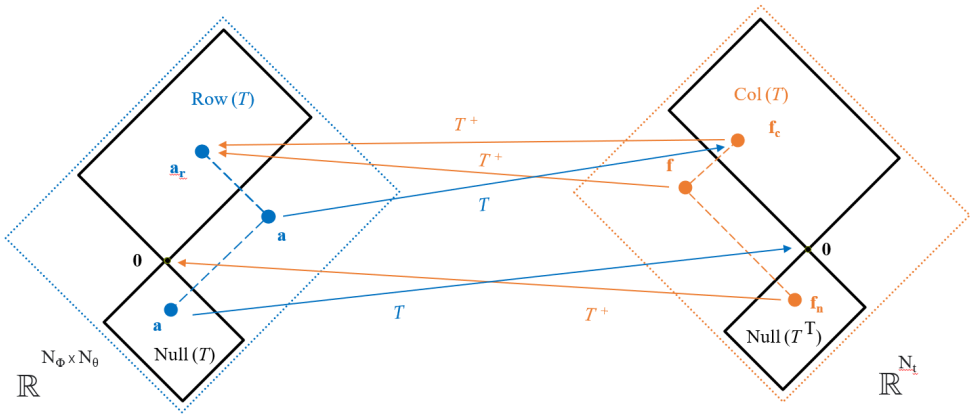


Figure 4.1: Geometric interpretation of the action of the transformation matrix T and its pseudo-inverse T^+.

However, every vector $\mathbf{f}_c \in \text{Col}(T)$ $\mathbf{f}_c$ comes from one and only one vector $\mathbf{a}_r \in \text{Row}(T)$ [see Strang, 2006, section 3.1]. The inverting operator that finds $\mathbf{a}_r$, is called the *Moore-Penrose pseudo-inverse*, denoted by T^+. It is defined such that:

$$T^+\mathbf{f} = \begin{cases} \mathbf{a}_r & \text{, if } \mathbf{f} \in \text{Col}\,(T) \\ \mathbf{0} & \text{, if } \mathbf{f} \in \text{Null}\,(T^T) \end{cases} \tag{4.1}$$

Thus, for an albedo map $\mathbf{a}$, the pseudo-inverse has the following property:

$$T^+\mathbf{f} = T^+(\mathbf{f}_c + \mathbf{f}_n) = T^+\mathbf{f}_c + T^+\mathbf{f}_n = \mathbf{a}_r + \mathbf{0} = \mathbf{a}_r \tag{4.2}$$

Therefore, the pseudo-inverse returns the best fitting solution up to addition from a vector from the null space $\mathbf{a}_n$. Furthermore, since $\text{Row}(T)$ is perpendicular to $\text{Null}(T)$, we have that for all solutions $\hat{\mathbf{a}}$:

$$\|\hat{\mathbf{a}}\|_2 = \|\mathbf{a}_r\|_2 + \|\mathbf{a}_n\|_2 \geq \|\mathbf{a}_r\|_2 \tag{4.3}$$

This means that the solution that found using the Moore-Penrose pseudo-inverse $\mathbf{a}_r$ is the also the solution with minimum Euclidean norm. A direct consequence of this is that any pixel on the planetary surface that has never contributed to the reflected light curve, will be mapped as a pixel with minimal albedo. As ocean is the surface type with minimal albedo, these pixels will be viewed as oceanic pixels. In this thesis, the Moore-Penrose pseudo-inverse of T will be computed using the method of singular value decomposition.

Singular Value Decomposition

Singular value decomposition is an extension of eigendecompostion, stating that we can write any matrix in the following way:

$$T = U\Sigma V^T$$

where U and V are unitary matrices, whose columns are eigenvectors of TT^T and T^TT, respectively. The matrix Σ is a rectangular diagonal matrix, whose

entries (denoted by σ) are the so-called singular values of T. If T has rank r, then also r singular values exist. A schematic depiction of singular value decomposition is shown in Figure 4.2.

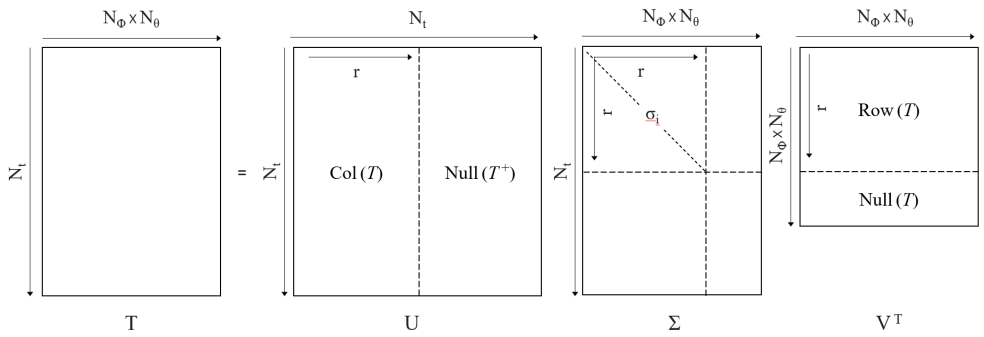


Figure 4.2: Schematic illustration of the singular value decomposition of the matrix T with rank r. The columns of the unitary matrices U and V span the column space and the row space of T, respectively. The matrix Σ is a diagonal matrix, whose entries σ_i are the singular values of T.

The singular value decomposition of a matrix can be used to compute its pseudo-inverse in the following way:

$$T^{+} = V\Sigma^{+}U^{T}$$

where Σ^{+} is the pseudo-inverse of Σ, which is found by replacing every singular value by its reciprocal and transposing the resulting matrix. In this thesis, the pseudo-inverses of the transformation matrices are computed using the NumPy function *pinv*, which uses the method of singular value decomposition.

However, using the Moore-Penrose pseudo-inverse to calculate the best fitting surface map, can lead to problems when noise is added to the reflected light curve. During the initial construction of T, numerical rounding errors can lead to small but non-zero entries in Σ that actually should have been zero instead. These "extra" singular values then lead to very big entries of Σ^{+} that also should have been zero. Therefore, unlike in Equation 4.2, f_n does not get mapped to zero by T^{+}, but to some other unwanted vector in $\mathbb{R}^{N_\phi \times N_\theta}$, which can lead to enormous errors during the albedo map recovery.

In order to prevent this from happening, very small singular values are assumed to be numerically equivalent to zero. Just enough, so that the amount of non-zero diagonal entries of Σ is approximately equal to the "true" rank of the transformation matrix. This process is called matrix truncation and will be done by setting all singular values that are smaller then $SVCR \times \sigma_{max}$ to zero, where $SVCR$ stands for Singular Value Cutoff Ratio and σ_{max} is the greatest singular value.

As can be seen in Figure 4.3 and Figure 4.4, where the singular values are displayed from largest to smallest, they form an L-shaped graph. The true rank of T can be approximated by setting all singular values smaller than the singular values in the elbow of the L-curve to zero [**?**]. For noiseless reflected light curve, T does not have to be truncated since the original albedo map always is a solution of Equation 3.6.

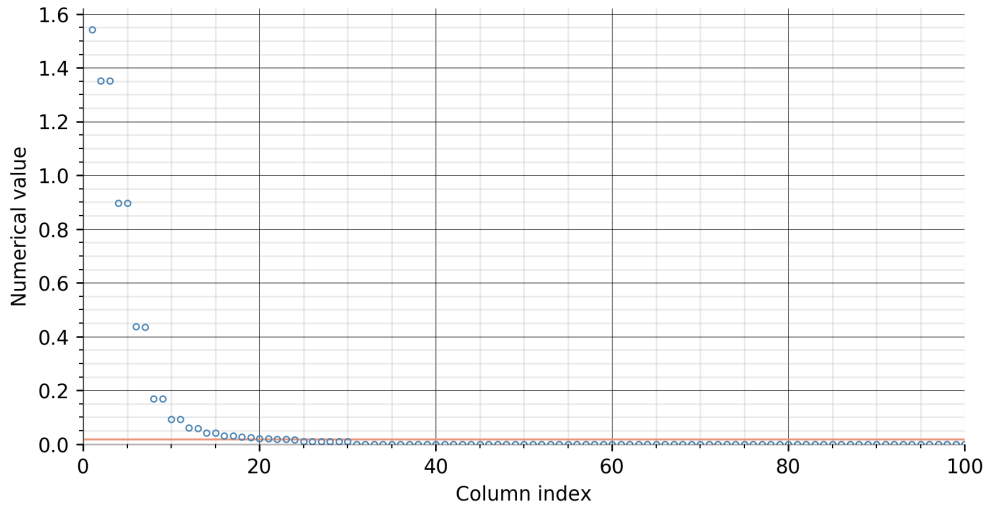


Figure 4.3: Graph of the first 100 singular values of the transformation matrix T for an edge-on observation with axial tilt parameters $\alpha = 0°$ and $\beta = 0°$. The orange line represents the singular value cutoff ratio (SVCR = 1/40).

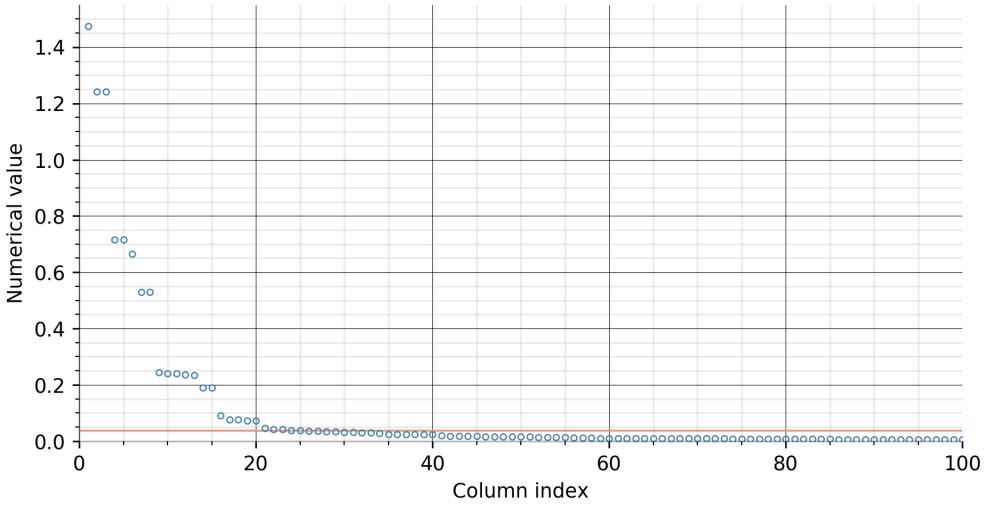


Figure 4.4: Graph of the first 100 singular values of the transformation matrix T for an edge-on observation with axial tilt parameters $\alpha = 90°$ and $\beta = 90°$. The orange line represents the singular value cutoff ratio (SVCR = 1/40).

4.1.2 Recovering albedo maps with known axial tilt

Combining our knowledge of reflected light curves and the Moore-Penrose pseudo-inverse, we can simulate the recovery of a planet's albedo map $\mathbf{a}$ when the axial tilt of the planet is known:

$$\mathbf{a}_r = [T(\alpha, \beta)]^+ \mathbf{f} = [T(\alpha, \beta)]^+ T(\alpha, \beta)\mathbf{a}$$

In the simulations in this report, the resolution of the albedo map is taken to be $N_\phi = 90$ and $N_\theta = 45$. Furthermore, the reflected light curve is sampled 4800 times: 200 time intervals of 24 hours in the planets orbit, unless specified otherwise.

