SIMULACIÓN DE ESCENARIOS DE RADAR USANDO MATLAB

Joseph Rivadeneira-Aguirre

Piura, diciembre de 2014

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Mecánico-Eléctrica

Rivadeneira, J. (2014). S*imulación de escenarios de radar usando MATLAB.* Tesis de pregrado no publicado en Ingeniería Mecánico Eléctrica. Universidad de Piura. Facultad de Ingeniería. Programa Académico de Ingeniería Mecánico Eléctrica. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

SIMULACIÓN DE ESCENARIOS DE RADAR USANDO MATLAB

Tesis para optar el Título de
Ingeniero Mecánico Eléctrico

Joseph Jonathan Rivadeneira Aguirre

Asesor: Dr. Ing. César Chinguel Arrese

Piura, Diciembre 2014

Dedico el presente trabajo de investigación a mis padres, a quienes les debo lo que soy, a mi hermano quién siempre me dio ánimos para seguir adelante, y a mis profesores que siempre creyeron en mí y me brindaron todo su apoyo.

Prólogo

El presente trabajo de investigación se enmarca en un proyecto que realiza la Universidad de Piura en conjunto con los Servicios Industriales de la Marina (SIMA, Perú), donde se busca desarrollar tecnología local para mejorar las embarcaciones nacionales. El objetivo global del proyecto, se orienta a desarrollar un Sistema Integral de Comando y Control (SCC) de un buque. Este SCC debe recibir información de diversos radares de navegación y con esta información, deberá calcular la localización de embarcaciones u otros elementos cercanos.

La presente investigación surge de la necesidad de hacer más eficientes los algoritmos de detección y tracking, ayudándose de los distintos escenarios de radar que se podrán simular usando como herramienta el software Matlab; con la gran ventaja de ahorro de tiempo y de recursos económicos debido a que solo se necesitará tener una computadora con el software Matlab instalado en su sistema operativo.

Por otro lado, quiero agradecer a todas las personas que hicieron posible y colaboraron con el desarrollo de la presente investigación; como es el Doctor Ingeniero César Chinguel Arrese, mi Asesor en el presente trabajo de investigación, los Ingenieros Diego Purizaga y Marco Pérez quiénes hicieron aportes valiosos para enriquecer el marco teórico de la presente investigación.

Resumen

En el presente trabajo de investigación se diseña un algoritmo para simular un escenario de radar, en el que participarán buques moviéndose a una velocidad constante y siguiendo una trayectoria especificada por el usuario.

Para la simulación de cada una de las trayectorias, se usarán las ecuaciones del modelo matemático de Fossen, que describe la cinemática del movimiento de un buque. Las ecuaciones de Fossen se adecuarán al caso de dos dimensiones, debido a que los resultados de la simulación se presentarán en un PPI o se mostrarán sobre una imagen real obtenida por un radar de superficie SPQ.

El algoritmo es elaborado con Matlab, versión R2013a, y para su ejecución es acondicionado en una GUI, elaborada también en Matlab, con la finalidad de presentar al usuario las acciones disponibles del algoritmo de una manera sencilla y amigable.

En el primer capítulo se presenta la teoría de sistemas de radar y la ecuación de radar, en el segundo capítulo se hace un estudio de la "*Radar Cross Section*", en el tercer capítulo se estudia la cinemática de un buque mediante el modelo matemático de Fossen y por último en el cuarto capítulo se presenta la plataforma de simulación en Matlab.

Índice general

Introducción

Uno de los campos más trascendentales y explorados por la ingeniería radica en las aplicaciones orientadas a proporcionar seguridad a embarcaciones mediante la instalación de radares de superficie.

Actualmente, uno de los problemas más atendidos por diferentes campos de ingeniería es la Navegación de Móviles, ya que estas técnicas se usan en sistemas de vuelo, embarcaciones marinas, manipuladores autónomos inalámbricos, etc. La atención a este problema abarca la implementación de sistemas electrónicos cada vez más complejos y la aplicación de procesamiento de señales cada vez más sofisticadas.

El presente trabajo de investigación, centra su atención en el análisis de la eficacia de algoritmos de detección y tracking, usando como herramienta principal de ensayo un algoritmo simulador de radar usando Matlab, en el que se pueden simular diversos escenarios de radar, desde un escenario muy simple hasta un escenario complejo; con la gran ventaja del ahorro de tiempo y dinero, ya que sólo se requiere de un computador con la herramienta de Matlab instalada en su sistema operativo.

Capítulo 1
Fundamentos teóricos

1.1. Teoría de sistemas de radar

Los sistemas de radar se clasifican de acuerdo a distintos criterios, (Jarabo Amores, 2009) los que se listan a continuación:

- Según el tipo de blanco:
  - Primarios.- La señal recibida es el resultado de la reflexión o dispersión de la onda transmitida por el propio sistema al incidir sobre un objeto.
  - Secundarios.- Poseen sistemas de identificación de blancos en los que se transmite una señal codificada, esperando respuesta del blanco. El radar interroga al blanco, que responde, normalmente con una serie de datos (altura del avión, velocidad, etc.)
- Según la posición relativa del transmisor y receptor
  - Monoestáticos.- Transmiten y reciben la señal mediante una antena en común.
  - Multiestáticos.- Poseen dos o más antenas transmisoras o receptoras separadas distancias mayores que su tamaño.
- Según su finalidad
  - De vigilancia o exploración (*scanning radar*).- Explora todo el espacio, o un sector de él, mostrando todos los blancos que aparecen.
  - De seguimiento (*tracking radar*).- Es capaz de seguir el movimiento de un blanco.

  - Multifunción (*scanning & tracking radar*).- Son radares capaces de explorar el espacio y seguir blancos.

- Según su resolución

  - Convencionales.- Detectan blancos puntuales en dos dimensiones.

  - Alta resolución.- Son sistemas de radar tridimensionales, es decir son capaces de determinar la altura del blanco, además de su posición en el plano. Dentro de este grupo están los radares de imágenes laterales o radares de apertura sintética (SAR: *Syntetic Aperture Radar*) y los radares de alta resolución en distancia (HRR: *High Resolution Radar*).

- Según el tipo de señal

  - Radar de onda continua (CW).- Transmite ininterrumpidamente una señal de exploración.

  - Radar de onda continua con modulación (CW-FM, CW-PM).- Se le añade a la señal modulación de fase o frecuencia con objeto de determinar cuándo se transmitió la señal correspondiente a un eco (permite estimar distancias)

1.1.1. ¿Qué es radar?

La palabra radar es un acrónimo formado a partir de las primeras letras de las palabras inglesas *radio detection and ranging*, que significan: detección y medición de distancia empleando las ondas de radio. Las aplicaciones actuales del radar rebasan el significado del citado acrónimo, ya que esta técnica se emplea también en la determinación de los parámetros del movimiento del objeto que es iluminado por las ondas de radio (velocidad, aceleración y trayectoria), así como de su tamaño, de la forma del mismo, de la naturaleza de los materiales que lo constituyen, etc. (Chávez Ferry & Cabrera, 2012).

Los objetos que el radar ilumina o de los que recibe las señales emitidas por los mismos se denominan blancos u objetivos. Como ejemplo de blancos u objetivos de radar pueden citarse los siguientes: naves, aeronaves, proyectiles, satélites artificiales, hidrometeoros, la superficie de la mar, la ionósfera, vehículos terrestres, personas, montañas, etc.

El radar es un sistema muy complejo, integrado por dispositivos radioelectrónicos, con un alto nivel de automatización, donde el procesamiento digital de las señales y las computadoras son elementos esenciales.

El radar emite señales de sondeo, se entiende por señales de sondeo a aquellas que son emitidas al espacio con el fin de encontrar a su paso uno o más blancos en los cuales poder reflejarse. Estas señales de sondeo pueden ser, desde pulsos de radiofrecuencia muy breves, del orden de varios nanosegundos, como es el caso de las señales de banda ultra

ancha, hasta señales de onda continua; desde las más sencillas, como pudiera ser el caso de una señal monocromática de onda continua, hasta señales constituidas por pulsos de banda ancha obtenidas mediante la modulación adecuada de la frecuencia de la portadora o la manipulación de su fase o de su frecuencia. (Chávez Ferry & Cabrera, 2012).

1.1.2. Bandas de frecuencia empleadas en un sistema de radar

Las frecuencias en que operan los radares actualmente ocupan la región del espectro electromagnético comprendida desde aproximadamente 2 MHz (λ=150m) hasta 3000 GHz (λ=0,1 mm.). La banda de frecuencia empleada por un radar depende de la misión o fin para el cual ha sido diseñado. El empleo de las frecuencias más altas dentro de esta región permite lograr, utilizando antenas de pequeñas dimensiones, exactitudes y poderes resolutivos elevados.

Por otra parte, la utilización de las frecuencias más bajas dentro de la citada región facilita obtener mayores potencias de transmisión a la vez que pueden extenderse los alcances de localización más allá del límite establecido por el horizonte de radio. Claro está, no son sólo estas las causas del empleo de unas u otras frecuencias en radar; existen otras, tales como el efecto de la difracción y la emisión (irradiación) secundaria al tener en cuenta las características reflectoras de los blancos, la obtención de patrones de radiación específicos teniendo en cuenta la influencia de la superficie de la tierra, la absorción de la energía electromagnética por los gases presentes en la atmósfera, etc.

En la tabla 1.1 se muestra la región del espectro electromagnético ocupada por el radar, la nomenclatura empleada para designar las diferentes bandas de frecuencias utilizadas por el mismo y la aplicación específica de cada una de ellas.

1.2. La ecuación del radar

La distancia de un objetivo, su velocidad y dirección pueden ser medibles con gran precisión siempre que la variación entre señal y ruido de retorno (SNR, *signal to noise ratio*) sea muy alta. Si, el SNR es bajo, la distancia de un objetivo, su velocidad y dirección se pueden determinar también, con cierta precisión, siempre que este objetivo se esté moviendo muy cerca del radar.

Por otro lado, se puede decir que la ecuación del radar es una fuente de información que nos indica como retorna la señal después de haberse reflejado en un objetivo y además nos da información de cómo es el SNR.

Tabla 1.1. Bandas de frecuencia utilizadas por un radar

Banda	Rango de frecuencia	Utilización en radar
HF	3-30 MHz	Exploración de muy largo alcance
VHF	30-300 MHz	Exploración de mediano y largo alcance
UHF	300 – 1 GHz	Exploración de mediano y largo alcance
L	1-2 GHz	Exploración de largo alcance
S	2-4 GHz	Exploración de mediano alcance
C	4-8 GHz	Seguimiento de largo alcance
X	8-12 GHz	Seguimiento de corto alcance, Guiado de misiles, Radares de navegación marítima y aérea , control de tiro de la Artillería Antiaérea.
K_u	12-18 GHz	Cartografía de alta resolución, Satélites.
K	18-27 GHz	Muy poco empleado, debido a la gran absorción por vapor de agua.
K_σ	27-40 GHz	Cartografía de muy alta resolución
Milimétrica	40-300 GHz	Experimental

Fuente: (Chávez Ferry & Cabrera, 2012)

1.3. Densidad de potencias en un sistema de radar

Se asume que un radar transmite un pulso con potencia P_T. Si la antena de transmisión del radar tiene un patrón de radiación isotrópica, entonces irradiaría la potencia con simetría esférica; por lo tanto, el flujo de potencia por unidad de área (densidad de potencia) en un alcance determinado "R" es:

$$Densidad\ de\ Potencia = \frac{P_T}{4\pi R^2}\left[\frac{W}{m^2}\right] \qquad (\mathbf{1.1})$$

Donde:

$P_T = Potencia\ en\ la\ transmisión\ [W]$

$R = Alcance\ [m]$

Se asume que la antena tiene una ganancia de transmisión G_T y está apuntando en la dirección del objetivo, entonces la densidad de potencia queda multiplicada por esa ganancia de transmisión. Ahora, asumimos que el objetivo

refleja toda la potencia interceptada por su área efectiva y además que el patrón de reflexión es isotrópico. Si el área efectiva del objetivo isotrópico es σ, entonces la potencia que se refleja isotrópicamente está dada por:

$$Potencia\ reflejada = \frac{P_T G_T \sigma}{4\pi R^2} \qquad (\mathbf{1.2})$$

Donde:

$P_T = Potencia\ en\ la\ transmisión\ [W]$

$R = Alcance\ [m]$

$\sigma = Área\ efectiva\ de\ reflexión\ del\ objetivo\ [m^2]$

$G_T = Ganancia\ de\ transmisión\ de\ la\ antena$

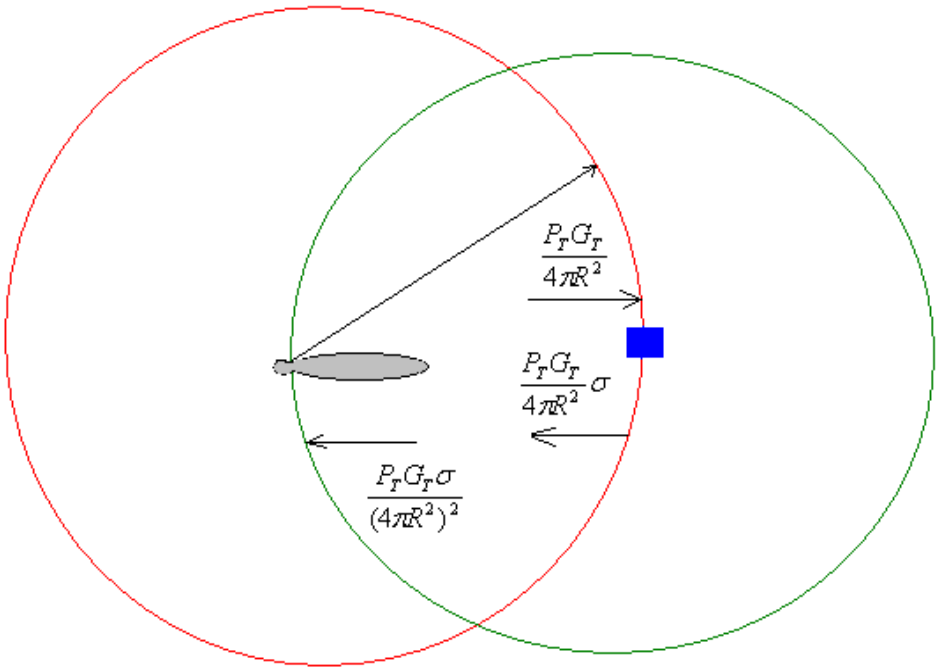


Figura 1.1. Densidades de potencia en un escenario de radar.
Fuente: Elaboración propia.

Dado que la potencia es reflejada con un patrón isotrópico, entonces se puede decir que es reflejada con simetría esférica; por lo tanto, la densidad de potencia reflejada será:

$$Densidad\ de\ potencia\ reflejada = \frac{Potencia\ reflejada}{Área\ esférica}$$

$$Densidad\ de\ potencia\ reflejada = \frac{\frac{P_T G_T \sigma}{4\pi R^2}}{4\pi R^2}$$

$$Densidad\ de\ potencia\ reflejada = \frac{P_T G_T \sigma}{(4\pi R^2)^2}\left[\frac{W}{m^2}\right] \qquad (\mathbf{1.3})$$

Donde:

$P_T = Potencia\ en\ la\ transmisión\ [W]$

$R = Alcance\ [m]$

$\sigma = Área\ efectiva\ de\ reflexión\ del\ objetivo\ [m^2]$

$G_T = Ganancia\ de\ transmisión\ de\ la\ antena$

Teniendo en cuenta que la antena del radar tiene un área efectiva para recibir la señal reflejada, entonces la expresión (1.3) queda afectada por esta área efectiva convirtiéndose así en la potencia reflejada:

$$Potencia\ reflejada = \frac{P_T G_T A_e \sigma}{(4\pi R^2)^2}\ [W] \qquad (\mathbf{1.4})$$

Donde:

$P_T = Potencia\ en\ la\ transmisión\ [W]$

$R = Alcance\ [m]$

$\sigma = Área\ efectiva\ de\ reflexión\ del\ objetivo\ [m^2]$

$G_T = Ganancia\ de\ transmisión\ de\ la\ antena$

$A_e = Área\ efectiva\ de\ la\ antena\ [m^2]$

Ahora tendremos que analizar la relación que existe entre la ganancia y la apertura de la antena, con la finalidad de obtener una expresión que nos brinde mayor información acerca del radar. Para hallar esta relación usaremos dos antenas parabólicas puestas una frente de otra. Observemos la figura 1.2, allí podemos ver el esquema de una antena parabólica.
Considerando que el ángulo θ es muy pequeño y teniendo en cuenta la geometría se podrá afirmar que:

$$P'P'' = D \times \sin\theta$$

$$P'P'' \approx D \times \theta$$

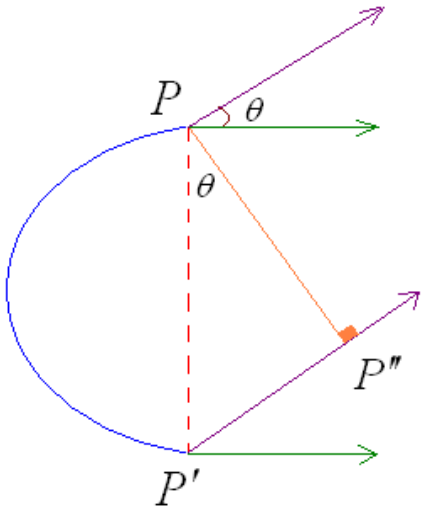


Figura 1.2. Distintas trayectorias de los bordes de la apertura de una antena a una dirección inclinada.
Fuente: Elaboración propia.

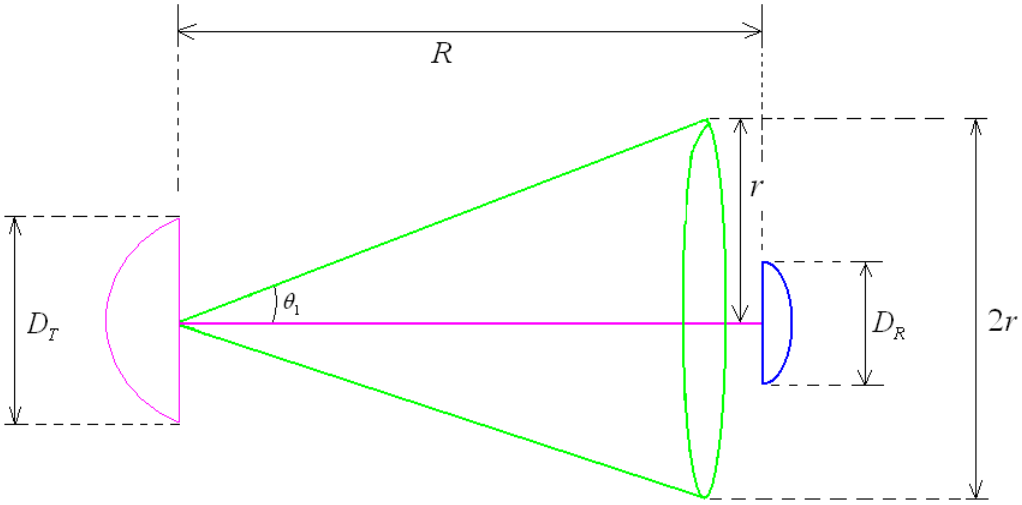


Figura 1.3. Geometría de iluminación entre dos antenas parabólicas.
Fuente: Elaboración propia.

Si analizamos la figura 3, mostrada en la parte inferior, podremos hallar las siguientes relaciones:

$$\theta_1 = \frac{\lambda}{2 \times D_T} \qquad (\mathbf{1.5})$$

$$r = R\theta_1 \qquad (\mathbf{1.6})$$

Reemplazando (1.5) en (1.6) obtenemos:

$$r = R \times \left(\frac{\lambda}{2 \times D_T}\right) \qquad (\mathbf{1.7})$$

Por lo tanto:

$$2 \times r = 2 \times R \times \left(\frac{\lambda}{2 \times D_T}\right)$$

Simplificando la expresión anterior obtenemos:

$$2r = R \times \left(\frac{\lambda}{D_T}\right) \qquad (\mathbf{1.8})$$

Como se dijo líneas arriba, trataremos de hallar una relación entre la apertura de antena y la ganancia.

