UNIVERSIDAD DEL PAPALOAPAN

CAMPUS LOMA BONITA

IMPLEMENTACIÓN DEL ALGORITMO DE BÚSQUEDA BINARIA PARA CALCULAR LA GEOMETRÍA DE LA PARED INTERNA DE UNA CAVIDAD ABIERTA POR UN PLANETA EMBEBIDO EN UN DISCO CIRCUMESTELAR

T E S I S

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

HÉCTOR MEDEL VILLAR

DIRECTOR DE TESIS:

DR. FRANCISCO RENDÓN ACOSTA

LOMA BONITA, OAXACA 2023

Universidad del Papaloapan

FECHA:	10 de Octubre del 2023
ÁREA:	Vice-Rectoría Académica
OFICIO NÚMERO:	UNPA/VAC/249/2023
ASUNTO:	Autorización de Impresión de Tesis.

C. HÉCTOR MEDEL VILLAR
P R E S E N T E:

En base al artículo 120 del reglamento de alumnos, por medio de la presente se aprueba la impresión de la tesis titulada ***"Implementación del algoritmo de búsqueda binaria para calcular la geometría de la pared interna de una cavidad abierta por un planeta embebido en un disco circumestelar"*** así como la programación del examen profesional bajo la dirección del Dr. Francisco Rendón Acosta.

Sin más por el momento aprovecho la ocasión para enviarle un cordial saludo.

Atentamente.
terra ubérrima, mens aperta
Bou Lo-tama, chi jí jú

VICE-RECTORIA ACADEMICA

DRA. TANIA ZUÑIGA MARROQUIN
Encargada de Despacho de la Vicerrectoría Académica.

C.c.p. Dr. Sergio Fabián Ruiz Pax.- Jefe de Carrera de Ing. en Computación
C.c.p. L.P. Yesenia Barrientos Arenal.- Jefa del Departamento de Servicios Escolares
C.c.p. Dr. Francisco Rendón Acosta.- Director de Tesis.
C.c.p. Archivo.

UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

Oficio: ICOM/10/002/23
Loma Bonita, Oaxaca, a 17 de octubre de 2023

M.E. Yesenia Barrientos Arenal
Jefa del Departamento de Servicios Escolares
PRESENTE

Mediante la presente, le informo que la jefatura de carrera a mi cargo, con visto bueno de la Vice-Rectoría Académica, ha designado a los siguientes profesores como sinodales para el examen profesional del egresado **C. Héctor Medel Villar**, quien defenderá su trabajo de tesis titulado ***"Implementación del Algoritmo de Búsqueda Binaria para Calcular la Geometría de la Pared Interna de una Cavidad Abierta por un Planeta Embebido en un Disco Circumestelar"***, para obtener el título de Licenciado en Ingeniería en Computación.

Titulares:

Presidente: **Dr. Sergio Fabián Ruiz Paz**
Secretario: **M.M. Iván Guadalupe Mendoza Alonso**
Vocal: **Dr. Francisco Rendon Acosta**

Suplentes:

M.C. Ariel López Rodríguez
Dra. Carolina G. Maldonado Méndez

A t e n t a m e n t e
terra uberrima, mens aperta
Bou lo-tama, chi, jí, jú

UNIVERSIDAD DEL PAPALOAPAN
UNPA
terra uberrima, mens aperta
Bou Lo-tama, chi jí jú
OAXACA
INGENIERIA EN COMPUTACION

Dr. Sergio Fabián Ruiz Paz
Jefe de la Carrera de Ingeniería en Computación

Dra. Tania Zúñiga Marroquín
Encargada de Despacho de la Vice-Rectoría Académica

c.c.p. Dra. Tania Zúñiga Marroquín. Encargada de Despacho de la Vice-Rectoría Académica. Para su conocimiento
c.c.p. Archivo

Esta tesis se la dedico a mis papás Héctor y Soledad, y a mis abuelos Ignacio y María Guadalupe.

Agradecimientos

Quiero agradecer a mis padres por el apoyo incondicional que me dieron a lo largo de mi carrera académica, por el apoyo moral en las noches de desvelo, por todos los sacrificios que hicieron para que yo esté en estos momentos titulándome.

A mi hermana la cual muchas veces me aconsejó sobre la vida académica y apoyó con alguna redacción de los trabajos que tenía que entregar y nunca me dejó solo.

A mi hermano, quien tuvo que soportar tantas noches con la luz prendida y no lo dejaba dormir.

A mi abuelo por todos los consejos, enseñanza y cuando tenía dudas y necesidad siempre tenía una respuesta positiva y dio su apoyo incondicional.

A mi abuela Lupe que aun ya no estando aquí influyó en la persona que me he convertido ahora.

A mi tía Isabel quien en momentos de necesidad me tendió la mano y me dio su apoyo.

A mi novia Julieta quien me impulsó día y noche las últimas semanas a terminar esta tesis.

A mi asesor Dr. Francisco Rendón por el apoyo incondicional en este proyecto, mi más grande agradecimiento.

A todos ellos ya que sin su apoyo no hubiera podido culminar esta etapa de mi vida académica.

Resumen

Un disco protoplanetario es un disco circunestelar que rodea una estrella joven, típicamente del tipo T Tauri. Estos discos en ocasiones se presentan en escenarios donde se llevan a cabo ciertos procesos físicos que conducen a la formación de planetas. Estos discos están compuestos principalmente de gas, polvo y son cruciales en el proceso de formación de sistemas planetarios. A medida que el material en el disco protoplanetario colapsa y se aglutina, se forman planetesimales que eventualmente pueden convertirse en planetas.

Un ejemplo de este tipo de discos es el sistema estelar LkCa 15, que es conocido por ser un objeto de estudio en la formación de sistemas planetarios. Se ha destacado por la observación de un disco en transición que rodea la estrella central y un posible planeta que ha sido capaz de abrir una cavidad en el disco.

Con la finalidad de establecer si el acompañante de LkCa 15 es un planeta, se ha desarrollado el código computacional ARTeMiSE2.0, escrito en el lenguaje de programación Fortran 90, con el que se pretende estudiar la geometría vertical de la cavidad abierta por el presunto planeta.

Parte de la elección del lenguaje Fortran 90 para la elaboración del código ARTeMiSE2.0 se debe a que este lenguaje introdujo algunas características orientadas a objetos, por lo cual es bueno implementar la programación modular y Programación Orientada a Objetos (POO) para cualquier proyecto.

Y debido a que el análisis del coste computacional es esencial para la optimización de algoritmos y la toma de decisiones sobre el enfoque que se va a utilizar en la resolución de cualquier tipo de problema computacional, en la presente tesis se acatan dos enfoques computacionales, los cuales son:

El primero trata sobre la estructura, mantenibilidad y legibilidad del código, para esto se implementa parte del código legible, modular y la implementación de programación orientada a objetos. Y el segundo, que es la implementación del mejor algoritmo para

la búsqueda con el menor coste computacional, pero sin dejar atrás la precisión que es parte importante para la obtención de los resultados.

Existen varios tipos de algoritmos de búsqueda que se utilizan en informática para encontrar un elemento específico en un conjunto de datos. Búsqueda Secuencial, Búsqueda Binaria, Búsqueda por Interpolación, Búsqueda Exponencial, Búsqueda Hash, Búsqueda en Árboles, por mencionar algunos.

El coste computacional del método de búsqueda secuencial en el peor caso es $O(n)$, por otro lado en el método de búsqueda binaria el coste computacional en el peor caso es $O(\log n)$, donde n es el número de elementos en la lista. La búsqueda binaria tiende a ser más rápida que la búsqueda secuencial, pero requiere que la lista esté ordenada.

Con la implementación del código ARTeMiSE2.0 a una simulación tridimensional de mediana resolución del sistema estelar LkCa 15 se obtuvo que la pared de la cavidad abierta por un planeta en formación es curva y concuerda con datos observados por Thalmann y colaboradores (1).

Palabras clave: Disco circunestelar, Fortran 90, Búsqueda Binaria, Programación Orientada a Objetos.

Abstract

A protoplanetary disk is a circumstellar disk surrounding a young star, typically of the T Tauri's type. These disks sometimes appear in settings where certain physical processes take place, which lead to planets' formations. These disks are composed mainly by gas and dust and are crucial in the process of planetary system formation. As the material in the protoplanetary disk collapses and clumps together, planetesimals are formed that can eventually become planets.

An example of such a disk is the LkCa 15-star system, which is known to be an object of study in planetary system formations. It has been highlighted by the observation of a transitional disk surrounding the central star and a possible planet that has been able to open a cavity in the disk.

To establish whether the LkCa 15's companion is a planet, the ARTeMiSE2.0 computer code, written in the Fortran 90 programming language, has been developed to study the vertical geometry of the cavity opened by the presumed planet.

Part of the choice of the Fortran 90 language for the elaboration of the ARTeMiSE2.0 code is because this language introduced some object-oriented features, therefore it is good to implement modular programming and Object-Oriented Programming (OOP) for any project.

Also, because computational cost analysis is essential for algorithm optimization and decision making about the approach to be used in solving any kind of computational problem, in the present thesis two computational approaches are followed, which are:

The first deals with the structure, maintainability and readability of the code, for this part of the readable code, modular and object-oriented programming implementation is implemented. And the second, which is the implementation of the best algorithm in order to search with the lowest computational cost, but without leaving behind the search accuracy, which is an important part for obtaining the results.

There are several types of search algorithms used in computer science to find a specific element in a data set. Sequential Search, Binary Search, Interpolation Search, Exponential Search, Hash Search, Tree Search, to name a few.

In the worst case scenario computational cost of the sequential search method is $O(n)$, on the other hand in the binary search method the worst case scenario in computational cost is $O(\log n)$, where n is the number of elements in the list. Binary search tends to be faster than sequential search but requires that the list be sorted.

With the implementation of the ARTeMiSE2.0 code to a medium-resolution three-dimensional simulation of the LkCa 15 stellar system it was obtained that the cavity wall opened by a forming planet is curved and agrees with data observed by Thalmann and collaborators (1).

Keywords: Disk Circumsteller, Fortan 90, Binary Search, Object-Oriented Programming.

Índice general

Capítulo 1

Introducción

1.1. Formación planetaria en discos circumestelares

1.1.1. Objetos estelares jóvenes

La formación estelar comienza con el colapso gravitacional de una nube molecular gigante de gas y polvo con una densidad de entre 10^{-17} y 10^{-16} g/cm^3. La nube se fragmenta en subnubes más pequeñas, y el gas y el polvo continúan contrayéndose en los núcleos centrales hasta alcanzar una densidad crítica de 10^{-13} g/cm^3, volviéndose ópticamente opacos. Esto provoca que la presión térmica aumente adiabáticamente y se alcance el equilibrio con la fuerza de gravedad. Estos objetos estables son llamados *protoestrellas* y representan la primera etapa de la formación estelar.

Se han establecido esquemas de clasificación estelar utilizando análisis de infrarrojo cercano (NIR) e infrarrojo medio (MIR); los más utilizados se basan en que los YSO (*Young Stellar Objets*, por sus siglas en inglés) evolucionan a través de una serie de etapas denominadas de Clase 0 a Clase III como consecuencia de los procesos de colapso y acrección (vease Tabla 1.1).

YSOs					
	Clase 0	Clase I	Clase II	Clase III	Secuencia principal
Edad (años)	10^4	10^5	10^6	10^7	$> 10^7$
Disco de acrección	opaco	opaco	opaco	transparente o sin disco	disco protoplanetario(?)
Emisión X	?	fuerte	fuerte	fuerte	débil
Chorros y emisiones	sí	sí	sí	no	no

Tabla 1.1: Características de los YSOs (2)

1.1.2. Discos circumestelares

El nacimiento de una proto-estrella deja como remanente un disco circumestelar de gas y polvo en diferentes composiciones físico-químicas que rota junto con ella (3), (4). La mayor parte de la materia que forma este disco servirá de alimento para la proto-estrella y así, potencialmente, se transforme en una estrella. Asimismo, es posible que dentro de este disco, se den las condiciones idóneas para que el material genere la formación de diferentes tipos de planetas, satélites, asteroides y otros cuerpos menores (5) , (6).

Observaciones realizadas por telescopios espaciales, que observan a los YSOs en diferentes bandas del espectro electromagnético, han revelado que los discos circumestelares presentan surcos, brechas o cavidades (7). En el sistema estelar HL Tauri se pueden apreciar dichas estructuras, véase la Figura 1.1.

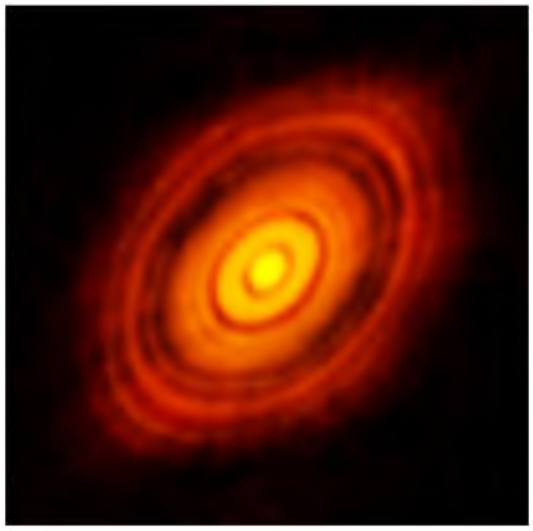


Figura 1.1: La estrella HL Tauri brilla en el centro de un sistema de anillos concéntricos hechos de gas y polvo y que producen planetas, uno para cada espacio en el anillo. Imagen tomada por el *Atacama Large Millimeter Array* (ALMA).

A estos discos también llamados discos protoplanetarios que presentan brechas en el interior se les denomina discos transicionales y hacen referencia a un posible estado intermedio de su evolución, una transición entre un disco joven (con gran contenido de polvo y gas) y un disco ya casi vacío de materia (8).

Existen diferentes mecanismos astrofísicos que explican la presencia de estas brechas; sin embargo, el mecanismo más común es el de la **formación planetaria** (9); (10);(11);(12);(13). Y es el que se aborda en la presente tesis.

1.1.3. Pared interna de una brecha

Los discos protoplanetarios en etapas de transición muestran espacios o agujeros en su estructura de polvo tridimensional (14), formando planetas gigantes incrustados en el disco (15). Estos espacios o agujeros están delimitados por una pared que puede afectar la distribución espectral de energía Strain Energy Density (SED por sus siglas en inglés) de todo el sistema (estrella más disco) en la banda del infrarrojo. Esta distribución se representa con una gráfica en la que se mide la cantidad de radiación emitida por el YSO por unidad de frecuencia o longitud de onda (16).

Los telescopios actuales, aunque sean muy sofisticados, no son capaces de resolver la estructura interna de los discos circumestelares, ya que los YSOs se encuentran demasiado lejos de la Tierra, por lo que no es posible saber directamente si la brecha fue abierta por un planeta en formación o no.

Es por ello que para saber si las cavidades han sido abiertas por planetas en formación, se deben realizar modelos teóricos del sistema estrella-disco-planeta, crear una SED sintética y compararla con la SED observada por algún telescopio en cierta banda del espectro electromagnético.

Se han realizado modelos de discos protoplanetarios en investigaciones previas, dando indicios que las paredes son verticales ((11); (17); (18)), véase la Figura 1.2. Sin embargo, esto es considerado físicamente incorrecto, al menos en el caso de una brecha abierta por un planeta en formación. La interacción gravitacional del planeta con la estructura vertical del borde interno del disco (en z) tiene una función importante: la distancia desde el planeta a cada porción de materia que define el límite del disco es diferente, y la densidad del disco disminuye a medida que z aumenta debido al equilibrio hidrostático, conduciendo así a producir la curvatura de la pared (19), obsérvese la Figura 1.3.

