UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE POSGRADO EN ASTROFÍSICA

INSTITUTO DE ASTRONOMÍA

Algoritmo de trazado de rayos exacto para la determinación del punto de coma cero en telescopios reflectores clásicos y no clásicos

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIAS (ASTROFÍSICA)

PRESENTA:

Morgan Rhaí Najera Roa

TUTORES:

Dr. Carlos Alberto Guerrero Peña Instituto de Astronomía

Dr. Joel Herrera Vázquez Instituto de Astronomía

Ensenada, Baja California, Enero, 2022

A Marisela Navarro y Demian Najera,
mi fuente de inspiración y motivación.

A Rafael Najera, la estrella que me ilumina.

Agradecimientos

Si bien siempre es mejor creer en uno mismo,
un poco de ayuda de los demás puede ser una gran bendición.

A mis asesores, Dr. Carlos Alberto Guerrero Peña y Dr. Joel Herrera Vázquez. Por todo su apoyo, conocimiento, tiempo, dedicación y paciencia a lo largo del desarrollo de este proyecto.

A Marisela Navarro que desde el día que nací me ha apoyado en todas las decisiones que he tomado a lo largo de mi corta vida. Asimismo, como apoyarme financiera y emocionalmente cuando lo necesité. Eres una gran madre, amiga y sobre todo una gran persona.

A mis grandes amistades que realicé en mis años de licenciatura en la UAM. A Ivan Castorena y Martin Rabelo, por todo su apoyo y comprensión durante esta etapa. A Emmanuel Calderón por tu amistad y motivación en esas tardes de estudio. Mariana Ladrón y Atila Orlov por guiarme a lo largo de la licenciatura. Mariana Palacios, Yolotzín Bazaldúa y Leonora Gomez por ser tan buenas personas y apoyarme desde el momento que nos conocimos.

A mis amigos que hice durante el desarrollo de este pequeño proyecto llamado maestría. A Elizabeth Moreno y Andrés Sixtos que me aconsejaron, escucharon y se

relajaron conmigo cuando no podía más. Hecsari Bello y Luis Bermudez por ayudarme al inicio de la maestría y su compañía.

A Horacio Najera que cuidó de mí y me guió cuando falleció mi papá. Asimismo, me apoyó tanto financiera como emocionalmente a lo largo de este pequeño viaje.

A Yoalli Téllez que aunque ya no estemos juntos fue y será una persona muy importante en este viaje y sobre todo en mi vida. Me apoyaste cuando todo comenzó y me apoyas aún cuando todo esto ya terminó. Agradezco la paciencia y determinación que tuviste el tiempo que estuvimos juntos. Fuiste y serás la inspiración de este proyecto.

Finalmente a CONACYT, que sin duda alguna sin el apoyo que recibí de su parte no hubiera podido realizar mi sueño de terminar una maestría en mi ciudad favorita, Ensenada.

Índice general

Resumen

Los problemas científicos empujan los límites de la calidad de imagen que los telescopios deben proporcionar. De acuerdo con los propósitos de esta tesis, con calidad de imagen nos referimos a un cierta tolerancia en las aberraciones ópticas, basada en el tamaño del diagrama de manchas con respecto al límite de difracción, considerando el radio del disco de Airy. Para alcanzar esta tolerancia, se requiere que todos los elementos ópticos de un telescopio estén en una casi perfecta coaxialidad. De esta manera, el proceso de colimación de los espejos de un telescopio astronómico consiste en alinear sus ejes ópticos para llevarlos a su posición de diseño, donde la calidad de imagen es óptima para la ciencia que se deseé realizar. Existen varios procedimientos de colimación, sin embargo, usaremos el cálculo del punto de coma cero, donde el espejo secundario se mueve sobre un punto físico que permite una colimación determinista. En este trabajo, presentamos un algoritmo general para calcular el punto de coma cero para telescopios reflectores clásicos y no clásicos con superficies asféricas. Como parte de nuestro algoritmo, programamos una función de trazado de rayos general aplicable a superficies cónicas y superficies con coeficientes de asfericidad, desplazamientos e inclinaciones, que pueden ser modificadas por los usuarios, de acuerdo con sus especificaciones. Esta función de trazado de rayos se utiliza para evaluar la coma transversal y determinar la conjunción óptima de desplazamiento e inclinación que evita la introducción de coma axial durante el proceso de colimación. Como ejemplo del uso de nuestro algoritmo, calculamos el punto de coma cero para los telescopios de 0.84 m, 1.5 m, 2.1 m, 1.3 m (TAOS-II), 1.0 m (SAINT-EX) y 6.5 m (TSPM) pertenecientes al OAN-SPM, que se

localiza en la Sierra de San Pedro Mártir, Baja California, México. Asimismo, incluimos el telescopio de 2.12 m del Observatorio Astrofísico Guillermo Haro, localizado en Cananea, Sonora, México. Por otra parte, también presentamos un ejemplo de un telescopio de campo amplio con superficies asféricas, como será el caso de los telescopios del proyecto TAOS-II (Tansneptunian Automated Occultation Survey), donde no se puede utilizar la expresión analítica clásica.

Abstract

Scientific questions push the limits of the image quality that telescopes must provide. For the purposes of this thesis, image quality refers to a certain tolerance in optical aberrations, based on the size of the speckle pattern with respect to the diffraction limit considering the radius of the Airy disk. In order to achieve this tolerance, we require that every optical element of a telescope to be in near perfect coaxiality. In this way, the collimation process of the mirrors of an astronomical telescope consists of aligning their optical axis to bring them to their design position, where the image quality is optimal for the science to be performed. There are several procedures for the collimation process, however, we will use the calculation of the zero coma point, where the secondary mirror is moving around a physical point that allows a deterministic collimation. In this work, we present a general algorithm to calculate the zero coma point for classical and non classical reflective telescopes with aspherical surfaces. As part of our algorithm, we programmed a general ray tracing function applicable to conic surfaces and surfaces with asphericity coeficients, displacements and inclinations, which can be modified by the users, according to their specifications. This ray-tracing function is used to evaluate the transverse coma and determine the optimal conjunction of displacement and tilt that avoids the introduction of axial coma during the collimation process. As an example of the usage of our algorithm, we calculate the zero point for the telescopes of 0.84 m, 1.5 m, 2.1 m, 1.3 m (TAOS-II), 1.0 m (SAINT-EX) and 6.5 m (TSPM) belonging to the OAN-SPM, located in the Sierra de San Pedro Mártir, Baja California, Mexico. We also included the 2.12 m telescope of the Gui-

llermo Haro Astrophysical Observatory, located in Cananea, Sonora, Mexico. We also present an example of a wide-field telescope with aspherical surfaces, as will be the case of the telescopes of the TAOS-II project (Tansneptunian Automated Occultation Survey), where the classical analytical expression cannot be used.

Capítulo 1

Introducción

En un inicio los telescopios estuvieron limitados por el estado de la técnica involucrada en su construcción. Hoy en día, en mi consideración, se constituyen como uno de los sistemas más precisos que la humanidad ha creado, en virtud de la extraordinaria sensibilidad y la gama de longitudes de onda con la que funcionan ahora los telescopios. Han transformado nuestro conocimiento del Universo, al hacer visible lo invisible (Cottrell, 2016).

Debido a la precisión requerida en los telescopios, estos sistemas deben poseer una adecuada coaxialidad entre todos sus elementos ópticos. Por lo tanto, para telescopios compuestos de dos espejos, tenemos el punto de coma cero (ZCP, por sus siglas en ingles "*Zero Coma Point*"), el cual determina un lugar físico sobre el eje óptico del espejo primario, M_1, donde el espejo secundario, M_2, puede ser pivoteado durante el proceso de colimación sin introducir coma transversal en la imagen resultante. La expresión analítica para el ZCP en telescopios reflectores aplanáticos (ver Sección 3.2) es bien conocida y resulta de gran relevancia en el alineamiento de telescopios clásicos, como los telescopios tipo Cassegrain, Gregoriano o Ritchey-Chrétien (ver Sección 2.5.2); sin embargo, esta ecuación no se aplica directamente a algunas variantes de telescopios de

campo amplio, los cuales requieren no solamente de superficies cónicas sino también términos de asfericidad en el espejo secundario, o telescopios con lentes asféricas correctivas incorporadas como se muestra en la Sección 2.1.2.

En el presente trabajo, se desarrolló un algoritmo de trazo exacto de rayos para superficies asféricas fuera de eje, en el cual se implementa la determinación del punto de coma cero. Como ejemplo de la aplicación de este algoritmo, se presenta el cálculo del ZCP para los telescopios de 0.84 m, 1.5 m, 2.1 m, 1.3 m (TAOS-2), 1.0 m (SAINT-EX) y 6.5 m (TSPM) pertenecientes al OAN-SPM. Además de incluir el telescopio de 2.12 m del Observatorio Astrofísico Guillermo Haro y un ejemplo de telescopio de campo amplio .

El contenido de este trabajo se divide de la siguiente manera: en el Capítulo 2 se presentan brevemente conceptos útiles sobre óptica; se describen los dos tipos de telescopios y, además, se expone el planteamiento del problema que nos llevó a realizar este trabajo, así como los objetivos tanto generales como particulares. En el Capítulo 3 se describe el diseño óptico a tercer orden con el cual se pueden corregir las aberraciones debidas al diseño óptico (ver Sección 2.3), así como la solución clásica para determinar el punto de coma cero. En el Capítulo 4 presentamos dos metodologías de trazado de rayos, tanto el matricial como el trazo exacto de rayos; sin embargo, en el trazo exacto de rayos podremos introducir rotaciones y desplazamientos a superficies ópticas de modo que, utilizando el criterio de la aberración de coma transversal, estemos en condiciones de escribir un algoritmo con una función que se utiliza para determinar la conjunción óptima de desplazamiento e inclinación del espejo secundario, que evita la introducción de coma axial durante el proceso de colimación. En el Capítulo 5 se presentan los valores del ZCP para los seis telescopios mencionados anteriormente pertenecientes al OAN-SPM , además de incluir el telescopio de 2.12 m del OAGH,

utilizando nuestro algoritmo y la expresión clásica, también incluimos un ejemplo de un telescopio de campo amplio teórico, similar a los telescopios del proyecto TAOS-II, como muestra de la aplicación de nuestro algoritmo. En el Capítulo 6 presentamos el artículo: "*OFF-AXIS EXACT RAY TRACING ALGORITHM FOR ZERO COMA POINT DETERMINATION IN CLASSICAL AND NON-CLASSICAL REFLECTIVE TELESCOPES*" (Herrera, Nájera & Guerrero, 2021), que condensa los resultados de esta tesis y que fue aceptado para publicación en la Revista Mexicana de Astronomía y Astrofísica. Finalmente, las conclusiones se exponen en el Capítulo 7.

Capítulo 2

Óptica básica

2.1. Refracción y Reflexión

Para entender los efectos que ocurren en un sistema óptico que interactúa con la luz proveniente de un objeto astronómico, debemos empezar por comprender los fenómenos de refracción y reflexión en la aproximación de la óptica geométrica. La refracción ocurre cuando el camino de un haz de luz que viaja de un medio transparente a otro, tiene una desviación en la interfaz. La desviación es consecuencia de que la velocidad de la luz es diferente en los dos medios, por ello se define el índice de refracción (n) como la razón de la velocidad de la luz en el vacío (c) con respecto a la velocidad en el medio (v) (Kidger, 2001), esto es:

$$n = \frac{c}{v}. \tag{2.1}$$

La frecuencia de un haz de luz que viaja de un medio a otro no cambia, pero sí su longitud de onda. Si esto no ocurriese, los frentes de onda se apilarían en la interfaz de los medios y estos se destruirían o superpondrían (Serway R. A. and S., 2013). Asimismo, tendremos dos ángulos medidos con respecto a la normal de la interfaz de los dos medios. El primero es θ_1, conocido como el ángulo de incidencia y el segundo será θ_2, el cual es el ángulo de refracción.

Este fenómeno puede ser entendido usando la ley de Snell, dada por:

$$n_{1\lambda}\sin(\theta_1) = n_{2\lambda}\sin(\theta_2). \tag{2.2}$$

La reflexión acaece cuando un haz de luz viajando en un medio incide sobre una superficie lisa. Los ángulos formados por el haz incidente y el haz reflejado deben cumplir la ley de reflexión, esto es:

$$\theta_1 = \theta_2. \tag{2.3}$$

Se observa que a diferencia de la refracción, en la reflexión no ocurre un cambio en la velocidad de la luz o su longitud de onda, ya que no hay un cambio de medio.

La ley de Snell además de utilizarse en su forma usual, como lo muestra la Ecuación 2.2, también, se puede encontrar en su forma vectorial:

$$\hat{s} = \frac{n_1}{n_2}\left[\hat{N}\times(-\hat{N}\times\hat{s}_1)\right] - \hat{N}\sqrt{1-\left(\frac{n_1}{n_2}\right)^2(\hat{N}\times\hat{s}_1)\cdot(\hat{N}\times\hat{s}_1)}, \tag{2.4}$$

donde $\hat{N}$ es el vector normal y $\hat{s}_1$ es el vector incidente a la superficie. Para transformar la ley de Snell de la forma usual a la forma vectorial se parte de que: $\frac{n_1}{n_2}\sin(\theta_1) = \sin(\theta_2)$, además de la definición del producto, punto dada por: $\hat{N}\times\hat{s}_1 = |\hat{N}||\hat{s}_1|\sin(\theta_1)\hat{\eta}$, donde $\hat{\eta}$ es un vector perpendicular al rayo y a la normal de la superficie. Combinando la ley de Snell escalar y la identidad $\cos^2(\theta_2) = 1-\sin^2(\theta_2)$, observamos la componente negativa del haz refractado a lo largo de la normal a la superficie, $-\hat{N}\cdot\hat{s}_2 = |\hat{N}||\hat{s}_2|\cos(\theta)$, e incorporando la componente a lo largo de la superficie en el plano de incidencia, $\sin(\theta_2)(\hat{N}\times\hat{\eta})$, obtenemos finalmente la ecuación buscada.

2.1.1. Superficies refractoras

Un haz de luz colimada con una determinada longitud de onda que incide sobre una superficie, puede ser llevado al foco en un punto a lo largo del eje óptico, estas superficies son conocidas como convergentes (Hecht, 2016). En consecuencia, una imagen luminosa que se forme en una pantalla colocada en el foco, se dice que es real. Al contrario, el haz diverge con superficies divergentes (Hecht, 2016). En este caso, el haz parecerá que se originó en un punto a lo largo del eje óptico, si se pusiera una pantalla en ese punto no se generaría ninguna imagen luminosa y nos referiríamos a ella como virtual. Dicho punto a lo largo del eje donde es orientado el haz se conoce como punto focal, además, el plano que cruza el punto focal y es perpendicular al eje óptico, se le llama plano focal (Hecht, 2016). El eje óptico es el eje de simetría (imaginario) de un sistema óptico (Kidger, 2001) y nos referimos a la longitud focal o distancia focal (f) como la distancia del vértice de la superficie al punto focal. En superficies convergentes la distancia focal es positiva y para superficies divergentes es negativa (Hecht, 2016).

Cuando un haz incide sobre una de las superficies mencionadas anteriormente y ocurre el fenómeno de refracción se tiene que la superficie es una lente. Existe una amplia gama de formas de lentes que pueden tener dos o más interfaces refractoras, de las cuales al menos una está curvada. Las lentes que están constituidas por sólo dos superficies refractoras (un elemento) son lentes simples. La presencia de más superficies refractoras las convierte en lentes compuestas. Conjuntamente, dependiendo de si el espesor es efectivamente despreciable o no, una lente se puede clasificar como delgada o gruesa (Hecht, 2016). Para lentes delgadas con forma esferoidal, se puede demostrar que la longitud focal se obtiene de la fórmula del constructor de lentes, dada por:

$$\frac{1}{f_\lambda} = (n_\lambda - 1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right), \tag{2.5}$$

donde R_1 y R_2 son los radios de curvatura de cada superficie. Para una superficie convexa, la longitud focal será positiva y para una cóncava será negativa.

Para un sistema compuesto por dos elementos, la distancia desde el vértice de la última superficie del sistema hasta el segundo punto focal se conoce como distancia focal posterior o d.f.p. Matemáticamente se define como:

$$\text{d.f.p} = \frac{f_2(d - f_1)}{d - (f_1 + f_2)}, \tag{2.6}$$

donde d es la distancia de separación entre las lentes. De igual modo, la distancia del vértice de la primera lente hasta el primer punto focal se le conoce como d.f.f o distancia focal frontal. De esta manera, matemáticamente es:

$$\text{d.f.f} = \frac{f_1(d - f_2)}{d - (f_1 + f_2)}. \tag{2.7}$$

De estas dos últimas ecuaciones, nos podemos percatar de que si $d \to 0$ entonces d.f.f=d.f.p= $f_2 f_1 / f_2 + f_1$, de tal forma que, obtenemos una distancia focal efectiva dada por:

$$\frac{1}{f} = \frac{1}{f_1} + \frac{1}{f_2}. \tag{2.8}$$

Consecuentemente, si consideramos un sistema de N lentes, obtendremos:

$$\frac{1}{f} = \frac{1}{f_1} + \frac{1}{f_2} + \ldots + \frac{1}{f_N}, \tag{2.9}$$

siempre y cuando $d_i \to 0$.

2.1.2. Superficies reflectivas

Una superficie sobre la cual incide un haz de luz y ocurre el fenómeno de reflexión se conoce como espejo. Existen espejos planos, esféricos y asféricos[1]. Esta última clase de espejos son considerablemente más difíciles de fabricar que los espejos planos y esféricos. La distancia focal se calcula a partir de la ecuación para la sección transversal circular de una esfera, dada por:

$$y^2 + (x - R_c)^2 = R_c^2, \tag{2.10}$$

donde x y y son coordenadas sobre la sección transversal circular de la esfera y el centro está desplazado del origen por un radio R_c, conocido como radio de curvatura. Despejando x de la Ecuación 2.10 se considera únicamente un hemisferio de la sección transversal circular correspondiente al signo negativo (el hemisferio abierto hacia la derecha) y utilizando la fórmula de Newton (Bronshtein et al., 1973), adquiere la forma:

$$x = \frac{y^2}{2R_c} + \frac{y^4}{2^2 2! R_c^3} + \frac{3y^6}{2^3 3! R_c^5} + \dots \tag{2.11}$$

Comparando la Ecuación 2.11 con la ecuación canónica de la parábola ($y^2 = 4fx$), se obtiene que:

$$f = R_c/2, \tag{2.12}$$

es decir, a primera aproximación la serie se puede considerar como parabólica mientras que los términos restantes figuran como la desviación. Por lo tanto:

$$\Delta x = \frac{y^4}{2^2 2! R_c^3} + \frac{3y^6}{2^3 3! R_c^5} + \dots \tag{2.13}$$

La desviación podrá observarse cuando $y \gg R_c$, es decir, cuando y sea lo suficien-

[1]Espejos asféricos: Consitituidos por superficies ópticas que involucran el parámetro e, conocido como excentricidad de una sección cónica (ver Sección 2.1.2) (Hecht, 2016). Además, se tienen superficies ópticas constituidas por superficies con el parametro de excentricidad más términos de asfericidad (ver Sección 2.1.2).

temente grande en comparación con R_c.

La Ecuación 2.12 se conoce como la ecuación de los espejos y es aplicable tanto para espejos convergentes como para espejos divergentes, donde la longitud focal será positiva o negativa.

Superficies cónicas

Una extensa gama de superficies ópticas pueden ser elaboradas con diferentes formas geométricas, comúnmente son utilizadas las superficies esféricas por la simplicidad de elaboración. Sin embargo, si deseamos corregir las aberraciones ópticas que se presentan, es necesario que se utilicen superficies diferentes a las mencionadas, por ejemplo, las superficies cónicas. La obtención de la ecuación generalizada de las superficies cónicas mostradas en la Figura 2.1 (Bronshtein et al., 1973), es sencillamente derivable a partir de las ecuaciones canónicas de las mismas (hipérbola, elipse, etc.):

$$z = \frac{c_p s^2}{1 + \sqrt{1 - (1+\kappa) c_p^2 s^2}}. \tag{2.14}$$

Los parámetros presentes en la Ecuación 2.14 son: la curvatura paraxial c_p que se define como el inverso del radio de curvatura $(1/R_c)$, la distancia del eje óptico a un punto sobre la superficie s y la constante de conicidad κ que a su vez está relacionada con la excentricidad $(\kappa = -e^2)$. El parámetro que determina el tipo de superficie es la constante de conicidad, de acuerdo con la Tabla 2.1, que muestra los valores para cada superficie.

Superficies asféricas

Existen sistemas ópticos compuestos de superficies cónicas más términos de asfericidad. Los términos mencionados son términos correctores a la superficie y son usados

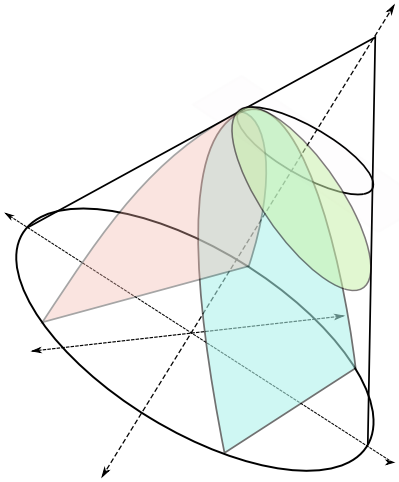


Figura 2.1: Secciones cónicas. Se puede a apreciar un círculo (color blanco), una elipse (verde pastel), hipérbola (azul pastel) y parábola (rosa pastel). Todas estas Secciones se pueden obtener dependiendo del valor que tome κ como se muestra en la Tabla 2.1.

Tabla 2.1: Valores posibles de κ para la Ecuación 2.14.

Superficie	Constante de conicidad
Elipse Oblata	$\kappa > 0$
Círculo	$\kappa = 0$
Elipse prolata	$-1 < \kappa < 0$
Parábola	$\kappa = -1$
Hipérbola	$\kappa < -1$

en lentes asféricas correctoras, cuya ecuación de la sagita esta definida como:

$$z_1 = \frac{c_p s^2}{1 + \sqrt{1 - (1+k)c_p^2 s^2}} + \sum_{j=1}^{n} \alpha_j s^{2j}. \tag{2.15}$$

2.2. Formación de imágenes

Como se mencionó anteriormente, el plano focal es el plano que pasa a través del punto focal y está orientado de tal manera que es perpendicular al eje óptico del sistema. Para poder registrar la imagen de cualquier objeto astronómico, se utilizan sistemas ópticos (telescopios) que hacen converger la luz del objeto en un detector ubicado en

el plano focal del telescopio. Por ejemplo, un objeto ubicado en el eje óptico formará una imagen en el centro del plano focal (ver Figura 2.2). En virtud de que la distancia de un objeto astronómico es lo suficientemente grande comparado con las distancias focales de los telescopios, cualquier objeto astronómico se asume que está localizado en el infinito con respecto al telescopio; subsecuentemente, todos los rayos provenientes del objeto serán paralelos uno del otro (haz colimado). Es decir, vienen desde un campo con respecto al eje óptico, esto se verá mas adelante.

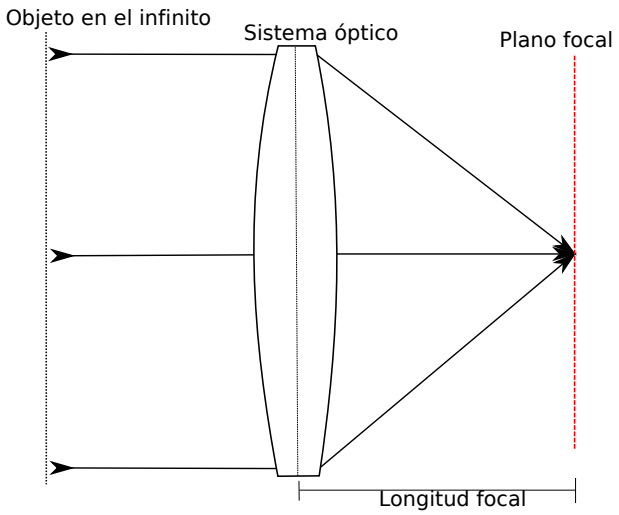


Figura 2.2: Objeto astronómico ubicado en el eje óptico y formando una imagen en el plano focal.

Por otro lado, un objeto ubicado un cierto ángulo α con respecto al eje óptico formará una imagen a una distancia s del centro del plano focal (ver Figura 2.3). Por geometría encontramos que la distancia s, esta dada por:

$$s = f \tan \alpha, \tag{2.16}$$

donde f es la longitud focal. Cuando el campo de visión del telescopio es mínimo (cercano al eje óptico), se puede asumir que el ángulo α es muy pequeño[2]. Por lo tanto, se puede utilizar la aproximación para ángulos pequeños ($\tan \alpha \approx \alpha$); con lo cual se

[2]Esta suposición o aproximación de ángulos pequeños se le denomina aproximación gaussiana o aproximación paraxial (Mahajan, 2014b).

puede reescribir la Ecuación 2.16, como:

$$s = f\alpha. \tag{2.17}$$

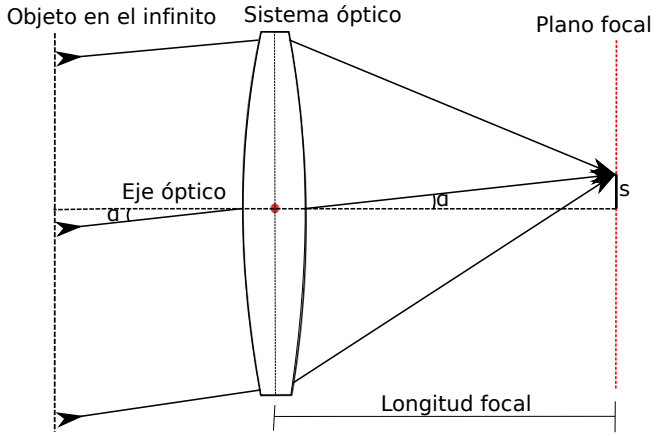


Figura 2.3: Objeto astronómico ubicado fuera del eje óptico y formando una imagen a una distancia s fuera del eje. El ángulo del haz con respecto al eje óptico se denomina como α.

Equivalentemente, dos objetos astronómicos que se encuentran separados por un ángulo α en el cielo, formarán imágenes separadas por una distancia s en el plano focal. La distancia de separación estará dada por la Ecuación 2.16 y si el campo de visión del telescopio es cercano al eje óptico, se puede utilizar la Ecuación 2.17. Asimismo, un objeto extendido que subtiende un ángulo α en el cielo formará en el plano focal una imagen con extensión dada por las ecuaciones anteriormente mencionadas.

La Ecuación 2.17 se puede reescribir como:

$$\frac{\alpha}{s} = f^{-1}\text{rad/m} = EP. \tag{2.18}$$

El resultado de la ecuación anterior se conoce como escala de placa (EP), la cual caracteriza el tamaño de la imagen en el plano focal. Sin embargo, las unidades más

utilizadas son segundos de arco/mm (″/mm). Por lo tanto, realizando la conversión, se tiene que la Ecuación 2.18 se reescribe como:

$$EP = \frac{206265}{f}\frac{''}{\text{mm}}. \tag{2.19}$$

Para entender cual es la función de la escala de placa, supongamos que se empleará el telescopio de 2.1 m perteneciente al OAN-SPM para realizar una noche de observaciones, dicho telescopio tiene una distancia focal efectiva de 15824 mm. De esta manera, si empleamos la Ecuación 2.19 se tiene que la escala de placa es de 13.034″/mm. Este resultado nos dice que cada milímetro del detector cubrirá 13.034 segundos de arco en el cielo. Es decir, si nuestro objetivo es observar la Luna (1800″ de tamaño angular) con dicho telescopio, sería necesario tener un detector en el plano imagen de al menos 138.036 mm.

Por lo tanto, vemos que un incremento en la longitud focal conllevará a que las imágenes aumenten de tamaño, como función del incremento en el ángulo α. Para fuentes puntuales, la sensibilidad del sistema óptico depende únicamente de la apertura de la superficie óptica y de la magnitud del objeto astronómico. Sin embargo, para fuentes extendidas se tiene también una dependencia en la magnitud superficial del objeto astrónomico, por lo que la escala de placa es fundamental en la sensibilidad del sistema, debido a que el área cubierta por dicho objeto depende de la EP, siendo mayor cuando la escala es pequeña. Es decir, un objeto extendido circular de diámetro s, tendrá un área proporcional a s^2. Por lo tanto, si el objeto extendido subtiende un ángulo α en el cielo, la tasa de energía que llegará al plano focal para un telescopio con una apertura fija está dada por: $EP \propto s^2 \propto f^{-2}$, donde $s \propto f^{-1}$ por la escala de placa. Además, una apertura D mayor permitirá que se recolecten más fotones proporcionales al área de recolección a del espejo, donde $a \propto D^2$. Así se puede definir la eficiencia del

sistema, dada por:

$$E \propto (D/f)^2. \tag{2.20}$$

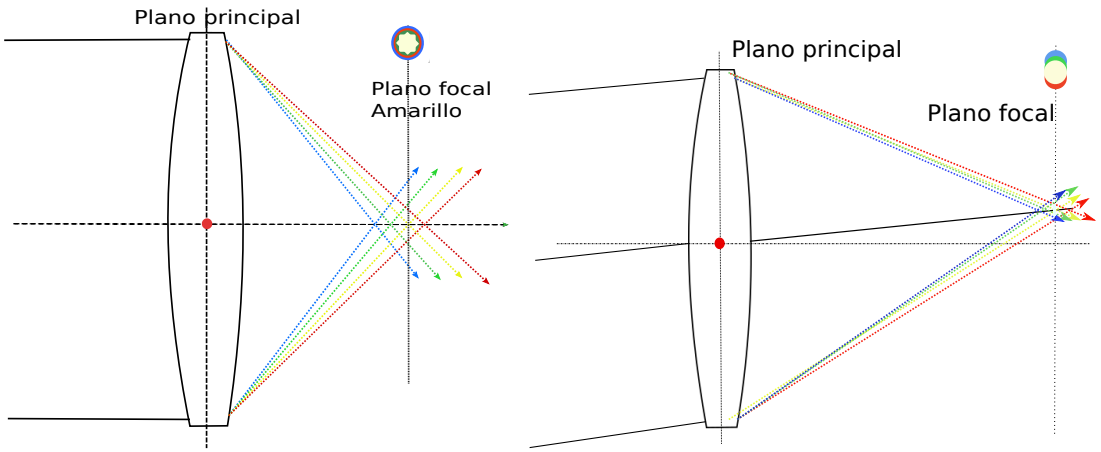
Por otro lado, al término $x = f/D$ se le denomina *Razón focal* o *número f* de un telescopio y se identifica con la notación $f/\#$. El cociente focal es una medida inversa de la rapidez con la que se deposita la energía en un elemento del plano focal (Bradt, 2004).

2.3. Aberraciones

La distorsión en imágenes producidas por elementos ópticos se conoce como aberración e involucra tanto a espejos como a lentes (Carroll and Ostlie, 2017). Existe una aberración única de los telescopios refractores, conocida como *aberración cromática*. Esto ocurre debido a que la longitud focal de las lentes (ver Ecuación 2.5) dependerá de la longitud de onda que se considere. Esta aberración se presenta de dos formas diferentes: la aberración cromática longitudinal (ver Figura 2.4a) es un error en el enfoque, es decir, se produce cuando diferentes longitudes de onda de la luz procedente de un objeto puntual se refractan en planos focales separados a diferentes distancias focales a lo largo del eje óptico (Kidger, 2001). La aberración cromática lateral (ver Figura 2.4b) es un error de aumento producido porque diferentes longitudes de onda de la luz procedente de un objeto puntual fuera de eje se dispersan a diferentes potencias[3]. (Kidger, 2001).

Otra clase de aberración surge de construir lentes o espejos con forma esferoidal, en virtud de que se tendrá un mal enfoque causado por aquellos rayos que son para-

[3]Se le denomina potencia al inverso de la distancia focal efectiva de la superficie óptica. La potencia es una medida de la capacidad de la superficie óptica para convertir un haz paralelo en un haz convergente; cuanto menor sea la distancia a la que se enfoca el haz, mayor será la potencia de la superficie óptica (Mahajan, 2014a)



(a) Longitudinal (b) Lateral

Figura 2.4: a) Aberración cromática longitudinal o axial enfocando diferentes longitudes de onda a distancias diferentes a lo largo del eje óptico y b) Aberración cromática lateral aparece en las zonas del plano imagen fuera del eje. Los colores en ambas imágenes representan los colores de la luz a diferentes longitudes de onda.

lelos al eje óptico. Que no se pueda enfocar nítidamente un objeto se debe a que se tiene diferente longitud focal dependiendo de la distancia z del rayo con respecto al eje óptico (Smith, 2000). Esta clase de aberración se conoce como *aberración esférica* y es uniforme en todo el plano imagen.

En la aberración esférica encontramos dos casos, conocidos como: *aberración esférica negativa* y *aberración esférica positiva*. La aberración esférica negativa ocurre cuando un rayo incidente a mayor distancia con respecto al eje óptico tiene una longitud focal menor que aquellos rayos de menor distancia al eje óptico o rayos centrales, esta clase de aberración es característica de espejos esféricos o lentes simples convergentes (ver Figura 2.5 (izquierda)) (Hecht, 2016). Por otro lado, la aberración esférica positiva sucede cuando los rayos incidentes a mayor distancia con respecto al eje óptico tienen una longitud focal mayor que aquellos rayos centrales, esta aberración es característica de espejos esferoidales o lentes divergentes (ver Figura 2.5 (derecha)) (Hecht, 2016).

Debido a que los rayos de luz se intersectan entre sí antes y después del punto focal,

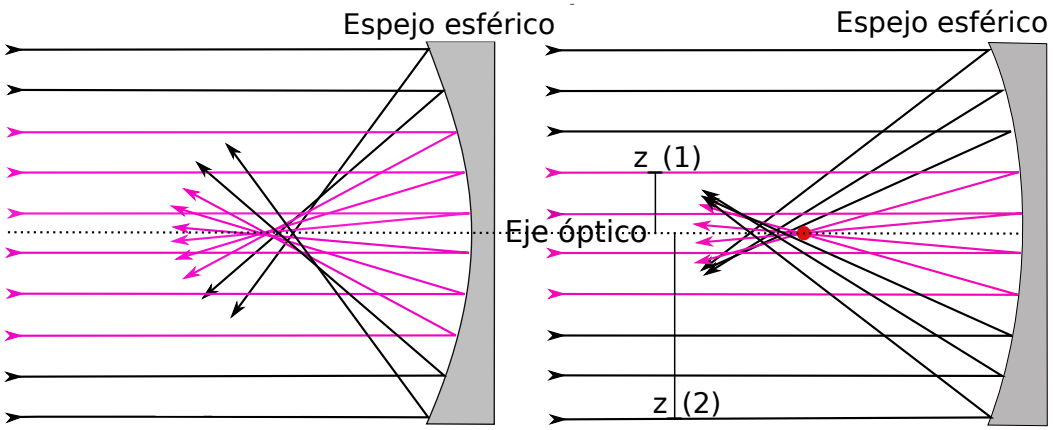


Figura 2.5: (Izquierda) Aberración esférica negativa donde se aprecia que los rayos a una separación z del eje óptico (línea punteada) tienen una longitud menor que los rayos centrales o con menor separación con el eje óptico (líneas color magenta). (Derecha) Aberración esférica positiva, en la cual se aprecia que, por el contrario, los rayos centrales tendrán una longitud focal menor a aquellos rayos que se encuentren con una mayor distancia con respecto al eje óptico.

la aberración esférica producirá un desenfoque a lo largo del eje óptico. Sin embargo, se puede encontrar el mejor enfoque disponible cuando el diámetro de la intersección de estos rayos de luz es mínimo, como se aprecia en la Figura 2.6c). Por otra parte, puede existir más de un diámetro donde se considere el mejor enfoque, como se puede ver en la Figura 2.6b) y 2.6c). Para determinar cual es el mejor enfoque se compara el diámetro de la imagen con el criterio del Disco de Airy [4]. Finalmente, si contemplamos la imagen formada fuera del foco o extra–focal, veremos que el diámetro producido por los rayos aumentará conforme los rayos tengan una mayor altura con respecto al eje óptico, como se muestra en la Figura 2.6d). En cambio, en la imagen formada en la posición intra–focal o dentro del foco se tienen diámetros de los rayos, en partícular los rayos centrales o cercanos al eje óptico (ver Figura 2.6a).

Existen diferentes formas para poder corregir la aberración esférica, una de ellas será presentada en la Sección 3.1. Sin embargo, previamente se mencionó que la abe-

[4]Disco de Airy: Las ondas planas que llegan a un sistema óptico con una apertura circular no se enfocan en un *punto*, la energía se difundirá en una diminuta mancha circular, conocida como disco de Airy, el cual contiene $\approx 80\,\%$ de la energía. La mancha circular define la superposición de imágenes contiguas y, de esta manera, la resolución. Consecuentemente, no es posible tener un sistema con enfoque perfecto, dado que debemos considerar que se encontrará limitado por la difracción (Hecht, 2016).

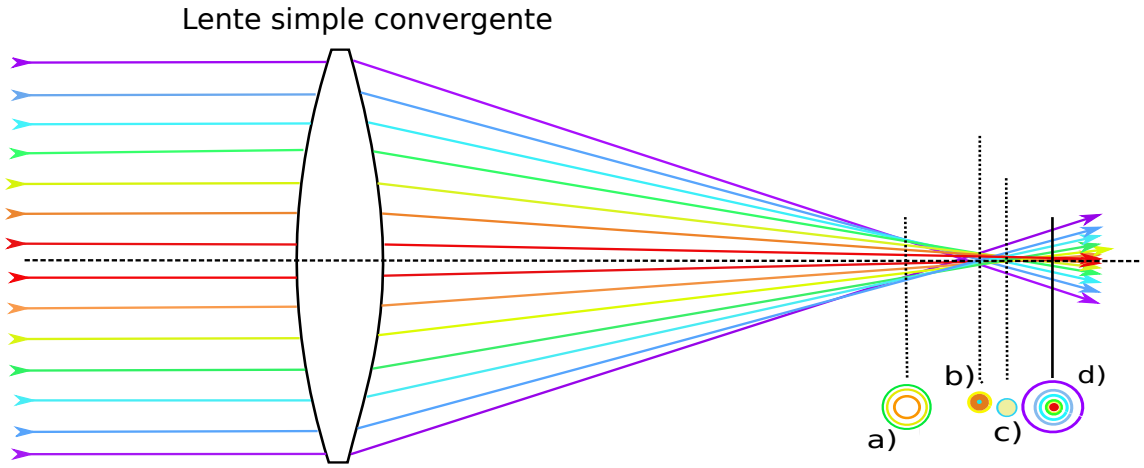


Figura 2.6: Diagrama de manchas (ver Sección 2.4) producido por la aberración esférica a diferentes posiciones. a) diagrama de manchas producido en una posición intra–focal, b) y c) diagrama de manchas producido a una distancia donde se considera que el diámetro de la mancha o imagen producido es mínimo y d) diagrama de manchas producido a una distancia extra-focal. Los colores de cada rayo representan la distancia que se tiene al eje óptico, los rayos color violeta representan rayos separados a una distancia considerablemente mayor del eje óptico en comparación con aquellos rayos de color rojo.

rración esférica esta presente cuando se emplean superficies esféricas y, por lo tanto, podemos corregir dicha aberración si escogemos otra superficie cónica, por ejemplo una parábola. No obstante, las superficies parabólicas igualmente producen una aberración denominada *aberración de coma*. Esta aberración se debe a que la longitud focal depende del ángulo con el que llega el haz con respecto al eje óptico (Carroll and Ostlie, 2017); esto produce que las imágenes se vean distorsionadas como se puede apreciar en la Figura 2.7.