La potencia de transmisión se puede expresar como:

$$P_T = \frac{\pi}{4}\left(\frac{R\lambda}{D_T}\right)^2 \qquad (\mathbf{1.9})$$

La potencia de recepción se puede expresar como:

$$P_R = \frac{\pi}{4} \times {D_R}^2 \qquad (\mathbf{1.10})$$

Si relacionamos (1.10) y (1.9):

$$\frac{P_R}{P_T} = \frac{\frac{\pi}{4} \times {D_R}^2}{\frac{\pi}{4}\left(\frac{R\lambda}{D_T}\right)^2}$$

Simplificando la expresión anterior:

$$\frac{P_R}{P_T} = \frac{{D_R}^2}{\left(\frac{R\lambda}{D_T}\right)^2}$$

$$\frac{P_R}{P_T} = \frac{{D_R}^2 {D_T}^2}{(R\lambda)^2} \qquad (\mathbf{1.11})$$

Sabemos que:

$$A_R = \frac{\pi}{4} D_R^2 \rightarrow D_R^2 = \frac{4A_R}{\pi} \qquad (\mathbf{1.12})$$

$$A_T = \frac{\pi}{4} D_T^2 \rightarrow D_T^2 = \frac{4A_T}{\pi} \qquad (\mathbf{1.13})$$

Reemplazando (1.12) y (1.13) en (1.11):

$$\frac{P_R}{P_T} = \frac{\left(\frac{4A_R}{\pi}\right)\left(\frac{4A_T}{\pi}\right)}{(R\lambda)^2} \rightarrow \frac{P_R}{P_T} = \frac{16A_RA_T}{\pi^2(R\lambda)^2} \rightarrow \frac{P_R}{P_T} = 1.62\,\frac{A_RA_T}{(R\lambda)^2}$$

El factor 1.62 es el resultado de varias simplificaciones, por lo tanto deducimos que el factor verdadero es 1, entonces la expresión anterior queda:

$$\frac{P_R}{P_T} = \frac{A_RA_T}{(R\lambda)^2} \qquad (\mathbf{1.14})$$

Por otro lado, se puede afirmar que la potencia recibida (y reflejada) por un área$A_R = \sigma$; y según la ecuación (1.2) sabemos que la potencia reflejada (P_R) es:

$$P_R = \frac{P_TG_T\sigma}{4\pi R^2}$$

Reemplazando σ por A_R y pasando P_T a dividir al otro miembro de la ecuación, la expresión anterior quedará como:

$$\frac{P_R}{P_T} = \frac{G_TA_R}{4\pi R^2} \qquad (\mathbf{1.15})$$

Igualaremos las ecuaciones (1.14) y (1.15):

$$\frac{A_RA_T}{(R\lambda)^2} = \frac{G_TA_R}{4\pi R^2}$$

Simplificando la expresión anterior:

$$\frac{A_T}{(\lambda)^2} = \frac{G_T}{4\pi}$$

$$A_T = \frac{G_T}{4\pi}\,(\lambda)^2 \qquad (\mathbf{1.16})$$

Finalmente, para llegar a la ecuación básica del radar, reemplazamos (1.16) en (1.4), sabiendo que $A_T = A_e$:

$$P_R = \frac{P_TG_T\sigma\left(\frac{G_T\lambda^2}{4\pi}\right)}{\left(4\pi R^2\right)^2}$$

Simplificando y acomodando la expresión anterior, obtenemos **la ecuación básica del radar**:

$$P_R = \frac{P_T G_T^2 \sigma \lambda^2}{(4\pi)^3 R^4} \qquad (\mathbf{1.17})$$

1.4. Relación señal ruido

Otra forma de la ecuación del radar es el SNR (*Signal to Noise Ratio*) en función del alcance. Se puede decir que la expresión hallada anteriormente como ecuación del radar, es ideal; esto es debido a que en su estructura no toma en cuenta el ruido que está presente en el medio en que se propagan las ondas de radar.

Debemos tener claro que el ruido siempre estará presente en un sistema de radar e influirá en las señales de radar; además debemos saber que existen distintos tipos de ruido; por ejemplo el ruido térmico que puede ser considerado ruido blanco para efectos prácticos en los sistemas de radar. (Kingsley & Quegan, 1992, pág. 41).

A continuación definiremos lo que es ancho de banda y potencia del ruido. El ancho de banda (f_B) está definido por la siguiente expresión:

$$f_B = \frac{1}{t_p}\ [Hz] \qquad (\mathbf{1.18})$$

Donde:

t_p es la duración de un pulso de radar en segundos.

A continuación definiremos potencia del ruido térmico (N):

$$N = FKT_E f_B \qquad (\mathbf{1.19})$$

$$N = N_0 f_B \qquad (\mathbf{1.20})$$

Donde:

$F = es\ la\ forma\ del\ ruido\ en\ el\ receptor.$

$K = es\ la\ constante\ de\ Boltzmann = 1.38 \times 10^{-23} \left[\frac{W.s}{K}\right]$

$N_0 = es\ la\ densidad\ de\ potencia\ espectral \left[\frac{W}{Hz}\right]$

$T_E = es\ la\ temperatura\ en\ grados\ Kelvin\ [K]$

Ahora como ya tenemos una expresión para la potencia del ruido y tenemos la ecuación ideal de la potencia reflejada del radar (ecuación básica del radar hallada en el apartado anterior), estamos en condiciones de hallar la relación

señal ruido (SNR). Podemos definir a la relación señal ruido o *Signal to Noise Ratio* como:

$$SNR = \frac{Potencia\ de\ la\ señal\ ideal\ en\ la\ recepción}{Potencia\ del\ ruido}$$

Finalmente, como son inevitables las ineficiencias en un sistema de radar, entonces éstas son consideradas en la ecuación del SNR con un factor de pérdidas L_S, el que puede ser arreglado para aparecer en el numerador o denominador de la ecuación; sin embargo en la presente investigación se adoptará la convención de que siempre el factor L_S, será menor que 1, por lo tanto siempre aparecerá en el numerador de la ecuación del SNR.

Reemplazando la ecuación básica del radar hallada en el apartado anterior, la ecuación para la potencia del ruido y el factor de pérdidas en la expresión del SNR, obtenemos lo siguiente:

$$SNR = \frac{P_T G_T^2 \sigma \lambda^2 L_S}{(4\pi)^3 R^4 N_0 f_B} \qquad (\mathbf{1.21})$$

Trabajando la expresión anterior y despejando el alcance (R), podemos obtener una expresión para determinar cuál será la máxima distancia de detección del radar:

$$R_{max} = \left(\frac{P_T G_T^2 \sigma \lambda^2 L_S}{(4\pi)^3 SNR(N_0 f_B)} \right)^{1/4} \qquad (\mathbf{1.22})$$

Capítulo 2
"Radar cross section"

1.1. Definición de *radar cross section*

Las ondas electromagnéticas, con alguna polarización determinada[1], son difractadas o dispersadas en todas direcciones cuando inciden sobre un objetivo. Estas ondas dispersadas se dividen en dos partes. La primera parte está compuesta de ondas que tienen la misma polarización que la antena receptora. La otra parte de ondas dispersadas tendrán diferente polarización para la que la antena receptora no responderá. Las dos polarizaciones son ortogonales y están referidas al Principio de Polarización (PP) y Polarización Ortogonal (OP), respectivamente. (Mahafza, 2000). La intensidad de energía re-irradiada que tiene la misma polarización que la antena receptora del radar es usada para definir el RCS del objetivo.

Asumimos la densidad de potencia de una onda incidente en un objetivo localizado a una distancia R del radar es P_{Di} . La cantidad de potencia reflejada desde el objetivo es:

$$P_r = \sigma P_{Di} \qquad (\mathbf{2.1})$$

σ denota la sección transversal del objetivo. Definimos P_{Dr} como la densidad de potencia de las ondas dispersadas que llegan a la antena receptora:

$$P_{Dr} = \frac{P_r}{4\pi R^2} \qquad (\mathbf{2.2})$$

Reemplazando la ecuación (2.1) en (2.2) y despejando σ :

$$\sigma = 4\pi R^2 \left(\frac{P_{Dr}}{P_{Di}}\right) \qquad (\mathbf{2.3})$$

[1] Hace referencia a la dirección de polarización de la antena. La dirección de polarización está definida por la dirección del vector de campo eléctrico. La mayoría de antenas de radar son polarizadas linealmente; esto es, la dirección del vector de campo eléctrico es vertical u horizontal. La polarización puede ser también elíptica o circular. (Skolnik, 1980, pág. 227).

Y para generalizar la ecuación anterior, la modificamos de la siguiente manera:

$$\sigma = 4\pi R^2 \lim_{R\to\infty}\left(\frac{P_{Dr}}{P_{Di}}\right) \qquad (\mathbf{2.4})$$

Entonces el RCS queda definido por la ecuación (2.4). (Mahafza, 2000). Por lo tanto, podemos decir que el RCS (*Radar Cross Section*) o sección transversal de radar es la medida de la energía electromagnética que un objetivo dispersa en una cierta dirección normalizada hacia una antena receptora respecto a la densidad de la onda incidente de un radar que lo ha iluminado.

El RCS re-irradiado (*backscattered RCS*) es medido según las ondas dispersadas en la dirección del radar y que tienen la misma polarización que la antena receptora. Esta energía re irradiada representa una porción del total de RCS dispersado por el objetivo (σ_t) , donde $(\sigma_t > \sigma)$. Asumiendo un sistema de coordenadas esféricas definido por (ρ, θ, φ) , entonces para un alcance ρ el *cross section* dispersado del objetivo es una función de (θ, φ) . Los ángulos (θ_i, φ_i) definen la dirección de propagación de las ondas incidentes. Los ángulos (θ_s, φ_s) definen la dirección de propagación de las ondas dispersadas. El caso especial, cuando $\theta_s = \theta_i$ y $\varphi_s = \varphi_i$ define la RCS monoestática.[2] La sección transversal medida por el radar, cuando los ángulos son $\theta_s \neq \theta_i$ y $\varphi_s \neq \varphi_i$ es llamada RCS biestática.[3]La RCS total dispersada por un objetivo está dada por:

$$\sigma_t = \frac{1}{4\pi}\int_{\varphi_s=0}^{2\pi}\int_{\theta_s=0}^{\pi}\sigma\,(\theta_s, \varphi_s)\sin\theta_s\, d\theta\, d\varphi_s \qquad (\mathbf{2.5})$$

La cantidad de ondas dispersadas desde un objetivo es proporcional a la relación de la magnitud del objetivo (tamaño) con la longitud de onda (λ) de las ondas incidentes.

En efecto, un radar no podrá detectar objetivos más pequeños que la longitud de onda con la que está operando. Por ejemplo, si un radar climático usa frecuencias en la banda L, las gotas de lluvia serían casi invisibles al radar debido a que éstas son más pequeñas que la longitud de onda. RCS medidas en la región de frecuencias, donde la magnitud del objetivo y la longitud de onda son comparables, están referidas a la región de Rayleigh. Alternativamente, la región de frecuencia donde la magnitud del objetivo es más grande que la longitud de onda de operación del radar está referida a la región óptica. En la práctica, la mayoría de aplicaciones de radar se encuentran dentro de la región óptica. (Mahafza, 2000).

[2] RCS monoestática: sección transversal de radar medida por un radar monoestático, es decir un radar que posee una antena que funciona como emisora y receptora a la vez, alternando esta función mediante un *switch.*

[3] RCS biestática: sección transversal de radar medida por un radar biestático, es decir un radar que posee la antena receptora en un lugar distinto y alejado de la antena emisora.

1.2. Técnicas de predicción de *radar cross section*

Antes de presentar los diferentes métodos de cálculo de RCS, es importante entender el significado de predicción de RCS. Muchos sistemas de radar usan RCS como un instrumento de discriminación. Por lo tanto, la precisión de la predicción de la RCS de un objetivo es importantísima en orden de diseñar y desarrollar algoritmos de discriminación robustos. Adicionalmente, midiendo e identificando los centros de dispersión (fuentes) para un objetivo dado, ayuda al desarrollo de técnicas de reducción de RCS.
Existen dos categorías de técnicas o métodos de predicción de RCS:

- Métodos exactos
- Métodos aproximados

Los métodos exactos de predicción de RCS son muy complejos, aún para objetos de forma simple. Esto se debe porque estos métodos requieren resolver ecuaciones diferenciales e integrales que describen las ondas dispersadas de un objeto bajo unas condiciones de frontera. Estas condiciones de frontera son gobernadas por las ecuaciones de Maxwell. Aún cuando las soluciones son obtenidas, éstas son a menudo difíciles de interpretar y programar usando computadores.

Debido a las dificultades asociadas con la predicción exacta de RCS, los métodos aproximados llegan a ser una alternativa viable. La mayoría de métodos aproximados son validados en la región óptica, y cada uno tiene sus propias fortalezas y limitaciones. Muchos de los métodos aproximados pueden predecir la RCS con un margen de error muy pequeño, por lo que los valores de RCS obtenidos son muy cercanos a valores los reales. Esto hace que los métodos aproximados sean la principal fuente para predecir la RCS de objetivos complejos como aviones, barcos y misiles. Cuando se dispone de RCS experimentales, pueden ser usadas para validar y verificar los resultados obtenidos por los métodos aproximados. (Mahafza, 2000).

Algunos de los métodos aproximados comúnmente utilizados se listan a continuación[4]:

- Método de óptica geométrica (GO)
- Método de óptica física (PO)
- Teoría geométrica de difracción(GTD)
- Teoría física de difracción (PTD)
- Método de corrientes equivalentes (MEC)

[4] Las abreviaturas mostradas entre paréntesis, al listar los métodos aproximados para la predicción de RCS, responden a los nombres en Inglés de cada uno de éstos métodos, como se detalla a continuación:
GO : *Geometrical Optics*
PO : *Physical Optics*
GTD : *Geometrical Theory of Diffraction*
PTD : *Physical Theory of Diffraction*
MEC : *Method of Equivalent Currents*

Para brindar más información acerca de los métodos de predicción de la RCS de un objetivo, he considerado oportuno añadir algunos párrafos importantes del Proyecto de trabajo de grado de Sara Isabel Pérez Restrepo de la Universidad Pontificia Bolivariana, Escuela de Ingenierías, Facultad de Ingeniería Aeronáutica. A continuación presento algunos acápites importantes del mencionado trabajo de grado (Pérez Restrepo, 2013) :

Debido a que la RCS depende de la frecuencia de la onda incidente, es necesario nombrar las tres regiones de frecuencia donde la RCS de un objetivo es diferente. Las regiones se definen basándose en el tamaño del objetivo en términos de la longitud de onda incidente. Para un objetivo uniforme de longitud L, las tres regiones de frecuencia son: (Chatzigiorgiadis, 2004)

- **Región de Rayleigh o región de baja frecuencia:**

  Tamaño típico del cuerpo < λ. A estas frecuencias la variación de fase de la onda plana incidente a través de la extensión del objetivo es pequeña. Por lo tanto, la corriente inducida en el cuerpo del objetivo es aproximadamente constante en amplitud y fase. La forma particular del cuerpo no es importante.

- **Región de Mie o región de resonancia:**

  λ < tamaño típico del cuerpo < 3λ. Para estas frecuencias, la variación de fase de las corrientes a través del cuerpo del objetivo es significativa y todas las partes contribuyen a la dispersión.

- **Región óptica o región de alta frecuencia:**

  3λ < tamaño típico del cuerpo**.** Para estas frecuencias hay muchos ciclos en la variación de fase de las corrientes a través del cuerpo del objetivo y en consecuencia, el campo disperso será muy dependiente del ángulo.

1.2.1. Métodos exactos o métodos analíticos

La RCS de cuerpos simples puede ser computada exactamente por una solución de la ecuación de onda en un sistema coordenado, para la cual, una coordenada constante coincide con la superficie del cuerpo:

$$\nabla^2 F + K^2 F = 0 \qquad (\mathbf{2.6})$$

Donde F representa cualquier vector de campo eléctrico E o campo magnético H y, k es el número de onda. La ecuación anterior es una ecuación diferencial de segundo orden que puede ser resuelta como un problema de condiciones en la frontera donde los campos sobre la superficie del obstáculo de dispersión son especificados. La solución exacta requiere que los campos eléctricos y magnéticos justo adentro y justo afuera de la superficie satisfagan ciertas condiciones que dependen de las propiedades electromagnéticas del material de la superficie del cuerpo. (Skolnik, Radar Handbook, 2008).

Esta solución analítica para la predicción de la RCS consiste en resolver las ecuaciones de Maxwell con las condiciones de contorno impuestas por la superficie del blanco (Rius Casals, 1991). Método que requiere de la solución de ecuaciones diferenciales o integrales, que describen las ondas dispersadas del blanco, bajo unas condiciones de frontera específicas (Díaz, 2005). La ecuación (2.6) se puede satisfacer con cada una de las tres componentes del campo F. Si se representa cualquiera de los componentes del vector por una función V, entonces V es una solución a la ecuación de onda escalar:

$$\nabla^2 V + K^2 V = 0 \qquad (\mathbf{2.7})$$

Para resolver esta ecuación usando separación de variables, se representa la función V en términos de otras tres funciones, cada una de las cuales depende solamente de una coordenada:

$$V(u_1, u_2, u_3) = V_1(u_1)V_2(u_2)V_3(u_3) \qquad (\mathbf{2.8})$$

Donde u1, u2, u3 representan las tres coordenadas. Estas podrían ser x, y, y z del sistema rectangular ó r, θ y ϕ del sistema esférico. La separación de variables se enfoca en resolver ecuaciones diferenciales parciales dando tres ecuaciones diferenciales ordinarias, donde cada una envuelve un par de constantes separadas. Estas constantes deben ser determinadas recurriendo a las condiciones de frontera del problema a resolver. (Knott, Tuley, & Shaeffer, 2004).

Cuando se utiliza este método, la ecuación de onda da la solución exacta para los campos totales en cualquier parte del espacio, y típicamente el campo incidente es expandido en términos de ondas elementales en el sistema coordenado inicialmente usado. Cuando el campo disperso es normalizado al campo incidente, se eleva al cuadrado y luego se multiplica por el área de una esfera cuyo radio es la distancia al punto de observación, se obtiene la RCS.

Aunque las soluciones exactas son realizables, son a menudo difíciles de interpretar y de programar. (Díaz, 2005).

1.2.2. Métodos de baja frecuencia

Los métodos numéricos de baja frecuencia se plantean a partir de las ecuaciones de Maxwell en su forma diferencial o a partir de ecuaciones integrales derivadas de las ecuaciones de Maxwell. Según el planteamiento utilizado se trata de métodos integrales o de diferencias finitas. Además, puede plantearse un problema integro-diferencial que lleva al método de elementos finitos (Rius Casals, 1991). Las ecuaciones electromagnéticas pueden formularse en el dominio del tiempo o de la frecuencia.

Aunque las soluciones analíticas se constituyen en ejercicios académicos interesantes y pueden en algunos casos describir los mecanismos de dispersión, éstas no se ajustan a blancos tácticos, por lo que es necesario utilizar otro tipo de métodos. Además, la complejidad existente para resolver la RCS por medio de métodos analíticos para cuerpos complejos o eléctricamente grandes, hace necesario el uso de ordenadores para el cálculo numérico y de métodos que calculen una aproximación del campo dispersado. En aplicaciones reales, estos métodos son prohibitivos al aplicarlos a cuerpos de gran tamaño, excepto para

radares de baja frecuencia, por esta razón, se le da este nombre a dichos métodos. (Pérez Restrepo, 2013).

En la actualidad, el avance de la tecnología permite disponer de velocidades de cálculo cada vez más grandes, lo que ha posibilitado realizar cálculos que en el pasado sólo se podían llevar a cabo con superordenadores. Por este motivo, cada día aumenta el tamaño de los cuerpos que pueden ser analizados por este tipo de método, al igual que los esfuerzos por desarrollar herramientas que permitan llegar a dicho objetivo. (Pérez Restrepo, 2013).

Un enfoque alternativo es la solución de ecuaciones integrales gobernadas por la distribución de los campos inducidos sobre la superficie del objetivo. La aproximación más utilizada es una solución por medio del método de los momentos (MoM), en el que, las ecuaciones integrales son reducidas a un sistema de ecuaciones lineales homogéneas. La razón para utilizar este método es que el perfil de superficie del cuerpo no tiene restricción, lo que permite el cálculo de la dispersión de cuerpos tácticos complejos. Otra razón es que métodos ordinarios de solución como inversión de matrices o eliminación gaussiana, pueden ser empleados para lograr una solución. Este tipo de métodos es limitado por memoria computacional y tiempo de ejecución (Skolnik, Radar Handbook, 2008).

1.2.2.1. Método de los momentos

Las ecuaciones integrales formuladas a partir de las ecuaciones de Maxwell son exactas y válidas para cuerpos muy grandes o muy pequeños comparados con la longitud de onda (Knott, Tuley, & Shaeffer, 2004). En este tipo de métodos, se destaca el método de momentos (MoM), el cual es muy similar al procedimiento Rayleigh Ritz para resolver ecuaciones integrales.