Con el objetivo de estudiar la formación planetaria acorde con la física gravitacional, en (19) se propone un modelo de pared curva (véase la Figura 1.3). Este modelo se basa en simulaciones hidrodinámicas tridimensionales del sistema estrella-disco-planeta en coordenadas esféricas.

Donde posteriormente, se analiza una sección transversal del disco en la posición del planeta, para encontrar la geometría de la pared estimando la profundidad óptica del disco, la cual se puede definir como:

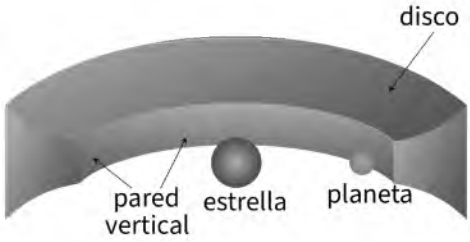


Figura 1.2: Pared vertical

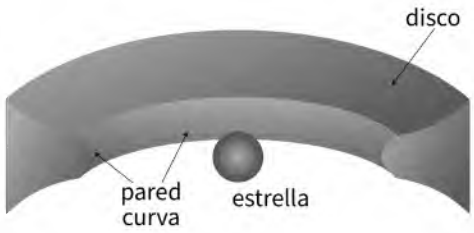


Figura 1.3: Pared curva

$$\tau(x,z) = \int_{\ell} \kappa(x,z)\rho(r,z)dl \tag{1.1}$$

donde κ y ρ son la opacidad y la densidad del disco, respectivamente, y ℓ es una línea recta que representa la radiación emitida por la estrella en forma de rayo. De este modo, la pared estará conformada por todos los puntos $(x;z)$ del disco para los que la profundidad óptica sea $\tau_{wall} = \frac{2}{3}$.

1.1.3.1. Opacidad promedio de Rosseland

El coeficiente general de absorción de energía del campo de radiación, en relación a la materia, se encuentra definido por la opacidad de absorción κ_ν, mismo que está en función de parámetros de densidad, temperatura y composición química con base al equilibrio termodinámico local.

La opacidad de absorción está determinada por la suma de la contribución de todas las partículas que componen la matria del disco:

$$\kappa_\nu = \sum_i \eta_i \sigma_i(\nu) \tag{1.2}$$

donde η_i es la densidad numérica de la i-ésima partícula y $\sigma_i(\nu)$ es la sección recta de absorción por partícula como función de la frecuencia ν.

Debido a que la opacidad de la atmósfera de los discos protoplanetarios muestra variaciones constantes de frecuencias, ésto puede conducir a buscar la solución de ecuaciones muy complejas. En este sentido, el uso de los cálculos de opacidad promedio es muy útil para simplificar el desarrollo de las soluciones matemáticas (20).

Específicamente, la opacidad promedio de Rosseland es útil para determinar una solución inicial a la estructura el disco para posteriormente utilizar una opacidad por

flujo monocromática. Sin embargo, la desventaja principal de la opacidad promedio de Rosseland en el caso de un disco o región de un disco de gas puro -libre de polvo-, es la formación de un disco transparente a su radiación propia, lo que conlleva a no utilizar la aproximación de difusión en el interior del disco. Esto quiere decir, que la aproximación sólo será efectiva si el disco es ópticamente grueso (20).
La ecuación de la opacidad de Rosseland se encuentra descrita a continuación:

$$\frac{1}{\langle \kappa^R \rangle} := \frac{\int_0^\infty \frac{1}{\kappa_\lambda} \frac{\partial B_\lambda}{\partial T} d\lambda}{\int_0^\infty \frac{\partial B_\lambda}{\partial T} d\lambda}, \tag{1.3}$$

donde κ_λ es la opacidad monocromática, $B_\lambda(T)$ es la función de Planck, y T es la temperatura del disco.

1.2. Herramientas matemáticas

Algunos conceptos básicos de geometría analítica son de gran importancia para realizar los modelos teóricos propuestos, ya que de las simulaciones se debe estudiar una sección transversal del disco tridimensional.

En otras palabras, se debe reconstruir una malla bidimensional en coordenadas cartesianas, misma que está constituida por una serie de sectores anulares, los cuales son el resultado de la intersección de segmentos de rectas y circunferencias como se muestra en la Figura 1.4.

Por estas razones surge la necesidad de aplicar algunos conceptos matemáticos en la presente tesis. Dichos conceptos se introducen a continuación.

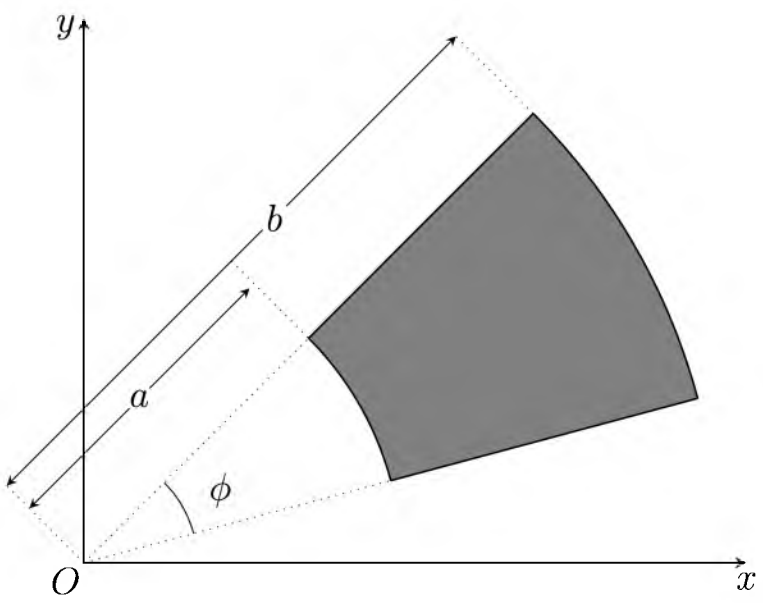


Figura 1.4: Sector anular. Donde ϕ es el ángulo de apertura, a y b son los radios de los dos arcos circulares que delimitan al sector siempre que $0 < a < b$.

1.2.1. Coordenadas rectangulares

La representación de pares ordenados de números reales mediante puntos en un plano se denomina sistema coordenado rectangular o plano cartesiano, mismo que fue introducido por el matemático francés René Descartes (1596-1650) (21).

El sistema coordenado rectangular consta de dos rectas de números reales que se intersecan de manera perpendicular: $x'x$ y $y'y$, llamadas ejes de coordenadas. La recta $x'x$ se llama eje x ; mientras que la $y'y$ se llama eje y; y su punto de intersección $(0, 0)$, es el origen. Estos ejes coordenados dividen al plano en cuatro regiones llamados cuadrantes, mismos que están numerados como se indica en la Figura 1.5. La dirección positiva del eje x es hacia la derecha y la del eje y hacia arriba (22).

La notación (x, y) indica tanto un punto en el plano como un intervalo abierto sobre la recta numérica real. Lo importante de un sistema coordenado rectangular es que permite ver relaciones entre dos variables (21).

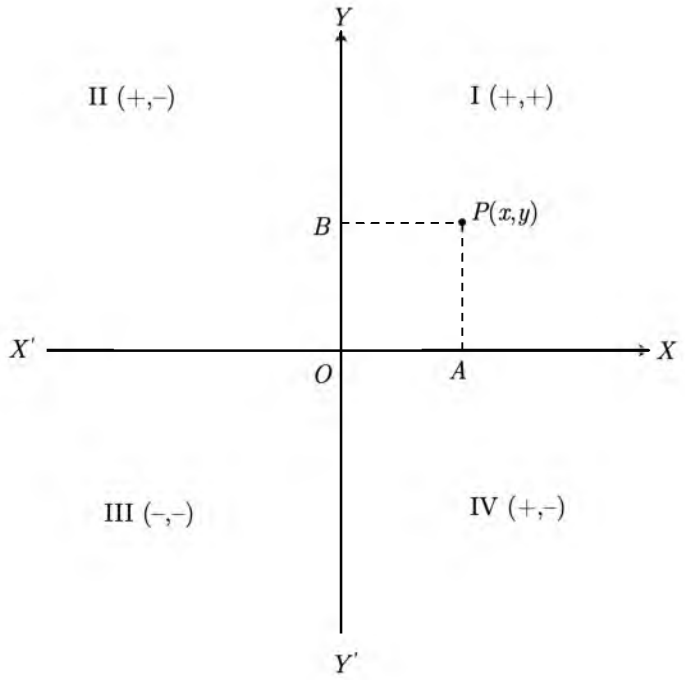


Figura 1.5: Sistema coordenado rectangular o plano Cartesiano.

1.2.2. Ecuación de la Recta

EL modelo matemático más simple para relacionar dos variables es la ecuación lineal de dos variables $y = mx + b$. Dicha ecuación se denomina lineal porque su gráfica es una recta (línea recta). Una línea recta es el lugar geométrico donde se encuentran los puntos tales que, tomados dos puntos diferentes cualesquiera del lugar, el valor de la pendiente m resulta siempre constante (Figura 1.6).

La pendiente de una recta no vertical es el número de unidades que la recta sube (o baja), de manera vertical, por cada unidad de cambio horizontal, de izquierda a derecha. Una ecuación lineal escrita en la forma $y = mx + b$ está escrita en la forma pendiente-intersección, donde $(0, b)$ es el punto de intersección con el eje y.

Dados dos puntos de una recta $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$, la pendiente se define como:

$$m = \frac{y_2 - y1}{x_2 - x1}, \text{ si } \quad x_1 \neq x_2 \tag{1.4}$$

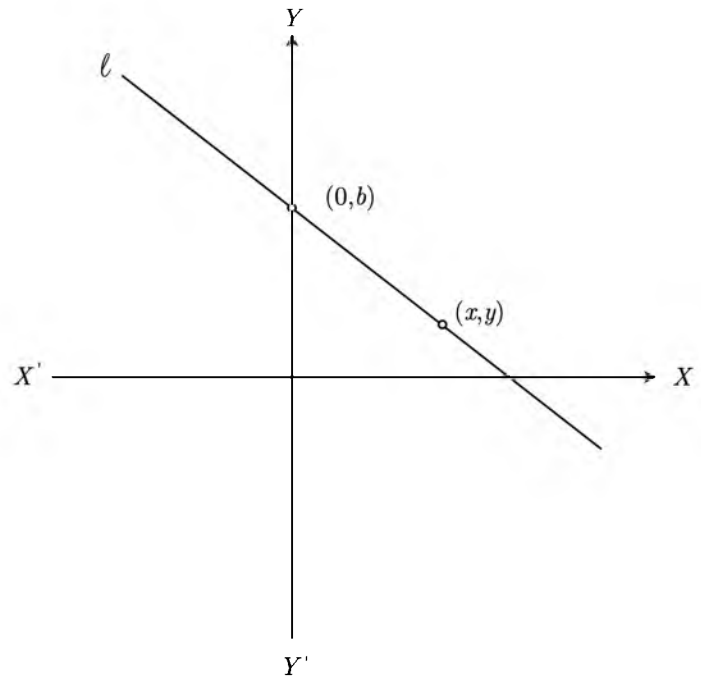


Figura 1.6: Representación de la recta en un plano cartesiano.

Ahora bien, ya que se tiene la pendiente m, se puede obtener la ecuación de la recta que pasa por el punto $P_1(x_1, y_1)$ de la siguiente forma:

$$y - y_1 = m(x - x_1) \tag{1.5}$$

o bien, la ecuación de la recta que pasa por el punto $P_2(x_2, y_2)$:

$$y - y_2 = m(x - x_2) \tag{1.6}$$

Nótese que, las ecuaciones 1.5 y 1.6 representan la misma recta ℓ (22).

Considérese ahora, que la recta ℓ en la Figura 1.6 tiene una pendiente m y una ordenada al origen dada por b, es decir, la recta pasa por el punto $(0; b)$, entonces, según las ecuaciones 1.5 o 1.6 la ecuación de la recta es:

$$y = mx + b, \tag{1.7}$$

donde $b = y_1 - mx_1 = y_2 - mx_2$

La ecuación anterior es la forma pendiente–ordenada al origen de la recta.

Este resultado se enuncia en el siguiente teorema:

Teorema 1. *La recta cuya pendiente es m y cuya ordenada en el origen es b tiene por ecuación*

$$y = mx + b \tag{1.8}$$

1.2.3. Ecuación de la circunferencia

La circunferencia se denomina como el conjunto de puntos que están a una distancia fija de un punto fijo (23).

El punto fijo C es el centro de la circunferencia, y el segmento de recta que une a cualquier punto de la circunferencia con el centro recibe el nombre de radio y se denota con la letra r, como se observa en la Figura 1.7 (22).

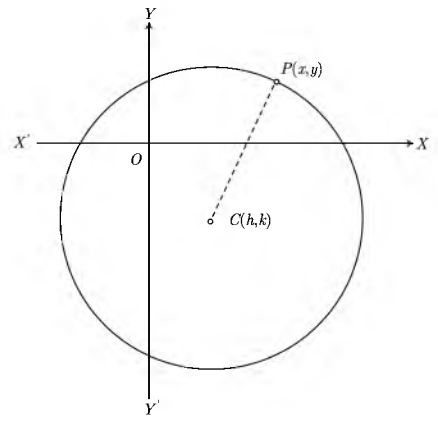


Figura 1.7: Representación gráfica de la circunferencia.

Teorema 2. *La circunferencia cuyo centro es el punto (h, k) y cuyo radio es la constante r tiene por ecuación:*

$$(x - h)^2 + (y - k)^2 = r^2 \tag{1.9}$$

Cuando el centro de la circunferencia está localizado en el origen, es decir, en la coordenada $(0, 0)$, entonces $h = k = 0$, por lo que la ecuación se modifica, y se tiene lo siguiente:

Corolario 1. *La circunferencia con centro en el origen y radio r tiene por ecuación (22).*

$$x^2 + y^2 = r^2 \tag{1.10}$$

1.2.4. Coordenadas Polares

A través de un sistema de coordenadas en un plano, es posible localizar cualquier punto de ese plano. Como se mencionó anteriormente, en el sistema rectangular esto se realiza refiriendo el punto a los dos ejes de coordenadas perpendiculares; no obstante, en el sistema polar, un punto se localiza especificando su posición relativa con respecto a una recta fija y a un punto fijo de esa recta. La recta fija se llama eje polar y el punto fijo se denomina polo (24).

En el sistema de coordenadas polares, las coordenadas del punto $P(r, \theta)$ están representadas por dos valores, en este caso r representa una distancia, mientras que θ representa un ángulo.

El punto O en la Figura 1.8 es el origen o polo y el segmento de recta dirigido o rayo OA es el eje polar (equivalente al eje x del sistema cartesiano). Sean r la longitud del segmento de recta OP y θ el ángulo formado entre el eje polar y el segmento de recta dirigido OP, entonces todo punto P del plano corresponde a un par ordenado (r, θ) donde θ crece en sentido anti-horario y decrece en sentido horario. La distancia r también suele llamarse radio del vector, mientras que θ es el ángulo polar (22).