Recovery without noise

The results of this process for noiseless light-curves are visible in Figure 4.5, Figure 4.6, Figure 4.7, Figure 4.8 for edge-on observations, and in Figure 4.9 for face-on observations. The first figure displaying the edge-on observations includes recoveries for four different values of $\beta = 0°, 30°, 60°, 90°$, while the other figures displaying the edge-on observations only include recoveries for three different values of $\beta = 30°, 60°, 90°$, as the quality of the recoveries with obliquity $\beta = 0°$ is independent of the equinox angle α. Since the quality of the recoveries for face-on observations is also independent of α, only one figure displaying the recovery for face-on observations is shown, displaying recoveries for four different values of β. Alongside the recovered albedo maps, maps displaying linear difference between the original and recovered map are presented. Furthermore, Figure 4.10 shows the remapping of a planet using significantly fewer sampling points of f. Lastly, Figure 4.11 and Figure 4.12 show the mean squared error (MSE) of the recovery as a function of the obliquity for edge-on observations and face-on observations, respectively.

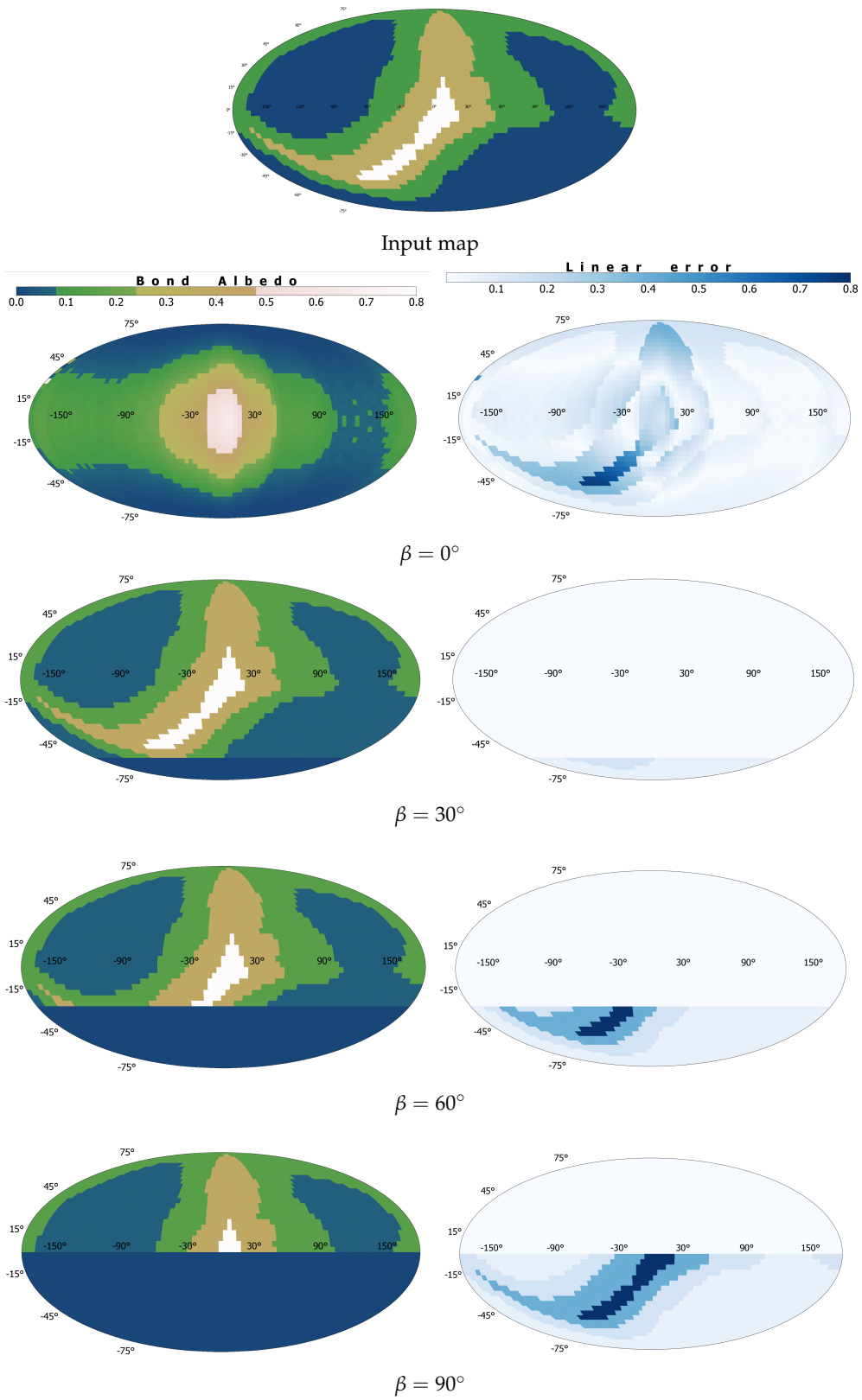


Figure 4.5: Albedo map retrievals for a noiseless, edge-on observation of a generated planet with equinox angle $\alpha = 0°$. Top: the original albedo map. Left: the recovered albedo maps for $\beta = 0°$, $30°$, $60°$ and $90°$. The pixels of the recovered maps are divided into four surface types: water (blue), vegetation (green), sand (yellow) and snow (white), based on their Bond albedo. The surface pixels that never have been observable, are mapped as oceanic pixels, as expected. Right: maps displaying the linear difference between the original and reconstructed albedo maps.

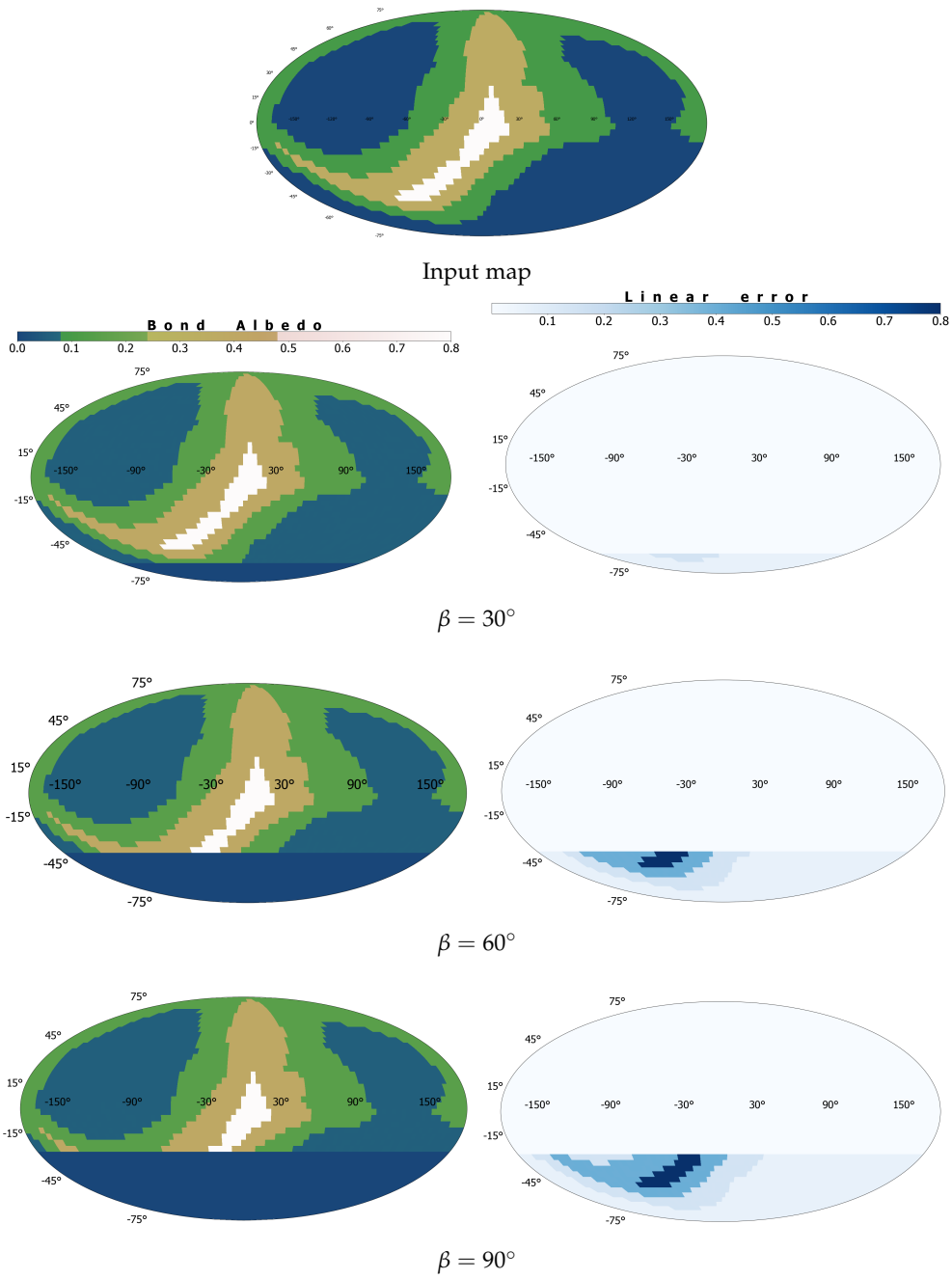


Figure 4.6: Albedo map retrievals for a noiseless, edge-on observation of a generated planet with equinox angle $\alpha = 30°$. The layout is the same as in Figure 4.5, expect the recovery for $\beta = 0°$ is not shown.