La formulación exacta calcula la influencia de cada parte del cuerpo con otra parte, por lo que todos los fenómenos electromagnéticos están incluidos: región especular, región final, difracción, rebote múltiple, *traveling, creeping,* entre otros. (Pérez Restrepo, 2013)

El método de los momentos es la técnica más usada para resolver ecuaciones integrales. En el caso de la predicción de la RCS por medio del método de los momentos (MoM), las ecuaciones integrales se derivan de las ecuaciones de Maxwell y de las condiciones de frontera. El método de los momentos reduce las ecuaciones integrales a un conjunto de ecuaciones lineales simultáneas que pueden ser resueltas utilizando álgebra matricial estándar. La mayoría de formulaciones en MoM requiere una discretización del cuerpo del blanco y por lo tanto, es compatible con el método de elementos finitos (Chatzigiorgiadis, 2004).

Este método provee una rigurosa solución del problema de predicción de la RCS, dando resultados muy exactos. Por este motivo, este método tiende a producir largas matrices, resultando en altos requerimientos computacionales e incrementando el tiempo de utilización de cualquier software. El MoM no es práctico para blancos grandes a altas

frecuencias, debido entonces a limitaciones computacionales. (Pérez Restrepo, 2013).

1.2.2.2. Método de elementos de contorno

El método de los elementos de contorno (BEM) ha sido utilizado en ingeniería civil para el cálculo de estructuras y poco a poco se ha introducido para la solución del problema electromagnético. Se basa en las ecuaciones integrales de reciprocidad de Lorentz, las cuales están formuladas en términos de campos y no de corrientes. La discretización se realiza sobre superficies que separan regiones homogéneas, utilizando técnicas del método de elementos finitos (Rius Casals, 1991). Este método, es considerado por algunos autores como un caso específico del método de momentos, debido a la elección de las funciones base y peso. (Pérez Restrepo, 2013, pág. 38).

1.2.2.3. Método de diferencias finitas

Se basa en la discretización de las ecuaciones de Maxwell en forma diferencial utilizando la técnica de las diferencias finitas (Rius Casals, 1991).
Este problema puede ser planteado en el dominio del tiempo o de la frecuencia.
En este tipo de métodos debe asegurarse la condición de radiación que evita que el campo dispersado se refleje de nuevo hacia el objeto al alcanzar los límites de la región discretizada (Rius Casals, 1991).

1.2.2.4. Método de elementos finitos

El método de los elementos finitos (FEM) se ha utilizado en la solución de problemas mecánicos. Se basa en la definición de funciones integro-diferenciales, cuya minimización conduce a la solución del problema. Este método se trata de una técnica variacional.
La discretización del problema en el espacio requiere de un mallado tridimensional denominado elementos finitos, sobre los que se aplican de forma analítica y numérica los operadores diferenciales e integrales. (Pérez Restrepo, 2013).
La aplicación del FEM requiere de la imposición de la componente de radiación, o la combinación de este método con el BEM, el cual, se aplica en la superficie del objeto y al ser un método integral lleva implícita la condición requerida. Las soluciones de ambos métodos se acoplan en la superficie que separa el problema externo del interno. (Rius Casals, 1991).

1.2.3. Métodos de alta frecuencia

Cuando las dimensiones características del objeto, como longitud, anchura y radios de curvatura de las superficies son grandes comparados con la longitud de onda, se aplican métodos de alta frecuencia. En este rango de banda, el obstáculo de dispersión debe ser por lo menos cinco longitudes de onda en tamaño, aunque

pueden obtenerse resultados razonables para objetos más pequeños que esto. (Knott, Tuley, & Shaeffer, 2004).
La simplicidad de los métodos de alta frecuencia se debe a que consideran la difracción como un fenómeno local: cada parte del objetivo difracta los campos incidentes de forma independiente de los demás centros de difracción. De esta forma, pueden aproximarse los campos inducidos en una región del objetivo como debidos únicamente al campo incidente, sin incluir el campo re-irradiado por otras partes del mismo. (Pérez Restrepo, 2013).
Dada la complejidad presentada en los métodos de baja frecuencia para el análisis de cuerpos eléctricamente grandes, es necesario utilizar métodos aproximados que requieran de un esfuerzo computacional relativamente menor para el cálculo de la RCS de cuerpos complejos. La región de alta frecuencia es de gran importancia, pues incluye la mayoría de los blancos de radar de interés a las frecuencias más habituales. (Rius Casals, 1991).

1.2.3.1. Óptica geométrica

La aproximación de óptica geométrica se basa en calcular la sección recta de radar a partir del producto de los radios principales de curvatura de la superficie en los puntos de reflexión especular.[5] (Rius Casals, 1991).
Este método analiza la propagación de la luz a frecuencias ópticas sin considerar la naturaleza ondulatoria de los campos electromagnéticos. Por esta razón, el método no es capaz de predecir la difracción en los obstáculos, sino únicamente la reflexión en superficies de curvatura suave y que obtenga siempre regiones de sombra con transiciones abruptas.
Para la predicción de la RCS, el método de óptica geométrica se basa en el concepto de los rayos reflejados por la superficie conductora según la ley de Snell (rayo incidente, reflejado y normal a la superficie en el mismo plano, con ángulos de incidencia y reflexión iguales). Por tanto, el método no podrá aplicarse a vértices ni aristas, en los que la normal a la superficie no está definida. (Rius Casals, 1991).
La dificultad del método aparece al intentar identificar el punto de reflexión especular y los radios de curvatura en ese punto. Por ello, el método no es utilizado para la solución de la RCS para objetos complejos ni es aplicable a superficies planas, cilíndricas o cónicas. (Pérez Restrepo, 2013).

1.2.3.2. Óptica física

El método de óptica física (PO) estima la corriente inducida sobre la superficie de un cuerpo arbitrario debido a la radiación incidente. Los campos radiados se obtienen a partir de la solución de la ecuación integral de Stratton – Chu en campo lejano mediante la aproximación del plano tangente. Las ecuaciones integrales de Stratton-Chu dan el

[5] Reflexión especular: ocurre cuando la dirección de un rayo reflejado está en el plano perpendicular a la superficie reflectora que contiene al rayo incidente. (Serway & Jewett, 2008).

campo dispersado por el obstáculo a partir del campo total sobre la superficie del mismo:

$$\vec{E} = \oint_S \left[-ik\eta(\hat{n} \times \vec{H})G + (\hat{n} \cdot \vec{E})\nabla G\right]ds \qquad (\mathbf{2.9})$$

$$\vec{H} = \oint_S \left[ik\frac{1}{\eta}(\hat{n} \times \vec{E})G + (\hat{n} \times \vec{H}) \times \nabla G + (\hat{n} \cdot \vec{H})\nabla G\right]ds \qquad (\mathbf{2.10})$$

Donde:

k: número de onda

$\hat{n}$: vector normal a la superficie

η: impedancia de la onda en el vacío

G: función de Green[6] en el espacio libre

Estas ecuaciones son válidas para analizar la dispersión de una superficie cerrada, ya que si la superficie es abierta es necesario añadir integrales de superficie adicionales para limitar la superficie libre.
El método de óptica física permite hacer dos simplificaciones inmediatas, una de ellas es la aproximación en campo lejano, en el que la distancia R desde el objeto hasta el punto de observación es mucho mayor que cualquier dimensión del obstáculo. Bajo las condiciones de campo lejano, las integrales de línea de las ecuaciones anteriores pueden ser aproximadas como integrales de superficie; de allí resulta la otra simplificación, la cual propone que no habrá ninguna componente de la distribución del campo a lo largo de la dirección de dispersión de la superficie.
En el cálculo de objetos complejos en alta frecuencia las componentes de los campos dispersos pueden ser calculadas y añadidas antes de elevar al cuadrado para obtener la potencia dispersada. Así se mantendrá la relación de fase entre los distintos objetos de dispersión, de manera que los efectos de interferencia son adecuadamente representados. (Knott, Tuley, & Shaeffer, 2004).

1.3. *Radar cross section* de objetos simples

Para objetivos simples, como una placa plana rectangular, cilindros, esferas, la RCS puede ser calculada usando las ecuaciones de Maxwell con ciertas condiciones de

[6] Las funciones de Green constituyen una herramienta para abordar problemas con ecuaciones diferenciales no homogéneas bajo ciertas condiciones de contorno (Pérez Alcaraz, 2012). Muchos problemas en electrostática involucran la determinación del campo eléctrico en regiones finitas del espacio que pueden contener distribuciones de carga y que se encuentran limitadas por superficies sujetas a determinadas condiciones de frontera; para encontrar la solución a este tipo de problemas resultan útiles las funciones de Green. (Cosenza, 2012, pág. 59).

frontera. (Harre, 2004). La figura 2.1 muestra tres objetos simples con la dimensión principal de 1m y sus RCS:

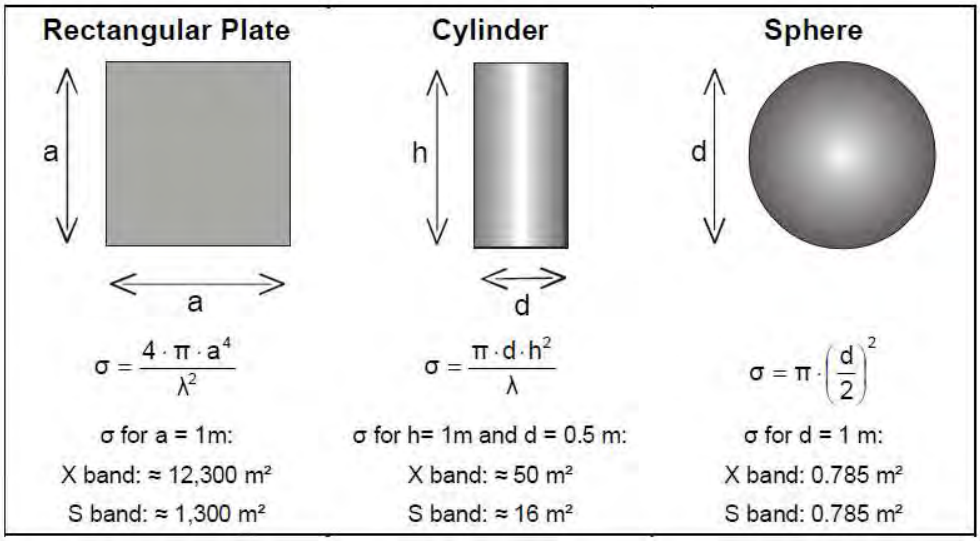


Figura 2.1. RCS de Objetivos Simples.
Fuente: (Harre, 2004)

1.4. *Radar cross section* de un buque

De estudios experimentales, para el cálculo de *radar cross section* de objetivos marítimos, se logró construir una tabla de valores típicos de la RCS, basándose en la longitud de las embarcaciones y en el tonelaje de las mismas

Tabla 2.1. RCS de buques

Target Ship			Median radar cross section of target vessel, m²								
Type	Overall length (m)	Cross tonnage	10	100	1.000	10.000	100.000	1.000.000	10.000.000	approx. min. RCS	approx. max. RCS
Inshore fishing vessel	9	5								3	10
Small coaster	40-46	200-250								20	800
Coaster	55	500								40	2.000
Coaster	55	500								300	4.000
Coaster	57	500								1.000	16.000
Large coaster	67	836-1.000								1.000	5.000
Collier	73	1.570								300	2.000
Warship (frigate)	103	2000*								5.000	100.000
Cargo liner	114	5.000								10.000	16.000
Cargo liner	137	8.000								4.000	16.000
Bulk carrier	167	8.200								400	10.000
Cargo	153	9.400								1.600	12.500
Cargo	166	10.430								400	16.000
Bulk carrier	198	15.000-20.000								1.000	32.000
Ore carrier	206	25.400								2.000	25.000
Container carrier	212	26436**								10.000	80.000
Medium tanker	213-229	30.000-35.000								5.000	80.000
Medium tanker	251	44.700								16.000	1.600.000

* Displacement
** Considerable deck cargo
S = stern on
Q = quarter
B = broadside
BW = bow
BWO = bow on
n = near

Fuente: (Williams, Cramp, & Curtis, 1978)

Por último, se puede concluir que para calcular el RCS de blancos complejos, como es el caso de un buque, es necesario conocer muy bien la geometría del mismo, tener sus dimensiones a detalle plasmadas en un plano; y usar un software CAD (AUTOCAD, ACADS, DEMACO, CIFER, etc.) como herramienta para el modelado de su geometría.

Por tal motivo, el software CAD se convierte en una herramienta muy útil y necesaria para el cálculo del RCS de un buque y con la finalidad de no desviar la investigación hacia el modelado de un buque en un software CAD, debido a su gran complejidad, se deja al lector la libertad de profundizar más en el tema y realizar una investigación mucho más detallada acerca del cálculo del RCS de objetivos complejos, presentes en un escenario de radar.

Capítulo 3
Cinemática del movimiento de un buque

3.1. Marcos de referencia

Una embarcación en el mar se mueve en seis grados de libertad (6DOF: *six degrees of freedom*). Por lo tanto, para definir su movimiento, necesitamos considerar tres coordenadas para definir las traslaciones y tres coordenadas para definir la orientación. Estas coordenadas son definidas utilizando dos tipos de marcos de referencia: marco de referencia inercial y marco de referencia fijo. (Perez & Fossen, 2007).

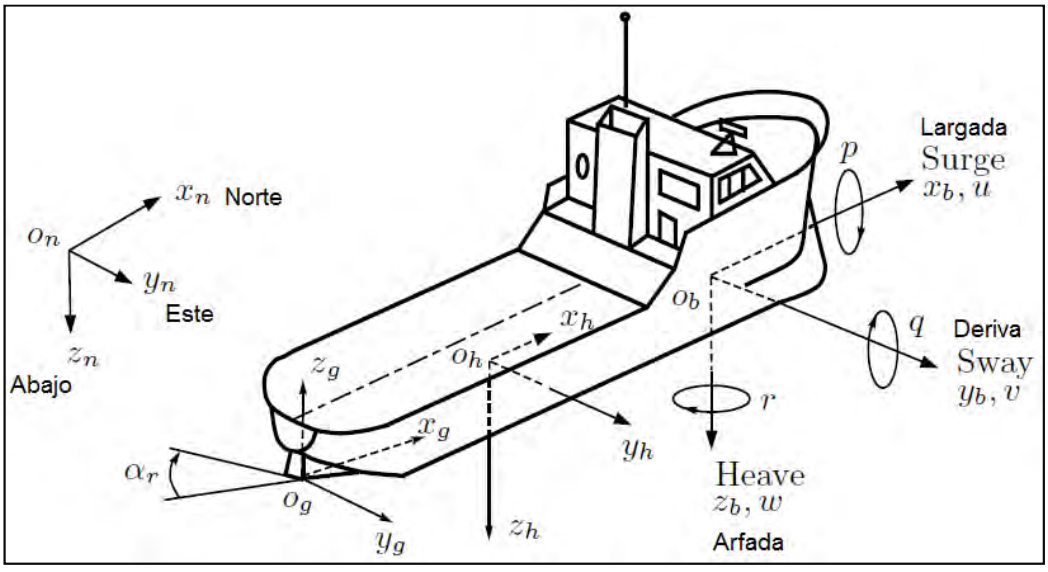


Figura 3.1. Notación y convención de signos para la descripción del movimiento de un buque.
Fuente: (Perez & Fossen, 2007, pág. 20)

Tabla 3.1. Notación en español que describe formas de movimiento de un buque

En Inglés	Traducción técnica	Traducción coloquial
Heave	Arfada	Levantar
Yaw	Guiñada	Desviarse
Pitch	Cabeceo	Declive
Roll	Balance	Balanceo
Sway	Deriva	Mecerse
Surge	Largada	Rompiente

Fuente: Elaboración propia.

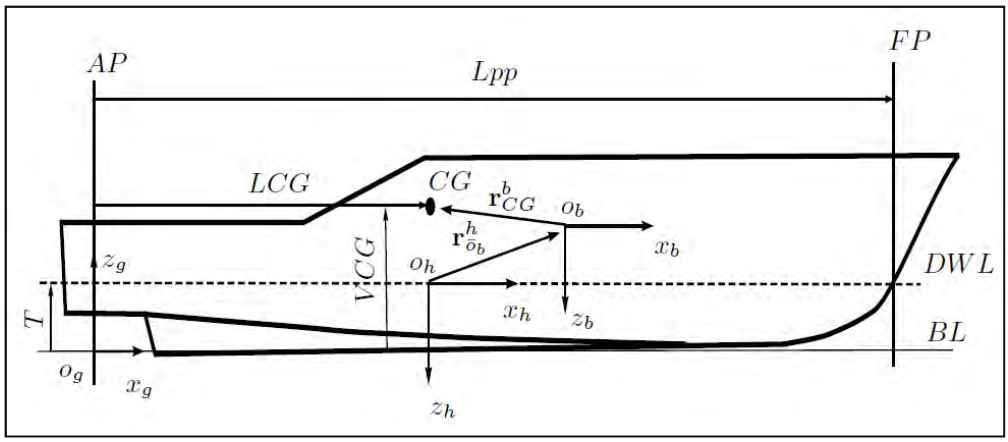


Figura 3.2. Principales ejes y marcos de referencia.
Fuente: (Perez & Fossen, 2007)

Tabla 3.2. Siglas de ejes principales y marcos de referencia.

Sigla	Nombre en Inglés	Traducción
0_g	*Geometric origin*	Origen geométrico
0_h	*Hydrodynamic origin*	Origen hidrodinámico
0_b	*Body fixed origin*	Origen fijado al cuerpo
CG	*Centre of gravity*	Centro de gravedad
LCG	*Lateral centre of gravity*	Centro de gravedad lateral
VCG	*Vertical centre of gravity*	Centro de gravedad vertical
AP	*Aft perpendicular*	Perpendicular de popa
FP	*Front perpendicular*	Perpendicular de frente
L_{pp}	*Length between perpendiculars*	Longitud entre perpendiculares
T	*Draught*	Calado
DWL	*Design waterline*	Diseño de flotación
BL	*Baseline*	Línea de base

Fuente: Elaboración propia.

A continuación se hará una descripción de los principales marcos de referencia que se utilizaran para describir la cinemática del movimiento de un buque.

3.1.1. Marco de referencia norte-este-abajo (marco de referencia "n")

El marco de referencia "n" (O_n, X_n, Y_n, Z_n) es fijado a tierra. El eje X_n positivo apunta hacia el norte, el eje Y_n positivo apunta hacia el este, y el eje Z_n positivo apunta hacia el centro de la tierra. El origen, O_n, está localizado en el centro de la superficie libre de agua en un lugar apropiado. Este marco de referencia es considerado inercial. Esta es una hipótesis razonable porque la velocidad de las embarcaciones es lo suficientemente pequeña para que las fuerzas debido a la rotación de la tierra sean insignificantes en comparación con las fuerzas hidrodinámicas que actúan sobre la embarcación.

3.1.2. Marco de referencia geométrico (marco de referencia "g")

El marco de referencia "g" (O_g, X_g, Y_g, Z_g) está fijado al casco de la embarcación. El eje X_g positivo apunta hacia la proa, el eje Y_g positivo apunta hacia estribor, y el eje Z_g positivo apunta hacia arriba. El origen, O_g , está localizado en la intersección de la línea base y la perpendicular de popa (ver figura 3.2).

3.1.3. Marco de referencia fijado al cuerpo (marco de referencia "b")

El marco de referencia "b" (O_b, X_b, Y_b, Z_b) está fijado al casco de la embarcación. El eje X_b positivo apunta hacia la proa, el eje Y_b positivo apunta hacia estribor, y el eje Z_b positivo apunta hacia abajo. Para naves marítimas, los ejes de este marco de referencia son elegidos para coincidir con los ejes principales de inercia; esto determina la posición del origen O_b de este marco de referencia.

3.1.4. Marco de referencia hidrodinámico (marco de referencia "h")

El marco de referencia "h" (O_h, X_h, Y_h, Z_h) no está fijado al casco de la embarcación; este marco de referencia se mueve a la velocidad promedio de la embarcación siguiendo su trayectoria. Los planos X_h, Y_h coinciden con la superficie media libre de agua. El eje X_h positivo apunta hacia delante y está alineado con el ángulo de desviación de baja frecuencia $\overline{\psi}$[1]. El eje Y_h positivo apunta hacia estribor, y el eje Z_h positivo apunta hacia abajo. El origen O_h está determinado tal que el eje Z_h pasa por el centro de gravedad. Este marco de referencia es considerado usualmente cuando la embarcación viaja a una velocidad promedio constante (que incluye el caso de velocidad cero); y por lo tanto, el movimiento oscilatorio inducido hace que la embarcación oscile con respecto al

[1] El ángulo $\overline{\psi}$ es obtenido al filtrar la onda del movimiento inducido de primer orden (movimiento oscilatorio) y manteniendo el movimiento de baja frecuencia, el que puede estar en equilibrio o variando muy suavemente. Por lo tanto, $\overline{\psi}$ es constante para una embarcación navegando en trayectoria recta.

marco de referencia “h”. Este marco de referencia es considerado inercial.