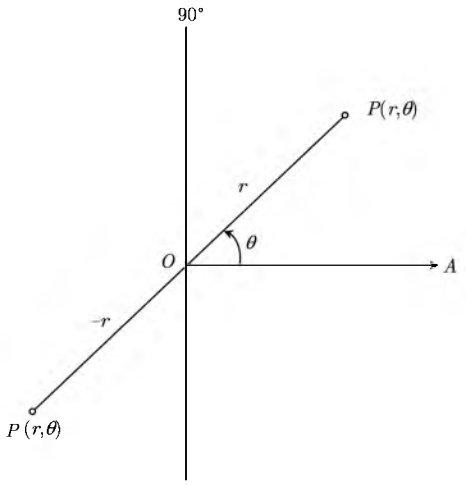


Figura 1.8: Representación geométrica de un punto P en coordenadas polares.

1.2.5. Transformación de coordenadas polares a rectangulares

Para un lugar geométrico determinado es conveniente saber transformar la ecuación polar en una ecuación rectangular, y recíprocamente.

Dado que, las coordenadas rectangulares (x, y) de cualquier punto de un plano implican solamente dos variables (x y y) la ecuación de cualquier lugar geométrico en un sistema de coordenadas rectangulares en un plano, contiene una o ambas de estas variables, pero no otras. Por esta razón, resulta oportuno llamar a una ecuación de esta clase como ecuación rectangular del lugar geométrico.

De la misma manera, las coordenadas polares (r, θ) de cualquier punto de un plano implican solamente dos variables, r y θ, de manera que la ecuación de cualquier lugar geométrico en el plano coordenado polar contiene una o ambas variables, pero no otras. Dicha ecuación se llama, de acuerdo a este concepto, la ecuación polar del lugar geométrico. Así $\theta = \frac{\pi}{4}$ y $r = 4\cos(\theta)$ son las ecuaciones polares de dos lugares geométricos planos.

Primeramente, para llevar a cabo la transformación de ecuaciones polares a rectangulares, es importante conocer las relaciones que existen entre las coordenadas polares y rectangulares de cualquier punto del lugar geométrico. Para ello, se toman en consideración los aspectos enunciados del teorema a continuación:

Teorema 3. *Si el polo y el eje polar del sistema de coordenadas polares coinciden, respectivamente, con el origen y la parte positiva del eje x del sistema de coordenadas rectangulares, el paso de uno a otro de estos dos sistemas puede efectuarse por medio de las siguientes formulas de transformación:*

$$x = r\cos(\theta) \tag{1.11}$$

$$y = r\sin(\theta) \tag{1.12}$$

$$x^2 + y^2 = r^2 \tag{1.13}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \tag{1.14}$$

$$r = \pm\sqrt{x^2 + y^2} \tag{1.15}$$

$$\sin(\theta) = \pm\frac{y}{\sqrt{x^2 + y^2}} \tag{1.16}$$

$$\cos(\theta) = \pm\frac{x}{\sqrt{x^2 + y^2}} \tag{1.17}$$

1.2.6. Algoritmos de búsqueda

Un algoritmo de búsqueda es aquel que permite encontrar la ubicación de algún dato en un arreglo o matriz. Esta búsqueda toma como parámetros las características que el usuario defina para que el algoritmo encuentre el dato dentro de la estructura de datos. (25).

Existen varios tipos de algoritmos de búsqueda, entre los más comunes se encuentran: de búsqueda lineal, por transformación de claves, en textos, árboles de búsqueda, secuencial y binaria, donde los últimos dos son los que se describen a continuación:

1.2.6.1. Búsqueda secuencial

El algoritmo de búsqueda secuencial hace referencia a buscar un valor determinado en un vector, esta búsqueda se ejecuta cumpliendo las siguientes condiciones: que el algoritmo se ejecute hasta encontrar el valor que se busca o en dado caso parar hasta encontrarse en la última posición del vector, sin haber encontrado el valor buscado (25).

Con este algoritmo se pueden tener dos escenarios, el primero es que el valor que se está buscando no se encuentre en el vector, y el segundo es que se logre encontrar dicho valor; por lo cual se procede a indicar la posición dentro del vector en la que se encuentra dicho elemento. Este algoritmo es el más popular y más básico de los algoritmos de búsqueda.

1.2.6.2. El algoritmo de búsqueda binaria

Un algoritmo clásico sobre un vector ordenado es la búsqueda binaria o dicotómica. Para la búsqueda de un elemento en particular se divide el vector en dos partes y se consulta si el elemento se sitúa en la parte derecha o izquierda, en función del orden con el elemento central. Posteriormente, el algoritmo selecciona la parte del vector donde debía estar el elemento; dicho proceso, es repetido tantas veces hasta que el vector toma un valor de cero o finalmente logra encontrar el elemento en cuestión (26).

Algunos puntos clave a considerar para analizar e implementar los códigos del algoritmo son, en primera instancia, que el algoritmo depende del tamaño de entrada n del problema; por otro lado, para un mismo tamaño se pueden obtener distintos resultados, ya que el número de pasos que son necesarios para ejecutar la función depende del elemento x buscado (26).

El algoritmo de búsqueda binaria, según (25) se puede describir de la siguiente forma[2]:

```
1: procedure BUSQBIN(A,n,x,j) do
2:     inferior ← 1 ; superior ← n
3:     while inferior ⩽ superior  do
4:         media ← [(inferior+superior)/2]
5:         case :
6:             x>A(media): inferior ← media+1
7:             :x<A(media): superior ← media-1
8:             :si no: j ← media; retornar
9:         end case
10:    end while
11:    j ← 0
12: end procedure
```

1.2.7. Notación asintótica

En el área de la computación se pasa mucho tiempo mejorando un algoritmo, tratando de hacer la mejor implementación de ciertos conocimientos aplicados a un problema, intentando hacer que el algoritmo que use la menor cantidad de memoria, el menor número de iteraciones y también que resuelva el problema en el menor tiempo posible. Esto se consigue desde cómo se guarda la información, cómo se lee, o cómo se ordena la información e incluso en qué orden se ejecutan los algoritmos.

La finalidad de encontrar la eficiencia y rapidez de un algoritmo, ha llevado a la comunidad científica a desarrollar varios métodos para analizar los algoritmos. Una manera considerablemente fácil para cumplir con dicha finalidad, es incorporar un contador de tiempo al inicio y al final de la función o del programa que se desea medir; posteriormente, ejecutarlo un número determinado de veces y finalmente hacer una media. De esta forma, se obtendría un panorama amplio sobre el comportamiento del algoritmo; no obstante, esto continúa siendo insuficiente, debido a que muchas veces se puede ejecutar el mismo algoritmo en diferentes computadoras y el resultado tenderá a no ser el mismo.

Debido a que la ejecución del logaritmo depende del hardware del computador, es que

[2]Por la naturaleza de la investigación y el área del conocimiento en la que se enmarca se entiende que los algoritmos de programación suelen estar siempre escritos en inglés, con sus variables y sentencias, esto a su vez ayuda al lector a entender más rápido el algoritmo porque aparece en la sintaxis original y no se confunde con la traducción de los ciclos o bucles. Por lo tanto, en una decisión metodológica, mantendré la escritura del algoritmo en su idioma original.

se quiere encontrar o definir un concepto que describa de una manera más general el comportamiento de un algoritmo. Así, se podría calificar varios algoritmos bajo el mismo criterio y de este modo, poder comparar cuál es el mejor o cuál es el idóneo para la resolución del problema. En particular, esto es lo que se conoce como notación asintótica. Existen varias notaciones asintóticas como O-grande, θ-grande, Ω-grande , ω-pequeña y la o-pequeña pero en esta tesis se utilizará únicamente la notación, O-grande, o también conocida como Big-O.

La notación asintóticas es una manera facil de comprender el tiempo de ejecución de un algoritmo. Dado que se quiere conocer la eficiencia de un algoritmo y solamente se puede conocer en el momento en el que las entradas o el tamaño de la entrada es lo suficientemente grande, se utilizará un análisis asintótico. Un análisis de eficiencia asintótico permite saber cómo se comporta un algoritmo cuando el tamaño de la entrada es lo suficientemente grande; además, permite ignorar términos de orden menor que permite, con una sola función, describir cómo se comporta el algoritmo ante una entrada n grande y luego eliminar los coeficientes constantes que dependen de la computadora en que se ejecuta.

En otras palabras, el análisis asintótico permite utilizar solo los términos de mayor orden para representar o entender cómo el algoritmo se comportará para entradas lo suficientemente grandes. Cuando se emplea la notación asintótica, se hace referencia a cómo una función crece o cómo el tiempo de ejecución del algoritmo crece con respecto a una variable n (siendo n, el tamaño de la entrada) 1.9.

La notación asintótica O-grande permite definir una cota superior de ajuste, traduciéndose a que todas la funciones deben cumplir la siguiente condición:

Se denota por $O(f(n))$ que se pronuncia O *de* $f(n)$ al conjunto de todas las funciones $t : \mathbb{N} \to \mathbb{R}^{\geqslant 0}$ tales que $t(n) \leqslant cf(n)$ para todo $n \geqslant n_0$ para una constante real positiva c y un umbral entero n_0. (27) Dicho de otro modo, todas las funciones que cumplan la siguiente de definición:

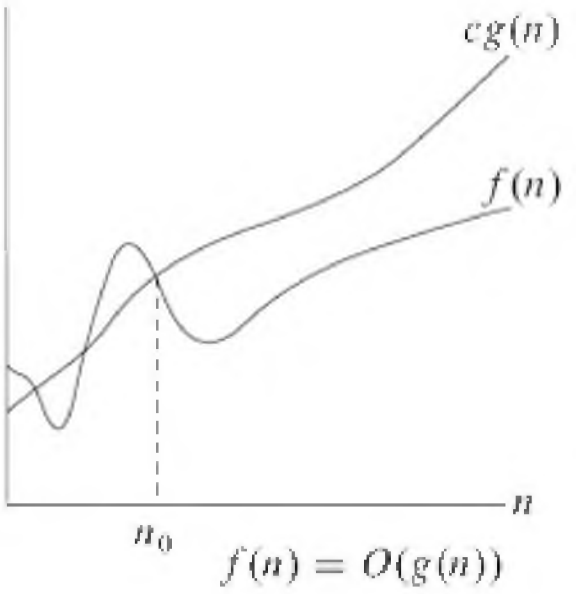


Figura 1.9: Gráfica representativa de la Notación asintótica.

Definición 1. $O(f(n)) = \left\{ t : \mathbb{N} \to \mathbb{R}^{\geq 0} | (\exists c \in \mathbb{R}^{+})(\overset{\infty}{\forall} n \in \mathbb{N}) \left[t(n) \leqslant c f(n) \right] \right\}$

Esto quiere decir que existe una función $f(n)$ en la que a partir de un punto n_0 siempre es inferior o igual a una función de la forma $cg(n)$, esto se refiere a todas aquellas funciones que cuando tienden a infinito acaban siendo inferiores o iguales a alguna función que sea lineal con c positiva.

1.3. FORTRAN 90

Cuando un computador ejecuta un programa, implementa una cadena de operaciones muy simples como cargar, almacenar, sumar, restar, multiplicar, etc. Cada operación tiene un único patrón binario llamado operación de código (op code) para especificarlo. Las operaciones de código son conocidas como lenguaje máquina, ya que son el verdadero lenguaje que un computador reconoce y ejecuta (28).

En otras palabras, el programa que un computador ejecuta es solo una cadena de instrucciones, en lenguaje máquina, ordenadas específicamente para alcanzar un propósito.

Sin embargo, para los humanos, nos resulta muy difícil trabajar con lenguaje máquina. Por esto, es preferible utilizar instrucciones en un idioma más cercano al lenguaje natural y ecuaciones algebraicas, ambas expresadas en formas que nos resultan entendibles,

en lugar de patrones binarios. Esto es programación con lenguaje de alto nivel. Se transcriben las instrucciones en un lenguaje de alto nivel y son convertidas en lenguaje máquina, por programas especiales llamados compiladores y enlazadores, para que el computador comprenda dichas órdenes (28).

Existe una gran variedad de lenguajes de alto nivel con diferentes características. Algunos están diseñados para funcionar bien para problemas financieros, otros están diseñados para uso científico. Otros son especialmente adecuados para aplicaciones como la programación de sistemas operativos. Es importante elegir un idioma adecuado para que coincida el problema que está tratando de resolver (28).

Algunos lenguajes informáticos de alto nivel comunes en la actualidad incluyen Ada, C, C++, Fortran, y Java. Históricamente, Fortran ha sido el lenguaje predominante para los estudios científicos generales. Ha existido de una forma u otra desde los años cincuenta, y se ha utilizado para implementar todo lo referente a esta área, desde modelos informáticos de plantas de energía nuclear hasta programas de diseño de aeronaves para sistemas de procesamiento de señales sísmicas, incluyendo también a algunos proyectos que requirieron millones de líneas de código. El lenguaje es especialmente útil para análisis numéricos y cálculos técnicos. Además, Fortran es el idioma dominante en el mundo de las supercomputadoras y las computadoras masivamente paralelas (28). Su nombre proviene de *Formula Translation* y como se pude entender, fue creado para poder facilitar la traducción de formulas matemáticas.

El lenguaje Fortran es un lenguaje dinámico que está en constante evolución para mantenerse al día con los avances en la práctica de la programación y la tecnología informática. Sale a la luz una nueva versión importante aproximadamente una vez por década (28).

1.3.1. Objetivos

1.3.1.1. objetivo general

Encontrar la pared interior de una cavidad abierta por un planeta en formación, basado en el algoritmo de búsqueda binaria.

1.3.1.2. objetivo especifico

1. Implementar un algoritmo computacional que reconstruya una sección transversal bidimensional del disco mediante una malla en coordenadas cartesianas compuesta

por sectores anulares, a partir de los datos de una simulación tridimensional en coordenadas esféricas de un sistema estrella-disco-planeta.

2. Crear un algoritmo computacional basado en el algoritmo de búsqueda binaria para encontrar los puntos que constituyen la pared de la cavidad en el disco, utilizando la malla bidimensional.

3. Implementar el código desarrollado en el sistema estelar LkCa15 para analizar la geometría de la cavidad.

4. Comparar los algoritmos de búsqueda binaria y búsqueda secuencial para conocer cual es mas eficiente para encontrar los puntos que definen la pared de la cavidad.

Capítulo 2

Reconstrucción de la sección transversal del disco en forma de una malla bidimensional

En este capítulo se encuntra representado el modelo y los algoritmos físico-matemáticos que representa la solución descrita con el lenguaje de programación Fortran, la cual se ve involucrada a lo largo de la tesis. Dicho modelo descrito a continuación, lleva el orden en cómo se ve implementada a lo largo del código. También encontraremos la primer parte del motivo de este trabajo, el cual es la obtención de los puntos los cuales son los vértices y las intersecciones de cada punto centro de la malla bidimensional, la cual es un preámbulo a la parte más técnica e importante de dicho trabajo, que es el poder encontrar los puntos de la pared curva.

2.1. Algoritmo Matemático

El siguiente algoritmo describe la solución matemática para encontrar los puntos de una pared curva, desde el ordenamiento de los datos de entrada hasta la obtener las coordenadas de los puntos de la pared.

El algoritmo se describe a continuación.

Considerando como *input* del programa las siguientes variables:

$\mathbf{r_{max}}$, $\mathbf{r_{min}}$, $\theta_{\mathbf{min}}$, $\theta_{\mathbf{max}}$, $\mathbf{m}$,$\mathbf{n}$ y los puntos de la simulación.

Se define:

$$\mathbf{\Delta_r} = \frac{r_{max} - r_{min}}{n}$$

y

$$\mathbf{\Delta_\theta} = \frac{\theta_{max} - \theta_{min}}{m}$$

Enlistamos los pasos a seguir para la obtención de los puntos de pared.

Paso 1.- Asignar el centro C en el sector anular que corresponde.

