Anatomía magnética de las estructuras de la cromosfera del Sol

Trabajo de Fin de Grado
Grado de Física

Mayo, 2024

Alumno:
José Jaime Martín Acevedo

Tutores:
Andrés Asensio Ramos
María Jesús Martínez González

Abstract

The anatomy of the magnetic field in the outer layer of the Sun is a key component for understanding the movements of the plasma, including the formation and behavior of plasma structures that emerge from the surface. In this project, I explore the link between the polarimetric profile of atoms and the magnetic field they are under, and the potential of this as a diagnostic tool for the solar magnetic field.

For this, I cover the Zeeman effect, which not only shifts energy levels but also produces polarization in the light emitted during transitions; the Hanle effect, which alters the polarization axis through rotation and reduces its amplitude; and the mechanism of scattering polarization, a complex phenomenon where atomic level populations are affected by an anisotropic environment, leading to unbalanced populations that influence the polarization of scattered light. Special emphasis is placed on the problem of

To this end, I will also learn and employ HAZEL, the global reference code for interpreting these spectral lines, to produce visual content to accompany the theoretical concepts.

Resumen general en español: La anatomía del campo magnético en la capa exterior del Sol es un componente clave para entender los movimientos del plasma, incluida la formación y el comportamiento de las estructuras de plasma que emergen de la superficie. En este proyecto, exploro la relación entre el perfil polarimétrico de los átomos y el campo magnético al que están sometidos, y el potencial de esto como una herramienta de diagnóstico para el campo magnético solar.

Para esto, cubro el efecto Zeeman, que no solo desplaza los niveles de energía, sino que también produce polarización en la luz emitida durante las transiciones; el efecto Hanle, que altera el eje de polarización mediante rotación y reduce su amplitud; y el mecanismo de polarización por dispersión, un fenómeno complejo donde las poblaciones de niveles atómicos son afectadas por un entorno anisotrópico, lo que lleva a poblaciones desbalanceadas que influyen en la polarización de la luz dispersada. Se hace especial énfasis en el problema de determinar la configuración exacta del campo magnético a partir de los datos polarimétricos observados.

Para este fin, también aprenderé y emplearé HAZEL, el código de referencia global para interpretar estas líneas espectrales, para producir contenido visual que acompañe los conceptos teóricos. español.

Acknowledgments

I would like to acknowledge the use of OpenAI's ChatGPT in assisting with the drafting and editing of this project. I have also used ChatGPT to help me learn and write python code. However, I have not used ChatGPT to produce new content; all original ideas and writing are my own.

Contents

1 Introduction

Resumen: La luz es uno de los medios más importantes por el cuál adquirir conocimiento sobre los sistemas físicos, y la información que porta esta codificada en diferentes características: dirección de propagación, frecuencia y amplitud, y polarización. En este projecto exploraré la información contenida en la polarización de la luz utilizando el código HAZEL como un medio.

In physics, one of the most critical means of information retrieval is light. Light allows us to deduce the transition states of electrons, construct surface topologies using LIDAR, analyze the microstructure of materials, measure gravitational waves, and peer inside a patient's body with computerized tomography, among other applications. Its importance is particularly pronounced in fields like astronomy, where in situ measurements are largely impractical beyond our immediate vicinity. Even for our closest star, the Sun, direct measurements are essentially impossible without relying on the information carried by light. Light conveys information through various forms: its propagation direction reveals the location of stellar objects, frequency and amplitude indicate their composition or relative velocities, and polarization (discussed here) also provides insights into the states of the system it traversed.

In the topic of stellar physics, the systems are so extreme that we have not arrived into an agreed explanation for some of its activities yet. One example is the behavior of solar plasma over its surface, like the formation of filaments and protuberances against the gravity of the star. But what has been clear is that the magnetic field plays a fundamental role in this formations. Understanding this connection is vital for solar physics. This project initially aimed to explore the intricate relationship between the magnetic field in the Sun's chromosphere and the polarimetric signatures of the light recieved from it, utilizing the HAZEL code.

However, as the research progressed, it becaim evident that the original objective was more challenging than anticipated due to the complexity of the theoretical models and the programming skills required (or my lack thereof). In response to these challenges, the focus was shifted to a more manageable scope that still offers significant insights into real solar physics. The new objective became to understand the fundamental principles underlying HAZEL code and to apply it to simpler, yet meaningful, scenarios. This adjustment allowed for a more though exploration of the code's capabilities and provided valuable learning opportunities.

This project now aims to present these finding, offering new insights into solar magnetic fields and their influence on polarimentric signatures through a more visual representation.

2 Objectives

Resumen: Los objetivos originales del proyecto para la introducción a la diagnosis del campo magnético incluian aplicación a un caso real. Esto no se pudo realizar y en su lugar me concentré en entender la física detrás de HAZEL y aplicarlo a casos sencillos.

The original objective of this project was to introduce the student to the diagnosis of the magnetic field of the Sun, using inference techniques applied to the polarized spectra of the solar light. For this, the study of real solar structures was proposed. However, the understanding of the theoretical models and the use of the HAZEL code proved to be more challenging than expected.

Therefore the objectives were revised as follows:

- **To understand the fundamental principles underlying the HAZEL code:** this involves studying the theoretical background and the functionalities of the HAZEL code, developed by Andrés Asensio Ramos. This will be done mainly through theory texts about

the Zeeman effect, Hanle effect and the mechanisms of scattering polarization by Trujillo Bueno Landi Degl'innocenti and Andrés Asensio Ramos.

- **To apply the HAZEL code to simpler scenarios:** This focuses mainly on the use of the synthesis function to simulate a measurement of a polarimetric signature, to then compare this to the polarimetric signatures produced under many different conditions to compare them, and see how this affect the signatures and how the ambiguities in the possible solutions appear.

3 Theoretical background

Resumen: En esta sección explico, en diferentes grados de rigor, los fenómenos físicos implicados en la polarización de la luz solar. Desarrollo las consecuencias del efecto Zeeman, no solo en los cambios energéticos de los niveles atómicos, sino también en la polarización de la luz emitida o absorbida en transiciones. Expongo una explicación clásica del efecto Hanle, el cual depolariza la luz y rota el eje de polarización lineal en presencia de un campo magnético. Y doy una presentación cualitativa del fenómeno de polarización por dispersión.

The phenomena that dictate the behavior of the plasma at the surface of the Sun are many and complex. In the following sections I will discuss about the ones that I have learned about and are most relevant for the HAZEL code.

I will also cite here the book by Egidio Landi Degl'Innocenti and Marco Landolfi [3], as it provides a comprehensive explanation of the mechanisms for polarization in spectral lines. I will only cover the concepts superficially, given the complexity and depth of the topic.

3.0.1 The Zeeman Effect

During the Physics Degree, we have studied the Zeeman effect in the Quantum Mechanics II course, following the book by Cohen-Tannoudji et al. [1], but not to the full extent relevant for this project. I will not explain the effect to its entirety, but I will talk about what I have learned and the basic ideas that are relevant for the effects at play. For this I will follow the explanation from the book.

The Hamiltonian for the hydrogen atom is:

$$H_0 = \frac{\mathbf{P}^2}{2\mu} + V(R) \tag{1}$$

where $V(R)$ is the coulomb potential and μ the reduced mass of electron and proton. After some work and physical considerations we arrived at the eigenstates $|\varphi_{n,l,m}\rangle$ and eigenvalues $E_n = -\frac{E_I}{n^2}$ where $E_I = \frac{\mu e^4}{2\hbar^2}$ is the energy to free the electron. The wave functions that we find are of the form:

$$\varphi_{n,l,m}(\mathbf{r}) = R_{n,l}(r) Y_l^m(\theta, \varphi) \tag{2}$$

The exact definitions of the functions $R_{n,l}(r)$ and $Y_l^m(\theta, \varphi)$ are not as important as the following property:

$$Y_l^m(-\mathbf{r}) = (-1)^l Y_l^m(\mathbf{r}) \tag{3}$$

Which mean that the parity of the wave function is determined by the quantum number l.

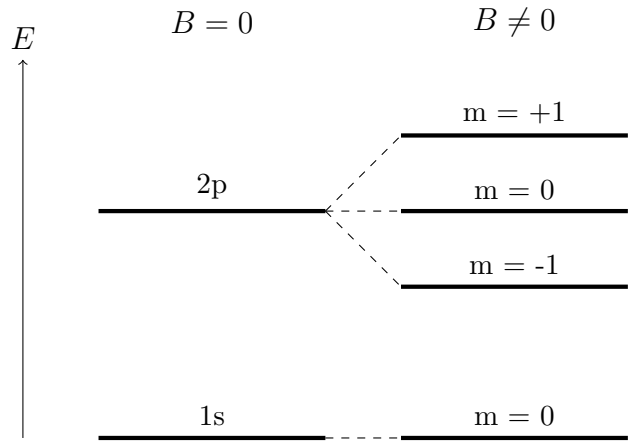

Figure 1: Diagram of the splitting of the energy levels of the first two energy levels of the hydrogen atom due to the Zeeman Effect (not to scale).

Under a magnetic field two new terms appear on the Hamiltonian:

$$H_1 = -\frac{\mu_B}{\hbar}\mathbf{L}\cdot\mathbf{B} \tag{4}$$

$$H_2 = \frac{q^2\mathbf{B}^2}{8m_e}\mathbf{R}^2 \tag{5}$$

H_2 is the diamagnetic term and, for magnetic field in astrophysics, except for maybe white dwarfs and compact bodies, can be safely ignored (pag 74 [3]).

For the sake of simplicity I will ignore the contribution of the fine and hyperfine structures. H_1 is sufficient to show the effects that I am looking for.

With that said, if we choose the magnetic field $\mathbf{B}$ to be parallel to the Z axis the eigenvalue equation becomes:

$$\begin{aligned}(H_0 + H_1)\left|\varphi_{n,l,m}\right\rangle &= (H_0 - \frac{\mu_B}{\hbar}BL_z)\left|\varphi_{n,l,m}\right\rangle \\ &= (E_n - m\mu_B B)\left|\varphi_{n,l,m}\right\rangle\end{aligned} \tag{6}$$

For the levels under consideration:

$$\begin{aligned}(H_0 + H_1)\left|\varphi_{1,0,0}\right\rangle &= (E_I)\left|\varphi_{1,0,0}\right\rangle \\ (H_0 + H_1)\left|\varphi_{2,1,m}\right\rangle &= [-E_I - \hbar(\Omega + m\omega_L)]\left|\varphi_{n,l,m}\right\rangle\end{aligned} \tag{7}$$

where:

$$\Omega = \frac{3E_I}{4\hbar} \tag{8}$$

$$\omega_L = -\frac{\mu_B B}{\hbar} \tag{9}$$

The originally degenerate $2p$ level splits into three different levels ($m = +1, 0, -1$). The ground state remains unaltered (Figure 1).

The second part is the effect of the magnetic field on the polarization of the light, a topic which I do not recall being covered during the course.