La dependencia del ángulo de incidencia de los rayos con respecto al eje óptico se debe a que los rayos de luz paralelos pero fuera del eje (por ejemplo, los rayos marginales[5]) no inciden en los lados opuestos del espejo con ángulos de reflexión iguales (ver Figura 2.7a): en un lado del espejo el ángulo se hace menor que en el lado opuesto; en otras palabras, es como si un lado del espejo se hiciera más plano y el otro más curvo. Un efecto similar se produce con la refracción desequilibrada en los lados opuestos de una lente, debido a los diferentes ángulos de incidencia de los rayos paralelos, como se

[5]Rayo marginal: es el rayo que pasa desde el centro del objeto, a la apertura máxima de la superficie óptica(Kidger, 2001). Rayo principal: es el rayo que pasa por el centro de la primera superficie óptica. (Hecht, 2016)

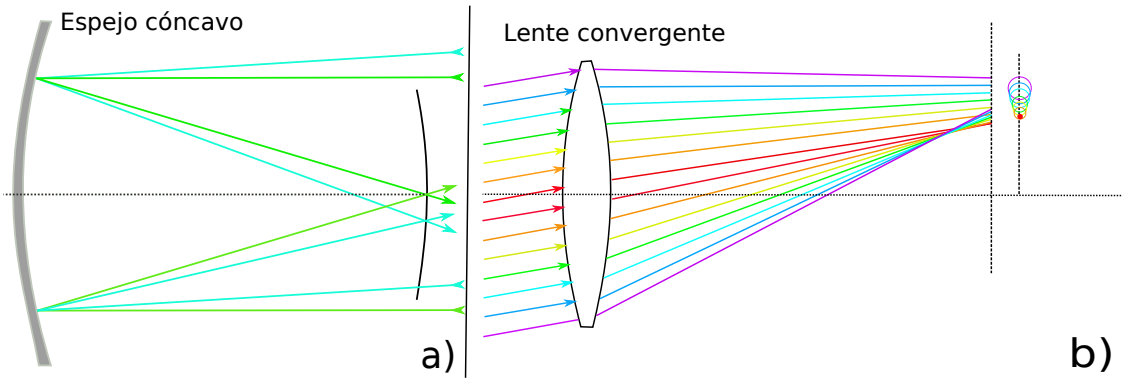


Figura 2.7: Visualización de rayos de luz paralelos pero fuera del eje incidiendo en los lados opuestos del espejo con ángulos de reflexión o refracción que no son iguales, es decir, en un lado del espejo o lente el ángulo se hace menor que en el lado opuesto. Desde punto de vista de uno de los rayos parecería que uno de los lados de la superficie óptica tendría una geometría diferente al rayo paralelo a nuestro marco de referencia; es decir, un lado del espejo o la lente se vería más plano y el otro más curvo dependiendo. La figura a) muestra dos conjuntos de rayos incidiendo en un espejo, donde los rayos azules representan rayos con un cierto ángulo con respecto al eje óptico y los rayos verdes son paralelos al eje óptico. En la figura b), los colores de cada rayo representan la distancia que se tiene al eje óptico, los rayos color magenta representan rayos separados a una distancia considerable del eje óptico en comparación con aquellos rayos color rojo.

muestra en la Figura 2.7b.

La aberración de coma también se puede definir como la variación de la amplificación con la apertura ó *coma transversal* (ver Ecuación 4.32) (Smith, 2000), debido a que un conjunto de rayos que inciden inclinados con respecto al eje óptico sobre una superficie óptica que no se encuentra corregida por coma, atravesarán porciones de los bordes (rayos marginales) de la superficie óptica a una altura diferente que los que pasan por la porción central (rayo principal).

Existen dos formas para que se presente la aberración de coma. La primera forma, como ya se mencionó, es intrínseca al diseño, en un telescopio perfectamente alineado la coma es nula o en su defecto se presenta con una simetría radial y su magnitud aumenta conforme nos alejamos del eje óptico. La segunda forma se presenta si alguno de los dos elementos ópticos se encuentra desalineado, cuando es producida por la desalineación, esta se presenta en una dirección de manera uniforme sobre todo el campo y se conoce como *coma axial* (McLeod, 1996), como se muestra en la Figura 2.12. En

el primer caso se puede corregir modificando nuestro diseño como se mostrará en la Sección 3.1.4, en el segundo caso, la coma se elimina cuando el sistema óptico está colimado.

Un método para encontrar la coaxialidad entre los espejos de un telescopio es expuesto por McLeod (1996) quien describe el procedimiento en el cual primero se elimina la coma por desplazamiento lateral del espejo secundario para posteriormente inclinar el mismo pivotado en un punto en el eje óptico del espejo primario en una forma tal que la aberración de coma no vuelva a ser inducida. Dicho punto se conoce como "punto de cero coma" o "punto neutro" (Schroeder, 1999). Existe una solución analítica para determinar el punto de cero coma de acuerdo a Wetherell and Rimmer (1972). Sin embargo, la solución analítica solamente determina dicho punto para telescopios clásicos, es decir, telescopios que no contienen lentes correctoras o superficies cónicas con polinomios de asfericidad, como se demostrará en la Sección 5.1.

Por otra parte, se puede presentar astigmatismo, que se originará por tener diferentes partes de la lente o espejo convergiendo en posiciones ligeramente diferentes al plano focal. Dicho de otra forma, el astigmatismo ocurre cuando los rayos que son perpendiculares a la sección transversal del plano imagen no tienen la misma distancia focal a lo largo del eje óptico. Por lo tanto, tendremos un error tanto en el enfoque como en la amplificación del objeto.

El astigmatismo produce dos puntos focales por cada punto en el plano imagen (Smith, 2000). Estos dos puntos focales son descritos en términos de dos planos relacionados con la posición y la altura del campo del objeto fuera de eje. El primer plano contiene tanto al rayo principal del objeto fuera de eje como al eje óptico, de esta manera, el plano divide el sistema óptico por la mitad, como se muestra en la Figura 2.8a;

es decir, es una vista lateral del sistema óptico y es conocido como plano meridional (plano yz). Todo el conjunto de rayos que están dentro de este plano conforman el plano tangencial de la Figura 2.8a. El segundo plano es perpendicular al plano tangencial, como se muestra en la Figura 2.8b. Se puede decir que es una visualización desde la parte superior del sistema (plano xz) y todo conjunto de rayos dentro de este plano conforma el plano sagital. El foco definido en el plano tangencial se conoce como foco tangencial y el definido en el plano sagital se conoce como foco sagital. Como resultado veremos que los rayos dentro del plano tangencial tendrán distancias focales más cortas en comparación con los rayos contenidos en el plano sagital (Hecht, 2016).

El foco más adecuado se localiza aproximadamente entre los puntos focales, y produce una imagen cuasi-circular y desenfocada de la fuente, esta imagen tiene la contribución de la aberración tangencial y sagital en la misma proporción. Por lo tanto, se puede eliminar la aberración de astigmatismo cuando las imagenes de ambos planos coinciden (Smith, 2000).

Si bien, el astigmatismo se puede corregir al igual que las otras aberraciones, la corrección causará, de la misma forma que en los casos anteriores, otra aberración que se conoce como *Distorsión del campo* o *curvatura de Petzval*, debido a que la escala depende de la distancia al eje óptico (Carroll and Ostlie, 2017).

2.4. Diagrama de manchas

Los diagramas de manchas son utilizados convencionalmente para evaluar la calidad de imagen de un diseño óptico (ver Figura 2.9). Los diagramas se obtienen a partir de la intersección de un rayo con el plano imagen seleccionado. Para obtener una mejor representación de la geometría de la imagen resulta necesario realizar el trazo de una

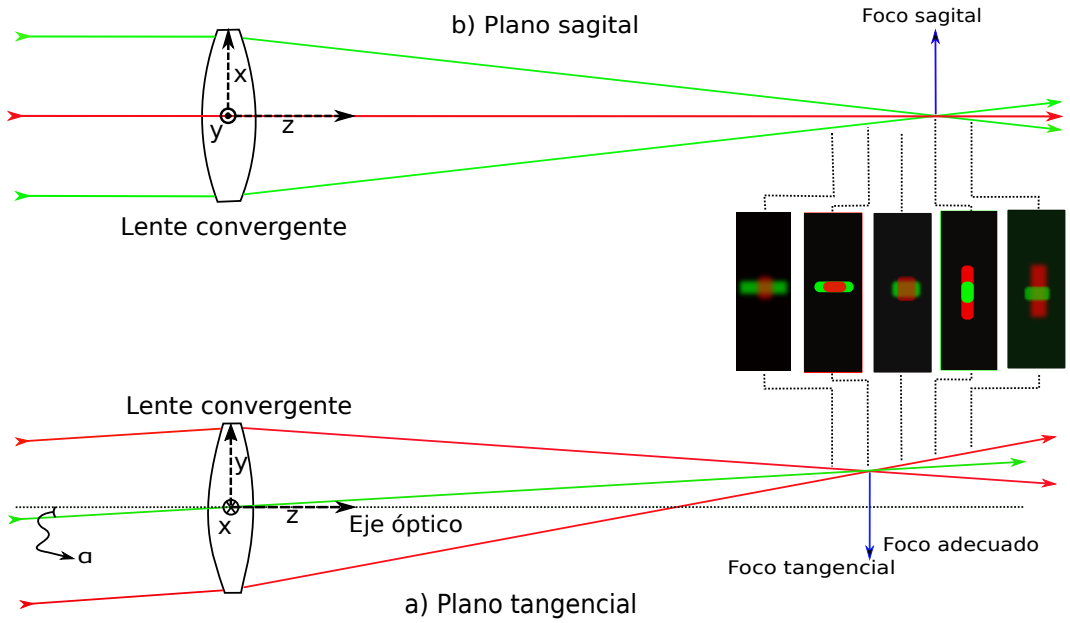


Figura 2.8: Representación del plano tangencial (Vista lateral del sistema óptico) a) y del plano sagital (Vista superior del sistema óptico) b). El plano sagital es simétrico alrededor del eje óptico, sin embargo, el plano tangencial es simétrico alrededor del rayo principal. El astigmatismo producirá dos puntos focales, uno debido a los rayos tangenciales y otro debido a los rayos sagitales, asimismo, podemos encontrar el foco más adecuado ubicado aproximadamente entre estos dos puntos. El ángulo que se forma con respecto al eje óptico se representa con la letra α en el plano tangencial. En esta representación, los rayos del plano sagital son mostrados en color verde, por el contrario, los rayos del plano tangencial son de color rojo.

cantidad significativa de rayos. De este modo, los diagramas de manchas muestran la distribución de una colección finita de rayos provenientes de un objeto puntual gracias a la intersección sobre el plano focal efectivo del sistema y su extensión se conoce como "*tamaño de la mancha*"(Mahajan, 2014b).

El tamaño de la mancha permite realizar una comparación entre los diseños ópticos, en virtud del tamaño relativo del núcleo (core) del diagrama de manchas. El diagrama de manchas está constituido por dos regiones, la primera se conforma por la región donde la densidad de puntos es alta, conocida como "*core*" y, por otro lado, la región de menor densidad que se denomina "*flare*" (Stavroudis and Sutton, 1965).

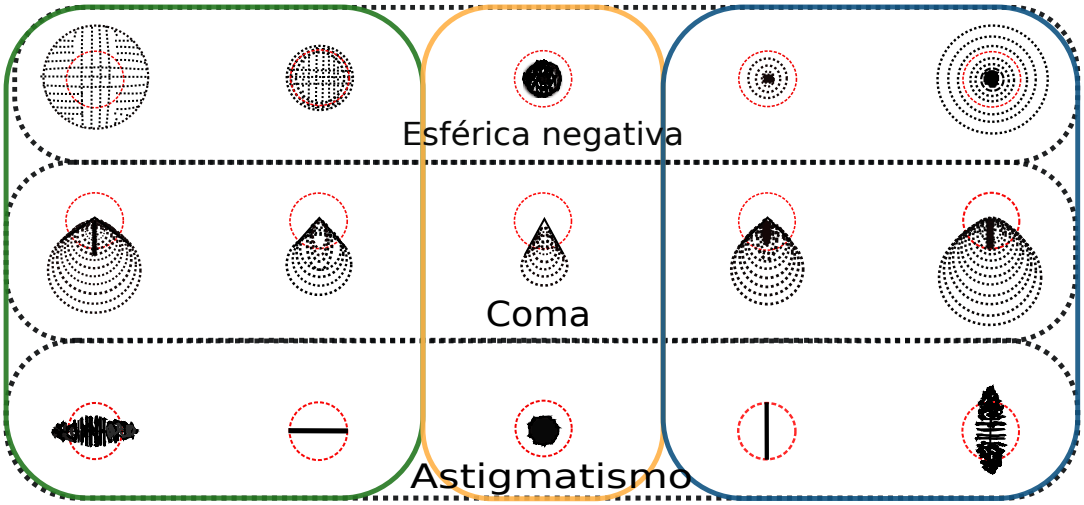


Figura 2.9: Diagrama de manchas para diferentes diseños ópticos, donde se muestran las tres aberraciones que fueron expuestas en la Sección 2.3. Los diagramas de manchas dentro del cuadro naranja muestran la mejor distancia focal donde se utiliza el criterio del disco de Airy. A la izquierda (cuadro verde) se muestran los diagramas de manchas para las tres correspondientes aberraciones, posicionado a una distancia intra-focal. Finalmente, los tres diagramas de manchas ubicados en la parte derecha (cuadro azul) son considerados a una distancia extra-focal. El circulo rojo representa el criterio del disco de Airy.

2.5. Telescopios ópticos

Las limitaciones que se les presentaron a quienes primero ejercieron la Astronomía para captar la luz de los objetos más débiles, menos brillantes o más profundos en el espacio, conllevó al diseño de los telescopios. De esta manera, como menciona Karttunen et al. (2007); los principales objetivos de estos instrumentos son:

- Recolectar la máxima cantidad de fotones provenientes de una fuente extendida distante.
- Mejorar la resolución y aumentar el diámetro angular de los objetos.
- Determinar la posición de objetos astronómicos con respecto a una referencia.

La luz proveniente de las fuentes astronómicas será colectada por los telescopios, que lo harán mediante el uso de lentes o espejos. Es por ello que tendremos dos tipos de diseños: los telescopios con lente (refractores) y telescopios con espejos (reflectores).

Estos tipos de telescopios cuentan con dos elementos ópticos, al primer elemento se le suele conocer como objetivo para el caso de telescopios refractores y para el caso de los telescopios reflectores el objetivo puede estar formado por un espejo primario (como el telescopio Herschel) y algunas veces incluye un espejo secundario (Telescopio Newtoniano, Telescopio Cassegrain, entre otros). El diámetro del espejo primario se denota como D y se le denomina "*apertura*". La razón de la apertura (D) con la distancia focal (f), se conoce como razón de apertura, dada por:

$$F = \frac{D}{f}. \tag{2.21}$$

La razón de apertura caracteriza la capacidad de recolectar luz de un telescopio. Si la razón de apertura es grande tendremos un telescopio que permite realizar observaciones con poco tiempo de exposición, en el caso de un objeto lo suficientemente brillante. Por el contrario, un telescopio tendrá una razón de apertura pequeña si la longitud focal es considerablemente mayor a la apertura, lo que significa que tardaremos más en colectar la misma cantidad de luz dada una magnitud constante. En astronomía, la razón de apertura se suele denotar como:

$$\frac{f}{n}, \tag{2.22}$$

donde n es la longitud focal dividida entre la apertura. Los telescopios que permiten recolectar fotones en un menor tiempo de observación se suelen llamar "*rápidos*", la razón de apertura de estos telescopios puede ser: $f/1$ ó $f/3$. Por el contrario, aquellos telescopios que utilizan una mayor cantidad de tiempo para recolectar la misma cantidad de fotones que en el caso anterior, tienen una razón de apertura como: $f/8$ ó $f/15$.

La apertura del sistema óptico limita el ángulo del cono de los rayos procedentes de una determinada área del cielo que pasa a través del sistema. Además, existe otro

limitante denominado *field stop*, que limita el ángulo del cono de los rayos principales del área del cielo. La imagen del *field stop* de la imagen de los elementos ópticos que lo preceden se denomina ventana de entrada (EnW), y su imagen por los elementos ópticos que le proceden se denomina ventana de salida (ExW) (Mahajan, 2014a). El ángulo θ_0 subtendido por la ventana de entrada con respecto al eje óptico, define el campo de visión angular del sistema en el espacio del objeto (Mahajan, 2014a). Asimismo, el ángulo θ_i subtendido por la ventana de salida con respecto al eje óptico es el campo de visión angular del del sistema en el espacio de la imagen (Mahajan, 2014a). Aquellos telescopios con un ángulo $\theta_0 > 1^\circ$ son conocidos como telescopios de campo amplio (*wide-field telescopes*). Por el contrario, aquellos telescopios con $\theta_0 < 1^\circ$ son telescopios de campo pequeño (*small-field telescopes*).

2.5.1. Telescopios refractores

En esta clase de telescopios, el objetivo puede estar compuesto por una o más lentes (doblete, triplete, etc...). El objetivo tiene la labor de colectar toda la luz posible con la mejor resolución, para posteriormente ser enfocada en el plano focal del objetivo. La imagen puede ser vista desde un ocular, el cual sirve como una lupa, o se puede colocar un detector para registrar la imagen. El objetivo y el ocular se ubican en los extremos opuestos del tubo del telescopio. La lente objetivo tendrá una longitud focal f_{obj}, mientras que la segunda lente (ocular) será f_{eye} (longitud focal del ocular); consecuentemente, el aumento angular, m, se puede obtener mediante:

$$m = \frac{f_{obj}}{f_{eye}}. \tag{2.23}$$

Sin embargo, el flujo de energía llevada por todos los rayos decrece con el cuadrado de la longitud focal de la lente objetivo. Así, para compensar esta disminución se necesita un incremento en el diámetro de la lente; pero, existen limitaciones en cuanto al

tamaño de la misma, debido a que la lente sólo debe ser sostenida de sus bordes para que la luz pueda ser recolectada sin obstrucciones. Si existe un incremento en el tamaño y la masa de la lente, se presentarán deformaciones a causa de la gravedad. Además, ambas lentes deben ser ópticamente perfectas. Cualquier defecto causará aberraciones en la observación.

Adicionalmente, esta clase de telescopios tienen una respuesta térmica lenta. Siempre se debe ajustar la temperatura de los telescopios cuando se abre el domo. A causa de esto, se tendrán corrientes térmicas de aire alrededor del telescopio que afectarán la obtención de imágenes. La expansión térmica entonces puede producir deformaciones en el telescopio (Carroll and Ostlie, 2017).

Finalmente, también se pueden presentar problemas mecánicos a consecuencia del gran brazo de palanca. Esto causa que la torca necesaria para compensar el tamaño del telescopio sea considerablemente grande (Carroll and Ostlie, 2017).

2.5.2. Telescopios reflectores

Hoy en día este tipo de telescopios son los más utilizados en la astronomía profesional. Remplazan el objetivo de lentes por un objetivo de espejos, de tal forma que la luz ya no atraviesa el elemento óptico. Esto reduce muchos de los problemas presentados con el anterior telescopio, por ejemplo, solo se necesita que la superficie reflectora sea fabricada con gran precisión. Como el espejo está sujetado por la parte trasera, se puede colocar un sistema activo de actuadores neumaticos de presión que ayuden a disminuir la deformación en la forma del espejo por efectos gravitacionales o térmicos, como el sistema de óptica activa que fue incorporado en el telescopio de 2.1 m del OAN-SPM (Ruiz et al., 2014) como se puede ver en la Figura 2.11.

Los primeros telescopios empleados dadas las insuficientes técnicas de fabricación y pulido de la época, consideraban únicamente espejos con forma esférica (Hecht, 2016; Cottrell, 2016; Wall, 2018). A pesar de ello, con el pasar del tiempo las técnicas de fabricación se volvieron cada vez más precisas, permitiendo la consideración de otro tipo de superficies, no solamente esféricas, por ejemplo, las superficies cónicas o superficies asféricas como las descritas en la Sección 2.1.2.

Los espejos primarios reflejan la luz que proviene de un objeto astronómico hacia un punto focal conocido como foco principal, como se muestra en la Figura 2.10a). Para evitar poner un detector o una observadora que obstruyan la cantidad de luz enfocada, Isaac Newton encontró la solución al colocar un pequeño y delgado espejo plano (inclinado a 45 grados) ubicado antes del plano focal que reflejara el haz, cambiando la localización del punto focal a una zona perpendicular fuera del telescopio (ver Figura 2.10b), no obstante, el telescopio Newtoniano sufre de inconvenientes con el detector. Si es ubicado en una distancia lejana al centro de masa y además es bastante grande, puede ejercer una torca sobre telescopio (Karttunen et al., 2007).

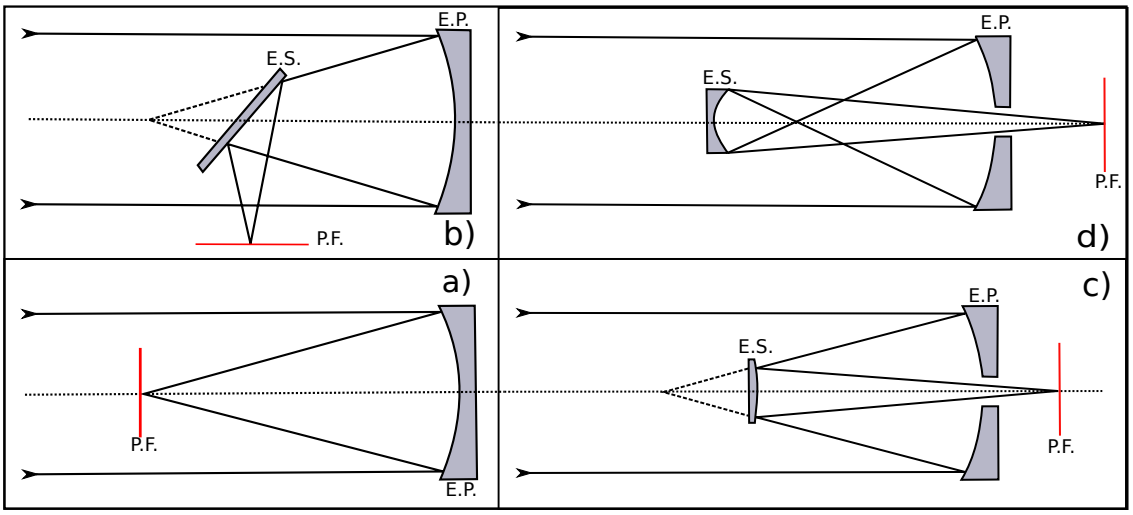


Figura 2.10: Conjunto de diferentes configuraciones de telescopios reflectores. La configuración a) muestra un telescopio con un espejo objetivo, la configuración b) muestra la configuración diseñada por Isaac Newton o mejor conocida como Newtoniano. Por otra parte, en c) tenemos la configuración de un telescopio Cassegrain, el cual tiene perforado el centro del espejo primario y, finalmente, la configuración d) es la de un telescopio Gregoriano. La línea roja representa el foco de los diferentes diseños. Nota: Espejo primario (E.P.), Espejo Secundario (E.S.) y Plano Imagen (P.I.).

Otra posibilidad de diseño para telescopios reflectores es la de perforar el centro del espejo primario y reflejar los rayos con un segundo espejo a través del agujero. En 1663, James Gregory presentaría el diseño de un telescopio compuesto por un espejo primario parabólico con un agujero en el centro primario y un espejo secundario elíptico, como se muestra en la Figura 2.10d). Otro ejemplo de este diseño lo presenta el telescopio de 2.1 m del OAN-SPM, el cual tiene un agujero en el espejo primario (ver Figura 2.11). Este tipo de diseño es llamado Cassegrain y permite colocar detectores pesados en el foco (foco Cassegrain), que se encuentra cerca del centro de masa como se visualiza en la Figura 2.10c). El primer espejo es cóncavo, y el segundo espejo tiene una forma convexa, que incrementa la longitud focal del sistema. Se conoce como Ritchey-Chrétien, a aquel diseño que utiliza espejos hiperbólicos en el espejo primario y secundario (Carroll and Ostlie, 2017).

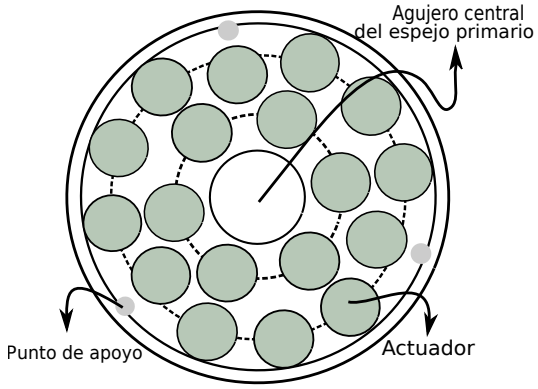


Figura 2.11: Visualización de la celda del espejo primario del Telescopio de 2.1 m ubicado en el OAN-SPM (Ruiz et al., 2014). Esta visualización permite conocer la ubicación de los tres puntos de apoyo y los 18 actuadores (bolsas de aire) del sistema de óptica activa.

Un arreglo más complicado de espejos puede guiar la luz a través del eje de declinación del telescopio al foco coudé fijo, situado en otra habitación (*coude room*). Esto es útil cuando los instrumentos a utilizar son bastante pesados o requieren alto grado de estabilización (e.g. HAPRS; PHASE (2003)). Esta clase de telescopios se utilizan principalmente en espectroscopia (Karttunen et al., 2007).

2.6. Planteamiento de problema

La astronomía observacional y los estrictos requerimientos de los problemas científicos empujan los límites de la calidad de imagen que los telescopios deben proporcionar. Por ejemplo, la coma producida por la rotación o el desplazamiento de uno de los elementos ópticos, reduce la calidad de imagen como se muestra en la Figura 2.12.

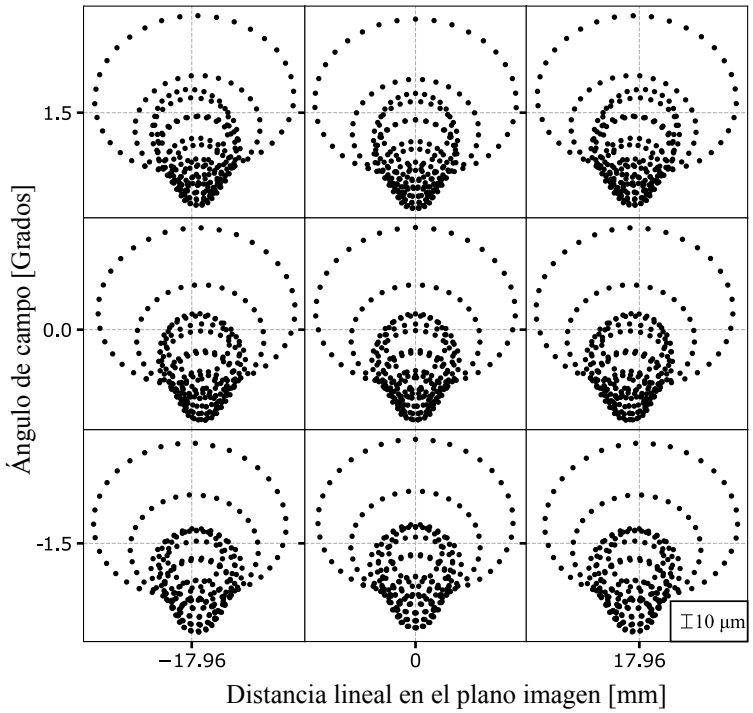


Figura 2.12: Diagrama de manchas que muestra la aberración de coma debida a 1 mm de desplazamiento lateral en el espejo secundario de un telescopio de campo amplio teórico, con $f/3$, para nueve ángulos diferentes con respecto al eje óptico.

Adicionalmente, un aumento en los requerimientos de calidad de imagen viene acompañado con un aumento en la complejidad de los sistemas ópticos y sus variables intrínsecas. De esta manera, la falta de una metodología para la colimación de los nuevos telescopios como los del proyecto TAOS-II en el OAN-SPM, que si bien se trata de un diseño ya utilizado en otros sitios, no existe en la literatura información sobre el "*punto de cero coma*" de este tipo de telescopios, tan importante en una colimación determinista.

Este tipo de telescopios de campo amplio cuentan con superficies asféricas, por lo tanto en este proyecto de tesis buscamos desarrollar un algoritmo que trabaje operando con rayos definidos como vectores, calculando la intersección de estos con las superficies ópticas, las normales de estas en los puntos de intersección y su rayo reflejado en el caso de los espejos, o refractado en el caso de lentes correctoras. Además, el algoritmo debe tomar en cuenta los desplazamientos e inclinaciones de las superficies, independientemente de la función matemática que las describe, esto nos permitirá realizar simulaciones de sistemas ópticos complejos. Todo esto propiciará la determinación del "*punto de cero coma*" de telescopios clásicos, también en los telescopios de campo amplio, donde la solución analítica clásica no puede determinar el punto de coma cero.

2.7. Objetivos

2.7.1. Generales

La escasa o nula información que se tiene sobre los telescopios no clásicos, la cual resulta relevante para el equipo técnico del OAN-SPM que se encuentran trabajando activamente en la integración de metrología de calidad de imagen en tiempo real mediante la detección del frente de onda para un proceso de colimación determinista, nos motiva a escribir un algoritmo como el que se describe en la Sección 2.6. Es importante mencionar que el algoritmo se desarrollará en el lenguaje de programación Python, ya que es un lenguaje que resulta accesible para todos los usuarios y, principalmente, tiene una licencia de código abierto.

Desarrollar dicho algoritmo facilitará el cálculo del punto de coma cero tanto para telescopios clásicos como para los telescopios no clásicos. De esta manera, podremos

realizar una comparación del valor numérico que se obtiene a partir del cálculo con la solución clásica que se encuentra en Wetherell and Rimmer (1972) (ver Sección 3.2) y la solución numérica calculada con el algoritmo que nosotros desarrollamos para ambos casos. Esta comparación nos permitirá discernir las principales diferencias entre ambos casos, así como proporcionar un método para obtener el punto de coma cero que resulta de gran importancia para las necesidades del OAN-SPM. En la Tabla 2.2 presentamos las características ópticas de los telescopios que analizaremos en este trabajo de tesis.

Tabla 2.2: Caracteristicas ópticas de los telescopios a analizar.

Telescopio (m)	Razón focal $f/\#$	Espejo	R_c [b] (mm)	k	Diámetro (mm)	Diseño
0.84	15	M_1	-5287.0	-1.0049	840.0	Ritchey-Chretien[c.1]
		M_2	-1555.0	-2.6990	250.0	
1.5	13	M_1	-5975.0	-1.0049	1540.0	Ritchey-Chretien[c.1]
		M_2	-1208.0	-1.8970	275.0	
2.1	7.5	M_1	-9638.0	-1.0773	2118.0	Ritchey-Chretien[c.2]
		M_2	-3930.0	-4.3281	673.0	
2.1	13.5	M_1	-9638.0	-1.0773	2118.0	Ritchey-Chretien[c.2]
		M_2	-2028.0	-2.7284	406.0	
2.1	30	M_1	-9638.0	-1.0773	2118.0	Ritchey-Chretien[c.2]
		M_2	-981.0	-2.3947	195.0	
6.5 (TSPM)	5.1	M_1	-16256.0	-1.0000	6502.4	Cassegrain[c.3]
		M_2	-5150.9	-2.6946	1714.5	
2.12 (INAOE[a])	11.9	M_1	-11340.0	-1.0274	2118.0	Ritchey-Chretien[c.4]
		M_2	-3114.1	-2.7747	330	

Note. — [a] El telescopio INAOE 2.12 m no forma parte del OAN-SPM, no obstante, forma parte de la comunidad astronómica mexicana.
[b] El signo negativo del radio de curvatura del espejo primario se debe a la ley de los signos para los radios de curvatura de las superficies, dado que cuando el centro de curvatura C está del mismo lado que la luz saliente, el radio de curvatura es positivo; en caso contrario, es negativo. Esto permite tener coherencia con el algoritmo de trazo exacto de rayos (ver Sección 4.2).
[c.1](González et al., 2018), vease también en https://www.astrossp.unam.mx/es/. [c.2](Herrera et al., 2017), vease también en https://www.astrossp.unam.mx/es/. [c.3](Uribe et al. (2016);González et al. (2018)), continua siendo un proyecto, aun no se encuentra en funcionamiento. [c.4](Carrasco et al., 2017), vease también https://www.inaoep.mx/˜astrofi/cananea/.

2.7.2. Particulares

- Utilizando la solución analítica, calcular el punto de coma cero para un telescopio clásico y, además, para un telescopio no clásico.
- Implementar un algoritmo que opere con rayos definidos como vectores, calculando la intersección de estos con las superficies, las normales de estas en estos los puntos de intersección y su rayo reflejado o refractado en el caso de lentes correctoras. Además, el algoritmo debe tomar en cuenta los desplazamientos e inclinaciones de las superficies independientemente de la función matemática que las describe.
- Implementar al algoritmo una función que permita la compensación de la aberración de coma utilizando como criterio el cambio de amplificación por coma.
- Realizar una comparación del valor numérico proporcionado por la solución analítica y la solución numérica calculada usando el trazo de rayos exacto.
- Aplicar el algoritmo para calcular el punto de coma cero para los telescopios de 0.84 m, 1.5 m, 2.1 m, 1.3 m (TAOS-II) y 1.0 m (SAINT-EX) pertenecientes al OAN-SPM, además de incluir el telescopio de 2.12 m del OAGH.

Capítulo 3

Solución Analítica

En la Sección 2.3 se discutió sobre la presencia de coma por dos procesos físicos diferentes. Es por ello que en la Sección 3.1 demostraremos como se puede corregir la coma intrínseca del diseño óptico a tercer orden[6] a partir de las sumas de Seidel (Kidger, 2001; Welford, 2017). A continuación, se deducirá la solución analítica del punto de cero coma que determina un punto pivote sobre el eje óptico del espejo primario, de tal manera que al inclinar en cualquier dirección el espejo secundario sobre este punto, la aberración de coma no vuelva a ser inducida. Finalmente, se muestra una tabla con los valores otorgados por dicha solución para los telescopios del OAN-SPM y el telescopio de 2.12 m del INAOE.

3.1. Diseño óptico a tercer orden

Los parámetros de diseño pueden ser divididos en tres categorías descritas a continuación, no obstante, esto será aplicable a diseños con dos cónicas aplanáticas (Wetherell and Rimmer, 1972). La primera categoría son los parámetros fundamentales de

[6]Se conoce como diseño óptico a tercer orden ya que se busca la corrección de aberraciones utilizando los parametros de diseño de tercer orden o constantes de conicidad (ver Sección 3.1).

diseño, los cuales representan la mínima cantidad de parámetros para especificar completamente una configuración óptica. Estos parámetros son: diámetro del haz óptico, las posiciones axiales de los espejos y la imagen en el sistema, la potencia de los dos espejos y la calidad de imagen fuera del eje. La segunda categoría la conforman los parámetros de primer orden, los cuales están conformados por: la separación del vértice del primer espejo al vértice del segundo espejo, distancia focal del sistema (valor absoluto), curvatura del vértice de los espejos, la ubicación de la pupila de salida y el diámetro de varias aberturas del diseño. Finalmente, la última categoría la conforman los parámetros de tercer orden, también conocidos como constantes de conicidad aplanáticas (Wetherell and Rimmer, 1972).

3.1.1. Sumas de Seidel

La aberración total del frente de onda puede ser representada como una suma de las contribuciones de diferentes partes del sistema (Welford, 2017). De esta manera, Welford (2017) muestra como los cinco coeficientes de una suma se pueden considerar como los coeficientes de las aberraciones de Seidel. El primer término representa la aberración esférica, el segundo término se asocia a la coma, el tercer término es el astigmatismo, el cuarto es la curvatura de campo y el último término se debe al desplazamiento del centro de la esfera de referencia en la dirección de la pupila. Los términos de las sumas de Seidel están asociados con aberraciones de las superficies individuales (Welford, 2017). Por lo tanto, determinar el valor de los coeficientes de Seidel permite calcular las aberraciones de un frente de onda de luz que atraviesa un determinado sistema óptico. Las sumas de Seidel están dadas por las siguientes ecuaciones (Kidger, 2001):

$$S_I = -\sum_{k=0}^{n} A_k^2 h_k \Delta\left(\frac{u}{n}\right)_k, \tag{3.1}$$

$$S_{II} = -\sum_{k=0}^{n} A_k \bar{A}_k h_k \Delta\left(\frac{u}{n}\right)_k, \tag{3.2}$$

$$S_{III} = -\sum_{k=0}^{n} \bar{A}_k{}^2 h_k \Delta\left(\frac{u}{n}\right)_k, \tag{3.3}$$

$$S_{IV} = -\sum_{k=0}^{n} H_k^2 c_k \Delta\left(\frac{1}{n}\right)_k, \tag{3.4}$$

$$S_V = -\sum_{k=0}^{n} \left\{ \frac{\bar{A}_k{}^3}{A_k} h_k \Delta\left(\frac{u}{n}\right)_k + \frac{\bar{A}_k}{A_k} H_k^2 c_k \Delta\left(\frac{1}{n}\right)_k \right\}. \tag{3.5}$$

El valor de H, en las ecuaciones anteriores se conoce como el invariante de Lagrange, cuyo valor es $H = \bar{h}A - h\bar{A}$. Los valores de $\bar{A}$ y A representan el invariante de refracción y translación, dados por $\bar{A} = n\bar{i}$ y $A = ni$; donde $\bar{i}$ es el ángulo de incidencia del rayo principal e i es el ángulo de incidencia del rayo marginal. Además, se tiene que la altura del rayo principal y el marginal en cada superficie está representada por $\bar{h}$ y h, respectivamente. El índice de refracción está dado por n y u, y describe el ángulo que forma el rayo con respecto al vector normal a la superficie óptica. Finalmente, los términos $\Delta(u/n)_k$ y $\Delta(1/n)_k$, se calculan de la siguiente manera:

$$\Delta\left(\frac{u}{n}\right)_k = \frac{u_{k+1}}{n_{k+1}} - \frac{u_k}{n_k}, \tag{3.6}$$

$$\Delta\left(\frac{1}{n}\right)_k = \frac{1}{n_{k+1}} - \frac{1}{n_k}. \tag{3.7}$$

La aberración del frente de onda se puede escribir en términos de los coeficientes de las sumas de Seidel (véase en Welford (2017)), pero esta dependerá del diámetro de

la pupila η, es decir:

$$W(x_p, y_p, \eta) = \frac{1}{8} S_I \frac{(x_p^2 + y_p^2)^2}{h_p^4} + \frac{1}{2} S_{II} \frac{y_p (x_p^2 + y_p^2)}{h_p^3} \frac{\eta}{\eta_{\text{máx}}} + \frac{1}{2} S_{III} \frac{y_p^2}{h_p^2} \frac{\eta^2}{\text{máx}^2} + \\ + \frac{1}{4} (S_{III} + S_{IV}) \frac{(x_p^2 + y_p^2)}{h_p^2} \frac{\eta^2}{\text{máx}^2} + \frac{1}{2} S_V \frac{y_p}{h_p} \frac{\eta^3}{\text{máx}^3}. \tag{3.8}$$

3.1.2. Sumas de Seidel para superficies cónicas

Los telescopios en la actualidad están constituidos por superficies cónicas, por ende, resulta necesario reescribir las Sumas de Seidel presentadas en las Ecuaciones 3.1 a 3.5 con un término adicional debido a la conicidad de las superficies. Por lo tanto, tendremos:

$$S_I = \sum_{k=0}^{n} A_k^2 h_k \Delta \left(\frac{u}{n} \right)_k - \sum_{k=0}^{n} (\kappa_k c_k^3 h_k^4 \Delta n_k), \tag{3.9}$$

$$S_{II} = \sum_{k=0}^{n} A_k \bar{A}_k h_k \Delta \left(\frac{u}{n} \right)_k - \sum_{k=0}^{n} (\kappa_k c_k^3 h_k^3 \bar{h}_k \Delta n_k), \tag{3.10}$$

$$S_{III} = \sum_{k=0}^{n} \bar{A}_k{}^2 h_k \Delta \left(\frac{u}{n} \right)_k - \sum_{k=0}^{n} (\kappa_k c_k^3 h_k^2 \bar{h}_k^2 \Delta n_k), \tag{3.11}$$

$$S_{IV} = \sum_{k=0}^{n} H_k^2 c_k \Delta \left(\frac{1}{n} \right)_k - \sum_{k=0}^{n} (\kappa_k c_k^3 h_k \bar{h}_k^3 \Delta n_k). \tag{3.12}$$

Reescribiendo las ecuaciones anteriores en términos de las sumas de Seidel (Hopkins, 1950; Wilson, 2013) obtenemos:

$$S_I^* = -\left[S_I - \sum_{k=0}^{n} (\kappa_k c_k^3 h_k^4 \Delta n_k) \right], \tag{3.13}$$

$$S_{II}^* = -\left[S_{II} - \sum_{k=0}^{n} (\kappa_k c_k^3 h_k^3 \bar{h}_k \Delta n_k) \right], \tag{3.14}$$

$$S^*_{III} = -\left[S_{III} - \sum_{k=0}^{n}(\kappa_k c_k^3 h_k^2 \bar{h}_k^2 \Delta n_k)\right], \tag{3.15}$$

$$S^*_{IV} = -\left[S_{IV} - \sum_{k=0}^{n}(\kappa_k c_k^3 h_k \bar{h}_k^3 \Delta n_k)\right]. \tag{3.16}$$

El término Δn_k, está dado por:

$$\Delta n_k = n_{k+1} - n_k. \tag{3.17}$$

3.1.3. Diseño de un telescopio tipo Cassegrain

En el diseño de un telescopio tipo Cassegrain como el que se muestra en la Figura 3.1, se requiere conocer las constantes de conicidad. Partimos de la Ecuación 3.13, y obtenemos:

$$S^*_I = \sum_{k=0}^{n} A_k^2 h_k \Delta\left(\frac{u}{n}\right)_k - \sum_{k=0}^{n}(\kappa_k c_k^3 h_k^4 \Delta n_k). \tag{3.18}$$

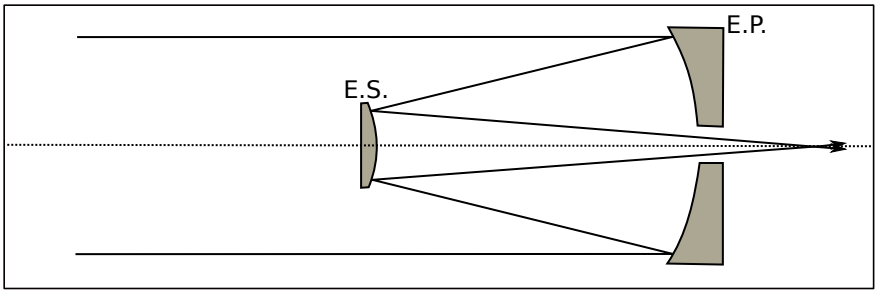


Figura 3.1: Configuración de un telescopio tipo Cassegrain. Nota: Espejo Primario (E.P.) y Espejo Secundario (E.S.)