Como primer instancia para asignar el centro C al sector anular j, k con base en (r, θ), el algoritmo usa la siguiente funcionalidad:

Usando un contador $q = 1, ..., n$, se pregunta si el radio r del centro C cumple la siguiente condición:

$$\mathbf{r_{min} + (q - 1)\Delta_r < r < r_{min} + q\Delta_r}$$

Si para cierto valor Q, se cumple lo anterior, entonces el centro esta en el sector Q, l.

Se repite el proceso con la condición para el angulo θ del centro C con $p = 1, ..., m$.

$$\mathbf{\theta_{min} + (p - 1)\Delta_\theta < \theta < \theta_{min} + p\Delta_\theta}$$

Si para cierto valor P se cumple lo anterior, el algoritmo situa al centro C en el sector anular $\mathbf{Q, P}$ y le otorga el nombre de $\mathbf{C_{q,p}}$.

Sean $C_{j,k}(r, \theta)$ lo puntos coordenados *inpunt's* del programa, estos representan el centro de un sector anular.

Paso 2.- Calcular vértices e intersecciones del sector anular.

Un sector anular esta compuesto por 4 vértices y 2 intersecciones. Por lo que para cada $C_{j,k}(r, \theta)$, calculamos dichos componentes antes mencionados:

$V^l_{j,k}$ donde $l = 1, 2, 3, 4$

Vértices del sector anular que tiene como centro $C_{j,k}$

$I^l_{j,k}$ donde $l = 1, 2$

intersecciones del sector anular que tienen como centro $C_{j,k}$

Para el calculo del vértice

$V^1_{j,k}$ tiene como coordenadas

$(r_{max} + (k-1)\Delta_r, (j-1)\Delta_\theta)$

$V^2_{j,k}$ tiene como coordenadas

$(r_{max} + (k-1)\Delta_r, j\Delta_\theta)$

$V^3_{j,k}$ tiene como coordenadas

$(r_{max} + k\Delta_r, (j-1)\Delta_\theta)$

$V^4_{j,k}$ tiene como coordenadas

$(r_{max} + k\Delta_r, j\Delta_\theta)$

Para calcular las intersecciones $I^l_{j,k}$

$I^1_{j,k}$ tiene como coordenadas:

$(r_{min} + (k-1)\Delta_r, \frac{\Delta_\theta}{2} + (j-1)\Delta_\theta)$

$I^2_{j,k}$ tiene como coordenadas:

$(r_{min} + k\Delta_r, \frac{\Delta_\theta}{2} + (j-1)\Delta_\theta)$

Para cada $C_{j,k}$ viene una cantidad llamada densidad.

Paso 3.- Obtención de los puntos que componen la pared curva.

Una vez ubicados los puntos que componen cada sector anular(vértices e intersecciones), se procede a encontrar los puntos de la pared curva.

Un punto de la pared curva puede estar ubicado en una fila de sectores anulares.

Dichos sectores anulares j,k posee un valor fijo de $p_{j,k}$ Dejando el valor k fijo,se va sumando sobre j hasta que la suma de densidades es menor a $\tau = 2/3$

$$p_j = \sum_{a=1}^{k} p_{j,a} \leqslant \tau$$

Para el valor mas grande de k en el que se cumple lo anterior, se toma el sector anular j,k y se comienza con el segundo algoritmo.

En el sector anular $j,k+1$, hacemos una partición de intervalo Δ_r desde $r_{min}+(k+1)\Delta_r$ hasta $r_{min} + k\Delta_r$ a la mitad.

Cada una de estas partes posee una densidad molecular $p^1_{j,k+1}, p^2_{j,k+1}, ...$

Se calcula

$p = p_j + p^i_{j,k}$

se hacer la pregunta

si $p = \tau$,se guarda el la coordenada r del punto $r = r_{min} + (k+1)\Delta_r + \frac{\Delta_r}{2}$

En caso contrario,se pregunta si $p < \tau$, entonces se repite el proceso anterior ahora partiendo el intervalo $[r_{min} + (k-1)\Delta_r + \frac{\Delta_r}{2}, r_{min} + k\Delta_r]$, de lo contrario se toma el siguiente intervalo:

$[r_{min} + (k-1)\Delta_r, r_{min} + (k-1)\Delta_r + \frac{\Delta_r}{2}]$

se hacen n iteraciones de este algoritmo sobre los 2^{n-1} intervalos que se crean hasta alcanzar $p = \tau$

2.2. Sintaxis del programa: Fortran

La revisión se describe conforme al orden en cómo se ejecuta el flujo del archivo *main*. A continuación, se mencionan y explican, las instrucciones que se encuentran en el módulo llamado ***archivo_class***, las cuales son las más destacadas:

En la subrutina ***leerArchivo***

open (access = 'stream', & action = 'read', & file = this%direccion, & form = 'unformatted', & iostat = rc, & newunit = fu)

Esta instrucción nos sirve para poder abrir un archivo. Existen muchas formas de abrir un archivo y puede llevar diferentes parámetros, en nuestro caso la estructura fue:

access método de acceso para la conexión del archivo.

action especificamos la acción de entrada/salida del archivo: lectura / escritura.

file representa el nombre del archivo.

form que se refiere al tipo de formato en que vamos a abrir nuestro archivo.

iostat que una variable en la cuela se guarda si fue exitosa la lectura o no.

newunit Especifica un valor entero para la conexión .

Para nosotros esta fue la mejor forma que se adapto a nuestras necesidades.

inquire (fu, size=n) : Esta instrucción nos sirve para asignar el tamaño de la variable fu, en n, para poder tener el tamaño en caracteres de nuestro archivo de entrada.

read (fu, iostat=rc) buffer : Esta instrucción nos permite leer un archivo fu y lo guardamos en la variable buffer

close (fu) : Instrucción para poder cerrar un archivo que se ocupado o hemos abierto

en el modulo archivo_class encontraremos algunas veces la IACHAR o ACHAR Estas instrucciones significan lo siguiente:

IACHAR(a) retorna el código del caracter ASCII(un número entero) ACHAR(i) retorna el caracter ubicado en la posición i en código ASCII. En otras palabras,IACHAR nos devuelve que valor en decimal tiene ese caracter en el código ASCII y ACHAR le mandamos el numero que queremos que nos convierta a caracter, estas dos instrucciones son muy útiles y se estarán usando en ese módulo en particular.

2.3. Obtención de los datos

El éxito de un buen programa, o bien, la culminación de un proyecto en cualquier área de trabajo, es un buen análisis, y en la rama de la computación no es la excepción. Realizar un buen análisis del problema es sumamente importante, ya que nos proporciona un mejor panorama de cómo solucionarlo. Por esto, en este trabajo se realizó un análisis minucioso de todo lo que se encuentra en este proyecto de investigación. En este capítulo se nombran y se explican las implementaciones que se consideran más importantes y que hacen la diferencia para un mejor desempeño de este código computacional. Se considera que a mayor número de datos de entrada (input), es más factible ver reflejada una buena estructura y optimización del código. Tales optimizaciones y estructura de los datos que se consideraron como mejor opción, se justifican a lo largo del capítulo. Sin embargo, damos por hecho que las optimizaciones implementadas no son las únicas posibles para resolver nuestro problema, ya que un código computacional siempre se puede mejorar de una u otra forma.

Otros comandos básicos que se utilizan varias veces a lo largo de este documento, se explican a continuación:

Sentencia **if**:

```
if(contador==10)then
! Sentencias a ejecutar
```

```
end if
```

Siendo esta la sentencia básica condicional, como en muchos otros lenguajes de programación, la escritura se basa en utilizar la palabra reservada **if**, abriendo paréntesis, dentro de ellos, vamos a declarar nuestra condición que en caso de cumplirse ejecutará el código que se encuentre dentro del **then** y el **end if**:

Sentencia **if-else**:

```
if(contador==10)then
! Sentencias a ejecutar
else
! Sentencias a ejecutar
end if
```

Este es un casi simular como el **if**, solo que aquí, como podemos notarlo ahora vemos la palabra reservada **else**, la cual nos sirve para ejecutar un código cuando la condición del **if** no se cumpla:

Sentencia **else-if**

```
if(contador==10)then
! Sentencias a ejecutar
else if(contador>=10)then
! Sentencias a ejecutar
end if
```

Por último, una forma que nos sirve por si tenemos que hacer varias preguntas sin tener la necesidad de llegar a un **swich-case** es el **else-if**, que esto es solo estar haciendo preguntas para poder saber si cae en alguna condición o cumple alguna condición, esta estructura se puede hacer igual que un **if-else**, solo que aquí, inmediatamente después del el agregamos otra condición **if() then** y ya no tendríamos que agregar un segundo **end if** .

Declaración de un ciclo do

```
do i=1,n
! bloque de instruciones
end do
```

Para ciclos o iteraciones podemos usar la sintaxis **do**, propia de Fortran, la cual está compuesta por la palabra reservada do seguida de la variable de control y la

inicialización de la variable seguida de una coma y el límite de nuestro ciclo. Este límite puede ser un número entero o una variable, previamente inicializada, dentro, tendremos el código que queremos ejecutar ***n*** veces, y cerramos la sentencia con un **end do**.

Con respecto a la declaración de una clase en Fortran, lo primero que debemos hacer para poder crear una clase es iniciar nuestro programa con la palabra reservada MODULE, seguida del nombre que queremos que reciba nuestra clase y el final del archivo debe llevar las palabras reservadas END MODULE. Posteriormente se inserta el nombre del archivo que pusimos al inicio del archivo. Después se guarda el archivo con el mismo nombre que le hemos puesto a nuestra clase, más la extensión ***.f90***. Ahora bien, dentro de estas dos instrucciones:

```
MODULE coor_class
PUBLIC
TYPE punto_ob
        PRIVATE
        real(8) :: x
        real(8) :: y
CONTAINS
PRIVATE
PROCEDURE          :: initPunto
PROCEDURE,PUBLIC :: getX
PROCEDURE,PUBLIC :: getY
PROCEDURE,PUBLIC :: setX
PROCEDURE,PUBLIC :: setY
PROCEDURE,PUBLIC :: imprimirPunto

END TYPE punto_ob

!aniadir nuevos metodos
CONTAINS

FUNCTION punto(x,y)
        TYPE(punto_ob) :: punto
        real(8) :: x
        real(8) :: y
        call punto%initPunto(x,y)
```

```
END FUNCTION punto
SUBROUTINE initPunto(this,x,y)
        class(punto_ob) :: this
        real(8) :: x
        real(8) :: y
        this%x=x
        this%y=y
END SUBROUTINE initPunto

real(8) FUNCTION getX(this)
        class(punto_ob) :: this
        getX=this%x
END FUNCTION getX


real(8) FUNCTION getY(this)
        class(punto_ob) :: this
        getY=this%Y
END FUNCTION getY

SUBROUTINE setX(this,x)
        class(punto_ob) :: this
        real(8) :: x
        this%x=x
END SUBROUTINE setX

SUBROUTINE setY(this,y)
        class(punto_ob) :: this
        real(8) :: y
        this%y=y
END SUBROUTINE setY


SUBROUTINE imprimirPunto(this)
        class(punto_ob) :: this
        print*,'(',this%x,',',this%y,')'
```

```
END SUBROUTINE imprimirPunto
END MODULE coor_class
```

Escribieremos la estructura de nuestra clase

Instanciar una clase TYPE(archivo_ob) :: **arch_lectura** Nos referimos a instanciar una clase cuando creamos copia de un objeto y

Cómo acceder a un método de un objeto

call ***inquirearch_lectura %leerArchivo()*** Haciendo uso de la palabra reservada call seguida del nombre del objeto creado que en este caso es un objeto de tipo ***inquirearchivo_class*** llamado ***inquirearchivo_lectura***

2.4. Lectura de datos

En la actualidad, Fortran es considerado un lenguaje de bajo nivel, ya que no existen librerías o subrutinas incluidas en el lenguaje que nos ayuden o faciliten la obtención de datos u otro tipo de herramientas para diferentes actividades. Fortran sólo cuenta con funciones intrínsecas, esta es una de las principales causas por las que algunos programadores o investigadores optan por no usarlo. Sin embargo, es destacable que al encontrarnos frente a un lenguaje de bajo nivel estamos más cerca del lenguaje máquina.

Lo anterior significa que la ejecución de nuestro código tarda menos en ejecutarse que las instrucciones de en un lenguaje alto nivel, esto lo tomaremos como una ventaja y crearemos nuestras librerías para poder leer datos de un archivo, escribir datos en un archivo y llevarlos a procesar para hacer los cálculos. Por ello, es importante explicar cómo se resolvió este inconveniente de lectura de un archivo, ya que, sin resolver este punto, no podríamos ir al siguiente paso para hacer un cálculo. En primer lugar, se realiza un archivo de entrada (input) con extensión ***csv***, por sus siglas en inglés 'Comma Separated Values' que significa valores separados por comas, por lo cual son N valores separados por comas, y M columna de datos, llamaremos un dato al conjunto de N valores separados por comas hasta antes de un salto de línea, un enter o un
n, como mejor se quiera entender. Teniendo en cuenta lo anterior, podemos descifrar de una manera más rápida el módulo archivo.class, en el cual el procedimiento a desarrollar es el siguiente:

Teniendo la dirección de nuestro archivo, ejecutamos los siguientes pasos:

- Se abre el archivo.

- Se asigna a la variable n la longitud total de caracteres del archivo.
- Se crea un arreglo de tipo char llamado **buffer** con la longitud de caracteres de tamaño ***n***.
- Leemos la variable ***fu*** y la asignamos a la variable **buffer** (aquí la variable **buffer** ya tienen la información del archivo **input**).
- Se cierra el flujo del archivo ya que ya tenemos la información.

Ahora se comienza un ciclo para poder saber cuántas filas y columnas tiene nuestro archivo.

Inicializamos 2 variables una llamada filas y otra columnas para poder saber de qué tamaño se debe crear la matriz la cual va a alojar los datos.

```
filas=1
columnas=1
do i=1,len(buffer)
    if(buffer(i:i)==achar (10)) then
        if(banderaCol==0)then
            do j=1,len(buffer(1:i-1))
                if(iachar(buffer(j:j))==
                iachar(this \%delimitador))then
                    columnas=columnas+1
                endif
            enddo
            banderaCol=1
        else
        filas=filas+1
        endif
    endif
enddo
```

Aunque el lector crea que no es necesario agregar un delimitador para la lectura de nuestro archivo, sí es de fundamental importancia, ya que si es un formato ***.csv*** y su delimitador es una coma, cabe mencionar que si utilizara cualquier carácter del código ASCCI como delimitador tales como @, %, ^ , * , excepto un punto (en consecuencia, se crearía conflicto en los datos en caso de ser con punto decimal), el programa podría leerlos sin ningún problema. A diferencia de que si no existiera esa variable, el usuario

debería comprender el código y tendría que saber en qué parte del código cambiar el delimitador.