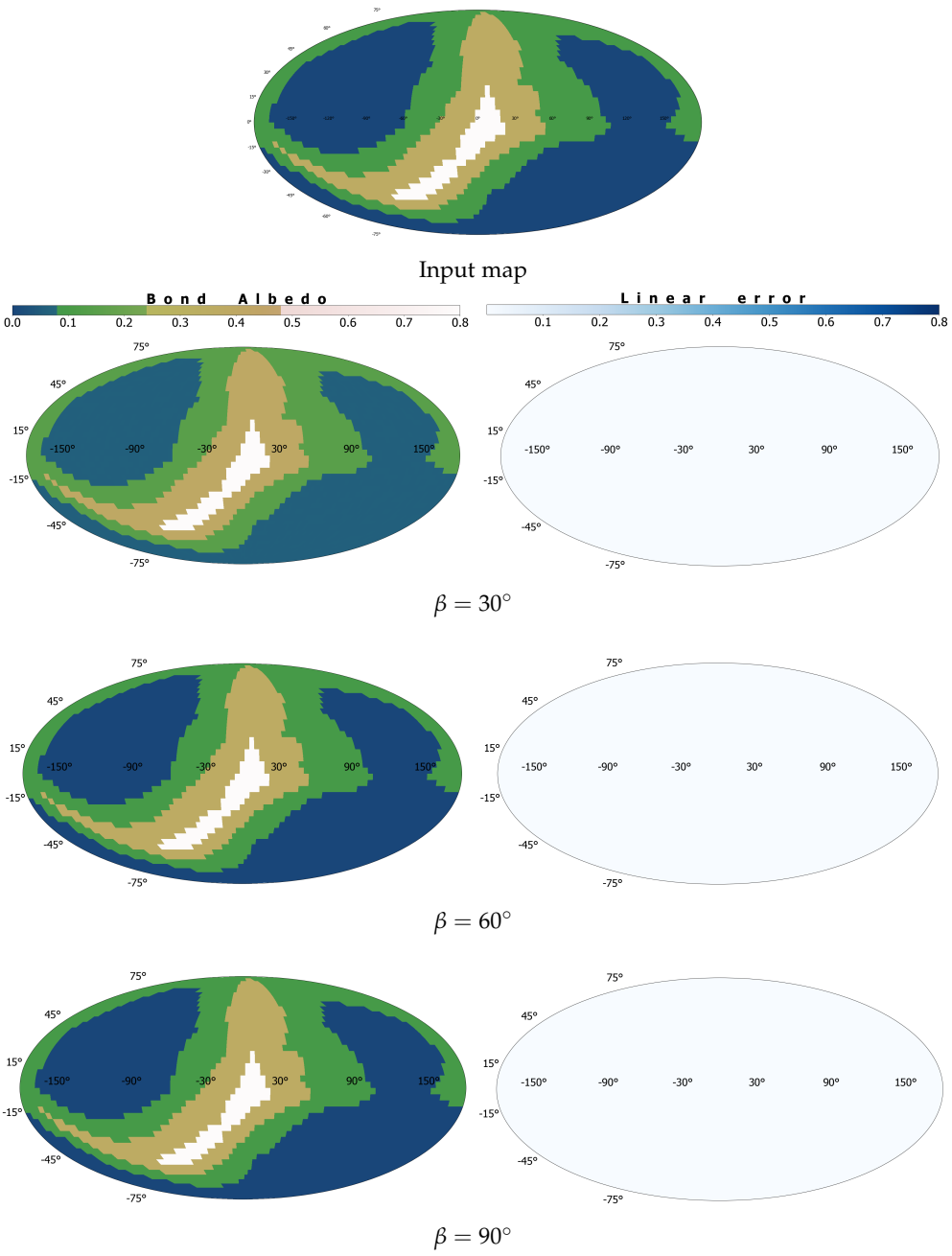


Figure 4.7: Albedo map retrievals for a noiseless, edge-on observation of a generated planet with equinox angle $\alpha = 60°$. The layout is the same as in Figure 4.6.

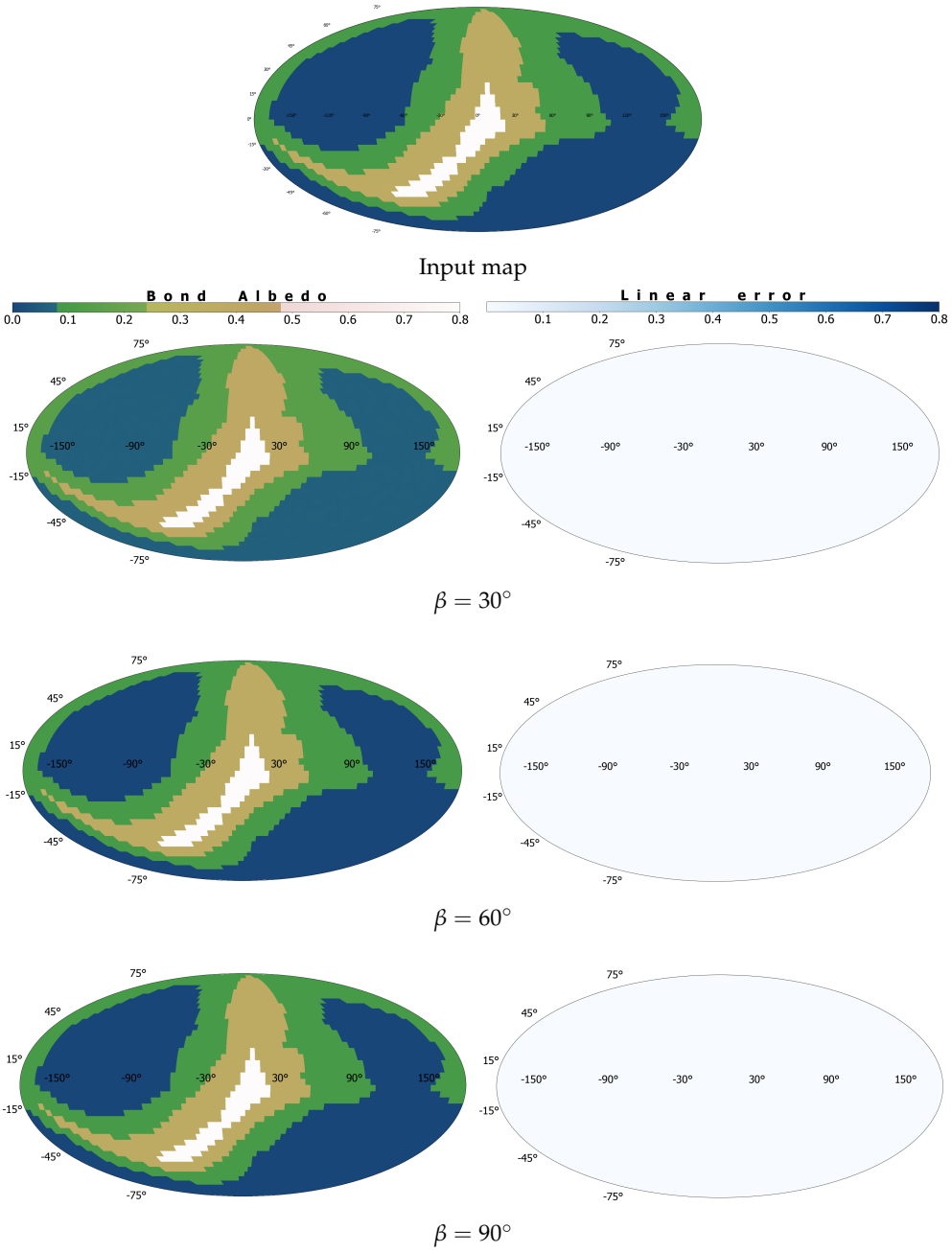


Figure 4.8: Albedo map retrievals for a noiseless, edge-on observation of a generated planet with equinox angle $\alpha = 90°$. The layout is the same as in Figure 4.6.

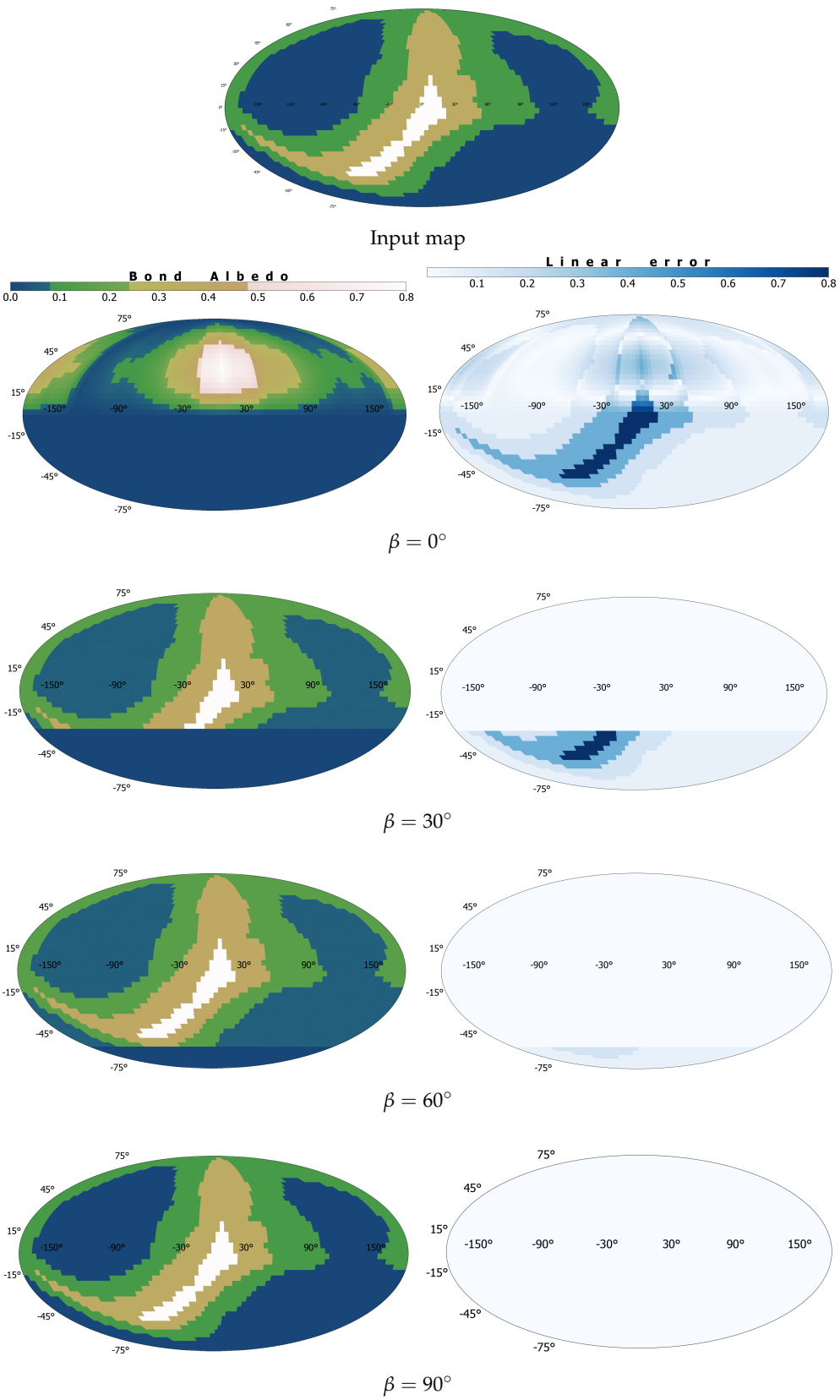


Figure 4.9: Albedo map retrievals for a noiseless, face-on observation of a generated planet. The layout is the same as in Figure 4.5.