La suma de la ecuación anterior va de 0 a 1 al considerarse dos espejos, entonces:

$$S^*_I = A_0^2 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1^2 h_1 \Delta\left(\frac{u}{n}\right)_1 - \kappa_0 c_0^3 h_0^4 \Delta n_0 - \kappa_1 c_1^3 h_1^4 \Delta n_1. \tag{3.19}$$

Sin embargo, el cociente de aberración esférica deber ser nulo para un telescopio tipo Cassegrain (Kidger, 2001; Smith, 2000), entonces:

$$0 = A_0^2 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1^2 h_1 \Delta\left(\frac{u}{n}\right)_1 - \kappa_0 c_0^3 h_0^4 \Delta n_0 - \kappa_1 c_1^3 h_1^4 \Delta n_1. \tag{3.20}$$

Debido a que la configuración de un telescopio tipo Cassegrain emplea un espejo primario parabólico, la constante de conicidad del primer espejo es $\kappa_0 = -1$ (Carroll and Ostlie, 2017). Conociendo los parámetros gaussianos (H, $\bar{A}$, A, etc.), el problema se reduce a simplemente calcular la constante de conicidad para el segundo espejo (κ_1), tal que la igualdad de la Ecuación 3.20 se cumpla. Consecuentemente, despejando κ_1, se obtiene:

$$\kappa_1 = \frac{1}{c_1^3 h_1^4 \Delta n_1}\left(A_0^2 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1^2 h_1 \Delta\left(\frac{u}{n}\right)_1 - \kappa_0 c_0^3 h_0^4 \Delta n_0\right), \tag{3.21}$$

$$\kappa_1 = \frac{1}{c_1^3 h_1^4 \Delta n_1}\left(S_I + c_0^3 h_0^4 \Delta n_0\right). \tag{3.22}$$

De esta manera, considerando los parametros del Telescopio de 2.1 m del OAN-SPM con razón focal 7.5 presentados (Radio de curvatura, diametro, separación entre espejos, etc.) en las Tablas 2.2 y 3.1, obtenemos que la constante de conicidad del espejo secundario debe ser $\kappa_1 = -3.581$.

3.1.4. Diseño de un telescopio tipo Ritchey-Chrétien

Un telescopio tipo Ritchey-Chrétien, se encuentra corregido de aberración de coma y esférica (Kidger, 2001; Smith, 2000), por lo que requerimos conocer las constantes de conicidad para que esto se cumpla. Las dos primeras sumas de Seidel que nos otorgan los coeficientes de aberración esférica y coma para superficies cónicas (Welford, 2017;

Kidger, 2001) están dadas por:

$$S_I^* = \sum_{k=0}^{n} A_k^2 h_k \Delta\left(\frac{u}{n}\right)_k - \sum_{k=0}^{n}(\kappa_k c_k^3 h_k^4 \Delta n_k), \tag{3.23}$$

$$S_{II}^* = \sum_{k=0}^{n} A_k \bar{A}_k h_k \Delta\left(\frac{u}{n}\right)_k - \sum_{k=0}^{n}(\kappa_k c_k^3 h_k^3 \bar{h}_k \Delta n_k). \tag{3.24}$$

Desarrollando las dos sumas desde 0 hasta 1, obtendremos:

$$S_I^* = A_0^2 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1^2 h_1 \Delta\left(\frac{u}{n}\right)_1 - \kappa_0 c_0^3 h_0^4 \Delta n_0 - \kappa_1 c_1^3 h_1^4 \Delta n_1, \tag{3.25}$$

$$S_{II}^* = A_0 \bar{A}_0 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1 \bar{A}_1 h_1 \Delta\left(\frac{u}{n}\right)_1 - \kappa_0 c_0^3 h_0^3 \bar{h}_0 \Delta n_0 - \kappa_1 c_1^3 h_1^3 \bar{h}_1 \Delta n_1. \tag{3.26}$$

Definimos las siguientes constantes:

$$S_I = A_0^2 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1^2 h_1 \Delta\left(\frac{u}{n}\right)_1, \tag{3.27}$$

$$S_{II} = A_0 \bar{A}_0 h_0 \Delta\left(\frac{u}{n}\right)_0 + A_1 \bar{A}_1 h_1 \Delta\left(\frac{u}{n}\right)_1, \tag{3.28}$$

$$\alpha_i = c_i^3 h_i^4 \Delta n_i \quad \& \quad \beta_i = c_i^3 h_i^3 \bar{h}_i \Delta n_i. \tag{3.29}$$

Las constantes definidas anteriormente, nos permiten reescribir las Ecuaciones 3.25 y 3.26 de la siguiente forma:

$$S_I^* = S_I - \kappa_0 \alpha_0 - \kappa_1 \alpha_1, \tag{3.30}$$

$$S_{II}^{*} = S_{II} - \kappa_0\beta_0 - \kappa_1\beta_1. \tag{3.31}$$

Debido a que necesitamos que el valor para los coeficientes de aberración esférica y coma para superficies cónicas sean nulos, nos enfocamos en encontrar los valores de κ_i, por lo tanto, se deben cumplir las siguientes igualdades:

$$S_I = \kappa_0\alpha_0 + \kappa_1\alpha_1, \tag{3.32}$$

$$S_{II} = \kappa_0\beta_0 + \kappa_1\beta_1. \tag{3.33}$$

Matricialmente se pueden visualizar como:

$$\begin{pmatrix} S_I \\ S_{II} \end{pmatrix} = \begin{pmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{pmatrix} \begin{pmatrix} \kappa_0 \\ \kappa_1 \end{pmatrix}, \tag{3.34}$$

donde para resolver el sistema de ecuaciones, nos limitamos a resolver:

$$\begin{pmatrix} \kappa_0 \\ \kappa_1 \end{pmatrix} = \begin{pmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{pmatrix}^{-1} \begin{pmatrix} S_I \\ S_{II} \end{pmatrix}. \tag{3.35}$$

Otra forma de obtener los valores de κ es despejando de la Ecuación 3.32, de esta manera se obtiene:

$$\kappa_0 = \alpha_0^{-1}(S_I - \kappa_1\alpha_1). \tag{3.36}$$

Sustituyendo la Ecuación 3.36 en la Ecuación 3.33, obtenemos:

$$S_{II} = \frac{\beta_0}{\alpha_0}(S_I - \kappa_1\alpha_1) + \kappa_1\beta_1. \tag{3.37}$$

Despejando κ_1 de la ecuación anterior, se obtiene:

$$\kappa_1 = \left(\beta_1 - \alpha_1\frac{\beta_0}{\alpha_0}\right)^{-1}\left(S_{II} - S_I\frac{\beta_0}{\alpha_0}\right), \tag{3.38}$$

lo cual nos permite escribir la Ecuación 3.36 de la siguiente manera:

$$\kappa_0 = \alpha_0^{-1}\left(S_I - \alpha_1\left(\beta_1 - \alpha_1\frac{\beta_0}{\alpha_0}\right)^{-1}\left(S_{II} - S_I\frac{\beta_0}{\alpha_0}\right)\right). \tag{3.39}$$

Consecuentemente, utilizando los parametros del telescopio de 2.1 m con razón focal 7.5 (radio de curvatura, diametro, separación entre espejos, etc.) expuestos en las Tablas 2.2 y 3.1. los valores de κ_i para que se cumpla el diseño planteado en un principio son:

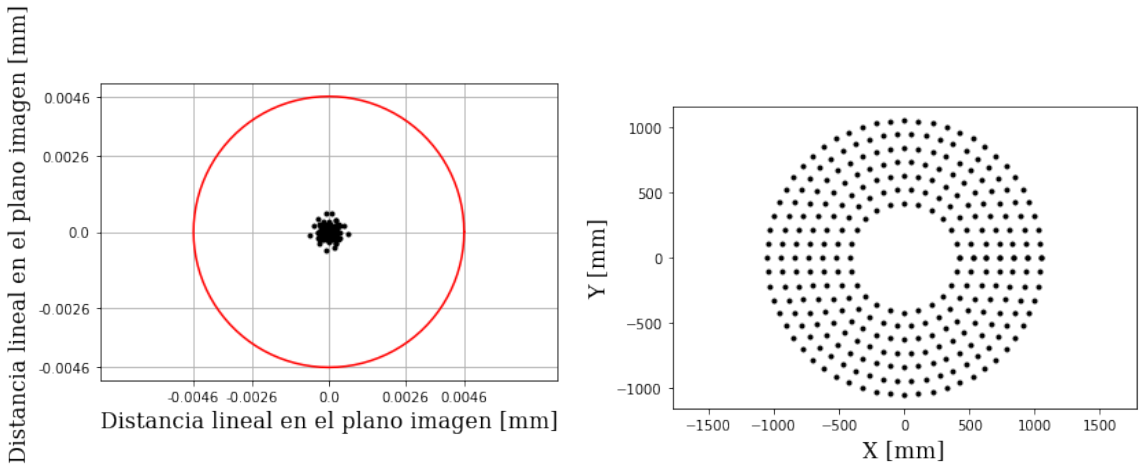
$$\begin{pmatrix}\kappa_0\\ \kappa_1\end{pmatrix} = \begin{pmatrix}-1.0748\\ -4.3824\end{pmatrix}. \tag{3.40}$$

En la Figura 3.2 se observa que el diagrama de manchas tiene un diámetro de 12 micras aproximadamente en comparación con el tamaño de los pixeles que tiene un CCD típico de 13.5 micras. La imagen resultante de este diseño en el campo cero (rayos paralelos al eje óptico), es una imagen puntal y, además, vemos que es más pequeña que el disco de Airy con lo cual está limitada por difracción.

3.2. Solución analítica clásica

La deducción de la solución analítica del punto de coma cero empieza al analizar la Figura 3.3, la cual nos permite observar que se cumple:

$$\cos(I) = \frac{PH}{PQ}, \quad \cos(I - I') = \frac{PG}{PQ}. \tag{3.41}$$



(a) Diagrama de manchas (b) Espejo primario

Figura 3.2: a) Diagrama de manchas para un telescopio tipo Ritchey-Chrétien con los parámetros de diseño del Telescopio de 2.1 m perteneciente al OAN-SPM. b) Conjunto de rayos que llegan al espejo primario. El círculo rojo representa el criterio del disco de Airy para el telescopio de 2.1 m del OAN-SPM a una longitud de onda de 500 nm.

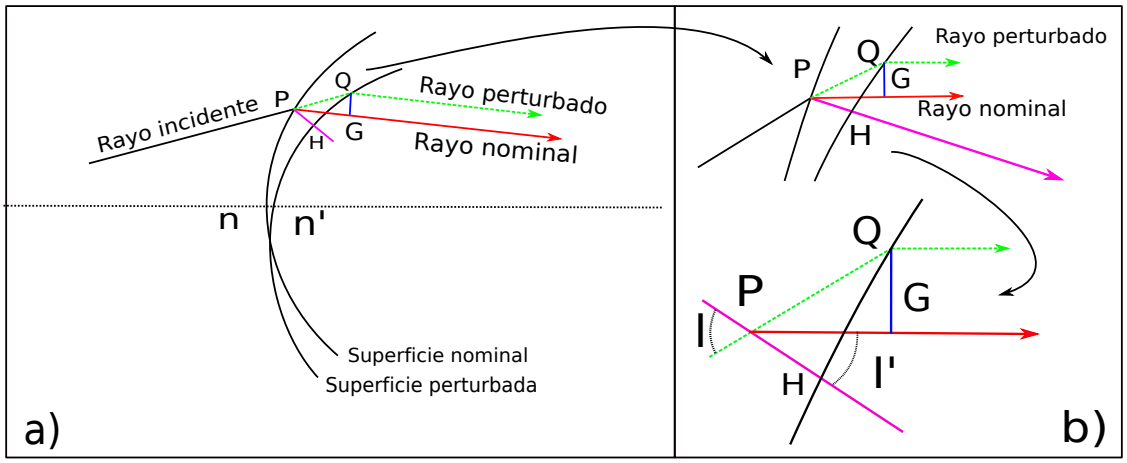


Figura 3.3: a) Diagrama de un haz incidente en una superficie dada n y esa misma superficie desplazada o perturbada n'. b) Amplificación a la zona donde el haz cruza tanto la superficie normal como la perturbada.

Utilizando el principio de Fermat: "**la longitud de camino óptico de la luz que viaja de un punto *A* a otro punto *B* tiene el mismo valor con respecto a una variación infinitesimal del camino**" (Hecht, 2016), es decir:

$$\delta L = \delta \sum n_i l_i = 0, \tag{3.42}$$

donde n_i es el indice de refración del medio y l_i es la trayectoria geométrica que sigue el rayo.

Podemos escribir la diferencia de camino óptico, dada por:

$$\delta w = \delta P = [PQ] - [PG] = n\overline{PQ} - n'\overline{PG}. \tag{3.43}$$

La Ecuación 3.41, permite reescribir la ecuación anterior de la siguiente forma:

$$\delta w = \frac{PH}{\cos(I)}[n - n'\cos(I - I')]. \tag{3.44}$$

Si tomamos en cuenta la ley de Snell dada en la Ecuación 2.2 y multiplicamos ambos lados por $\sin(I)$, tenemos:

$$n'\sin(I)\sin(I') = n\sin^2(I). \tag{3.45}$$

Además, la propiedad $\cos(a-b) = \cos(a)\cos(b)+\sin(a)\sin(b)$, aplicada a la ecuación anterior, nos otorga:

$$n'\cos(I - I') = n\sin^2(I) + n'\cos(I)\cos(I'). \tag{3.46}$$

Por lo tanto, sustituyendo la ecuación anterior en la Ecuación 3.44, vemos que:

$$\delta w = \frac{PH}{\cos(I)}[n - \{n\sin^2(I) + n'\cos(I)\cos(I')\}], \tag{3.47}$$

$$\delta w = PH[n\cos(I) - n'\cos(I')]. \tag{3.48}$$

En virtud de que sabemos la diferencia de camino óptico dada por la Ecuación 3.48, definamos lo siguiente: sea $[i = (k, l, m)]$ un vector unitario del rayo incidente y $[i' = (k', l', m')]$ un vector unitario del rayo refractado, entonces $[g = (K, L, M)]$ es un vector a lo largo de la normal (no normalizado) y $[\delta s = (\delta x, \delta y, \delta z)]$ es el desplazamiento de la superficie en P.

Con base en las definiciones realizadas, la Ecuación 3.48 se generaliza como:

$$\delta w = (\delta s \cdot g)\Gamma, \tag{3.49}$$

donde definimos:

$$\Gamma = \frac{(ni \cdot g - n'i' \cdot g)}{g^2}. \tag{3.50}$$

El caso especial de un telescopio reflector, simplemente se reduce a $\Gamma = -2i \cdot g$. Reescribiendo los vectores unitarios i y g en las Ecuaciones 3.49 y 3.50 en términos de d, (x, y) y c_s como se muestra en la Figura 3.4, obtendremos:

$$i = \left[-\frac{x}{d}, -\frac{y}{d}, 1 - \frac{s^2}{2d}\right] \quad \& \quad g = \left[-\frac{dz}{dx}, -\frac{dz}{dy}, 1 - c_s\right], \tag{3.51}$$

donde x y y son coordenadas sobre la superficie, $s^2 = x^2 + y^2$ y z es la ecuación de la sagita del espejo secundario dada por la ecuación de superficies cónicas. Realizando el

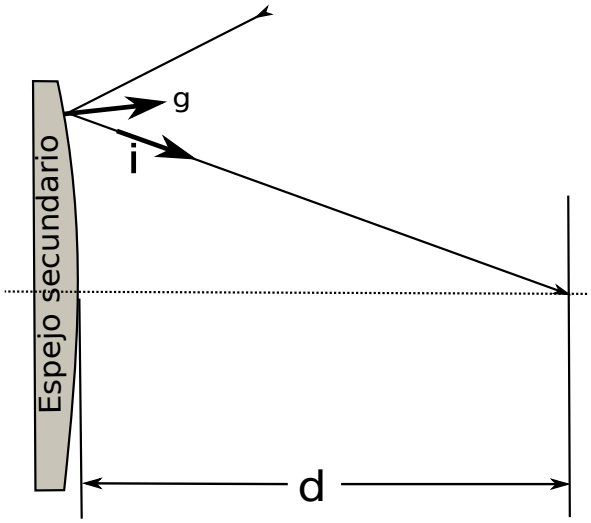


Figura 3.4: Sistema de coordenadas del espejo secundario.

producto punto entre estos dos vectores, se obtiene:

$$i \cdot g \approx 1 - \left(\frac{s^2}{2d^2}\right)(1 - c_s d^2). \tag{3.52}$$

Tenemos los valores de casi todos los elementos que son involucrados para resolver la Ecuación 3.49, sin embargo, el único que nos falta conocer es δs.

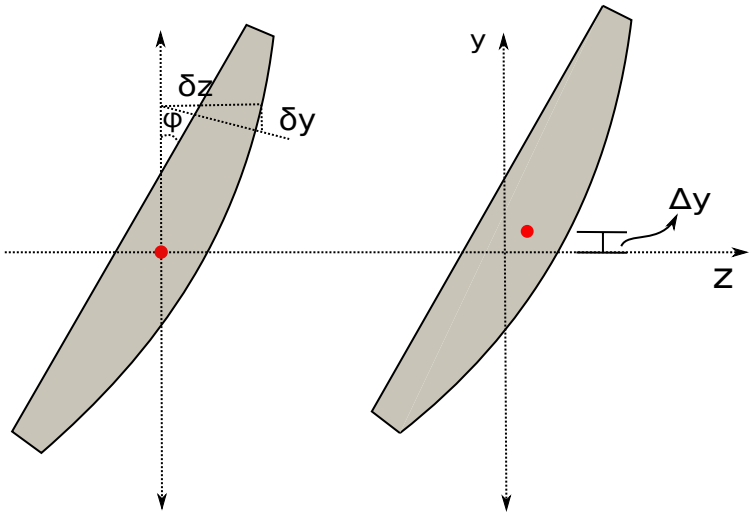


Figura 3.5: Inclinación y desplazamiento del espejo secundario con respecto al eje y.

Debido a que δs describe el desplazamiento de la superficie en un punto dado, se

puede realizar un desplazamiento en la dirección y una cierta cantidad Δy, como se observa en la Figura 3.5. Una perturbación Δy, implica una cierta perturbación δy. Para conocer el valor de esta perturbación vemos que el valor del ángulo formado debido al desplazamiento es:

$$\tan(\phi) \approx \phi = \frac{\delta z}{y} \rightarrow -\delta z \approx \phi y,$$

$$\tan(\phi) = \frac{\delta y}{z} \rightarrow \delta y = \frac{\phi c_s s^2}{2}.$$

De esta manera, con estas dos expresiones se puede construir δs, dado por:

$$\delta s = (0, \Delta y + \frac{\phi c_s s^2}{2}, -\phi y). \tag{3.53}$$

Definiendo $i \cdot g = A$ y sustituyendo todos los elementos deducidos en la Ecuación 3.49, obtenemos:

$$\delta w = -2\delta s \cdot g(A) = A\delta s_y g_y + A\delta s_z g_z = \Delta y A g_y + \phi\left[\frac{c_s s^2}{2} A g_y - y A g_z\right]. \tag{3.54}$$

Desarrollando todos los términos de la ecuación de la sagita dada por la Ecuación 2.14, la ecuación anterior se reescribe como:

$$\delta w = \left\{-\frac{\Delta y}{2R_s}\left[\left(\frac{1}{d} - \frac{1}{R_s}\right)^2 + \frac{(1-\kappa)}{R_s^2}\right] + \phi\left[\frac{1}{R_s^2} + \frac{1}{2}\left(\frac{1}{d} - \frac{1}{R_s}\right)^2\right]\right\} s^2 y + \phi[\ldots] s^4 y. \tag{3.55}$$

La Ecuación 3.55 define la diferencia de camino óptico cuando existe una inclinación y un desplazamiento del espejo secundario. La inclinación y el desplazamiento del espejo secundario introducen la misma forma de coma axial, de esta manera, la manera de compensar la coma introducida es realizando una inclinación o un desplazamiento.

Consecuentemente, la ecuación analítica para calcular la posición del punto de coma cero (ZCP) se determina para la corrección de tercer orden (Wetherell and Rimmer, 1972), dada por:

$$\frac{ZCP}{R_s} = \frac{(m+1)}{[(m+1)-(\kappa-1)(m-1)]}, \tag{3.56}$$

donde ZCP es la distancia al punto neutro desde el secundario, R_s, κ y $m = F/F_p$ son el radio de curvatura, la constante de conicidad y la amplificación del espejo secundario. Asimismo, F es la razón focal del sistema y F_p es la razón focal del espejo primario.

Para realizar el cálculo del ZCP utilizando la expresión analítica, resulta necesario conocer los parámetros de construcción de los telescopios del OAN-SPM, los cuales se presentan en la Tabla 3.1, donde los telescopios SAINT-EX y TAOS-II fueron omitidos, debido a que sus parámetros aún no son públicos.

Empleando los parámetros expuestos en la Tabla 3.1 se puede proceder a calcular los valores del ZCP para dichos telescopios, utilizando la expresión analítica presentada en la Ecuación 3.56. Los valores se muestran en la Tabla 3.2.

Tabla 3.1: Parámetros de construcción de los Telescopios del OAN-SPM e INAOE.

Telescopio (m)	Razón focal $f/\#$	Espejo	R_c [b] (mm)	Distancia entre espejos (mm)	k	Diámetro (mm)	Vértice de M_1 al plano imagen (mm)
0.84 [c.1]	15	M_1	−5287.0	2029.7	−1.0049	840.0	877.973
		M_2	−1555.0		−2.6990	250.0	
1.5 [c.1]	13	M_1	−5975.0	2475.7	−1.0049	1540.0	876.98
		M_2	−1208.0		−1.8970	275.0	
2.1 [c.2]	7.5	M_1	−9638.0	3452.2	−1.0773	2118.0	1037.53
		M_2	−3930.0		−4.3281	673.0	
2.1 [c.2]	13.5	M_1	−9638.0	3974.7	−1.0773	2118.0	1069.09
		M_2	−2028.0		−2.7284	406.0	
2.1 [c.2]	30	M_1	−9638.0	4366.7	−1.0773	2118.0	1449.467
		M_2	−981.0		−2.3947	195.0	
6.5 (TSPM)[c.3]	5.1	M_1	−16256.0	6178	−1.0000	6502.4	1851.28
		M_2	−5150.9		−2.6946	1714.5	
2.12 (INAOE[a]) [c.4]	11.9	M_1	−11340.0	4463.3	−1.0274	2118.0	900.063
		M_2	−3114.1		−2.7747	330	

Note. — [a] El telescopio INAOE 2.12 m no forma parte del OAN-SPM, no obstante, forma parte de la comunidad astronómica mexicana.
[b] El signo negativo del radio de curvatura del espejo primario se debe a la ley de los signos para los radios de curvatura de las superficies, dado que cuando el centro de curvatura C está del mismo lado que la luz saliente, el radio de curvatura es positivo; en caso contrario, es negativo. Esto permite que haya coherencia con el algoritmo de trazo exacto de rayos (ver Sección 4.2)
[c.1](González et al., 2018), vease también en https://www.astrossp.unam.mx/es/. [c.2](Herrera et al., 2017), vease también en https://www.astrossp.unam.mx/es/. [c.3](Uribe et al. (2016);González et al. (2018)), continua siendo un proyecto, aun no se encuentra en funcionamiento. [c.4](Carrasco et al., 2017), vease también https://www.inaoep.mx/˜astrofi/cananea/.

Tabla 3.2: Punto de cero coma para los telescopios el OAN-SPM.

Telescopio (m)	Razón focal $f/\#$	Solución Clásica (mm)
2.1	7.5	1188.06
2.1	13.5	688.38
2.1	30.5	321.75
1.5	13	504.46
0.84	15	563.37
1.3 (TAOS-2[a])	4	591.00[a]
1.0 (SAINT-EX)	7.8	535.87
6.5 (TSPM*b*)	5.1	1950.15
2.12 (INAOE)	11.9	1130.17

Note. — [a] La solución clásica no se aplica para los telescopios TAOS-II. Los diagramas de manchas que muestran la veracidad de este argumento se mostrarán en la Sección 5.1. *b* El telescopio TSPM continua siendo un proyecto, aun no se encuentra en funcionamiento.

Capítulo 4

Algoritmo SoS_ZCP

En el presente Capítulo describiremos el algoritmo desarrollado en Python para el trazado de rayos exacto, comenzando con la Seccción 4.1 en donde proporcionamos una breve explicación sobre el trazo de rayos matricial y describiremos los parámetros ópticos que pueden ser calculados a partir de este. Lo anterior nos servirá como punto de comparación en virtud de que se expondrá el trazo exacto de rayos. La Sección 4.2 se dividirá en dos subsecciones, en la Sección 4.2.1 se comentará el procedimiento general para el desarrollo del trazo exacto de rayos y en la Sección 4.2.2, se discutirá la implementación de las transformaciones de rotación y desplazamiento. Además, se discutirá un procedimiento para poder evaluar y eliminar la presencia de coma axial, a partir de la coma transversal. Finalmente, se realiza una explicación de la funcionalidad de las tres secciones que conforman nuestro algoritmo.

4.1. Trazo de rayos matricial

El trazo de rayos matricial utiliza la representación de los elementos ópticos y de sus separaciones como matrices (Hecht, 2016), lo cual agiliza la comprensión del funcionamiento del sistema óptico. Las dos matrices que componen esta representación

son conocidas como la matriz de refracción y de transmisión (Hecht, 2016). De esta manera, se pueden emplear distintos parámetros para llevar acaba el trazo de rayos; verbigracia, las distancias focales o potencias de cada elemento o sus radios de curvatura e índices de refracción. Sin embargo, cuando los elementos son superficies reflectoras se hace uso de sus potencias o distancias focales, debido a que la distancia focal a primera aproximación es la mitad del radio de curvatura (ver Sección 2.1.2) (Hecht, 2016).

La matriz de refracción se define como:

$$\begin{pmatrix} 1 & -\frac{1}{f_i} \\ 0 & 1 \end{pmatrix}. \tag{4.1}$$

Por otra parte, la matriz de transmisión es:

$$\begin{pmatrix} 1 & 0 \\ d_i & 1 \end{pmatrix}. \tag{4.2}$$

La información del rayo incidente se introduce al sistema mediante una matriz que contiene información sobre la altura h_i del rayo y su ángulo θ_i (Hecht, 2016), con respecto al eje óptico:

$$\begin{pmatrix} \theta_i \\ h_i \end{pmatrix}. \tag{4.3}$$

La matriz que contiene la información del rayo saliente, su altura h_{fin} y su ángulo θ_{fin}, se obtiene multiplicando la matriz del rayo de entrada con la matriz de refracción del primer elemento, posteriormente con la matriz de transmisión y finalmente con la

matriz de refracción del segundo elemento. Como se muestra continuación:

$$\begin{pmatrix} \theta_{fin} \\ h_{fin} \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{f_{i+1}} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ d_i & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{f_i} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_i \\ h_i \end{pmatrix}. \tag{4.4}$$

De esta manera, un sistema conformado por elementos de izquierda a derecha se debe resolver de derecha a izquierda en esta representación matricial. La Ecuación 4.4 se puede reescribir como:

$$\begin{pmatrix} \theta_{fin} \\ h_{fin} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \theta_i \\ h_i \end{pmatrix}, \tag{4.5}$$

donde la matriz $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ es conocida como matriz del sistema (Hecht, 2016). Además, tiene la peculiar característica de que $-b$ corresponde a la potencia total del sistema $b = -F_{total}^{-1}$ (Hecht, 2016). Igualmente, se pueden utilizar algunas componentes que constituyen la matriz del sistema para calcular los planos principales, anterior y posterior del sistema (Hecht, 2016), como se muestra a continuación:

$$P = \frac{1-d}{b}, \tag{4.6}$$

$$P' = \frac{a-1}{b}. \tag{4.7}$$

La sencillez del método permite ser utilizado en el telescopio de 2.1 m del OAN-SPM. No obstante, un sistema de espejos puede ser tratado de forma lineal como si fueran elementos refractores, como se muestra en la Figura 4.1.

Consecuentemente, considerando los espejos del telescopio de 2.1 m como si fueran

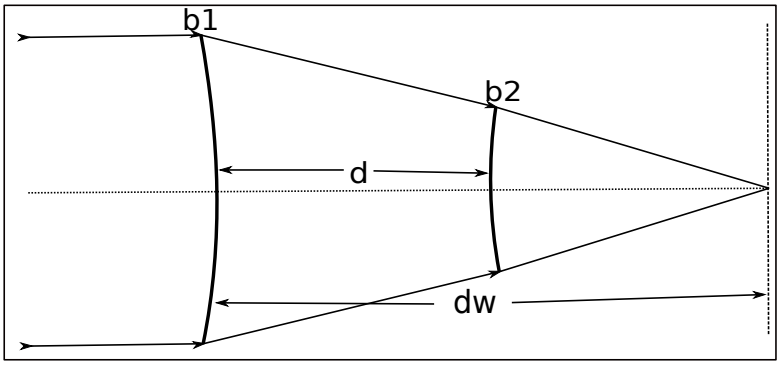


Figura 4.1: Configuración del sistema óptico del telescopio de 2.1 m del OAN-SPM, tratado de forma lineal como si estuviera compuesto por elementos refractores. La componente $b1$ es el espejo primario, la componente $b2$ es el espejo secundario, d es la distancia entre espejos y dw es la distancia de trabajo[1].

elementos refractores, se obtiene la siguiente matriz del sistema:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ d+dw & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{f_2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ d & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{f_1} \\ 0 & 1 \end{pmatrix}. \tag{4.8}$$

La matriz que contiene el término $d + dw$ se debe a que en el diseño del telescopio se establece una distancia de trabajo[7] en donde se coloca un detector o, en su defecto, algún ocular. Resolviendo la Ecuación 4.8, obtenemos:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1-\frac{d}{f_2} & -(\frac{1}{f_1}+\frac{1}{f2}-\frac{d}{f_1 f_2}) \\ d+dw+d\left(1-\frac{d+dw}{f_2}\right) & 1-\frac{(d+dw+d(1-\frac{(d+dw)}{f2}))}{f1}-\frac{(d+dw)}{f2} \end{pmatrix}. \tag{4.9}$$

Por lo tanto, dado que $F_{total} = -b^{-1}$ y utilizando el resultado de la Ecuación 4.9, se sustituyen los valores de los tres espejos secundarios disponibles para el telescopio de 2.1 m del OAN-SPM. Se obtienen los resultados presentados en la Tabla 4.1.

No obstante, recordemos que si queremos conocer la matriz del rayo saliente, debemos multiplicar la matriz del sistema (ver Ecuación 4.9) por la matriz del rayo de

[7]La distancia de trabajo se determina mediante la medición lineal desde la última superficie óptica al plano focal, para sistemas ópticos compuestos.

Tabla 4.1: Distancia focal total de los tres espejos secundarios para el telescopio de 2.1 del OAN-SPM.

Razón focal $f/\#$	R_1 (mm)	R_2 (mm)	d (mm)	F_{total} (mm)
7.5	9638	−3930	3452.2	15829.7140
13.5	9638	−2028	3974.6	28811.7099
30.5	9638	−981	4365.7	63487.4288

entrada, donde para un rayo incidente con los parámetros $h_i = D_{prim}/2$ y $\theta_i = 0$, se tendrá:

$$\begin{pmatrix} \theta_{fin} \\ h_{fin} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 0 \\ D_{prim}/2 \end{pmatrix}. \tag{4.10}$$

Sustituyendo los valores que se tienen para cada espejo secundario en la matriz del sistema de la Ecuación 4.10, obtenemos los parámetros de la Tabla 4.2.

Tabla 4.2: Parámetros de un rayo saliente dado un rayo incidente para los tres espejos secundarios del telescopio de 2.1 m del OAN-SPM.

Razón focal $f/\#$	R_2 (mm)	θ_{fin} (°)	h_{fin} (mm)
7.5	−3930	−3.8331	0.0361
13.5	−2028	−2.1060	−0.0156
30.5	−981	−0.9557	0.0135

4.2. Trazo exacto de rayos

Considerando que los rayos y la formación de imágenes se efectúa exclusivamente en la región paraxial, el trazado de rayos únicamente será valido en esta región. No obstante, numéricamente podemos obtener valores tanto a la altura del rayo incidente como a los ángulos de convergencia de los rayos, para así estimar valores fuera de dicha región. Asimismo, de la misma forma como se realiza la formación de una imagen Gaussiana[8] para un par conjugado, podemos efectuar un proceso donde los rayos pueden

[8]La imagen gaussiana es la imagen producida cuando se considera la aproximación de ángulos pequeños, también conocida como aproximación gaussiana o paraxial (Kidger, 2001)

ser trazados desde un determinado objeto O_1 en el plano–objeto, sin llevar acabo una aproximación de la óptica Gaussiana. Lo anterior se realiza utilizando la ley de Snell sin ninguna aproximación (Equación 2.2).

De esta forma se pueden conocer las intersecciones con el plano de la imagen Gaussiana. Empero, debido a las aberraciones ópticas del sistema (ver la Sección 2.3) dichas intersecciones generalmente no coincidirán en O_1' (la imagen Gaussiana de O_1). Lo anteriormente discutido se conoce como trazado exacto de rayos, el cual resulta distinto a los rayos paraxiales, que por construcción, tienen una forma matemáticamente infinitesimal. Existe también una complicación adicional inherente al hecho de que se puede tener un sistema con superficies fuera de eje, además de superficies que pueden no ser las superficies cónicas de revolución que se utilizan en los diseños clásicos.

El trazado exacto de rayos se sustenta en la secuencia iterativa de dos operaciones. La primera de ellas es la transferencia, la cual sencillamente considera un rayo a partir de donde deja una superficie óptica hasta donde se encuentra con la siguiente y esto comprende exclusivamente la geometría de las líneas rectas y las superficies con las formas apropiadas. La segunda operación se conoce como refracción o reflexión, la cual se resuelve encontrando la dirección del rayo después de haber pasado o haber sido reflejado o difractado por la superficie; esto involucra la ley de Snell, o una forma de la misma, apropiada para los elementos ópticos considerados.

4.2.1. Procedimiento general del trazo exacto de rayos

El trazo exacto de rayos, también conocido como "*skew ray tracing*" (Spencer and Murty, 1962), implica la utilización de vectores como la representación de rayos de luz. De esta manera, las coordenadas del punto origen de cada rayo están dadas por: $P_{-1}(x_{-1}, y_{-1}, z_{-1})$ y cada rayo tiene una dirección definida por sus cosenos directores

(L, M, N)[9]. Resulta necesario encontrar el punto de intersección con la superficie definida por la ecuación de la sagita (ver Ecuación 2.15), la cual se encuentra centrada en un nuevo sistema de coordenadas, desplazada a una distancia d. Sin embargo, antes de encontrar la intersección con la superficie debemos encontrar la intersección con el nuevo sistema en el que se encuentra la superficie (ver la Figura 4.2), donde las coordenadas de este punto para cada rayo son: $P_0(x_0, y_0, z_0)$. Las coordenadas de P_0 se pueden determinar utilizando:

$$x_0 = x_{-1} + \frac{L}{N}(d - z_{-1}), \tag{4.11}$$

$$y_0 = y_{-1} + \frac{M}{N}(d - z_{-1}). \tag{4.12}$$

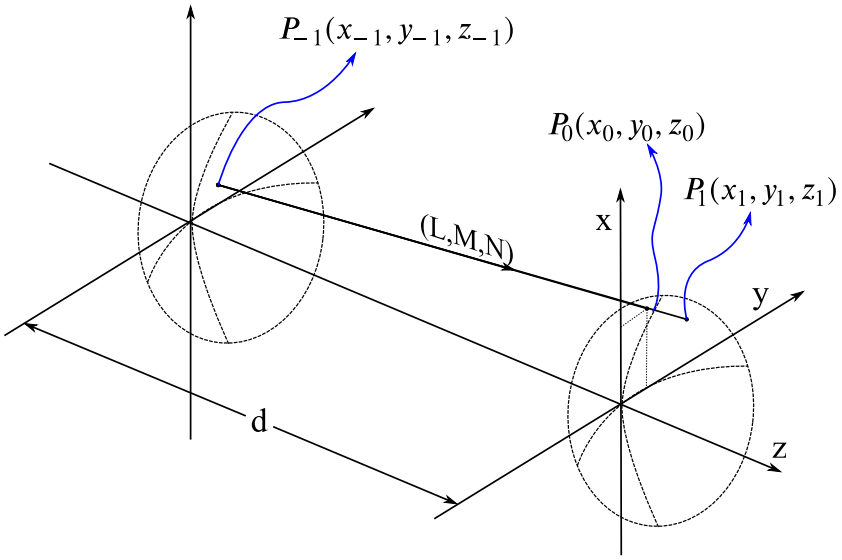


Figura 4.2: Trazo de rayos de una superficie al vértice de la siguiente superficie ubicada a una distancia d, optimizaciones posteriores convergen a la intersección con la segunda superficie.