Después de saber cuántas filas y columnas se tienen, se procede a crear una matriz de datos reales (numérica con punto flotante, o conocida como numero decimal), de tamaño ***m-filas, n-columnas***.

```
columnas=1
filas=1
inicioSubCadena=1
InicioNumero=1
numero=1
do i=1,len(buffer)
    if(buffer(i:i)==achar (10)) then
        do j=inicioSubCadena,i
            if(iachar(buffer(j:j))==
            iachar(this\% delimitador))then
                matriz(filas,columnas)=
                this%stringReal(buffer(InicioNumero:j-1),
                len(buffer(InicioNumero:j-1)))

                columnas=columnas+1
                numero=numero+1
                InicioNumero=j+1
            else if(buffer(j:j)==achar (10))then
                matriz(filas,columnas)=
                this%stringReal(buffer(InicioNumero:j-1),
                len(buffer(InicioNumero:j-1)))

                numero=numero+1
            endif
        enddo
        filas=filas+1
        columnas=1
        inicioSubCadena=i+1
        InicioNumero=inicioSubCadena
    endif
```

```
enddo
```

Para la estructuración de este proyecto, los desafíos fueron dos:

- Leer los datos de tal manera que no se perdiera precisión, ya que las particiones de las simulaciones entre cada dato podían o pueden ser de 0.0001, por poner un ejemplo.
- En Fortran es un poco más complicado hacer un casting (convertir algún tipo de dato en otro), como podría hacerse de manera fácil en java, o cualquier otro lenguaje, aquí se debe poner un poco más de empeño. Para esto fue creada esta simple, pero eficiente función llamada **stringReal**, que recibe una cadena de caracteres y la longitud de esa cadena.

El punto clave a considerar para esta función es la forma en que se lee la cadena entrante, así como también el formato de lectura que se le asigna:

```
real FUNCTION stringReal(this,cadena,n)
        class(archivo_ob) :: this
        integer           ::n
        character(len=n) :: cadena
        real :: x
        read (unit=cadena,fmt=*) x
        stringReal=x
END FUNCTION stringReal
```

De esta forma, se logra que la clase que considerada como soporte del proyecto, no va a fallar, ya que de esta manera logramos leer cualquier longitud de datos enteros y se cerciora que no falle, siempre y cuando tengan las precauciones que se mencionaron con los delimitadores y que sean solo números.

2.5. Ordenamiento de los datos

El siguiente algoritmo describe la solución matemática para encontrar los puntos de una pared curva, desde el ordenamiento de los datos de entrada hasta la obtener las coordenadas de los puntos de la pared.

El algoritmo se describe a continuación.

Considerando como input del programa las siguientes variables:

$\mathbf{r_{max}}$, $\mathbf{r_{min}}$, $\theta_{\mathbf{min}}$, $\theta_{\mathbf{max}}$, $\mathbf{m}$,$\mathbf{n}$ y los puntos de la simulación.

Se define:

$$\mathbf{\Delta_r} = \frac{r_{max} - r_{min}}{n}$$

y

$$\mathbf{\Delta}_\theta = \frac{\theta_{max} - \theta_{min}}{m}$$

Enlistamos los pasos a seguir para la obtención de los puntos de pared.

Paso 1.- Asignar el centro C en el sector anular que corresponde.

Como primer instancia para asignar el centro C al sector anular j, k con base en (r, θ), el algoritmo usa la siguiente funcionalidad:

Usando un contador $q = 1, ..., n$, se pregunta si el radio r del centro C cumple la siguiente condición:

$$\mathbf{r_{min}} + (\mathbf{q} - \mathbf{1})\mathbf{\Delta_r} < \mathbf{r} < \mathbf{r_{min}} + \mathbf{q}\mathbf{\Delta_r}$$

Si para cierto valor Q, se cumple lo anterior, entonces el centro esta en el sector Q, l.

Se repite el proceso con la condición para el angulo θ del centro C con $p = 1, ..., m$.

$$\theta_{\mathbf{min}} + (\mathbf{p} - \mathbf{1})\mathbf{\Delta}_\theta < \theta < \theta_{\mathbf{min}} + \mathbf{p}\mathbf{\Delta}_\theta$$

Si para cierto valor P se cumple lo anterior, el algoritmo situa al centro C en el sector anular $\mathbf{Q}, \mathbf{P}$ y le otorga el nombre de $\mathbf{C_{q,p}}$.

Sean $C_{j,k}(r, \theta)$ lo puntos coordenados inpunt's del programa, estos representan el centro de un sector anular.

Una parte importe que se usará posteriormente, es ordenar los puntos centros $(C_{j,k})$, esto se usara posteriormente para obtener los puntos exactos de la pared curva haciendo una sumaria en la que se ven implicados los $(C_{j,k})$ y sus dencidades.

Cuando decimos ordenamiento de los datos, nos referimos que cada punto (centro) de los datos de entrada pertenece a un sector anular, esto quiere decir los datos de la simulación se encuentran desordenados, hay que recordar que los datos de entrada se

encuentran en coordenadas polar, más un dato, el cual hace referencia a la densidad de dicho sector anular.

Para una mejor manipulación de los datos, se opta por crear un tipo de dato llamada Centro la cual está compuesta por una estructura punto y un valor llamado densidad, con el fin de siempre tener relacionado el centro con su densidad y no tener que crear otra matriz, además de que se puede seguir la misma dinámica de tener una única matriz por cada dato que se quiere buscar como lo son vértices, intersecciones y en este caso, los centros de cada sector anular.

Existe varias formas de atacar esta tarea, la forma más simple, tomar 4 vértices y preguntar si el punto $C_{j,k}$ esta dentro de esa área o dentro de esos 4 puntos, pero esto sería como hacer una búsqueda por fuerza bruta ya que iríamos preguntando sector por sector si es ahí a donde pertenece, esta sería una forma tardada y lo que se busca es hacer todos los procesos de la manera mas optima posible,por ellos, se aaborda el problema de la siguiente manera.

Como primer paso tomamos el radio (r) del punto $C_{j,k}$ que se quiera encontrar, después de eso se recorre solo 1 fila (la que se quiera) y se inicia el recorrido en la columna o posición 2 de dicha fila, la pregunta es: el radio del centro $C_{j,k}$ es menor al $V_{i,j}$, en cuanto la respuesta sea si por primera vez, el ciclo se detiene, y de esta forma se obtiene la coordenada j que sería la columna.

Para obtener la fila o posición i, es simular. Tomando cualquier columna de la matriz vértices se hace el siguiente recorrido preguntado:

theta del punto C_i es menor que la variable theta del punto $V_{i,j}$, en cuanto la respuesta sea si, se detiene el recorrido y se toma la variable j. Con estos dos ciclos ya se tiene en que posición se encuentra cada centro, el algoritmo seria de la manera siguiente

```
1: for a=1:m*n do
2:     for j=1:n-1 do
3:         r_vertice = vertice_{1,j}.getR()
4:         r_centro = listaCentrosa.getR()
5:         if r_centro ⩽ r_vertice then
6:             col = j
7:         end if
8:     end for
9:     for i=1:m-1 do
10:        theta_vertice = vertice_{i,j}.getTheta()
```

11: $theta_{centro} = listaCentrosa.getTheta()$
12: **if** $theta_{centro} \leqslant theta_{vertice}$ **then**
13: $fil = i$
14: **end if**
15: **end for**
16: **end for**

2.6. Vertices e intersecciones

Un sector anular esta compuesto por 4 vértices y 2 intersecciones. Por lo que para cada $C_{j,k}(r, \theta)$, calculamos dichos componentes antes mencionados:

$V_{j,k}^{l}$ donde $l = 1, 2, 3, 4$

Vértices del sector anular que tiene como centro $C_{j,k}$

$I_{j,k}^{l}$ donde $l = 1, 2$

intersecciones del sector anular que tienen como centro $C_{j,k}$

Para el calculo del vértice

$V_{j,k}^{1}$ tiene como coordenadas

$(r_{max} + (k-1)\Delta_r, (j-1)\Delta_\theta)$

$V_{j,k}^{2}$ tiene como coordenadas

$(r_{max} + (k-1)\Delta_r, j\Delta_\theta)$

$V_{j,k}^{3}$ tiene como coordenadas

$(r_{max} + k\Delta_r, (j-1)\Delta_\theta)$

$V_{j,k}^{4}$ tiene como coordenadas

$(r_{max} + k\Delta_r, j\Delta_\theta)$

Para calcular las intersecciones $I_{j,k}^{l}$

$I_{j,k}^{1}$ tiene como coordenadas:

$(r_{min} + (k-1)\Delta_r, \frac{\Delta_\theta}{2} + (j-1)\Delta_\theta)$

$I_{j,k}^{2}$ tiene como coordenadas:

$(r_{min} + k\Delta_r, \frac{\Delta_\theta}{2} + (j-1)\Delta_\theta)$ Para cada $C_{j,k}$ viene una cantidad llamada densidad.
$\boldsymbol{n}$ se obtiene tomando toda la primera fila de datos (hasta antes de un salto de línea \n, el cual en código ASCII corresponde al valor 10), el siguiente paso es, contar cuantos datos están el esa fila y se asigna haciendo un conteo de datos de esta forma obtener cuantos valores(n) tiene dicha fila, y dando por entendido que todas las filas de datos tienen el mismo número de datos, solo queda contar los saltos de línea que tenemos y asignarlos a las columnas para así tener las m columnas.

Después de calcular m y n, se debe ser lo mas preciso posible ya que con ello se hacen los primeros cálculos para poder cuadrar los vértices de cada sector anular.
Se calcula los siguientes datos:

$Delta_{theta}$: este dato es el incremento en theta, el cual se calcula de la siguiente manera.

$Delta_theta = (theta_{max} - theta)/m$ $Delta_r$: incremento en el radio (en el eje x) el cual se obtiene de la siguiente manera $Delta_r = (r_{max} - r)/n$

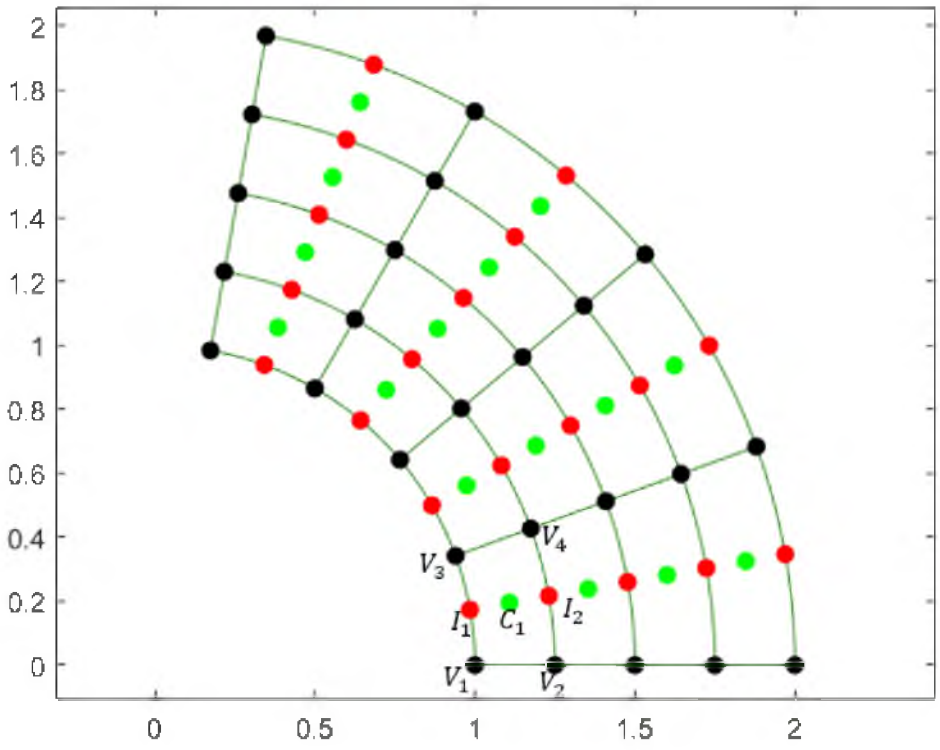


Figura 2.1: componentes de un sector anular

Para calcular un punto vértice V_i de cualquier sector anular, en este caso el primer sector anular, Teniendo en cuenta que se está manejando todo por coordenadas polares, se puede calcular el vértice 1 de la siguiente forma:

$Delta_{theta} = (theta_{max} - theta)/m$

$Delta_{theta} = (80 - 0)/4$

$Delta_{theta} = 20$

$Delta_r = (r_{max} - r_{min})/n$

$Delta_r = (2 - 1)/4$

$Delta_r = 0.25$

Habiendo obtenido $Delta_{theta}$ y $Delta_r$ y si se quiere calcular el punto V_1 se puede de la siguiente manera:

$r = (r_{min} + (i * Delta_r)) * \cos(theta + (j * delta_{theta}))$

$r = (1 + (0 * 0.25)) * \cos(0 + (0 * 20))$

$r = 1$

$tetha = (r_min + (i * Delta_r)) * \sin(theta + (i * delta_{theta}))$ $tetha = (1 + (0 * 0.25)) * \sin(0 + (0 * 20))$ $tetha = 0$

siendo i la fila y j la columna en la que se encuentra el V_1

otro ejemplo puede ser calcular el punto V_4

$r = (1 + (1 * 0.25)) * \cos(0 + (1 * 20))$

$r = 1.17462$

$tetha = (1 + (1 * 0.25)) * \sin(0 + (1 * 20))$

$tetha = 0.427525$

Viendo el ejemplo anterior se puede deducir la una formula para poder calcular cualquier vértice que se encuentre dentro del rango de la simulación con la siguiente formula:

$r = (r_{min} + (i * Delta_r)) * \cos(theta + (j * delta_{theta})$

$tetha = (r_{min} + (i * Delta_r)) * \sin(theta + (i * delta_{theta}))$

Pero si se es necesario calcular todos los vértices, esto se puede hacer de una forma más fácil, se resuelve haciendo un recorrido de dos ciclos anidados, los cuales serán

controlados por las variables i y j, que son el número total de filas y columnas respectivamente. El algoritmo sería el siguiente:

1: **for** i=0:m **do**
2: **for** j=0:n **do**
3: $r = (r_{min} + (i * Delta_r)) * \cos(theta + (j * delta_{theta}))$
4: $tetha = (r_{min} + (i * Delta_r)) * \sin(theta + (i * delta_{theta}))$
5: $v_{i+1,j+1} = (r, thteta)$
6: **end for**
7: **end for**

De esta forma pueden ser calculado todos los vértices que se encuentran dentro del rango de la simulación.

Para poder calcular las intersecciones se sigue un procedimiento similar al anterior, la diferencia está en que se tiene una variable llamada $theta_{inicial}$ será la suma de $theta + delta_{theta}$, esto se hace porque las intersecciones siempre van a tener el mismo ángulo de abertura que los centros del cada sector anular, esto quiere decir que tiene la misma pendiente que los puntos centros.

Para calcular las intersecciones de la simulación se calculan con la siguiente formula:

$r = (r_{min} + (j * Delta_r)) * \cos(theta_{inicial} + (i * delta_{theta}))$

$tetha = (r_{min} + (j * Delta_r)) * \sin(theta_{inicial} + (i * delta_{theta}))$

$I_{i+1,j+1} = (r, thteta)$

Para calcular todas las intersecciones se aplica el siguiente algoritmo:

1: **for** i=0:m-1 **do**
2: **for** j=0:n **do**
3: $r = (r_{min} + (j * Delta_r)) * \cos(theta_{inicial} + (i * delta_{theta}))$
4: $tetha = (r_{min} + (j * Delta_r)) * \sin(theta_{inicial} + (i * delta_{theta}))$
5: $I_{i+1,j+1} = (r, thteta)$
6: **end for**
7: **end for**

2.7. Creación de la malla

Una vez ubicados los puntos que componen cada sector anular(vértices e intersecciones), se procede a encontrar los puntos de la pared curva.

Un punto de la pared curva puede estar ubicado en una fila de sectores anulares. Dichos sectores anulares j,k poseen un valor fijo de $p_{j,k}$. Dejando el valor k fijo,se va sumando sobre j hasta que la suma de densidades es menor a $\tau = 2/3$

$$p_j = \sum_{a=1}^{k} p_{j,a} \leqslant \tau$$

Para el valor más grande de k en el que se cumple lo anterior, se toma el sector anular j,k y se comienza con el segundo algoritmo.