If we consider the electric dipole operator of the form:

$$\mathbf{D} = q\mathbf{R} \tag{10}$$

and try to find its mean value $\langle \mathbf{D} \rangle$, will see that the operator is odd, since $\mathbf{R}$ is an odd operator. This means that it will not "connect" states of the same parity, and we know that the parity of the states depends only on the azimuthal quantum number l, therefore:

$$\begin{cases} \langle \varphi_{1,0,0} | \, \mathbf{D} \, | \varphi_{1,0,0} \rangle = 0 \\ \langle \varphi_{2,l,m} | \, \mathbf{D} \, | \varphi_{2,l,m'} \rangle = 0 \ ; \ \forall m, m' \end{cases} \tag{11}$$

Using the representations of the spherical harmonics in equations (12) and the spherical transformation for the cartesian coordinates, we can easily see that equations (13) are true.

$$\begin{cases} Y_1^{-1}(\theta, \varphi) = \dfrac{1}{2}\sqrt{\dfrac{3}{2\pi}} \sin\theta e^{-i\varphi} \\ \\ Y_1^{0}(\theta, \varphi) = \dfrac{1}{2}\sqrt{\dfrac{3}{\pi}} \cos\theta \\ \\ Y_1^{1}(\theta, \varphi) = \dfrac{-1}{2}\sqrt{\dfrac{3}{2\pi}} \sin\theta e^{i\varphi} \end{cases} \tag{12}$$

$$\begin{cases} x = \sqrt{\dfrac{2\pi}{3}} r \left[Y_1^{-1}(\theta, \varphi) - Y_1^{1}(\theta, \varphi) \right] \\ \\ y = i\sqrt{\dfrac{2\pi}{3}} r \left[Y_1^{-1}(\theta, \varphi) + Y_1^{1}(\theta, \varphi) \right] \\ \\ z = \sqrt{\dfrac{4\pi}{3}} r Y_1^{0}(\theta, \varphi) \end{cases} \tag{13}$$

We can use this to calculate the matrix elements exploiting the orthogonality properties of the spherical harmonics and setting the radial integral equal to some value χ:

$$\begin{cases} \langle \varphi_{2,1,\pm 1} | \, D_x \, | \varphi_{1,0,0} \rangle = \mp \dfrac{q\chi}{\sqrt{6}} \\ \langle \varphi_{2,1,0} | \, D_x \, | \varphi_{1,0,0} \rangle = 0 \\ \\ \langle \varphi_{2,1,1} | \, D_y \, | \varphi_{1,0,0} \rangle = \langle \varphi_{2,1,-1} | \, D_y \, | \varphi_{1,0,0} \rangle = \dfrac{iq\chi}{\sqrt{6}} \\ \langle \varphi_{2,1,0} | \, D_y \, | \varphi_{1,0,0} \rangle = 0 \\ \\ \langle \varphi_{2,1,\pm 1} | \, D_z \, | \varphi_{1,0,0} \rangle = 0 \\ \langle \varphi_{2,1,0} | \, D_z \, | \varphi_{1,0,0} \rangle = \dfrac{q\chi}{\sqrt{3}} \end{cases} \tag{14}$$

With the previous results, we know that if the system is in a stationary state the mean value of $\mathbf{D}$ is zero, but if we consider a superposition of the ground sate with any of the upper three levels we have:

$$|\psi_m(0)\rangle = \cos\alpha \, |\varphi_{1,0,0}\rangle + \sin\alpha \, |\varphi_{2,1,m}\rangle \ ; \ \alpha \in \mathbb{R} \tag{15}$$

$$|\psi_m(t)\rangle = \cos\alpha \, |\varphi_{1,0,0}\rangle + \sin\alpha e^{-i(\Omega + m\omega_L)t} \, |\varphi_{2,1,m}\rangle \tag{16}$$

$$\langle \mathbf{D} \rangle_m (t) = \langle \psi_m(t) | \, \mathbf{D} \, | \psi_m(t) \rangle \tag{17}$$

For $m = +1$:

$$\begin{cases} \langle D_x \rangle_1 = -\dfrac{q\chi}{\sqrt{6}} \sin 2\alpha \cos\left[(\Omega + \omega_L)t\right] \\ \\ \langle D_x \rangle_1 = -\dfrac{q\chi}{\sqrt{6}} \sin 2\alpha \sin\left[(\Omega + \omega_L)t\right] \\ \\ \langle D_z \rangle_1 = 0 \end{cases} \tag{18}$$

Here $\langle \mathbf{D} \rangle_1 (t)$ rotates around the z axis counterclockwise with a frequency $\Omega + \omega_L$, this produces and emission of light, circularly polarized in the direction of z, linearly polarized if observed perpendicular to the z axis, and ellipticaly polarized in other directions. This is usually called the σ_+ polarization.

For $m = 0$:

$$\begin{cases} \langle D_x \rangle_0 = \langle D_y \rangle_0 = 0 \\ \\ \langle D_z \rangle_0 = \dfrac{q\chi}{\sqrt{3}} \sin 2\alpha \cos \Omega t \end{cases} \tag{19}$$

Now $\langle \mathbf{D} \rangle_0 (t)$ oscillates along the z axis with frequency Ω, producing no light if observed in the direction of the z axis, and linearly polarized light in any other direction. This is usually called the π polarization.

For $m = -1$:

$$\begin{cases} \langle D_x \rangle_{-1} = -\dfrac{q\chi}{\sqrt{6}} \sin 2\alpha \cos\left[(\Omega + \omega_L)t\right] \\ \\ \langle D_x \rangle_{-1} = -\dfrac{q\chi}{\sqrt{6}} \sin 2\alpha \sin\left[(\Omega + \omega_L)t\right] \\ \\ \langle D_z \rangle_{-1} = 0 \end{cases} \tag{20}$$

In contrast to the first case, here $\langle \mathbf{D} \rangle_{-1} (t)$ rotates around the z axis clockwise with a frequency $\Omega - \omega_L$, the types of polarization observed are the same as the first case, except for the named differences. This is usually called the σ_- polarization.

In summary, the Zeeman effect does not only shift the energy levels of the atom (which I already new from the quantum mechanics course), but also produce the light emitted during transitions to be polarized, depending on the change in the magnetic quantum number. While this description is not exhaustive and ignores many aspects of the quantum system, the main ideas derived (frequency split and polarization) still stand under a more elaborate description.

3.0.2 The Hanle Effect

The Hanle Effect alters the polarization of the light emitted by an atom under a magnetic field. To illustrate this I will use the classical explanation for transitions between states of total angular momentum $J_l = 0$ (lower level) and $J_u = 1$ (upper level) as explained by Trujillo Bueno in [6]. This explanation helped me understand what the effect does to the polarization.

In this model the atom is treated as an oscillating charge with angular frequency Ω and damping constant $\gamma = \dfrac{1}{t_{\text{life}}}$, with t_{life} the life time of the excited state.

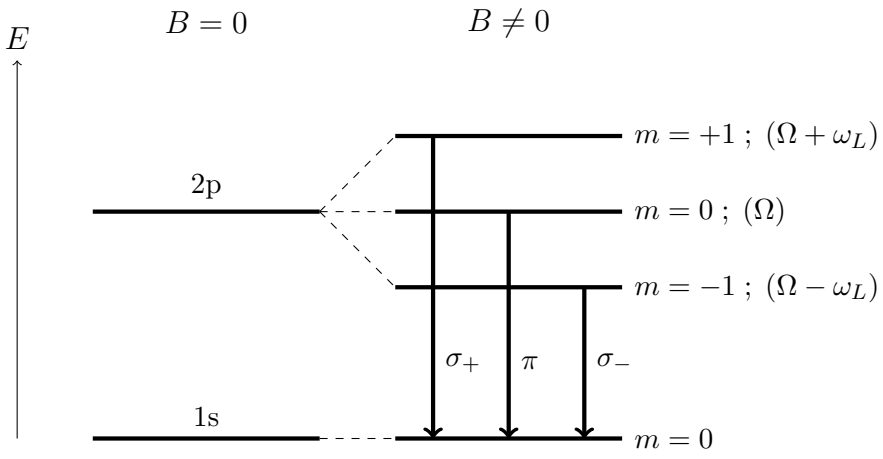

Figure 2: Zeeman splitting diagram with the transitions, the frequency of their emission and their polarization types.

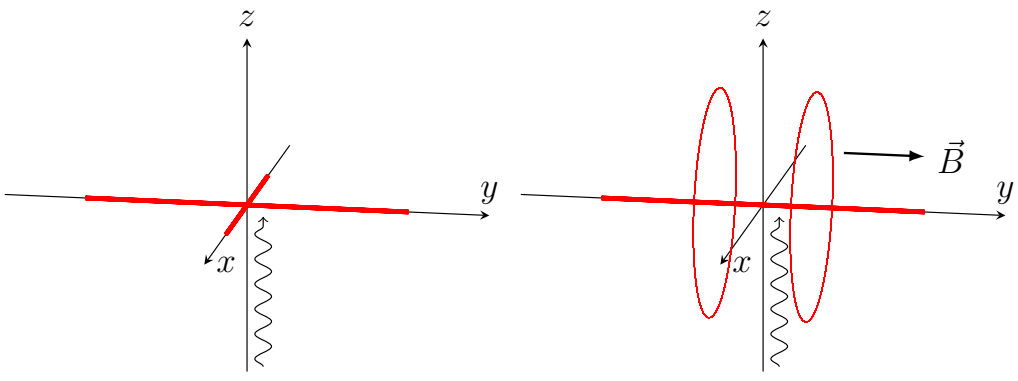

Figure 3: Excited modes due to the interaction with unpolarized light. To the left: no magnetic field case. To the right: magnetic field along the y axis.

In the absence of a magnetic field the atom can be treated as three independent linear oscillators with frequency Ω along each axes of the reference system. When unpolarized light coming parallel to the z direction is absorbed by the atom, only the x and y modes are exited. When this oscillators emit light, it is seen as unpolarized in the z direction or as linearly polarized in the x or y direction. This corresponds to the left diagram of Figure 3.

On the other hand, in the presence of a magnetic field, we no longer can interpret the atom as three independent linear oscillators due to the Lorentz force, but as a linear oscillator along the magnetic field with frequency (unaffected by the field) Ω; and two counter-rotating circular oscillators around the magnetic field vector, with frequencies $\Omega + \omega_L$ and $\Omega - \omega_L$. The resulting trajectory of the electron in the $x - z$ plane is:

$$x(t) = Ae^{-\frac{\gamma t}{2}} \cos(\omega_L t) \cos(\Omega t) \tag{21}$$

$$z(t) = Ae^{-\frac{\gamma t}{2}} \sin(\omega_L t) \cos(\Omega t) \tag{22}$$

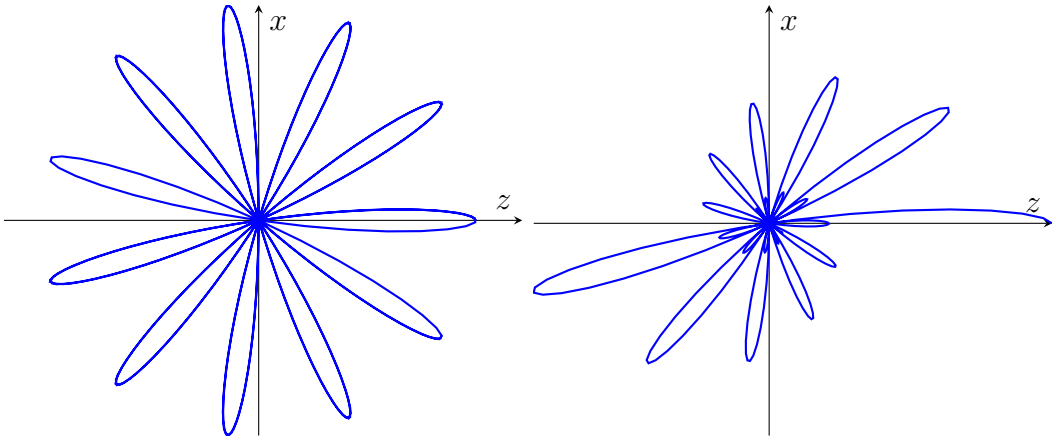

Figure 4: To the left: the oscillation of the negative charge under a magnetic field with $\omega_L \gg 1/t_{\text{life}}$. To the right: the oscillation of the negative charge under a magnetic field with $\omega_L \approx 1/t_{\text{life}}$.

These describe the oscillation along an axis that, simultaneously, rotates around the y axis. The resulting trajectory is showed in Figure 4. Depending on the strength of the field the Larmor frequency ω_L can be bigger or similar to the damping constant γ. If it is much greater then the axis oscillates many times before the amplitude is significantly diminished, leading to a patter similar to that of the left one of Figure 4. In the case where $\omega_L \approx \gamma$ the amplitude is much smaller at about one turn of the axis, resulting in the left patter of Figure 4.

In the first case, with the symmetrical pattern, we would see unpolarized light in the y direction. In the second case the patter is no longer symmetrical, and the measured polarization would be the average of the pattern, resulting in some weaker linear polarization with some angle α, this is a depolarization and rotation of the polarization plane.

3.0.3 Scattering polarization

Another contribution to the polarization of light from the chromosphere is scattering polarization. An introduction to the mechanisms can be found in [6]. While the scattering polarization is also an important phenomenon, its intricate mathematical descriptions and the time constraints of this study have precluded a detailed exploration. Therefore, I will not be expanding on this topic here, but I will try to give a qualitative explanation. For a comprehensive treatment of scattering polarization, readers are referred to the work of Landi Degl'innocenti [3].