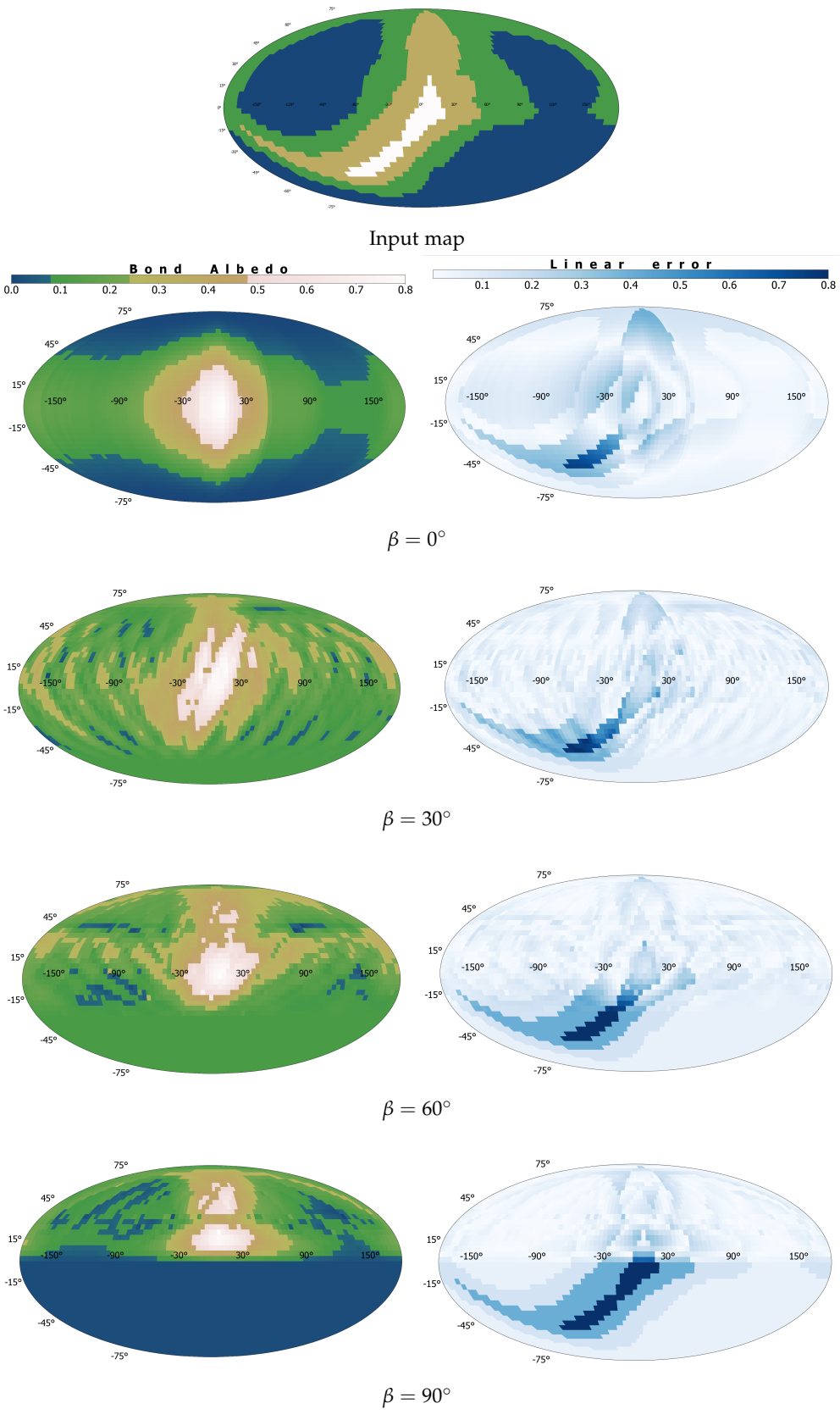


Figure 4.10: Albedo map retrievals for a noiseless, edge-on observation of a generated planet with equinox angle $\alpha = 0°$, using a reflected light curve that was only sampled at 10 different locations in the planet's orbit. The layout is the same as in Figure 4.5.

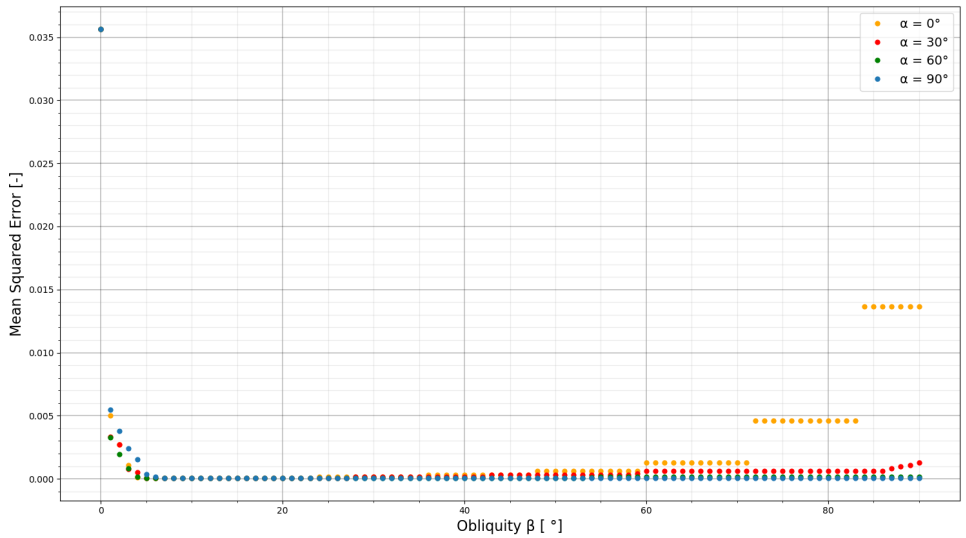


Figure 4.11: The retrieval accuracy for noiseless, edge-on observations of the input planet from Figure 4.5 as a function of the obliquity of the planet for four different equinox angles: $\alpha = 0°, 30°, 60°$ and $90°$. In order to speed up computation, an albedo map with $N_\phi = 30$ and $N_\theta = 15$ is used and the reflected light curve was only sampled for 10 different locations in the planet's orbit. The discrete behaviour of the graph corresponding to $\alpha = 0°$ is a direct result of the reduced resolution of the surface maps.

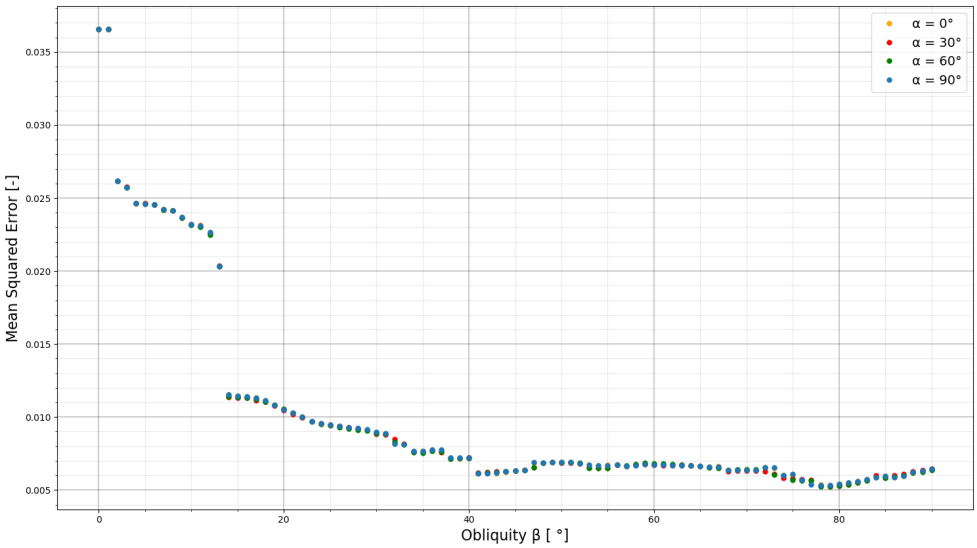


Figure 4.12: The retrieval accuracy for noiseless, face-on observations of the input planet from Figure 4.5 as a function of the obliquity of the planet for four different equinox angles: $\alpha = 0°, 30°, 60°$ and $90°$. However, most data points are not visible as the data points overlap each other. As in Figure 4.11, albedo map and light curve resolution were reduced.

Recovery with noise

As explained in Equation 4.1.1, when shot noise is added to the light curves, the truncation of Σ becomes essential if one still wants to accurately retrieve the planet's surface map. In Figure 4.13 the results of the process of remap-

ping an albedo map using a reflected light curve with added shot noise for different truncation values are shown. Furthermore, Figure 4.14 and Figure 4.15 show the MSE of the recovery as a function of the obliquity when shot noise is added to the reflected light curve.

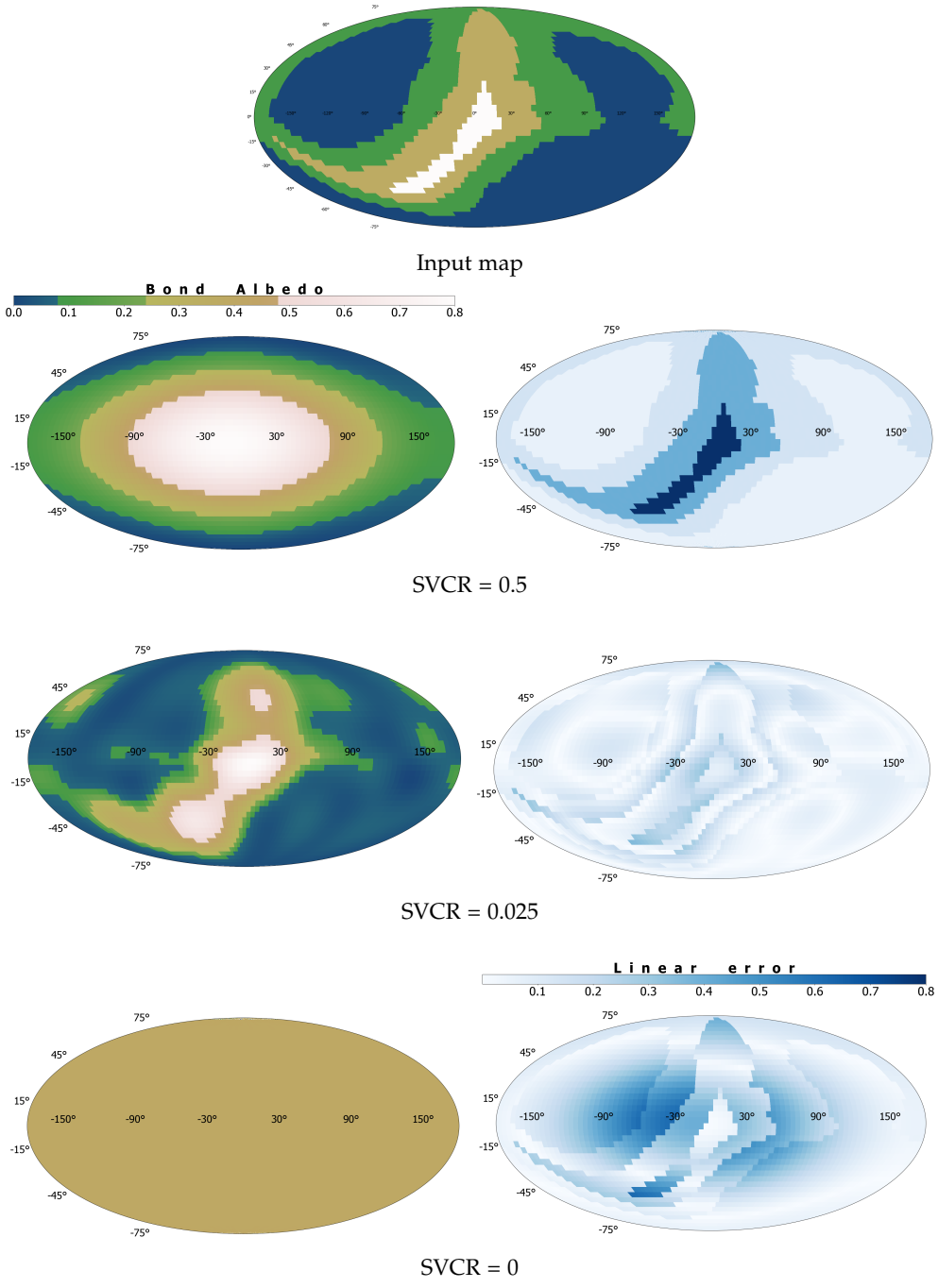


Figure 4.13: Albedo map retrievals for different truncation values for an edge-on observation of a generated planet with axial tilt $\alpha = 90°$ and $\beta = 90°$. Shot noise was added to the reflected light curve ($SNR = 13.7$). Top: the original albedo map. Left: the recovered albedo maps for $SVCR = 0.5$, 0.025 and 0. The pixels of the recovered maps are divided into four surface types: water (blue), vegetation (green), sand (yellow) and snow (white), based on their Bond albedo. Right: maps displaying the linear difference between the original and reconstructed albedo maps.