A continuación se resuelve iterativamente la función rayo–superficie, como fue de-

[9]Todas las direcciones en el espacio están definidas a través de sus coordenadas o un vector unitario $\hat{u}$, de tal manera que, los cosenos directores se definen mediante los cosenos de los ángulos que forma el vector $\hat{u}$ con el espacio.

finida por Spencer and Murty (1962):

$$f_{RS}(x_1, y_1, z_1) = 0 = \frac{cr^2}{1 + \sqrt{1 - c^2 r^2 (\kappa + 1)}} + \sum_{j=1}^{n} \alpha_j r^{2j} - z_1, \tag{4.13}$$

donde $r^2 = x_1^2 + y_1^2$, y:

$$x_1 = z_1 \frac{L}{N} + x_0, \tag{4.14}$$

$$y_1 = z_1 \frac{M}{N} + y_0. \tag{4.15}$$

Las funciones descritas por las Ecuaciones 4.13, 4.14 y 4.15 se resuelven numéricamente utilizando el método de Newton–Raphson (Akram and Ann, 2015; Ypma, 1995) con un loop que incluye las ecuaciones:

$$f'_{RS}(z_1) = \lim_{h \to 0} \frac{f_{RS}(z_1 + h) - f_{RS}(z_1 - h)}{2h}, \tag{4.16}$$

$$z_t = z_1 - \frac{f_{RS}(z_1)}{f'_{RS}(z_1)}, \tag{4.17}$$

$$z_1 = z_t. \tag{4.18}$$

Una vez obtenido el punto de intersección $P_1(x_1, y_1, z_1)$, se calcula el vector normal a la superficie mediante la derivada numérica de la función rayo-superficie (ver Ecuación 4.13):

$$\hat{S} = -\frac{\left(\frac{\partial f_{RS}}{\partial x}, \frac{\partial f_{RS}}{\partial y}, \frac{\partial f_{RS}}{\partial z} \right)}{\sqrt{\left(\frac{\partial f_{RS}}{\partial x} \right)^2 + \left(\frac{\partial f_{RS}}{\partial y} \right)^2 + \left(\frac{\partial f_{RS}}{\partial z} \right)^2}}. \tag{4.19}$$

4.2.2. Transformaciones de rotación y desplazamiento en superficies ópticas

En el proceso de colimación, determinar la influencia en la dirección de un rayo debido a la rotación y/o desplazamiento de una superficie óptica es indispensable, específicamente porque necesitamos modificar la posición axial y la orientación del espejo secundario M_2. Partimos del hecho de que podemos calcular los valores de un segmento de rayo que viaja a través de dos superficies; en este caso, el segmento está definido por dos puntos extremos P_0 y P_1, los cuales tienen coordenadas (x, y, z) en un espacio tridimensional, como se puede observar en la Figura 4.3. La Figura 4.3 muestra también un sistema de coordenadas (x, z) centrado en el vértice de la superficie óptica de la izquierda, la cual representa el espejo secundario.

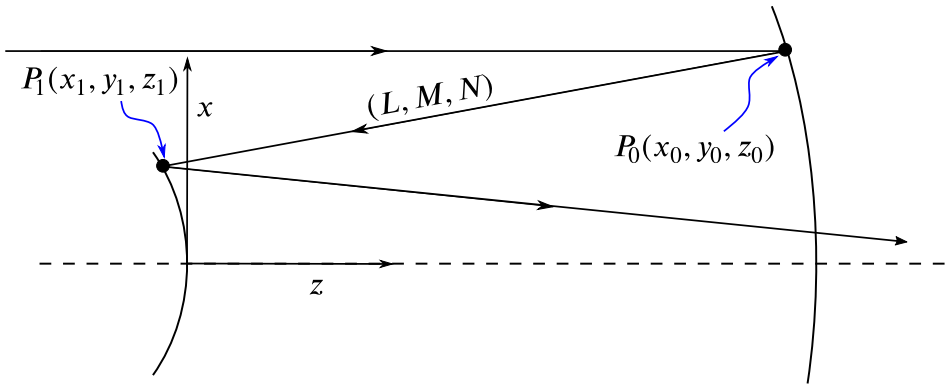


Figura 4.3: Un rayo representado como una línea en un espacio tridimensional como función de sus cosenos directores. Después de la intersección con el sistema de coordenadas de la izquierda, P_1 se convierte en el nuevo sistema de referencia.

Realizar un desplazamiento y una rotación del espejo secundario suscita que el sistema de coordenadas (x, z) ahora contenga a los ejes (x', z'), en los cuales se aplicó la transformación TR, como se muestra en la Figura 4.4. Además, desde el sistema de coordenadas transformado los puntos P_0 y P_1 tienen diferentes coordenadas, si los visualizamos desde el espacio de coordenadas transformado. Este es un nuevo espacio en el cual nos interesa calcular el punto de intersección con la superficie, sobre el cual se ha efectuado una trasformación P_T.

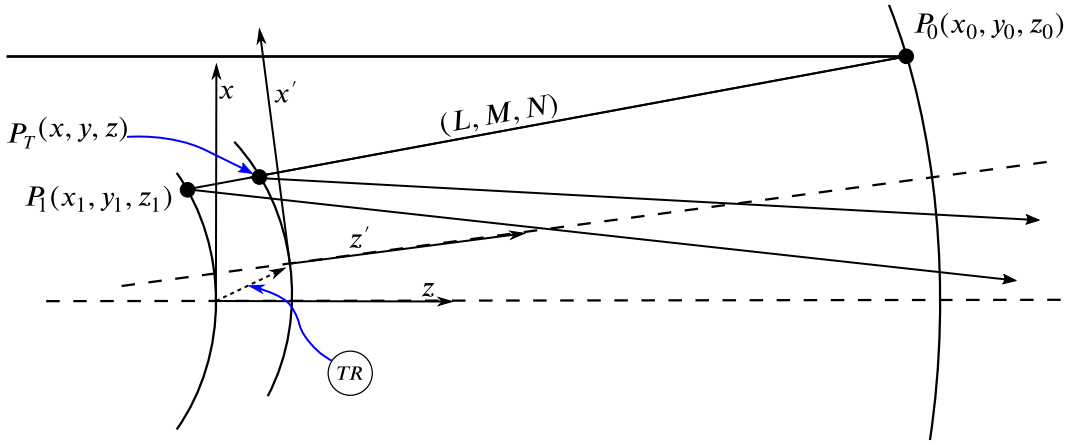


Figura 4.4: Detalles de la transformación TR de la superficie y del sistema de coordenadas. La operación es homóloga a una transformación inversa en el rayo antes de que este impacte en la superficie.

Los valores de estos puntos en el espacio transformado P_0' y P_1' se obtienen conociendo las matrices de traslación (T_{xyz}) y de rotación (R_x, R_y y R_z). La matriz de traslación contiene los desplazamientos en las tres direcciones espaciales (T_x, T_y y T_z), es decir:

$$T_{xyz} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.20}$$

Por otro lado, las matrices de rotación contienen los ángulos aplicados (θ_x, θ_y y θ_z), en la rotación alrededor de los ejes, entonces:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) & 0 \\ 0 & -\sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.21}$$

$$R_y = \begin{pmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.22}$$

$$R_z = \begin{pmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.23}$$

La matriz que contiene la transformación TR se obtiene del producto de las cuatro matrices anteriores y está dada por la Ecuación 4.24. Resulta trascendental recordar que la multiplicación de matrices no cumple con la propiedad de conmutativad, consecuentemente, en la Ecuación 4.24 las transformaciones se realizan de derecha a izquierda, ya que no es lo mismo trasladar y rotar que rotar y trasladar.

$$TR = T_{xyz} R_x R_y R_z. \tag{4.24}$$

Cada transformación TR aplicada a una superficie óptica implicará una transformación inversa TR^{-1} sobre los puntos que definen al rayo, por ejemplo, si la superficie se desplaza hacia arriba, desde el marco de referencia de la superficie es el rayo quien se desplaza hacia abajo.

La transformación de los puntos P_0 y P_1 empleando la matriz TR es:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = TR \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{4.25}$$

Obteniendo los puntos P'_0 y P'_1 es posible calcular los cosenos directores en el espacio transformado (L', M', N'). Los cosenos directores se obtienen de la siguiente manera:

$$L' = \frac{x'_1 - x'_0}{\sqrt{(x'_1 - x'_0)^2 + (y'_1 - y'_0)^2 + (z'_1 - z'_0)^2}}, \tag{4.26}$$

$$M' = \frac{y'_1 - y'_0}{\sqrt{(x'_1 - x'_0)^2 + (y'_1 - y'_0)^2 + (z'_1 - z'_0)^2}}, \tag{4.27}$$

$$N' = \frac{z'_1 - z'_0}{\sqrt{(x'_1 - x'_0)^2 + (y'_1 - y'_0)^2 + (z'_1 - z'_0)^2}}. \tag{4.28}$$

Una vez se calculados los cosenos directores y las coordenadas del rayo origen en el espacio transformado $\hat{I} = (L', M', N')$, se utilizan las Ecuaciones 4.16 a 4.18 para calcular el punto de intersección $P'_t(x', y', z')$ en la superficie transformada, usando la forma vectorial de la ley de Snell, dada en la Ecuación 2.4 (Bass et al., 2009). Sin embargo, la reescribimos con una notación adecuada, es decir:

$$\hat{R} = \frac{n_1}{n_2}\left[\hat{S} \times \left(- \hat{S} \times \hat{I}\right)\right] - \hat{S}\sqrt{1 - \left(\frac{n_1}{n_2}\right)^2 \left(\hat{S} \times \hat{I}\right) \cdot \left(\hat{S} \times \hat{I}\right)}. \tag{4.29}$$

La dirección del rayo refractado se define por el vector $\hat{R} = (L'_r, M'_r, N'_r)$ como se muestra en la Figura 4.5.

En el caso particular de que la superficie sea un espejo, el índice de refracción n_1 y n_2 son 1 y -1. En la presencia de una lente correctora se utiliza la fórmula de Sellmeier (Gooch, 2011) para una longitud de onda especifica λ y los coeficientes para el vidrio

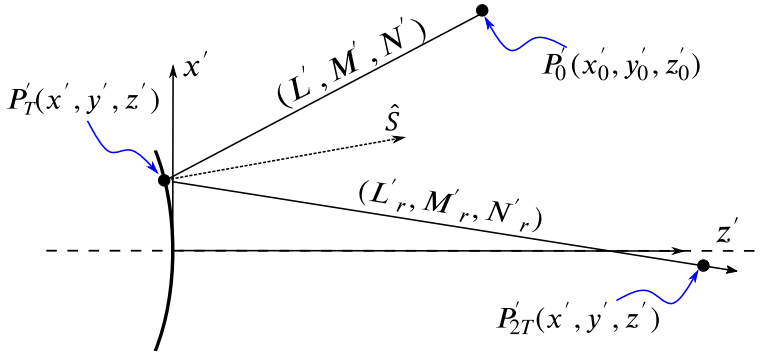


Figura 4.5: El punto de intersección en el sistema transformado y un punto sobre el rayo resultante se transforman de forma inversa al espacio original, para calcular los cosenos directores en el espacio original.

en particular, como se describe en la siguiente ecuación:

$$n^2 - 1 = \frac{K_1\lambda^2}{\lambda^2 - L_1} + \frac{K_2\lambda^2}{\lambda^2 - L_2} + \frac{K_3\lambda^2}{\lambda^2 - L_3}, \tag{4.30}$$

donde $K_{1,2,3}$ y $L_{1,2,3}$ son los coeficientes de un vidrio en específico, determinados experimentalmente.

A posteriori, se realiza la transformación inversa al espacio del sistema de coordenadas original, como:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = TR^{-1} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}. \tag{4.31}$$

No obstante, los cosenos directores no se transforman directamente al espacio original, de esta manera con (L'_r, M'_r, N'_r) y $P'_t(x', y', z')$ se debe determinar otro punto $P'_{2t}(x', y', z')$ en cierta posición de la recta, como se muestra en la Figura 4.5. Con los puntos P'_t y P'_{2t} se procede a calcular la transformación inversa y con los nuevos puntos en el espacio original se determinan los nuevos cosenos directores para el rayo $\hat{R}$ en el

espacio original.

4.3. Cambio en la amplificación por la presencia de coma

Se tiene varios procedimientos para poder evaluar y eliminar la presencia de la coma axial, como es el caso de la condición del seno de Abbe y el OSC (del inglés "*Offense Against Sine Condition*") (Smith, 2000). También se tienen formas generales de la condición del seno de Abbe que contemplan sistemas sin simetría de revolución (Elazhary et al., 2015). Sin embargo, la aberración de coma transversal no requiere calcular la posición de la pupila de salida del sistema, esta se define como el cambio en la amplificación en función de la apertura (ver Figura 4.6) y se expresa como:

$$Coma_T = H_{ab} - H_p. \tag{4.32}$$

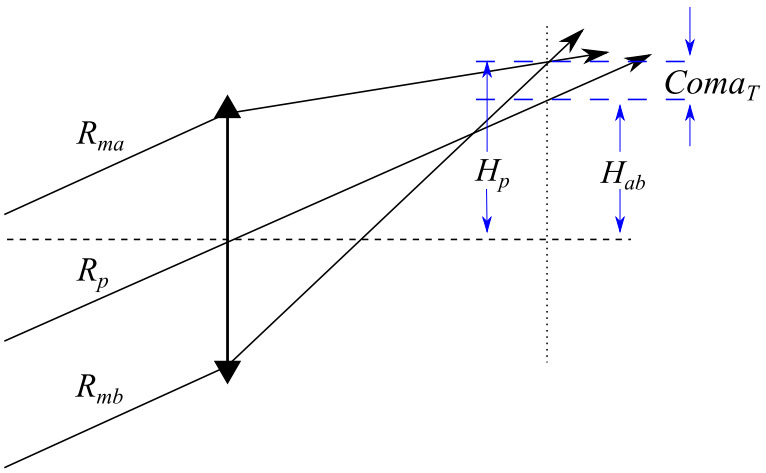


Figura 4.6: La coma transversal produce una amplificación diferente para el mismo objeto con una dependencia en la apertura del sistema.

En la Figura 4.6 podemos observar tres rayos que llegan al plano meridional, dos de ellos son R_{ma} y R_{mb} que pasan a través de los bordes de la pupila de sistema y el tercero pasa por el centro de la pupila, es decir, el rayo principal R_p. De lo visto en

la Sección 4.2, podemos estimar los cosenos directores de estos tres rayos y sus puntos de intersección con las superficies. Con esta información estamos en condiciones de determinar el punto de intersección entre los rayos R_{ma} y R_{mb}, la distancia de este punto al eje óptico, la posición del plano que pasa por este punto y es perpendicular al eje óptico y, finalmente, el punto de intersección del rayo principal R_p y su distancia al eje óptico.

4.3.1. Función de compensación de Coma

Aprovechando las ventajas del procedimiento para calcular el punto de coma cero utilizando el trazo de rayos exacto descrito en la Sección 4.2, se programó una función en Python para calcular el trazo de rayos exacto a través de un telescopio que tiene lentes correctoras. En este programa es posible definir los parámetros de conicidad y asféricidad como constantes. Con esta rutina, además, nos es posible introducir desplazamientos laterales en el espejo secundario. Como resultado de este programa, podemos calcular los parámetros de los tres rayos a la salida del sistema, como se describe de manera esquemática en la Figuras 4.6 y la Figura 4.7.

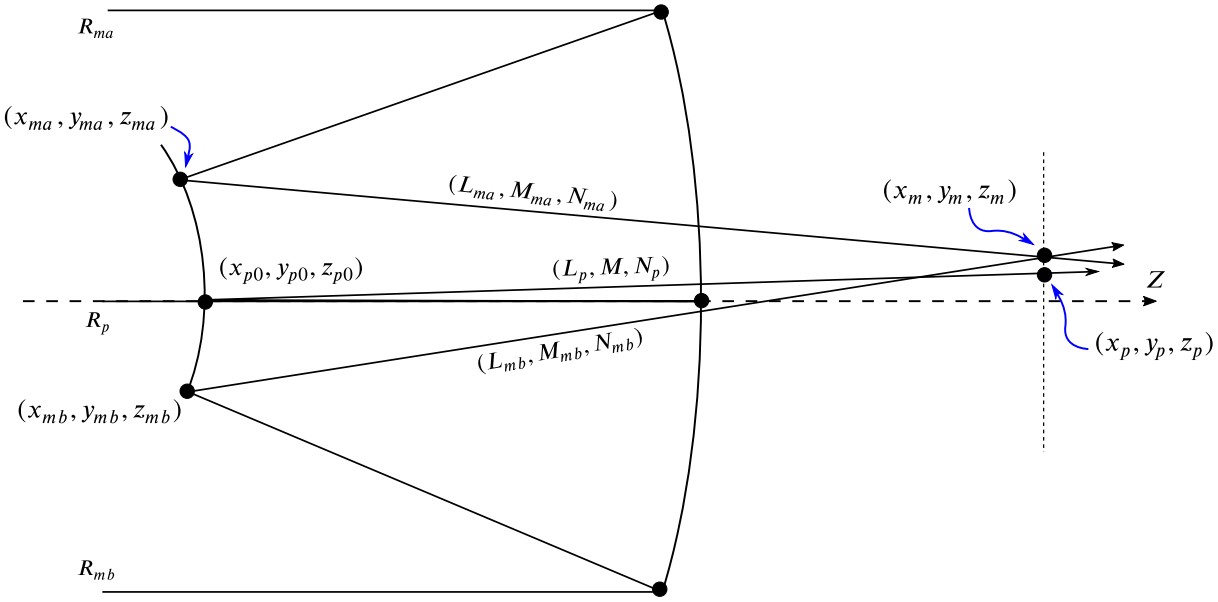


Figura 4.7: Parámetros de salida de los rayos utilizados para evaluar las Ecuaciones 4.33, 4.34 y 4.35, que nos ayudan a calcular la coma transversal.

Con los parámetros de los rayos marginales R_{ma} y R_{mb}, podemos determinar el punto y_m donde los vectores se cruzan y también la altura del rayo principal y_m:

$$y_m = \left(\frac{M_{ma}M_{mb}}{N_{ma}M_{mb} - N_{mb}M_{ma}}\right) \cdot \left(z_{mb} - z_{ma} - y_{mb}\frac{N_{mb}}{M_{mb}} + y_{ma}\frac{N_{ma}}{M_{ma}}\right), \tag{4.33}$$

$$z_m = (y_m - y_{mb})\frac{N_{mb}}{M_{mb}} + z_{mb}, \tag{4.34}$$

$$y_p = y_{p0} + \frac{M_p}{N_p}(z_m - z_{p0}). \tag{4.35}$$

Con estos parámetros podemos calcular la coma transversal, descrita por la Ecuación 4.32. Resolviendo con el método de Newton-Raphson para un valor de tilt para el espejo secundario que minimice la coma transversal, tendremos:

$$y_m - y_p = 0. \tag{4.36}$$

4.4. Serpent of Stars-Zero Coma Point

Serpent of Stars-Zero Coma Point (Nájera et al., 2021) es un algoritmo escrito en Python (Van Rossum and Drake Jr, 1995) y se divide en tres secciones en términos de su funcionalidad. La primera sección incorpora el minucioso procedimiento general del trazo exacto de rayos (Sección 4.2) para una superficie continua tridimensional general (ver Figura 4.8). En este código el usuario provee las propiedades de la superficie y su posición (tip/tilt y desplazamiento) en el espacio, por lo tanto, esta parte del algoritmo puede ser usada como un operador que es suministrado con rayos y regresa los parámetros de dichos rayos a través del sistema. Los rayos son proporcionados

como un conjunto de datos con las coordenadas de origen y sus cosenos directores, los datos entregados tendrán el mismo formato. La función es configurada con todos los parámetros del telescopio, superficie por superficie, incluyendo lentes correctoras de ser requeridas.

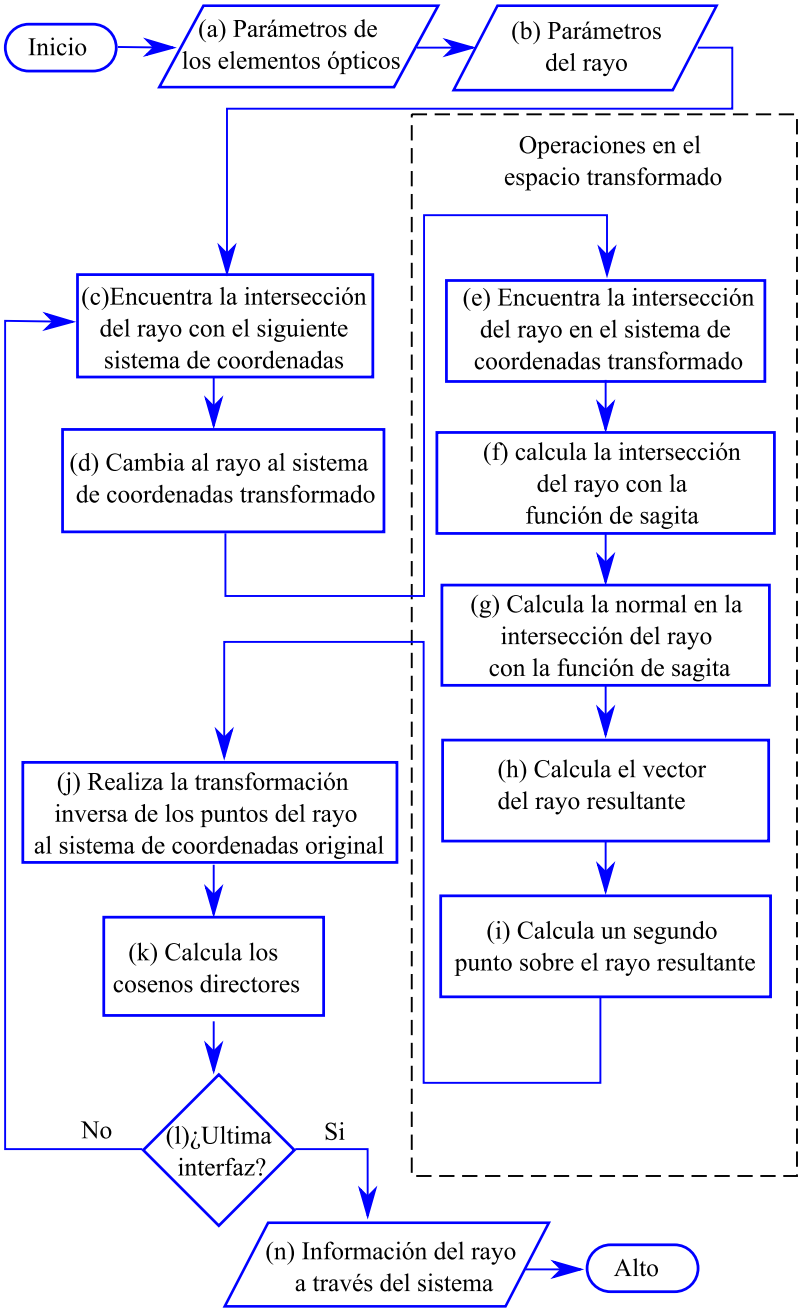


Figura 4.8: Diagrama de flujo del trazado de rayos exacto general.

La segunda sección del algoritmo, Figura 4.9, invoca a la primera función para trazar los tres rayos presentados en la Figura 4.7. Utilizando las Ecuaciones 4.33, 4.34

y 4.35 se calcula la aberración de coma, mediante la Ecuación 4.32. Resulta sustancial mencionar que, en esta segunda parte del algoritmo, el usuario puede modificar las propiedades del sistema, como el tilt y el desplazamiento de algún elemento.

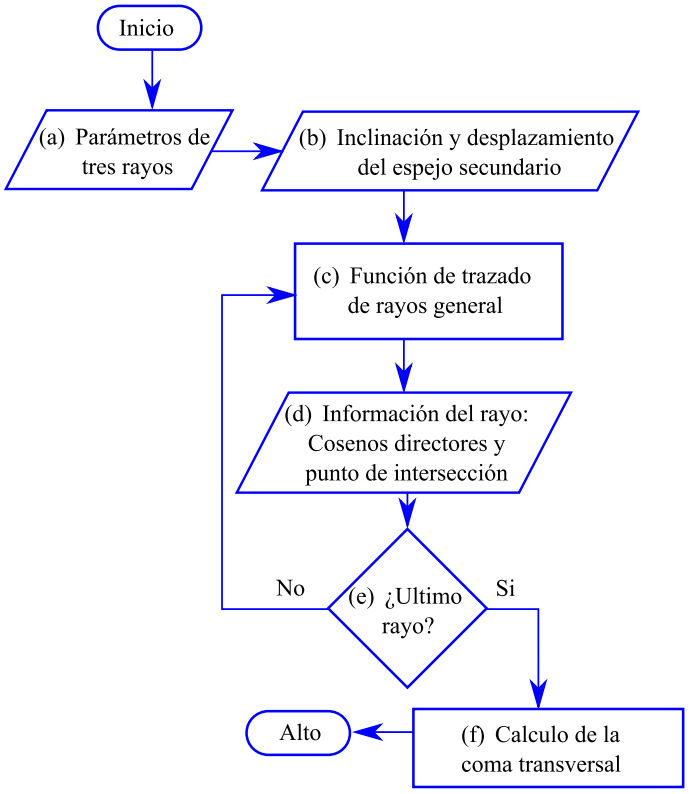


Figura 4.9: Diagrama de flujo de la función de evaluación Coma.

La tercera sección del algoritmo, Figura 4.10, toma los valores de salida de la segunda sección y los utiliza como una función de dos variables, T_x y θ_y, que son la traslación lateral y el ángulo tilt del espejo secundario M_2, descritos en la representación esquemática de la Figura 4.11.

Nuestro algoritmo emplea el método de Newton–Raphson en el proceso de optimización, con el objetivo de calcular el valor de θ_y tal que minimice la cantidad de coma, fijando un criterio cercano a cero (1×10^{-7} mm), para una determinada traslación T_x de M_2. La proyección del eje óptico compensado resultante de M_2 en el eje de M_1 es calculada directamente a partir del triángulo que se muestra en la Figura 4.11, como:

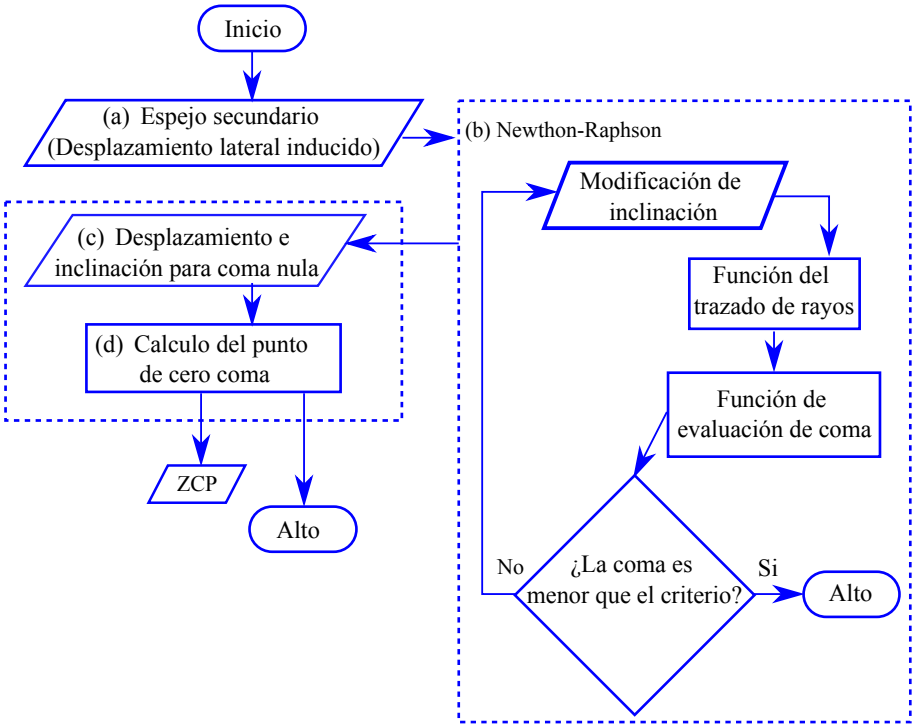


Figura 4.10: Diagrama de flujo de la función del punto de cero coma.

$$ZCP = \frac{T_x}{\tan(\theta_y)} \tag{4.37}$$

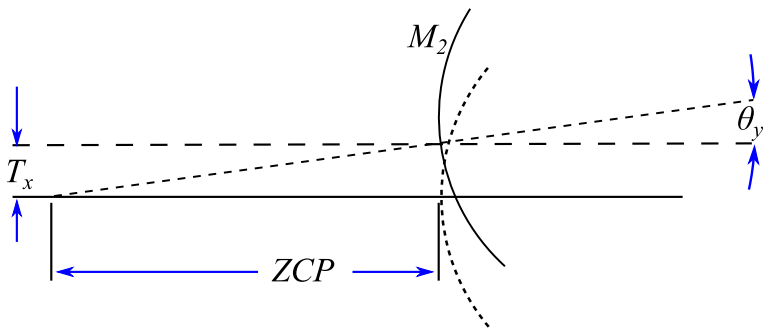


Figura 4.11: Cálculo del punto de cero coma mediante un triángulo definido por la translación lateral y el tilt.

En el Apéndice A, se muestran dos ejemplos del código desarrollado. Este está implementado en Python 2.7 con una licencia del MIT.

Capítulo 5

Resultados

En este Capítulo se presentarán los resultados más relevantes obtenidos a partir de calcular el punto de coma cero empleando tanto la solución analítica como la obtenida con nuestro algoritmo de forma numérica. El cálculo del punto de coma cero será realizado para los telescopios de 0.84 m, 1.5 m, 2.1 m, 1.3 m (TAOS-2), 1.0 m (SAINT-EX) y 6.5 m (TSPM) pertenecientes al OAN-SPM , además de incluir el telescopio de 2.12 m del Observatorio Astrofísico Guillermo Haro, localizado en Cananea, Sonora, México. Además, también mostraremos el cálculo del punto neutro para un telescopio teórico de campo amplio, similar a los telescopios del proyecto TAOS-II.

5.1. Telescopios del OAN-SPM

El OAN-SPM está localizado en la Sierra de San Pedro Mártir, en Baja California, México. En este sitio, encontramos un grupo de telescopios astronómicos profesionales en operación, además de proyectos en desarrollo que contemplan el funcionamiento otros telescopios, como el proyecto TAOS-II (Lehner et al., 2016) y el TSPM 6.5 (Richer et al., 2016). Calculamos el punto de coma cero para todos los telescopios clásicos del OAN-SPM, usando la expresión analítica (ver Ecuación 3.56), junto con los resultados

numéricos con nuestro algoritmo para realizar una comparación.

Los resultados son presentados en la Tabla 5.1, donde la primera columna lista los respectivos telescopios (el telescopio de 2.1 m del OAN-SPM tiene la particularidad de que su espejo primario puede ser acoplado con tres diferentes espejos secundarios), la segunda columna contiene la razón focal de cada telescopio, la tercera columna muestra el cálculo del punto de cero coma usando la solución analítica y en la cuarta columna contiene el punto de cero coma usando nuestro algoritmo; la última columna indica el error porcentual entre el valor clásico y el valor calculado con el trazo de rayos exacto.

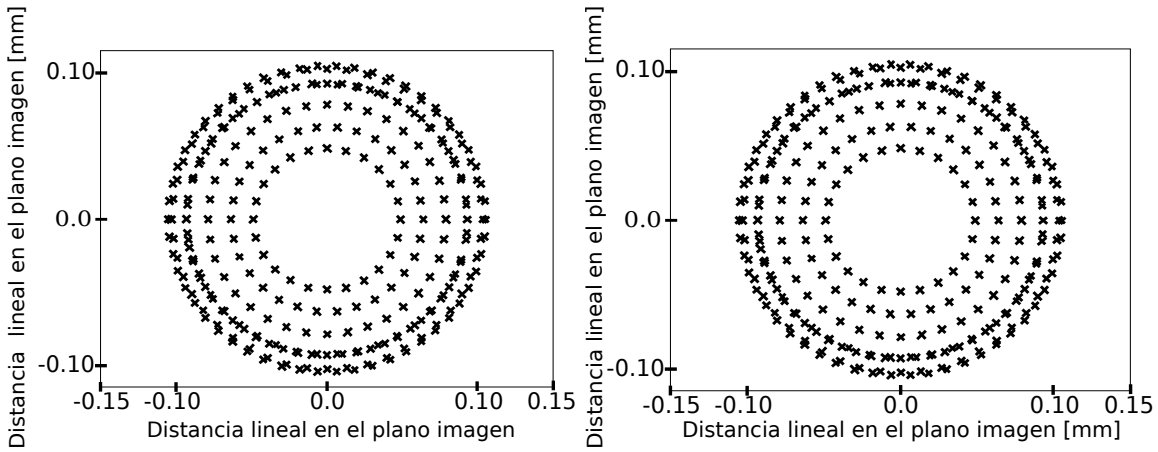
Tabla 5.1: Punto de cero coma para los telescopios el OAN-SPM.

Telescopio (m)	Razón focal $f/\#$	Solución Clásica (mm)	Trazo exacto de rayos (mm)	Error porcentual (%)
2.1	7.5	1188.06	1186.32	0.15
2.1	13.5	688.38	686.68	0.25
2.1	30.5	321.75	320.10	0.51
1.5	13	504.46	504.25	0.04
0.84	15	563.37	563.51	0.02
1.3 (TAOS-2)	4	591.00[a]	380.64	35.59
1.0 (SAINT-EX)	7.8	535.87	538.54	0.50
6.5 (TSPM)	5.1	1950.15	1953.58	0.18
2.12 (INAOE[b])	11.9	1130.17	1129.78	0.35

Note. — [a] La solución clásica no es aplicable para los telescopios TAOS-II (ver Figura 5.1).
[b] El telescopio INAOE 2.12 m no forma parte del OAN-SPM, no obstante, forma parte de la comunidad astronómica mexicana.

El resultado más relevante mostrado en la Tabla 5.1 es el hecho de que el valor calculado empleando la solución clásica y solución numérica del trazo de rayos exacto son completamente diferentes para telescopios con superficies asféricas, como podemos ver en el caso de los telescopios TAOS-II, donde el error porcentual es mayor del 35 %. Esto es de esperarse en virtud de que la solución clásica, como se demostró en la Sección 3.2, no contempla términos de asfericidad.

Como se puede ver en la Figura 5.1, empleando ambas soluciones del cálculo del



(a) ZCP = 591.00 mm (b) ZCP = 380.4 mm

Figura 5.1: a) Diagramas de manchas del telescopio del tipo TAOS-II. a) Muestra el diagrama de manchas considerando el punto de coma cero obtenido con la solución analítica. b) Diagrama de manchas considerando el valor del punto de coma cero obtenido con el algoritmo $SoS - ZCP$.

punto de coma cero aparentemente perecería que se compensa la aberración de coma para los telescopios del tipo TAOS-II. Sin embargo, recordemos que el punto de cero coma es un punto en el cual se puede pivotear sin introducir aberración de coma en el plano imagen.

Podemos ver que la corrección no es adecuada cuando se utiliza el punto de cero coma calculado con la solución analítica (ver Figura 5.2 parte superior), donde estamos moviendo el espejo secundario y aparece coma nuevamente, a diferencia del punto neutro calculado con el trazo de rayos exacto que, aunque se esté pivoteando con tres diferentes ángulos, no hay introducción de coma, sino únicamente un aumento lineal en el astigmatismo (Figura 5.2 parte inferior). Vemos entonces que ignorar los términos de asfericidad produce, sin duda, un error sistemático durante la colimación.

Tres telescopios mostrados en la Tabla 5.1 (1.5 m, SAINT-EX (Demory et al., 2020) y TAOS-II) tienen espejos secundarios con un sistema de posicionamiento electromecánico que permite todos los grados de libertad; sin embargo, el movimiento del espejo puede no producirse con respecto al vértice del espejo secundario o el punto de coma cero. Para aprovechar este mecanismo, debemos estudiar las componentes del

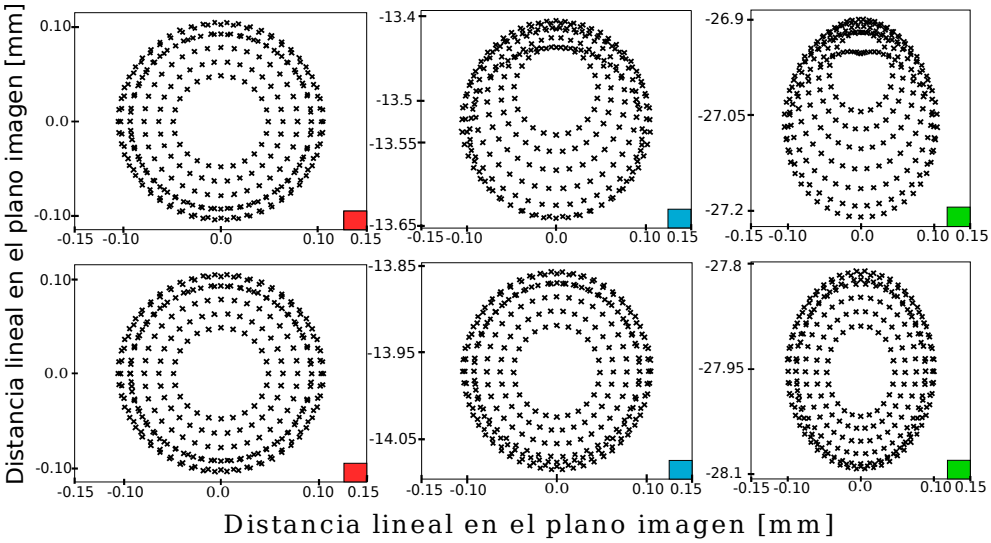


Figura 5.2: Diagrama de manchas para un telescopio del tipo TAOS-II, considerando los dos puntos de coma cero. La fila superior muestra el punto neutro calculado con la solución analítica, con esta se obtiene un punto de pivoteo a 591.00 mm sobre el espejo secundario. La parte inferior muestra el punto de coma cero obtenido con el trazo de rayos exacto lo que resulta en un punto de pivoteo a 380.64 mm sobre el espejo secundario. Ambos diagramas están desenfocados para mostrar imágenes anulares y resaltar la discrepancia entre ambos valores del punto neutro, en la parte superior vemos la aparición de coma nuevamente, mientras que en la parte inferior sólo hay astigmatismo. Los diagramas de manchas con un cuadro rojo presentan el espejo secundario sin pivoteo, es decir un ángulo de 0°, los de cuadro azul un ángulo de 0.2° y, finalmente, los de cuadro verde un ángulo de 0.4°.

vector de movimiento, con el objetivo de obtener un operador geométrico que nos permita realizar una combinación de pasos que, en conjunto, equivalen a un movimiento pivoteado en el punto de coma cero.