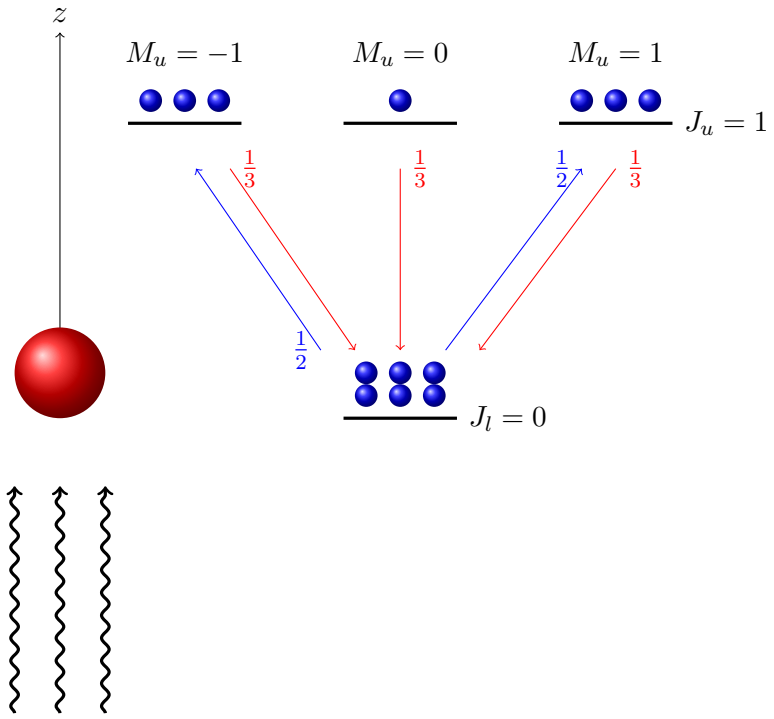

Figure 5: Example of scattering polarization though a unpolarized one directional radiation field. The red sphere represents a group of atoms, the squiggly lines the radiation and the blue spheres the level populations.

The gas at the chromosphere lives within an anisotropic environment, also called an "ordered" environment, in the sense that not everything is the same in every direction, but maybe illumination is stronger from a given direction, or there is a magnetic field in some direction. In the same way, atomic polarization can also be called "ordered", again, in the sense that some levels are more likely to be populated than others, so the distribution is not "the same everywhere".

This "order" from the environment can be transmitted to the "order" of the atom. For example, like we saw in the Hanle effect section (Figure 3), an unpolarized light beam illuminating an atom with $J_l = 0$ (lower level) and $J_u = 1$ (upper level), will only have transitions with $\Delta M = \pm 1$, and no transitions occur to the $M = 0$ sublevel. With uniform relaxations the $M = \pm 1$ sublevels will be more populated than the $M = 0$ sublevel (see Figure 5). This is what Trujillo Bueno calls a transmission of "order" from the radiation field to the atom polarization.

This is represented by a connection between the evolution of the density matrix (ρ^K_Q in the spherical statistical tensor representation), that encodes the information of the quantum system, and the radiation field tensors ($\bar{J}^K_Q$), that encodes the information regarding the radiation field, and therefore the possible degrees of anisotropy in it.

In the case of the Sun, even if there is no anisotropy in the polarization of the light (this may be that all light is unpolarized, for example), there is still anisotropy in the intensity of the radiation, since, in the outermost layers of the Sun, most of the radiation comes from the center, parallel to the normal of the surface, as showed in Figure 6.

Trujillo Bueno in [6] emphasizes that this contribution can not be ignored in order to understand the second solar spectrum (referring to the polarization signals of the light normally measured close to the solar limb, where these signals are larger).

In [7] Trujillo Bueno explains how the combination of this effects can be used as a diagnostic tool for understanding the electromagnetic structure of the solar atmosphere. And thus, codes like HAZEL are born, to try to construct a numerical tool to solve the corresponding equations.

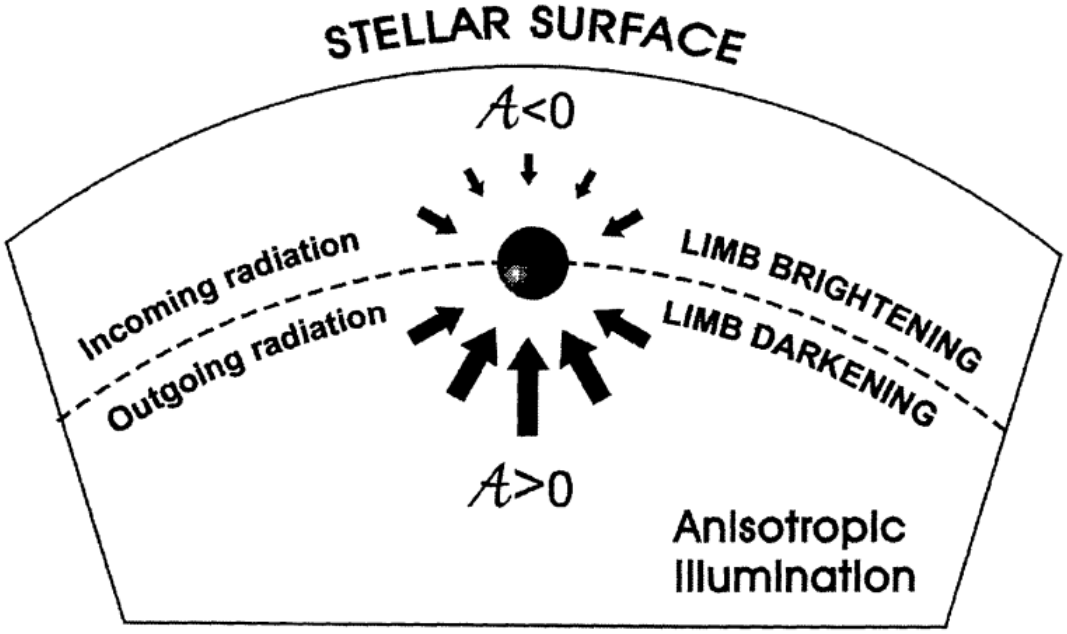

Figure 6: The anisotropic illumination in a stellar atmosphere. Figure from [6].

4 Methodology

Resumen: La metodología consiste principalmente en la aplicación del codigo HAZEL para estudiar el comportamiento de las soluciones en función de distintas variables, ya que la principal problemática con la inversión son las ambigüedades. Explico cuales son las ambiguedades y cuántas pueden aparecer. Explico el funcionamiento general del código HAZEL y la aplicación que le he dado yo personalmente generando mapas de calor, videos y esculturas del espacio de fases.

The methodology of the study consists, mainly, in simulating a measurement of polarized light from the Sun using the HAZEL synthesis function, and trying to compare the finding to the theoretical concepts related to the system, but mainly the ambiguities, which represent the main obstacle with this approach for studying the plasma structures.

4.1 The ambiguities

Since deriving the magnetic field at the solar atmosphere from the polarimetric signatures consists in an inversion problem, there will be, in general, ambiguous solutions. These ambiguities rise in two levels. The first is the ambiguities due to the theory itself, there exists more than one magnetic field that leads to the measured polarization. The second is due to lack of information. Since the polarity of the field is decided by the V signal, if the signal is comparable to the noise, then one can not decide the polarity and new solutions appear.

In the first case, there are two distinct ambiguities, the first is the Hanle ambiguity, that appears at $\varphi'_B = \varphi_B + 180°$; and the second is the Van Vleck ambiguity, that appears at $\varphi'_B = \varphi_B \pm 90°$. The Hanle ambiguity is due to the fact that the effect seen by a field with azimuth angle φ in the plane of the sky, and another with azimuth angle $\varphi + 180°$ is the same; think of Figure 3, an observer in the z direction will see the same polarization even if the magnetic field was pointing in the opposite direction because the rotation of the polarization happens in the zx plane. The Van Vleck ambiguity is more complex, since it may or may not appear depending on

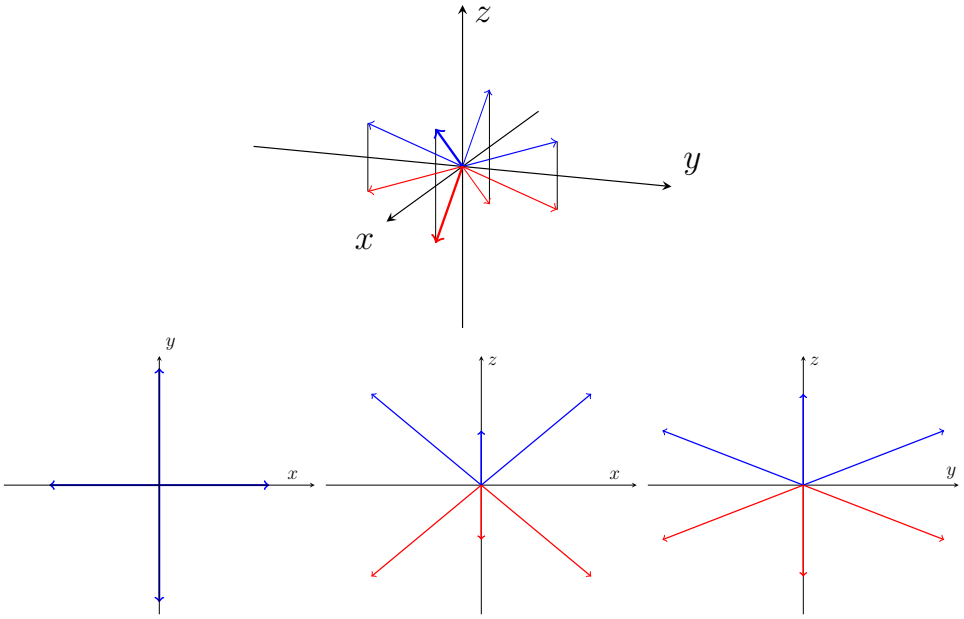

Figure 7: An example of all possible ambiguities in the inversion process for the saturated regime. On the bottom are the flat projections on different planes for better understanding. In this representation the z axis is parallel to the normal of the sun surface and the measurement is done on disk center.

the combination of angles of the solutions. I did not dive into the theory behind the Van Vleck ambiguity. The combination of this two imply that one has two or four possible solutions.

A calculation of the expressions for the ambiguities for a case of: two level atom with $J_l = 0$ and $J_u = 1$, optically thin limit and saturation regime for the Hanle effect; can be found in [2]. There, starting from the equations for Q and U polarization, arrives at a group of 4th degree equations, one for each ambiguity in the azimuth angle, that returns the polar angle that keeps the polarization unchanged. This also returns mathematical solutions that are not physically valid. I will also show that this equations correctly approximate the positions of the ambiguities.

So, in worst case scenario, there might be up to eight possible solutions (combination of Hanle ambiguity, Van Vleck ambiguity and lack of V signal ambiguity combined) and, at best, one can have two possible solutions (Hanle ambiguity).

My tutor, Andrés Asensio Ramos, explained to me that at best one can only choose a solution based on if it produces a smooth field with the surroundings. There are methods to try and find the best fitting solution, but they are not easy to execute (may need a great number of measurements) and may not be definitive.

4.2 The HAZEL code

Before continuing with the code, I would like to point out that, for the physical system, one has to choose a frame of reference. There are, of course, better and worse choices depending on the objective. HAZEL takes the local vertical as the z axis for it's own convenience (see Figure 8).

The HAZEL code is pretty straightforward in the functionality. It has two modes, synthesis and inversion. Synthesis produces a polarimetric signature from a set of variables, and inversion produces a set of variables from the polarimetric signature. This set of variables that affect are, among other, the magnetic field strength and its direction; they are many and make up a multidimensional phase space where the program looks for an optimal fit to the signature by finding minima in the χ^2 function. An explanation about the inner workings of the HAZEL code can be found in [4].

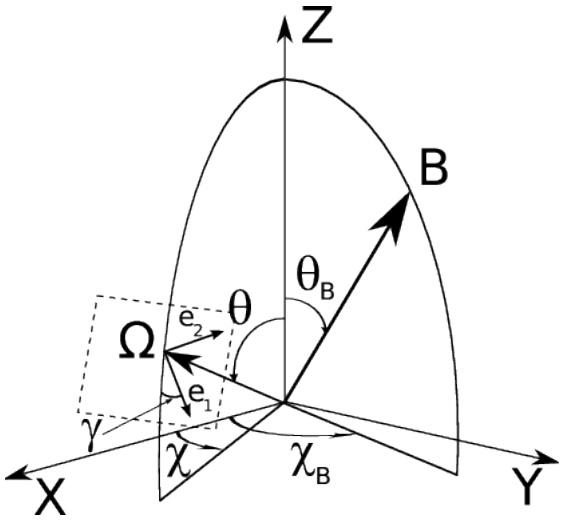

Figure 8: The scattering geometry as showed in the HAZEL documentation [5]. $\mathbf{\Omega}$ denotes the line of sight vector (LOS), $\mathbf{e_1}$ and $\mathbf{e_2}$ are the axes for polarization (the angle γ can be freely chosen to decide what line corresponds to Q polarization). The dashed square represent the plane of the sky. The z axis is parallel to the local vertical of the Sun.

The full documentation of the HAZEL code can be found in [5]. The "complexity" for the user (it is probably very user friendly for people with knowledge in Python and on the theoretical background) resides in the configuration within the code. This configuration files are the ones where the user specifies what the physical system is. This is done though the definition of the atmospheres involved, these can be photospheres, chromospheres, parametric atmospheres (telluric lines, fringes, smooth continua) and straylight components. The user can define in the configuration file the topology of the atmospheres by layering them or combining different atmospheres of the same type onto one layer. This atmospheres have specific configurations of their own, with the relevant physical data.