Cada uno de estos marcos de referencia explicados, tiene un uso específico. Por ejemplo, el marco de referencia “g” es usado por arquitectos navales para definir la geometría del casco de la embarcación, detalles principales, localización del centro de gravedad, localización del origen O_b. El marco de referencia “n” es utilizado para definir la posición de la embarcación y conjuntamente con el marco de referencia “b” definir la orientación de la embarcación.

Todas las mediciones hechas a bordo de la embarcación (velocidades, aceleraciones, etc.) son referidas al marco de referencia “b”, que es también utilizado para formular las ecuaciones del movimiento. El marco de referencia “h” es utilizado en hidrodinámica para calcular las fuerzas y el movimiento debido a la interacción del casco de la nave y las olas.

3.2. Notación de vector

Debido al uso de distintos marcos de referencia, es necesario establecer una notación matemática que nos permita identificar posición, velocidad y aceleración de diferentes puntos de interés en la embarcación y expresarlos en los distintos marcos de referencia considerados. Por lo tanto, para un punto genérico de interés “x” sobre la embarcación:

r_x^f denota la posición de “x” con respecto al marco de referencia “f”.

$$r_x^f = x_x^f f_x + y_x^f f_y + z_x^f f_z \equiv \begin{bmatrix} x_x^f \\ y_x^f \\ z_x^f \end{bmatrix} = \left[x_x^f, y_x^f, z_x^f\right]^T \qquad \textbf{(3.1)}$$

V_x^f denota la velocidad de “x” con respecto al marco de referencia “f”.

$\dot{V}_x^f$ denota la aceleración de “x” con respecto al marco de referencia “f”.

Θ_{ab} ángulos de Euler del marco de referencia “a” llevados al marco de referencia “b”. Las rotaciones son consideradas positivas cuando se hacen en sentido horario.

ω_{ab}^c denota la velocidad angular relativa del marco de referencia “b” respecto del marco de referencia “a”, descompuesta en el marco de referencia “c”.

El producto cruz de los vectores será escrito como:

$$a \times b \triangleq S(a)b \qquad \textbf{(3.2)}$$

Donde la matriz diagonal simétrica s está definida como:

$$S(\lambda) = -S^T(\lambda) \triangleq \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix} \qquad \textbf{(3.3)}$$

$$\lambda \triangleq \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

3.3. Coordenadas usadas para describir el movimiento de un buque

3.3.1. Maniobra y navegación

El estudio de la dinámica de un buque tradicionalmente se divide en dos áreas:

- Maniobra
- Navegación

La maniobra trata acerca del movimiento del buque en ausencia de ondas de excitación (agua en calma). El movimiento resulta de la acción de los dispositivos de control. La maniobra está asociada con los cambios de curso, paradas, etc. Por otro lado, la navegación está asociada con el movimiento debido a las ondas de excitación mientras la embarcación mantiene su curso y su velocidad constante.

3.3.2. Coordenadas de maniobra y marcos de referencia

La posición norte – este – debajo (n, e, d) de una embarcación está definida por las coordenadas del origen del marco de referencia "b" relativo al marco de referencia "n":

$$r_{o_b}^n \triangleq \begin{bmatrix} n \\ e \\ d \end{bmatrix}$$

La posición de una embarcación está definida por la orientación del marco de referencia "b" relativo al marco de referencia "n". Esta se da por tres rotaciones consecutivas:

1. Rotación alrededor del eje z_n con un ángulo de rotación llamado Yaw, ψ (ángulo de guiñada o ángulo de desviación).

2. Rotación alrededor del eje y_n con un ángulo de rotación llamado Pitch, θ (ángulo de cabeceo o ángulo de declive).

3. Rotación alrededor del eje x_n con un ángulo de rotación llamado Roll, Ø (ángulo de balanceo o ángulo de balance).

Con las rotaciones desarrolladas en este orden particular, los ángulos de rotación son llamados ángulos de Euler. El vector de ángulos de Euler está definido como:

$$\Theta_{nb} \triangleq \begin{bmatrix} \emptyset \\ \theta \\ \psi \end{bmatrix} \quad \textbf{(3.4)}$$

Siguiendo la notación de Fossen, el vector de posición – orientación (o vector generalizado de posición) está definido como:

$$\eta \triangleq \begin{bmatrix} r_{o_b}^n \\ \Theta_{nb} \end{bmatrix} = [n, e, d, \emptyset, \theta, \psi]^T \quad \textbf{(3.5)}$$

Las velocidades lineal y angular de la embarcación están convenientemente expresadas en el marco de referencia "b". El vector de velocidad lineal – angular (o simplemente vector generalizado de velocidad) dado en el marco de referencia "b" está definido como:

$$\nu \triangleq \begin{bmatrix} V_{o_b}^b \\ \omega_{nb}^b \end{bmatrix} = [u, v, w, p, q, r]^T \quad \textbf{(3.6)}$$

Donde:

$V_{o_b}^b = [u, v, w]^T$ es la velocidad lineal del punto o_b expresada en el marco de referencia "b".

$\omega_{nb}^b = [p, q, r]^T$ es la velocidad angular del marco de referencia "b" con respecto al marco de referencia "n" expresada en el marco de referencia "b".

3.3.3. Coordenadas de navegación y marcos de referencia

En navegación, el estudio del movimiento de la embarcación es desarrollado bajo la hipótesis de que la embarcación se está moviendo con curso estable y una velocidad promedio de avance constante (lo que incluye el caso de velocidad cero). Esto define un estado de equilibrio del movimiento y las olas hacen que la embarcación oscile con respecto a este estado de equilibrio (Movimiento de primer orden inducido por ondas).

En la teoría de navegación, el movimiento de la embarcación es descrito usando el marco de referencia "h" el que es fijado al equilibrio del movimiento. Aquí, sin embargo, también permitiremos el movimiento de la embarcación para maniobrar, pero bajo la hipótesis que la maniobra es mucho más lenta que el movimiento inducido por las olas. De este modo, el marco de referencia "h" puede ser usado para describir el movimiento.

Tabla 3.3. Nomenclatura adoptada para describir el movimiento de un buque y marcos de referencia en los que las componentes están definidas.

Componente	Nombre	Marco de referencia
n	Posición norte	Marco "n"
e	Posición este	Marco "n"
d	Posición abajo	Marco "n"
$\emptyset$	Ángulo de balance	Ángulo de Euler
θ	Ángulo de cabeceo	Ángulo de Euler
ψ	Ángulo de guiñada	Ángulo de Euler
u	Velocidad de largada	Marco "b"
v	Velocidad de deriva	Marco "b"
w	Velocidad de arfada	Marco "b"
p	Tasa de balance	Marco "b"
q	Tasa de cabeceo	Marco "b"
r	Tasa de guiñada	Marco "b"

Fuente: Elaboración propia.

Una vez que el origen o_h es elegido, este coincide con la ubicación de un punto "s" que varía muy lentamente en la embarcación, por ejemplo $o_h \equiv \bar{s}$, donde la notación $\bar{s}$ quiere decir equilibrio o componente que varía muy lentamente. Por lo tanto, $o_h \equiv \bar{s} = s$ cuando no hay olas (ver figura 3.3). Las coordenadas de navegación definidas en el marco de referencia "h" serán denotadas por:

$$\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6]^T$$

Cuando estas coordenadas describen la posición de s (ver figura 3.3), las coordenadas lineales son normalmente referidas como se muestra:

- $\xi_1 \rightarrow desplazamiento\ de\ largada$
- $\xi_2 \rightarrow desplazamiento\ de\ deriva$
- $\xi_3 \rightarrow desplazamiento\ de\ arfada$

Donde las coordenadas angulares ξ_4, ξ_5, ξ_6 son los ángulos de Euler:

$$\Theta_{hb} \triangleq \begin{bmatrix} \xi_4 \\ \xi_5 \\ \xi_6 \end{bmatrix} = \begin{bmatrix} \emptyset \\ \theta \\ \psi - \bar{\psi} \end{bmatrix} \quad \textbf{(3.8)}$$

Donde:

- $\xi_4 \rightarrow$ *ángulo de perturbación de balance*
- $\xi_5 \rightarrow$ *ángulo de perturbación de cabeceo*
- $\xi_6 \rightarrow$ *ángulo de perturbación de guiñada*

Las coordenadas de perturbación pueden ser usadas para describir la posición oscilatoria de cualquier punto de interés con respecto al marco de referencia "h". Efectivamente, para el punto genérico de interés "x" con equilibrio en la posición $\bar{x}$ con respecto al o_h se cumplen las siguientes relaciones:

$$\begin{aligned} r_x^h &= [\xi_1, \xi_2, \xi_3]^T + [\xi_4, \xi_5, \xi_6]^T \times r_{\bar{x}}^h \\ V_x^h &= \left[\dot{\xi}_1, \dot{\xi}_2, \dot{\xi}_3\right]^T + \left[\dot{\xi}_4, \dot{\xi}_5, \dot{\xi}_6\right]^T \times r_{\bar{x}}^h \\ \dot{V}_x^h &= \left[\ddot{\xi}_1, \ddot{\xi}_2, \ddot{\xi}_3\right]^T + \left[\ddot{\xi}_4, \ddot{\xi}_5, \ddot{\xi}_6\right]^T \times r_{\bar{x}}^h \end{aligned} \quad \textbf{(3.9)}$$

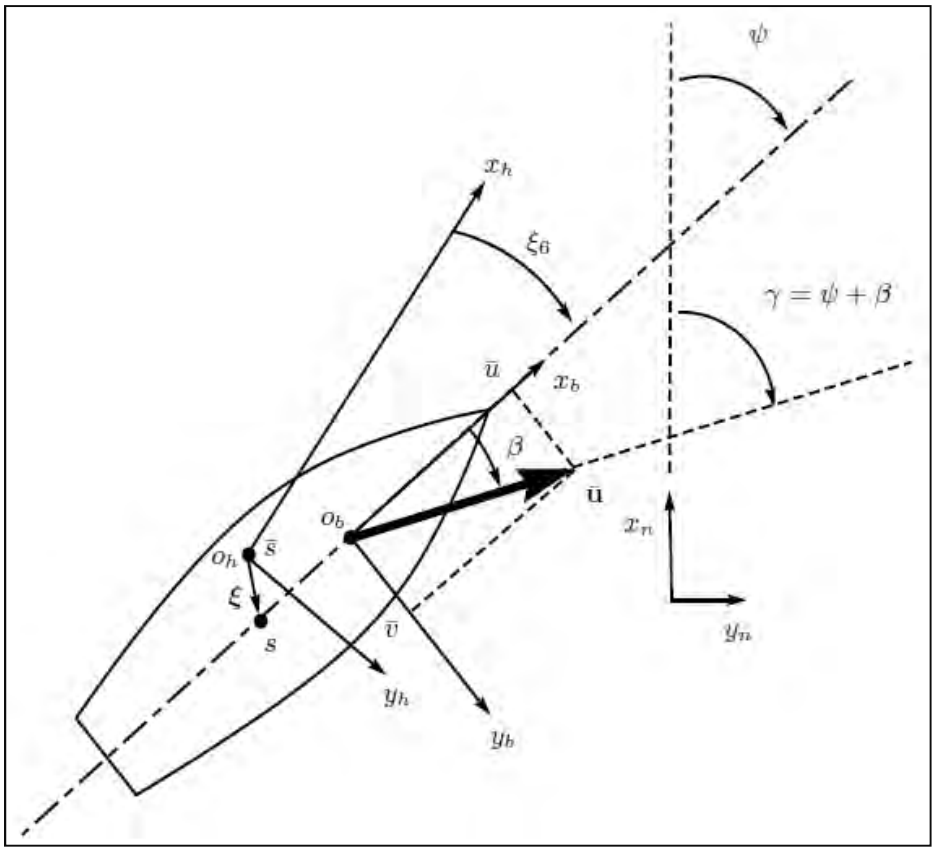


Figura 3.3. Ángulos para el plano horizontal.
Fuente: (Perez & Fossen, 2007)

3.3.4. Ángulos alrededor del eje Z

Definamos el vector velocidad de la embarcación total (en el marco de referencia "b") como:

$$\overline{U} = [\bar{u}, \bar{v}, \overline{w}]^T \quad \textbf{(3.10)}$$

Tal que:

$$u = \bar{u} + \delta u$$
$$v = \bar{v} + \delta v$$
$$w = \overline{w} + \delta w$$

Para embarcaciones de superficie (no submarinas) $\overline{w} = 0$ y para una maniobra lenta las velocidades de largada (*surge*) y deriva (*sway*) $\bar{u}$ y $\bar{v}$ son aproximadamente las mismas tanto en el marco de referencia "b" como en el marco de referencia "h". En este caso la velocidad de largada (*surge*) es denotada:

$$U \equiv \bar{u}^h \approx \bar{u}$$

Por otro lado, para los ángulos alrededor del eje Z de la superficie de las embarcaciones, es conveniente distinguir los siguientes:

- **Ángulo de guiñada (*Heading* o *yaw angle*) $\boldsymbol{\psi}$**, esta es la primera rotación de la secuencia de rotaciones (ángulos de Euler) que realiza el marco de referencia "n" dentro del marco de referencia "b".

- **Ángulo de navegación (*Yaw pertubation angle*) $\boldsymbol{\xi_6}$**, esta es la primera rotación de la secuencia de rotaciones (ángulos de Euler) que realiza el marco de referencia "h" dentro del marco de referencia "b".

- **Ángulo de deriva (*Drift angle*) $\boldsymbol{\beta}$,** este es el ángulo entre el eje positivo "x" del marco de referencia "b" y el vector de velocidad promedio de la embarcación $\overline{U}$ donde β se puede definir como:

$$\beta = arctan\left(\frac{\bar{v}}{\bar{u}}\right) \quad \textbf{(3.11)}$$

  Previniendo que $\bar{u}$ no sea cero.

- **Ángulo de curso (*Course angle*) $\boldsymbol{\gamma}$,** este es el ángulo entre el eje positivo "x" del marco de referencia "n" y el vector velocidad promedio de la embarcación $\overline{U}$.

3.4. Transformaciones de velocidad

3.4.1. Matrices de rotación

La transformación de coordenadas vectoriales entre diferentes marcos de referencia es desarrollada por matrices de transformación. El vector

genérico **"r"** puede ser expresado tanto en el marco de referencia "a" como en el marco de referencia "b" como se muestra a continuación:

$$r = \sum_{i=1}^{3} r_i^a a_i \qquad y \qquad r = \sum_{i=1}^{3} r_i^b b_i \tag{3.12}$$

Donde los vectores $a_i \; y \; b_i$ son los vectores unitarios a lo largo de los ejes de los marcos de referencia "a" y "b" respectivamente, y $r_i^a = r.a_i$ y $r_i^b = r.b_i$. Entonces,

$$r_i^a = r.a_i = \left(\sum_{i=1}^{3} r_i^b b_i\right).a_i = \sum_{i=1}^{3} r_i^b (a_i.b_i) \tag{3.13}$$

Esto da paso a la notación R_b^a para la matriz de transformación con entradas $\{a_i.b_i\}$, la que lleva los vectores expresados en el marco de referencia "b" al marco de referencia "a":

$$r^a = R_b^a r^b \tag{3.14}$$

Esta matriz es llamada **la matriz de rotación de "b" a "a".**

Las matrices de rotación son elementos en SO(3), el grupo especial ortogonal de orden 3:

$$SO(3) = \{R/R \in \mathbb{R}^{3\times3}, RR^T = I_{3\times3}, det(R) = 1\} \tag{3.15}$$

Por lo tanto:

$$(R_b^a)^{-1} = (R_b^a)^T = R_a^b$$

3.4.2. Transformación cinemática entre el marco de referencia "b" y el marco de referencia "n"

La transformación entre las velocidades lineales del marco de referencia "b" y la derivada en el tiempo de las posiciones en el marco de referencia "n" pueden ser expresadas como:

$$\begin{bmatrix} \dot{n} \\ \dot{e} \\ \dot{d} \end{bmatrix} = R_b^n(\Theta_{nb}) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.16}$$

Donde la matriz de transformación de velocidades lineales $R_b^n(\Theta_{nb})$ está dada por:

$$R_b^n(\Theta_{nb}) = \begin{bmatrix} c\psi c\theta & -s\psi c\emptyset + c\psi s\theta s\emptyset & s\psi s\emptyset + c\psi c\emptyset s\theta \\ s\psi c\theta & c\psi c\emptyset + s\emptyset s\theta s\psi & -c\psi s\emptyset + s\psi c\emptyset s\theta \\ -s\theta & c\theta s\emptyset & c\theta c\emptyset \end{bmatrix} \tag{3.17}$$

Donde $s \equiv \sin(.)$ y $c \equiv \cos(.)$ y

$$R_n^b(\Theta_{nb}) = R_b^n(\Theta_{nb})^{-1} = R_b^n(\Theta_{nb})^T \tag{3.18}$$

Esta transformación es el resultado de tres rotaciones consecutivas alrededor de los ejes principales:

$$R_b^n(\Theta_{nb}) \triangleq R_{z,\psi} R_{y,\theta} R_{x,\emptyset} \tag{3.19}$$

Donde cada uno de los términos son los siguientes:

$$R_{x,\emptyset} \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\emptyset & -\sin\emptyset \\ 0 & \sin\emptyset & \cos\emptyset \end{bmatrix}$$

$$R_{y,\theta} \triangleq \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.20}$$

$$R_{z,\psi} \triangleq \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La transformación entre la velocidad angular del marco de referencia "b" ω_{nb}^b y la derivada respecto al tiempo de los ángulos de Euler $\dot{\Theta}_{nb}$ puede ser expresado como:

$$\dot{\Theta}_{nb} = T_\Theta(\Theta_{nb})\omega_{nb}^b$$

$$\begin{bmatrix} \dot{\emptyset} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T_\Theta(\Theta_{nb}) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.21}$$

Donde la matriz de transformación de velocidad angular $T_\Theta(\Theta_{nb})$ y su inversa están dadas como se muestra a continuación:

$$T_\Theta(\Theta_{nb}) \triangleq \begin{bmatrix} 1 & s\emptyset t\theta & c\emptyset t\theta \\ 0 & c\emptyset & -s\emptyset \\ 0 & s\emptyset/c\theta & c\emptyset/c\theta \end{bmatrix} \tag{3.22}$$

Con $t \equiv \tan(.)\ y\ c\theta \neq 0$

$$T_\Theta(\Theta_{nb})^{-1} \triangleq \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\emptyset & c\theta s\emptyset \\ 0 & -s\emptyset & c\emptyset c\theta \end{bmatrix}$$

Por otro lado tenemos también la siguiente relación:

$$\omega_{nb}^b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\emptyset} \\ 0 \\ 0 \end{bmatrix} + R_{x,\emptyset}^T \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_{x,\emptyset}^T R_{y,\theta}^T \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \triangleq T_\Theta(\Theta_{nb})^{-1}\dot{\Theta} \tag{3.23}$$

Nótese que $T_\Theta(\Theta_{nb})^{-1} \neq T_\Theta(\Theta_{nb})^T$ y también que $T_\Theta(\Theta_{nb})$ no está definido para $\theta = \pm\frac{\pi}{2}$ esto no es un problema para embarcaciones de

superficie, es decir para embarcaciones que se desplazan sobre la superficie del agua y no debajo de ella (submarinos). Por lo tanto podemos definir la siguiente transformación cinemática para las coordenadas de maniobra:

$$\dot{\eta} = J_b^n(\Theta_{nb})\nu = \begin{bmatrix} R_b^n(\Theta_{nb}) & 0_{3\times3} \\ 0_{3\times3} & T_\Theta(\Theta_{nb}) \end{bmatrix}\nu \qquad \textbf{(3.24)}$$

Para transformar las aceleraciones necesitamos considerar las derivadas con respecto del tiempo de (3.16) y (3.21):

$$[\dot{u},\dot{v},\dot{w}]^T = R_n^b(\Theta_{nb})\left[\ddot{n},\ddot{e},\ddot{d}\right]^T + \dot{R}_n^b(\Theta_{nb})\left[\dot{n},\dot{e},\dot{d}\right]^T$$

$$[\dot{p},\dot{q},\dot{r}]^T = T_\Theta(\Theta)\left[\ddot{\phi},\ddot{\theta},\ddot{\psi}\right]^T + \dot{T}_\Theta(\Theta)\left[\dot{\phi},\dot{\theta},\dot{\psi}\right]^T \qquad \textbf{(3.25)}$$

3.4.3. Transformación cinemática entre el marco de referencia "b" y el marco de referencia "h"

La transformación para el vector velocidad generalizado ν (dado en el marco de referencia "b") al vector de velocidad generalizado $\dot{\xi} = \left[{V_{oh}^h}^T, {\omega_{hb}^h}^T\right]^T$ dado en el marco de referencia "h", es desarrollada en dos pasos: una rotación y una traslación.