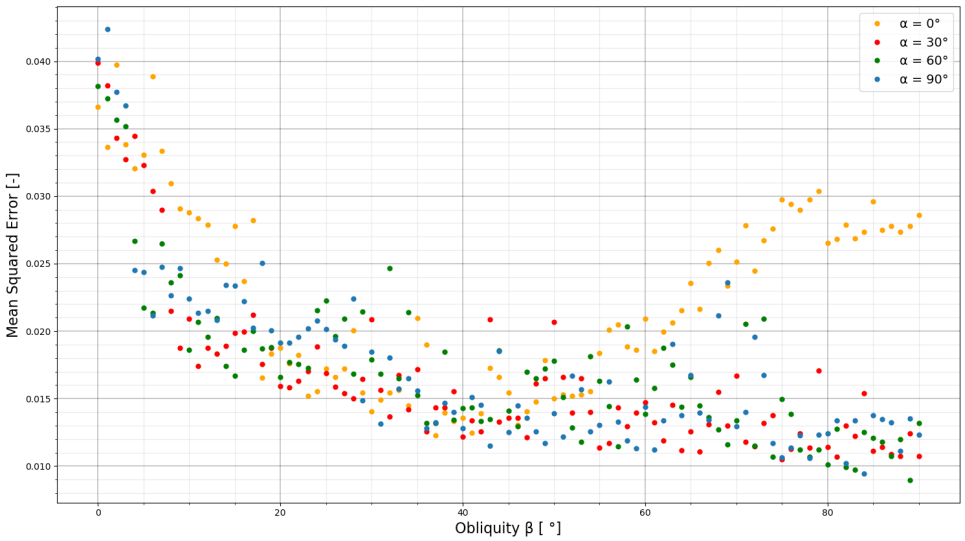


Figure 4.14: The retrieval accuracy of edge-on observations of the input planet from Figure 4.5 as a function of the obliquity of the planet for four different equinox angles: $\alpha = 0°$, $30°$, $60°$ and $90°$. Shot noise was added to the reflected light curve ($SNR = 13.7$) and the matrix was truncated ($SVCR = 0.025$). As in Figure 4.11, albedo map and light curve resolution were reduced.

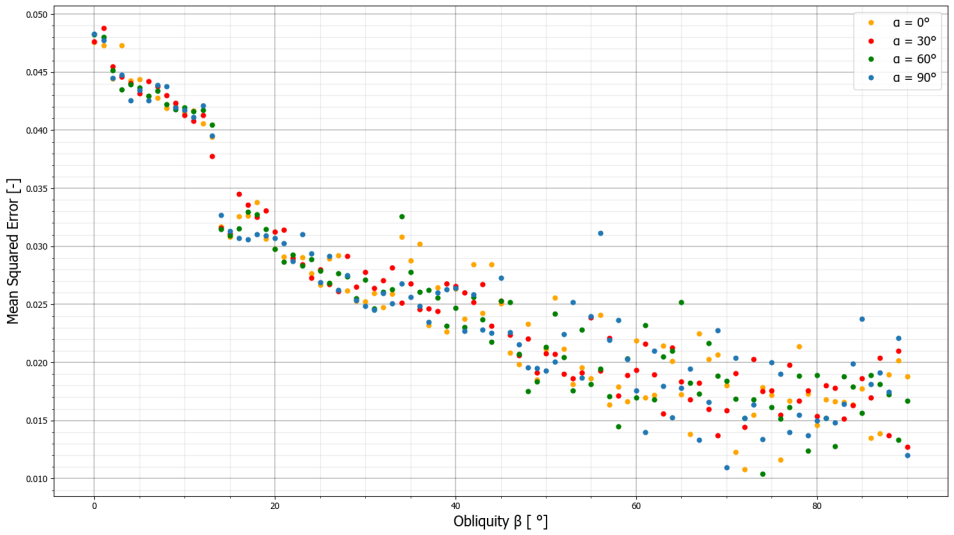


Figure 4.15: The retrieval accuracy of face-on observations of the input planet from Figure 4.5 as a function of the obliquity of the planet for four different equinox angles: $\alpha = 0°$, $30°$, $60°$ and $90°$. Shot noise was added to the reflected light curve ($SNR = 13.7$) and the matrix was truncated ($SVCR = 0.025$). As in Figure 4.11, albedo map and light curve resolution were reduced.

4.2 RECOVERING AXIAL TILT

So far, we have assumed that the equinox angle α and the obliquity β were know a priori. In this section, we do not assume that the axial tilt is known,

but we try to estimate the axial tilt by determining the best-fit value of α and β. Since TT^+ is the orthogonal projector onto $\mathrm{Col}(T)$, this will be done by minimizing the following Euclidean distance:

$$\begin{aligned} &\|\mathbf{f} - T(\alpha,\beta)\mathbf{a}_r(\alpha,\beta)\|_2 \quad = \\ &\|\mathbf{f} - T(\alpha,\beta)[T(\alpha,\beta)]^+\mathbf{f}\|_2 \end{aligned} \tag{4.4}$$

where $T(\alpha,\beta)$ and $\mathbf{a}_r(\alpha,\beta)$ are the transfer matrix and recovered albedo map, which are computed assuming that α and β are the axial tilt parameters of the planet.

However, if the matrix $T(\alpha,\beta)$ is full rank we run into a problem. Since for any full rank matrix T, TT^+ does not only map all column vectors of T to themselves, but is also equal to the identity matrix. Thus for full rank matrices, the distance defined in Equation 4.4 is always zero:

$$\begin{aligned} \|\mathbf{f} - T(\alpha,\beta)\mathbf{a}_r(\alpha,\beta)\|_2 &= \\ \|\mathbf{f} - T(\alpha,\beta)[T(\alpha,\beta)]^+\mathbf{f}\|_2 &= \\ \|\mathbf{f} - \mathbf{f}\|_2 &= 0 \end{aligned}$$

This problem can, however, be bypassed by truncating $[T(\alpha,\beta)]^+$, since this reduces the rank of the matrix.

The axial tilt is then estimated by minimizing the distance mentioned in Equation 4.4 for axial tilt parameters: $\alpha = 0°, 5° \ldots, 180°$ and $\beta = 0°, 5°, \ldots, 90°$ [3]. If the distance is minimized for multiple obliquities, the average of these values is taken. In Figure 4.16 and Figure 4.18, the recovery of the obliquity as a function of the obliquity for edge-on observations is shown for different equinox angles ($\alpha = 0°$, $30°$, $60°$ and $90°$). Figure 4.17 and Figure 4.19 show the same process for face-on observations. Furthermore, for Figure 4.16 and Figure 4.17 noiseless light curves are used, while for Figure 4.18 and Figure 4.19 shot noise is added to the light curve.

3 Increasing the resolution of the axial tilt recovery grid will, of course, also increase recovery accuracy. However, the reduced resolution was chosen to lower computation costs.

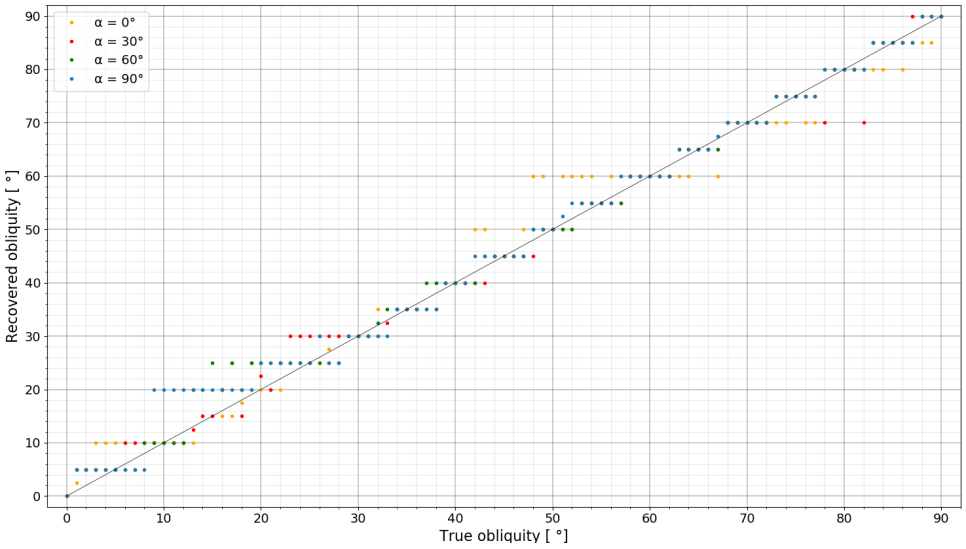


Figure 4.16: Graph illustrating the recovery of the obliquity β for edge-on observations of noiseless light curves, for four different equinox angles $\alpha = 0°, 30°, 60°$ and $90°$.

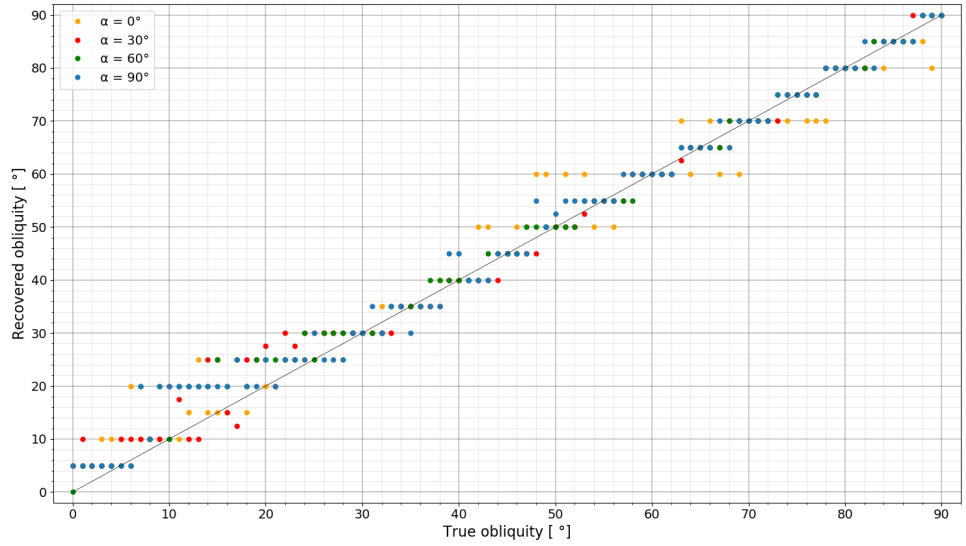


Figure 4.17: Graph illustrating the recovery of the obliquity β for edge-on observations of light curves with added shot noise (SNR = 13.7), for four different equinox angles $\alpha = 0°$, $30°$, $60°$ and $90°$.