One also needs to set configurations for the inversion process, and the results heavily depend on this inputs. It is explicitly said in the documentation itself, that the Stokes weights (one of the variables that the user defines for the inversion) require some trial and error to find a reliable solution. All of this adds to many parameters within the configuration that one needs to get acquainted with. Another important aspect of the inversion function, is that the program consists in iterations, trying to improve the merit function, so a stop must be set. This can be by number of iteration or by variance of the merit function. The point is that the stop may be reached before the fitting is done.

I will not delve on all this configurations, I think is a better use of time to directly show what one can do and see with the code.

> **Note:** from now on I will use φ_B as the azimuthal angle, intead of the χ_B shown in Figure 8. The letter χ will be used instead for the merit function of HAZEL (Section 4.2.1).

4.2.1 The Heat Maps

During my time practicing with the HAZEL code I tried the functionalities of the code and how the configurations affect the results. This is a very important part for those trying to infer the magnetic field of the solar atmosphere though measurements, but, in my case, I would like to develop a more visual tool, since I believe that visualization is a very helpful way of developing intuition.

The idea came from the article [4]. The Figure 13 of the article showed slices of the phase space where only the angles of the magnetic field are free. This produces a heat map in which one can see where the better fits (what would be considered the solutions) of observed profile are. I became curious of how the ambiguities appear or move as the field intensifies or the angles change.

Of course, this are not the only variables that effect the map, but there are too many of them for me to talk about it all, so I chose to simplify the approach by: working with the magnetic field angle phase space and using in the configuration of the atmosphere only one chromosphere, focusing only on the wavelength region around the 10830 Å where the He_I triplet transitions occur.

The code that I wrote could be separated in 3 functionalities:

- **The heat map generator:** this is done by first making a polarization profile, then adding noise to simulate a measurement , and then generate all possible profiles (given some phase space resolution) and compare them with the original one via the merit function used by HAZEL, generating a grid of dots that can be draw into a heat map. The merit function is:

$$\chi^2 = \frac{1}{4N_\lambda} \sum_{i=1}^{4} \sum_{j=1}^{N_\lambda} \frac{\left[S_i^{syn}(\lambda_j) - S_i^{obs}(\lambda_j)\right]^2}{\sigma_i^2(\lambda_j)} \tag{23}$$

  where N_λ is the number of wavelength pints, S^{obs} is the observed stokes signal, in this case simulated by making a profile with HAZEL and adding noise, S^{syn} is the signals it is comparing to, and $\sigma_i^2(\lambda_j)$ is the variance associated to the j-th wavelength point of the i-th Stoke profile. The sum is done over all wavelength and all signals.

- **Video generator:** in combination with the previous function, it takes the generated heat maps and joins them into a video. The heat map is prepared to receive the value of some variables by name, this way a set of values can be given to generate the video.

- **Volume generator:** the last function is due to the fact that video can not be showed here, so I though that another way of showing the change of the heat map, is by plotting the dots that have χ^2 smaller than some selected value. This way one can see the movement of the solution though 3D space. This is done by reusing the results from the heat map, since it consists in a grid of values, this function just filters them by the χ^2 condition and plots them using the plotly library, since there are so many points that other options are too slow.

5 Result discussion:

Resumen: Relaciono los comportamientos de las ambiguedades en fuención de distintas variables con la teoría, y utilizando las representaciones gráficas como apoyo.

The result is that the figures that I will show now, can demonstrate some of the properties of the ambiguities discussed earlier in a very engaging and easy-to-understand way.

5.1 Evolution of ambiguities with φ_B

In Figure 9, the white color represents where the merit function takes the lowest value (and therefor, where the best fitting profiles are). Because this is a simulation with a known field, the exact value of the merit function is not relevant (we know that the solutions that we see are correct). In a real experiment, where we do not know if the found solution is correct or not, we

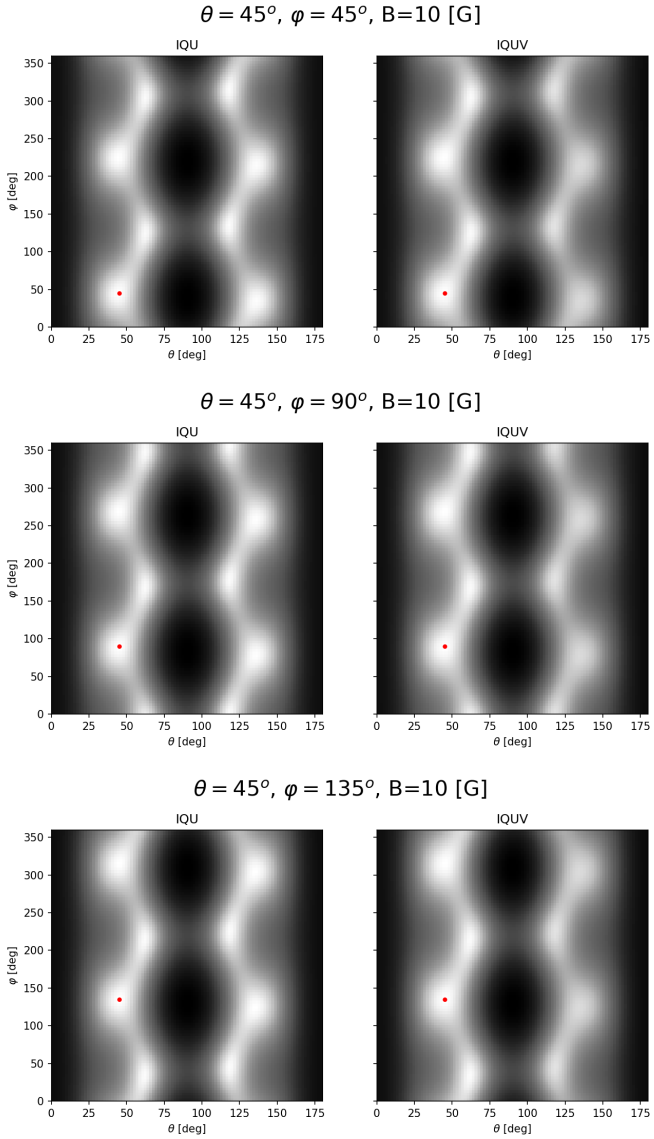

Figure 9: Possible solutions to the inversion problem (white represents the better fit), for an on-disk ($\theta_{LOS} = 0$) measurement, for φ_B values ok 45°, 90° and 135°. The red dot represents the position of the real field. The letters I, Q, U and V indicate what Stokes signals of the profile were used for the calculation of the merit function.

would need to take into account the merit function value to decide if the found solutions are good or not. This is because the number of variables that HAZEL is trying to fit are many more than shown here, and therefore it might get stuck in a χ^2 minimum somewhere else in phase space.

We can see that the shape of the distribution does not change with φ_B, all the ambiguities move uniformly up or down in the phase space. Note also that for the figures with the I, Q and U signals there are a total of 8 ambiguities, like explained in Section 4.1, corresponding to the combination of the Hanle, Van Vleck and V noise ambiguities. For the maps calculated with the V signal we see that the right solutions are no longer as good as the left ones, leaving 4 solutions instead.

5.2 Evolution of ambiguities with θ_B

In Figure 10 we have the same situation, but changing the magnetic field polar angle θ_B. Here we can see that the distribution of the ambiguities does change. Note how ambiguities appear

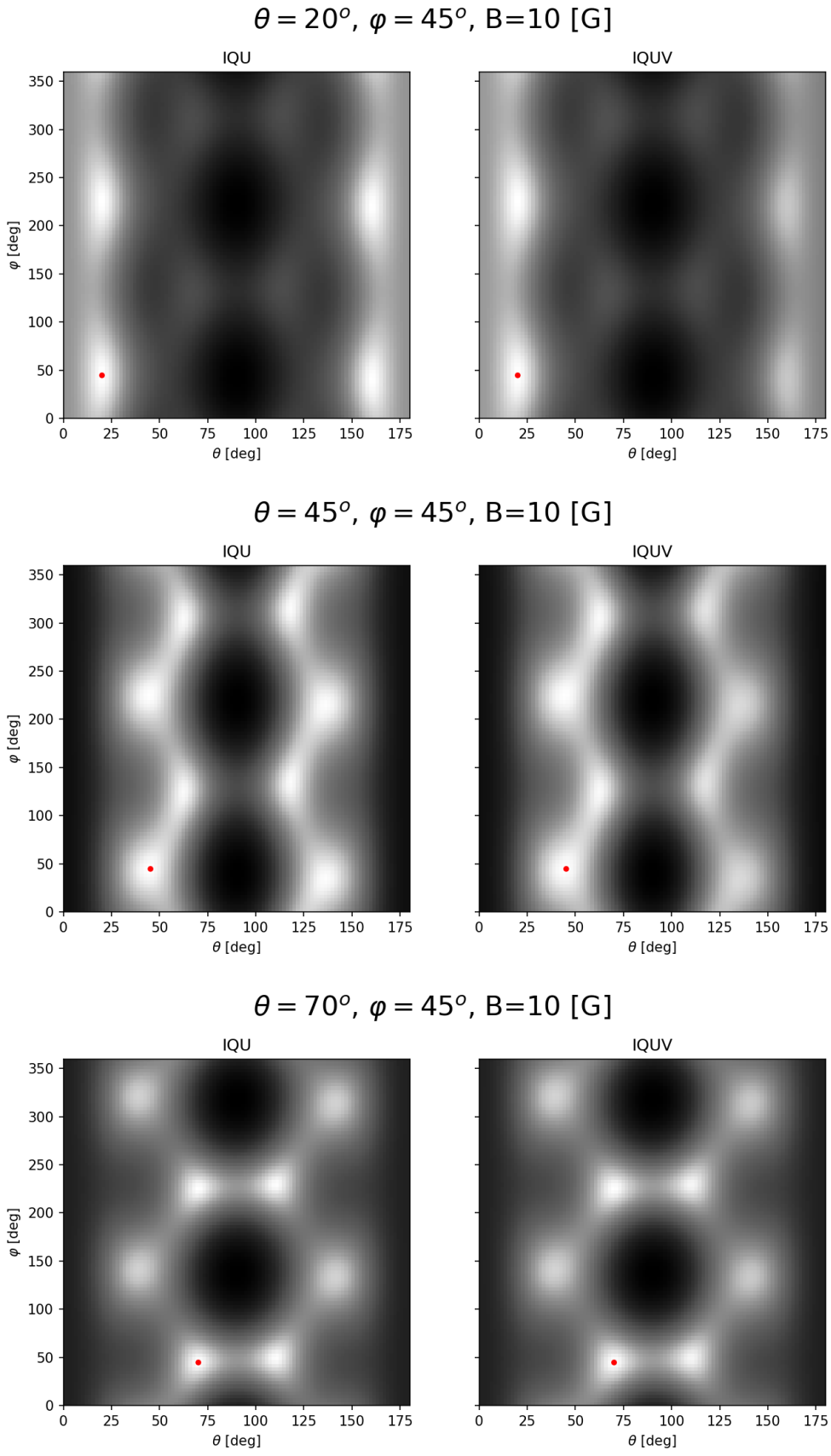

Figure 10: Same conditions as Figure 9, but the changing angles is θ_B instead.