Consideremos en primer lugar el caso de velocidad cero de avance. La velocidad lineal de $\bar{o}_h$ en el marco de referencia "b" está dado por:

$$V_{\bar{0}_h}^b = V_{o_b}^b + \omega_{nb}^b \times r_{\bar{o}_h}^b \qquad \textbf{(3.26)}$$

Donde $\bar{o}_h$ es la posición de equilibrio de o_h con respecto al marco de referencia "b" (recordar que el marco de referencia "h" no está fijo a la embarcación y por lo tanto el vector $r_{o_h}^b$ es variable del tiempo, así que consideramos $r_{\bar{o}_h}^b$). Ya que el marco de referencia "h" es considerado inercial y su velocidad angular relativa con respecto al marco de referencia "n" es aproximadamente cero ($\omega_{nh}^b \approx 0$), la siguiente relación sostiene:

$$\omega_{hb}^b = \omega_{nb}^b - \omega_{nh}^b \approx \omega_{nb}^b \qquad \textbf{(3.27)}$$

Expresando el producto vectorial de la relación (3.26) en términos de una matriz simétrica diagonal:

$$S\left(r_{\bar{o}_h}^b\right) = -S\left(r_{\bar{o}_h}^b\right)^T = \begin{bmatrix} 0 & -z_{\bar{o}_h}^b & y_{\bar{o}_h}^b \\ z_{\bar{o}_h}^b & 0 & -x_{\bar{o}_h}^b \\ -y_{\bar{o}_h}^b & x_{\bar{o}_h}^b & 0 \end{bmatrix} \qquad \textbf{(3.28)}$$

Las expresiones (3.26) y (3.27) pueden ser combinadas como:

$$\begin{bmatrix} V_{\bar{o}_h}^b \\ \omega_{hb}^b \end{bmatrix} = H\left(r_{\bar{o}_h}^b\right) \begin{bmatrix} V_{o_b}^b \\ \omega_{nb}^b \end{bmatrix} \quad \textbf{(3.29)}$$

$$H(\lambda) \triangleq \begin{bmatrix} I_{3\times3} & S(\lambda)^T \\ 0_{3\times3} & I_{3\times3} \end{bmatrix}, \lambda \in \mathbb{R}^3 \quad \textbf{(3.30)}$$

La transformación de la velocidad lineal entre el marco de referencia "b" y el marco de referencia "h" está dada por la siguiente rotación:

$$V_{\bar{o}_h}^h = R_b^h(\Theta_{hb}) V_{\bar{o}_h}^b \quad \textbf{(3.31)}$$

Donde $R_b^h(\Theta_{hb})$ tiene la forma de la expresión (3.17). En una forma similar la transformación para la velocidad angular está dada por:

$$\omega_{hb}^h = T_\Theta(\Theta_{hb}) \omega_{hb}^b \quad \textbf{(3.32)}$$

Donde $T_\Theta(\Theta_{hb})$ tiene la forma de la expresión (3.22). Combinando (3.31), (3.32) y (3.29) obtenemos:

$$\begin{bmatrix} V_{\bar{o}_h}^h \\ \omega_{hb}^h \end{bmatrix} = \begin{bmatrix} R_b^h(\Theta_{hb}) & 0_{3\times3} \\ 0_{3\times3} & T_\Theta(\Theta_{hb}) \end{bmatrix} H\left(r_{\bar{o}_h}^b\right) \begin{bmatrix} V_{o_b}^b \\ \omega_{nb}^b \end{bmatrix} \quad \textbf{(3.33)}$$

Por lo tanto la transformación buscada es obtenida combinando las expresiones anteriores con $\dot{\Theta}_{hb} = T_\Theta(\Theta_{hb}) \omega_{nb}^b$:

$$\dot{\xi} = J_b^h\left(\Theta_{hb}, r_{\bar{o}_h}^b\right) \nu \quad \textbf{(3.34)}$$

$$J_b^h\left(\Theta_{hb}, r_{\bar{o}_h}^b\right) \triangleq \begin{bmatrix} R_b^h(\Theta_{hb}) & R_b^h(\Theta_{hb}) S\left(r_{\bar{o}_h}^b\right)^T \\ 0_{3\times3} & T_\Theta(\Theta_{hb}) \end{bmatrix} \quad \textbf{(3.35)}$$

Para el caso de velocidad de avance, separamos la velocidad del marco de referencia "b" en una componente que varía lentamente y una componente oscilatoria inducida por el movimiento de las olas.

$$\nu = \bar{\nu} + \delta\nu \quad \textbf{(3.36)}$$

Con:

$$\bar{\nu} = [U, 0,0,0,0,0]^T \quad \textbf{(3.37)}$$

$$\delta\nu = [\delta u, \delta v, \delta w, \delta p, \delta q, \delta r]^T = [\delta u, v, w, p, q, r]^T$$

Entonces la transformación cinemática entre el marco de referencia "b" y el marco de referencia "h" para el caso de velocidad de avance está dada por:

$$\dot{\xi} = J_b^h\left(\Theta_{hb}, r_{\bar{o}_h}^b\right)(\nu - \bar{\nu}) \quad \textbf{(3.38)}$$

Con $J_b^h\left(\Theta_{hb}, r_{\bar{o}_h}^b\right)$ dado en (3.35). Para ángulos pequeños, por ejemplo $\Theta_{hb} \approx 0$ se pueden utilizar las siguientes aproximaciones:

$$\begin{aligned} &R_b^h(\Theta_{hb}) \approx I_{3\times3} \\ &T_\Theta(\Theta_{hb}) \approx I_{3\times3} \\ &J_b^h\left(\Theta_{hb}, r_{\bar{o}_h}^b\right) \approx H\left(r_{\bar{o}_h}^b\right) \end{aligned} \quad \textbf{(3.39)}$$

Si expandimos esta transformación cinemática, considerando ángulos pequeños, y asumimos una embarcación esbelta (así que los grados de libertad 1, 3, 5 pueden ser desacoplados de los grados de libertad 2, 4, 6) y mantener solo los términos lineales, obtenemos:

$$\dot{\xi}_1 \approx \delta u + z_{oh}^b \delta q \quad \textbf{(3.40)}$$

$$\dot{\xi}_2 \approx \delta v + x_{oh}^b \delta r - z_{oh}^b \delta p + U\delta\psi \quad \textbf{(3.41)}$$

$$\dot{\xi}_3 \approx \delta w - x_{oh}^b \delta q - U\delta\theta \quad \textbf{(3.42)}$$

$$\dot{\xi}_4 = \delta p \quad \textbf{(3.43)}$$

$$\dot{\xi}_5 = \delta q \quad \textbf{(3.44)}$$

$$\dot{\xi}_6 = \delta r \quad \textbf{(3.45)}$$

Derivamos las expresiones anteriores con respecto al tiempo y obtenemos:

$$\ddot{\xi}_1 = \dot{\delta} u + z_{oh}^b \dot{\delta} q \quad \textbf{(3.46)}$$

$$\ddot{\xi}_2 = \dot{\delta} v + x_{oh}^b \dot{\delta} r - z_{oh}^b \dot{\delta} p + U\delta r \quad \textbf{(3.47)}$$

$$\ddot{\xi}_3 = \dot{\delta} w - x_{oh}^b \dot{\delta} q - U\delta q \quad \textbf{(3.48)}$$

$$\ddot{\xi}_4 = \dot{\delta} p \quad \textbf{(3.49)}$$

$$\ddot{\xi}_5 = \dot{\delta} q \quad \textbf{(3.50)}$$

$$\ddot{\xi}_6 = \dot{\delta} r \quad \textbf{(3.51)}$$

Si asumimos movimientos sinusoidales, se cumple que:

$$\delta\psi = \sin\omega_e t$$

$$r = \omega_e \cos\omega_e t$$

$$\dot{r} = -\omega_e^2 \sin\omega_e t = -\omega_e^2 \delta\psi \quad \textbf{(3.52)}$$

Lo que puede ser usado para expresar:

$$\delta\psi = -\frac{1}{\omega_e^2}\dot{r}\,, \delta\theta = -\frac{1}{\omega_e^2}\dot{q} \tag{3.53}$$

Sustituyendo (3.53) en las expresiones (3.40) a (3.45):

$$\dot{\xi}_1 \approx \delta u + z_{oh}^b \delta q \tag{3.54}$$

$$\dot{\xi}_2 \approx \delta v + x_{oh}^b \delta r - z_{oh}^b \delta p - U\frac{1}{\omega_e^2}\dot{\delta} r \tag{3.55}$$

$$\dot{\xi}_3 \approx \delta\omega - x_{oh}^b \delta q + U\frac{1}{\omega_e^2}\dot{\delta} q \tag{3.56}$$

$$\dot{\xi}_4 = \delta p \tag{3.57}$$

$$\dot{\xi}_5 = \delta q \tag{3.58}$$

$$\dot{\xi}_6 = \delta r \tag{3.59}$$

Las expresiones desde (3.54) hasta (3.59) y desde (3.46) hasta (3.51) representan una aproximación lineal de (3.38) y puede ser escrita de una forma compacta como se muestra a continuación:

$$\dot{\xi} = J_b^h \delta\nu - \frac{U}{\omega_e^2} L\dot{\delta}\nu \tag{3.60}$$

$$\ddot{\xi} = J_b^h \dot{\delta}\nu + UL\delta\nu \tag{3.61}$$

Donde:

$$J_b^h \triangleq J_b^h\left(0, r_{\bar{o}_h}^b\right) = H\left(r_{\bar{o}_h}^b\right) \tag{3.62}$$

Con $H\left(r_{\bar{o}_h}^b\right)$ dado en (3.30) y

$$L \triangleq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.63}$$

Capítulo 4
Plataforma de simulación en Matlab

En este capítulo se describirá el algoritmo elaborado con Matlab, versión R2013a, que nos permitirá simular un escenario de radar. Los objetivos que participan en la escena de radar son buques, que describen una determinada trayectoria especificada por el usuario. La cantidad de buques que participan en la escena de radar, la determina el usuario y queda limitada únicamente por la carga computacional que se genera al ejecutar la simulación; es decir, a mayor número de buques, más tiempo tomará la ejecución de la simulación; dependiendo del número de procesadores que posea el computador donde se ejecute la simulación.

Para la ejecución de la simulación, se diseñó una GUI en Matlab ("*Graphical User Interface*") con la finalidad de presentar al usuario las acciones disponibles del algoritmo de una manera sencilla y amigable.

Es oportuno señalar, que la simulación de las trayectorias de los buques que intervengan en la escena de radar, se hará usando las ecuaciones del modelo matemático de Fossen, que describe la cinemática del movimiento de un buque.

También, se presentarán los diagramas de flujo del algoritmo principal y sus funciones, con la finalidad de que esta plataforma de simulación pueda ser elaborada con otro lenguaje de programación.

Por último, se mostrará el código en Matlab del algoritmo que simula un escenario de radar acondicionado en una GUI para su ejecución.

4.1. Ecuaciones para el modelado de las trayectorias de los buques

En el capítulo anterior, se presentó un estudio de la cinemática de un buque, con seis grados de libertad (3D), usando el modelo matemático de Fossen.

En el presente trabajo de investigación, para simular la trayectoria de un buque en una escena de radar, se usarán las mismas ecuaciones del modelo matemático de

Fossen pero expresadas en dos dimensiones, debido a que el radar que estudiamos muestra los contactos en un plano y no en el espacio. Teniendo en cuenta todo lo estudiado en el capítulo anterior, acerca de la cinemática de un buque, podemos expresar la matriz de transformación de velocidad R_b^n en dos dimensiones como se muestra a continuación:

$$R_b^n = \begin{bmatrix} cos\psi & -sen\psi \\ sen\psi & cos\psi \end{bmatrix} \quad \textbf{(4.1)}$$

Por otro lado, según (Perez & Fossen, 2007) la trayectoria de un buque está dada por:

$$r_{nb}^n(t) = \int_0^t R_b^n V_{nb}^b dt + r_{nb}^n(0) \quad \textbf{(4.2)}$$

Donde:

$r_{nb}^n(t)$, es la posición del buque que varía con el tiempo.

R_b^n , es la matriz de transformación de velocidad del marco de referencia "b" al marco de referencia "n", expresada en dos dimensiones.

V_{nb}^b , es el módulo del vector velocidad del buque en el marco de referencia "b".

$r_{nb}^n(0)$, es la posición del buque en el tiempo cero; es decir, el punto inicial desde donde el buque comenzará a moverse.

Además, para embarcaciones de superficie (no submarinas), como es el caso de un buque, la componente vertical "*Heave*" (w) del vector velocidad del buque es cero (w=0). El vector velocidad del buque, en el marco de referencia "b", es $\overline{U} = [u, v, w]^T$; por lo tanto, si $w = 0$ entonces el vector velocidad del buque queda expresado como sigue a continuación:

$$\overline{U} = [u, v]^T = V_{nb}^b \quad \textbf{(4.3)}$$

Reemplazando las expresiones (4.1) y (4.3) en (4.2) tenemos:

$$r_{nb}^n(t) = \int_0^t \begin{bmatrix} cos\psi & -sen\psi \\ sen\psi & cos\psi \end{bmatrix} [u, v]^T dt + r_{nb}^n(0) \quad \textbf{(4.4)}$$

De la expresión (4.4) obtenemos las componentes del vector $r_{nb}^n(t)$:

$$\begin{aligned} n(t) &= \int_0^t u \cdot \cos\psi \cdot dt - \int_0^t v \cdot \sin\psi \cdot dt + n(0) \\ e(t) &= \int_0^t u \cdot \sin\psi \cdot dt + \int_0^t v \cdot \cos\psi \cdot dt + e(0) \end{aligned} \quad \textbf{(4.5)}$$

Resolviendo las integrales con respecto al tiempo de las expresiones (4.5):

$$n(t) = u \cdot t \cdot \cos\psi - v \cdot t \cdot \sin\psi + n(0) \quad \textbf{(4.6)}$$

$$e(t) = u \cdot t \cdot \sin\psi + v \cdot t \cdot \cos\psi + e(0) \quad \textbf{(4.7)}$$

Donde:

$n(t)$, representa la coordenada X_n de posición del buque, con respecto al marco de referencia "n".

$e(t)$, representa la coordenada Y_n de posición del buque, con respecto al marco de referencia "n".

u, es la velocidad de largada (velocidad de "*surge*") del buque.

v, es la velocidad de deriva (velocidad de "*sway*") del buque.

ψ, es el ángulo de orientación del buque ("*heading*").

t, es el tiempo, en segundos, en el instante en que el buque alcanza la posición $\big(n(t), e(t)\big)$, siguiendo una determinada trayectoria.

$n(0)$, es la coordenada de la posición inicial del buque medida sobre el eje X_n.

$e(0)$, es la coordenada de la posición inicial del buque medida sobre el eje Y_n.

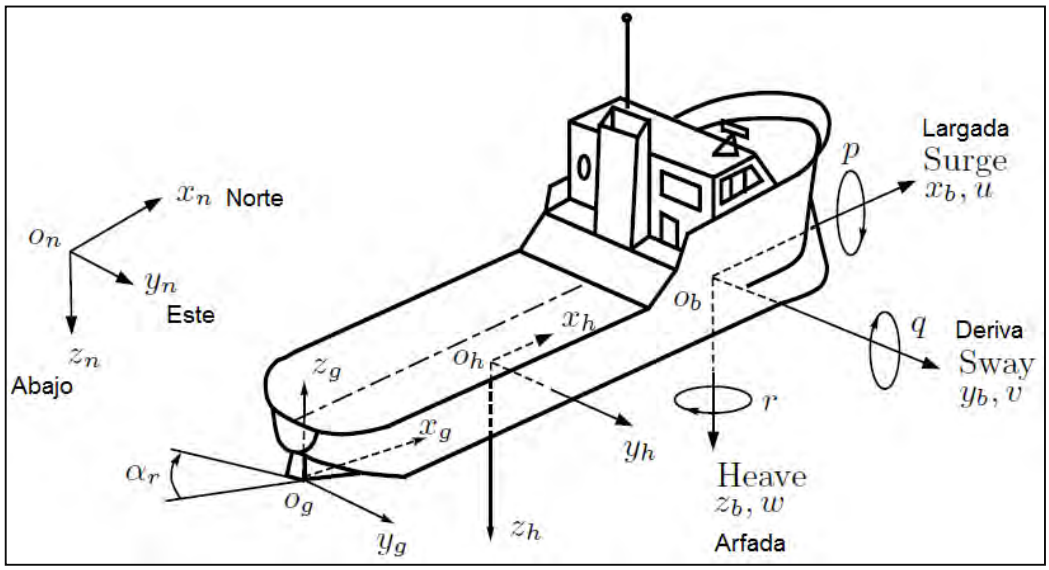


Figura 4.1. Notación y convención de signos para la descripción del movimiento de un buque.
Fuente: (Perez & Fossen, 2007, pág. 20)

Trabajamos y simplificamos las expresiones (4.6) y (4.7) con la finalidad de obtener una ecuación que contenga de manera explícita el ángulo de orientación del buque (ψ).

$$n(t) - n(0) = u \cdot t \cdot \cos\psi - v \cdot t \cdot \sin\psi \quad \textbf{(4.8)}$$

$$e(t) - e(0) = u \cdot t \cdot \sin\psi + v \cdot t \cdot \cos\psi \quad \textbf{(4.9)}$$

Dividiendo entre "t" cada una de las expresiones anteriores:

$$\frac{n(t) - n(0)}{t} = u \cdot \cos\psi - v \cdot \sin\psi \quad (\mathbf{4.10})$$

$$\frac{e(t) - e(0)}{t} = u \cdot \sin\psi + v \cdot \cos\psi \quad (\mathbf{4.11})$$

Dividiendo entre $\cos\psi$ las expresiones (4.10) y (4.11):

$$\frac{n(t) - n(0)}{t \cdot \cos\psi} = u - v \cdot \tan\psi \quad (\mathbf{4.12})$$

$$\frac{e(t) - e(0)}{t \cdot \cos\psi} = u \cdot \tan\psi + v \quad (\mathbf{4.13})$$

Dividiendo (4.12) entre (4.13):

$$\frac{n(t) - n(0)}{e(t) - e(0)} = \frac{u - v \cdot \tan\psi}{u \cdot \tan\psi + v} \quad (\mathbf{4.14})$$

Al primer miembro de la ecuación (4.14) le llamamos "k":

$$k = \frac{n(t) - n(0)}{e(t) - e(0)}$$

Entonces la expresión (4.14) queda de la siguiente manera:

$$k = \frac{u - v \cdot \tan\psi}{u \cdot \tan\psi + v} \quad (\mathbf{4.15})$$

Trabajamos la expresión anterior:

$$k \cdot (u \cdot \tan\psi + v) = u - v \cdot \tan\psi$$

$$k.u \cdot \tan\psi + k \cdot v = u - v \cdot \tan\psi$$

$$k.u \cdot \tan\psi + v \cdot \tan\psi = u - k \cdot v$$

$$\tan\psi\,(k \cdot u + v) = u - k \cdot v$$

$$\tan\psi = \frac{u - k \cdot v}{v + k \cdot u}$$

$$\psi = \tan^{-1}\left(\frac{u - k \cdot v}{v + k \cdot u}\right) \qquad (\mathbf{4.16})$$

La ecuación (4.16) es usada en el algoritmo para calcular la orientación del buque en cada instante de tiempo, mientras el buque recorre la trayectoria especificada por el usuario. Además, el ψ depende de las componentes del vector velocidad del buque, $\overline{U}$, y de la variable "k" definida como:

$$k = \frac{n(t) - n(0)}{e(t) - e(0)}$$

Por otro lado, trabajando con las ecuaciones (4.10) y (4.11) hallaremos una expresión para calcular el tiempo "t" que emplea el buque en recorrer, a cada instante, las distintas posiciones de acuerdo a la trayectoria especificada por el usuario. Para tal efecto, sumamos las ecuaciones (4.10) y (4.11) y acomodando los términos obtenemos la expresión (4.17) que se usará en el algoritmo para calcular "t":

$$\frac{n(t) - n(0) + e(t) - e(0)}{t} = u \cdot \cos\psi - v \cdot \sin\psi + u \cdot \sin\psi + v \cdot \cos\psi$$

$$\frac{n(t) - n(0) + e(t) - e(0)}{t} = u \cdot (\cos\psi + \sin\psi) + v \cdot (\cos\psi - \sin\psi)$$

$$t = \frac{n(t) - n(0) + e(t) - e(0)}{u \cdot (\cos\psi + \sin\psi) + v \cdot (\cos\psi - \sin\psi)} \qquad (\mathbf{4.17})$$

4.2. Diagramas de flujo del algoritmo principal y funciones utilizadas para la simulación de las escenas de radar

El algoritmo que simula un escenario de radar se acondicionó en una GUI para su ejecución. Éste hace uso de diez funciones que se listan a continuación:

1. PPI_plot_ONE_TARGET
2. PPI_plot_MULTIPLE_TARGETS
3. Mostrar_PPI
4. Mostrar_Menu
5. Mostrar_Menu_para_Elegir_la_Trayectoria_del_Buque
6. Mostrar_mensaje_al_usuario
7. Generador_de_posiciones_para_buques
8. Generar_data_para_simulacion

9. Cargar_Data_Sobre_Imagen_Real
10. Gausianos

Los diagramas de flujo del algoritmo principal y de las diez funciones utilizadas se muestran en el anexo A.