En el sector anular $j,k+1$, hacemos una partición de intervalo Δ_r desde $r_{min}+(k+1)\Delta_r$ hasta $r_{min}+k\Delta_r$ a la mitad.

Cada una de estas partes posee una densidad molecular $p^1_{j,k+1}, p^2_{j,k+1}, ...$

Se calcula

$$p = p_j + p^i_{j,k}$$

y se determina

si $p = \tau$,se guarda el la coordenada r del punto $r = r_{min} + (k+1)\Delta_r + \dfrac{\Delta_r}{2}$

En caso contrario,se pregunta si $p < \tau$, entonces se repite el proceso anterior ahora partiendo el intervalo $[r_{min} + (k-1)\Delta_r + \dfrac{\Delta_r}{2}, r_{min} + k\Delta_r]$, de lo contrario se toma el siguiente intervalo:

$$[r_{min} + (k-1)\Delta_r, r_{min} + (k-1)\Delta_r + \frac{\Delta_r}{2}]$$

se hacen n iteraciones de este algoritmo sobre los 2^{n-1} intervalos que se crean hasta alcanzar $p = \tau$

$$\rho\kappa\Delta_r$$

Una vez teniendo todos las variables (vértices e intersecciones) y teniendo un orden con los puntos centros, se puede proceder a calcular los puntos que satisfagan la profundidad óptica $\tau = 2/3$, esto se consigue calculando la radiación radial proveniente de la estrella central con los puntos centros de una misma fila hasta.

Para la primer parte se usan las intersecciones de cada sector anular, esto con el fin de calcular el tau que existe a la largo de dicho sector, la formula es la siguiente:

esto se repetirá para todos los siguiente sectores anulares, hasta que tau sea mayor a 2/3,ya que se tomara el sector anular anterior y se hará la segunda parte del algoritmo.

Una vez que se tiene ubicado el sector anular en el cual se esta complemente seguro que se van a encontrar el valor,se aplica el método de búsqueda binaria.

A continuación, se explica la implementación de método de búsqueda binaria. Para aplicar dicho algoritmo solo se deben cumplir previamente 2 condiciones:

- El conjunto o vector de elementos, en el cual se va a buscar el valor, debe estar ordenado
- Conocer el número de elementos que conforman el conjunto.

Si se quisiera saber en que iteración se encontró el valor buscado y en la primer iteracion que es cuando se tiene el arreglo completo no existe una división del mismo, en la segunda iteración el arreglo es dividido a la mitad, en tercer iteración ya se encuentra dividido en 4 partes y así sucesivamente.

La notación asintótica $O(n)$ indica que el tiempo de ejecución es proporcional al tamaño de entrada n esta representa al alogitmo de busqueda secuencial,por otra parte la notación asintótica $O(\log n)$ indica que el tiempo de ejecución crece de forma logarítmica con respecto al tamaño de entrada.

Una representacion grafica puede ser la que se muestra en la Figura 2.2

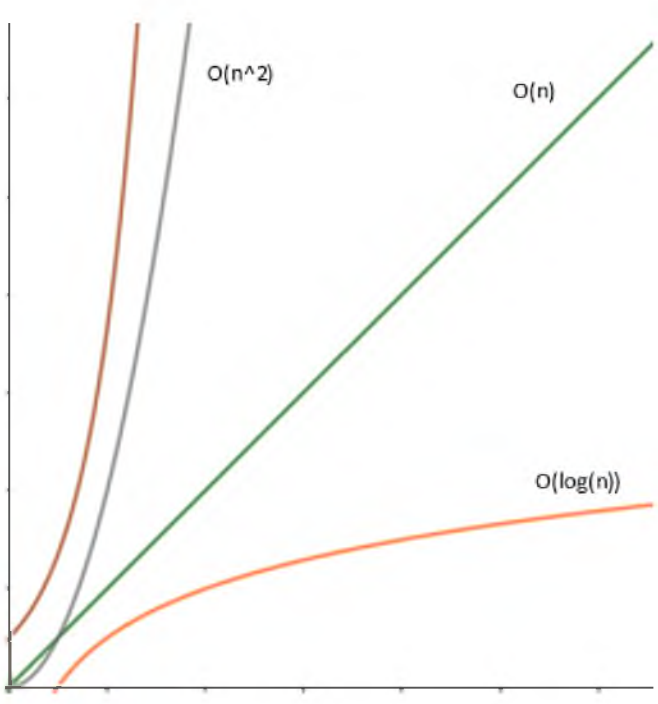


Figura 2.2: Grafica de los algoritmos

Donde el algiritmo secuencial seria de $O(n)$ y el de busqueda binaria seria $O(\log(n))$

```
real(8) FUNCTION binary_search(kappa, rho, p_in ,
p_fin, tauSum)
TYPE(punto_ob)       :: p_in , p_fin, p_malla
real(8) :: kappa, rho, dist, tauSum
real(8) :: get_optical_depth ,tauTem
real(8) :: distance
real(8) :: min, max, index, tau
real(8) :: const, errorMin, theta, r, x, y
integer :: bandera, num
bandera = 1
const =1
dist =distance(p_in , p_fin)
min = 0
max = dist
tau = (2.0/3.0)*const
theta = atan(p_in%getY()/p_in%getX())
do while (bandera == 1)
        index= (max - min) / 2
        tauTem = get_optical_depth(kappa, rho,
        index, tauSum)
        if(tauTem == tau ) then
                x = index*cos(theta)
                y = index*sin(theta)
                p_malla = punto(x,y)
                call p_malla%imprimirPunto()
                bandera=0
        else if(tauTem > tau) then
                max = index
        else
                min = index
        end if
end do
binary_search = index
end FUNCTION binary_search

real(8) FUNCTION get_optical_depth(kappa,
rho, dist, tauSum)
```

```
real(8) :: kappa, rho, tauSum
real(8) :: tau, dist
tau = kappa*rho*abs(dist)
get_optical_depth = tau + tauSum
end FUNCTION get_optical_depth
```

En el código de la función binary_search, el apartado esencial para lograr la acción propuesta es la que se encuentra dentro de la estructura repetitiva **do while**.

El código anterior presenta un algoritmo de búsqueda binaria, el cual fue modificado para cubrir las necesidades del proyecto sin perder la esencia de dicho algoritmo con su orden de complejidad temporal. A continuación se describe las partes importantes del algoritmo:

La primer parte que tener un numero finito de puntos, su puede ver en que, en este caso, el arreglo de longitud **n** será la distancia que existe entre los dos que son las intersecciones del sector anular, ya que en ese rango se estará buscando satisfacer el $\tau = 2/3$.

2.8. Diagrama de flujo

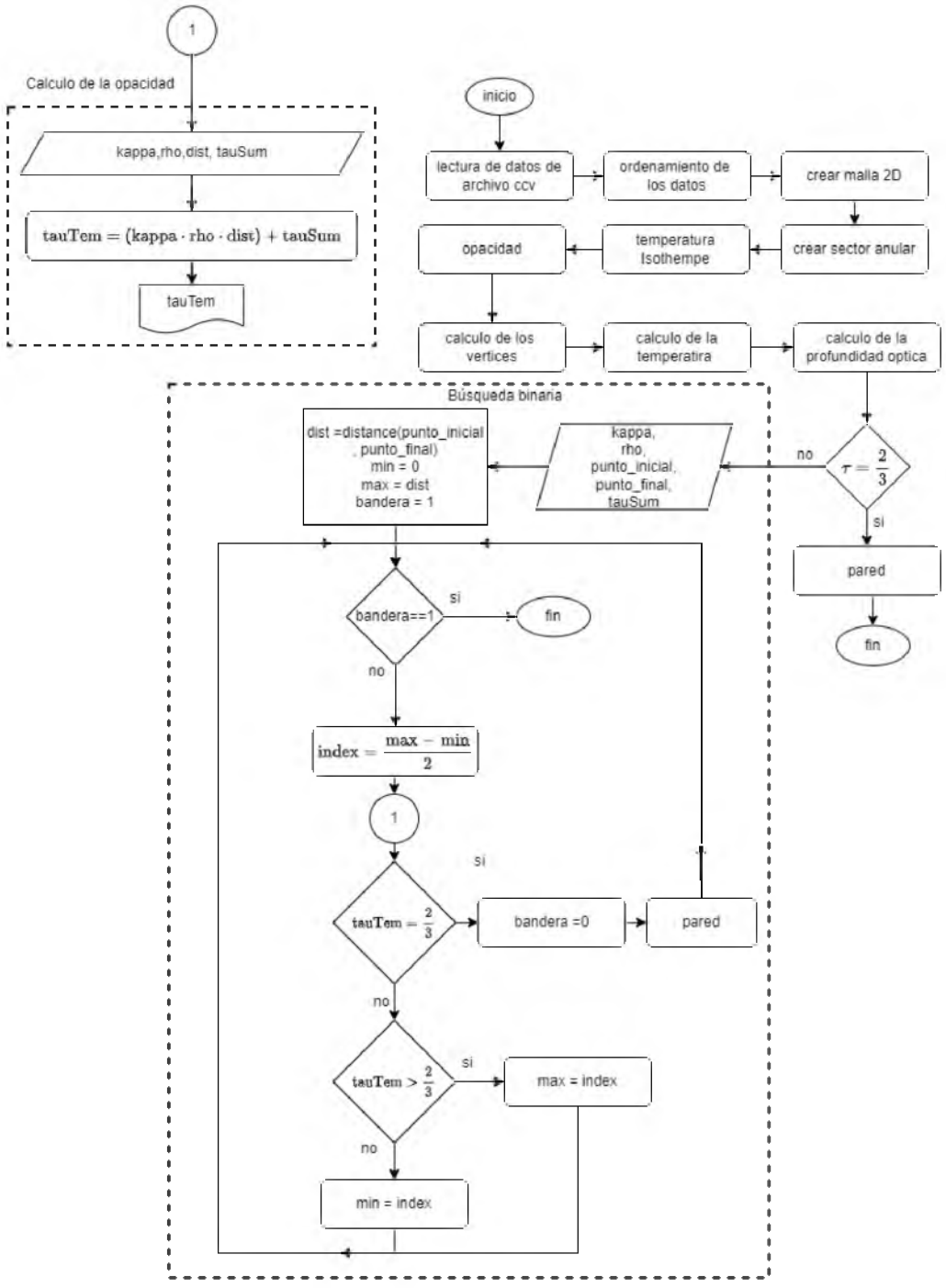


Figura 2.3: diagrama de flujo del codigo ARTeMiSE2.0

Capítulo 3

Resultados y conclusiones

3.1. Implementaciones Astronomicas

3.1.1. El sistema estelar LkC15

LkCa 15 es una estrella de tipo T Tauri con un espectro K5, situada en la región de formación estelar Tauro-Auriga, a una distancia aproximada de 145 $\pm$ 15 parsecs (pc) de la Tierra. Esta estrella exhibe un sistema de discos que incluye un disco interno, una cavidad y un disco externo. Los modelos de su SED (Distribución Espectral de Energía) indican que la pared interna del disco externo, asumida como vertical, se encuentra a una distancia de aproximadamente 58 Unidades Astronómicas (AU).

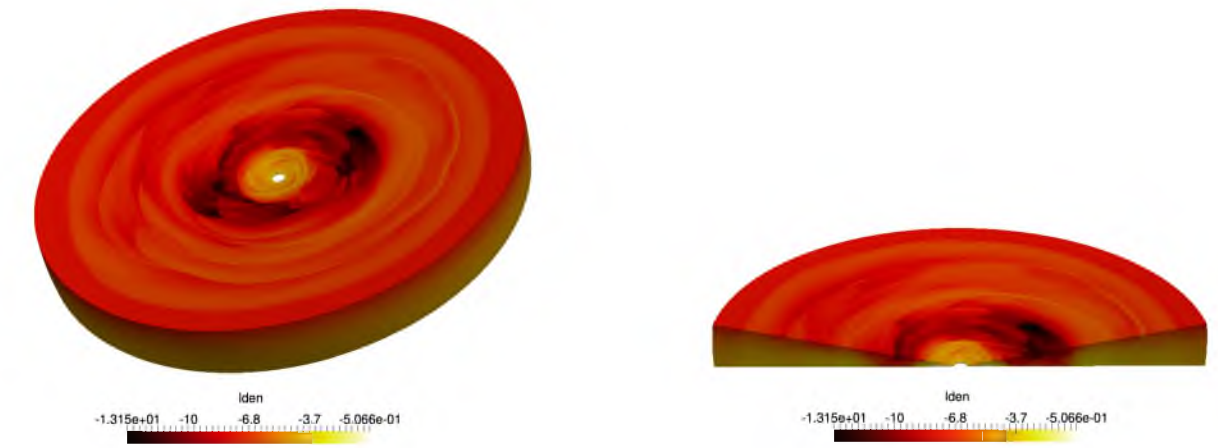
Inicialmente, se propuso que la cavidad en el disco fue causada por un planeta con una masa cercana a 6 veces la de Júpiter (ME) (29). Sin embargo, se determinó que un planeta de este tamaño no sería suficiente para crear la cavidad en el disco. En consecuencia, se planteó la hipótesis de la existencia de un planeta de menor tamaño ubicado cerca del borde del disco externo para explicar la cavidad.

Observaciones realizadas por el telescopio Gemini NIRI sugieren que un planeta más masivo podría haber abierto la cavidad (1). Por lo tanto, en investigaciones actuales, se ha propuesto un planeta con una masa de aproximadamente 10 veces la de Júpiter como candidato para la formación de esta cavidad.

3.1.2. Simulación

Para identificar rápidamente la órbita en la que el sistema alcanza un estado cuasi-estacionario Se lleva a cabo una simulación de mediana resolución (250 x 250 x 100) con el codigo FARGO 3D, como se observa en la Figura 3.1. Los parámetros principales de la simulación están detallados en la Tabla 3.1. Estos parámetros son fundamentales para configurar la simulación y lograr resultados precisos. Además, la elección de una resolución mediana es crucial para equilibrar la precisión de la simulación con la eficiencia computacional. Esta aproximación permite obtener una visión detallada de la estructura de la pared en el contexto del sistema en estudio y facilita la identificación de estados cuasi-estacionarios.

En las Figuras 3.2 y 3.3 se puede observar como el planeta, que se localiza en el punto (1,0), ha logrado abrir completamente una cavidad en el disco tanto en el plano medio como en el plano vertical. Los isocontornos de densidad muestran las regiones en el disco que carece de polvo; sin embargo, en dichas regiones aún existe la presencia de de gas.



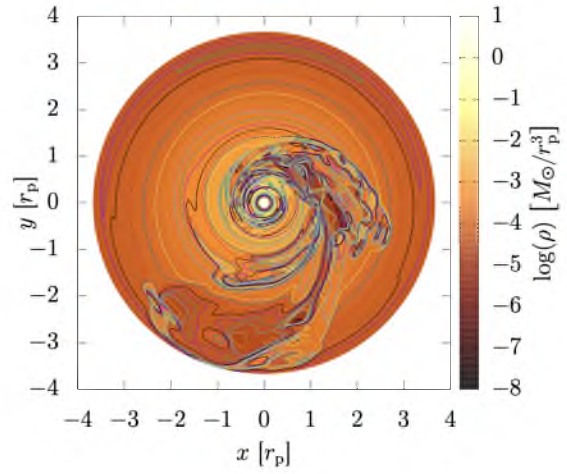
(a) Estructura tridimensional del disco.