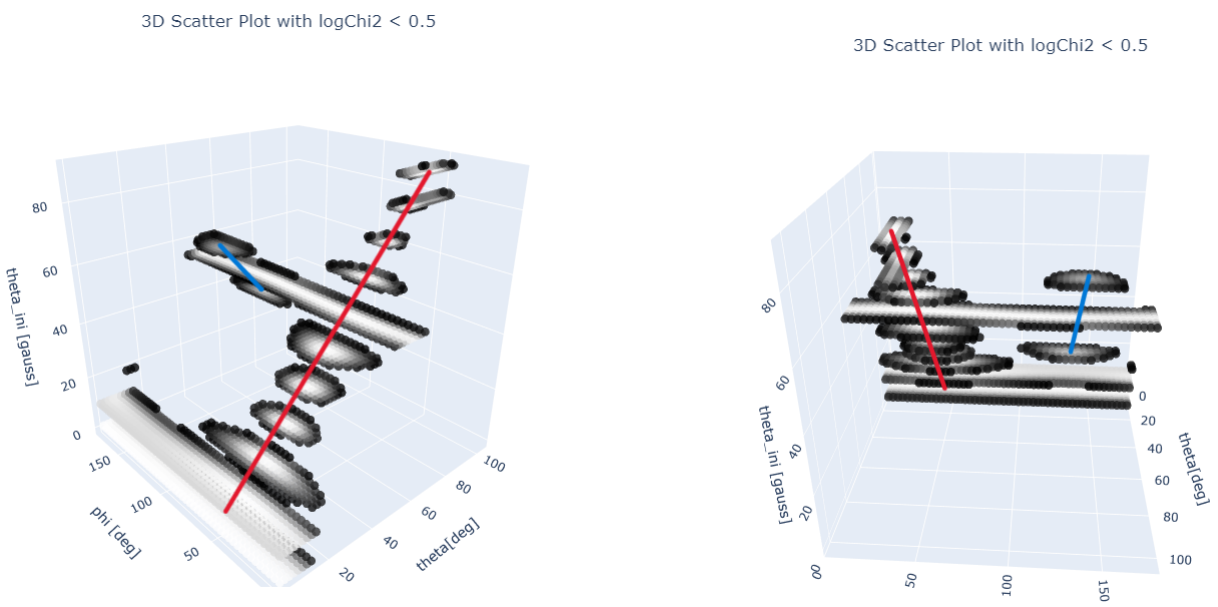

Figure 11: Sculpture showing a more smooth evolution for the maps at Figure 10. The magnetic field moves from $\theta_B = 0$ to $\theta_B = 90$. The red line denotes the evolution of the real field. The dots shown are those for which $\log \chi^2 < 0.5$. The blue line is the Van Vleck ambiguity movement.

and disappear depending on the value of θ_B. For the first and third row we can only see four solutions, corresponding to the Hanle and V noise ambiguities. But in the middle row we see again the eight ambiguities because the Van Vleck ambiguity has appeared. Therefore, we can see the behavior of the Van Vleck ambiguity explained in Section 4.1. The Van Vleck ambiguities only appear under certain conditions of the magnetic field and the line of sight.

A smoother evolution can be seen in the sculpture shown in Figure 11. We can see that, as the θ_B goes from 0° to 90°, how the new Van Vleck ambiguities appear (blue line). The white stripe that appear approximately at the middle of the evolution is the consequence of the alignment of the Van Vleck ambiguity with the Hanle ambiguity, forming a kind of valley in the phase space.

5.3 Evolution of ambiguities with magnetic field strength B

Figure 12 shows the evolution of the ambiguity distribution when the magnetic field strength changes from 1G to 10G. We can see that for low strengths there are only four solutions, but as the field intensifies a new branch emerges from every solution, making the eight ambiguities. When they reach the top (10G) we regain the same distribution that we saw in Figure 9 (compare the right side view of Figure 12 with the first row of Figure 9). This were calculated using all the Stokes signals, so we can see that when reaching the saturation regime the V noise ambiguities start to worsen, eventually not being able to stay under $\log \chi^2 < 0.4$.

5.4 Evolution of solution for increased error in field strength B

Sculpture in Figure 13 simulates what happens when we make a mistake when trying to identify the field strength. Like I said in Section 4.2, there are a considerable number of variables that compose the hyper volume of phase space. It is very possible for HAZEL to find a wrong answer that seems like a good fit. For simulating that I, instead of changing the field of the actual solution and compared profiles, changed the field of the compared profiles alone, so that we see other

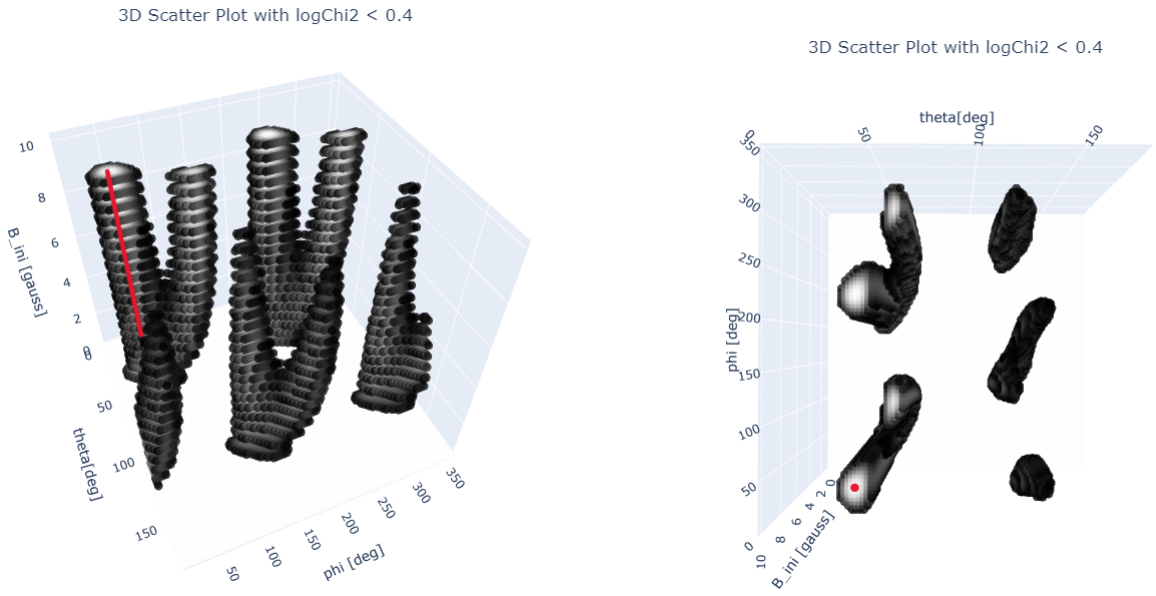

Figure 12: Evolution of ambiguity distribution with magnetic field strength.

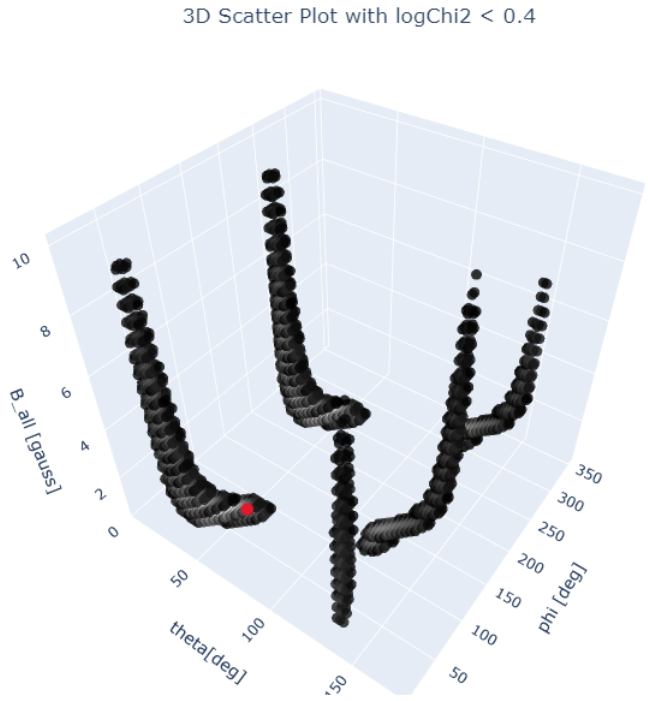

Figure 13: Evolution of the ambiguities as we commit an increasing mistake in the field strength. B_all indicates the field strength of the compared profiles.

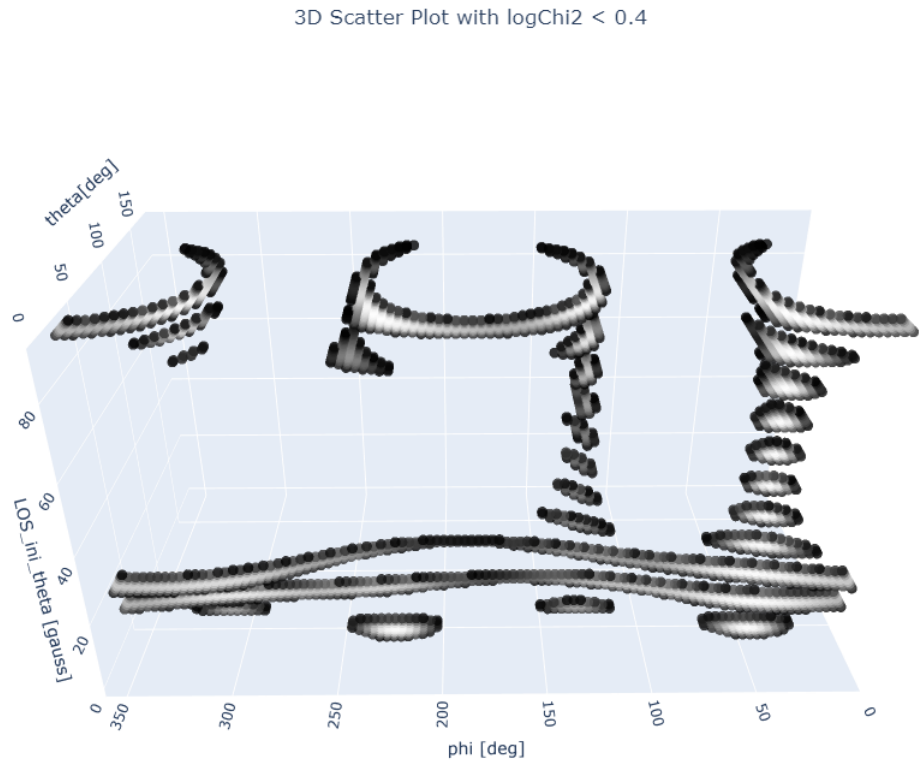

Figure 14: Evolution of ambiguities as θ_{LOS} evolves from 0° to 90°.

points in phase space that would give a good fit still. The bottom of the sculpture corresponds to the real solution, so the base of the columns are the real ambiguities of the correct solution. As we go up the arms these displace, changing the angles of the four solutions trying to keep the fit good. So, by making a mistake in the field strength one gets a different set of angles as the solution. We can also see that as the strength keeps increasing the code is having a harder time finding a good fit. They stop changing when approaching saturation. The same thing could be done with other variables, so we can imagine the complexity of finding a solution.

5.5 Dependence of ambiguities on line of sight

Another interesting demonstration is what happens when we observe the same spot (under the same conditions) while changing the LOS angle θ_{LOS} (Figure 14), or in other words, when we look at a spot as we let it move from disk-center to the border of the sun (off-limb) if it did not evolve over time. We see that there are only two columns connecting bottom to top, this means that only two ambiguities persist as we look from many directions. Of course, this can not be done just from one point of view, since the spot will not remain in the same condition as times passes, but we could try to set observations simultaneously from different angles to try to get rid of some of the ambiguities. This is obviously harder than it sounds, because looking at a spot at the sun from a sufficiently different angle requires measurement from great distances apart.

5.6 Example of ambiguity equation solutions for simple case

Here we can see that the solutions to the polynomial equations mentioned section 4.1 (red dots) obtained in [2] (can also be found in the HAZEL documentation [5]) pretty closely follow the ambiguities that HAZEL obtains. In all of them we can see that the red dots are not perfectly

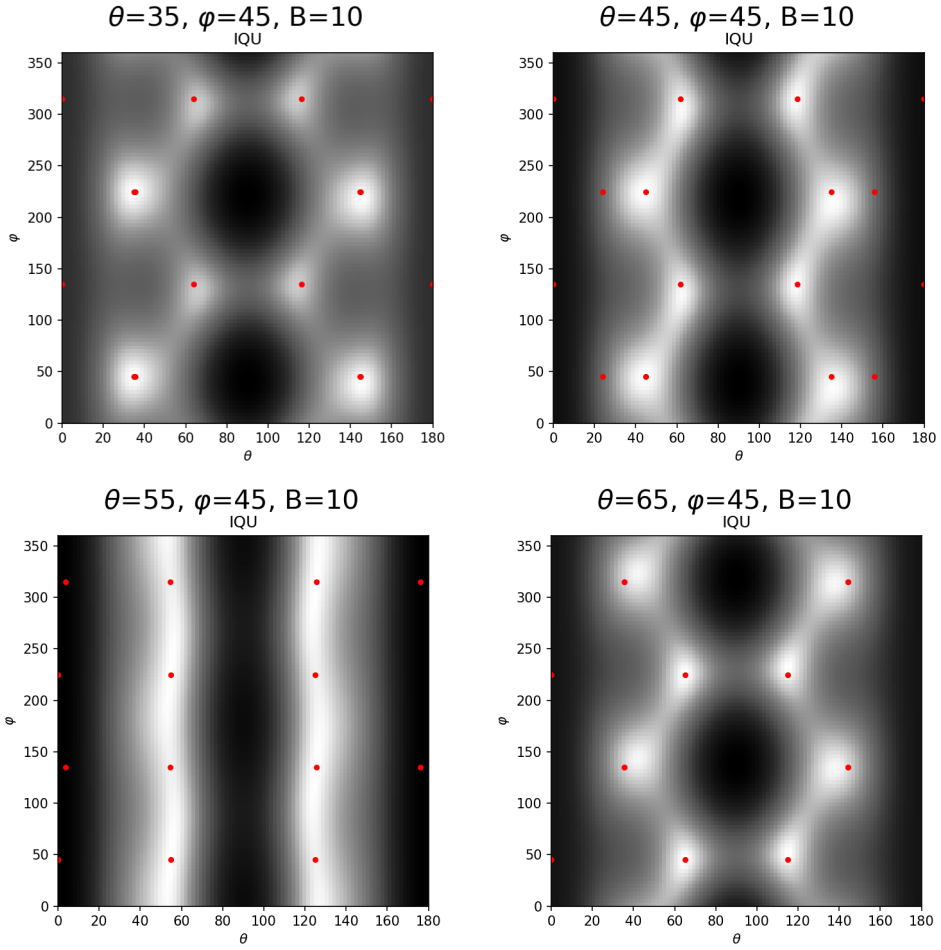

Figure 15: Solutions to the ambiguity equations displayed over the heat maps.

placed, this is because to obtain the solutions an approximate model was used. We can also see that many of the dots plotted are mathematical artifacts that do not match with any of the ambiguities. If any solution to the equations is complex, the real part was plotted.