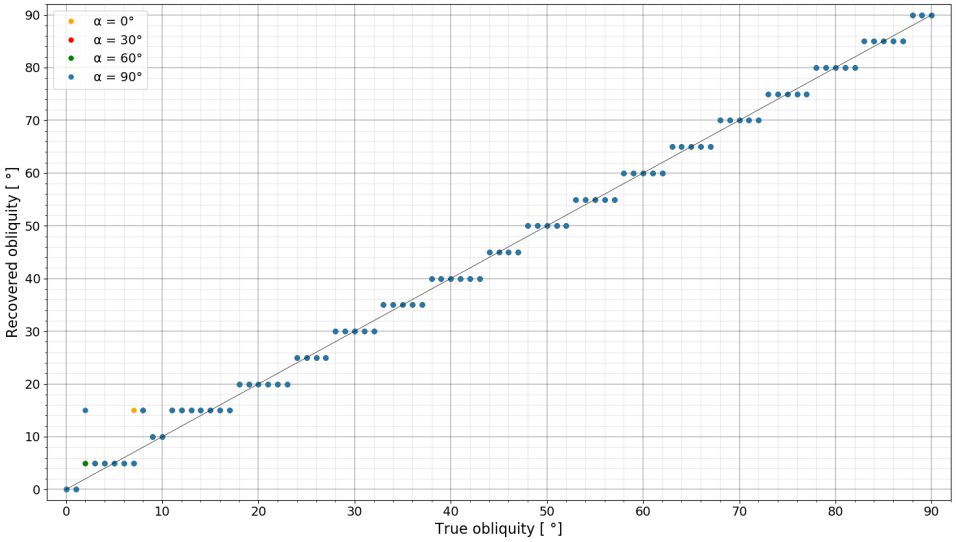


Figure 4.18: Graph illustrating the recovery of the obliquity β for face-on observations of noiseless light curves, for four different equinox angles $\alpha = 0°$, $30°$, $60°$ and $90°$. However, most data points are not visible as the data points overlap each other.

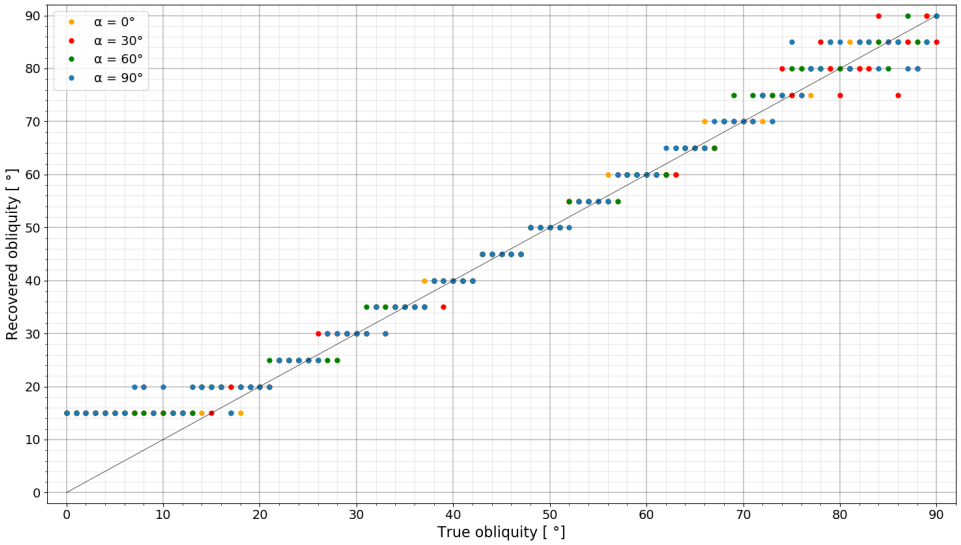


Figure 4.19: Graph illustrating the recovery of the obliquity β for face-on observations of light curves with added shot noise (SNR = 13.7), for four different equinox angles $\alpha = 0°$, $30°$, $60°$ and $90°$.

5 DISCUSSION & CONCLUSION

In this paper, we have developed an alternative approach to spin-orbit tomography: a method of reconstructing the two- dimensional planetary surface using the daily and yearly variation of the scattered starlight that gets reflected by exoplanets. Applying our method to mock light curve data, we have demonstrated that our method works for self-generated surface maps consisting of four surface types: ocean, vegetation, sand and snow. Even when shot noise is added to the light curve ($SNR \approx 14$), a detailed surface map can be retrieved, which accurately contains the planet's continents and oceans. Furthermore, we also found that the planet's axial tilt can be estimated using this method.

However, one should keep in mind that several simplifying assumption were made: cloudless surface, Lambertian reflection. Also, only shot noise was taken into account, which is only a lower bound to the noise that is present during actual light curve measurement. Not omitting clouds, other types of reflection and other sources of noise may significantly reduce the retrieval quality.

Furthermore, it should be noted that even though we have only considered edge-on and face-on observations, our code works for other observer inclinations as well.

RECOVERY OF PLANETARY SURFACES

The accuracy of the map retrieved using our approach to spin-orbit tomography, depends primarily on two factors: the number of observations and planet's axial tilt.

The first one should be obvious, as more observations of the planet's reflected light curve should lead to more information about the planetary surface and thus better retrieval. However, as the number of observations surpasses the number of recovered surface pixels, the MSE does not seem to decrease much further. This diminished increase in retrieval accuracy can probably be attributed to the fact that the maximum rank of the transformation matrix is capped off above by both the number of observations and the number of surface pixels. Thus, the rank of the matrix, and with it the amount of extractable information, will probably not increase much further after the number of observations has become greater than the number of reconstructed surface pixels. This means that after one has acquired light curve data, one should adjust the resolution of the recovery such that the transformation matrix is (approximately) square. Since for optimal retrieval quality, one wants to have high resolution (to retrieve as many surface features as possible), while keeping the MSE small (to make sure that

the retrieved surface features are actually present), which is achieved for (approximately) square transformation matrices.

The retrieval quality also seems to strongly depend on the axial tilt of the planet. For some equinox angles α and obliquities β, the singular values seem to converge significantly faster than for others. Indicating that the ranks of these transformations, and thus the maximum attainable recovery qualities, are significantly lower than others.

Differences in retrieval quality for different spin-axis configurations can also directly be seen when looking at the mean squared errors of the recovered surface maps. For both edge-on and face-on observations, the retrieval quality seems to increase as the obliquity of the exoplanet increases. However, for edge-on observations of planets with very small equinox angles, the retrieval quality diminishes again after the obliquity surpasses $\beta \approx 45°$.

The reduced surface retrieval quality of exoplanets with smaller obliquity for edge-on observation can be attributed to the reduced accuracy in the retrieval of latitudinal surface features, which is the result of a lack of seasonal variation in the reflected light curve of the planet. The obliquity of an exoplanet causes different latitudinal regions to be illuminated at different locations in the planet's orbit. This results in seasonal differences in the light curve intensity based on latitudinal differences in the albedo map, and thus allowing retrieval of latitudinal surface features.

When no noise is present, an obliquity of $\beta = 5°$ produces sufficient seasonal variation for maximizing retrieval quality. However, when shot noise is added, the retrieval quality is not yet maximized for $\beta = 5°$, and it keeps increasing as the obliquity increases. This can be explained by the increase in seasonal variation of the reflected light curve as the obliquity of a planet increases. Therefore, when a planet has greater obliquity, less of the variation gets obscured by noise. In the special case when the planet has no axial tilt ($\beta = 0$), no seasonal variation is present, thus no latitudinal distinction can be made and the recovered surface map is perfectly symmetrical around the equator.

The reduced retrieval quality for edge-on observations of planets with small equinox angle but greater obliquity and for face-on observations with smaller obliquity can be attributed to another phenomenon. In both cases, significant parts of the planetary surface are never visible to the observer during the planet's orbit and thus never contribute to the reflected light curve. Hence, no information about these surface pixels is known and they cannot be accurately retrieved. In this thesis, these parts of the planetary surface are mapped as ocean, as the Moore-Penrose pseudo-inverse always returns the best fitting solution with minimum norm and oceanic pixels have minimum albedo.

RECOVERY OF AXIAL TILT

Unlike the orbital and rotation period of the planet, the axial tilt of the planet is often not known a priori. However, all three parameters need to be known for the computation of the (right) transformation matrix. Therefore, we have

also tried to retrieve the axial tilt of the planet. This was done by minimizing the following distance on a discretized grid of values for α and β:

$$\|\mathbf{f} - T(\alpha, \beta)[T(\alpha, \beta)]^{+}\mathbf{f}\|_2$$

The process was generally very successful. For both edge-on and face-on observations, the recovered obliquity was almost always equal to the obliquity value on the grid that was closest to the true obliquity of the exoplanet. However, analogous to the retrieval quality, the axial tilt recovery was less accurate when the exoplanet had small obliquity.

RECOMMENDATIONS

Some recommendations for future research are:

- Include a time-dependent surface map (e.g. clouds and seasonal changes)
- Include other types of reflection (e.g. non-Lambertian reflection of water)
- Include more types of noise (e.g. a layer of Gaussian noise due to instrumental noise and background noise from other celestial bodies)
- Adjust spin-orbit tomography for other inhomogeneous surfaces (e.g. adding more surface types found on earth, or adjusting the method for exoplanets that are not Earth-like)
- Try an equal-area pixelization scheme of the surface map instead of equirectangular pixelization scheme that we have used in this thesis (e.g. HEALPix pixelization)
- Since Earth-like exoplanets will probably have reasonably thick atmospheres, research the effects of Rayleigh scattering on the retrieval quality

BIBLIOGRAPHY

Aizawa, M., Kawahara, H., and Fan, S. (2020). Global mapping of an exo-earth using sparse modeling. *The Astrophysical Journal*, 896(1):22.

Betts, A. K. and Ball, J. H. (1997). Albedo over the boreal forest. *Journal of Geophysical Research: Atmospheres*, 102(D24):901–28.

Cowan, N. B. and Agol, E. (2008). Inverting phase functions to map exoplanets. *The Astrophysical Journal*, 678(2).

Cowan, N. B. and Fujii, Y. (2021). Mapping exoplanets. *Handbook of Exoplanets*, page 1–18.

Dorian (2017). Représentation des angles d'euler avec tikz. https://blog.dorian-depriester.fr/latex/tikz/representation-des-angles-deuler-avec-tikz. Last accessed on 31-07-2021.

Dressing, C. D. and Charbonneau, D. (2015). The occurrence of potentially habitable planets orbiting m dwarfs estimated from the fullkepler-dataset and an empirical measurement of the detection sensitivity. *The Astrophysical Journal*, 807(1):45.

Fan, S., Li, C., Li, J., Bartlett, S., Jiang, J. H., Natraj, V., Crisp, D., and Yung, Y. L. (2019). Earth as an exoplanet: A two-dimensional alien map. *The Astrophysical Journal*, 882(1).

Farr, B., Farr, W. M., Cowan, N. B., Haggard, H. M., and Robinson, T. (2018). exocartographer: A bayesian framework for mapping exoplanets in reflected light. *The Astronomical Journal*, 156(4):146.

Goldstein, H. (1997). *Classical mechanics*. Addison-Wesley.

Haus, R., Kappel, D., Tellmann, S., Arnold, G., Piccioni, G., Drossart, P., and Häusler, B. (2016). Radiative energy balance of venus based on improved models of the middle and lower atmosphere. *Icarus*, 272:178–205.

Kawahara, H. and Fujii, Y. (2010). Global mapping of earth-like exoplanets from scattered light curves. *The Astrophysical Journal*, 720(2):1333–1350.

Kawahara, H. and Fujii, Y. (2011). Mapping clouds and terrain of earth-like planets from photometric variability: Demonstration with planets in face-on orbits. *The Astrophysical Journal*, 739(2).

Kawahara, H. and Masuda, K. (2020). Bayesian dynamic mapping of an exo-earth from photometric variability. *The Astrophysical Journal*, 900(1):48.

Mallama, A. (2017). The spherical bolometric albedo of planet mercury.

Mayor, M. and Queloz, D. (1995). A jupiter-mass companion to a solar-type star. *Nature*, 378(6555):355–359.