(b) Corte vertical del disco en la posición del planeta.

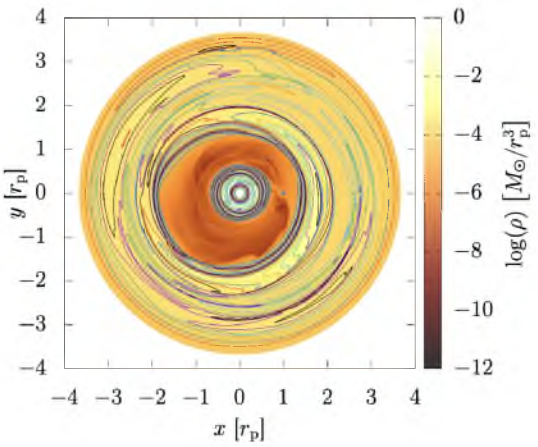
Figura 3.1: Simulación 3D del disco LkCa 15 realizada con el código FARGO 3D

	Parametro	valor
Disco	Relación de aspecto (H)	0.045
	Densidad superficial (Σ_0)	$1.44666x10^{-4}$
	α-viscosidad	0.0
	Pendiente Σ_0	1.0
	Índice de ensanchamiento	0.0
Planeta	Masa (m_p)	$10M_\mu$
	Posicion (a_p)	23 AU
	RocheSmoothing	0.4
	Acreción	No
Malla	Unidades	No
	Dimensión	3D
	Geometría	Esférica
	$[X_{mín}$, $X_{máx}]$	$[-\pi,\pi]$
	$[Y_{mín}$, $Y_{máx}]$	[0.1 , 3.666]
	$[Z_{mín}$, $Z_{máx}]$	[1.37340073 , $\pi/2$]
Tiempo	Órbitas	150

Tabla 3.1: Parametros de entrada de la simulación

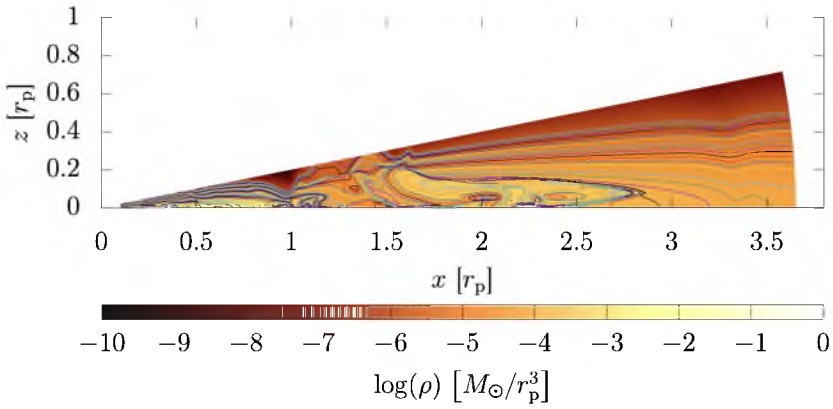


(a) Órbita 1: El planeta que está en la posición (1,0) está acretando material.

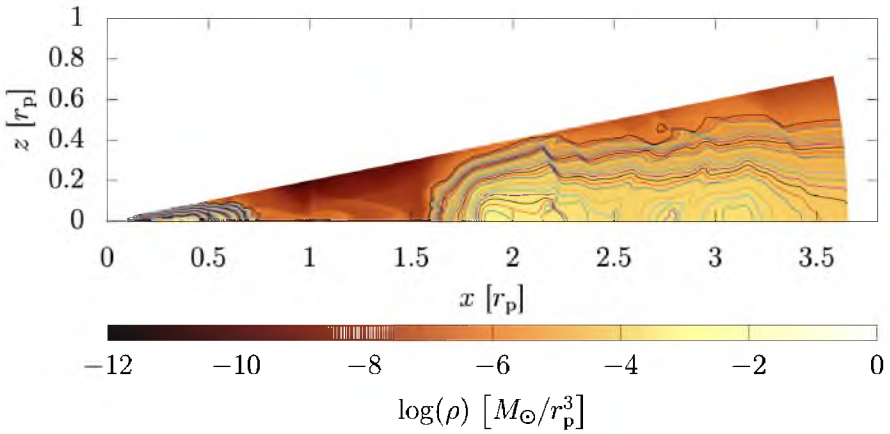


(b) Órbita 100: El planeta que está en la posición (1,0) ha logrado abrir una cavidad en el disco.

Figura 3.2: Apertura de la cavidad observada en el plano medio del disco (isocontornos de densidad).



(a) Órbita 1: El planeta que está en la posición (1,0) se observa como apenas se está abriendo la cavidad.



(b) Órbita 100: El planeta que está en la posición (1,0) ha logrado abrir una cavidad por completo en el disco

Figura 3.3: Apertura de la cavidad observada en el plano vertical del disco (isocontornos de densidad).

3.1.3. Geometria de la Pared

Con la implementación del codigo ARTeMiSE2.0 al sistema estelar LkCa15 se estima que la pared curva inicia a una distancia de 53 Unidades Astronómicas (AU) desde la estrella central y se extiende hasta 68.7 AU a lo largo del plano medio, alcanzando una altura de 12.22 AU medida desde el plano medio hacia arriba, como se ilustra en la Figura 3.4. En este modelo, el polvo que compone la pared está compuesto por una combinación de granos pequeños y grandes de olivino cristalino (silicato) con una composición de 50 % de hierro (Fe) y 50 % de magnesio (Mg). Además, se encuentra presente una cantidad reducida de granos orgánicos y troilita en esta mezcla.

Esta información proporciona detalles cruciales sobre la ubicación y la composición de

la pared curva en el contexto del sistema estudiado, lo que es relevante para comprender su estructura y propiedades en un entorno astronómico.

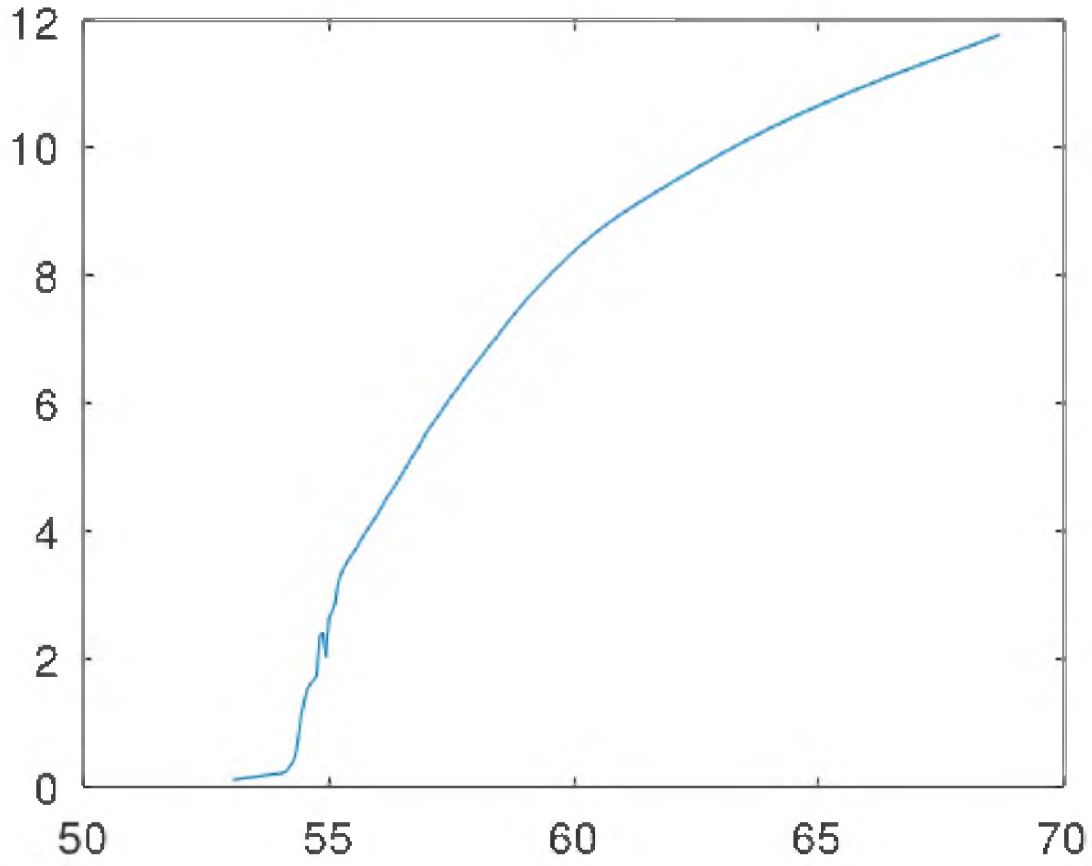


Figura 3.4: Geometría de la pared

3.2. Implicaciones computacionales

Cabe recalcar que en este problema específico que aborda este trabajo de investigación, la implementación del método de búsqueda binaria se ajusta muy bien como estrategia de solución al problema presentado, ya que, como se explicó en secciones anteriores, uno de los requisitos es que el arreglo esté ordenado. En esta tesis se parte de que, al situarse en el sector anular en donde se va a encontrar el valor de $\tau = 2/3$.

Para esta parte vamos a hacer una comparación con el objetivo ver el comportamiento en un caso real de manera práctica para saber cuál es mejor, cuáles son los pro y los contra del algoritmo de búsqueda binaria, y si el algoritmo escogido es mejor o el más optimo. En la simulación utilizada se lograron encontrar 91 puntos que cumplen con la

condición que hemos puesto para encontrar los puntos de la pared, la cual es cuando se halla una profundidad óptica $\tau = 2/3$.

Una de las primeras comparaciones que se puede hacer es observar los tiempos en los cuales se ejecutaron los algoritmos. Lo primero que se puede notar es que se cumple la teoría en la cual dice el algoritmo de búsqueda binaria al ser de orden $O(\log(n))$ será más rápido que uno de orden $O(n)$ que es el de búsqueda secuencial; esta comparación se hace con una precisión del algoritmo de búsqueda secuencial en la que el sector anular se divide en 10,000 partes iguales.

Tiempo de ejecución por dato		
Dato	Búsqueda Binaria	Búsqueda Secuencial
1	7.62939E-06	1.14441E-05
2	7.62939E-06	5.72205E-06
3	7.62939E-06	1.52588E-05
4	7.62939E-06	2.47955E-05
5	3.8147E-06	1.90735E-06
6	3.8147E-06	1.14441E-05
7	7.62939E-06	1.90735E-05
8	1.90735E-05	1.14441E-05
9	7.62939E-06	7.62939E-06
10	7.62939E-06	1.14441E-05
11	7.62939E-06	2.28882E-05
12	7.62939E-06	1.52588E-05
13	3.8147E-06	1.14441E-05
14	3.8147E-06	1.52588E-05
15	7.62939E-06	1.14441E-05
16	3.8147E-06	1.52588E-05
17	7.62939E-06	1.52588E-05
18	7.62939E-06	2.28882E-05
19	3.8147E-06	3.8147E-06
20	7.62939E-06	1.52588E-05
21	3.8147E-06	2.28882E-05
22	7.62939E-06	1.52588E-05
23	7.62939E-06	3.8147E-06
24	3.8147E-06	1.14441E-05
25	7.62939E-06	3.8147E-06

Dato	Búsqueda Binaria	Búsqueda Secuencial
26	7.62939E-06	1.52588E-05
27	7.62939E-06	7.62939E-06
28	7.62939E-06	2.28882E-05
29	2.67029E-05	1.52588E-05
30	7.62939E-06	7.62939E-06
31	7.62939E-06	2.28882E-05
32	7.62939E-06	1.52588E-05
33	7.62939E-06	1.14441E-05
34	2.28882E-05	2.67029E-05
35	3.8147E-06	1.52588E-05
36	7.62939E-06	7.62939E-06
37	7.62939E-06	3.8147E-06
38	7.62939E-06	2.28882E-05
39	7.62939E-06	1.14441E-05
40	7.62939E-06	7.62939E-06
41	7.62939E-06	3.8147E-05
42	7.62939E-06	1.90735E-05
43	3.8147E-06	1.14441E-05
44	7.62939E-06	7.62939E-06
45	7.62939E-06	2.67029E-05
46	7.62939E-06	1.52588E-05
47	7.62939E-06	1.14441E-05
48	7.62939E-06	7.62939E-06
49	3.8147E-06	3.8147E-06
50	7.62939E-06	1.90735E-05
51	7.62939E-06	1.14441E-05
52	1.52588E-05	7.62939E-06
53	7.62939E-06	3.8147E-06
54	7.62939E-06	2.28882E-05
55	7.62939E-06	1.90735E-05
56	7.62939E-06	1.14441E-05
57	7.62939E-06	7.62939E-06
58	7.62939E-06	0
59	7.62939E-06	2.28882E-05

Dato	Búsqueda Binaria	Búsqueda Secuencial
60	7.62939E-06	1.14441E-05
61	7.62939E-06	7.62939E-06
62	7.62939E-06	3.8147E-06
63	3.8147E-06	3.8147E-06
64	7.62939E-06	3.8147E-06
65	1.14441E-05	0
66	7.62939E-06	2.28882E-05
67	7.62939E-06	1.90735E-05
68	3.8147E-06	1.52588E-05
69	3.8147E-06	1.14441E-05
70	3.8147E-06	7.62939E-06
71	7.62939E-06	7.62939E-06
72	7.62939E-06	1.14441E-05
73	7.62939E-06	1.52588E-05
74	1.90735E-05	7.62939E-06
75	7.62939E-06	7.62939E-06
76	7.62939E-06	0
77	1.14441E-05	1.90735E-05
78	7.62939E-06	1.52588E-05
79	2.28882E-05	1.14441E-05
80	7.62939E-06	1.14441E-05
81	3.8147E-06	7.62939E-06
82	7.62939E-06	7.62939E-06
83	7.62939E-06	7.62939E-06
84	3.8147E-06	7.62939E-06
85	7.62939E-06	3.8147E-06
86	7.62939E-06	3.8147E-06
87	7.62939E-06	7.62939E-06
88	3.8147E-06	3.8147E-06
89	7.62939E-06	1.90735E-05
90	7.62939E-06	1.52588E-05
91	7.62939E-06	7.62939E-06

Tabla 3.2: Tabla de tiempos de ejecución

Ahora se muestra una breve estadística de como fue el desempeño de dichos algoritmos.

	Tiempos de ejecución		
Mtd. de búsqueda	Mejor	Peor	Promedio
Busq. Binaria	$3.81E-06$	$2.67E-05$	$7.84E-06$
Busq. Secuencial	$0.00E+00$	$2.67E-05$	$1.22E-05$

Tabla 3.3: Análisis de tiempos

Tal vez a simple vista no se nota, pero graficando los datos podemos notar que nuestros datos están en cierto rango, más adelante podremos reafirmarlo en la visualización de las iteraciones.