6 Conclusions:

Resumen: En este proyecto, hemos explorado los efectos fundamentales que influyen en la polarización de la luz solar en la superficie. Observamos cómo el efecto Zeeman desplaza los niveles de energía, resultando en luz polarizada durante transiciones específicas. Además, estudiamos el efecto Hanle, que causa despolarización y rotación del eje de polarización. También investigamos cómo un campo de radiación anisotrópico puede inducir polarización atómica. Estos hallazgos subrayan la complejidad de la teoría de polarización solar, abordada principalmente mediante simulaciones numéricas con HAZEL. Mejorar la precisión polarimétrica es crucial para resolver ambigüedades observacionales, y discutimos el potencial de mediciones simultáneas desde múltiples ángulos para mejorar la técnica de medición y la comprensión general del fenómeno.

In this project, we have explored the fundamental mechanisms influencing the polarization of solar light at the surface.

The Zeeman effect induces shifts in energy levels, which we observed lead to polarized light emission in two-level systems during top-to-bottom transitions. The degree of polarization hinges

on the difference in magnetic quantum numbers between these levels. For more complex level structures, we anticipate a more intricate polarization pattern in transitions.

We have demonstrated that the Hanle effect can be effectively elucidated using a classical electromagnetic model. This effect predominantly results in depolarization of light and rotation of the polarization axis.

Additionally, we explored how an anisotropic radiation field can induce atomic polarization. This phenomenon is primarily governed by the interplay between the system's density matrix and the radiation field tensors. We also examined a simpler scenario where unidirectional unpolarized light selectively promotes energy levels, leading to unbalanced population distributions.

These findings underscore the complexity of developing a comprehensive theory of light polarization in solar atmospheres, often necessitating numerical solutions. In this context, HAZEL stands out as a robust program capable of quantitative solar studies. Its versatility allows for simulations across varied atmospheric conditions, making it invaluable not only for advanced research but also as an educational tool for newcomers to stellar physics.

Understanding the theory purely through polarimetric signals can be challenging. Therefore, we utilized visual representations to enhance comprehension of theoretical concepts and solution behaviors in phase space. Our exploration revealed that the Van Vleck ambiguity is dependent on the polar angle rather than the azimuthal angle in on-disk measurements. Moreover, we observed distinct regimes: weak magnetic fields (around $B < 1$G) exhibit four solutions, while saturation at stronger fields ($B > 10$G) results in eight solutions.

We also addressed the issue of the weakness of the V signal, where noise can lead to duplicated solutions due to ambiguity in determining field polarity. Improvements in polarimetric precision are crucial and expected to advance with evolving measurement technologies.

To mitigate these ambiguities, we discussed the potential of simultaneous measurements from multiple angles at a single point. This approach, while complex and currently feasible primarily through satellite observations or distant terrestrial measurements, holds promise in resolving ambiguities.

Looking forward, we anticipate that advancements in dedicated tools and measurement techniques will strengthen the methodologies discussed here over time. Finally, we validated our findings by analytically calculating ambiguities in a simplified scenario, finding agreement with HAZEL's computational results, thus affirming the program's accuracy.

References

[1] Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloë. *Quantum Mechanics Volume I*. 2nd. Weinheim, Germany: Wiley-VCH, 2006.

[2] M. J. Martínez González et al. "SPECTRO-POLARIMETRIC IMAGING REVEALS HELICAL MAGNETIC FIELDS IN SOLAR PROMINENCE FEET". In: *The Astrophysical Journal* 802.1 (Mar. 2015), p. 3. DOI: 10.1088/0004-637X/802/1/3. URL: https://dx.doi.org/10.1088/0004-637X/802/1/3.

[3] Egidio Landi Degl'Innocenti and Marco Landolfi. *Polarization in Spectral Lines*. Vol. 307. Astrophysics and Space Science Library. Dordrecht: Springer, 2004. ISBN: 9781402014291. DOI: 10.1007/978-1-4020-2415-3.

[4] A. Asensio Ramos, J. Trujillo Bueno, and E. Landi Degl'Innocenti. "Advanced Forward Modeling and Inversion of Stokes Profiles Resulting from the Joint Action of the Hanle and Zeeman Effects". In: *The Astrophysical Journal* 683.1 (Aug. 2008), p. 542. DOI: 10.1086/589433. URL: https://dx.doi.org/10.1086/589433.

[5] Andrés Asensio Ramos. *HAZEL2 Documentation*. Accessed: 2024-03-05. 2023. URL: https://aasensio.github.io/hazel2/index.html.

[6] J. Trujillo Bueno. "Atomic Polarization and the Hanle Effect". In: *Advanced Solar Polarimetry – Theory, Observation, and Instrumentation*. Ed. by Michael Sigwarth. Vol. 236. Astronomical Society of the Pacific Conference Series. Jan. 2001, p. 161. DOI: 10.48550/arXiv.astro-ph/0202328. arXiv: astro-ph/0202328 [astro-ph].

[7] J. Trujillo Bueno. "New Diagnostic Windows on the Weak Magnetism of the Solar Atmosphere". In: *Solar Polarization*. Ed. by Javier Trujillo-Bueno and Jorge Sanchez Almeida. Vol. 307. Astronomical Society of the Pacific Conference Series. Jan. 2003, p. 407.

7 Appendices:

7.1 Appendix 1. Heat maps

```
import numpy as np
from numpy import pi as pi
from HazelAmbig2 import solve_ambiguities
import matplotlib.pyplot as pl
import hazel
import os as os
import time

os.chdir('')

label = ['I', 'Q', 'U', 'V']

def round_to_p(x, p=0):
    """Rounds to p + 1  significant figures

    Args:
        x (float): number to round
        p (int, optional): significant figures - 1. Defaults to 0.

    Returns:
        _type_: _description_
    """
    if x == 0:
        return 0
    return np.round(x, int(-np.floor(np.log10(abs(x))) + p))

def edit_configuration_file(file_path, parameter_name, new_value):
    """Rewrite the variable in the config file

    Args:
        file_path (str): path to config fle
        parameter_name (str): parameter to write
        new_value (float): value
    """
    # Read the content of the file
    with open(file_path, 'r') as file:
        lines = file.readlines()

    # Modify the identified value
    for i, line in enumerate(lines):
        # Strip leading and trailing whitespace from the line
        stripped_line = line.strip()
        # Check if the stripped line starts with the specified
           parameter name
        if stripped_line.startswith(parameter_name):
            # Replace the original line with the modified line
            lines[i] = f"{stripped_line.split(' = ')[0]} = {new_value}\
               n"
            break  # Stop searching once the parameter is found