4.3. Código en Matlab del algoritmo principal y funciones, agrupados en una GUI (*Graphical User Interface*) para su ejecución

En el anexo B se muestra el código elaborado en Matlab versión R2013a para la simulación de un escenario de radar.
Los resultados de la simulación pueden presentarse al usuario de dos maneras:

1. Sobre un PPI ("*Plan Position Indicator*")
2. Sobre una imagen real de radar

Queda a elección del usuario la manera que desea ver los resultados de la simulación.

Por otro lado, cabe señalar que para elaborar las funciones *PPI_plot_ONE_TARGET* y *PPI_plot_MULTIPLE_TARGETS*, que muestran los resultados de la simulación sobre un PPI, tomé como referencia el algoritmo desarrollado por (Roopchand, 2013) y que se encuentra publicado en la *website* de *Matlab Central.*

4.4. Cómo usar el algoritmo

En el anexo C se da una breve explicación al usuario de cómo usar el algoritmo que simula un escenario de radar.

Conclusiones

El presente trabajo de investigación tiene la característica de un proyecto piloto, se comenzó con el diseño de un algoritmo que simule un escenario de radar en el que participan buques moviéndose a velocidad constante; en este punto, se da vía libre a otros investigadores a mejorar el algoritmo, simulando un escenario de radar donde participen buques que presenten aceleración.

Por otro lado, el algoritmo también se puede mejorar añadiendo a la escena de radar otro tipo de objetivos como helicópteros, aviones u otro tipo de móviles que podrían participar en la escena de radar, moviéndose con velocidad constante y/o aceleración.

En el presente trabajo de investigación, se diseñó un algoritmo que simula un escenario de radar de superficie, por lo que los resultados de la simulación se muestran sobre un plano, ya sea un PPI o sobre una imagen real; en este punto, se da vía libre a otros investigadores que deseen desarrollar algoritmos que simulen escenarios de radar de otro tipo de tecnología como radares tridimensionales, radares de apertura sintética (SAR), radares de apertura sintética inversa (ISAR), etc.

Por último, en cuanto al estudio que se hizo en el capítulo dos, para el cálculo de la RCS de un objetivo complejo, como es el caso de un buque, se concluye que es necesario conocer muy bien la geometría del mismo, tener sus dimensiones a detalle plasmadas en un plano; y usar un software CAD (AUTOCAD, ACADS, DEMACO, CIFER, etc.) como herramienta para el modelado de su geometría. Hacerlo de la forma tradicional usando alguno de los métodos expuestos, ocasionaría emplear un gran esfuerzo en tratar de resolver ecuaciones muy complejas; por tal motivo, el uso de un software CAD se convierte en una herramienta muy útil y necesaria para el cálculo del RCS de un buque y con la finalidad de no desviar la investigación hacia el modelado de un buque en un software CAD, debido a su gran complejidad, se da vía libre a otros investigadores a profundizar más en el tema y realizar un estudio mucho más detallado acerca del cálculo del RCS de objetivos complejos, presentes en un escenario de radar.

Referencias bibliográficas

Chatzigiorgiadis, F. (Septiembre de 2004). Scatterings regions. *Development of code for a physical optics radar cross section prediction and analysis application*. Monterey, California.

Chávez Ferry, N., & Cabrera, M. A. (2012). Fundamentos Físicos. Determinación de las Coordenadas Espaciales y la Velocidad Radial del Blanco.Tipo de Radares. En N. Chávez Ferry, & M. A. Cabrera, *Fundamentos de Radar.* Habana, Cuba y Argentina: Instituto Superior Politécnico, J.A. Echevarría y Universidad Nacional de Tucumán.

Cosenza, M. (2012). Teorema de Green. *Electromagnetismo*. Mérida, Mérida, Venezuela: Centro de Física Fundamental.

Díaz, M. A. (2005). *Modelado de un radar doppler de pulsos (PDR)*. Tijuana, México.

Harre, I. (2004). RCS of Simple Target Objects. *RCS in Radar Range Calculations for Maritime Targets*. Bremen, Germany.

Jarabo Amores, M. P. (2009). Introducción a los Sistemas de Radar. *Tratamiento de la Señal de Radar*. España: Universidad de Alcalá.

Kingsley, S., & Quegan, S. (1992). Noise. En *Understanding Radar Systems.* England: McGraw Hill.

Knott, E. F., Tuley, M. T., & Shaeffer, J. F. (2004). *Radar Cross Section.* SciTech Publishing Inc.

Mahafza, B. R. (2000). Radar Cross Section (RCS). En B. R. Mahafza, *Radar Systems Analysis and Design Using MATLAB.* United States of America.Florida: Chapman & Hall.

Pérez Alcaraz, C. (2012). Funciones de Green. Aspectos Computaconales. Universidad de Murcia.

Pérez Restrepo, S. I. (14 de Agosto de 2013). Métodos de predicción de la RCS. *Cálculo de la sección transversal de radar (RCS) de una placa plana y de un cilindro, para la evaluación del impacto de aerogeneradores en sistemas de navegación aérea*. Medellín, Colombia: Universidad Pontificia Bolivariana.Escuela de Ingenierías. Facultad de Ingeniería Aeronáutica.

Perez, T., & Fossen, T. I. (2007). Kinematic Models for Manoeuvring and Seakeeping of Marine Vessels. En *Modeling, Identification and Control* (págs. 19-30). Norway: Norwegian University of Science and Technology.

Rius Casals, J. M. (8 de Julio de 1991). Métodos de predicción de sección recta radar. *Sección recta de blancos radar complejos en tiempo real*. Barcelona, Catalunia, España: Universidad Politécnica de Catalunia.

Roopchand, V. (17 de Enero de 2013). *Matlab Central*. (T. M. Inc., Productor) Obtenido de Math Works: http://www.mathworks.com/matlabcentral/fileexchange/index?term=id%3A39894

Serway, R. A., & Jewett, J. W. (2008). *Física para Ciencias e Ingeniería con Física Moderna.* México: CENGAGE Learning.

Skolnik, M. I. (1980). Radar Antennas. En *Introducction to Radar Systems* (págs. 223-273). Singapore: McGraw Hill Inc.

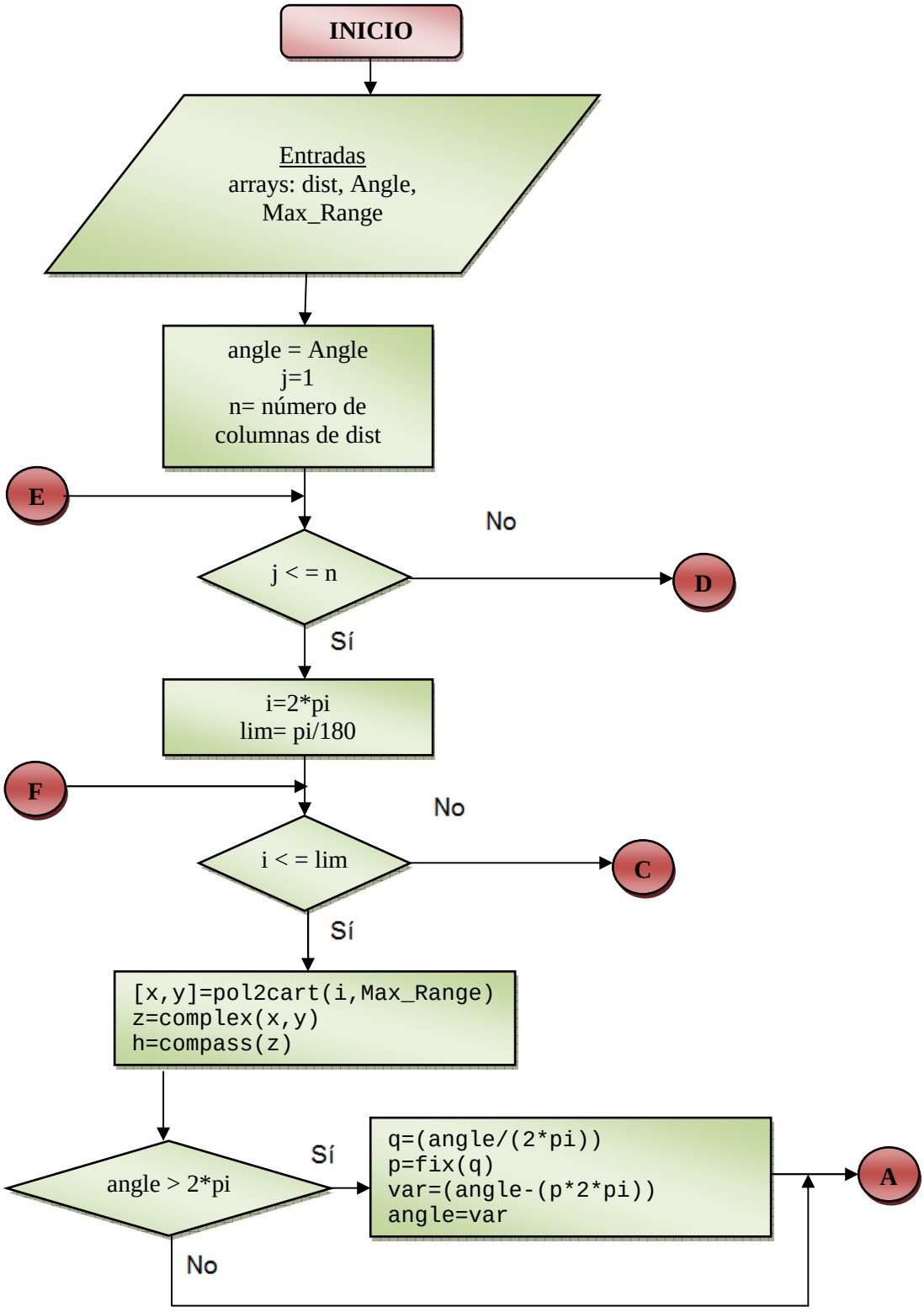
Skolnik, M. I. (2008). *Radar Handbook.* United States of America: McGraw Hill.

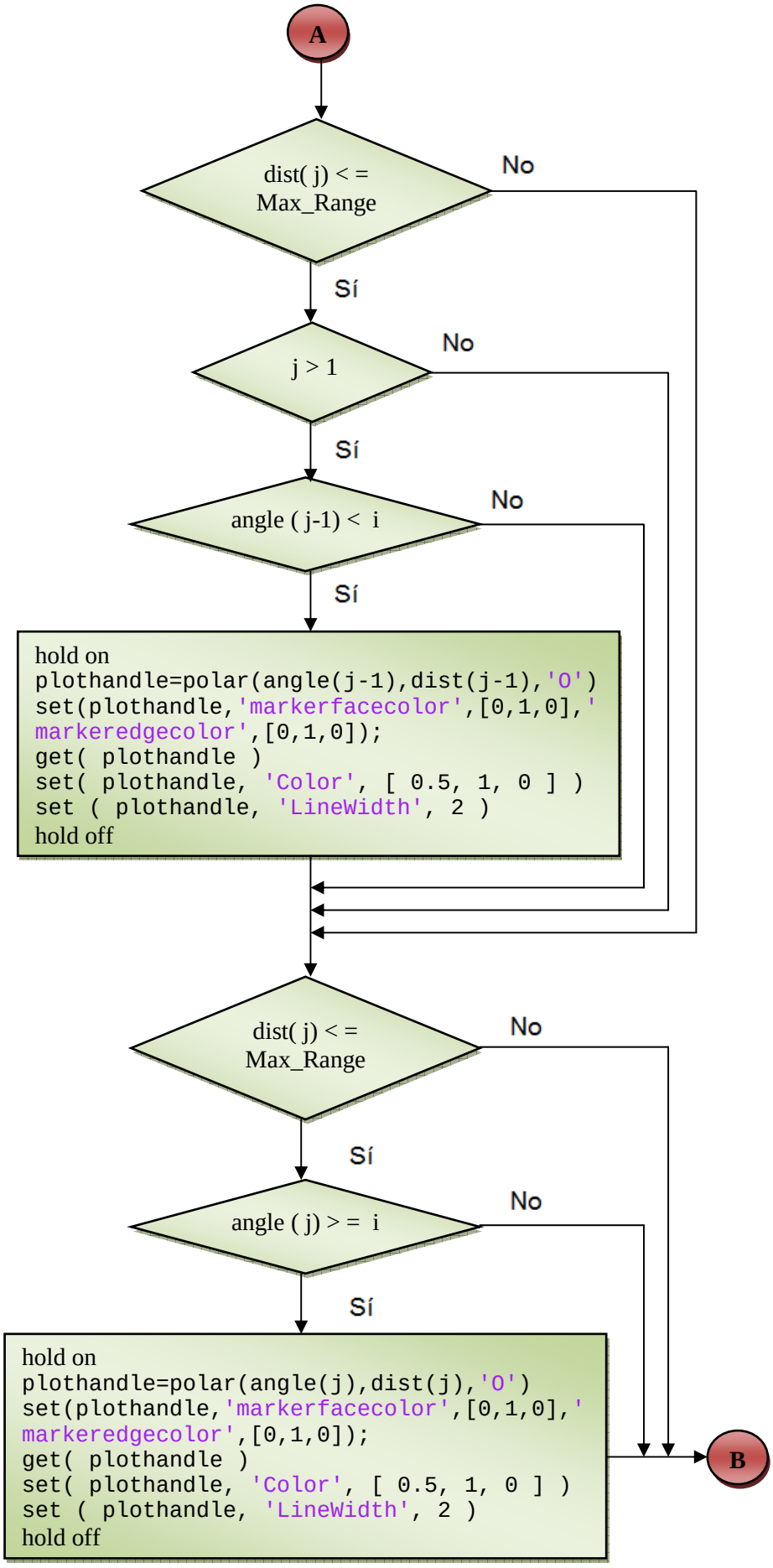
Williams, P. D., Cramp, H. D., & Curtis, K. (26 de Julio de 1978). Experimental study of the radar cross section of maritime targets. (IEEE, Ed.) *Electronics Circuits and Style, 2*(4), 121-136.

Williams, P. D., Cramp, H. D., & Curtis, K. (s.f.). Experimental Study +++.

Anexo A
Diagramas de flujo del algoritmo principal y funciones utilizadas para la simulación de las escenas de radar

Diagrama de flujo de la función *PPI_plot_ONE_TARGET*




A
dist(j) < =
Max_Range
No
Sí
j > 1
No
Sí
angle (j-1) < i
No
Sí
hold on
plothandle=polar(angle(j-1),dist(j-1),'O')
set(plothandle,'markerfacecolor',[0,1,0],'
markeredgecolor',[0,1,0]);
get(plothandle)
set(plothandle, 'Color', [0.5, 1, 0])
set (plothandle, 'LineWidth', 2)
hold off
dist(j) < =
Max_Range
No
Sí
angle (j) > = i
No
Sí
hold on
plothandle=polar(angle(j),dist(j),'O')
set(plothandle,'markerfacecolor',[0,1,0],'
markeredgecolor',[0,1,0]);
get(plothandle)
set(plothandle, 'Color', [0.5, 1, 0])
set (plothandle, 'LineWidth', 2)
hold off
B