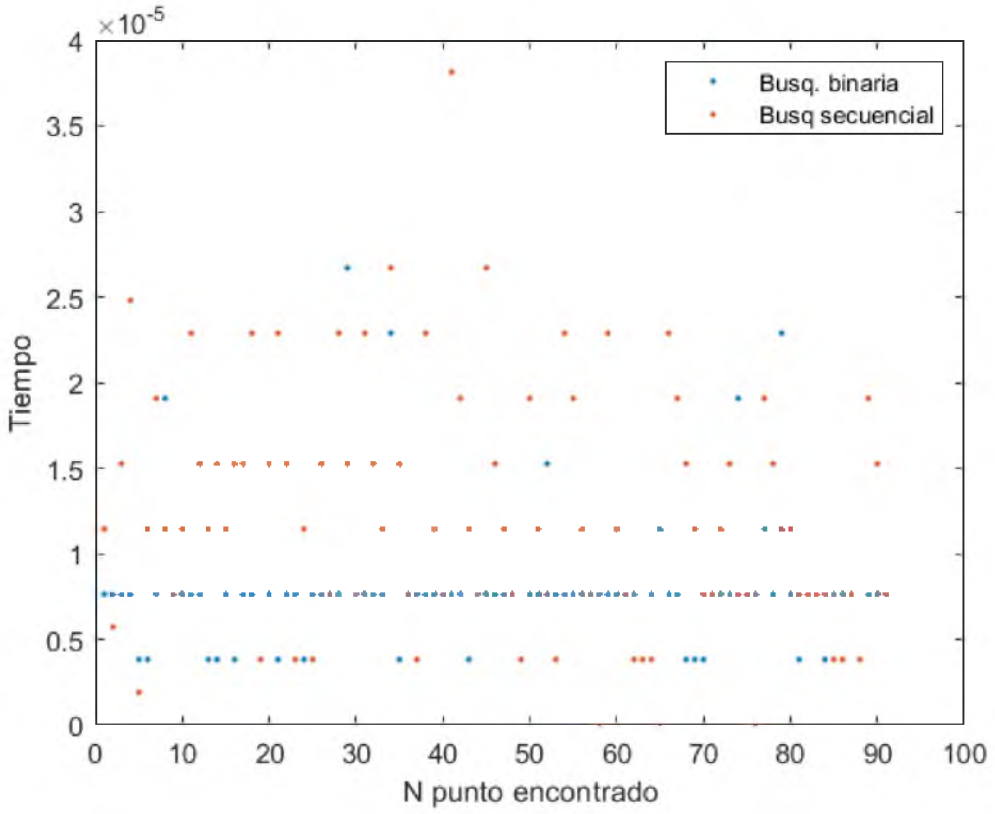


Figura 3.5: Comparación del tiempo de ejecuciones de los algoritmos.

Por último, se revisa el comportamiento de los algoritmos pero ahora tomando como base el número de iteraciones que tuvieron que dar para encontrar cada uno de los valores.

Iteraciones por dato		
Dato	Búsqueda Binaria	Búsqueda Secuencial
1	45	3432
2	42	1507
3	47	4709
4	45	8661
5	48	133
6	45	4050
7	48	7915
8	48	2704
9	49	1598
10	49	4181
11	49	9489
12	49	6774
13	49	5527
14	49	5226
15	48	5423
16	50	6017
17	50	7154
18	50	9047
19	50	1846
20	50	5494
21	50	9871
22	47	4677
23	48	28
24	50	5361
25	48	1145
26	48	6904
27	51	2975
28	46	9443
29	51	5826
30	51	2571
31	48	9454
32	51	6007
33	46	2798

Dato	Búsqueda Binaria	Búsqueda Secuencial
34	50	9801
35	50	6415
36	47	3340
37	50	577
38	50	7553
39	50	4680
40	49	2019
41	47	9410
42	46	6481
43	50	3939
44	49	1731
45	50	9657
46	50	7189
47	49	4866
48	49	2640
49	50	540
50	46	8029
51	50	5537
52	51	3331
53	47	1284
54	50	9146
55	50	6673
56	50	4261
57	51	2062
58	50	87
59	50	7712
60	50	5413
61	50	3426
62	45	1955
63	46	1054
64	49	599
65	50	153
66	44	9253
67	49	7795

Dato	Búsqueda Binaria	Búsqueda Secuencial
68	50	5866
69	50	3753
70	48	2145
71	51	2284
72	49	4130
73	50	4862
74	51	4019
75	51	2530
76	51	485
77	50	7759
78	51	5803
79	51	4987
80	50	4370
81	48	3677
82	50	2801
83	48	2024
84	51	1838
85	51	1713
86	50	1253
87	49	914
88	50	30
89	51	8166
90	50	5616
91	48	2914

Tabla 3.4: Tabla de iteraciones por dato.

Como se puede ver, en este caso, con el numero de iteraciones podemos visualizar mas la diferencia entre algoritmos.

	Número de iteraciones		
Mtd. de búsqueda	Mejor	Peor	Promedio
Busq. Binaria	42	51	49
Busq. Secuencial	28	9,871	4,511

Tabla 3.5: Análisis de iteraciones por dato.

Complementando con la estadística anterior, podemos notar que además de encontrar los datos en pocas iteraciones, el algoritmo de búsqueda binaria es más constante, ya que normalmente el algoritmo logra encontrar el valor en la iteración 42 y la 51, mientras que el algoritmo de búsqueda secuencial es más constante al encontrar el valor en un intervalo de iteraciones.

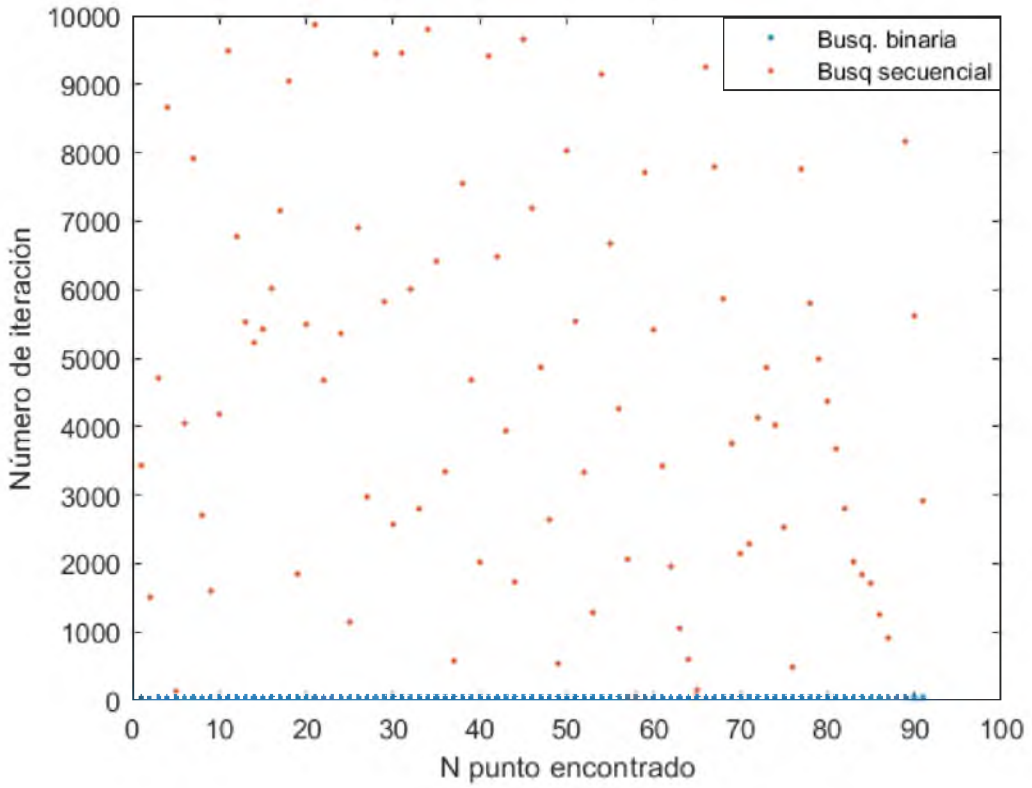


Figura 3.6: Comparación del número de ejecuciones de los algoritmos.

3.3. Conclusiones

Se ha desarrollado el código computacional ARTeMiSE 2.0 que ha sido creado con el propósito de investigar la estructura vertical (pared) de las cavidades en discos protoplanetarios en estados transicionales y pretransicionales, donde se asume que dichas cavidades son el resultado de planetas en formación.

Con la implementación del código ARTeMiSE 2.0 al sistema estelar LkCa 15, se ha logrado demostrar que la geometría de la parede de la cavidad es curvas y no vertical como se asume en modelos tradicionales de discos.

El código ARTeMiSE 2.0 está basado en el algoritmo de búsqueda binaria para hallar

la localización de los puntos que definen la pared de la cavidad en aquellos lugares del disco donde la profundidad óptica es $\tau = 2/3$.

La ventaja computacional del algoritmo de búsqueda binaria en comparación con el algoritmo secuencial radica en la eficiencia que posee en la búsqueda de elementos en conjuntos de datos ordenados, la búsqueda binaria divide repetidamente el conjunto a la mitad, reduciendo significativamente el número de comparaciones necesarias.

Las ventajas importantes del algoritmo de búsqueda binaria en el código ARTeMiSE 2.0 fueron:

Eficiencia: La búsqueda binaria es notablemente más rápida ya que los datos de la simulación del disco están ordenados, lo que se ve reflejado en una reducción en el número de comparaciones a una fracción logarítmica del tamaño del conjunto total de datos, lo que lo hace mucho más eficiente que el algoritmo de búsqueda secuencial.

Reducción del tiempo de búsqueda: La búsqueda binaria permite encontrar elementos en tiempo $O(\log n)$, mientras que el algoritmo secuencial opera en tiempo $O(n)$, donde n es el número de elementos en el conjunto. Esta particularidad resulta de suma importancia, ya que al estudiar la geometría de las cavidades en los discos, de un solo sistema estelar, se requiere analizar una multitud de simulaciones, ya que en ocasiones la simulación del disco posee cierta temperatura que no coincide con la temperatura de la pared, la cual es calculada por el código RHADaMAnTe cuyo desarrollo no es competencia de la presente tesis.

Bibliografía

[1] C. Thalmann y col. The architecture of the LkCa 15 transitional disk revealed by high-contrast imaging. En: A&A 566, A51 (jun. de 2014), A51. doi:10.1051/0004-6361/201322915. arXiv: 1402.1766 [astro-ph.SR]

[2] López-García, M. A. (2018). Caracterización de la población estelar joven en la Nebulosa Molecular de Orión B. Tesis doctoral. Madrid, España.

[3] Hayashi, C. (1966). Evolution of Protostars. ARA&A, 4, 171.

[4] Poveda-Tejada, N., Rodríguez Cuevas, L., Vera-Villamizar N. (2018). Statistical model of the distribution of matter of protoplanetary disks. Ciencia en Desarrollo, 10, 1.

[5] Smith, B. A., & Terrile, R. J. (1984). A circumstellar disk around Beta Pictoris. Science, 226, 1421–1424.

[6] Cánovas, H. (2018). Discos protoplanetarios: fábricas de planetas en acción. Astronomía, 22, 226.

[7] Calvet, N., D'Alessio, P., Hartmann, L., Wilner, D., Walsh, A., & Sitko, M. 2002, ApJ, 568, 1008

[8] Strom et al. 1989, Circumstellar material associated with solar-type pre-main-sequence stars, a possible constraint on the timescale for planet building, The Astronomical Journal, 97, 1451.

[9] Hughes, A. M., Andrews, S. M., Espaillat, C., Wilner, D. J., Calvet, N., D'Alessio, P., Qi, C., Williams, J. P., & Hogerheijde, M. R. (2009). A Spatially Resolved Inner Hole in the Disk Around GM Aurigae. ApJ, 698, 131–142. Hughes, A. M., Wilner.

[10] Espaillat, C., Calvet, N., D'Alessio, P., Bergin, E., Hartmann, L., Watson, D., Furlan, E., Najita, J., Forrest, W., McClure, M., Sargent, B., Bohac, C., & Harrold,

S. T. (2007a). Probing the Dust and Gas in the Transitional Disk of CS Cha with Spitzer. ApJ, 664, L111–L114.

[11] Espaillat, C., Calvet, N., D'Alessio, P., Hernandez, J., Qi, C., Hartmann, L., Furlan, E., & Watson, D. M. (2007b). On the Diversity of the Taurus Transitional Disks: UX Tauri A and LkCa 15. ApJ, 670, L135–L138.

[12] Eisner, J. A., Monnier, J. D., Tuthill, P., & Lacour, S. (2009). Spatially Resolved Mid-Infrared Imaging of the SR 21 Transition Disk. ApJL, 698, L169–L173.

[13] Uchida, K. I., Calvet, N., Hartmann, L., Kemper, F., Forrest, W. J., Watson, D. M., D'Alessio, P., Chen, C. H., Furlan, E., Sargent, B., Brandl, B. R., Herter, T. L., Morris, P., Myers, P. C., Najita, J., Sloan, G. C., Barry, D. J., Green, J., Keller, L. D., & Hall, P. (2004). The State of Protoplanetary Material 10 Million years after Stellar Formation: Circumstellar Disks in the TW Hydrae Association. ApJS, 154, 439–442.

[14] Dullemond, C. P., Dominik, C., & Natta, A. (2001). Passive Irradiated Circumstellar Disks with an Inner Hole. ApJ, 560, 957–969.

[15] Merin, B., Montesinos, B., Eiroa, C., Solano, E., Mora, A., D'Alessio, P., Calvet, N., Oudmaijer, R. D., de Winter, D., Davies, J. K. Harris, A. W., Collier Cameron, A., Deeg, H. J., Ferlet, R., Garzon, F., Grady, C. A., Horne, K., ´ Miranda, L. F., Palacios, J., Penny, A., Quirrenbach, A., Rauer, H., Schneider, J., & Wesselius, P. R. (2004). Study of the properties and spectral energy distributions of the Herbig AeBe stars HD 34282 and HD 141569. A&A, 419, 301–318.

[16] Lada, C. J., & Wilking, B. A. (1984). The nature of the embedded population in the Rho Ophiuchi dark cloud - Mid-infrared observations. ApJ, 287, 610–621.

[17] D'Alessio, P., Calvet, N., Hartmann, L., Franco-Hernández, R., y Servín, H. (2005). Effects of dust growth and settling in T Tauri disks. The Astrophysical Journal, 638(1), 314.

[18] Calvet, N., D'alessio, P., Watson, D. M., Franco-Hernández, R., Furlan, E., Green, J., y Hall, P. (2005). Disks in transition in the Taurus population: Spitzer IRS spectra of GM Aurigae and DM Tauri. The Astrophysical Journal, 630(2), L185.

[19] Rendon, F. & Nagel, E. (2015). Geometry of Internal Walls and Gaps in ´ Transitional and Pre-transitional Disks. IAU General Assembly, 22, 25264. Smith,

B. A., & Terrile, R. J. (1984). A circumstellar disk around Beta Pictoris. Science, 226, 1421–1424.

[20] Mihalas, D. (1978). Stellar atmospheres. San Francisco: WH Freeman.

[21] Larson, Ron, Hostetler, Robert. 2008. Precalculo.

[22] Lehmann, C. H. (1980). Geometría Analítica. LIMUSA.

[23] Varberg, D. E., Rigdon, S. E., Purcell, E. J. (2007) Calculo.

[24] Lehmann, C. H. (1989). Geometría Analítica. LIMUSA.

[25] Horowitz, E., & Sahni, S. (1982). Fundamentals of data structures. Computer Science Press.

[26] Garrido, C. A. y Fernández-Valdivia, Joaquín. (2006). Abstracción y estructuras de datos em C++. E.T.S de Ingeniería Informática y Telecomunicaciones. Universidad de Granada.

[27] G. Brassard, P. Bratley. 1997. fundamentals of algorithmics.

[28] Chapman, S., J. (2017). Fortran for scientists and engineers. McGraw Hill.

[29] A. L. Kraus y M. J. Ireland. LkCa 15: A Young Exoplanet Caught at Formation? En: ApJ 745, 5 (ene. de 2012), pág. 5. doi: 10.1088/0004- 637X/745/1/5. arXiv: 1110.3808 [astro-ph.EP].