5.2. Ejemplificación de un telescopio de campo amplio

Como vimos en la Tabla 5.1 los valores son completamente diferentes cuando se realiza una comparación entre la solución clásica y la solución numérica para los telescopios del proyecto TAOS-II. Por este motivo, nos interesa mostrar como calcular un punto de coma cero para un telescopio similar a aquellos pertenecientes al proyecto TAOS-II, dado que tienen espejos secundarios asféricos, con un campo de visión muy amplio ($FOV = 1.7°$), que es el escenario ideal para esta aplicación. Los parámetros

de los telescopios TAOS-II aún no son públicos, sin embargo, el fabricante ha proporcionado toda la información para telescopios comerciales (Melsheimer and MacFarlane 2000; Bowen and Vaughan1973), en el cual nosotros hemos inspirado nuestro ejemplo de telescopio.

El diseño teórico de un telescopio $f/3$ con un espejo primario con diámetro de 2.1 m y un FOV de 1.4° ha sido preparado. El espejo primario y secundario tienen el mismo radio de curvatura para disminuir la curvatura de Petzval, el sistema tiene una placa asférica correctora y un espejo secundario, el cual también tiene términos de asfericidad. Los parámetros del diseño óptico son mostrados en la Tabla 5.2.

Tabla 5.2: Telescopio ϕ 2.1 m $f/3$ (Ejemplo Teórico).

Elemento	Radio* (mm)	Espesor (mm)	Vidrio	Semi-Diámetro (mm)	κ	α_1	α_2	α_3
M_1	−7670.112	−2272.41	Espejo	1005.0	−1.597			
M_2	−7670.112	2272.41	Espejo	465.0	−37.027			5.940e−19
Vértice M_1		56.538	Aire					
Corrector Adelante		7.0	Silica Schott	191.0		3.955e−5	−1.021e−9	
Corrector Atrás		300.0	Aire	191.0				
Plano Imagen				78.4				

Note. — * El signo negativo del radio de curvatura del espejo primario se debe a la ley de los signos para los radios de curvatura de las superficies, dado que cuando el centro de curvatura C está del mismo lado que la luz saliente, el radio de curvatura es positivo; en caso contrario, es negativo. Esto permite que haya coherencia con el algoritmo de trazo exacto de rayos (ver Sección 4.2)

Con el objetivo de calcular el punto de cero coma para este telescopio, introducimos los parámetros del sistema en nuestro algoritmo. De esta manera, encontramos que el punto de coma cero se encuentra a una distancia de 564.8 mm detrás del espejo secundario. En la Figura 5.3 se muestra un diagrama esquemático del telescopio, con el espejo secundario rotado 0.1°, pivoteado en torno al punto de coma cero. En la Figura 5.3 se muestran solamente los dos rayos marginales y el rayo principal, los cuales son paralelos al eje óptico. En el espacio imagen del sistema los rayos convergen al mismo punto sobre el eje, minimizando la coma. Para los rayos que llegan a un plano meridional (x, z) la convergencia sucede en un punto del rayo principal, para rayos incidentes en

el plano sagital (y, z) el punto en el rayo principal donde ellos convergen es diferente, causado por la presencia de astigmatismo.

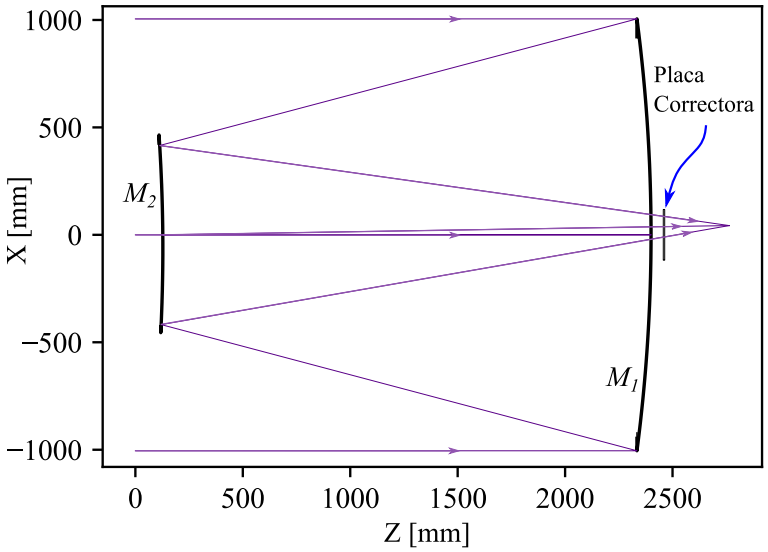


Figura 5.3: Ejemplo de telescopio de campo amplio. Los tres rayos paralelos convergen a un punto donde M_2 es pivoteado en torno al punto de coma cero del telescopio, el cual se encuentra localizado a 564.8 mm detrás del espejo secundario.

La Figura 5.4 (izquierda) muestra un conjunto de 9 diagramas de manchas para diferentes campos en el plano imagen para nuestro telescopio de campo amplio, en este caso el telescopio está perfectamente alineado y enfocado. La longitud de onda utilizada para el presente ejemplo es de 600 nm. Los diagramas de manchas fueron realizados con un arreglo de rayos con patrón polar (Malacara-Hernández and Malacara-Hernández, 2013) en la pupila de entrada M_1, donde se puede percibir un poco de astigmatismo, especialmente en las imágenes que se encuentran en las esquinas, sin ser esto debido a la escala de las gráficas. En la Figura 5.4 (derecha), la aberración de coma puede ser claramente visualizada sobre todo el campo, en este ejemplo el espejo secundario M_2 está desplazado lateralmente por 1 mm fuera del eje óptico, sin añadir tilt.

En la Figura 5.5 (izquierda) se muestra el diagrama de manchas producido por el mismo sistema que el de la Figura 5.4, pero esta vez se realizó un movimiento pivoteado con respecto al punto de coma cero. El plano imagen fue sacado ligeramente de foco para permitir que las imágenes muestren las típicas imágenes anulares producidas por

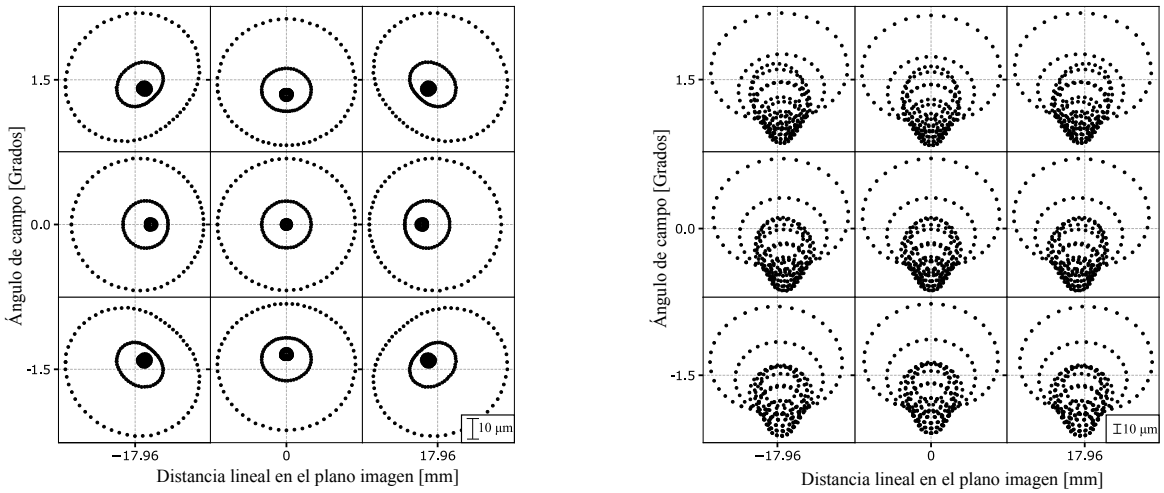


Figura 5.4: Diagrama de manchas de un telescopio de campo amplio, (izquierda) telescopio alineado, (derecha) aplicando un desplazamiento lateral de 1 mm en el espejo secundario, donde la aberración de coma es evidente.

telescopios fuera de foco. Estos diagramas de manchas presentan astigmatismo (Z_{22} de los polinomios de Zernike) que puede ser estimado utilizando la Ecuación 5.1 (Luna et al., 2007), la cual no es simétrica en el campo:

$$Z_{22} = (A - B)\frac{1}{4\sqrt{6}f/\#}, \tag{5.1}$$

donde A y B son los ejes mayor y menor de la elipse producida debido al astigmatismo. Es importante resaltar que, cuando el espejo secundario es pivoteado alrededor del punto de coma cero, no se introduce aberración de coma en el campo.

La Figura 5.5 (derecha) muestra los mismos diagramas de manchas como los mostrados en la Figura 5.5 (izquierda), pero con el enfoque correcto. A partir de la escala podemos percatarnos de que, incluso cuando se hace uso de las imágenes enfocadas, no será posible para nosotros evaluar con precisión la cantidad de astigmatismo presente realizando el proceso de colimación, especialmente bajo condiciones de mal "seeing". Consecuentemente, la corrección del astigmatismo se debe realizar en una alineación fina utilizando un sensor de frente de onda.

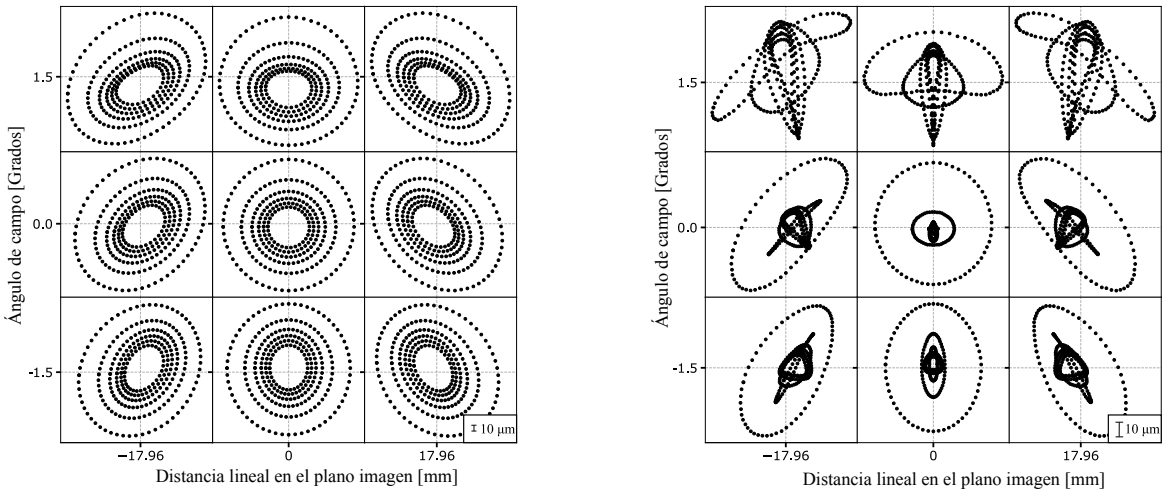


Figura 5.5: Diagrama de manchas de un telescopio de campo amplio, (izquierda) el espejo secundario está pivoteado con respecto al punto de coma cero, las imágenes se encuentran fuera de foco mostrando imágenes anulares, (derecha) los diagramas de manchas se encuentran enfocados correctamente, en ambos diagramas podemos percatarnos de la presencia de astigmatismo y una ligera contribución de coma ya que el diseño no tiene aberración nula originalmente.

Cabe destacar que los diagramas de manchas fueron realizados con la misma función programada para el trazo exacto de rayos (Figura 4.8, descrita en el Apéndice A.2.1). Esto es un subproducto del presente trabajo, el cual puede ser usado, modificado y aplicado a diferentes problemas académicos, tales como la optimización de parámetros para el diseño óptico, mediante el uso directo de una aberración particular como función de mérito; es decir, qué tanto valor o importancia queremos que tenga esa aberración en nuestro sistema óptico.

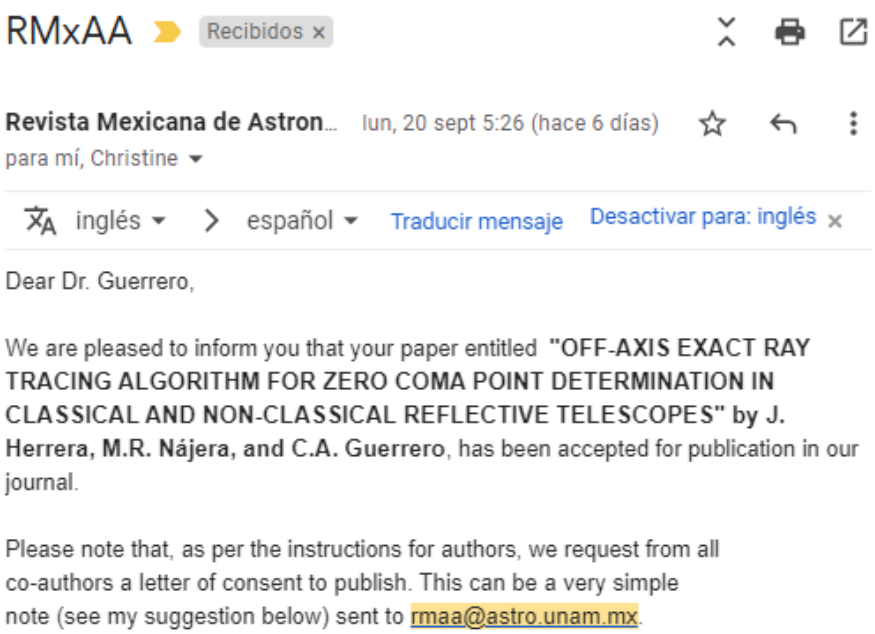
Capítulo 6

Trabajo de investigación del SOS_ZCP

En este Capítulo se presenta el artículo "*OFF-AXIS EXACT RAY TRACING ALGORITHM FOR ZERO COMA POINT DETERMINATION IN CLASSICAL AND NON-CLASSICAL REFLECTIVE TELESCOPES*" escrito a partir de los resultados obtenidos en la presente tesis. El artículo fue redactado por J. Herrera, M. R. Nájera y C. A. Guerrero y fue enviado a la Revista Mexicana de Astronomía y Astrofísica. El 20 de septiembre de 2021 (ver Figura 6.1) fue aceptado formalmente para su publicación.

6.1. Introducción

El equipo técnico del OAN-SPM, requiere de información que permita mejorar la colimación de los telescopios pertenecientes al Observatorio, ya que se encuentran trabajando activamente en la integración de metrología de calidad de imagen en tiempo real mediante la detección del frente de onda para un proceso de colimación determinista. El proceso de colimación de los espejos de un telescopio astronómico resulta


RMxAA Recibidos ×

Revista Mexicana de Astron... lun, 20 sept 5:26 (hace 6 días)
para mí, Christine

inglés > español Traducir mensaje Desactivar para: inglés ×

Dear Dr. Guerrero,

We are pleased to inform you that your paper entitled "OFF-AXIS EXACT RAY TRACING ALGORITHM FOR ZERO COMA POINT DETERMINATION IN CLASSICAL AND NON-CLASSICAL REFLECTIVE TELESCOPES" by J. Herrera, M.R. Nájera, and C.A. Guerrero, has been accepted for publication in our journal.

Please note that, as per the instructions for authors, we request from all co-authors a letter of consent to publish. This can be a very simple note (see my suggestion below) sent to rmaa@astro.unam.mx.

Figura 6.1: Correo de confirmación donde se nos informa de la aceptación formal del artículo.

fundamental dado que consiste en alinear su eje óptico para llevarlos a su posición de diseño, donde la calidad de imagen es óptima.

Existen varios procedimientos para llegar a este punto, sin embargo, en este estudio empleamos el cálculo del punto de coma cero, donde el espejo secundario pivota sobre un punto físico sobre el eje del espejo primario que permite una colimación determinista. De esta manera, desarrollamos un algoritmo general para calcular el punto de coma cero escrito en Python por mí (M. R. Nájera), para telescopios reflectores clásicos y no clásicos con superficies asféricas. Además, programé una función de trazado de rayos general aplicable a superficies cónicas con asfericidad, desplazamientos e inclinaciones, que puede ser modificada por los usuarios según las especificaciones que requieran y que está disponible para toda la comunidad astronómica.

6.2. Manuscrito para la RevMexAA

Manuscript for *Revista Mexicana de Astronomía y Astrofísica* (2007)

OFF-AXIS EXACT RAY TRACING ALGORITHM FOR ZERO COMA POINT DETERMINATION IN CLASSICAL AND NON-CLASSICAL REFLECTIVE TELESCOPES

J. Herrera,[1] M. R. Nájera,[1] and C. A. Guerrero[1]

Draft version: September 22, 2021

RESUMEN

El proceso de colimación de los espejos de un telescopio astronómico consiste en alinear su eje óptico para llevarlos a su posición de diseño, donde la calidad de imagen es óptima. Existen varios procedimientos para llegar a este punto, sin embargo, usaremos el cálculo del punto de coma cero, donde el espejo secundario se pivota sobre un punto físico que permite una colimación determinista. En este trabajo, presentamos un algoritmo general para calcular el punto de coma cero para telescopios reflectores clásicos y no clásicos con superficies asféricas. Como parte de nuestro algoritmo, programamos una función de trazado de rayos general aplicable a superficies cónicas con asfericidad, desplazamientos e inclinaciones, que pueden ser modificadas por los usuarios, según sus especificaciones. Esta función de trazado de rayos se utiliza para evaluar la coma transversal y determinar la conjunción óptima de desplazamiento e inclinación que evita la introducción de coma axial durante el proceso de colimación. Como ejemplo del uso de nuestro algoritmo, calculamos el punto de coma cero para cada telescopio clásico del Observatorio Astronómico Nacional, San Pedro Mártir, México. Además, también presentamos un ejemplo de un telescopio de campo amplio con superficies asféricas, como será el caso de los telescopios TAOS-II, donde no se puede utilizar la expresión analítica clásica.

ABSTRACT

The collimation process of the mirrors of an astronomical telescope consists of aligning their optical axis to bring them to their design position, where the image quality is optimal. There are several procedures to reach this point, however, we will use the calculation of the zero coma point, where the secondary mirror is pivoted around a physical point that allows a deterministic collimation. In this work, we present a general algorithm to calculate the zero coma point for classical and non classical reflective telescopes with aspherical surfaces. As part of our algorithm, we programmed a general ray tracing function applicable to conic surfaces with asphericity, displacements and inclinations, which can be modified by the users, according to their specifications. This ray tracing function is used to evaluate the transverse coma and determine the optimal conjunction of displacement and tilt that avoids

[1]Universidad Nacional Autónoma de México, Instituto de Astronomía, AP 106, Ensenada 22860, B.C., México

the introduction of axial coma during the collimation process. As an example of the usage of our algorithm, we calculated the zero coma point for every classical telescope of the Observatorio Astronómico Nacional, San Pedro Mártir, México. In addition, we also present an example of a wide-field telescope with aspherical surfaces, as will be the case of the TAOS-II telescopes, where the classical analytical expression cannot be used.

Key Words: Ground-based telescopes (687) — Wide-field telescopes (1688) — Astronomical instrumentation (799) — Interdisciplinary astronomy (804)

1. INTRODUCTION

For telescopes composed of two mirrors, the zero coma point (ZCP) determines a physical point over the optical axis of the primary mirror, M_1, where the secondary mirror, M_2, can rotate during the collimation process, without introducing transverse coma in the resulting image. The analytical expression for the ZCP in aplanatic reflector telescopes is well known and it is of great importance in the alignment of classical telescopes, such as the Cassegrain, Gregorian or Ritchey-Chretien type (Schroeder & Inc 2000; Schechter & Levinson 2011); however, this equation is not directly applicable to some variants of wide-field reflector telescopes, composed not only by conical surfaces but also containing asphericity terms in the secondary mirror, or telescopes coupled with aspherical corrective lenses.

In the present work, we have developed an exact-ray-tracing algorithm for off-axis aspherical surfaces, which we applied to the numerical determination of the ZCP for classical and non-classical telescopes. As an example of the application of this algorithm, we show the calculation of the ZCP for every telescope of the Observatorio Astronómico Nacional (OAN-SPM), which is located in the Sierra San Pedro Mártir, Baja California, México.

The content of this article is divided as follows: in § 2 we describe the analytical solution and how it is used in the collimation process, in addition, the concept of the ZCP is described. § 3 describes the concept of non-classical design for a telescopes with asphericity terms in one or more surfaces of the system. In § 4 we describe the transverse coma aberration by means of three rays traveling through the surfaces of the telescope, given that this process will be the basis for the function to be minimized using our proposed algorithm. In § 5 we describe the approach required for programming the general ray tracing function, using off-axis arbitrary surfaces. We inclided flowcharts representing the complete algorithm procedure in § 6, while in § 7 we discuss the spot diagrams and output plots for a theoretical wide-field telescope used as an example of the aplication of our algorithm. In § 8 we present the values of the ZCPs for the telescopes of the OAN-SPM and future projects under development. Finally, we present our conclusions in § 9.

2. CLASSICAL ANALYTIC SOLUTION

An aplanatic telescope composed of two mirrors with conical surfaces enables the possibility to correct the spherical aberration in an exact form or,

at least, a correction to third order. While the powers of both surfaces allow us to balance and control the Petzval field curvature and define the system's power, the conic constants allow us to control the spherical aberration, in the case of a Cassegrain telescope. However, with the improvement in the construction precision of conical surfaces of revolution, we have been able to reduce an additional optical aberration, typically, the coma.

A reflector telescope with a laterally misaligned or inclined element becomes an optical system without symmetry of revolution; this means that the optical axis is no longer coaxial with the mechanical axis, which in turn produces a displacement of the exit pupil that originates coma arising from misalignment. The nature of this induced aberration has been widely studied, for example by Schroeder & Inc (2000) and Schechter & Levinson (2011), of which the latter presents an exhaustive analysis of the patterns generated by misalignment. The understanding of these effects is of special relevance, since real-world telescopes are opto-mechanical systems subject to mechanical bending and misalignment.

A problem associated with the misalignment of reflecting telescopes is that it is impossible to unequivocally determine the origin of axial coma, as it can be originated due to lateral displacement, inclination of the secondary mirror, or a combination of both. The misalignment produced by displacement and inclination can even be of the same magnitude and of the opposite sign (Schroeder & Inc 2000), which allows the effect to be compensated and eliminated. In a misaligned system, astigmatism is asymmetric and linear in the field; this is added to the natural astigmatism due to optical design or astigmatism produced by the shape of the elements, either by construction or by errors in the support of the optical components.

In the collimation process of a telescope we look for the coaxiality of all the optical elements. A very useful collimation procedure is described by McLeod (1996). Prior to achieving an optimal collimation, the image plane of a slightly out-of-focus telescope exhibits donut-shaped images, which will be defocused and will display astigmatism and coma.

Typically the aberration of astigmatism is not a problem in nearly collimated telescopes, where the coma dominates the image quality. In this case, it may be sufficient to correct the coma by means of only lateral displacement or tilt of the secondary mirror. However, our main interest in this work is the collimation of wide-field telescopes, for which the astigmatism at the edges of the wide field will be more relevant than in small-field telescopes.

In the first collimation stage, we compensate the axial coma aberration by means of lateral displacements of the secondary mirror, until the image plane is dominated only by astigmatism. When the optical systems are aligned, the astigmatism is distributed with symmetry of revolution over the entire field, or it can be zero depending on the design of the system. There are strategies to measure the wavefront around the field to estimate the tilt or shift of the secondary mirror required to correct the astigmatism (McLeod 1996), however, this correction should not introduce coma aberration again.

For this reason, it is important to perform continuous compensation of astigmatism, by simultaneously moving and tilting the secondary mirror. This combination is homologous to move the secondary mirror, with its optical axis pivoting around a single point on the optical axis of the primary mirror. This point is known as the "zero coma point", also called "neutral point", which has been calculated in different ways by several authors; however, one of the most cited works is that of Schroeder & Inc (2000). Thus, by moving the secondary mirror with respect to this point, we avoid inducing axial coma in the image plane, when trying to compensate for astigmatism.

It is important to point out that, before starting any movement with respect to the ZCP, we must first reach the compensation of coma during the first collimation stage, as described before. Once this is achieved, we guarantee that the optical axes of the secondary and primary mirrors are aligned. Then, the analytical equation to calculate the position of the ZCP is determined for the third order correction, according to Equation 1 (Wetherell & Rimmer 1972):

$$ZCP = \frac{1}{C_{M_2}} \frac{m+1}{(m+1) - (K_{M_2} - 1)(m-1)}, \tag{1}$$

where ZCP is the distance to the zero coma point measured from the secondary mirror, C_{M_2} and K_{M_2} are the secondary mirror's curvature and conic constant, and $m = F/F_p$ is the amplification of the system.

3. NON-CLASSICAL TELESCOPES

There is, however, a caveat to the previously described procedure. There are telescopes that are composed of conical surfaces plus an asphericity polynomial term α_j. Other telescopes also contemplate one or more correcting aspherical lenses, whose sagittas are defined according to Equation 2:

$$z_1 = \frac{cs^2}{1 + \sqrt{1 - (K+1)c^2 s^2}} + \sum_{j=1}^{n} \alpha_j s^{2j}. \tag{2}$$

where z is the sagitta of the surface, $s^2 = x^2 + y^2$ is the distance of the optical axis to a point over the surface, K is the conic constant and $c = 1/R_c$ is the paraxial curvature, with R_c the radius of curvature of the surface.

As expected, Equation 1 for the ZCP is not appropriate for telescopes with these aspheric components. The objective of this work is to numerically determine the position of the ZCP in the most general way, given that corrective lenses can have different contributions in the correction of aberrations and an analytical estimation of the ZCP can rapidly grow in complexity, as we increase the number of surfaces to be considered.

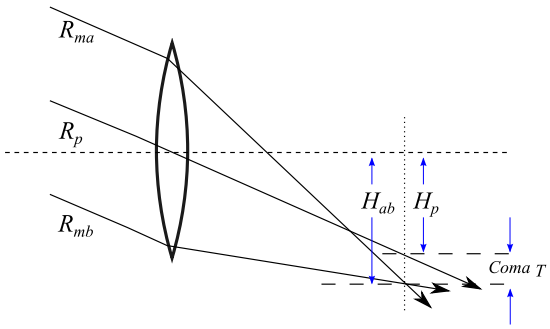


Fig. 1. Transverse coma produces a different amplification for the same object with a dependence on the system's aperture.

4. COMA-INDUCED CHANGE IN AMPLIFICATION

There are several procedures to evaluate and compensate the axial coma, such as the Abbe's sine condition and the offense against the sine condition (OSC) (Smith 2000). There are also general forms for the Abbe's sine condition that contemplate systems without symmetry of revolution (Elazhary et al. 2015). However, the transverse coma aberration does not require the calculation of the position of the system's exit pupil, which is defined as the change in amplification as a function of the aperture (see Figure 1), expressed as:

$$Coma_T = H_{ab} - H_p. \tag{3}$$

In Figure 1 we can see three rays arriving on a meridional plane, two of them, R_{ma} and R_{mb}, are passing through the edges of the pupil of the system and the third ray is passing through the center of the pupil, called the principal ray R_p. From the exact ray tracing we can find the direction cosines of these three rays and the intersection point on the surfaces. With this information we can determine the point of intersection between rays R_{ma} and R_{mb}, we can find the distance from this point to the optical axis, the position of the perpendicular plane to the optical axis passing through this point and, finally, the point of intersection of the principal ray R_p and its distance from the optical axis.

5. COMA COMPENSATION FUNCTION

The evaluation of the aberrations of an optical system can be done based on the exact ray tracing procedure, which determines the optical path of a ray passing through a set of surfaces. The exact ray tracing, is also known as "skew ray tracing" (Spencer & Murty 1962).

Taking advantage of the procedure for calculating the ZCP and using the exact ray tracing described in § 4, we programmed a function in Python for

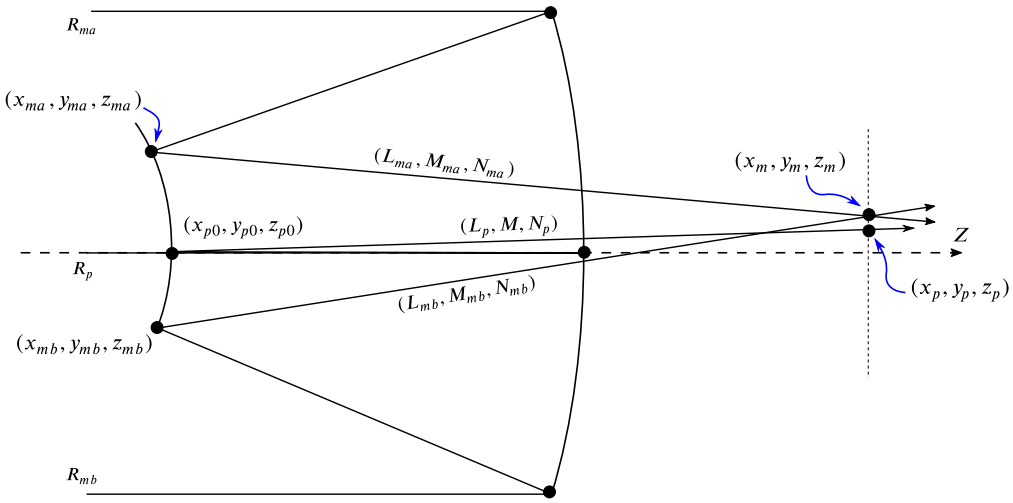


Fig. 2. Output ray parameters used in the evaluations of Equations 4, 5 and 6, that help us to compute the transverse coma.

calculating the exact ray tracing through a telescope that has a correcting lens. In this program it is possible to define the conic and asphericity parameters as constants. With this routine, it is also possible to introduce lateral displacements on the secondary mirror. As a result of this program, we can calculate the parameters of the three rays at the output of the system, as schematically described in Figure 1 and Figure 2.

With the parameters of the marginal rays R_{ma} and R_{mb} we can calculate the point y_m, where the vectors intersect, and also the height of the chief ray y_p:

$$y_m = \left(\frac{M_{ma}M_{mb}}{N_{ma}M_{mb} - N_{mb}M_{ma}}\right) \cdot \cdot \left(z_{mb} - z_{ma} - y_{mb}\frac{N_{mb}}{M_{mb}} + y_{ma}\frac{N_{ma}}{M_{ma}}\right), \tag{4}$$

$$z_m = (y_m - y_{mb})\frac{N_{mb}}{M_{mb}} + z_{mb}, \tag{5}$$

$$y_p = y_{p0} + \frac{M_p}{N_p}(z_m - z_{p0}). \tag{6}$$

With this parameters we are able to calculate the transverse aberration, described by Equation 3. Solving by the Newton-Raphson method for the tilt value of the secondary mirror that minimizes the transverse aberration, we have:

$$y_m - y_p = 0. \tag{7}$$

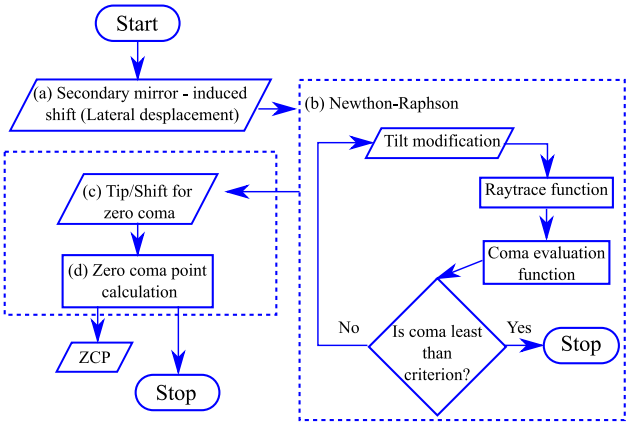


Fig. 3. ZCP function flowchart.

6. SOS-ZCP ALGORITHM

The SoS-ZCP (*Serpent of Stars-Zero Coma Point*) is an algorithm written in Python, divided into three sections in terms of its functionality. The first section incorporates the thorough General ray tracing function for a three-dimensional continuous surface. In this script the user provides the surface properties and position (tip/tilt and shift) in space, therefore, this part of the algorithm can be used as an operator that is supplied with a ray and then returns the ray parameters through the entire system. The ray is provided as a data set with the origin coordinates and direction cosines, the returned data are in the same format. The function is configured with all the telescope parameters, surface by surface, including corrector lenses if they are required.

The second section of the algorithm invokes the first function to trace the three rays presented in Figure 2. Using Equation 4, 5 and 6 the coma aberration is calculated, by means of Equation 3. It is important to mention that, in this second part of the algorithm, the user can modify the properties of the system, such as the tilt and shift of an element.

The third section of the algorithm (schematically described in Figure 3) takes the output of the second section and uses it as a function of two variables, T_x and θ_y, which are the secondary mirror M_2 lateral translation and tilt angle, respectively.

Our algorithm applies the Newton-Raphson method in an optimization process, in order to calculate the value of θ_y that minimizes the amount of coma, then setting a criterion near to zero (1e-7 mm), for a proposed translation T_x of M_2. The projection of the resulting compensated M_2 optical axis on the M_1 axis is calculated directly from the triangle defined by the lateral translation and tilt, thus, the ZCP is calculated as:

$$ZCP = \frac{T_x}{\tan(\theta_y)}. \tag{8}$$

TABLE 1

TELESCOPE ϕ 2.1 M $f/3$ (THEORETICAL EXAMPLE)

Element	Radius (mm)	Thickness (mm)	Glass	Semi-Diameter (mm)	k	α_1	α_2	α_3
M_1	−7670.112	−227241	Mirror	1005.0	−1.597			
M_2	−7670.112	227241	Mirror	465.0	−37.027			5.940e−19
M_1 Vertex		56.538	Air					
Corrector Front		7.0	Silica Schott	191.0		3.955e−5	−1.021e−9	
Corrector Back		300.0	Air	191.0				
Image Plane				78.4				

The source code of the SoS-ZCP is available at https://github.com/MNajeraR/SoS-ZCP (Nájera et al. 2021). It is implemented in Python 2.7 under the MIT License. Version v2.0.0 is used in this paper and is archived at http://doi.org/10.5281/zenodo.4929555.

7. WIDE-FIELD TELESCOPE EXEMPLIFICATION

We are especially interested in calculating the ZCP for a telescope similar to those of the TAOS-II project, given that they will have an aspheric secondary mirror and a very wide field of view ($FOV = 1.7°$), which is the ideal scenario for this implementation. The parameters of the TAOS-II telescopes are not yet public, however, the manufacturer has provided all the information for the commercial telescopes (Melsheimer & MacFarlane 2000; Bowen & Vaughan 1973), which we have used to design our example telescope.

We have designed a theoretical $f/3$ telescope, whose primary mirror is 2.1 m in diameter and has a $FOV = 1.4°$. The primary and secondary mirrors have the same radius of curvature to decrease the Petzval curvature, the system has an aspherical correction plate and the secondary mirror also has an asphericity term. The parameters of the design data are listed in Table 1.

In order to calculate the zero coma point for this telescope, we introduced its parameters in our algorithm. We calculated the position of the ZCP at a distance of 564.8 mm, behind the secondary mirror. In Figure 4 we show an schematic diagram of the telescope, with the secondary mirror rotated by 0.1°, pivoted around the ZCP. In Figure 4 we only show the two marginal rays and the principal ray, they are parallel to the optical axis. At the image space these rays converge to the same of-axis point, minimizing the coma. For rays arriving in a meridional plane (x, z) the convergence occurs at a point on the chief ray; for rays incident on the sagittal plane (y, z) the converging point on the chief ray is different, which is the physical origin of the astigmatism.

Figure 5 (left) shows a set of 9 spot diagrams for different fields on the image plane of our wide-field telescope, in this case the telescope is perfectly aligned and focused. The wavelength used for the present example is 600 nm. The spot diagrams were made with a polar-array ray pattern (Malacara-Hernández & Malacara-Hernández 2013) in the entrance pupil M_1, where

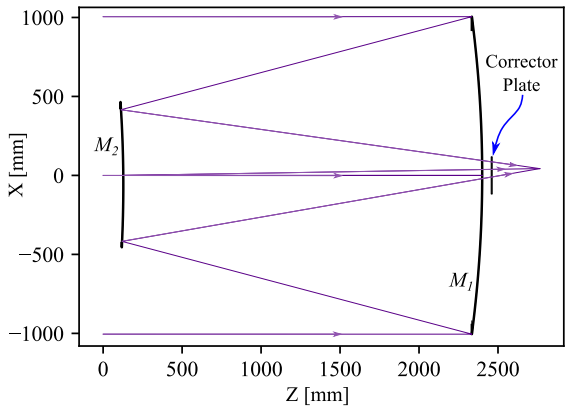


Fig. 4. Wide-field telescope example. The three parallel rays converge to the same point, when M_2 is pivoting around the telescope's ZCP, located at 564.8 mm behind the secondary mirror.

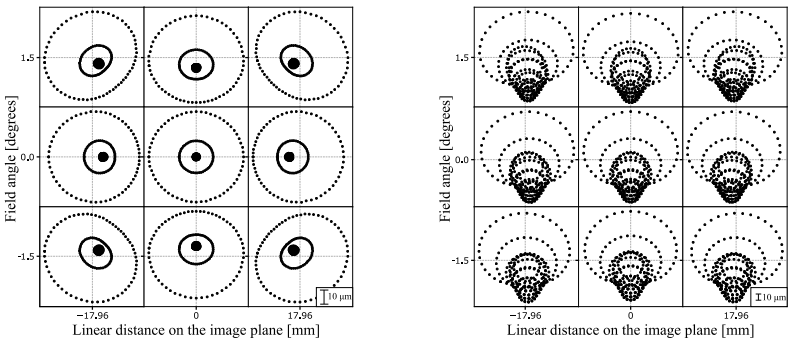


Fig. 5. Spot diagrams for a wide-field telescope, (left) aligned telescope, (right) introduction of a 1 mm lateral displacement in the secondary mirror, where the coma aberration is evident.

a slight astigmatism aberration is perceived, especially in the corner images, which is not related to the scale of the plots. In Figure 5 (right), coma aberration can be clearly seen over the entire field, in this example the secondary mirror M_2 is laterally displaced by 1 mm outside of the optical axis, without adding tilt.

Figure 6 (left) shows the spot diagram produced by the same system as Figure 5, but this time a pivoting movement has been made with respect to the ZCP. The image plane has been slightly putted out of focus to allow the images to display the typical annular images produced by out-of-focus telescopes. These spot diagrams display astigmatism aberration (Z_{22} from Zernike polynomials) that can be estimated using Equation 9 (Luna et al.

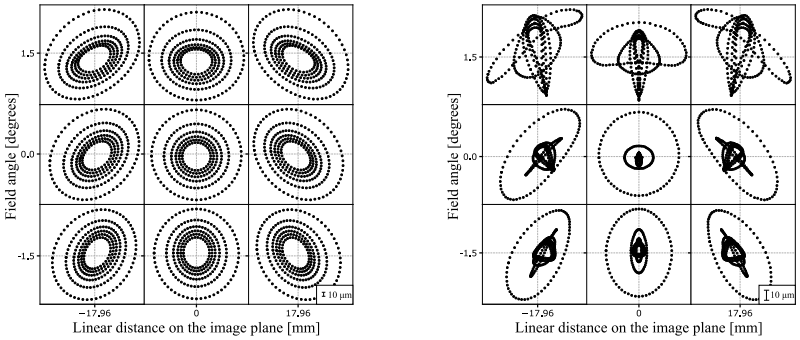


Fig. 6. Spot diagrams for a wide field telescope, (left) pivoting of the secondary mirror with respect to its ZCP, the images are out of focus to show annular images, (right) spot diagrams at the correct focus. In both diagrams we can see the presence of astigmatism and very little coma contribution, given that the original design has a small coma aberration.