Mogensen, T. (2010). Planet map generation by tetrahedral subdivision. In Pnueli, A., Virbitskaite, I., and Voronkov, A., editors, *Perspectives of Systems Informatics*, Lecture notes in computer science, pages 306–318, Switzerland. Springer. 7th International Andrei Ershov Memorial Conference on Perspectives of System Informatics, PSI 2009 ; Conference date: 15-06-2009 Through 19-06-2009.

NSIDC (2020). Thermodynamics: Albedo. https://nsidc.org/cryosphere/seaice/processes/albedo.html. Last accessed on 31-07-2021.

Strang, G. (2006). *Linear algebra and its applications*. Thomson.

Tetzlaff, G. (1983). Albedo of the Sahara. Satellite Meas. of Radiation Budget Parameters.

Tuomi, M., Jones, H. R. A., Barnes, J. R., Anglada-Escudé, G., and Jenkins, J. S. (2014). Bayesian search for low-mass planets around nearby m dwarfs – estimates for occurrence rate based on global detectability statistics. *Monthly Notices of the Royal Astronomical Society*, 441(2):1545–1569.

Winn, J. N., Holman, M. J., Torres, G., Mccullough, P., Johns-Krull, C., Latham, D. W., Shporer, A., Mazeh, T., Garcia-Melendo, E., Foote, C., and et al. (2008). The transit light curve project. ix. evidence for a smaller radius of the exoplanet xo-3b. *The Astrophysical Journal*, 683(2):1076–1084.

A APPENDIX

CODE FOR SURFACE MAP GENERATION

```
import matplotlib.pyplot as plt
import numpy as np
import random
from mpl_toolkits.mplot3d import Axes3D
from scipy.spatial import Delaunay, ConvexHull
from matplotlib import cm
from matplotlib.colors import ListedColormap,
    LinearSegmentedColormap



def create_sphere1(r, res):
    '''
    creates points on a sphere in the form:
        [[ x1.  y1.  z1.]
         ....
         [ xn.  yn.  zn.]]
    '''
    phi = np.linspace(0, 2*np.pi, 2*res)
    theta = np.linspace(0, np.pi, res)

    theta, phi = np.meshgrid(theta, phi)

    theta_flat = np.ndarray.flatten(theta, order = 'C')
    phi_flat = np.ndarray.flatten(phi, order = 'C')

    r_pre = r*np.sin(theta_flat)
    x = np.cos(phi_flat)*r_pre
    y = np.sin(phi_flat)*r_pre
    z = r*np.cos(theta_flat)

    coordinates = np.vstack((x,y,z)).transpose()

    return coordinates, theta, phi



def pnt_in_cvex_hull(hull, pnt):
    '''
    Checks if `pnt` is inside the convex hull.
    '''
    new_hull = ConvexHull(np.concatenate((hull.points, [pnt])))
    if np.array_equal(new_hull.vertices, hull.vertices):
        return True
    else:
```

```
47          return False
48
49
50
51  def height(p, vertices, seeds, altitudes):
52      max_dist = 0
53      '''
54      finding shortest edge
55      '''
56      for i in range(3):
57          for j in range(i+1, 4):
58              dist = np.linalg.norm(vertices[i] - vertices[j])
59              if dist ¿ max_dist:
60                  max_dist = dist
61                  furthest = np.array([i, j])
62
63      '''
64      reordering everything
65      '''
66      vertices[[0, furthest[0]]] = vertices[[furthest[0], 0]]
67      vertices[[1, furthest[1]]] = vertices[[furthest[1], 1]]
68
69      seeds[[0,furthest[0]]] = seeds[[furthest[0], 0]]
70      seeds[[1,furthest[1]]] = seeds[[furthest[1], 1]]
71
72      altitudes[[0, furthest[0]]] = altitudes[[furthest[0], 0]]
73      altitudes[[1, furthest[1]]] = altitudes[[furthest[1], 1]]
74
75      '''
76      creating new edge
77      new altitude = average altitude + random(-0,05; 0,05)*sqrt(
        distance)
78      '''
79      v_new = (vertices[0]+vertices[1])/2
80      s_new = (seeds[0]+seeds[1])/2
81      random.seed(s_new)
82      a_new = (altitudes[0]+ altitudes[1])/2 + 0.01*(random.random()
        - 0.5)*max_dist**1.5
83
84
85      '''
86      finding in which tetrahedon our point p is
87      '''
88      tetra = np.copy(vertices)
89      tetra[1] = v_new
90      hull = ConvexHull(tetra)
91
92      if pnt_in_cvex_hull(hull, p):
93          vertices[1] = v_new
94          seeds[1] = s_new
95          altitudes[1] = a_new
96      else:
97          vertices[0] = v_new
98          seeds[0] = s_new
99          altitudes[0] = a_new
100
101     '''
102     stop if resolution is great enough
103     '''
104     if max_dist ¡ 0.001:
```

```
        return np.sum(altitudes)/4
    else:


        height(p, vertices, seeds, altitudes)
    return np.sum(altitudes)/4




'''
initialize sphere and final altitudes vector
'''
res = 45
sphere, theta, phi = create_sphere1(1, res)
alts = np.zeros(len(sphere[:, 0]))
#alts2 = alts.copy()
#alts3 = alts.copy()
#alts4 = alts.copy()
#alts5 = alts.copy()
#alts6 = alts.copy()
albedo_map = np.zeros(len(sphere[:, 0]))
#albedo_map2 = albedo_map.copy()
#albedo_map3 = albedo_map.copy()
#albedo_map4 = albedo_map.copy()
#albedo_map5 = albedo_map.copy()
#albedo_map6 = albedo_map.copy()

albedo_map_comp = albedo_map.copy()
'''
initialize vertices, seeds and altitudes(sea level)
'''


'''
calculate altitude for every point on sphere
'''
for i in range(len(sphere[:, 0])):
    vertices_ = np.array([[-2   ,  2.1,  2.2], [ 2   , -2.1,  2.2],
    [ 2   ,  2.1, -2.2],[-2.3,  -2.4,  -2.5]])

    seeds_ , altitudes_ = np.array([8.3e5, 2.5e5, 3.65e5, 7.4e5]),
     np.array([0., 0. , 0., 0.])
    point = sphere[i, :]
    alt = height(point, vertices_, seeds_, altitudes_)
    alts[i] = alt

#    vertices_ = np.array([[-2   ,  2.1,  2.2], [ 2   , -2.1,  2.2],
     [ 2   ,  2.1, -2.2],[-2.3,  -2.4,  -2.5]])
#    seeds_ , altitudes_ = np.array([6.5e6, 2.8e6, 3.6e6, 7.3e6]),
     np.array([0., 0. , 0., 0.])
#    point = sphere[i, :]
#    alt2 = height(point, vertices_, seeds_, altitudes_)
#    alts2[i] = alt2
#
#    vertices_ = np.array([[-2   ,  2.1,  2.2], [ 2   , -2.1,  2.2],
     [ 2   ,  2.1, -2.2],[-2.3,  -2.4,  -2.5]])
#    seeds_ , altitudes_ = np.array([6.7e7, 4.5e7, 9.8e7, 3.1e7]),
     np.array([0., 0. , 0., 0.])
#    point = sphere[i, :]
```

```
#     alt3 = height(point, vertices', seeds', altitudes')
#     alts3[i] = alt3
#
#     vertices' = np.array([[-2   ,  2.1,  2.2], [ 2   , -2.1,  2.2],
      [ 2   ,  2.1, -2.2],[-2.3,  -2.4,  -2.5]])
#     seeds' , altitudes' = np.array([7.9e5, 1.3e5, 0.6e5, 4.2e5]),
      np.array([0., 0 , 0., 0.])
#     point = sphere[i, :]
#     alt4 = height(point, vertices', seeds', altitudes')
#     alts4[i] = alt4
#
#     vertices' = np.array([[-2   ,  2.1,  2.2], [ 2   , -2.1,  2.2],
      [ 2   ,  2.1, -2.2],[-2.3,  -2.4,  -2.5]])
#     seeds' , altitudes' = np.array([2.7e6, 1.6e6, 5.3e6, 0.4e6]),
      np.array([0., 0.5 , -0.25, -0.25])
#     point = sphere[i, :]
#     alt5 = height(point, vertices', seeds', altitudes')
#     alts5[i] = alt5
#
#     vertices' = np.array([[-2   ,  2.1,  2.2], [ 2   , -2.1,  2.2],
      [ 2   ,  2.1, -2.2],[-2.3,  -2.4,  -2.5]])
#     seeds' , altitudes' = np.array([7.5e7, 4.2e7, 0.8e7, 3.6e7]),
      np.array([0., 0. , 0., 0.])
#     point = sphere[i, :]
#     alt6 = height(point, vertices', seeds', altitudes')
#     alts6[i] = alt6
#     print(i)


'''
calculate albedo map
'''


ave = np.sum(alts)/len(alts)
alts = alts-ave
max'alt = max(alts)
min'alt = min(alts)

alt'grid1 = alts.reshape((res, 2*res), order = 'F')
alt'grid1' = np.copy(alt'grid1)

alts1 = np.ndarray.flatten(alt'grid1', order = 'F')


for i in range(len(sphere[:, 0])):
    if alts1[i] ¿= 0.8*max'alt:
        albedo'map[i] = 0.8  #snow
    elif alts1[i] ¡ 0.0:
          albedo'map[i] = 0.06 #ocean
    elif (alts1[i] ¿= 0.4*max'alt and alts1[i] ¡ 0.8*max'alt):
        albedo'map[i] = 0.4 #soil
    elif (alts1[i] ¿= 0.0 and alts1[i] ¡ 0.4*max'alt):
        albedo'map[i] = 0.15  #forest

'''
create corresponding meshgrid
'''
alt'grid = alts1.reshape((res, 2*res), order = 'F')
albedo'map'grid = albedo'map.reshape((res, 2*res), order = 'F')
```

```
213
214 new_gist_earth = cm.get_cmap('gist_earth', 4096)
215 newcolors = np.vstack((new_gist_earth(np.linspace(0, 0.35, 256)),
        new_gist_earth(np.linspace(0.45, 1, 256))))
216 newcmp = ListedColormap(newcolors)
217 albedo_colors = np.vstack((new_gist_earth(np.linspace(0.15, 0.22,
        100)), new_gist_earth(np.linspace(0.45, 0.55, 200)),
        new_gist_earth(np.linspace(0.7, 0.8, 300)), new_gist_earth(np.
        linspace(0.95, 1, 400))))
218 albedo_cmp = ListedColormap(albedo_colors)
219 albedo_colors1 = np.vstack((new_gist_earth(np.linspace(0.15, 0.22,
         140)), new_gist_earth(np.linspace(0.45, 0.55, 180)),
        new_gist_earth(np.linspace(0.7, 0.8, 280)), new_gist_earth(np.
        linspace(0.95, 1, 400))))
220 albedo_cmp1 = ListedColormap(albedo_colors1)
221 lon = np.linspace(-np.pi, np.pi, 2*res)
222 lat = np.linspace(-np.pi/2., np.pi/2, res)
223 Lon,Lat = np.meshgrid(lon,lat)
224
225
226 fig1 = plt.figure(1, figsize=(20, 9))
227 ax1 = fig1.add_subplot(111, projection='mollweide')
228 im1 = ax1.pcolormesh(Lon,Lat, np.flipud(alt_grid), cmap= newcmp)
229 plt.colorbar(im1,shrink=0.75, aspect=40)
230 fig1.show()
```