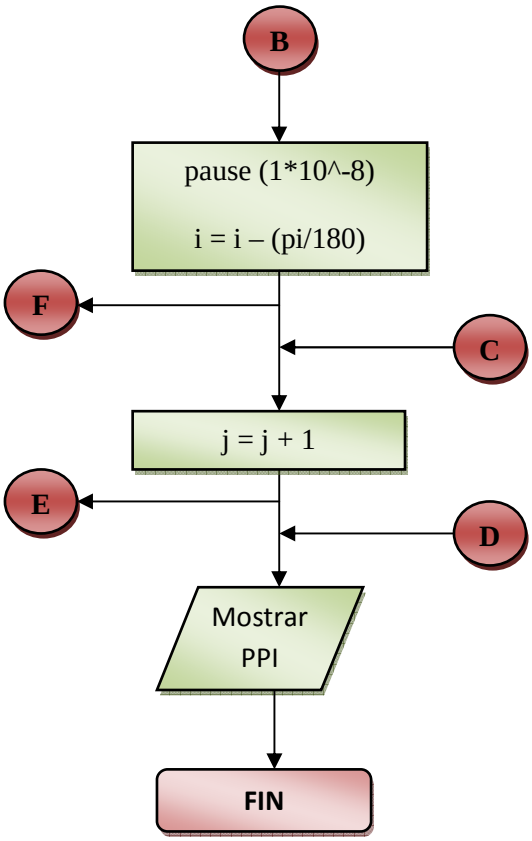
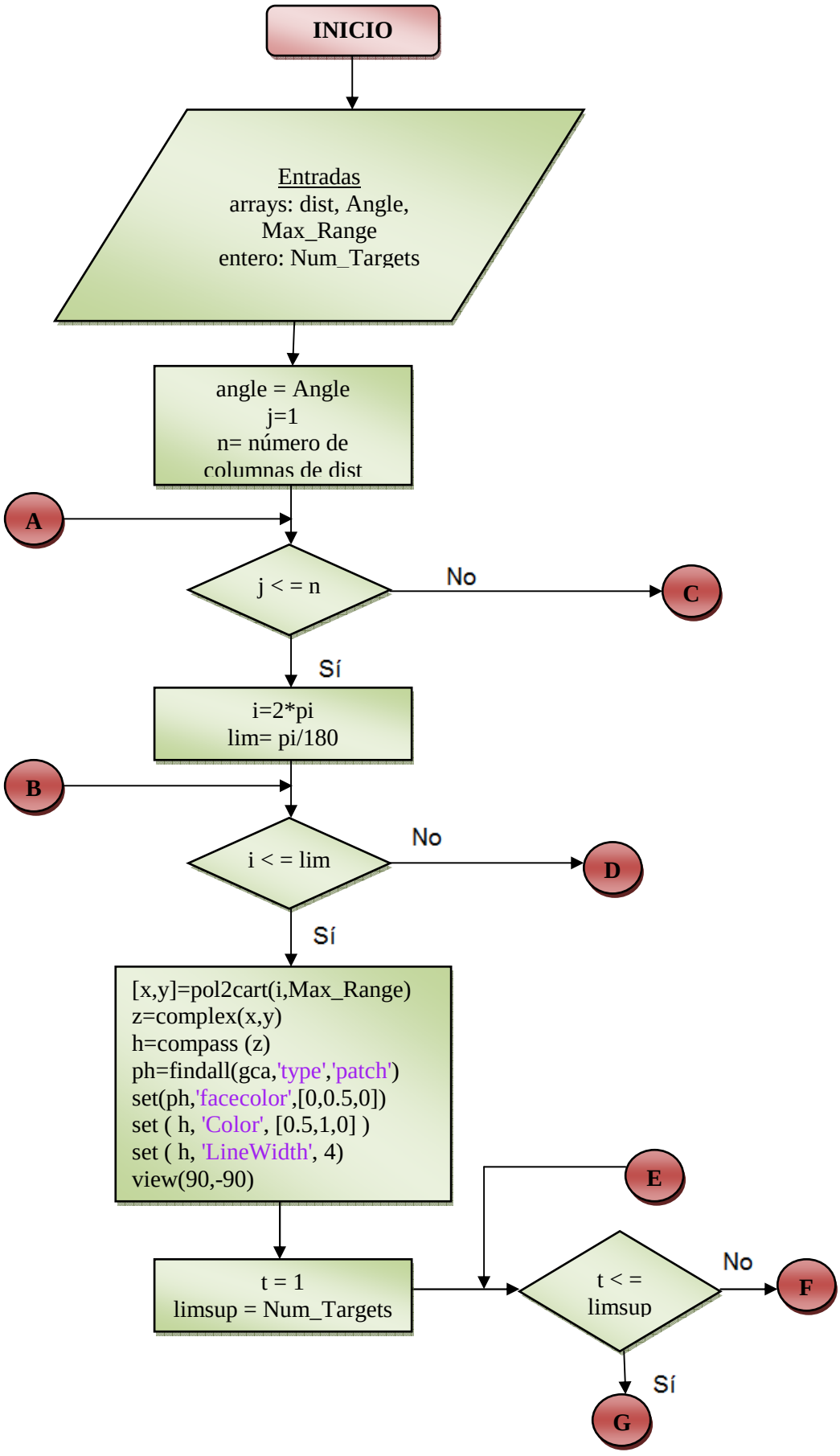
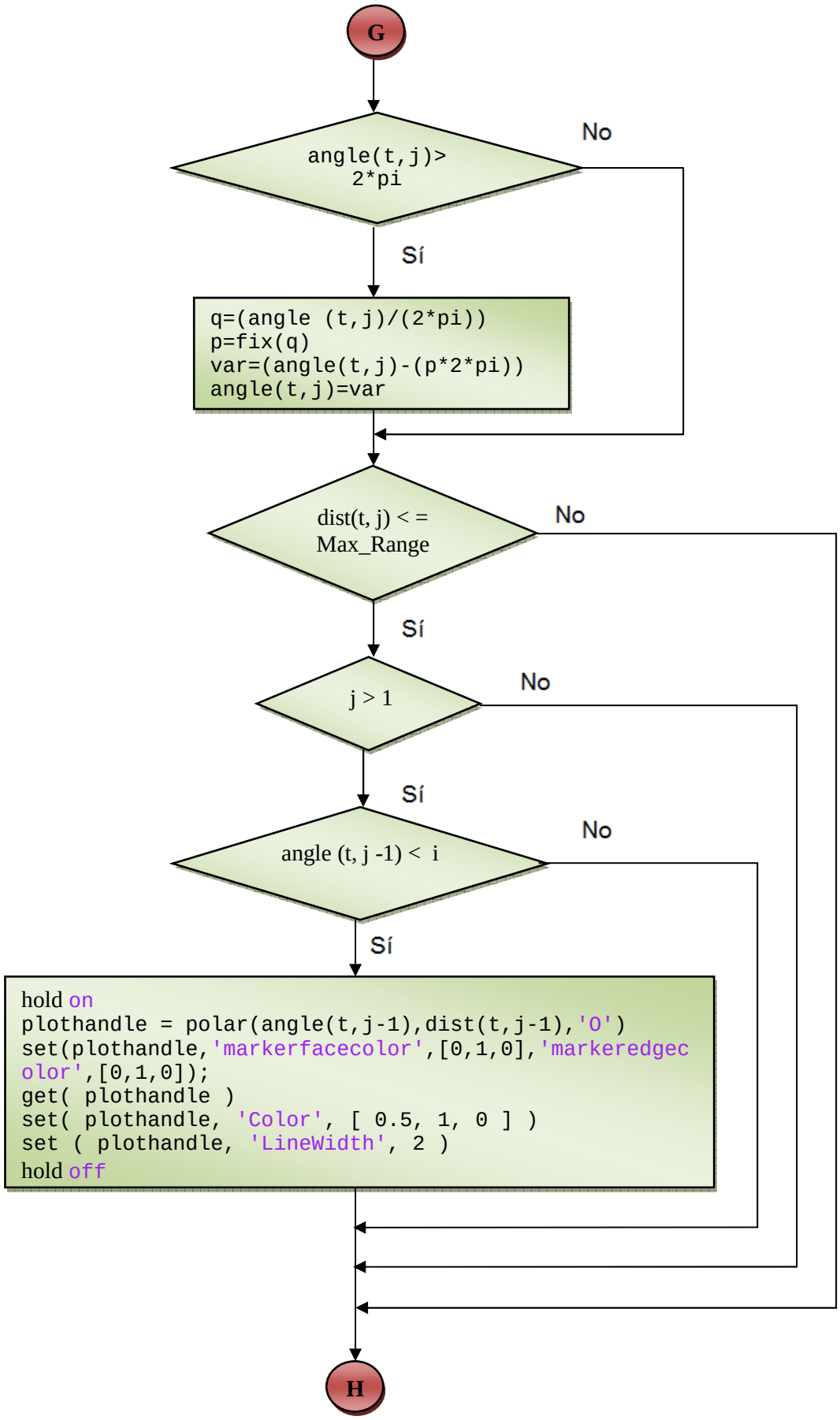

B
pause (1*10^-8)
i = i – (pi/180)
F
C
j = j + 1
E
D
Mostrar
PPI
FIN

Diagrama de flujo de la función *PPI_plot_MULTIPLE_TARGETS*




G
angle(t,j)>
2*pi
No
Sí
q=(angle (t,j)/(2*pi))
p=fix(q)
var=(angle(t,j)-(p*2*pi))
angle(t,j)=var
dist(t, j) < =
Max_Range
No
Sí
j > 1
No
Sí
angle (t, j -1) < i
No
Sí
hold on
plothandle = polar(angle(t,j-1),dist(t,j-1),'O')
set(plothandle,'markerfacecolor',[0,1,0],'markeredgec
olor',[0,1,0]);
get(plothandle)
set(plothandle, 'Color', [0.5, 1, 0])
set (plothandle, 'LineWidth', 2)
hold off
H

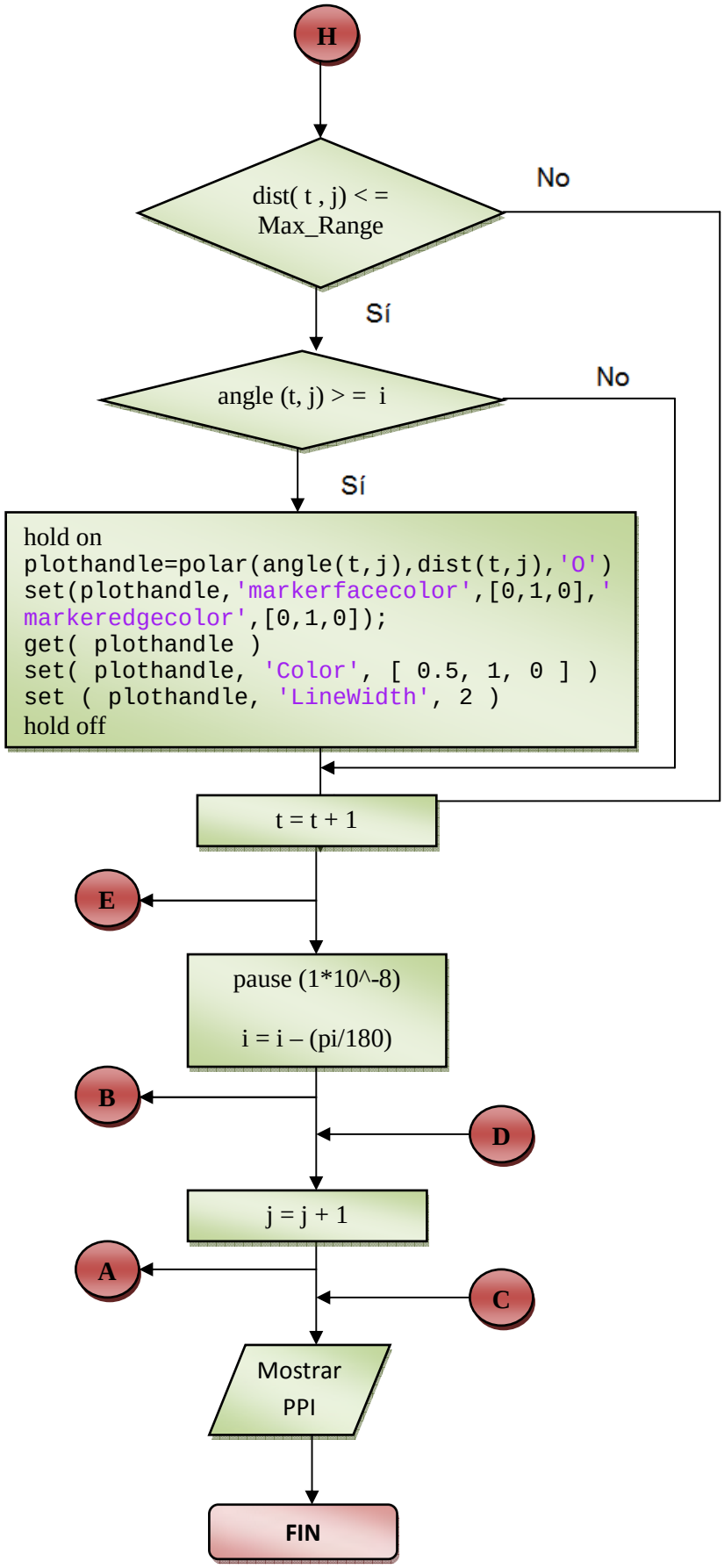

H
dist(t , j) < =
Max_Range
No
Sí
angle (t, j) > = i
No
Sí
hold on
plothandle=polar(angle(t,j),dist(t,j),'O')
set(plothandle,'markerfacecolor',[0,1,0],'
markeredgecolor',[0,1,0]);
get(plothandle)
set(plothandle, 'Color', [0.5, 1, 0])
set (plothandle, 'LineWidth', 2)
hold off
t = t + 1
E
pause (1*10^-8)
i = i – (pi/180)
B
D
j = j + 1
A
C
Mostrar
PPI
FIN

Diagrama de flujo de la función *Mostrar_PPI*

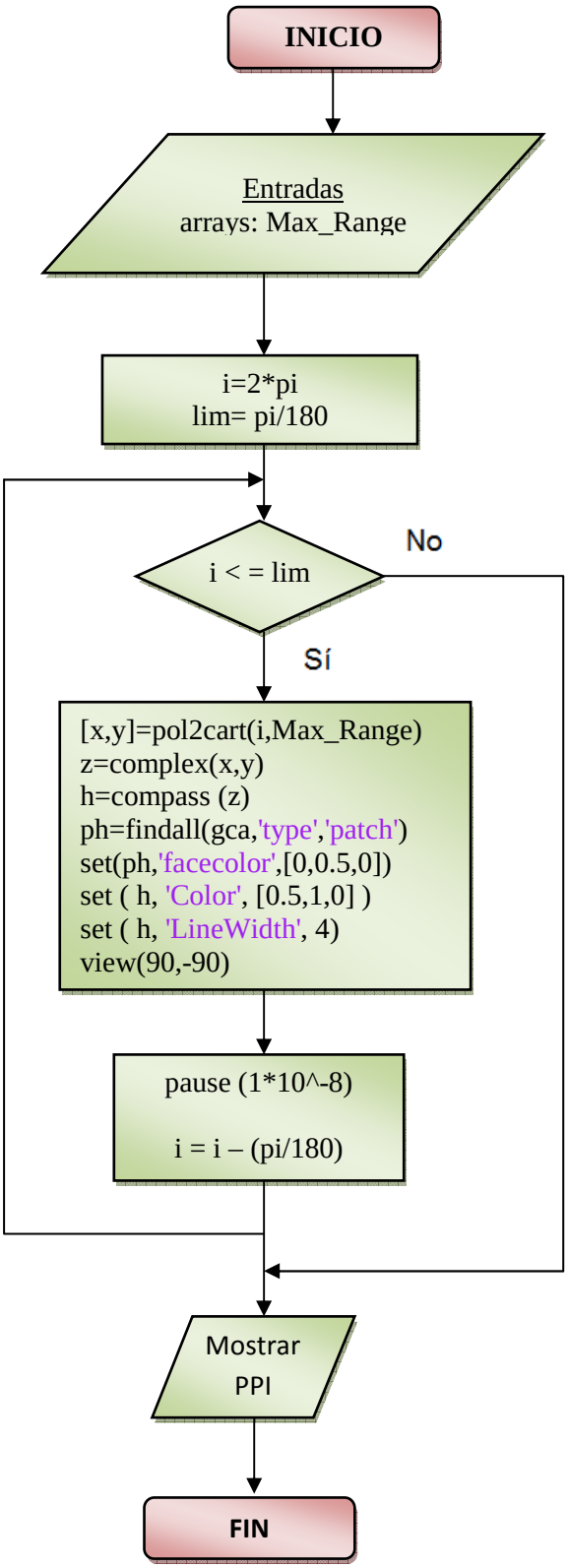


Diagrama de flujo de la función *Mostrar_Menu*

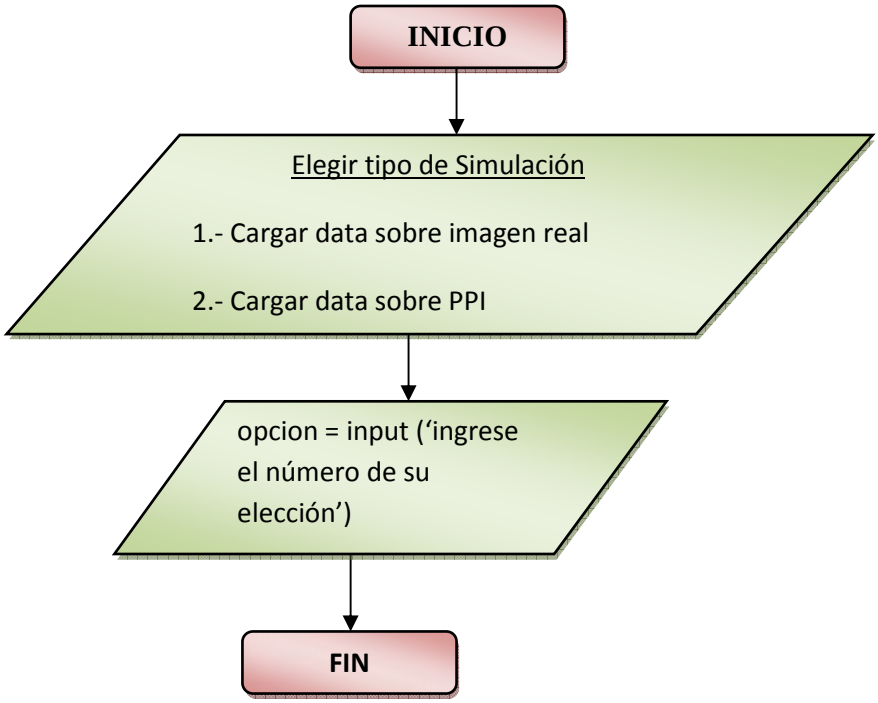


Diagrama de flujo de la función
Mostrar_Menu_para_Elegir_la_Trayectoria_del_Buque