2007), which is not symmetrical in the field:

$$Z_{22} = (A - B)\,\frac{1}{4\sqrt{6}f/\#}, \tag{9}$$

where A and B are the major and minor axes of the ellipse produced by astigmatism. It is important to stress that, when pivoting around the ZCP, coma aberration is not added to the field.

Figure 6 (right) shows the same spot diagrams as those in the left, but with the correct focus; we can see from the scale that, even when using focused images, it is not possible for us to accurately evaluate the amount of astigmatism when performing collimation, specially under bad seeing conditions. Therefore, the correction of astigmatism must be done in fine alignment using a wave-front sensor.

In Figure 6, the small proportion of spherical aberration is due to the presence of spherical aberration in the design of the primary mirror, because it was not completely corrected. The aberrations produced by misalignment are only coma and astigmatism, where the proportion of induced astigmatism depends on the particular design, which must be considered in each case.

It should be noted that the spot diagrams were made with the same function programmed for the exact ray tracing (General ray tracing function). This is a by-product of the present work, which can be used, modified and applied to various other academic problems, such as the optimization of the parameters of an optical design, by direct use of a particular ray aberration as a merit function.

TABLE 2

CONSTRUCTIONAL PARAMETERS FOR OAN-SPM TELESCOPES

Telescope (m)	Focal ratio $f/\#$	Mirror	R_c	Thickness	k	Diameter	M_1 Vertex to Image plane
2.1	7.5	M_1	−9638.000	3452.200	−1.0773	2118.000	
		M_2	−3930.000	1037.526	−4.3281		673.000
2.1	13.5	M_1	−9638.000	3974.732	−1.0773	2118.000	
		M_2	−2028.000	1069.094	−2.7284		406.000
2.1	30	M_1	−9638.000	4366.668	−1.0773	2118.000	
		M_2	−981.000	1449.467	−2.3947		185.000
0.84	15	M_1	−5287.000	2029.700	−1.0049	840.000	
		M_2	−1555.000	877.973	−2.6990		250.000
6.5 (TSPM)	5.1	M_1	−16256.000	6178.000	−1.0000	6502.400	
		M_2	−5150.974	1851.280	−2.6946		1714.500
1.5	13	M_1	−5975.000	2475.680	−1.0049	1540.000	
		M_2	−1208.000	877.973	−1.8970		275.000
2.12 (INAOE[a])	11.9	M_1	−11340.000	4463.260	−1.0274	2118.000	
		M_2	−3114.154	900.063	−2.7747		330.000

[a]The INAOE 2.12 m telescope is not part of the OAN-SPM, however, it is in Mexican territory and part of the Mexican astronomical community.

8. OAN-SPM TELESCOPES

We calculated the ZCP for every one of the classical telescopes of the OAN-SPM, using the analytical expression, together with the numerical results obtained with our algorithm for comparison. Additionally, we have included the 2.12 m telescope of the Observatorio Astrofísico Guillermo Haro located in Cananea, Sonora, México. In order to perform this calculation, we need the constructional parameters of the telescopes, which are presented in Table 2, where the SAINT-EX and TAOS-II telescopes have been omitted, because these parameters are not yet public.

The results of the ZCP are presented in Table 3, where the first column list the respective telescope (the 2.1 m telescope of the OAN-SPM can be coupled to three different secondary mirrors), the second column contains the focal ratio of each telescope, the third column displays the calculation of the ZCP using the analytical equation, while the fourth column contains the calculation of the ZCP using our algorithm. The last column shows the percentage error between the neutral point equations and the numerical value calculated from ray tracing.

The most relevant result shown in Table 3 is the fact that the value calculated using the classical and the numerical solution are very different for telescopes with aspherical surfaces, as can be seen in the case of the TAOS-II telescope. This is to be expected because the classical solution does not take asphericity terms into account. Ignoring the asphericity terms undoubtedly

TABLE 3

ZERO COMA POINT FOR OAN-SPM TELESCOPES

Telescope (m)	Focal ratio $f/\#$	Classical solution (mm)	Exact ray tracing (mm)	Percentage error (%)
2.1	7.5	1188.06	1186.32	0.15
2.1	13.5	688.38	686.68	0.25
2.1	30.5	321.75	320.10	0.51
1.5	13	504.46	504.25	0.04
0.84	15	563.37	563.51	0.02
1.3 (TAOS-2)	4	591.00[a]	380.64	35.59
1.0 (SAINT-EX)	7.8	535.87	538.54	0.50
6.5 (TSPM)	5.1	1950.15	1953.58	0.18
2.12 (INAOE)	11.9	1130.17	1129.78	0.35

[a]Classical solution that is actually not applicable to the TAOS-II telescopes.

produces a systematic error during collimation.

Three of the telescopes shown in the Table 3 (1.5 m, SAINT-EX (Sabin et al. 2018) and TAOS-II) have secondary mirrors with an electromechanical positioning system that allows all degrees of freedom; however, the movement of the mirror may not occur with respect to the vertex of the secondary mirror or the ZCP. To take advantage of these mechanisms, we must study the vector components of the movements in order to obtain a geometric operator that allows us to perform a combination of steps that, together, are equivalent to a movement pivoting at the ZCP.

9. CONCLUSIONS

We presented the SoS-ZCP general algorithm that performs the exact ray tracing computation for surfaces defined by a sagitta function, to calculate the ZCP, even though the telescope may have aspheric surfaces. The algorithm has been implemented to calculate the ZCP numerically, which is very relevant, especially in the case of non-classical telescopes, such as those of the TAOS-II project, which will display a secondary mirror with an asphericity term. In these circumstances, the classical solution is unable to produce a correct result, as expected.

We applied the calculation of the ZCP using the exact ray tracing for every telescope of the OAN-SPM, with which we are involved, and the results have been reported in this document. This information is relevant for the OAN-SPM technical staff, given that the Observatory is actively working on integrating real-time image quality metrology by means of wavefront sensing for a deterministic collimation process. After the determination of the ZCP, the next step is to evaluate the astigmatism over the field and find a solution to correct it in the optical axis.

The work presented in this article could have been carried out with commercial software, however, many tasks do not merit the acquisition of a license. There are other open source packages, but we decided to use our own implementation because, in the future, we will consider ray tracing through surfaces with arbitrary shapes.

The SoS-ZCP algorithm has been released for other users to take advantage of it. The exact ray tracing might seem complex, but the algorithm was written in the simplest way possible, thus allowing every section of the code to easily be reused in other academic problems, where a sophisticated optical design program can be avoided.

REFERENCES

Bowen, I. S. & Vaughan, A. H. 1973, Appl. Opt., 12, 1430 [LINK]

Elazhary, T. T., Zhou, P., Zhao, C., & Burge, J. H. 2015, Appl. Opt., 54, 5037 [LINK]

Luna, E., Salas, L., Gutiérrez, L., & nez, J. M. N. 2007, Appl. Opt., 46, 3439 [LINK]

Malacara-Hernández, D. & Malacara-Hernández, Z. 2013, Handbook of Optical Design, Third Edition, Optical science and engineering (Taylor & Francis) [LINK]

McLeod, B. A. 1996, Publications of the Astronomical Society of the Pacific, 108, 217 [LINK]

Melsheimer, F. M. & MacFarlane, M. J. 2000, in Optical Design, Materials, Fabrication, and Maintenance, ed. P. Dierickx, Vol. 4003, International Society for Optics and Photonics (SPIE), 456 – 463 [LINK]

Nájera, M. R., Herrera, J., & Guerrero, C. A. 2021, MNajeraR/SoS-ZCP: SoS-ZCP [LINK]

Sabin, L., Gómez Maqueo Chew, Y., Demory, B.-O., Petrucci, R., & Consortium, S.-E. 2018, SAINT-EX: Searching for exoplanets from the San Pedro Mártir Observatory, This work is supported by UNAM-PAPIIT grant IN-107518 [LINK]

Schechter, P. L. & Levinson, R. S. 2011, Publications of the Astronomical Society of the Pacific, 123, 812–832 [LINK]

Schroeder, D. & Inc, E. I. 2000, Astronomical Optics, Electronics & Electrical (Elsevier Science) [LINK]

Smith, W. 2000, Modern Optical Engineering: The Design of Optical Systems, Optical Engineering Series (McGraw Hill) [LINK]

Spencer, G. H. & Murty, M. V. R. K. 1962, J. Opt. Soc. Am., 52, 672 [LINK]

Wetherell, W. B. & Rimmer, M. P. 1972, Appl. Opt., 11, 2817 [LINK]

Joel Herrera, Morgan R. Nájera and Carlos A. Guerrero: Universidad Nacional Autónoma de México, Instituto de Astronomía, AP 106, Ensenada 22860, B.C., México (joel,mnajera,cguerrero@astro.unam.mx).

Capítulo 7

Conclusiones

En esta tesis presentamos el algoritmo general SoS-ZCP que efectúa el cálculo del trazo de rayos exacto para superficies definidas por una función sagita, para calcular el punto de coma cero, aunque el telescopio contenga superficies asféricas. El algoritmo ha sido implementado para calcular el punto de coma cero numéricamente, lo cual resulta sumamente relevante especialmente para el caso de los telescopios no clásicos, como es el caso de aquellos que pertenecen al proyecto TAOS-II, que dispondrán de un espejo secundario con términos de asfericidad. En estas circunstancias, la solución clásica resulta inadecuada para producir resultados correctos, cuando se utiliza el punto de cerocoma calculado con la solución analítica (ver Figura 5.2 parte superior), donde estamos moviendo el espejo secundario y aparece coma nuevamente, a diferencia del punto neutro calculado con el trazo de rayos exacto que, aunque se esté pivoteando con tresdiferentes ángulos, no hay introducción de coma, sino únicamente un aumento lineal en el astigmatismo (Figura 5.2 parte inferior).

En la Figura 5.5 de la Sección 5.2 se aprecia una pequeña contribución de aberración esférica, su presencia se debe al diseño del espejo primario, porque no se ha corregido completamente. Las aberraciones presentes por la desalineación de los elementos ópti-

cos son solamente coma y astigmatismo, donde la proporción de astigmatismo inducido depende del diseño particular, que debe ser considerado en cada caso.

Realizamos este cálculo del ZCP para para los telescopios de 0.84 m, 1.5 m, 2.1 m, 1.3 m (TAOS-2), 1.0 m (SAINT-EX) y 6.5 m (TSPM) pertenecientes al OAN-SPM, además de incluir el telescopio de 2.12 m del Observatorio Astrofísico Guillermo Haro, con los que estamos involucrados. Los puntos de coma cero han sido reportados en la presente tesis. Esta información resulta relevante para el equipo técnico del OAN-SPM, dado que en el Observatorio se encuentran trabajando activamente en la integración de metrología de calidad de imagen en tiempo real mediante la detección del frente de onda para un proceso de colimación determinista. El siguiente paso será evaluar el astigmatismo sobre todo el campo y encontrar una solución para corregirlo en el eje óptico. Es importante mencionar que en un sistema desalineado, el astigmatismo es asimétrico y lineal en el campo; esto se suma al astigmatismo natural debido al diseño óptico o al astigmatismo producido por la forma de los elementos, ya sea por construcción o por errores en el soporte de los componentes ópticos.

Normalmente, la aberración de astigmatismo no es un problema en los telescopios casi colimados, donde la coma domina la calidad de imagen. En relación con esto, puede ser suficiente corregir la coma mediante el desplazamiento lateral o la inclinación del espejo secundario. Sin embargo, nuestro principal interés en este trabajo es la colimación de telescopios de campo amplio, para los cuales el astigmatismo en los bordes del campo amplio será más relevante que en los telescopios de campo pequeño.

El trabajo presentado se pudo haber realizado con el uso de un software comercial, sin embargo, muchas tareas no requieren ni justifican la adquisición de una licencia. Podemos encontrar otros paquetes de código abierto, sin embargo, una de las motivaciones

para nosotros decidir utilizar nuestra propia implementación se debe a que en un futuro podemos considerar el trazo de rayos a través de superficies con formas arbitrarias.

El algoritmo SoS-ZCP se encuentra público para que todos los usuarios lo puedan aprovechar (Apéndice A). El trazo exacto de rayos puede parecer complejo, no obstante, el algoritmo fue escrito de la manera más accesible posible, permitiendo que cada sección del código sea fácilmente reutilizada en otros problemas académicos, donde los programas sofisticados de diseño óptico pueden ser evitados.

Apéndice A

Ejemplos de la implementación de SoS_ZCP.

A continuación se presentan dos ejemplos del uso del algoritmo SoS_ZCP para poder determinar el punto de coma cero, este fue desarrollado a partir de lo discutido en el Capítulo 4. Se comienza exponiendo un ejemplo para telescopios clásicos (Apéndice A.1) y finalizamos con el ejemplo para telescopios no clásicos (Apéndice A.2). El código fuente de ambos ejemplos está disponible para la comunidad astronómica https://github.com/MNajeraR/SoS-ZCP (Nájera et al., 2021), por este motivo se encuentran en inglés. La versión v2.0.0 usada en la tesis se encuentra almacenada en https://doi.org/10.5281/zenodo.4968507. Este programa ha sido probado con Python 2.7 y las librerías numpy 1.8.0 rc1 y matplotlib 1.3.1.

Table A.1. Telescope ϕ 2.1 m $f/7.5$ (2.1 from OAN-SPM)

Element	Radius (mm)	Thickness (mm)	Glass	Semi-Diameter (mm)	k	α_1	α_2	α_3
M_1	-9638.0	-3452.20	Mirror	1005.0	-1.077			
M_2	-3930	-3452.20	Mirror	336.5	-4.328			
M_1 Vertex		1037.526	Air					
Image Plane				78.4				

A.1. SoS_ZCP Classical telescopes

Developer: Najera Roa, Morgan Rhai

email: mnajera@astro.unam.mx

orcid: http://orcid.org/0000-0003-3283-0407

Telescope's parameters at table A.1.

A.1.1. General exact ray tracing

a) Parameters of the Optical Elements.

In this Section we define some parameters for the optical system, such as:

1. Number of surfaces: this is fundamental in accordance with the general exact ray tracing procedure. This procedure uses vectors to represent the rays of light, hence, we know the coordinates of the origin point $(P_{-1}(x_{-1}, y_{-1}), z_{-1})$ of each ray and its direction. Subsequently, we found the intersection point on the next surface defined by the sagitta equation (see Section c.2), which is centered on a new coordinate system positioned at a distance d; however, it is necessary to first find the intersection with the coordinate system $(P_0(x_0, y_0, z_0))$. If we already know the intersection points with the i-th surface, the procedure is the same to get the intersection points of the $i - th + 1$ surface.

2. Radius of curvature of the different surfaces that we are going to use in the sagitta equation.

3. Conic constant for the surfaces: this constant, as well as the radius of curvature, is going to be used in the sagitta equation.

4. Semi-diameters of the primary and the secondary mirrors to define the set of the rays that we will intersect with the primary mirror. Furthermore, given its position, the semi-diameter of the secondary mirror defines a surface that is blocking some rays of our original set of rays.

5. Distance between surfaces: as we mentioned, every surface is positioned at a distance d from each other.

6. Asphericity polynomial terms: a non-classical telescope is composed of conical surfaces (as the sagitta equation) plus asphericity polynomial terms; these terms allow one or more aspherical surface or correcting lenses to be contemplated in the optical system. For classical telescopes, the asphericity polynomial terms are equal to zero.

7. Wavelength: This parameter allows us to use differents wavelengths according to our requirements.

8. Refractive index: In the case of classical telescopes the refractive index is specified as 1 and -1,because we are working only with mirrors. However, we can change the refractive index if we have correcting aspherical lenses. To define the refractive index, you have to set the dispersion coefficients.

a) Parameters of the optical elements and the rays.

```
import numpy as np
```

```
import matplotlib.pyplot as plt


cosL = np.zeros(10) * 0.0
cosM = np.zeros(10) * 0.0
cosN = np.zeros(10) * 0.0
coorx_0 = np.ones(10) * 1.0
coory_0 = np.ones(10) * 1.0
coorz_0 = np.ones(10) * 1.0
coorx_1 = np.ones(30) * 1.0
coory_1 = np.ones(30) * 1.0
coorz_1 = np.ones(30) * 1.0
Rc = np.zeros(10)
kk = np.zeros(10)
'''
1a.- Number of surfaces of the optical system
'''
number_of_surfaces = 3
'''
2a.- Radius of curvature of the different surfaces
'''
Rc[0] = -9638.0
Rc[1] = -3.93E+003
Rc[2] = 9999999999999999.
'''
3a.- Conic constant for the surfaces
'''
kk[0] = -1.077310000000000E+000
kk[1] = -4.328100000000000E+000
kk[2] = 0.0
'''
4a.- Semi-diameters of the primary mirror and the secondary mirror
'''
semidiameter_pm = 1.0500E+003
semidiameter_sm = 3.3650E+002
```

```
'''
5a.- Distance between surfaces
'''
d = np.zeros(10) * 0.0
d[0] = 0.0
d[1] = -3452.20
d[2] =  3452.20 + 1037.525880125084
'''
6a.- Asphericity polynomial terms
'''
aalpha_1 = np.zeros(10) * 0.0
aalpha_2 = np.zeros(10) * 0.0
aalpha_3 = np.zeros(10) * 0.0
aalpha_4 = np.zeros(10) * 0.0
aalpha_5 = np.zeros(10) * 0.0
'''
7a.- Wavelength
'''
lmd = 0.4
'''
8a.- Refractive index
'''
nn0 = np.ones(10) * 1.0
nn1 = np.ones(10) * -1.0
'''
Coefficients for an specific glass: Silica SCHOOTT
'''
K_1 = 6.69422575E-001
L_1 = 4.48011239E-003
K_2 = 4.34583937E-001
L_2 = 1.32847049E-002
K_3 = 8.71694723E-001
L_3 = 9.53414824E+001
'''
```

```
Dispersion formula:
In this special case, we use the Sellmeier formula for an specific
wavelength
'''
nn1[2] = np.sqrt(1+(K_1*lmd**2)/(lmd**2-L_1)+(K_2*lmd**2)/(lmd**2
-L_2) +(K_3*lmd**2)/(lmd**2-L_3))
nn0[3] = np.sqrt(1+(K_1*lmd**2)/(lmd**2-L_1)+(K_2*lmd**2)/(lmd**2
-L_2) +(K_3*lmd**2)/(lmd**2-L_3))
nn1[3] = 1.0
print('This process may run slower depending on your hardware,
please wait...')
```

b) Ray parameters.

Here we are going to focus on defining two sets of rays, where each one will contain two subsets. One subset will define the surfaces and the second subset the grid-mesh. We have two kinds of mesh, the first is an hexapolar mesh and the other is a cartesian circular mesh.

We are going to briefly introduce some important variables that are used in this Section. First, the variable called *option*, allows us to choose what kind of subset we want, if *option* is equal to 0, then we are going to work with the hexapolar mesh. On the other hand, if *option* is equal to 1 it will use the circular mesh. NR and NR_1 set the number of elements that we want to use in our sets. LL and LL_1 define the range of the points, for this reason, we chose a *semidiameter_pm* (semidiameter of the primary mirror) as the maximum value. For the hexapolar mesh we have *num_i*, which indicates the number of multiple points of that parameter.

Moreover, *rad* and R define the radius of the surface which, in the same way as before, the values of this two variables are going to be set between *semidiameter_pm* and *semidiameter_sm*. In order to calculate the coordinates of the origin point of each ray and its direction, the variables *meshcoorx*, *meshcoory*, *systz*, *systL*, *systM* and

$systN$ are used to define the value of $(P_{-1}(x_{-1}, y_{-1}, z_{-1}))$. Finally, the two subsets will be saved in $assemblagesurf\_surf$ and $assemblageray\_ray$.

```
def setof_rayos1(option):
  option = option
  if option == 0:
    '''
    Parameters for surfaces
    '''
    NR_1 = 100.0
    LL_1 = semidiameter_pm
    systx1 = np.linspace(-LL_1, LL_1, num=NR_1)
    systy1 = np.random.randint(-LL_1,LL_1,NR_1)*0.0
    systz1 = np.ones(NR_1) * -100.0
    systL1 = np.zeros(NR_1)
    systM1 = np.zeros(NR_1)
    systN1 = np.ones(NR_1)
    assemblagesurf_surf = [ ]
    assemblagesurf_surf.append(systx1)
    assemblagesurf_surf.append(systy1)
    assemblagesurf_surf.append(systz1)
    assemblagesurf_surf.append(systL1)
    assemblagesurf_surf.append(systM1)
    assemblagesurf_surf.append(systN1)
    '''
    Parameters for rays
    '''
    num_0 = 10.0
    LL = 1.0500E+003
    rad = []
    NR = 10.0
    systx = []
    systy = []
```

```
    NR = []
    num_i = 6.0
    div_0 = LL/num_0
    for i in range(11):
        rad.append(LL-i*div_0)
        NR.append(num_i * i)
    NR = np.array(NR)
    rad = np.array(rad)
    rad = -(rad-1.0500E+003)
    rad[(rad<=3.3650E+002)] = np.NaN
    rad = rad[~np.isnan(rad)]
    numrad = len(rad)
    numrad1 = len(NR) - len(rad)
    for i in range(numrad1):
        NR = np.delete(NR,abs(i-i))
    for i in range(numrad):
        num_segmentos = int(NR[i])
        angle = np.linspace(0, 2*np.pi, num_segmentos+1)
        systx.append(rad[i] * np.cos(angle))
        systy.append(rad[i] * np.sin(angle))
    meshcoorx = []
    meshcoory = []
for i in range(numrad):
        numsum = int(NR[i]+1)
        for j in range(numsum):
            meshcoorx.append(systx[i][j])
            meshcoory.append(systy[i][j])
    meshcoorx = np.array(meshcoorx)
    meshcoory = np.array(meshcoory)
    NR_1 = len(meshcoorx)
    systz = np.ones(NR_1) * -100.0
    systL = np.zeros(NR_1)
    systM = np.zeros(NR_1)
    systN = np.ones(NR_1)
```

```
    assemblageray_ray = []
    assemblageray_ray.append(meshcoorx)
    assemblageray_ray.append(meshcoory)
    assemblageray_ray.append(systz)
    assemblageray_ray.append(systL)
    assemblageray_ray.append(systM)
    assemblageray_ray.append(systN)
  else:
    '''
    Parameters for surfaces
    '''
    NR_1 = 100.0
    LL_1 = semidiameter_pm
    systx1 = np.linspace(-LL_1, LL_1, num=NR_1)
    systy1 = np.random.randint(-LL_1,LL_1,NR_1)*0.0
    systz1 = np.ones(NR_1) * -100
    systL1 = np.zeros(NR_1)
    systM1 = np.zeros(NR_1)
    systN1 = np.ones(NR_1)
    assemblagesurf_surf = [ ]
    assemblagesurf_surf.append(systx1)
    assemblagesurf_surf.append(systy1)
    assemblagesurf_surf.append(systz1)
    assemblagesurf_surf.append(systL1)
    assemblagesurf_surf.append(systM1)
    assemblagesurf_surf.append(systN1)
    '''
   Parameters for rays
    '''
    NR = 45.0
    LL = 2.0 * semidiameter_pm
    R = []
    teta = []
    systx = np.linspace(LL, -LL, num=NR)
```

```
    systy = np.linspace(LL, -LL, num=NR)
    systxx, systyy = np.meshgrid(systx, systy)
    R = np.sqrt(systxx**2 + systyy**2)
    R[(R<=semidiameter_sm)] = np.NaN
    R[(R>=semidiameter_pm )] = np.NaN
    cos = systxx/R
    sin = systyy/R
    theta = np.where( sin >= 0. , np.arccos(cos) , -np.arccos(cos) )
    theta = np.where( R == 0. , 0., theta)
    systxx = R*np.cos(theta)
    systyy = R*np.sin(theta)
    meshcoorx = []
    meshcoory = []
    for i in range(NR):
      for j in range(NR):
        meshcoorx.append(systxx[i][j])
        meshcoory.append(systyy[i][j])
    meshcoorx = np.array(meshcoorx)
    meshcoory = np.array(meshcoory)
    meshcoorx = meshcoorx[~np.isnan(meshcoorx)]
    meshcoory = meshcoory[~np.isnan(meshcoory)]
    NR_1 = len(meshcoorx)
    systz = np.ones(NR_1) * -100.0
    systL = np.zeros(NR_1)
    systM = np.zeros(NR_1)
    systN = np.ones(NR_1)
    assemblageray_ray = []
    assemblageray_ray.append(meshcoorx)
    assemblageray_ray.append(meshcoory)
    assemblageray_ray.append(systz)
    assemblageray_ray.append(systL)
    assemblageray_ray.append(systM)
    assemblageray_ray.append(systN)
  return (assemblagesurf_surf, assemblageray_ray)
```

c) Find the intersection of the ray in the next coordinate system.

As we mentioned in Section a), the main goal in general exact ray tracing is to find the intersection of the ray in every coordinate system, because it determines the optical path of the ray, passing through a set of surfaces. As a first step, we define a function that will help us to find the intersection point. Hereafter this Section is divided into Subsections, because these subsections contain the functions that we will use. We discuss a brief explanation of each function below.

c.1) Coordinates of the P_0 function.

Before we find the intersection point P_1, which is the intersection point on a surface, we need to find the origin point P_0. To find this point, we use equation $x_1 = (L/N) * (z_1 - z_0) + x_0$ and $y_1 = (M/N) * (z_1 - z_0) + y_0$. Then, we iteratively solve the ray-surface function.

```
def F_RS(z_1,rayorigin,i):
    x_0 = rayorigin[0]
    y_0 = rayorigin[1]
    z_0 = rayorigin[2]
    L = rayorigin[3]
    M = rayorigin[4]
    N = rayorigin[5]
    x_1 = (L/N)*(z_1-z_0)+x_0
    y_1 = (M/N)*(z_1-z_0)+y_0
    f_RS = surface(x_1,y_1,z_1,i)
    return f_RS
```

c.2) Ray-Surface function.

After finding the coordinates x_1 and y_1, we iteratively solve the ray-surface fun-

ction. This function contains the sagitta function which defines a surface.

```
def surface(x_1,y_1,z_1,i):
    R_c = Rc[i]
    k = kk[i]
    Alpha_1 = aalpha_1[i]
    Alpha_2 = aalpha_2[i]
    Alpha_3 = aalpha_3[i]
    Alpha_4 = aalpha_4[i]
    Alpha_5 = aalpha_5[i]
    c = 1.0/R_c
    s2 = (x_1*x_1)+(y_1*y_1)
    s4 = s2*s2
    s6 = s4*s2
    s8 = s6*s2
    s10 = s8*s2
    f1 = c*s2/(1.0+np.sqrt(1.0-((k+1.0)*c*c*s2)))
    f2 = (Alpha_1*s2)+(Alpha_2*s4)+(Alpha_3*s6)+
       (Alpha_4*s8)+(Alpha_5*s10)
    return f1+f2-z_1
```

c.3) Derivative of the ray-surface function.

We compute a numerical derivative of the ray-surface function, instead of performing a simple derivative, in order to calculate the intersection point.

```
def Der_F_RS(z_1,rayorigin,i):
    h = 0.00000000001
    der = (F_RS(z_1+h, rayorigin,i)-F_RS(z_1-h, rayorigin,i))/(2*h)
    return der
```

c.4) NewtonRaphson function.

To find the point P_1, we use the Newton-Rhapson method within a loop. This

method is useful for finding the roots of the function that defines the intersection with the surface (c.3).

```
def Newton_Raphson(rayorigin,i):
    z_1 = 0
    cnt = 0
    while True:
        z_2 = z_1-(F_RS(z_1,rayorigin,i)/Der_F_RS(z_1,rayorigin,i))
        if np.abs(z_2-z_1)<0.00001:
            break
        else:
            z_1=z_2
        if cnt == 30:
            break
        cnt=cnt+1
    [x_0, y_0, z_0,L,M,N]=rayorigin
    x_1 = (L/N)*(z_1-z_0)+x_0
    y_1 = (M/N)*(z_1-z_0)+y_0
    return x_1, y_1, z_1
```

c.5) Normal Vector function

Once the position of the intersection point is known, we calculate the normal vector using the numerical derivative of the ray-surface function.

```
def surface_Derivative(x,y,z,i):
    delta=0.000001
    Dx = (surface(x+delta,y,z,i)-surface(x-delta,y,z,i))/(2.0*delta)
    Dy = (surface(x,y+delta,z,i)-surface(x,y-delta,z,i))/(2.0*delta)
    Dz = (surface(x,y,z+delta,i)-surface(x,y,z-delta,i))/(2.0*delta)
    Dr = np.sqrt((Dx*Dx)+(Dy*Dy)+(Dz*Dz))
    return Dx/Dr,Dy/Dr,Dz/Dr
```

c.6) Snell's function.

When the ray hits a surface, we already know the direction cosines and the normal vector to that surface, but we don't know the direction of the ray that is refracted or reflected. To know that we have to use Snell's law in its vector form.

```
def Snell_refraction_vector(Ray_Vect,Surf_Normal,n_0,n_1):
    Ray_Vect = np.asarray(Ray_Vect)
    Surf_Normal = np.asarray(Surf_Normal)
    Nsurf_Cros_s1 = np.cross(Surf_Normal,Ray_Vect)
    NN = n_0/n_1
    R = (NN*NN)*np.dot(Nsurf_Cros_s1,Nsurf_Cros_s1)
    if R>1:
        print("----- internal reflection= ", R)
    s2 = NN*(np.cross(Surf_Normal,np.cross(-Surf_Normal,Ray_Vect)))
    -Surf_Normal*np.sqrt(1.0-((NN*NN)*np.dot(Nsurf_Cros_s1,
    Nsurf_Cros_s1)))
    return s2
```

c.7) Skew ray tracing function.

This function uses all the previous functions to calculate the point of intersection and the direction of the ray when it interacts with the surface. This function is used as many times as there are surfaces.

```
def Trax(Ray_Param,i):
    ## Point of intersection on the surface
    x_1, y_1, z_1 = Newton_Raphson(Ray_Param,i)
    ## Normal vector at the point of intersection on the surface
    NORM = surface_Derivative(x_1,y_1,z_1,i)
    [x_0, y_0, z_0,L,M,N] = Ray_Param
    Ray_Vect = [L,M,N]
    Surf_Normal = NORM
```

```
    n_0 = nn0[i]
    n_1 = nn1[i]
    lmn = Snell_refraction_vector(Ray_Vect,Surf_Normal,n_0,n_1)
    return ([x_1, y_1, z_1, lmn[0], lmn[1], lmn[2]])
```

c.6) Director Cosines function.

If we don't know the director cosines of some point, we can calculate them with this function.

```
def director_cosines(x,y,z):
  magnitude = np.sqrt(x**2+y**2+z**2)
  L = x/magnitude
  M = y/magnitude
  N = z/magnitude
  return L, M, N
```

d) Change the ray to the transformed coordinate system.

We want to perform a displacement and a rotation on the secondary mirror. For this we calculate the values of a segment of a ray that travels between two surfaces. The segment is defined by two extreme points, P_0 and P_1, both have a coordinate system (x,y,z). The shift and rotation on the secondary mirror makes the original coordinate system to become (x',y',z'). Then, we calculate the intersection point with the surface in the new system that has undergone the transformation.

The information of the point of intersection in the new system can be obtained with different functions that we will explain below.

Transformation function: The information in the transformed space is known through the translation matrix (T_{xyz}) and the rotation matrices (R_x, R_y and R_z). The first matrix contains the three displacements of the three spatial directions and the other matrices contain the angles at which the rotation around the axes was applied.

Consequently, the matrix with the resulting transformation is the product of these four matrices. Also, this matrix will involve an inverse transformation to return to the original coordinate system.

```
def Transform(TRF):
    [TiltX, TiltY, TiltZ, dx, dy, dz]=TRF
    Tx=np.deg2rad(TiltX)
    Ty=np.deg2rad(TiltY)
    Tz=np.deg2rad(TiltZ)
    Rx_1A = np.matrix([[1.0, 0.0, 0.0 , 0.0], [0.0, np.cos(-Tx),
    np.sin(-Tx),0.0],[0.0,-np.sin(-Tx), np.cos(-Tx),0.0],[0.0,
    0.0, 0.0 ,1.0]])
    Ry_1A = np.matrix([[np.cos(-Ty),0.0, -np.sin(-Ty),0.0], [0.0,
    1.0, 0.0, 0.0],[np.sin(-Ty),0.0, np.cos(-Ty),0.0],[0.0,
    0.0, 0.0 ,1.0]])
    Rz_1A = np.matrix([[np.cos(-Tz),-np.sin(-Tz), 0.0, 0.0],
    [np.sin(-Tz),np.cos(-Tz), 0.0, 0.0],[0.0, 0.0, 1.0, 0.0],
    [0, 0, 0 ,1.0]])
    Dxyz_1A = np.matrix([[1.0, 0.0, 0.0, -dx],[0.0, 1.0, 0.0,
    -dy],[0.0, 0.0, 1.0, -dz],[0.0, 0.0, 0.0 ,1.0]])
    Start_trans1=Rz_1A*Ry_1A*Rx_1A*Dxyz_1A
    Start_trans2=np.linalg.inv(Start_trans1)
    return Start_trans1, Start_trans2
```

Operations in the transformed space system.

```
def Traceray(inx,iny,inz,inL,inM,inN, MT):
    coorx_0[0] = inx
    coory_0[0] = iny
    coorz_0[0] = inz
    cosL[0] = inL
    cosM[0] = inM
    cosN[0] = inN
```

```
    for i in range (number_of_surfaces):
        '''
        A1_Trans is the transformation matrix to the transformed
        space and A2_Trans contains the inverse transformation.
        '''
        A1_Trans = MT[i][0]
        A2_Trans = MT[i][1]
        x_0 = coorx_0[i]
        y_0 = coory_0[i]
        z_0 = coorz_0[i] - d[i]
        L = cosL[i]
        M = cosM[i]
        N = cosN[i]
        '''
        e), f) and g) We already know the points of origin and
        their direction, thus, we can calculate the second point,
        in order to find the extreme points P_0 and P_1.
        '''
        Ray_Param = [x_0, y_0, z_0, L, M, N]
        xyz_1=Trax(Ray_Param,i)
        x_1 = xyz_1[0]
        y_1 = xyz_1[1]
        z_1 = xyz_1[2]
        '''
        We transform the components of P_0 and P_1 using
        the transformation matrix as follows:
        '''
        prime_coor0 = np.dot(A1_Trans, ([x_0, y_0, z_0, 1.0]))
        prime_coor1 = np.dot(A1_Trans,([x_1, y_1, z_1, 1.0]))
        '''
        We have to calculate the director cosines in the
        transformed space because the director cosines are
        not transformed automatically.
        '''
```

```
prime_scosinD = Cosine_transform(prime_coor0, prime_coor1)
'''
Now, we know the points of origin in the transformed
space and their direction:
'''
x_0 = prime_coor0[0,0]
y_0 = prime_coor0[0,1]
z_0 = prime_coor0[0,2]
s=np.sign(d[i])
if i==0:
    s=1
L = prime_scosinD[0]*s
M = prime_scosinD[1]*s
N = prime_scosinD[2]*s
Ray_Param = [x_0, y_0, z_0, L, M, N]
xyz_1=Trax(Ray_Param,i)
x_1 = xyz_1[0]
y_1 = xyz_1[1]
z_1 = xyz_1[2]
L = xyz_1[3]
M = xyz_1[4]
N = xyz_1[5]
z_2=z_0
x_2=(L/N)*(z_2-z_1)+x_1
y_2=(M/N)*(z_2-z_1)+y_1
'''
h) At this point we know the point of intersection with
the system that underwent a displacement and a rotation.
We have to calculate two points because we need to
transform the director cosines to the original system.
j) In this way, prime_coor0 and prime_coor1 are the
intersection points in the original system.
k) Prime_scosinD are the director cosines of the ray
in the original system.
```

```
        '''
        prime_coor0 = np.dot(A2_Trans, ([x_1, y_1, z_1, 1.0]))
        prime_coor1 = np.dot(A2_Trans, ([x_2, y_2, z_2, 1.0]))
        prime_scosinD = Cosine_transform(prime_coor0,prime_coor1)
        x_1 = prime_coor0[0,0]
        y_1 = prime_coor0[0,1]
        z_1 = prime_coor0[0,2]
        L = prime_scosinD[0]
        M = prime_scosinD[1]
        N = prime_scosinD[2]
        coorx_1[i] = x_1
        coory_1[i] = y_1
        coorz_1[i] = z_1
        cosL[i+1] = L
        cosM[i+1] = M
        cosN[i+1] = N
        coorx_0[i+1] = coorx_1[i]
        coory_0[i+1] = coory_1[i]
        coorz_0[i+1] = coorz_1[i]
    return coorx_1,coory_1, coorz_1, cosL, cosM, cosN
```

Cosine transformation function: As we mention, we have to calculate the director cosines of the ray using two points, one in each systems. This function allows us to do it.

```
def Cosine_transform(system_cos0,system_cos1):
    x_0 = system_cos0[0,0]
    y_0 = system_cos0[0,1]
    z_0 = system_cos0[0,2]
    x_1 = system_cos1[0,0]
    y_1 = system_cos1[0,1]
    z_1 = system_cos1[0,2]
    magnitude_transform = np.sqrt((x_1-x_0)**2+(y_1-y_0)**2
    +(z_1-z_0)**2)
```

```
    pm_L = (x_1-x_0)/magnitude_transform
    pm_M = (y_1-y_0)/magnitude_transform
    pm_N = (z_1-z_0)/magnitude_transform
    return pm_L, pm_M, pm_N
```

n) Ray information through the system: All the information up to this point will be stored in two variables, each of which will contain ray and surface information.