```

```
    # Write the modified content back to the file
    with open(file_path, 'w') as file:
        file.writelines(lines)

def make_noise(stokes, noiseI, noiseQ, noiseU, noiseV):
    """Creates a profile from 'stokes' with added normal noise

    Args:
        stokes (array): original profile
        noiseI (float): standard deviation for I
        noiseQ (float): standard deviation for Q
        noiseU (float): standard deviation for U
        noiseV (float): standard deviation for V

    Returns:
        array: profile with noise
    """
    noise = np.zeros(stokes.shape)
    noise[0,:] = noiseI
    noise[1,:] = noiseQ
    noise[2,:] = noiseU
    noise[3,:] = noiseV
    stokes_noise = np.copy(stokes)
    stokes_noise[0] += np.random.normal(loc=0, scale=noise[0,0], size=
       stokes[0].shape)
    stokes_noise[1] += np.random.normal(loc=0, scale=noise[1,0], size=
       stokes[0].shape)
    stokes_noise[2] += np.random.normal(loc=0, scale=noise[2,0], size=
       stokes[0].shape)
    stokes_noise[3] += np.random.normal(loc=0, scale=noise[3,0], size=
       stokes[0].shape)

    return stokes_noise, noise

#%%
def make_map_perfil(recalculate = False, confpath='configurations/
   conf_single.ini',
                select = None,
                select_value = None,
                show = False,
                perfil = False,
                res_theta = 100,
                res_phi = 100,
                B_ini = 10,
                theta_ini = 45,
                phi_ini = 45,
                tau_ini = 1,
                v_ini = 0,
                deltav_ini = 8,
                LOS_ini_theta = 0,
                LOS_ini_phi = 0,
                LOS_ini_gamma = 90,
                noiseI = 10e-5,
                noiseQ = 10e-5,
```

```
98              noiseU = 10e-5,
99              noiseV = 10e-5,
100             B_all = None,
101             tau_all = None,
102             v_all = None,
103             deltav_all = None
104             ):
105     """Generates the Heat Map
106
107     Args:
108         recalculate (bool, optional): Should the stokes profiles
109                                       be racalculated?. Defaults to
                                             False.
110         confpath (str, optional): path to config file.
111                                   Defaults to 'configurations/
                                        conf_single.ini'.
112         select (str, optional): variable name to alter.
113                                     Defaults to None.
114         select_value (float, optional): value for 'select'.
115                                         Defaults to None.
116         show (bool, optional): show map.
117                                Defaults to False.
118         perfil (bool, optional): show profile.
119                                  Defaults to False.
120         res_theta (int, optional): number of divitions in theta.
121                                    Defaults to 100.
122         res_phi (int, optional): number of divitions in phi.
123                                  Defaults to 100.
124         B_ini (int, optional): strengh of file of measurement.
125                                Defaults to 10.
126         theta_ini (int, optional): theta of measured.
127                                    Defaults to 45.
128         phi_ini (int, optional): phi of measured.
129                                  Defaults to 45.
130         tau_ini (int, optional): optical thickess for measurement.
131                                  Defaults to 1.
132         v_ini (int, optional): plasma speed for measurement.
133                                Defaults to 0.
134         deltav_ini (int, optional): velocity delta in plasma.
               Measurement.
135                                     Defaults to 8.
136         LOS_ini_theta (int, optional): LOS theta. Measurement.
137                                        Defaults to 0.
138         LOS_ini_phi (int, optional): LOS phi. Measurement.
139                                      Defaults to 0.
140         LOS_ini_gamma (int, optional): LOS gamma. Measurement.
141                                        Defaults to 0.
142         noiseI (float, optional): standard deviation for I signal.
143                                   Defaults to 10e-5.
144         noiseQ (float, optional): standard deviation for Q signal.
145                                   Defaults to 10e-5.
146         noiseU (float, optional): standard deviation for U signal.
147                                   Defaults to 10e-5.
148         noiseV (float, optional): standard deviation for V signal.
```

```
                                    Defaults to 10e-5.
        B_all (float, optional): assumed field strenght when measuring.
                                    Defaults to None.
        tau_all (float, optional): assumed optical thickness when
            measuring.
                                    Defaults to None.
        v_all (float, optional): assumed plasma speed when measuring.
                                    Defaults to None.
        deltav_all (float, optional): assumed plasma speed delta when
            measuring.
                                    Defaults to None.
    """
    def calc_stokes_all():
        """Calculates all the stokes profiles

        Returns:
            array: stoke profiles values
        """
        all_stokes = np.zeros((res_theta, res_phi, 4, 100))
        for i in range(res_theta):
            print(f'Lineas calculadas: {i} de {res_theta}', end='\r')
            for j in range(res_phi):
                mod.atmospheres['ch1'].set_parameters(
                     [Bx_all[i, j], By_all[i, j], Bz_all[i, j],
                     datos['tau_all'], datos['v_all'], datos['
                        deltav_all'],
                     1.0, 0.0], 1.0
                )
                mod.synthesize()

                all_stokes[i, j, :, :] = mod.spectrum['spec1'].stokes
        print()
        print(f'Lineas calculadas: {i} de {res_theta}')
        np.save(npy_file_path, all_stokes)
        return all_stokes

    chromo_check = -1

    datos = {}
    datos['B_ini'] = B_ini
    datos['theta_ini'] = theta_ini
    datos['phi_ini'] = phi_ini
    datos['LOS_ini_theta'] = LOS_ini_theta
    datos['LOS_ini_phi'] = LOS_ini_phi
    datos['LOS_ini_gamma'] = LOS_ini_gamma
    datos['tau_ini'] = tau_ini
    datos['v_ini'] = v_ini
    datos['deltav_ini']= deltav_ini
    datos['noiseI'] = noiseI
    datos['noiseQ'] = noiseQ
    datos['noiseU'] = noiseU
    datos['noiseV'] = noiseV

    datos['B_all'] = B_all
```

```
    datos['tau_all'] = tau_all
    datos['v_all'] = v_all
    datos['deltav_all'] = deltav_all

    if select != None and select not in datos:
        raise ValueError(f"The key '{select}' is not one of the
           dictionary keys."
                         "Must be one of {datos.keys()}")

    if select != None:
        datos[select] = select_value

    for x in datos:
        if datos[x] == None:
            datos[x] = datos[x.replace('all', 'ini')]

    datos_rounded = {k:round_to_p(v, p=2) if (isinstance(v,float) or
       isinstance(v,int))
                     else v for k,v in datos.items()}
    datos_for_names = [v for k,v in datos_rounded.items()]

    if datos['LOS_ini_theta'] == 90:
        chromedef='offlimb'
    else:
        chromedef='disk'
    # Hay que actualizar la cromosfera?
    if chromo_check != datos['LOS_ini_theta']:
        tmp=hazel.tools.File_chromosphere(mode = 'single')
        tmp.set_default(n_pixel = 1, default = chromedef)
        tmp.save('chromospheres/model_chromosphere')
    chromo_check = datos['LOS_ini_theta']

    Bx_ini = datos['B_ini']*np.sin(datos['theta_ini']*pi/180)*np.cos(
       datos['phi_ini']*pi/180)
    By_ini = datos['B_ini']*np.sin(datos['theta_ini']*pi/180)*np.sin(
       datos['phi_ini']*pi/180)
    Bz_ini = datos['B_ini']*np.cos(datos['theta_ini']*pi/180)

    edit_configuration_file(confpath, 'LOS',
                            f"{datos['LOS_ini_theta']}, "
                            f"{datos['LOS_ini_phi']}, "
                            f"{datos['LOS_ini_gamma']}")

    mod = hazel.Model(confpath, working_mode='synthesis', verbose=0)
    mod.atmospheres['ch1'].set_parameters([Bx_ini, By_ini, Bz_ini,
       datos['tau_ini'],
                                           datos['v_ini'], datos['
                                              deltav_ini'],
                                           1.0, 0.0], 1.0)

    mod.synthesize()

    stokes = mod.spectrum['spec1'].stokes
```

```
    stokes_noise, noise = make_noise(stokes, datos['noiseI'], datos['noiseQ'],
                                     datos['noiseU'], datos['noiseV'])

    if perfil == True:
        fig_perfil, ax = pl.subplots(nrows=2, ncols=2, figsize=(10,10))
        fig_perfil.suptitle(r'$\theta$={}, $\varphi$={}, B={}'.format(datos['theta_ini'],
                                                                      datos['phi_ini'],
                                                                      datos['B_ini']), fontsize=20)
        ax = ax.flatten()
        for i in range(4):
            ax[i].plot(mod.spectrum['spec1'].wavelength_axis - 10830, stokes[i,:], color='k')
            ax[i].plot(mod.spectrum['spec1'].wavelength_axis - 10830, stokes_noise[i,:])

        for i in range(4):
            ax[i].set_xlabel('Wavelength - 10830[$\AA$]')
            ax[i].set_ylabel('{0}/Ic'.format(label[i]))
            ax[i].set_xlim([-4,3])

        perfil_image_path = f'Images/Perfiles/Perfil_{datos_for_names}.png'
        pl.savefig(perfil_image_path)
        pl.tight_layout()
    if perfil == False:
        perfil_image_path = None

    theta = np.linspace(pi/(2*res_theta), pi-pi/(2*res_theta), res_theta)
    phi = np.linspace(2*pi/(2*res_phi), 2*pi-2*pi/(2*res_phi), res_phi)

    Bx_all = datos['B_all']*np.outer(np.sin(theta), np.cos(phi))
    By_all = datos['B_all']*np.outer(np.sin(theta), np.sin(phi))
    Bz_all = datos['B_all']*np.outer(np.cos(theta), np.ones(phi.shape[0]))
    chi2 = np.zeros([res_theta, res_phi])
    chi2NoV = np.zeros([res_theta, res_phi])

    npy_datos = [datos_for_names[13],
                 [datos_for_names[14],datos_for_names[15], datos_for_names[16]],
                 datos_for_names[17], datos_for_names[18], datos_for_names[19]]
    npy_file_path = ('Perfiles calculados/models_' +
                     f'{npy_datos}.npy')
    if recalculate:
```

```
286         print(f'Calculando perfil con {npy_file_path.split("_")[-1]}')
287         st = time.time()
288         calc_stokes_all()
289         et = time.time()
290         # print('Ha tardado {} minutos.'.format((et-st)/60))
291     else:
292         if not os.path.exists(npy_file_path):
293             print(f'Calculando perfil con {npy_file_path.split("_")
                    [-1]}')
294             st = time.time()
295             calc_stokes_all()
296             et = time.time()
297             # print('Ha tardado {} minutos.'.format((et-st)/60))
298 
299         all_stokes = np.load(npy_file_path)
300 
301     print()
302     for i in range (res_theta):
303         for j in range(res_phi):
304             chi2[i, j] = np.mean(((all_stokes[i, j] - stokes_noise) /
                    noise) ** 2)
305             chi2NoV[i, j] = np.mean(((all_stokes[i, j, 0:3] -
                    stokes_noise[0:3]) / noise[0:3]) ** 2)
306 
307 
308     fig_mapa, ax = pl.subplots(nrows=1, ncols=2, figsize=(10,5),
309                                sharey=True, dpi=150)
310 
311     titulo =   (fr"$\theta={datos_rounded['theta_ini']}^o$, "
312                 fr"$\varphi={datos_rounded['phi_ini']}^o$, "
313                 f"B={datos_rounded['B_all']} [G]")
314     if select is not None:
315         titulo = titulo + f", {select}={select_value}"
316 
317     fig_mapa.suptitle(titulo, fontsize=20)
318 
319     # ambig = solve_ambiguities(0, theta_ini*pi/180, phi_ini*pi/180,
320     #                           theta_ini*pi/180, phi_ini*pi/180)
321     # ax[0].scatter(ambig[:,:,0]*180/pi, ambig[:,:,1]*180/pi,
322     #               color='r', marker='o', s=10, label='Real')
323 
324     ax[0].imshow(np.transpose(np.log10(chi2NoV)), cmap='Greys',
325                  extent=(0, 180, 0, 360), origin='lower', aspect=.5)
326     ax[0].set_title('IQU')
327     ax[0].scatter(datos['theta_ini'], datos['phi_ini'],
328                  color='r', marker='o', s=10, label='Real')
329     ax[0].set_xlabel(r'$\theta$ [deg]')
330     ax[0].set_ylabel(r'$\varphi$ [deg]')
331     ax[1].set_title('IQUV')
332     ax[1].set_xlabel(r'$\theta$ [deg]')
333     ax[1].imshow(np.transpose(np.log10(chi2)), cmap='Greys',
334                  extent=(0, 180, 0, 360), origin='lower', aspect=.5)
335     ax[1].scatter(datos['theta_ini'], datos['phi_ini'],
336                  color='r', marker='o', s=10, label='Real')
```

```
337
338     if show == True:
339         pl.show()
340     map_image_path = fr'Images/Mapas/Mapa_{datos_for_names}.png'
341     fig_mapa.savefig(map_image_path)
342     pl.close()
343     return map_image_path, perfil_image_path, datos_rounded
344
345 if __name__ == '__main__':
346
347     make_map_perfil(perfil=False, show=True,
348                     theta_ini=45, phi_ini=45)
```

7.2 Appendix 2. Sculptures

```
1
2  import numpy as np
3  from MapaHazel import round_to_p
4  from numpy import pi as pi
5  import plotly.graph_objects as go
6  from plotly.subplots import make_subplots
7  import hazel
8  import os as os
9
10 os.chdir('')
11
12 label = ['I', 'Q', 'U', 'V']
13
14 def edit_configuration_file(file_path, parameter_name, new_value):
15
16     with open(file_path, 'r') as file:
17         lines = file.readlines()
18
19     for i, line in enumerate(lines):
20
21         stripped_line = line.strip()
22
23         if stripped_line.startswith(parameter_name):
24
25             lines[i] = f"{stripped_line.split(' = ')[0]} = {new_value}\
                n"
26             break
27
28     with open(file_path, 'w') as file:
29         file.writelines(lines)
30
31
32 def make_noise(stokes, noiseI, noiseQ, noiseU, noiseV):
33     noise = np.ones(stokes.shape)
34     noise[0,:] = noiseI
35     noise[1,:] = noiseQ
36     noise[2,:] = noiseU
37     noise[3,:] = noiseV
```

```
    stokes_noise = np.copy(stokes)
    stokes_noise[0] += np.random.normal(loc=0, scale=noise[0,0], size=
       stokes[0].shape)
    stokes_noise[1] += np.random.normal(loc=0, scale=noise[1,0], size=
       stokes[0].shape)
    stokes_noise[2] += np.random.normal(loc=0, scale=noise[2,0], size=
       stokes[0].shape)
    stokes_noise[3] += np.random.normal(loc=0, scale=noise[3,0], size=
       stokes[0].shape)

    return stokes_noise, noise

def make_map_perfil(recalculate = False, confpath='configurations/
   conf_single.ini',
                select = None,
                select_value = None,
                res_theta = 100,
                res_phi = 100,
                B_ini = 10,
                theta_ini = 45,
                phi_ini = 45,
                tau_ini = 1,
                v_ini = 0,
                deltav_ini = 8,
                LOS_ini_theta = 0,
                LOS_ini_phi = 0,
                LOS_ini_gamma = 90,
                noiseI = 10e-5,
                noiseQ = 10e-5,
                noiseU = 10e-5,
                noiseV = 10e-5,
                B_all = None,
                tau_all = None,
                v_all = None,
                deltav_all = None
                ):
    """Same as the Heat Maps but adapted to return the points.
    """
    def calc_stokes_all():
        all_stokes = np.zeros((res_theta, res_phi, 4, 100))
        for i in range(res_theta):
            print(f'Lineas calculadas: {i} de {res_theta}', end='\r')
            for j in range(res_phi):
                mod.atmospheres['ch1'].set_parameters(
                     [Bx_all[i, j], By_all[i, j], Bz_all[i, j],
                     datos['tau_all'], datos['v_all'], datos['
                        deltav_all'],
                     1.0, 0.0], 1.0
                )
                mod.synthesize()

                all_stokes[i, j, :, :] = mod.spectrum['spec1'].stokes
        print()
        print(f'Lineas calculadas: {i} de {res_theta}')
```

```
        np.save(npy_file_path, all_stokes)
        return all_stokes

    chromo_check = -1

    datos = {}
    datos['B_ini'] = B_ini
    datos['theta_ini'] = theta_ini
    datos['phi_ini'] = phi_ini
    datos['LOS_ini_theta'] = LOS_ini_theta
    datos['LOS_ini_phi'] = LOS_ini_phi
    datos['LOS_ini_gamma'] = LOS_ini_gamma
    datos['tau_ini'] = tau_ini
    datos['v_ini'] = v_ini
    datos['deltav_ini']= deltav_ini
    datos['noiseI'] = noiseI
    datos['noiseQ'] = noiseQ
    datos['noiseU'] = noiseU
    datos['noiseV'] = noiseV

    datos['B_all'] = B_all
    datos['tau_all'] = tau_all
    datos['v_all'] = v_all
    datos['deltav_all'] = deltav_all

    if select != None and select not in datos:
        raise ValueError(f"The key '{select}' is not one of the
           dictionary keys."
                         "Must be one of {datos.keys()}")

    if select != None:
        datos[select] = select_value

    for x in datos:
        if datos[x] == None:
            datos[x] = datos[x.replace('all', 'ini')]

    datos_rounded = {k:round_to_p(v, p=2) if (isinstance(v,float) or
       isinstance(v,int))
                     else v for k,v in datos.items()}
    datos_for_names = [v for k,v in datos_rounded.items()]

    if datos['LOS_ini_theta'] == 90:
        chromedef='offlimb'
    else:
        chromedef='disk'
    # Hay que actualizar la cromosfera?
    if chromo_check != datos['LOS_ini_theta']:
        tmp=hazel.tools.File_chromosphere(mode = 'single')
        tmp.set_default(n_pixel = 1, default = chromedef)
        tmp.save('chromospheres/model_chromosphere')
    chromo_check = datos['LOS_ini_theta']