CODE FOR CREATING MOCK LIGHT CURVE DATA AND ALBEDO MAP RECOVERY

```
1
2
3 import matplotlib.pyplot as plt
4 import math
5 import numpy as np
6
7
8
9 '''
10 Parameters
11 '''
12 res = 45
13 rho = 6.371e6
14 R = 1.496e11
15 alpha = 60/180*np.pi
16 beta = 60/180*np.pi
17
18 day = 24
19 year = 365*day
20 omega_day =  2*np.pi/day
21 omega_year = 2*np.pi/year
22
23
24 delta_t = 1
25 delta_phi = np.pi/res
26 delta_theta = np.pi/res
27 delta_Omega = delta_phi*delta_theta
```

```
28
29 hours = np.arange(0, 24)
30 time_array = hours.copy()
31
32 N_ave = 187
33
34 for i in range(1, 200):
35     time_array = np.append(time_array, hours + i*year/200)
36
37 time_res = len(time_array)
38
39 '''
40 Albedo-map
41 '''
42 A = np.ones(2*res**2)
43 time = np.linspace(0, 1, round(year/delta_t))*365
44
45
46 '''
47 Euler rotation matrices
48 '''
49 def positive(arg):
50     return (arg + abs(arg))/2
51
52 def y_rotation(angle):
53     Y = np.array([[np.cos(angle), 0, np.sin(angle)],[0, 1, 0],[-np
       .sin(angle), 0, np.cos(angle)]])
54     return Y
55
56 def z_rotation(angle):
57     Z = np.array([[np.cos(angle), -np.sin(angle), 0],[np.sin(angle
       ), np.cos(angle), 0], [0, 0, 1]])
58     return Z
59
60
61
62 R_equinox = z_rotation(alpha)
63 R_tilt = y_rotation(beta)
64 R_axial = np.matmul(R_equinox, R_tilt)
65
66 '''
67 Initialize transfromation matrices (edge-on, face-on)
68 '''
69 T = np.zeros((time_res, 2*res**2))
70
71 T1 = T.copy()
72
73 '''
74 Observer
75 '''
76 o_vec = np.array([1, 0, 0])
77 o_vec_1 = np.array([0, 0, 1])
78
79
80 '''
81 Compute matrix elements
82 '''
83 for i in range(len(time_array)):
84     t = time_array[i]
```

```
    r_vec = np.array([-np.cos(omega_year*t), -np.sin(omega_year*t), 0])
    R_daily = z_rotation(omega_day*t)
    daily_rotation = np.matmul(R_axial,  R_daily)

    for j in range(2*res**2):
        phi = math.floor(j/res)*delta_phi
        theta = (j % res)*delta_theta

        s_vec = np.array([np.cos(phi)*np.sin(theta), np.sin(phi)*np.sin(theta), np.cos(theta)])
        s_vec_rotated = np.matmul(daily_rotation,s_vec)

        r_s = np.dot(r_vec, s_vec_rotated)
        s_o = np.dot(s_vec_rotated, o_vec)
        s_o_1 = np.dot(s_vec_rotated, o_vec_1)


        illuminated = positive(r_s)
        visible = positive(s_o)
        visible_1 = positive(s_o_1)

        T[i][j] = illuminated*visible*np.sin(theta)*delta_Omega
        T1[i][j] = illuminated*visible_1*np.sin(theta)*delta_Omega



'''
Compute light curves
'''
f_curve_edge = np.matmul(T, albedo_map)*rho**2/(R**2*np.pi)
f_curve_face = np.matmul(T1, albedo_map)*rho**2/(R**2*np.pi)


noise_poisson = np.random.poisson(N_ave, time_res)/N_ave
noise_gauss = np.random.normal(0,1,time_res)

#f_curve_edge_noisy = f_curve_edge  + noise_gauss*max(f_curve_edge)/100
#f_curve_face_noisy = f_curve_face + noise_gauss*max(f_curve_face)/100

f_curve_edge_noisy = np.multiply(f_curve_edge, noise_poisson)
f_curve_face_noisy = np.multiply(f_curve_face, noise_poisson)

'''
Save transfer matrices for later use
'''
#np.savez_compressed('Albedo_a90_b90_20d_res_15.npz', edge = T, face = T1)


'''
Compute and save SVD
'''
SVD_edge = np.linalg.svd(T, compute_uv= False)
SVD_face = np.linalg.svd(T1, compute_uv= False)


'''
```

```
Compute inverse transfer matrix
'''
T_pinv_edge = np.linalg.pinv(T, rcond = 0.025)
T_pinv_face = np.linalg.pinv(T1, rcond = 0.025)
A_edge = np.matmul(T_pinv_edge, f_curve_edge)
A_face = np.matmul(T_pinv_face, f_curve_face)

#
edge_factor = 0.8/(max(A_edge) - min(A_edge))
face_factor = 0.8/(max(A_face) - min(A_face))


A_edge_scaled = (np.matmul(T_pinv_edge, f_curve_edge)-min(A_edge))
    *edge_factor
A_face_scaled = (np.matmul(T_pinv_face, f_curve_face)-min(A_face))
    *face_factor

reconstuct_A_grid_edge_scaled = A_edge_scaled.reshape((res, 2*res)
    , order = 'F')
reconstuct_A_grid_face_scaled = A_face_scaled.reshape((res, 2*res)
    , order = 'F')

edge_diff_grid = np.abs(albedo_map_grid -
    reconstuct_A_grid_edge_scaled)
face_diff_grid = np.abs(albedo_map_grid -
    reconstuct_A_grid_face_scaled)

theta_array_plus = np.linspace(0, np.pi, res) + 1/2*delta_theta
theta_array_min = np.linspace(0, np.pi, res) - 1/2*delta_theta

theta_array_plus[res-1] = np.pi
theta_array_min[0] = 0

theta_matrix_plus = np.tile(theta_array_plus  , (2*res, 1)).
    transpose()
theta_matrix_min = np.tile(theta_array_min  , (2*res, 1)).
    transpose()

surf_matrix = delta_phi*(np.cos(theta_matrix_min) - np.cos(
    theta_matrix_plus))/(4*np.pi)

edge_error = np.sum(np.multiply(np.square(edge_diff_grid),
    surf_matrix))
face_error = np.sum(np.multiply(np.square(face_diff_grid),
    surf_matrix))

print(edge_error)
print(face_error)



A_edge_noisy = np.matmul(T_pinv_edge, f_curve_edge_noisy)
A_face_noisy = np.matmul(T_pinv_face, f_curve_face_noisy)


edge_factor_noisy = 0.8/(max(A_edge_noisy) - min(A_edge_noisy))
face_factor_noisy = 0.8/(max(A_face_noisy) - min(A_face_noisy))


A_edge_scaled_noisy = (A_edge_noisy)*edge_factor_noisy
```

```
A_face_scaled_noisy = (A_face_noisy)*face_factor_noisy

reconstuct_A_grid_edge_scaled_noisy = A_edge_scaled_noisy.reshape
    ((res, 2*res), order = 'F')
reconstuct_A_grid_face_scaled_noisy = A_face_scaled_noisy.reshape
    ((res, 2*res), order = 'F')

edge_diff_grid_noisy = np.abs(albedo_map_grid -
    reconstuct_A_grid_edge_scaled_noisy)
face_diff_grid_noisy = np.abs(albedo_map_grid -
    reconstuct_A_grid_face_scaled_noisy)


edge_error_noisy = np.sum(np.multiply(np.square(
    edge_diff_grid_noisy), surf_matrix))
face_error_noisy = np.sum(np.multiply(np.square(
    face_diff_grid_noisy), surf_matrix))

print(edge_error_noisy)
print(face_error_noisy)
```

CODE FOR RECOVERING AXIAL TILT

```
import matplotlib.pyplot as plt
import math
import numpy as np

'''
Parameters
'''
res = 15
rho = 6.371e6
R = 1.496e11
res_alpha = 37
res_beta = 19

alpha = np.linspace(0/4*np.pi, 2/2*np.pi, res_alpha)
beta = np.linspace(0/4*np.pi, 1/2*np.pi, res_beta )

day = 24
year = 365*day
omega_day =  2*np.pi/day
omega_year = 2*np.pi/year

delta_t = 1
delta_phi = np.pi/res
delta_theta = np.pi/res
delta_Omega = delta_phi*delta_theta

hours = np.arange(0, 24)
time_array = hours.copy()

for i in range(1, 20):
    time_array = np.append(time_array, hours + i*year/20)

time_res = len(time_array)

```

```
'''
Albedo-map
'''
A = np.zeros((2*res**2, res_alpha*res_beta))
f_arrays = np.zeros((480, res_alpha*res_beta ))
time = np.linspace(0, 1, round(year/delta_t))*365

'''
Euler rotation matrices
'''
def positive(arg):
    return (arg + abs(arg))/2

def y_rotation(angle):
    Y = np.array([[np.cos(angle), 0, np.sin(angle)],[0, 1, 0],[-np.sin(angle), 0, np.cos(angle)]])
    return Y

def z_rotation(angle):
    Z = np.array([[np.cos(angle), -np.sin(angle), 0],[np.sin(angle), np.cos(angle), 0], [0, 0, 1]])
    return Z

'''
Initialize transfromation matrices (edge-on, face-on)
'''

T = np.zeros((time_res, 2*res**2, res_alpha*res_beta))
T_pinv = np.zeros((2*res**2, time_res, res_alpha*res_beta))

'''
Observer
'''
o_vec = np.array([0, 0, 1])

'''
Compute matrix elements
'''
for k in range(res_alpha):
    R_equinox = z_rotation(alpha[k])

    for l in range(res_beta):
        R_tilt = y_rotation(beta[l])
        R_axial = np.matmul(R_equinox, R_tilt)

        for i in range(len(time_array)):
            t = time_array[i]
            r_vec = np.array([-np.cos(omega_year*t), -np.sin(omega_year*t), 0])
            R_daily = z_rotation(omega_day*t)
            daily_rotation = np.matmul(R_axial,  R_daily)

            for j in range(2*res**2):
                phi = math.floor(j/res)*delta_phi
                theta = (j % res)*delta_theta

                s_vec = np.array([np.cos(phi)*np.sin(theta), np.sin(phi)*np.sin(theta), np.cos(theta)])
                s_vec_rotated = np.matmul(daily_rotation,s_vec)

```

```
                r's = np.dot(r'vec, s'vec'rotated)
                s'o = np.dot(s'vec'rotated, o'vec)

                illuminated = positive(r's)
                visible = positive(s'o)


                T[i][j][k*res'beta+l] = illuminated*visible*np.sin
    (theta)*delta'Omega

        T' = T[:, :, k*res'beta+l]
        T'pinv' = np.linalg.pinv(T', rcond = 0.01)
        T'pinv[:,:, k*res'beta+l] = T'pinv'
        A' = np.matmul(T'pinv', f'curve'edge)
        A[:, k*res'beta+l] =A'
        f'arrays[:, k*res'beta+l] = np.matmul(T', A')*rho**2/(R
    **2*np.pi)
        print(k*res'beta+l)

f'curve'matrix = np.tile(f'curve'edge, (res'alpha*res'beta, 1)).
    transpose()
f'diff = np.square(f'curve'matrix - f'arrays)
square'diff = np.sum(f'diff, axis = 0)
minima = np.where(square'diff == square'diff.min())[0]

alpha' = np.floor(minima/res'beta).astype(int)
beta' = minima%res'beta

print('alpha =', np.degrees(alpha[alpha']))
print('beta =', np.degrees(beta[beta']))
```