We defined two functions (n.1 and n.2), the first function calls all the previous functions that we define and it carries out the exact ray tracing. Moreover, this function will store all the information that it generates. The second function specifies the displacement and rotation of all the surfaces that will undergo a change and calculates the two transformation matrices. Finally, we use the (n.1) function to compute the parameters of the rays and the surface.

```
'''
n.1 function.
'''
def Rays_infomation(ray_param,Matrix):
  initial_coor0 = ray_param[0]
  initial_coor1 = ray_param[1]
  initial_coor2 = ray_param[2]
  director_cos0 = ray_param[3]
  director_cos1 = ray_param[4]
  director_cos2 = ray_param[5]
  NR2 = len(initial_coor0)
  systcoorx_3 = np.ones(NR2)
  systcoory_3 = np.ones(NR2)
  systcoorz_3 = np.ones(NR2)
  systcosL_3 = np.ones(NR2)
  systcosM_3 = np.ones(NR2)
  systcosN_3 = np.ones(NR2)
```

```
systcoorx_2 = np.ones(NR2)
systcoory_2 = np.ones(NR2)
systcoorz_2 = np.ones(NR2)
systcosL_2 = np.ones(NR2)
systcosM_2 = np.ones(NR2)
systcosN_2 = np.ones(NR2)
systcoorx_1 = np.ones(NR2)
systcoory_1 = np.ones(NR2)
systcoorz_1 = np.ones(NR2)
systcosL_1 = np.ones(NR2)
systcosM_1 = np.ones(NR2)
systcosN_1 = np.ones(NR2)
planoimagen = np.ones(NR2)
for i in range(NR2):
    inx = initial_coor0[i]
    iny = initial_coor1[i]
    inz = initial_coor2[i]
    inL = director_cos0[i]
    inM = director_cos1[i]
    inN = director_cos2[i]
    cx_1, cy_1, cz_1, cL, cM, cN = Traceray(inx,iny,inz,inL,
    inM, inN, Matrix)
    systcoorx_3[i] = cx_1[2]
    systcoory_3[i] = cy_1[2]
    systcoorz_3[i] = cz_1[2]
    systcosL_3[i] = cL[3]
    systcosM_3[i] = cM[3]
    systcosN_3[i] = cN[3]
    systcoorx_2[i] = cx_1[1]
    systcoory_2[i] = cy_1[1]
    systcoorz_2[i] = cz_1[1]
    systcosL_2[i] = cL[2]
    systcosM_2[i] = cM[2]
    systcosN_2[i] = cN[2]
```

```
        systcoorx_1[i] = cx_1[0]
        systcoory_1[i] = cy_1[0]
        systcoorz_1[i] = cz_1[0]
        systcosL_1[i] = cL[1]
        systcosM_1[i] = cM[1]
        systcosN_1[i] = cN[1]
    Object_plane = (initial_coor0, initial_coor1, initial_coor2,
    director_cos0, director_cos1, director_cos2)
    Primary_mirror = (systcoorx_1, systcoory_1, systcoorz_1,
    systcosL_1, systcosM_1, systcosN_1)
    Secondary_mirror = (systcoorx_2, systcoory_2, systcoorz_2,
    systcosL_2, systcosM_2, systcosN_2)
    Image_plane = (systcoorx_3, systcoory_3, systcoorz_3,
    systcosL_3, systcosM_3, systcosN_3)
    surf_opt = (Primary_mirror, Secondary_mirror)
    plane_opt = (Object_plane,  Image_plane)
    return surf_opt, plane_opt
'''
n.2 function
'''
def TiltandShift_transtormation(tilt_smirror, shift_smirror,
set_rayos):
    tiltthetax = tilt_smirror[0]
    tiltthetay = tilt_smirror[1]
    shiftx = shift_smirror[0]
    shifty = shift_smirror[1]
    MT=[]
    ma_rt_s0 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
    ma_rt_s1 = [tiltthetax , tiltthetay, 0.0, shiftx, shifty,
    0.0]
    ma_rt_s2 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
    ma_rt_s3 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
    ma_rt_s4 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
    ma_rt_s5 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
ma_rt_s6 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
ma_rt_s7 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
M = Transform(ma_rt_s0)
MT.append(M)
M = Transform(ma_rt_s1)
MT.append(M)
M = Transform(ma_rt_s2)
MT.append(M)
M = Transform(ma_rt_s3)
MT.append(M)
M = Transform(ma_rt_s4)
MT.append(M)
surf_transformate, plane_transformate =
Rays_infomation(set_rayos,MT)
return surf_transformate, plane_transformate
```

A.1.2. Coma evaluation function

To evaluate and eliminate the presence of axial coma we use the transverse coma aberration, which is defined as the change in amplification as a function of the aperture. To achieve this, we have to use three rays that arrive on a meridional plane, two of them are marginal rays passing through the edges of the pupil of the system and the last one is a principal ray.

From A.1.1, we calculate the director cosines of these three rays and the intersection point in the last surface. In this way, we can find the intersection point between the two marginal rays and we can calculate the distance from this point to the optical axis. Furthermore, with the intersection between the marginal rays and the distance from this intersection to the optical axis, we determine the position of the perpendicular plane that generates the intersection point and, finally, we can calculate the intersection of the principal ray with the perpendicular plane, and its distance from the optical axis.

We need three rays for the coma evaluation function, thus, we define a set of three rays. Two of them are marginal rays passing through the edges of the pupil of the system. However, as the pupil of the system is the primary mirror, both rays pass through *semidiameter_pm* and *semidiameter_pm*. The last one is a principal ray that passes through the center of the pupil.

```
def setof_rayos(option):
  option = option
  if option == 0:
    '''
    Parameters for surfaces
    '''
    NR_1 = 100
    LL_1 = semidiameter_pm
    systx1 = np.linspace(-LL_1, LL_1, num=NR_1)
    systy1 = np.random.randint(-LL_1,LL_1,NR_1)*0.0
    systz1 = np.ones(NR_1) * -100.0
    systL1 = np.zeros(NR_1)
    systM1 = np.zeros(NR_1)
    systN1 = np.ones(NR_1)
    assemblagesurf_surf = []
    assemblagesurf_surf.append(systx1)
    assemblagesurf_surf.append(systy1)
    assemblagesurf_surf.append(systz1)
    assemblagesurf_surf.append(systL1)
    assemblagesurf_surf.append(systM1)
    assemblagesurf_surf.append(systN1)
    '''
    Parameters for rays
    '''
    meshcoorx = ( 0.0, 0.0, 0.0)
    meshcoory = ( 0.0, semidiameter_pm , -semidiameter_pm )
    meshcoorx = np.array(meshcoorx)
```

```
    meshcoory = np.array(meshcoory)
    NR_1 = len(meshcoorx)
    systz = np.ones(NR_1) * -100
    systL = np.zeros(NR_1)
    systM = np.zeros(NR_1)
    systN = np.ones(NR_1)
    assemblageray_ray = []
    assemblageray_ray.append(meshcoorx)
    assemblageray_ray.append(meshcoory)
    assemblageray_ray.append(systz)
    assemblageray_ray.append(systL)
    assemblageray_ray.append(systM)
    assemblageray_ray.append(systN)
  return (assemblagesurf_surf, assemblageray_ray)
```

The skew ray tracing gives us the information necessary to calculate the intersection between the marginal rays, the distance of this intersection to the optical axis, the position of the perpendicular plane that generates the intersection point, the intersection of the principal ray with the perpendicular plane and its distance from the optical axis. For this, we define the next function.

```
def Height_yp(Set_lsurface):
    last_surface = Set_lsurface
    valuex_LS = last_surface[0]
    valuey_LS = last_surface[1]
    valuez_LS = last_surface[2]
    valueL_LS = last_surface[3]
    valueM_LS = last_surface[4]
    valueN_LS = last_surface[5]
    yp0 = valuey_LS[0]
    yma = valuey_LS[1]
    ymb = valuey_LS[2]
```

```
    zp0 = valuez_LS[0]
    zma = valuez_LS[1]
    zmb = valuez_LS[2]
    Mp0 = valueM_LS[0]
    Mma = valueM_LS[1]
    Mmb = valueM_LS[2]
    Np0 = valueN_LS[0]
    Nma = valueN_LS[1]
    Nmb = valueN_LS[2]
    ym = ((Mma*Mmb)/(Nma*Mmb-Nmb*Mma))*(zmb-zma-ymb*(Nmb/Mmb)+
    yma*(Nma/Mma))
    zm = (ym-ymb)*(Nmb/Mmb)+zmb
    yp = yp0+(Mp0/Np0)*(zm-zp0)
    dt_yp = np.abs(yp-ym)
    return dt_yp
```

A.1.3. Zero coma point function

We use the Newton-Raphson method in an optimization way, this allows us to calculate a tilt in "y"that minimizes the amount of coma, by setting a criterion near to zero, for a certain displacement. The projection of the resulting compensation is calculated using trigonometric properties.

The proposed displacement is 1 mm in the y-axis.

```
shift_smirror = (0.0, 1.0)
tilt_smirror = (0.0, 0.0)
opc = 0

surf_rset , ray_rset = setof_rayos(0)
surf_surf, surf_plane = TiltandShift_transtormation(tilt_smirror,
shift_smirror, ray_rset)
```

```
ray_surf, ray_plane = TiltandShift_transtormation(tilt_smirror,
shift_smirror, ray_rset)
def Der_highym(shift_set, set_rayos):
    shift_set = np.array(shift_set)
    tilt_set = (0.0, 0.0)
    h = 0.000000008
    shift_ph = shift_set + h
    shift_nh = shift_set - h
    surf_shiftph, plane_shiftph = TiltandShift_transtormation(
    tilt_set,shift_ph,set_rayos)
    surf_shiftnh, plane_shiftnh = TiltandShift_transtormation(
    tilt_set,shift_nh,set_rayos)
    l_surfaceph = surf_shiftph[1]
    l_surfacenh = surf_shiftnh[1]
    dt_ymph = Height_yp(l_surfaceph)
    dt_ymnh= Height_yp(l_surfacenh)
    derivate_ym = (dt_ymph-dt_ymnh)/(2*h)
    return derivate_ym
def NewtonRaphson_tilt(shift_array, set_rayos):
    tilt_array = (0.0,0.0)
    shift_array = np.array(shift_array)
    tilt_array = np.array(tilt_array)
    numcnt = 50000
    tilt_0 = 0.0
    cnt = 0
    while True:
      surf_NRprocedure, plane_NRprocedure =
      TiltandShift_transtormation(tilt_array,shift_array,set_rayos)
      final_surface = surf_NRprocedure [1]
      delta_ym = Height_yp(final_surface)
      Derivate_ym = Der_highym(shift_array, set_rayos)
      tilt_array[0] = tilt_0 - delta_ym/Derivate_ym
      if delta_ym < 1.0E-5:
          break
```

```
    else:
        tilt_0 = tilt_array[0]
    if cnt == numcnt:
      break
    cnt = cnt + 1
  Tx = np.deg2rad(tilt_array[0])
  pointzcoma = np.abs(shift_array[1]/np.tan(Tx))
  return tilt_array, pointzcoma
```

Here we calculate the value of the angle that minimizes the amount of coma and its respective zero coma point.

```
tilt_tocorrect, pointncoma = NewtonRaphson_tilt(shift_smirror,
ray_rset)
```

With the value of the angle and the skew ray tracing, we obtain information of the compensated system.

```
surf_correctedset , ray_correctedset = setof_rayos1(opc)
ray_shiftysurf, ray_shiftyplane = TiltandShift_transtormation(
tilt_smirror, shift_smirror, ray_correctedset)
surf_shiftysurf, surf_shiftyplane = TiltandShift_transtormation(
tilt_smirror, shift_smirror, surf_correctedset)
ray_correctedsurf, ray_correctedplane = TiltandShift_transtormation(
tilt_tocorrect,shift_smirror, ray_correctedset)
surf_correctedsurf, surf_correctedplane =
TiltandShift_transtormation(tilt_tocorrect,shift_smirror,
surf_correctedset)
```

A.1.4. Plotting and results

In this last Section, we provide the angle (°) and the zero coma point (mm). Furthermore, five plots are shown which set out the optic system and the spot diagram with the system's displacement and the compensated system. The last plot shows the mesh of rays that we used.

Given that we have two sets ($assemblagesurf\_surf$ and $assemblageray\_ray$) which have six subsets, we redefine all this subsets into variables that will handle this information more easily.

```
print('Secondary displacement:',shift_smirror[1], 'mm')
print('Calculated tilt angle to correct:',tilt_tocorrect[0],
'degrees')
print('Position of the zero coma point:',pointncoma, 'mm')
print('')
print('')
```

a) Surface displacement

```
'''
Information of the rays
'''
objectshifty_ray = ray_shiftyplane[0]
objectshifty_rayx = objectshifty_ray[0]
objectshifty_rayy = objectshifty_ray[1]
objectshifty_rayz = objectshifty_ray[2]
pMirrorshifty_ray = ray_shiftysurf[0]
pMirrorshifty_rayx = pMirrorshifty_ray[0]
pMirrorshifty_rayy = pMirrorshifty_ray[1]
pMirrorshifty_rayz = pMirrorshifty_ray[2]
sMirrorshifty_ray = ray_shiftysurf[1]
```

```
sMirrorshifty_rayx = sMirrorshifty_ray[0]
sMirrorshifty_rayy = sMirrorshifty_ray[1]
sMirrorshifty_rayz = sMirrorshifty_ray[2]
Imageshifty_ray = ray_shiftyplane[1]
Imageshifty_rayx = Imageshifty_ray[0]
Imageshifty_rayy = Imageshifty_ray[1]
Imageshifty_rayz = Imageshifty_ray[2]
'''
Information of the surfaces
'''
objectshifty_surf = surf_shiftyplane[0]
objectshifty_surfx = objectshifty_surf[0]
objectshifty_surfy = objectshifty_surf[1]
objectshifty_surfz = objectshifty_surf[2]
pMirrorshifty_surf = surf_shiftysurf[0]
pMirrorshifty_surfx = pMirrorshifty_surf[0]
pMirrorshifty_surfy = pMirrorshifty_surf[1]
pMirrorshifty_surfz = pMirrorshifty_surf[2]
sMirrorshifty_surf = surf_shiftysurf[1]
sMirrorshifty_surfx = sMirrorshifty_surf[0]
sMirrorshifty_surfy = sMirrorshifty_surf[1]
sMirrorshifty_surfz = sMirrorshifty_surf[2]
Imageshifty_surf = surf_shiftyplane[1]
Imageshifty_surfx = Imageshifty_surf[0]
Imageshifty_surfy = Imageshifty_surf[1]
Imageshifty_surfz = Imageshifty_surf[2]
'''
Plot of the Optic System
'''
'''
Plot of the rays
'''
l = 243
m = 268
```

```
grs = 0.3
x = np.array([objectshifty_rayx[l:m],pMirrorshifty_rayx[l:m]])
z = np.array([objectshifty_rayz[l:m] + d[1],
pMirrorshifty_rayz[l:m]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([pMirrorshifty_rayx[l:m], sMirrorshifty_rayx[l:m]])
z = np.array([pMirrorshifty_rayz[l:m], sMirrorshifty_rayz[l:m]+
d[1]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([sMirrorshifty_rayx[l:m], Imageshifty_rayx[l:m]])
z = np.array([sMirrorshifty_rayz[l:m]+ d[1] ,
Imageshifty_rayz [l:m] + d[1] + d[2] ])
plt.plot(z, x, color = "red", markersize = grs)
'''
Plot of the surfaces
'''
grs_surf = 0.7
x = pMirrorshifty_surfx
y = pMirrorshifty_surfy
z = pMirrorshifty_surfz
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = sMirrorshifty_surfx
y = sMirrorshifty_surfy
z = sMirrorshifty_surfz + d[1]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = Imageshifty_surfx
y = Imageshifty_surfy
z = Imageshifty_surfz + d[1] + d[2]
plt.plot(z, x, colo r= 'blue', markersize = grs_surf)
plt.xlabel(r'Z [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.ylabel(r'X [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.xticks([-3452.20,-2500.00,-1100.00, 0.0, 1037.525880125084],
```

```
[r'$0.0$',r'$1500.00$',r'$2900.00$',r'3452.20',r'$4489.73$'])
plt.tick_params( axis = 'both', direction = 'out' , width = 2,
length = 5, labelsize=14)
plt.axis('equal')
plt.tight_layout()
plt.figure(1)
plt.show()
'''
Plot of the Spot Diagram
'''
print('')
print('')
plt.plot(Imageshifty_rayx, Imageshifty_rayy,'.', color = 'k',
linestyle='none')
plt.xlabel(r'Linear distance on the image plane [mm]', fontsize = 16,
fontstyle = "normal", family = 'serif')
plt.ylabel(r'Field angle [degrees]', fontsize = 16,
fontstyle = "normal", family = 'serif')
plt.xticks([0.0],[r'$0.0$'])
plt.yticks([-2.22],[r'$0.0$']
plt.tick_params( axis = 'both', direction = 'out' , width = 2,
length = 5, labelsize = 14)
plt.axis('equal')
plt.tight_layout()
plt.figure(2)
plt.show()
print('')
print('')
print('')
print('')
```

b) Plotting of the compensated system

```
'''
Information of the rays
'''
object_ray = ray_correctedplane[0]
object_rayx = object_ray[0]
object_rayy = object_ray[1]
object_rayz = object_ray[2]
pMirror_ray = ray_correctedsurf[0]
pMirror_rayx = pMirror_ray[0]
pMirror_rayy = pMirror_ray[1]
pMirror_rayz = pMirror_ray[2]
sMirror_ray = ray_correctedsurf[1]
sMirror_rayx = sMirror_ray[0]
sMirror_rayy = sMirror_ray[1]
sMirror_rayz = sMirror_ray[2]
Image_ray = ray_correctedplane[1]
Image_rayx = Image_ray[0]
Image_rayy = Image_ray[1]
Image_rayz = Image_ray[2]
'''
Information of the surfaces
'''
object_surf = surf_correctedplane[0]
object_surfx = object_surf[0]
object_surfy = object_surf[1]
object_surfz = object_surf[2]
pMirror_surf = surf_correctedsurf[0]
pMirror_surfx = pMirror_surf[0]
pMirror_surfy = pMirror_surf[1]
pMirror_surfz = pMirror_surf[2]
sMirror_surf = surf_correctedsurf[1]
sMirror_surfx = sMirror_surf[0]
sMirror_surfy = sMirror_surf[1]
sMirror_surfz = sMirror_surf[2]
```

```
Image_surf = surf_correctedplane[1]
Image_surfx = Image_surf[0]
Image_surfy = Image_surf[1]
Image_surfz = Image_surf[2]
'''
Plot of the Compensated Optic System
'''
'''
Plot of the rays
'''
l = 20
m = 300
grs = 0.3
x = np.array([object_rayx[l:m], pMirror_rayx[l:m]])
z = np.array([object_rayz[l:m] + d[1], pMirror_rayz[l:m]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([pMirror_rayx[l:m],sMirror_rayx[l:m]])
z = np.array([pMirror_rayz[l:m],sMirror_rayz[l:m]+ d[1]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([sMirror_rayx[l:m], Image_rayx[l:m]])
z = np.array([sMirror_rayz[l:m]+ d[1], Image_rayz [l:m] + d[1]
+ d[2]])
plt.plot(z, x, color = "red", markersize = grs)
'''
Plot of the surfaces
'''
grs_surf = 0.7
x = pMirror_surfx
y = pMirror_surfy
z = pMirror_surfz
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = sMirror_surfx
y = sMirror_surfy
z = sMirror_surfz + d[1]
```

```
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = Image_surfx
y = Image_surfy
z = Image_surfz + d[1] + d[2]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
plt.xlabel(r'Z [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.ylabel(r'X [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.xticks([-3452.20,-2500.00,-1100.00, 0.0,
1037.525880125084],[r'$0.0$',r'$1500.00$',r'$2900.00$',
r'3452.20',r'$4489.73$'])
plt.tick_params( axis = 'both', direction = 'out' , width=2,
length=5, labelsize=14)
plt.axis('equal')
plt.tight_layout()
plt.figure(3)
plt.show()
'''
Plot of the Compensated Spot Diagram
'''
print('')
print('')
plt.plot(Image_rayx, Image_rayy,'.', color = 'k',
linestyle = 'none')
plt.xlabel(r'Linear distance on the image plane [mm]', fontsize = 16,
fontstyle = "normal", family = 'serif')
plt.ylabel(r'Field angle [degrees]', fontsize = 16,
fontstyle = "normal", family = 'serif')
plt.xticks([0.0],[r'$0.0$'])
plt.yticks([5.2832],[r'$0.0$'])
plt.tick_params( axis = 'both', direction = 'out' , width = 2,
length = 5, labelsize = 16)
plt.axis('equal')
```

```
plt.tight_layout()
plt.figure(4)
plt.show()
```

c) Mesh of rays that hit the primary mirror

```
print('')
print('')
print('')
print('')
plt.plot(pMirror_rayx, pMirror_rayy,'.', color = 'k',
linestyle = 'none')
plt.xlabel(r'X [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.ylabel(r'Y [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.yticks([-1050,-700,-350, 0.0, 350, 700,1050],[r'$-1050$',
r'$-700$',r'$-350$',r'$0.0$',r'$350$',r'$700$',r'$1050$'])
plt.tick_params( axis = 'both', direction = 'out', width = 2,
length = 5, labelsize = 14)
plt.axis('equal')
plt.tight_layout()
plt.figure(5)
plt.show()
```

Table A.2. Telescope ϕ 2.1 m $f/3$ (Theoretical example)

Element	Radius (mm)	Thickness (mm)	Glass	Semi-Diameter (mm)
M_1	−7670.112	−227241	Mirror	1005.0
M_2	−7670.112	227241	Mirror	465.0
M_1 Vertex		56.538	Air	
Corrector Front		7.0	Silica Schott	191.0
Corrector Back		300.0	Air	191.0
Image Plane				78.4

Table A.3. Telescope ϕ 2.1 m $f/3$ (Theoretical example)

Element	k	α_1	α_2	α_3
M_1	−1.597			
M_2	−37.027			5.940e−19
M_1 Vertex				
Corrector Front		3.955e−5	−1.021e−9	
Corrector Back				
Image Plane				

A.2. SoS_ZCP Non-Classical telescopes

```
import numpy as np

import matplotlib.pyplot as plt
```

Developer: Najera Roa, Morgan Rhaí.

email: mnajera@astro.unam.mx

orcid: http://orcid.org/0000-0003-3283-0407

Telescope's parameters at table A.2 and table A.3.

A.2.1. General exact ray tracing

a) Parameters of the Optical Elements

In this Section we define some parameters for the optical system, such as:

1. Number of surfaces: this is fundamental in accordance with the general exact ray tracing procedure. This procedure uses vectors to represent the rays of light, hence, we know the coordinates of the origin point ($P_{-1}(x_{-1}, y_{-1}, z_{-1})$ of each ray and its direction. Subsequently, we found the intersection point on the next surface defined by the sagitta equation (see Section c.2), which is centered on a new coordinate system positioned at a distance d; however, it is necessary to to first find the intersection with the coordinate system ($P_0(x_0, y_0, z_0)$). If we already know the intersection points with the i-th surface, the procedure is the same to get the intersection points of the $i - th + 1$ surface.

2. Radius of curvature of the different surfaces that we are going to use in the sagitta equation.

3. Conic constant for the surfaces: this constant, as well as the radius of curvature, is going to be used in the sagitta equation.

4. Semi-diameters of the primary and the secondary mirrors to define the set of the rays that we will intersect with the primary mirror. Furthermore, given its position, the semi-diameter of the secondary mirror defines a surface that is blocking some rays of our original set of rays.

5. Distance between surfaces: as we mentioned, every surface is positioned at a distance d from each other.

6. Asphericity polynomial terms: a non-classical telescope is composed of conical surfaces (as the sagitta equation) plus asphericity polynomial terms; these terms allow one or more aspherical surface or correcting lenses to be contemplated in

the optical system. For classical telescopes, the asphericity polynomial terms are equal to zero.

7. Wavelength: This parameter allows us to use differents wavelengths according to our requirements.

8. Refractive index: In the case of classical telescopes the refractive index is specified as 1 and -1, because we are working only with mirrors. However, we can change the refractive index if we have correcting aspherical lenses. To define the refractive index, you have to set the dispersion coefficients.

```
'''
Parameters of the optical elements and the rays
'''
cosL = np.zeros(10) * 0.0
cosM = np.zeros(10) * 0.0
cosN = np.zeros(10) * 0.0
coorx_0 = np.ones(10) * 1.0
coory_0 = np.ones(10) * 1.0
coorz_0 = np.ones(10) * 1.0
coorx_1 = np.ones(30) * 1.0
coory_1 = np.ones(30) * 1.0
coorz_1 = np.ones(30) * 1.0
Rc = np.zeros(10)
kk = np.zeros(10)
'''
1a.- Number of surfaces of the optical system
'''
number_of_surfaces = 5
'''
2a.- Radius of curvature of the different surfaces
'''
Rc[0] = -7670.112
```

```
Rc[1] = -7670.112
Rc[2] = 9999999999999999.
Rc[3] = 9999999999999999.
Rc[4] = 9999999999999999
'''
3a.- Conic constant for the surfaces
'''
kk[0] = -1.597
kk[1] = -37.027
kk[2] = 0.0
kk[3] = 0.0
kk[4] = 0.0
'''
4a.- Semi-diameters of the primary mirror and
the secondary mirror
'''
semidiameter_pm = 1.071721E+003
semidiameter_sm = 4.64365E+002
'''
5a.- Distance between surfaces
'''
d = np.zeros(10) * 0.0
d[0] = 0.0
d[1] = - 2272.41
d[2] = -(- 2272.41) + 56.538
d[3] = 7.0
d[4] = 300.0
'''
6a.- Asphericity polynomial terms
'''
aalpha_1 = np.zeros(10) * 0.0
aalpha_2 = np.zeros(10) * 0.0
aalpha_3 = np.zeros(10) * 0.0
aalpha_4 = np.zeros(10) * 0.0
```

```
aalpha_5 = np.zeros(10) * 0.0
aalpha_3[1] = 5.943E-019
aalpha_1[2] = 3.955E-005
aalpha_2[2] = -1.021E-009
'''
7a.- Wavelength
'''
lmd=0.6
'''
8a.- Refractive index
'''
nn0 = np.ones(10) * 1.0
nn1 = np.ones(10) * -1.0
'''
Coefficients for an specific glass: Silica SCHOOTT
'''
K_1 = 6.69422575E-001
L_1 = 4.48011239E-003
K_2 = 4.34583937E-001
L_2 = 1.32847049E-002
K_3 = 8.71694723E-001
L_3 = 9.53414824E+001
'''
Dispersion formula:
In this special case, we use the Sellmeier formula for
an specific wavelength
'''
nn1[2] = np.sqrt(1+(K_1*lmd**2)/(lmd**2-L_1)+
(K_2*lmd**2)/(lmd**2-L_2)+(K_3*lmd**2)/(lmd**2-L_3))
nn0[3] = np.sqrt(1+(K_1*lmd**2)/(lmd**2-L_1)+
(K_2*lmd**2)/(lmd**2-L_2)+(K_3*lmd**2)/(lmd**2-L_3))
nn1[3] = 1.0
print('This process may run slower depending on your hardware,
please wait...')
```

b) Ray parameters

Here we are going to focus on defining two sets of rays, where each one will contain two subsets. One subset will define the surfaces and the second subset the grid-mesh. We have two kinds of mesh, the first is an hexapolar mesh and the other is a cartesian circular mesh.

We are going to briefly introduce some important variables that are used in this Section. First, the variable called *option*, allows us to choose what kind of subset we want, if *option* is equal to 0, then we are going to work with the hexapolar mesh. On the other hand, if *option* is equal to 1 it will use the circular mesh. NR and NR_1 set the number of elements that we want to use in our sets. LL and LL_1 define the range of the points, for this reason, we chose a $semidiameter\_pm$ (semidiameter of the primary mirror) as the maximum value. For the hexapolar mesh we have num_i, which indicates the number of multiple points of that parameter.

Moreover, rad and R define the radius of the surface which, in the same way as before, the values of this two variables are going to be set between $semidiameter\_pm$ and $semidiameter\_sm$. In order to calculate the coordinates of the origin point of each ray and its direction, the variables $meshcoorx$, $meshcoory$, $systz$, $systL$, $systM$ and $systN$ are used to define the value of $(P_{-1}(x_{-1}, y_{-1}, z_{-1})$. Finally, the two subsets will be saved in $assemblagesurf\_surf$ and $assemblageray\_ray$.

```
def setof_rayos1(option):
  option = option
  if option == 0.0:
    '''
    Parameters for surfaces
    '''
    NR_1 = 100.0
```

```
LL_1 = semidiameter_pm
systx1 = np.linspace(-LL_1, LL_1, num=NR_1)
systy1 = np.random.randint(-LL_1,LL_1,NR_1)*0.0
systz1 = np.ones(NR_1) * -100.0
systL1 = np.zeros(NR_1)
systM1 = np.zeros(NR_1)
systN1 = np.ones(NR_1)
assemblagesurf_surf = [ ]
assemblagesurf_surf.append(systx1)
assemblagesurf_surf.append(systy1)
assemblagesurf_surf.append(systz1)
assemblagesurf_surf.append(systL1)
assemblagesurf_surf.append(systM1)
assemblagesurf_surf.append(systN1)
'''
Parameters for rays
'''
num_0 = 10.0
LL = semidiameter_pm
rad = []
NR = 10.0
systx = []
systy = []
NR = []
num_i = 6.0
div_0 = LL/num_0
for i in range(11):
  rad.append(LL-i*div_0)
  NR.append(num_i * i)
NR = np.array(NR)
rad = np.array(rad)
rad = -(rad-semidiameter_pm)
rad[(rad<=semidiameter_sm)] = np.NaN
rad = rad[~np.isnan(rad)]
```

```
    numrad = len(rad)
    numrad1 = numrad-1
    for i in range(numrad1):
      NR=np.delete(NR,abs(i-i))
    for i in range(numrad):
      num_segmentos = int(NR[i])
      angle = np.linspace(0, 2*np.pi, num_segmentos+1)
      systx.append(rad[i] * np.cos(angle))
      systy.append(rad[i] * np.sin(angle))
    meshcoorx = []
    meshcoory = []
    for i in range(numrad):
      numsum = int(NR[i]+1)
      for j in range(numsum):
        meshcoorx.append(systx[i][j])
        meshcoory.append(systy[i][j])
    meshcoorx = np.array(meshcoorx)
    meshcoory = np.array(meshcoory)
    NR_1 = len(meshcoorx)
    systz = np.ones(NR_1) * -100
    systL = np.zeros(NR_1)
    systM = np.zeros(NR_1)
    systN = np.ones(NR_1)
    assemblageray_ray = []
    assemblageray_ray.append(meshcoorx)
    assemblageray_ray.append(meshcoory)
    assemblageray_ray.append(systz)
    assemblageray_ray.append(systL)
    assemblageray_ray.append(systM)
    assemblageray_ray.append(systN)
  else:
    '''
    Parameters for surfaces
    '''
```

```
NR_1 = 100.0
LL_1 = semidiameter_pm
systx1 = np.linspace(-LL_1, LL_1, num=NR_1)
systy1 = np.random.randint(-LL_1,LL_1,NR_1)*0.0
systz1 = np.ones(NR_1) * -100
systL1 = np.zeros(NR_1)
systM1 = np.zeros(NR_1)
systN1 = np.ones(NR_1)
assemblagesurf_surf = [ ]
assemblagesurf_surf.append(systx1)
assemblagesurf_surf.append(systy1)
assemblagesurf_surf.append(systz1)
assemblagesurf_surf.append(systL1)
assemblagesurf_surf.append(systM1)
assemblagesurf_surf.append(systN1)
'''
Parameters for rays
'''
NR = 45.0
LL = 2.0 * semidiameter_pm
R = []
teta = []
systx = np.linspace(LL, -LL, num=NR)
systy = np.linspace(LL, -LL, num=NR)
systxx, systyy = np.meshgrid(systx, systy)
R = np.sqrt(systxx**2 + systyy**2)
R[(R<=semidiameter_sm)] = np.NaN
R[(R>=semidiameter_pm )] = np.NaN
cos = systxx/R
sin = systyy/R
theta = np.where( sin >= 0. , np.arccos(cos) , -np.arccos(cos) )
theta = np.where( R == 0. , 0., theta)
systxx = R*np.cos(theta)
systyy = R*np.sin(theta)
```

```
    meshcoorx = []
    meshcoory = []
    for i in range(NR):
      for j in range(NR):
        meshcoorx.append(systxx[i][j])
        meshcoory.append(systyy[i][j])
    meshcoorx = np.array(meshcoorx)
    meshcoory = np.array(meshcoory)
    meshcoorx = meshcoorx[~np.isnan(meshcoorx)]
    meshcoory = meshcoory[~np.isnan(meshcoory)]
    NR_1 = len(meshcoorx)
    systz = np.ones(NR_1) * -100.0
    systL = np.zeros(NR_1)
    systM = np.zeros(NR_1)
    systN = np.ones(NR_1)
    assemblageray_ray = []
    assemblageray_ray.append(meshcoorx)
    assemblageray_ray.append(meshcoory)
    assemblageray_ray.append(systz)
    assemblageray_ray.append(systL)
    assemblageray_ray.append(systM)
    assemblageray_ray.append(systN)
  return (assemblagesurf_surf, assemblageray_ray)
```

c) Find the intersection of the ray in the next coordinate system

As we mentioned in Section a), the main goal in general exact ray tracing is to find the intersection of the ray in every coordinate system, because it determines the optical path of the ray, passing through a set of surfaces. As a first step, we define a function that will help us to find the intersection point. Hereafter this Section is divided into Subsections, because these subsections contain the functions that we will use. We present a brief explanation of each function below.

c.1) Coordinates of the P_0 function.

Before we find the intersection point P_1, which is the intersection point on a surface, we need to find the origin point P_0. To find this point, we use equation $x_1 = (L/N) * (z_1 - z_0) + x_0$ and $y_1 = (M/N) * (z_1 - z_0) + y_0$. Then, we iteratively solve the ray-surface function.

```
def F_RS(z_1,rayorigin,i):
    x_0 = rayorigin[0]
    y_0 = rayorigin[1]
    z_0 = rayorigin[2]
    L = rayorigin[3]
    M = rayorigin[4]
    N = rayorigin[5]
    x_1 = (L/N)*(z_1-z_0)+x_0
    y_1 = (M/N)*(z_1-z_0)+y_0
    f_RS = surface(x_1,y_1,z_1,i)
    return f_RS
```

c.2) Ray-Surface function

After finding the coordinates x_1 and y_1, we iteratively solve the ray-surface function. This function contains the sagitta function which defines a surface.

```
def surface(x_1,y_1,z_1,i):
    R_c = Rc[i]
    k = kk[i]
    Alpha_1 = aalpha_1[i]
    Alpha_2 = aalpha_2[i]
    Alpha_3 = aalpha_3[i]
    Alpha_4 = aalpha_4[i]
    Alpha_5 = aalpha_5[i]
    c = 1.0/R_c
```

```
    s2 = (x_1*x_1)+(y_1*y_1)
    s4 = s2*s2
    s6 = s4*s2
    s8 = s6*s2
    s10 = s8*s2
    f1 = c*s2/(1.0+np.sqrt(1.0-((k+1.0)*c*c*s2)))
    f2 = (Alpha_1*s2)+(Alpha_2*s4)+(Alpha_3*s6)+(Alpha_4*s8)+(Alpha_5*s10)
    return f1+f2-z_1
```

c.3) Derivative of the ray-surface function

We compute a numerical derivative of the ray-surface function, instead of performing a simple derivative, in order to calculate the intersection point.

```
def Der_F_RS(z_1,rayorigin,i):
    h = 0.00000000001
    der = (F_RS(z_1+h, rayorigin,i)-F_RS(z_1-h, rayorigin,i))/(2*h)
    return der
```

c.4) Newton_Raphson function

To find the point P_1, we use the Newton-Rhapson method within a loop. This method is useful for finding the roots of the function that defines the intersection with the surface (c.3).

```
def Newton_Raphson(rayorigin,i):
    z_1 = 0
    cnt = 0
    while True:
        z_2 = z_1-(F_RS(z_1,rayorigin,i)/Der_F_RS(z_1,rayorigin,i))
        if np.abs(z_2-z_1)<0.00001:
            break
        else:
```

```
            z_1=z_2
        if cnt == 30:
            break
        cnt=cnt+1
    [x_0, y_0, z_0,L,M,N]=rayorigin
    x_1 = (L/N)*(z_1-z_0)+x_0
    y_1 = (M/N)*(z_1-z_0)+y_0
    return x_1, y_1, z_1
```

c.5) Normal Vector function

Once the position of the intersection point is known, we calculate the normal vector using the numerical derivative of the ray-surface function.

```
def surface_Derivative(x,y,z,i):
    delta=0.000001
    Dx = (surface(x+delta,y,z,i)-surface(x-delta,y,z,i))/(2.0*delta)
    Dy = (surface(x,y+delta,z,i)-surface(x,y-delta,z,i))/(2.0*delta)
    Dz = (surface(x,y,z+delta,i)-surface(x,y,z-delta,i))/(2.0*delta)
    Dr = np.sqrt((Dx*Dx)+(Dy*Dy)+(Dz*Dz))
    return Dx/Dr,Dy/Dr,Dz/Dr
```

c.6) Snell's function

When the ray hits a surface, we already know the direction cosines and the normal vector to that surface, but we don't know the direction of the ray that is refracted or reflected. To know that we have to use Snell's law in its vector form.

```
def Snell_refraction_vector(Ray_Vect,Surf_Normal,n_0,n_1):
    Ray_Vect = np.asarray(Ray_Vect)
    Surf_Normal = np.asarray(Surf_Normal)
    Nsurf_Cros_s1 = np.cross(Surf_Normal,Ray_Vect)
    NN = n_0/n_1
```

```
    R = (NN*NN)*np.dot(Nsurf_Cros_s1,Nsurf_Cros_s1)
    if R>1:
        print("----- internal reflection= ", R)
    s2 = NN*(np.cross(Surf_Normal,np.cross(-Surf_Normal,Ray_Vect)))
    -Surf_Normal*np.sqrt(1.0-((NN*NN)*np.dot(Nsurf_Cros_s1,
    Nsurf_Cros_s1)))
    return s2
```

c.7) Skew ray tracing function

This function uses all the previous functions to calculate the point of intersection and the direction of the ray when it interacts with the surface. This function is used as many times as there are surfaces.

```
def Trax(Ray_Param,i):
    ## Point of intersection on the surface
    x_1, y_1, z_1 = Newton_Raphson(Ray_Param,i)
    ## Normal vector at the point of intersection on the
    surface
    NORM = surface_Derivative(x_1,y_1,z_1,i)
    [x_0, y_0, z_0,L,M,N] = Ray_Param
    Ray_Vect = [L,M,N]
    Surf_Normal = NORM
    n_0 = nn0[i]
    n_1 = nn1[i]
    lmn = Snell_refraction_vector(Ray_Vect,Surf_Normal,n_0,n_1)
    return ([x_1, y_1, z_1, lmn[0], lmn[1], lmn[2]])
```

c.6) Director Cosines function

If we don't know the director cosines of some point, we can calculate them with this function.

```
def director_cosines(x,y,z):
  magnitude = np.sqrt(x**2+y**2+z**2)
  L = x/magnitude
  M = y/magnitude
  N = z/magnitude
  return L, M, N
```

d) Change the ray to the transformed coordinate system

We want to perform a displacement and a rotation on the secondary mirror. For this we calculate the values of a segment of a ray that travels between two surfaces. The segment is defined by two extreme points, P_0 and P_1, both have a coordinate system (x,y,z). The shift and rotation on the secondary mirror make the original coordinate system to become (x',y',z'). Then, we calculate the intersection point with the surface in the new system that has undergone the transformation.

The information of the point of intersection in the new system can be obtained with different functions that we will explain below.

Transformation function. The information in the transformed space is known through the translation matrix (T_{xyz}) and the rotation matrices (R_x, R_y and R_z). The first matrix contains the three displacements of the three spatial directions and the other matrices contain the angles at which the rotation around the axes was applied.