```

```
    Bx_ini = datos['B_ini']*np.sin(datos['theta_ini']*pi/180)*np.cos(
        datos['phi_ini']*pi/180)
    By_ini = datos['B_ini']*np.sin(datos['theta_ini']*pi/180)*np.sin(
        datos['phi_ini']*pi/180)
    Bz_ini = datos['B_ini']*np.cos(datos['theta_ini']*pi/180)


    edit_configuration_file(confpath, 'LOS',
                            f"{datos['LOS_ini_theta']}, "
                            f"{datos['LOS_ini_phi']}, "
                            f"{datos['LOS_ini_gamma']}")

    mod = hazel.Model(confpath, working_mode='synthesis', verbose=0)
    mod.atmospheres['ch1'].set_parameters([Bx_ini, By_ini, Bz_ini,
        datos['tau_ini'],
                                           datos['v_ini'], datos['
                                              deltav_ini'],
                                           1.0, 0.0], 1.0)

    mod.synthesize()

    stokes = mod.spectrum['spec1'].stokes
    stokes_noise, noise = make_noise(stokes, datos['noiseI'], datos['
        noiseQ'],
                                              datos['noiseU'], datos['
                                                 noiseV'])

    theta = np.linspace(pi/(2*res_theta), pi-pi/(2*res_theta),
        res_theta)
    phi = np.linspace(2*pi/(2*res_phi), 2*pi-2*pi/(2*res_phi), res_phi)

    Bx_all = datos['B_all']*np.outer(np.sin(theta), np.cos(phi))
    By_all = datos['B_all']*np.outer(np.sin(theta), np.sin(phi))
    Bz_all = datos['B_all']*np.outer(np.cos(theta), np.ones(phi.shape
        [0]))
    chi2 = np.zeros([res_theta, res_phi])
    chi2NoV = np.zeros([res_theta, res_phi])

    npy_datos = [datos_for_names[13],
                 [datos_for_names[14],datos_for_names[15],
                    datos_for_names[16]],
                 datos_for_names[17], datos_for_names[18],
                    datos_for_names[19]]
    npy_file_path = ('Perfiles calculados/models_' +
                     f'{npy_datos}.npy')
    if recalculate:
        print(f'Calculando perfil con {npy_file_path.split("_")[-1]}')

        calc_stokes_all()

        all_stokes = np.load(npy_file_path)

    else:
        if not os.path.exists(npy_file_path):
```

```
181             print(f'Calculando perfil con {npy_file_path.split("_")
                    [-1]}')
182 
183             calc_stokes_all()
184 
185         all_stokes = np.load(npy_file_path)
186 
187     print()
188     for i in range (res_theta):
189         for j in range(res_phi):
190             chi2[i, j] = np.mean(((all_stokes[i, j] - stokes_noise) /
                    noise) ** 2)
191             chi2NoV[i, j] = np.mean(((all_stokes[i, j, 0:3] -
                    stokes_noise[0:3]) / noise[0:3]) ** 2)
192 
193     logChi2 = np.log10(chi2)
194     logChi2NoV = np.log10(chi2NoV)
195 
196     return [logChi2, logChi2NoV, res_theta, res_phi, theta, phi]
197 
198 def sculpt(select, select_values, max_chi=.5):
199     """Creates the sculptures from the dots of
200     the heat maps.
201 
202     Args:
203         select (str): variable name for z axis
204         select_values (float): values for 'select'
205         max_chi (float, optional): max value of logchi2 for the dots.
206                                    Defaults to .5.
207     """
208 
209     k = 0
210     for select_value in select_values:
211 
212         return_list = make_map_perfil(select = select, select_value =
                select_value)
213 
214         logChi2 = return_list[0]
215         logChi2NoV = return_list[1]
216         res_theta = return_list[2]
217         res_phi = return_list[3]
218         theta = return_list[4]
219         phi = return_list[5]
220 
221         print(logChi2.shape[0])
222 
223         if k == 0:
224             lista_puntos = np.zeros([len(select_values)*res_theta*
                    res_phi, 5])
225 
226         for i in np.arange(res_theta):
227             for j in np.arange(res_phi):
228                 N = k*res_theta*res_phi + i*res_phi + j
229                 lista_puntos[N,0] = theta[i]*180/pi
```

```
                lista_puntos[N,1] = phi[j]*180/pi
                lista_puntos[N,2] = select_value
                lista_puntos[N,3] = logChi2[i,j]
                lista_puntos[N,4] = logChi2NoV[i,j]

        k += 1

np.random.seed(0)
x = lista_puntos[:,0]
y = lista_puntos[:,1]
z = lista_puntos[:,2]
t = lista_puntos[:,3]

filtered_x = x[t < max_chi]
filtered_y = y[t < max_chi]
filtered_z = z[t < max_chi]
filtered_t = t[t < max_chi]

trace = go.Scatter3d(
    x=filtered_x,
    y=filtered_y,
    z=filtered_z,
    mode='markers',
    marker=dict(
        size=5,
        color=filtered_t,
        colorscale='Greys',
        opacity=0.8
    ),
    text=filtered_t,
    hovertemplate=
            'theta: %{x}<br>'+
            'phi: %{y}<br>'+
            'B_ini: %{z}<br>'+
            'logChi2: %{text}<br>',
)

fig = go.Figure(data=[trace])

fig.update_layout(
    scene=dict(
        aspectmode='manual',
        aspectratio=dict(x=1, y=2, z=1),
        yaxis=dict(range=[0, 360]),
        xaxis=dict(range=[0, 180]),
        xaxis_title='theta[deg]',
        yaxis_title='phi [deg]',
        zaxis_title=f'{select} [gauss]'
    ),
    title={
        'text': f"3D Scatter Plot with logChi2 < {max_chi}",
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
```

```
            'yanchor': 'top'
        }
    )

    fig.show()

if __name__ == '__main__':

    sculpt('theta_ini', np.linspace(0, 90, 21), max_chi=.4)
```

7.3 Appendix 3. Video generator

```
import numpy as np
import imageio
import os as os
import MapaHazel as mh

os.chdir('')

def make_video(select,
               select_values):
    """Generates a video of the heatmaps for all
    values passed for the selected variable

    Args:
        select (str): variable name
        select_values (array): values to give to 'select'
    """

    if not ((isinstance(select_values, np.ndarray) or isinstance(
       select_values, list)) and len(select_values)!=1):
        print('Must send numpy array with more than one value to
           iterate for the video.')
        return

    first_loop = True
    fotogramas_tot = len(select_values)
    fotograma = 0
    print("\033[2J")
    for select_value in select_values:
        print(f'\033[2;0HFotogramas generados: {fotograma} de {
           fotogramas_tot}')

        mapa_image_path, _, datos_rounded = mh.make_map_perfil(
           recalculate = False, select = select,
                select_value = select_value)

        if first_loop == True:
            datos_rounded[select] = (
                f'{mh.round_to_p(select_values[0],2)}-{mh.round_to_p(
                   select_values[-1],2)}')
            datos_for_names = [v for _,v in datos_rounded.items()]
```

```
            video_filename = f'Images/Videos/HAZEL-{datos_for_names}.mp4'
            writer_mapa = imageio.get_writer(video_filename, fps = 10)
            first_loop = False

        image = imageio.v3.imread(mapa_image_path)
        writer_mapa.append_data(image)

        fotograma += 1
    print(f'\033[2;0HFotogramas generados: {fotograma} de {fotogramas_tot}')

    writer_mapa.append_data(image)
    writer_mapa.close()

if __name__ == '__main__':

    make_video('LOS_ini_gamma', np.linspace(0, 90, 11))
    print()
```

7.4 Appendix 4. Ambiguities

```
import numpy as np
from numpy import pi as pi
import ctypes
import matplotlib.pyplot as pl

def calc_Q(theta_B, Theta_B, Phi_B):
    return (3*np.cos(theta_B)**2 - 1)*np.sin(Theta_B)**2*np.cos(2*Phi_B)
def calc_U(theta_B, Theta_B, Phi_B):
    return (3*np.cos(theta_B)**2 - 1)*np.sin(Theta_B)**2*np.sin(2*Phi_B)

def differences(case, Phi_B):
    """Returns the case differences for the coefficients.

    Args:
        case (int): number of the case
        Phi_B (_type_): Phi_B

    Returns:
        int, float: return the correct values for each case
    """
    # Case Phi = Phi'
    if case == 1:
        return -1, +1, np.cos(Phi_B), 0
    # Case Phi = Phi' + pi
    elif case == 2:
        return +1, +1, np.cos(Phi_B), pi
    # Case Phi = Phi' + pi/2
    elif case == 3:
        return +1, -1, np.sin(Phi_B), pi/2
```

```
    # Case Phi = Phi' - pi/2
    elif case == 4:
        return -1, -1, np.sin(Phi_B), -pi/2

def solve_ambiguities(theta, Theta_B, Phi_B, theta_B, phi_B):
    """Only for disk center solutions

    Args:
        theta (float): LOS theta
        Theta_B (float): field theta from LOS
        Phi_B (float): field phi from LOS
        theta_B (float): field theta from vertical
        phi_B (float): field phi from vertical

    Returns:
        _type_: possible solution values
    """

    cases = [1, 2, 3, 4]

    coeff = np.zeros(5)
    ambiguities = np.zeros([4, 8, 7], dtype='complex_')

    n = 0
    for case in cases:

        sign1, sign2, sin_cos_Phi_B, fase = differences(case, Phi_B)

        A = -3*np.cos(theta)**2 + 3*np.sin(theta)**2*sin_cos_Phi_B**2

        B = 3*np.cos(theta)**2 - 1

        C = sign1*6*np.cos(theta)*np.sin(theta)*sin_cos_Phi_B

        K = sign2*(3*(np.cos(Theta_B)*np.cos(theta)
                - np.sin(theta)*np.sin(Theta_B)*np.cos(Phi_B))**2 - 1)*
                   np.sin(Theta_B)**2

        # Coeff of equation in Z
        coeff = [
             (C**2 + A**2),
             (-C**2 + 2*A*B),
             (-2*A*K + B**2),
             (-2*B*K),
             (K**2)
            ]

        Z_roots = np.roots(coeff).astype(complex)

        t = np.zeros(4, dtype='complex_')
        # ind_real = np.where(np.abs(Z_roots.imag) < 1e-15)
        # Z_roots = Z_roots[ind_real]
        t = np.sqrt(Z_roots)
        t = np.append(t, -t)
```

```
83
84          Theta_B_prime = np.arcsin(t).real % np.pi
85          Phi_B_prime = (Phi_B + fase)
86
87          Q = calc_Q(theta_B, Theta_B, Phi_B)
88          U = calc_U(theta_B, Theta_B, Phi_B)
89
90          print(Q, U)
91
92          k = 0
93          for i, solution in enumerate(Theta_B_prime):
94              Q_t = calc_Q(solution, solution, Phi_B_prime)
95              U_t = calc_U(solution, solution, Phi_B_prime)
96              DeltaQ = np.abs(Q - Q_t)
97              DeltaU = np.abs(U - U_t)
98
99              ambiguities[n, k, 0] = solution
100             ambiguities[n, k, 1] = ( Phi_B_prime + np.pi) % (2 * np.pi
                    ) - np.pi    # Transform to -pi, pi
101             ambiguities[n, k, 2] = Q_t
102             ambiguities[n, k, 3] = U_t
103             ambiguities[n, k, 4] = DeltaQ
104             ambiguities[n, k, 5] = DeltaU
105
106             if (DeltaQ < 1e-3 and DeltaU < 1e-3):
107                 ambiguities[n, k, 6] = 1
108                 print(t[i],'| |', solution * 180/np.pi,'| |',
                        Phi_B_prime * 180/np.pi)
109
110             k += 1
111
112         n += 1
113
114     return ambiguities
115
116 def compute_chi2(Theta_B, Phi_B, theta_B, phi_B):
117     Qref = calc_Q(theta_B, Theta_B, Phi_B)
118     Uref = calc_U(theta_B, Theta_B, Phi_B)
119
120     TB = np.linspace(0.0, np.pi, 100)
121     PB = np.linspace(-np.pi, np.pi, 100)
122     TB, PB = np.meshgrid(TB, PB)
123     Q = calc_Q(TB, TB, PB)
124     U = calc_U(TB, TB, PB)
125     chi2 = (Q - Qref)**2 / 0.1**2 + (U - Uref)**2 / 0.1**2
126
127     return TB, PB, chi2
```