Consequently, the matrix with the resulting transformation is the product of these four matrices. Also, this matrix will involve an inverse transformation to return to the original coordinate system.

```
def Transform(TRF):
    [TiltX, TiltY, TiltZ, dx, dy, dz]=TRF
    Tx=np.deg2rad(TiltX)
    Ty=np.deg2rad(TiltY)
    Tz=np.deg2rad(TiltZ)
```

```
    Rx_1A = np.matrix([[1.0, 0.0, 0.0 , 0.0], [0.0, np.cos(-Tx),
    np.sin(-Tx),0.0],[0.0,-np.sin(-Tx), np.cos(-Tx),0.0],[0.0,
    0.0, 0.0 ,1.0]])
    Ry_1A = np.matrix([[np.cos(-Ty),0.0, -np.sin(-Ty),0.0], [0.0,
    1.0, 0.0, 0.0],[np.sin(-Ty),0.0, np.cos(-Ty),0.0],[0.0, 0.0,
    0.0 ,1.0]])
    Rz_1A = np.matrix([[np.cos(-Tz),-np.sin(-Tz), 0.0, 0.0],
    [np.sin(-Tz),np.cos(-Tz), 0.0, 0.0],[0.0, 0.0, 1.0, 0.0],
    [0.0, 0.0, 0.0 ,1.0]])
    Dxyz_1A = np.matrix([[1.0, 0.0, 0.0, -dx],[0.0, 1.0, 0.0, -dy]
    ,[0.0, 0.0, 1.0, -dz],[0.0, 0.0, 0.0 ,1.0]])
    Start_trans1=Rz_1A*Ry_1A*Rx_1A*Dxyz_1A
    Start_trans2=np.linalg.inv(Start_trans1)
    return Start_trans1, Start_trans2
```

Operations in the transformed space system.

```
def Traceray(inx,iny,inz,inL,inM,inN, MT):
    coorx_0[0] = inx
    coory_0[0] = iny
    coorz_0[0] = inz
    cosL[0] = inL
    cosM[0] = inM
    cosN[0] = inN
    for i in range (number_of_surfaces):
        '''
        A1_Trans is the transformation matrix to the
        transformed space and A2_Trans contains the
        inverse transformation.
        '''
        A1_Trans = MT[i][0]
        A2_Trans = MT[i][1]
        x_0 = coorx_0[i]
```

```
y_0 = coory_0[i]
z_0 = coorz_0[i] - d[i]
L = cosL[i]
M = cosM[i]
N = cosN[i]
'''
e), f) and g) We already know the points of origin and
their direction, thus, we can calculate the second point,
in order to find the extreme points P_0 and P_1.
'''
Ray_Param = [x_0, y_0, z_0, L, M, N]
xyz_1=Trax(Ray_Param,i)
x_1 = xyz_1[0]
y_1 = xyz_1[1]
z_1 = xyz_1[2]
'''
We transform the components of P_0 and P_1 using the
transformation matrix as follows:
'''
prime_coor0 = np.dot(A1_Trans, ([x_0, y_0, z_0, 1.0]))
prime_coor1 = np.dot(A1_Trans,([x_1, y_1, z_1, 1.0]))

'''
We have to calculate the director cosines in the
transformed space because the director cosines are
not transformed automatically.
'''
prime_scosinD = Cosine_transform(prime_coor0, prime_coor1)
'''
Now, we know the points of origin in the transformed
space and their direction:
'''
x_0 = prime_coor0[0,0]
y_0 = prime_coor0[0,1]
```

```
z_0 = prime_coor0[0,2]
s=np.sign(d[i])
if i==0:
    s=1
L = prime_scosinD[0]*s
M = prime_scosinD[1]*s
N = prime_scosinD[2]*s
Ray_Param = [x_0, y_0, z_0, L, M, N]
xyz_1=Trax(Ray_Param,i)
x_1 = xyz_1[0]
y_1 = xyz_1[1]
z_1 = xyz_1[2]
L = xyz_1[3]
M = xyz_1[4]
N = xyz_1[5]
z_2=z_0
x_2=(L/N)*(z_2-z_1)+x_1
y_2=(M/N)*(z_2-z_1)+y_1
'''
h) At this point we know the point of intersection with
the system that underwent a displacement and a rotation.
We have to calculate two points because we need to transform
the director cosines to the original system.
j) In this way, prime_coor0 and prime_coor1 are the
intersection points in the original system.
k) Prime_scosinD are the director cosines of the ray in
the original system.
'''
prime_coor0 = np.dot(A2_Trans, ([x_1, y_1, z_1, 1.0]))
prime_coor1 = np.dot(A2_Trans, ([x_2, y_2, z_2, 1.0]))
prime_scosinD = Cosine_transform(prime_coor0,prime_coor1)
x_1 = prime_coor0[0,0]
y_1 = prime_coor0[0,1]
z_1 = prime_coor0[0,2]
```

```
        L = prime_scosinD[0]
        M = prime_scosinD[1]
        N = prime_scosinD[2]
        coorx_1[i] = x_1
        coory_1[i] = y_1
        coorz_1[i] = z_1
        cosL[i+1] = L
        cosM[i+1] = M
        cosN[i+1] = N
        coorx_0[i+1] = coorx_1[i]
        coory_0[i+1] = coory_1[i]
        coorz_0[i+1] = coorz_1[i]
    return coorx_1,coory_1, coorz_1, cosL, cosM, cosN
```

Cosine transformation function. As we mention, we have to calculate the director cosines of the ray using two points, one in each systems. This function allows us to do it.

```
def Cosine_transform(system_cos0,system_cos1):
    x_0 = system_cos0[0,0]
    y_0 = system_cos0[0,1]
    z_0 = system_cos0[0,2]
    x_1 = system_cos1[0,0]
    y_1 = system_cos1[0,1]
    z_1 = system_cos1[0,2]
    magnitude_transform = np.sqrt((x_1-x_0)**2+(y_1-y_0)**2+
    (z_1-z_0)**2)
    pm_L = (x_1-x_0)/magnitude_transform
    pm_M = (y_1-y_0)/magnitude_transform
    pm_N = (z_1-z_0)/magnitude_transform
    return pm_L, pm_M, pm_N
```

n) Ray information through the system.

All the information up to this point will be stored in two variables, each of which will contain ray and surface information.

We defined two functions (n.1 and n.2), the first function calls all the previous functions that we define and it carries out the exact ray tracing. Moreover, this function will store all the information that it generates. The second function specifies the displacement and rotation of all the surfaces that will undergo a change and calculates the two transformation matrices. Finally, we use the (n.1) function to compute the parameters of the rays and the surface.

```
'''
n.1 function.
'''
def Rays_infomation(ray_param,Matrix):
  initial_coor0 = ray_param[0]
  initial_coor1 = ray_param[1]
  initial_coor2 = ray_param[2]
  director_cos0 = ray_param[3]
  director_cos1 = ray_param[4]
  director_cos2 = ray_param[5]
  NR2 = len(initial_coor0)
  systcoorx_5 = np.ones(NR2)
  systcoory_5 = np.ones(NR2)
  systcoorz_5 = np.ones(NR2)
  systcosL_5 = np.ones(NR2)
  systcosM_5 = np.ones(NR2)
  systcosN_5 = np.ones(NR2)
  systcoorx_4 = np.ones(NR2)
  systcoory_4 = np.ones(NR2)
  systcoorz_4 = np.ones(NR2)
  systcosL_4 = np.ones(NR2)
  systcosM_4 = np.ones(NR2)
  systcosN_4 = np.ones(NR2)
```

```
systcoorx_3 = np.ones(NR2)
systcoory_3 = np.ones(NR2)
systcoorz_3 = np.ones(NR2)
systcosL_3 = np.ones(NR2)
systcosM_3 = np.ones(NR2)
systcosN_3 = np.ones(NR2)
systcoorx_2 = np.ones(NR2)
systcoory_2 = np.ones(NR2)
systcoorz_2 = np.ones(NR2)
systcosL_2 = np.ones(NR2)
systcosM_2 = np.ones(NR2)
systcosN_2 = np.ones(NR2)
systcoorx_1 = np.ones(NR2)
systcoory_1 = np.ones(NR2)
systcoorz_1 = np.ones(NR2)
systcosL_1 = np.ones(NR2)
systcosM_1 = np.ones(NR2)
systcosN_1 = np.ones(NR2)
planoimagen = np.ones(NR2)
for i in range(NR2):
    inx = initial_coor0[i]
    iny = initial_coor1[i]
    inz = initial_coor2[i]
    inL = director_cos0[i]
    inM = director_cos1[i]
    inN = director_cos2[i]
    cx_1, cy_1, cz_1, cL, cM, cN = Traceray(inx,iny,inz,inL,
    inM, inN, Matrix)
    systcoorx_5[i] = cx_1[4]
    systcoory_5[i] = cy_1[4]
    systcoorz_5[i] = cz_1[4]
    systcosL_5[i] = cL[5]
    systcosM_5[i] = cM[5]
    systcosN_5[i] = cN[5]
```

```
    systcoorx_4[i] = cx_1[3]
    systcoory_4[i] = cy_1[3]
    systcoorz_4[i] = cz_1[3]
    systcosL_4[i] = cL[4]
    systcosM_4[i] = cM[4]
    systcosN_4[i] = cN[4]
    systcoorx_3[i] = cx_1[2]
    systcoory_3[i] = cy_1[2]
    systcoorz_3[i] = cz_1[2]
    systcosL_3[i] = cL[3]
    systcosM_3[i] = cM[3]
    systcosN_3[i] = cN[3]
    systcoorx_2[i] = cx_1[1]
    systcoory_2[i] = cy_1[1]
    systcoorz_2[i] = cz_1[1]
    systcosL_2[i] = cL[2]
    systcosM_2[i] = cM[2]
    systcosN_2[i] = cN[2]
    systcoorx_1[i] = cx_1[0]
    systcoory_1[i] = cy_1[0]
    systcoorz_1[i] = cz_1[0]
    systcosL_1[i] = cL[1]
    systcosM_1[i] = cM[1]
    systcosN_1[i] = cN[1]
Object_plane = (initial_coor0, initial_coor1, initial_coor2,
director_cos0, director_cos1, director_cos2)
Primary_mirror = (systcoorx_1, systcoory_1, systcoorz_1,
systcosL_1, systcosM_1, systcosN_1)
Secondary_mirror = (systcoorx_2, systcoory_2, systcoorz_2,
systcosL_2, systcosM_2, systcosN_2)
Correctingasphlens = (systcoorx_3, systcoory_3, systcoorz_3,
systcosL_3, systcosM_3, systcosN_3)
Corrector_plane = (systcoorx_4, systcoory_4, systcoorz_4,
systcosL_4, systcosM_4, systcosN_4)
```

```
  Image_plane = (systcoorx_5, systcoory_5, systcoorz_5,
  systcosL_5, systcosM_5, systcosN_5)
  surf_opt = (Primary_mirror, Secondary_mirror,
  Correctingasphlens)
  plane_opt = (Object_plane, Corrector_plane, Image_plane)
  return surf_opt, plane_opt
'''
n.2 function.
'''
def TiltandShift_transtormation(tilt_smirror, shift_smirror,
set_rayos):
  tiltthetax = tilt_smirror[0]
  tiltthetay = tilt_smirror[1]
  shiftx = shift_smirror[0]
  shifty = shift_smirror[1]
  MT=[]
  ma_rt_s0 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  ma_rt_s1 = [tiltthetax , tiltthetay, 0.0, shiftx, shifty, 0.0]
  ma_rt_s2 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  ma_rt_s3 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  ma_rt_s4 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  ma_rt_s5 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  ma_rt_s6 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  ma_rt_s7 = [0.0 , 0.0, 0.0, 0.0, 0.0, 0.0]
  M = Transform(ma_rt_s0)
  MT.append(M)
  M = Transform(ma_rt_s1)
  MT.append(M)
  M = Transform(ma_rt_s2)
  MT.append(M)
  M = Transform(ma_rt_s3)
  MT.append(M)
  M = Transform(ma_rt_s4)
  MT.append(M)
```

```
  surf_transformate, plane_transformate = Rays_infomation(set_rayos,MT)
  return surf_transformate, plane_transformate
```

A.2.2. Coma evaluation function

To evaluate and eliminate the presence of axial coma we use the transverse coma aberration, which is defined as the change in amplification as a function of the aperture. To achieve this, we have to use three rays that arrive on a meridional plane, two of them are marginal rays passing through the edges of the pupil of the system and the last one is a principal ray.

From A.2.1, we calculate the director cosines of these three rays and the intersection point in the last surface. In this way, we can find the intersection point between the two marginal rays and we can calculate the distance from this point to the optical axis. Furthermore, with the intersection between the marginal rays and the distance from this intersection to the optical axis, we determine the position of the perpendicular plane that generates the intersection point and, finally, we can calculate the intersection of the principal ray with the perpendicular plane, and its distance from the optical axis.

We need three rays for the coma evaluation function, thus, we define a set of three rays. Two of them are marginal rays passing through the edges of the pupil of the system. However, as the pupil of the system is the primary mirror, both rays pass through $semidiameter\_pm$ and $-semidiameter\_pm$. The last one is a principal ray that passes through the center of the pupil.

```
def setof_rayos(option):
  option = option
  if option == 0.0:
    '''
    Parameters for surfaces
```

```
'''
NR_1 = 100.0
LL_1 = semidiameter_pm
systx1 = np.linspace(-LL_1, LL_1, num=NR_1)
systy1 = np.random.randint(-LL_1,LL_1,NR_1)*0.0
systz1 = np.ones(NR_1) * -100.0
systL1 = np.zeros(NR_1)
systM1 = np.zeros(NR_1)
systN1 = np.ones(NR_1)
assemblagesurf_surf = []
assemblagesurf_surf.append(systx1)
assemblagesurf_surf.append(systy1)
assemblagesurf_surf.append(systz1)
assemblagesurf_surf.append(systL1)
assemblagesurf_surf.append(systM1)
assemblagesurf_surf.append(systN1)
'''
Parameters for rays
'''
meshcoorx = ( 0.0, 0.0, 0.0)
meshcoory = ( 0.0, semidiameter_pm , -semidiameter_pm )
meshcoorx = np.array(meshcoorx)
meshcoory = np.array(meshcoory)
NR_1 = len(meshcoorx)
systz = np.ones(NR_1) * -100.0
systL = np.zeros(NR_1)
systM = np.zeros(NR_1)
systN = np.ones(NR_1)
assemblageray_ray = []
assemblageray_ray.append(meshcoorx)
assemblageray_ray.append(meshcoory)
assemblageray_ray.append(systz)
assemblageray_ray.append(systL)
assemblageray_ray.append(systM)
```

```
    assemblageray_ray.append(systN)
  return (assemblagesurf_surf, assemblageray_ray)
```

The skew ray tracing gives us the information necessary to calculate the intersection between the marginal rays, the distance of this intersection to the optical axis, the position of the perpendicular plane that generates the intersection point, the intersection of the principal ray with the perpendicular plane and its distance from the optical axis. For this, we define the next function.

```
def Height_yp(Set_lsurface):
    last_surface = Set_lsurface
    valuex_LS = last_surface[0]
    valuey_LS = last_surface[1]
    valuez_LS = last_surface[2]
    valueL_LS = last_surface[3]
    valueM_LS = last_surface[4]
    valueN_LS = last_surface[5]
    yp0 = valuey_LS[0]
    yma = valuey_LS[1]
    ymb = valuey_LS[2]
    zp0 = valuez_LS[0]
    zma = valuez_LS[1]
    zmb = valuez_LS[2]
    Mp0 = valueM_LS[0]
    Mma = valueM_LS[1]
    Mmb = valueM_LS[2]
    Np0 = valueN_LS[0]
    Nma = valueN_LS[1]
    Nmb = valueN_LS[2]
    ym = ((Mma*Mmb)/(Nma*Mmb-Nmb*Mma))*(zmb-zma-ymb*(Nmb/Mmb)
    +yma*(Nma/Mma))
    zm = (ym-ymb)*(Nmb/Mmb)+zmb
```

```
    yp = yp0+(Mp0/Np0)*(zm-zp0)
    dt_yp = np.abs(yp-ym)
    return dt_yp
```

A.2.3. Zero coma point function

We use the Newton-Raphson method in an optimization way, this allows us to calculate a tilt in "y"that minimizes the amount of coma, by setting a criterion near to zero, for a certain displacement. The projection of the resulting compensation is calculated using trigonometric properties.

The proposed displacement is 1 mm in the y-axis.

```
shift_smirror = (0.0, 1.0)
tilt_smirror = (0.0, 0.0)
opc = 0.0
surf_rset , ray_rset = setof_rayos(opc)
surf_surf, surf_plane = TiltandShift_transtormation(tilt_smirror,
shift_smirror, ray_rset)
ray_surf, ray_plane = TiltandShift_transtormation(tilt_smirror,
shift_smirror, ray_rset)
def Der_highym(shift_set, set_rayos):
    shift_set = np.array(shift_set)
    tilt_set = (0.0, 0.0)
    h = 0.000000006
    shift_ph = shift_set + h
    shift_nh = shift_set - h
    surf_shiftph, plane_shiftph =
    TiltandShift_transtormation(tilt_set,shift_ph,set_rayos)
    surf_shiftnh, plane_shiftnh=
    TiltandShift_transtormation(tilt_set,shift_nh,set_rayos)
```

```
    l_surfaceph = plane_shiftph[2]
    l_surfacenh = plane_shiftnh[2]
    dt_ymph = Height_yp(l_surfaceph)
    dt_ymnh= Height_yp(l_surfacenh)
    derivate_ym = (dt_ymph-dt_ymnh)/(2*h)
    return derivate_ym
def NewtonRaphson_tilt(shift_array, set_rayos):
    tilt_array = (0.0,0.0)
    shift_array = np.array(shift_array)
    tilt_array = np.array(tilt_array)
    numcnt = 50000
    tilt_0 = 0.0
    cnt = 0
    while True:
      surf_NRprocedure, plane_NRprocedure =
      TiltandShift_transtormation(tilt_array,shift_array,
      set_rayos)
      final_surface = plane_NRprocedure[2]
      delta_ym = Height_yp(final_surface)
      Derivate_ym = Der_highym(shift_array, set_rayos)
      tilt_array[0] = tilt_0 - delta_ym/Derivate_ym
      if delta_ym < 1.5E-3:
          break
      else:
          tilt_0 = tilt_array[0]
      if cnt == numcnt:
        break
      cnt = cnt + 1
    Tx = np.deg2rad(tilt_array[0])
    pointzcoma = np.abs(shift_array[1]/np.tan(Tx))
    return tilt_array, pointzcoma

'''
Here we calculate the value of the angle that minimizes the
```

```
amount of coma and its respective zero coma point.
'''
tilt_tocorrect, pointncoma = NewtonRaphson_tilt(shift_smirror,
ray_rset)

'''
With the value of the angle and the skew ray tracing, we
obtain information of the compensated system.
'''
surf_correctedset , ray_correctedset = setof_rayos1(0)
ray_shiftysurf, ray_shiftyplane =
TiltandShift_transtormation(tilt_smirror, shift_smirror,
ray_correctedset)
surf_shiftysurf, surf_shiftyplane =
TiltandShift_transtormation(tilt_smirror, shift_smirror,
surf_correctedset)
ray_correctedsurf, ray_correctedplane =
TiltandShift_transtormation(tilt_tocorrect,shift_smirror,
ray_correctedset)
surf_correctedsurf, surf_correctedplane =
TiltandShift_transtormation(tilt_tocorrect,shift_smirror,
surf_correctedset)
```

A.2.4. Plotting and results

In this last Section, we provide the angle (°) and the zero coma point (mm). Furthermore, five plots are shown which set out the Optic system and the spot diagram with the system's displacement and the compensated system. The last plot shows the mesh of rays that we used. Given that we have two sets ($assemblagesurf\_surf$ and $assemblageray\_ray$) which have six subsets, we redefine all this subsets into variables that will more easily handle this information.

```
print('Secondary displacement:',shift_smirror[1], 'mm')
print('Calculated tilt angle to correct:',tilt_tocorrect[0],
'degrees')
print('Position of the zero coma point:',pointncoma, 'mm')
print('')
print('')
```

a) Surface displacement

```
'''
Information of the rays
'''
objectshifty_ray = ray_shiftyplane[0]
objectshifty_rayx = objectshifty_ray[0]
objectshifty_rayy = objectshifty_ray[1]
objectshifty_rayz = objectshifty_ray[2]
pMirrorshifty_ray = ray_shiftysurf[0]
pMirrorshifty_rayx = pMirrorshifty_ray[0]
pMirrorshifty_rayy = pMirrorshifty_ray[1]
pMirrorshifty_rayz = pMirrorshifty_ray[2]
sMirrorshifty_ray = ray_shiftysurf[1]
sMirrorshifty_rayx = sMirrorshifty_ray[0]
sMirrorshifty_rayy = sMirrorshifty_ray[1]
sMirrorshifty_rayz = sMirrorshifty_ray[2]
asphlenseshifty_ray = ray_shiftysurf[2]
asphlenseshifty_rayx = asphlenseshifty_ray[0]
asphlenseshifty_rayy = asphlenseshifty_ray[1]
asphlenseshifty_rayz = asphlenseshifty_ray[2]
planelenseshifty_ray = ray_shiftyplane[1]
planelenseshifty_rayx = planelenseshifty_ray[0]
planelenseshifty_rayy = planelenseshifty_ray[1]
planelenseshifty_rayz = planelenseshifty_ray[2]
```

```
Imageshifty_ray = ray_shiftyplane[2]
Imageshifty_rayx = Imageshifty_ray[0]
Imageshifty_rayy = Imageshifty_ray[1]
Imageshifty_rayz = Imageshifty_ray[2]
'''
Information of the surfaces
'''
objectshifty_surf = surf_shiftyplane[0]
objectshifty_surfx = objectshifty_surf[0]
objectshifty_surfy = objectshifty_surf[1]
objectshifty_surfz = objectshifty_surf[2]
pMirrorshifty_surf = surf_shiftysurf[0]
pMirrorshifty_surfx = pMirrorshifty_surf[0]
pMirrorshifty_surfy = pMirrorshifty_surf[1]
pMirrorshifty_surfz = pMirrorshifty_surf[2]
sMirrorshifty_surf = surf_shiftysurf[1]
sMirrorshifty_surfx = sMirrorshifty_surf[0]
sMirrorshifty_surfy = sMirrorshifty_surf[1]
sMirrorshifty_surfz = sMirrorshifty_surf[2]
asphlenseshifty_surf = surf_shiftysurf[2]
asphlenseshifty_surfx = asphlenseshifty_surf[0]
asphlenseshifty_surfy = asphlenseshifty_surf[1]
asphlenseshifty_surfz = asphlenseshifty_surf[2]
planelenseshifty_surf = surf_shiftyplane[1]
planelenseshifty_surfx = planelenseshifty_surf[0]
planelenseshifty_surfy = planelenseshifty_surf[1]
planelenseshifty_surfz = planelenseshifty_surf[2]
Imageshifty_surf = surf_shiftyplane[2]
Imageshifty_surfx = Imageshifty_surf[0]
Imageshifty_surfy = Imageshifty_surf[1]
Imageshifty_surfz = Imageshifty_surf[2]
'''
Plot of the Optic System
'''
```

```
'''
Plot of the rays
'''
l = 217
m = 258
grs = 0.3
x = np.array([objectshifty_rayx[l:m],pMirrorshifty_rayx[l:m]])
z = np.array([objectshifty_rayz[l:m]- 2.500000000000000E+003,
pMirrorshifty_rayz[l:m]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([pMirrorshifty_rayx[l:m],sMirrorshifty_rayx[l:m]])
z = np.array([pMirrorshifty_rayz[l:m],sMirrorshifty_rayz[l:m]
+ d[1]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([sMirrorshifty_rayx[l:m],asphlenseshifty_rayx[l:m]])
z = np.array([sMirrorshifty_rayz[l:m] + d[1],
asphlenseshifty_rayz[l:m]+ d[1] + d[2]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([asphlenseshifty_rayx[l:m],planelenseshifty_rayx[l:m]])
z = np.array([asphlenseshifty_rayz[l:m]+ d[1] + d[2],
planelenseshifty_rayz[l:m] + d[1] + d[2] + d[3]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([planelenseshifty_rayx[l:m],Imageshifty_rayx[l:m]])
z = np.array([planelenseshifty_rayz[l:m] + d[1] + d[2] + d[3],
Imageshifty_rayz [l:m] + d[1] + d[2] + d[3] +  d[4]])
plt.plot(z, x, color = "red", markersize = grs)
'''
Plot of the surfaces
'''
grs_surf = 0.7
x = pMirrorshifty_surfx
y = pMirrorshifty_surfy
z = pMirrorshifty_surfz
plt.plot(z, x, color = 'blue', markersize = grs_surf)
```

```
x = sMirrorshifty_surfx
y = sMirrorshifty_surfy
z = sMirrorshifty_surfz + d[1]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = asphlenseshifty_surfx
y = asphlenseshifty_surfy
z = asphlenseshifty_surfz  + d[1] + d[2]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = planelenseshifty_surfx
y = planelenseshifty_surfy
z = planelenseshifty_surfz  + d[1] + d[2] + d[3]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = Imageshifty_surfx
y = Imageshifty_surfy
z = Imageshifty_surfz + d[1] + d[2] + d[3] +  d[4]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
plt.xlabel(r'Z [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.ylabel(r'X [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.xticks([-2500.0,-2000.0,-1500.0,-1000.0,-500.0, 0.0],
[r'$0.0$',r'$500.0$',r'$1000.0$',r'$1500.0$',r'$2000.0$',r'$2500.0$'])
plt.yticks([-1000.0,-500.0, 0.0, 500.0, 1000.0,],
[r'$-1000.0$',r'$-500.0$',r'$0.0$',r'$500.0$',r'$1000.0$'])
plt.tick_params( axis = 'both', direction = 'out' , width = 2,
length = 5, labelsize = 14)
plt.axis('equal')
plt.tight_layout()
plt.figure(1)
plt.show()
'''
Plot of the Spot Diagram
'''
print('')
```

```
print('')
plt.plot(Imageshifty_rayx, Imageshifty_rayy,'.', color = 'k',
linestyle = 'none')
plt.xlabel(r'Linear distance on the image plane [mm]', fontsize = 16,
fontstyle = "normal", family = 'serif')
plt.ylabel(r'Field angle [degrees]', fontsize=16, fontstyle = "normal",
family = 'serif')
plt.xticks([0.0],[r'$0.0$'])
plt.yticks([-0.6],[r'$0.0$'])
plt.tick_params( axis = 'both', direction = 'out' , width = 2, length = 5,
labelsize = 14)
plt.axis('equal')
plt.tight_layout()
plt.figure(2)
plt.show()
print('')
print('')
print('')
print('')
```

---

b) Potting of the compensated system

```
'''
Information of the rays
'''
object_ray = ray_correctedplane[0]
object_rayx = object_ray[0]
object_rayy = object_ray[1]
object_rayz = object_ray[2]
pMirror_ray = ray_correctedsurf[0]
pMirror_rayx = pMirror_ray[0]
```

```
pMirror_rayy = pMirror_ray[1]
pMirror_rayz = pMirror_ray[2]
sMirror_ray = ray_correctedsurf[1]
sMirror_rayx = sMirror_ray[0]
sMirror_rayy = sMirror_ray[1]
sMirror_rayz = sMirror_ray[2]
asphlense_ray = ray_correctedsurf[2]
asphlense_rayx = asphlense_ray[0]
asphlense_rayy = asphlense_ray[1]
asphlense_rayz = asphlense_ray[2]
planelense_ray = ray_correctedplane[1]
planelense_rayx = planelense_ray[0]
planelense_rayy = planelense_ray[1]
planelense_rayz = planelense_ray[2]
Image_ray = ray_correctedplane[2]
Image_rayx = Image_ray[0]
Image_rayy = Image_ray[1]
Image_rayz = Image_ray[2]
'''
Information of the surfaces
'''
object_surf = surf_correctedplane[0]
object_surfx = object_surf[0]
object_surfy = object_surf[1]
object_surfz = object_surf[2]
pMirror_surf = surf_correctedsurf[0]
pMirror_surfx = pMirror_surf[0]
pMirror_surfy = pMirror_surf[1]
pMirror_surfz = pMirror_surf[2]
sMirror_surf = surf_correctedsurf[1]
sMirror_surfx = sMirror_surf[0]
sMirror_surfy = sMirror_surf[1]
sMirror_surfz = sMirror_surf[2]
asphlense_surf = surf_correctedsurf[2]
```

```
asphlense_surfx = asphlense_surf[0]
asphlense_surfy = asphlense_surf[1]
asphlense_surfz = asphlense_surf[2]
planelense_surf = surf_correctedplane[1]
planelense_surfx = planelense_surf[0]
planelense_surfy = planelense_surf[1]
planelense_surfz = planelense_surf[2]
Image_surf = surf_correctedplane[2]
Image_surfx = Image_surf[2]
Image_surfy = Image_surf[1]
Image_surfz = Image_surf[2]
'''
Plot of the Compensated Optic System
'''
'''
Plot of the rays
'''
l = 0
m = 258
grs = 0.3
x = np.array([object_rayx[l:m], pMirror_rayx[l:m]])
z = np.array([object_rayz[l:m] - 2.500000000000000E+003,
pMirror_rayz[l:m]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([pMirror_rayx[l:m],sMirror_rayx[l:m]])
z = np.array([pMirror_rayz[l:m],sMirror_rayz[l:m]+ d[1]])
plt.plot(z, x, color="red", markersize=grs)
x = np.array([sMirror_rayx[l:m],asphlense_rayx[l:m]])
z = np.array([sMirror_rayz[l:m] + d[1], asphlense_rayz[l:m]+ d[1]
+ d[2]])
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([asphlense_rayx[l:m],planelense_rayx[l:m]])
z = np.array([asphlense_rayz[l:m]+
d[1] + d[2], planelense_rayz[l:m] + d[1] + d[2] + d[3]])
```

```
plt.plot(z, x, color = "red", markersize = grs)
x = np.array([planelense_rayx[1:m],Image_rayx[1:m]])
z = np.array([planelense_rayz[1:m] + d[1] + d[2] + d[3],
Image_rayz [1:m] + d[1] + d[2] + d[3] +  d[4]])
plt.plot(z, x, color = "red", markersize = grs)
'''
Plot of the surfaces
'''
grs_surf = 0.7
x = pMirror_surfx
y = pMirror_surfy
z = pMirror_surfz
plt.plot(z, x, color = 'blue', markersize  = grs_surf)
x = sMirror_surfx
y = sMirror_surfy
z = sMirror_surfz + d[1]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = asphlense_surfx
y = asphlense_surfy
z = asphlense_surfz  + d[1] + d[2]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = planelense_surfx
y = planelense_surfy
z = planelense_surfz  + d[1] + d[2] + d[3]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
x = Image_surfx
y = Image_surfy
z = Image_surfz + d[1] + d[2] + d[3] +  d[4]
plt.plot(z, x, color = 'blue', markersize = grs_surf)
plt.xlabel(r'Z [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.ylabel(r'X [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.xticks([-2500.0,-2000.0,-1500.0,-1000.0,-500.0, 0.0],
```

```
[r'$0.0$',r'$500.0$',r'$1000.0$',r'$1500.0$',r'$2000.0$',r'$2500.0$'])
plt.yticks([-1000.0,-500.0, 0.0, 500.0, 1000.0,],
[r'$-1000.0$',r'$-500.0$',r'$0.0$',r'$500.0$',r'$1000.0$'])
plt.tick_params( axis = 'both', direction = 'out' , width = 2,
length = 5, labelsize = 14)
plt.axis('equal')
plt.tight_layout()
plt.figure(3)
plt.show()
'''
Plot of the Compensated Spot Diagram
'''
print('')
print('')
plt.plot(Image_rayx, Image_rayy,'.', color = 'k', linestyle =
'none')
plt.xlabel(r'Linear distance on the image plane [mm]', fontsize = 16,
fontstyle = "normal",
family = 'serif')
plt.ylabel(r'Field angle [degrees]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.xticks([0.0],[r'$0.0$'])
plt.yticks([8.565],[r'$0.0$'])
plt.tick_params( axis = 'both', direction = 'out' , width = 2,
length = 5, labelsize = 16)
plt.axis('equal')
plt.tight_layout()
plt.figure(4)
plt.show()
```

c) Mesh of rays that hit the primary mirror

```
print('')
```

```
print('')
print('')
print('')
plt.plot(pMirror_rayx, pMirror_rayy,'.', color = 'k', linestyle =
'none')
plt.xlabel(r'X [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.ylabel(r'Y [mm]', fontsize = 16, fontstyle = "normal",
family = 'serif')
plt.xticks([-1075,-717,-358, 0.0, 358, 717,1075],[r'$-1075$',
r'$-717$',r'$-358$',r'$0.0$',r'$358$',r'$717$',r'$1075$'])
plt.yticks([-1075,-717,-358, 0.0, 358, 717,1075],[r'$-1075$',
r'$-717$',r'$-358$',r'$0.0$',r'$358$',r'$717$',r'$1075$'])
plt.tick_params( axis='both', direction = 'out' , width=2, length=5,
labelsize=14)
plt.axis('equal')
plt.tight_layout()
plt.figure(5)
plt.show()
```

Bibliografía

Akram, S. and Ann, Q. U. (2015). Newton raphson method. *International Journal of Scientific & Engineering Research*, 6(7):1748–1752.

Bass, M., DeCusatis, C., Enoch, J., Lakshminarayanan, V., Li, G., Macdonald, C., Mahajan, V., and Van Stryland, E. (2009). *Handbook of Optics, Third Edition Volume I: Geometrical and Physical Optics, Polarized Light, Components and Instruments(Set)*. McGraw-Hill, Inc., USA, 3 edition.

Bowen, I. S. and Vaughan, A. H. (1973). The optical design of the 40-in. telescope and of the irenee dupont telescope at las campanas observatory, chile. *Appl. Opt.*, 12(7):1430–1435.

Bradt, H. (2004). *Astronomy methods: A physical approach to astronomical observations*. Cambridge University Press.

Bronshtein, I., Semendiaev, K., et al. (1973). *Manual de matemáticas para ingenieros y estudiantes*. Mir.

Carrasco, L., Hernández Utrera, O., Vázquez, S., Mayya, Y., Carrasco, E., Pedraza, J., Castillo-Domínguez, E., Escobedo, G., Devaraj, R., and Luna, A. (2017). Canica: The cananea near-infrared camera at the 2.1 m oagh telescope. *Revista mexicana de astronomía y astrofísica*, 53(2).

Carroll, B. W. and Ostlie, D. A. (2017). *An introduction to modern astrophysics*. Cambridge University Press.

Cottrell, G. (2016). *Telescopes: A Very Short Introduction*. Oxford University Press.

Demory, B.-O., Pozuelos, F., Chew, Y. G. M., Sabin, L., Petrucci, R., Schroffenegger, U., Grimm, S., Sestovic, M., Gillon, M., McCormac, J., et al. (2020). A super-earth and a sub-neptune orbiting the bright, quiet m3 dwarf toi-1266. *Astronomy & Astrophysics*, 642:A49.

Elazhary, T. T., Zhou, P., Zhao, C., and Burge, J. H. (2015). Generalized sine condition. *Appl. Opt.*, 54(16):5037–5049.

González, J. J., Ruiz, M. R., Lee, W. H., Watson, A. M., Bailón, E. J., Chew, Y. G. M., Sabin, L., Hiriart, D., Richer, M. G., and Rosales-Ortega, F. (2018). Recent developments at the oan-spm. In *Ground-based and Airborne Telescopes VII*, volume 10700, page 107005F. International Society for Optics and Photonics.

Gooch, J. W. (2011). *Sellmeier Equation*, pages 653–654. Springer New York, New York, NY.

Hecht, E. (2016). *Optics*. Pearson Education, Inc., 5 edition.

Herrera, J., Luna, E., Araiza-Durán, J. A., Salas, L., Sohn, E., Frayn, I. P., Ruiz, E., and Cruz-Gonzalez, I. (2017). Optical characterization of the active push–pull cell for the 2.1 m telescope at oan spm. *Applied Optics*, 56(5):1489–1494.

Hopkins, H. H. (1950). *Wave theory of aberrations*. Oxford at the clarendon press.

Karttunen, H., Kröger, P., Oja, H., Poutanen, M., and Donner, K. J. (2007). *Fundamental astronomy*, volume 4. Springer.

Kidger, M. J. (2001). Fundamental optical design. SPIE Bellingham.

Lehner, M. J., Wang, S.-Y., Reyes-Ruiz, M., Alcock, C., Castro, J., Chen, W.-P., Chu, Y.-H., Cook, K. H., Figueroa, L., Geary, J. C., Huang, C.-K., Kim, D.-W., Norton, T., Szentgyorgyi, A., Yen, W.-L., and Zhang, Z.-W. (2016). Status of the Transneptunian Automated Occultation Survey (TAOS II). In Hall, H. J., Gilmozzi, R., and Marshall, H. K., editors, *Ground-based and Airborne Telescopes VI*, volume 9906 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 99065M.

Luna, E., Salas, L., Gutiérrez, L., and nez, J. M. N. (2007). Geometric method to measure astigmatism aberration at astronomical telescopes. *Appl. Opt.*, 46(17):3439–3443.

Mahajan, V. N. (2014a). *Aberration Theory Made Simple*. Society of Photo-Optical Instrumentation Engineers (SPIE).

Mahajan, V. N. (2014b). *Fundamentals of geometrical optics*. Society of Photo-Optical Instrumentation Engineers (SPIE).

Malacara-Hernández, D. and Malacara-Hernández, Z. (2013). *Handbook of Optical Design, Third Edition*. Optical science and engineering. Taylor & Francis.

McLeod, B. A. (1996). Collimation of fast wide-field telescopes. *Publications of the Astronomical Society of the Pacific*, 108(720):217.

Melsheimer, F. M. and MacFarlane, M. J. (2000). Very wide field, very fast telescope. In Dierickx, P., editor, *Optical Design, Materials, Fabrication, and Maintenance*, volume 4003, pages 456 – 463. International Society for Optics and Photonics, SPIE.

Nájera, M. R., Herrera, J., and Guerrero, C. A. (2021). Mnajerar/sos-zcp: Sos-zcp.

PHASE, C. (2003). Setting new standards with harps. *The Messenger*, 114:20.

Richer, M. G., Lee, W. H., González, J., Jannuzi, B. T., Sánchez, B., Ortega, F. R., Alcock, C., Alonso, A. C., Díaz, M. T. G., Gutiérrez, L., Herrera, J., Hill, D., Norton, T. J., Pedrayes, M. H., Pérez-Calpena, A., Reyes-Ruíz, M., Guerrero, H. S., Sierra, G., Teran, J., Urdaibay, D., Uribe, J. A., Watson, A. M., Zaritsky, D., and Vargas, M. G. (2016). The Telescopio San Pedro Mártir project. In Hall, H. J., Gilmozzi, R., and Marshall, H. K., editors, *Ground-based and Airborne Telescopes VI*, volume 9906, pages 1979 – 1991. International Society for Optics and Photonics, SPIE.

Ruiz, E., Sohn, E., Salas, L., Luna, E., and Araiza-Durán, J. A. (2014). Common-pull, multiple-push, vacuum-activated telescope mirror cell. *Applied optics*, 53(33):7979–7984.

Schroeder, D. J. (1999). *Astronomical optics*. Elsevier.

Serway R. A., V. C. and S., F. J. (2013). *Fundamentos de física*. Cengage Learning.

Smith, W. (2000). *Modern Optical Engineering: The Design of Optical Systems*. Optical Engineering Series. McGraw Hill.

Spencer, G. H. and Murty, M. V. R. K. (1962). General ray-tracing procedure†. *J. Opt. Soc. Am.*, 52(6):672–678.

Stavroudis, O. N. and Sutton, L. E. (1965). *Spot diagrams for the prediction of lens performance from design data*, volume 93. US Department of Commerce, National Bureau of Standards.

Uribe, J., Bringas, V., Reyes, N., Tovar, C., López, A., Caballero, X., Martínez, C., Toledo, G., Lee, W., Carramiñana, A., et al. (2016). Mechanical conceptual design of 6.5 meter telescope: Telescopio san pedro mártir (tspm). In *Ground-based and Airborne Telescopes VI*, volume 9906, page 99062E. International Society for Optics and Photonics.

Van Rossum, G. and Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Wall, W. (2018). *A History of Optical Telescopes in Astronomy*. Springer.

Welford, W. T. (2017). *Aberrations of optical systems*. Routledge.

Wetherell, W. B. and Rimmer, M. P. (1972). General analysis of aplanatic cassegrain, gregorian, and schwarzschild telescopes. *Appl. Opt.*, 11(12):2817–2832.

Wilson, R. N. (2013). *Reflecting telescope optics II: manufacture, testing, alignment, Modern Techniques*. Springer Science & Business Media.

Ypma, T. J. (1995). Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551.