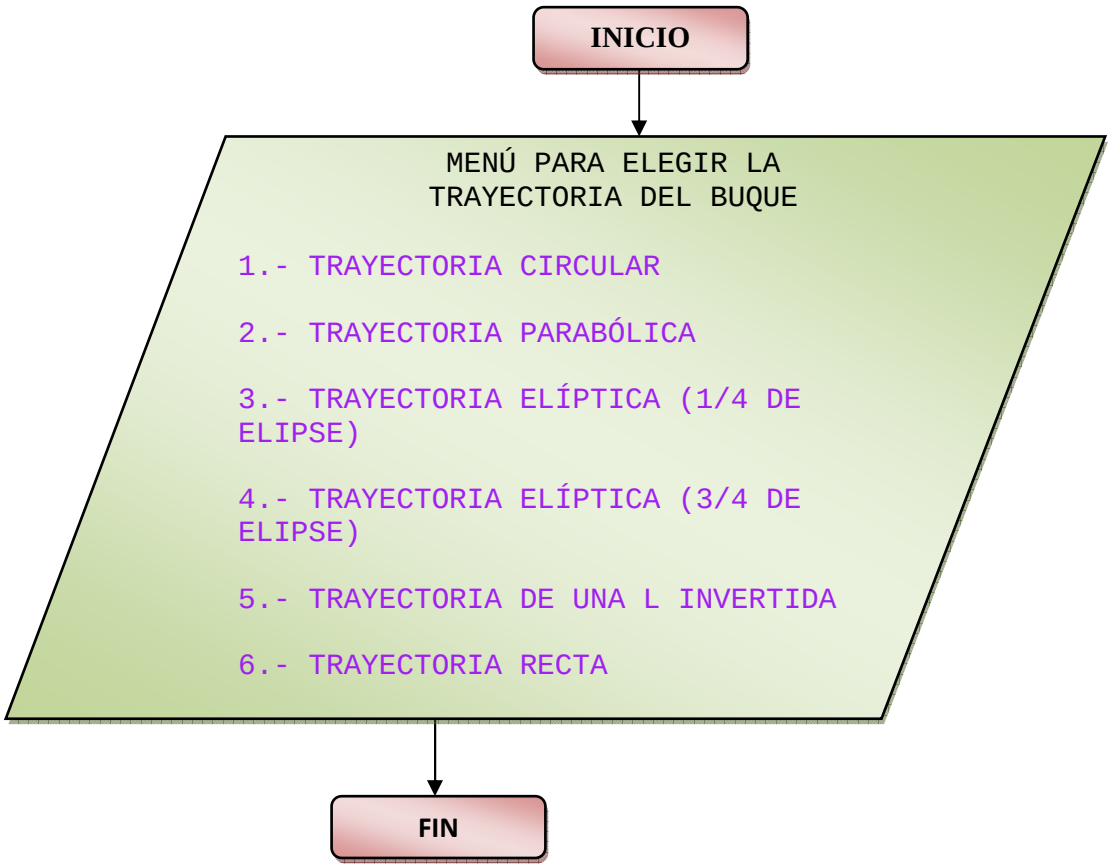
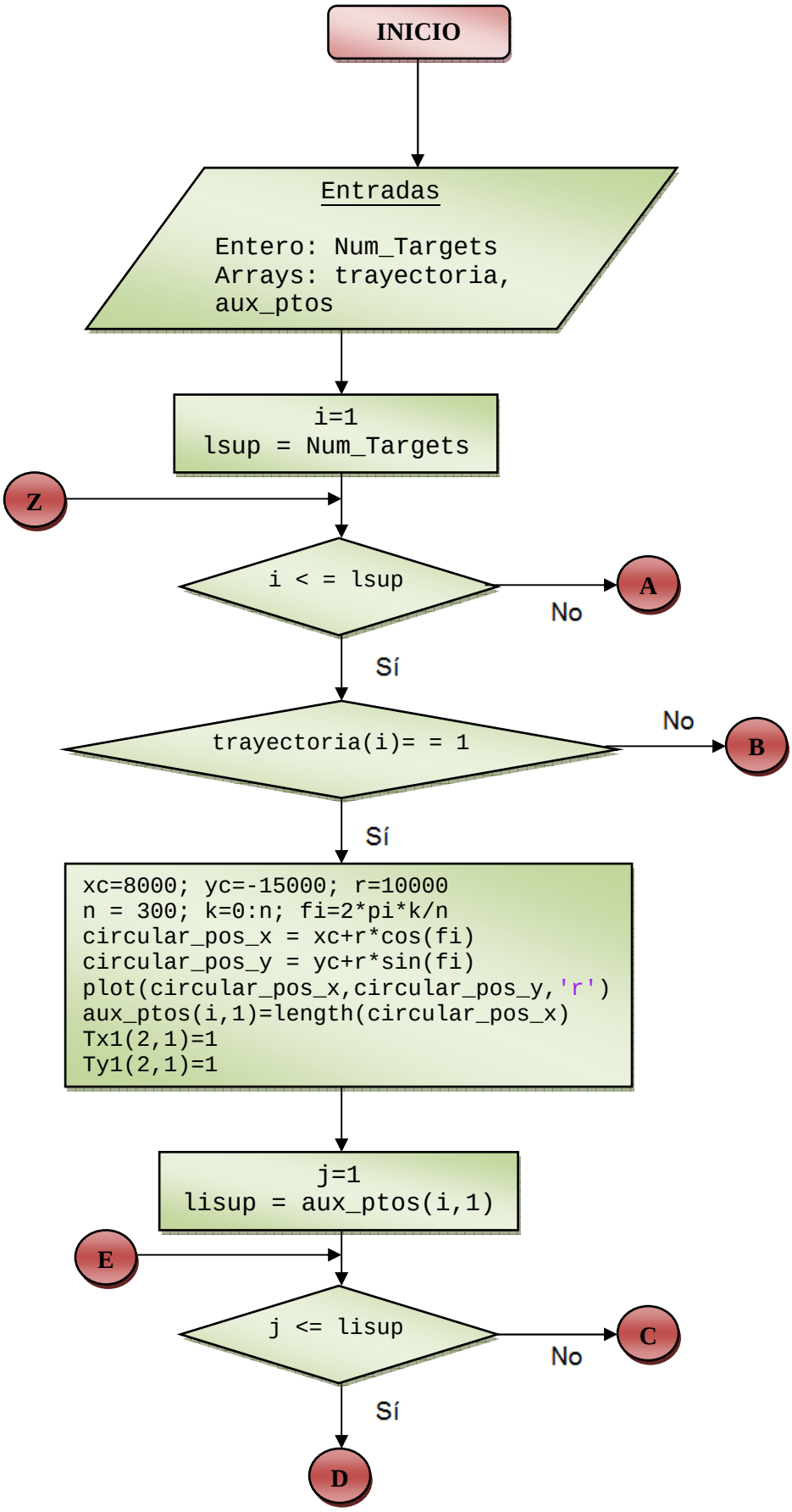
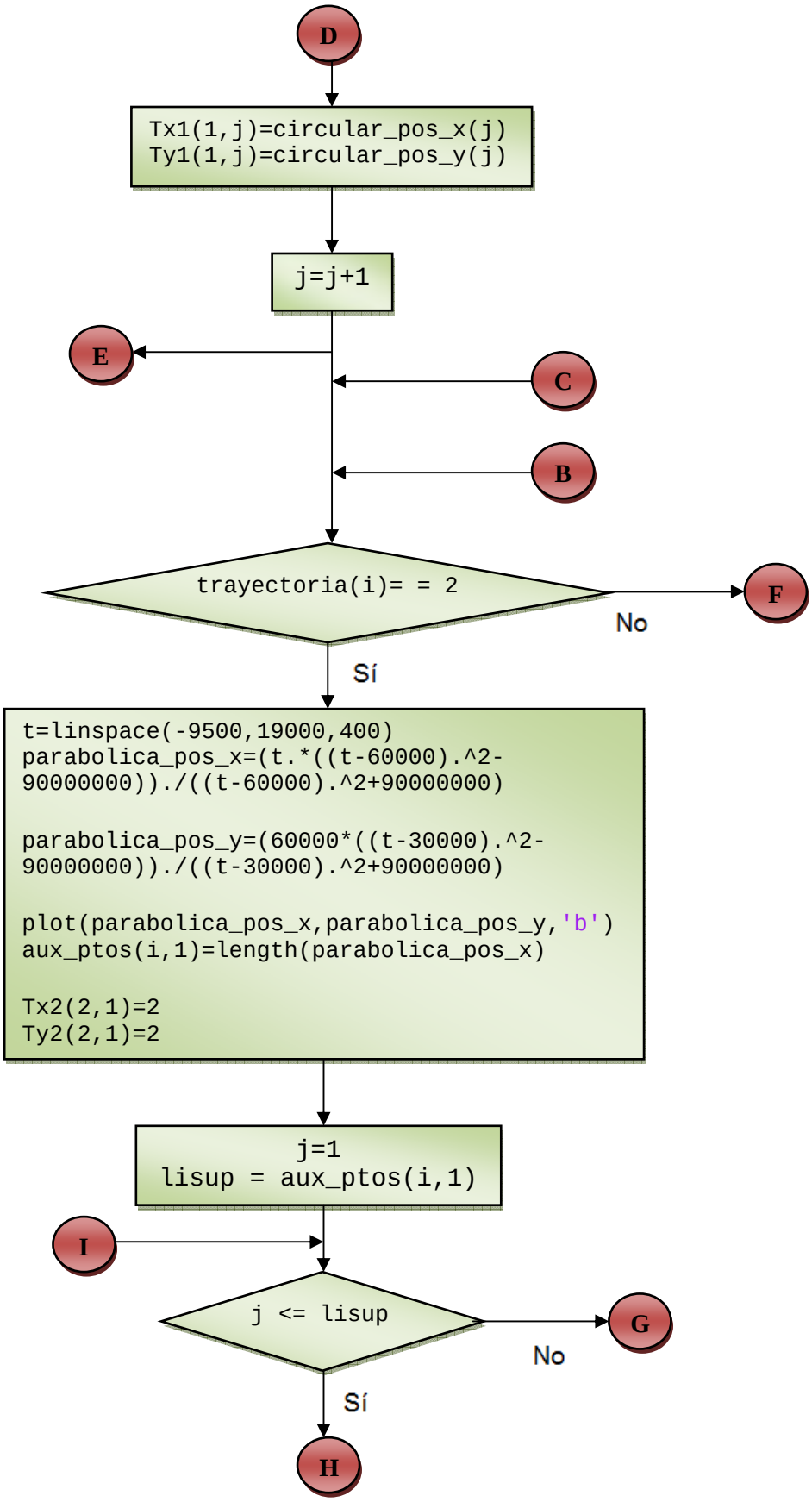
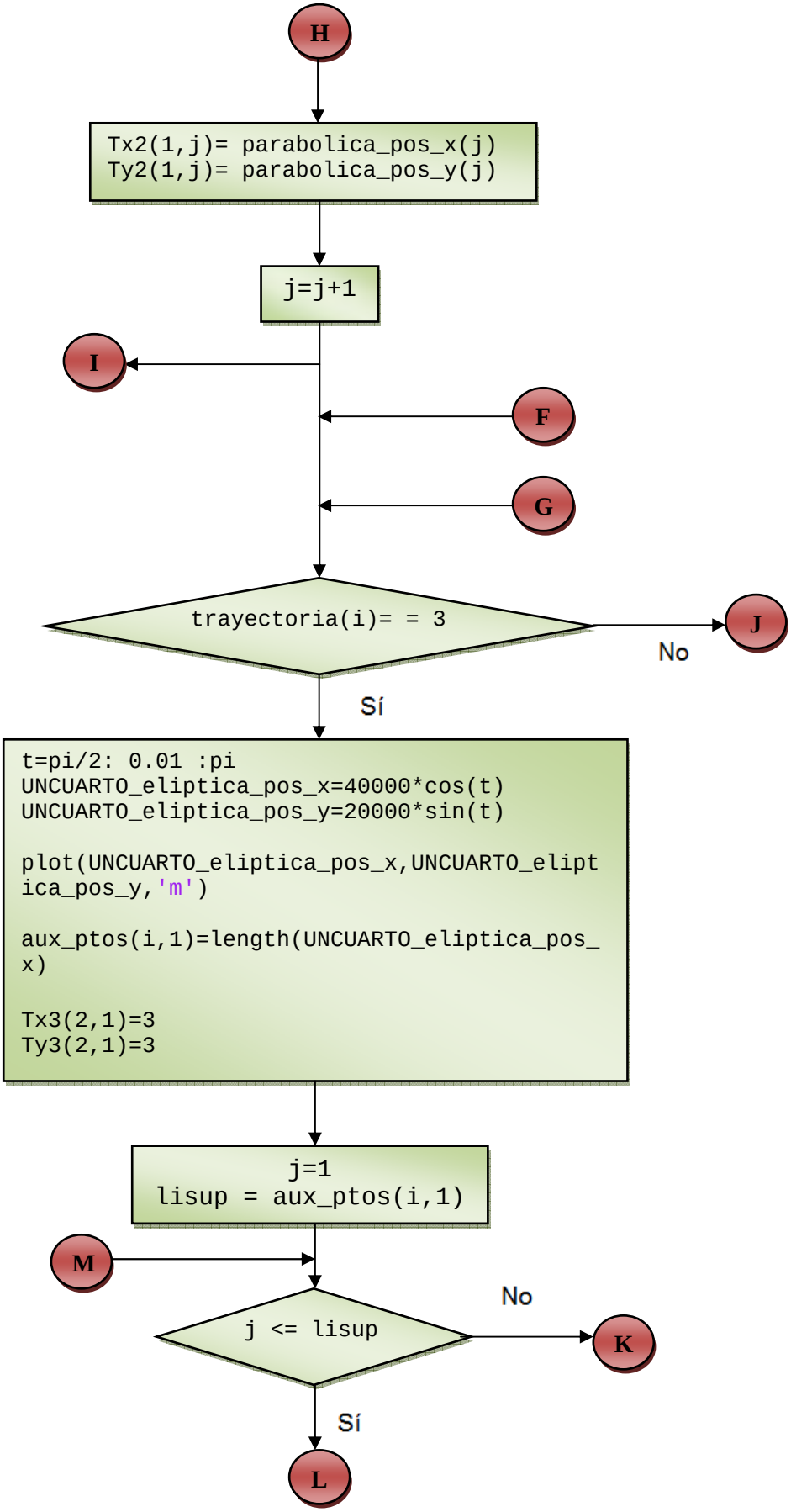


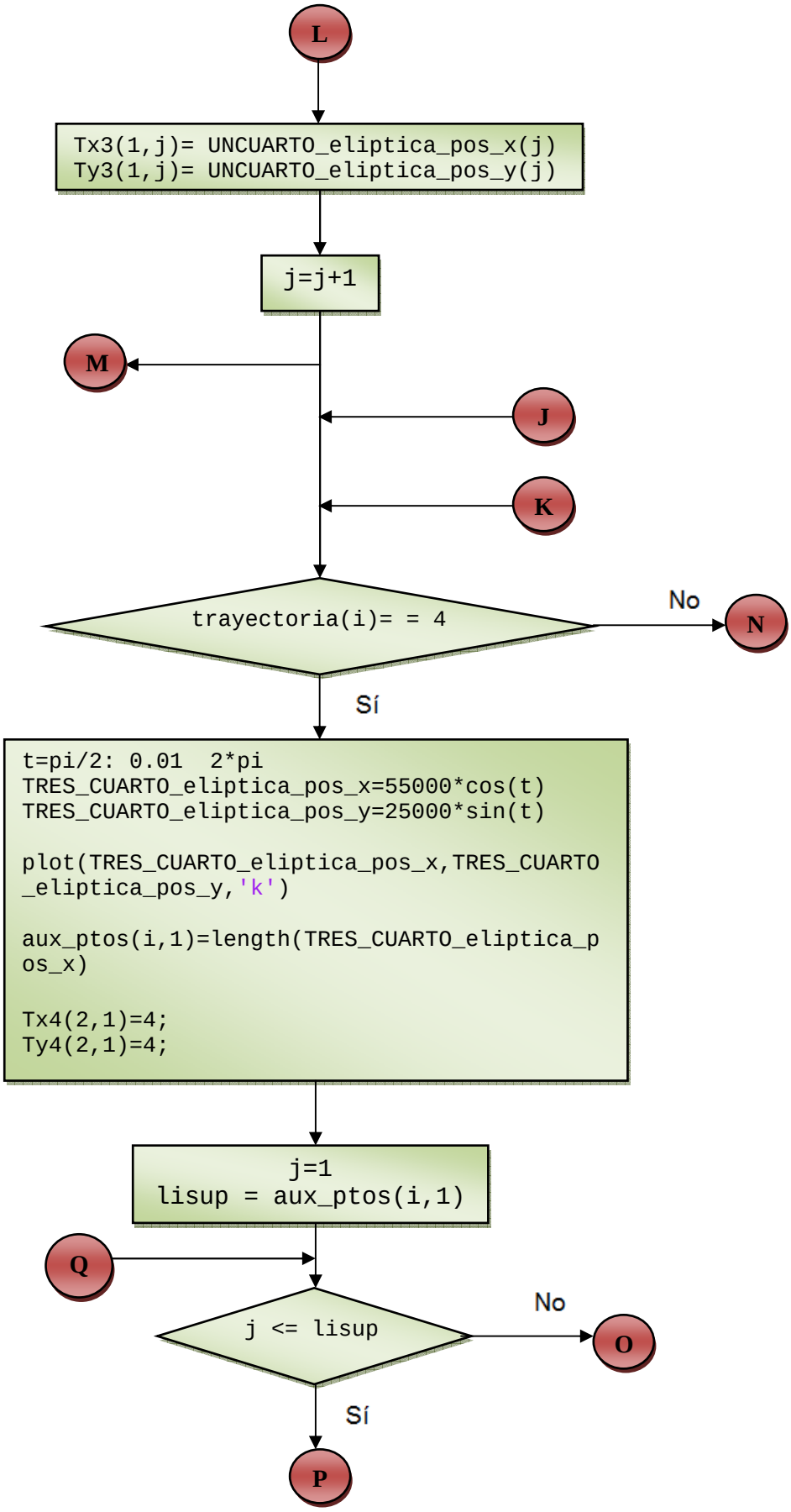
Diagrama de flujo de la función *Mostrar_mensaje_al_usuario*

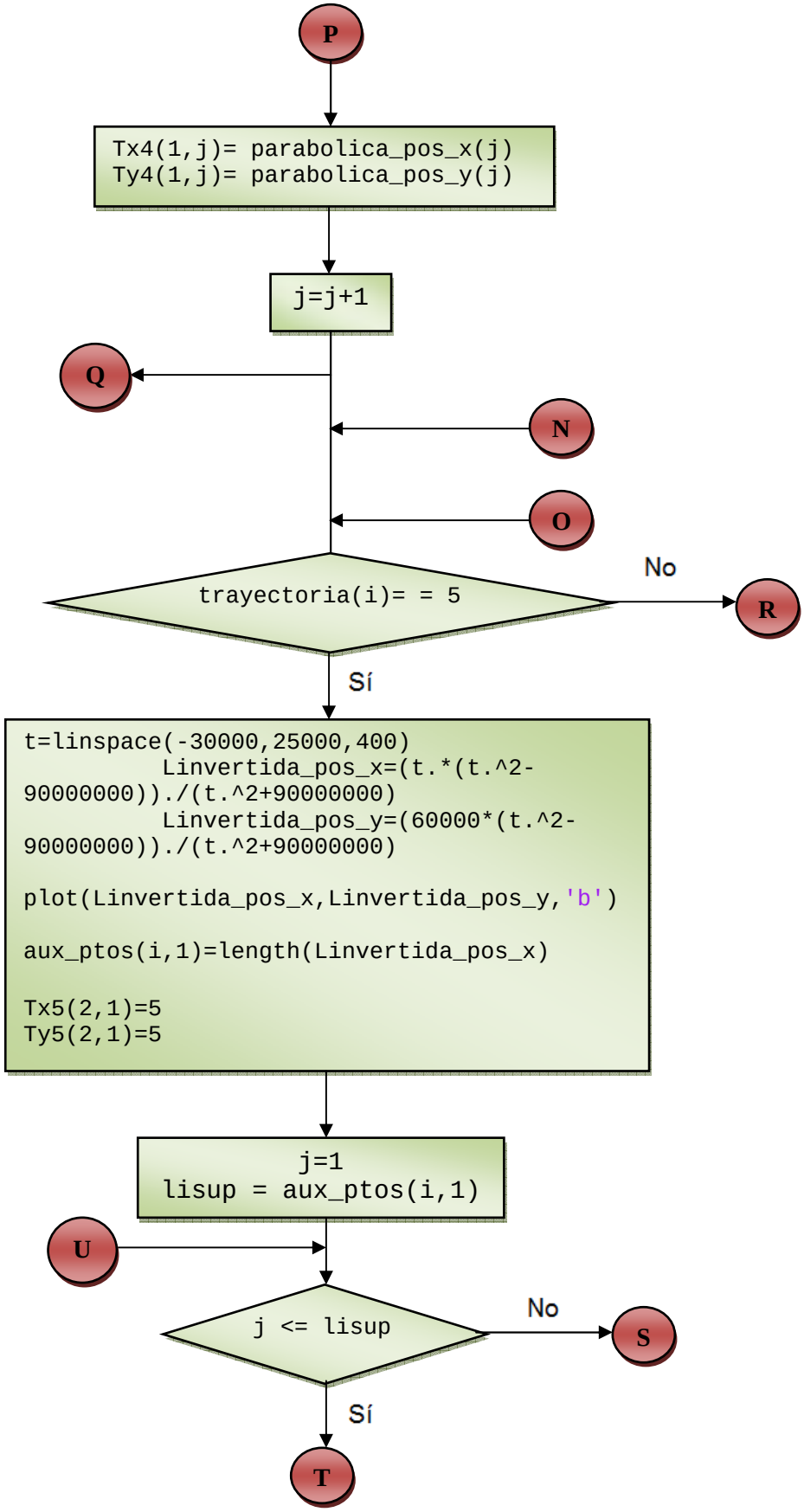
Diagrama de flujo de la función *Generador_de_posiciones_para_buques*

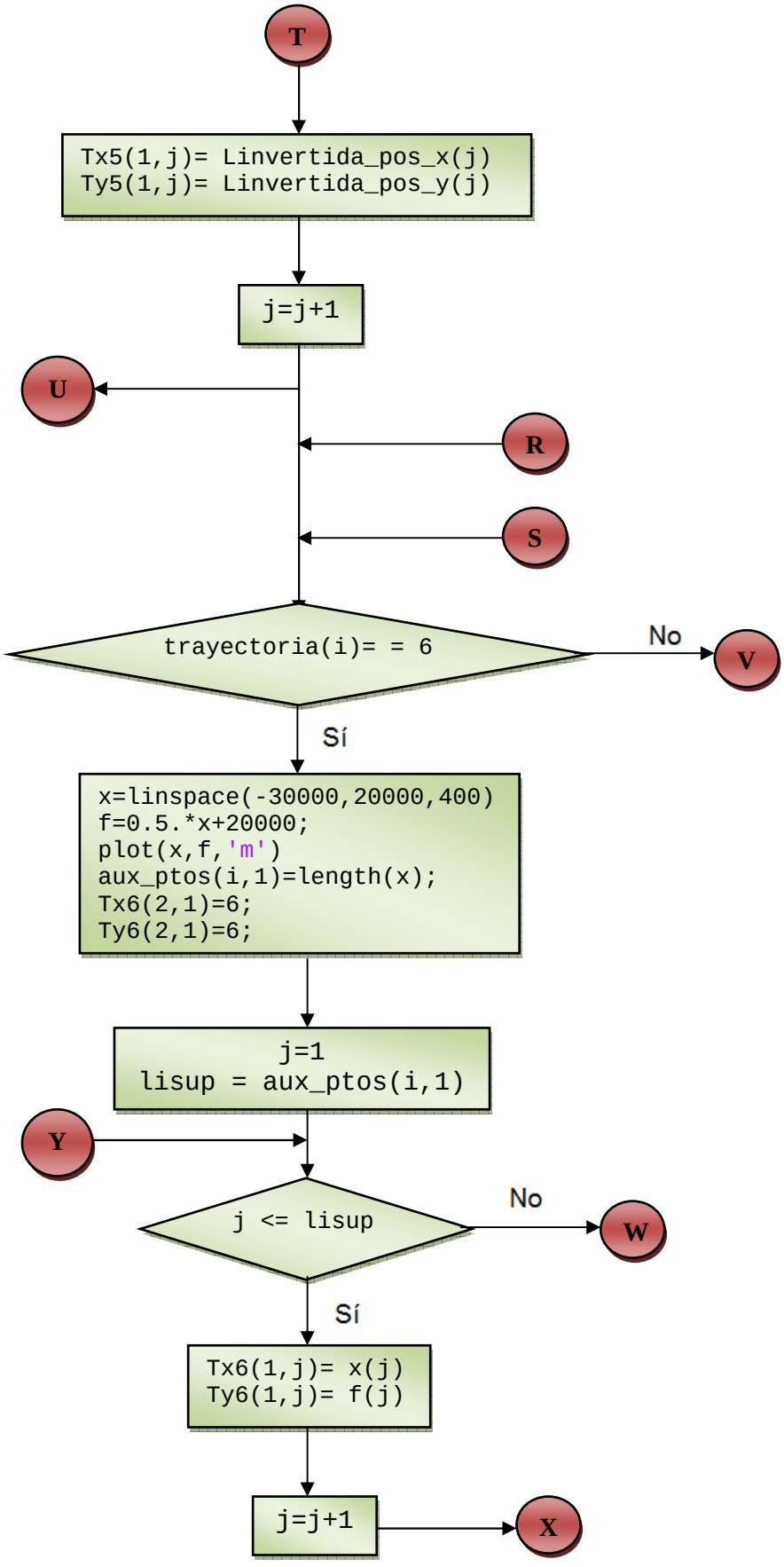


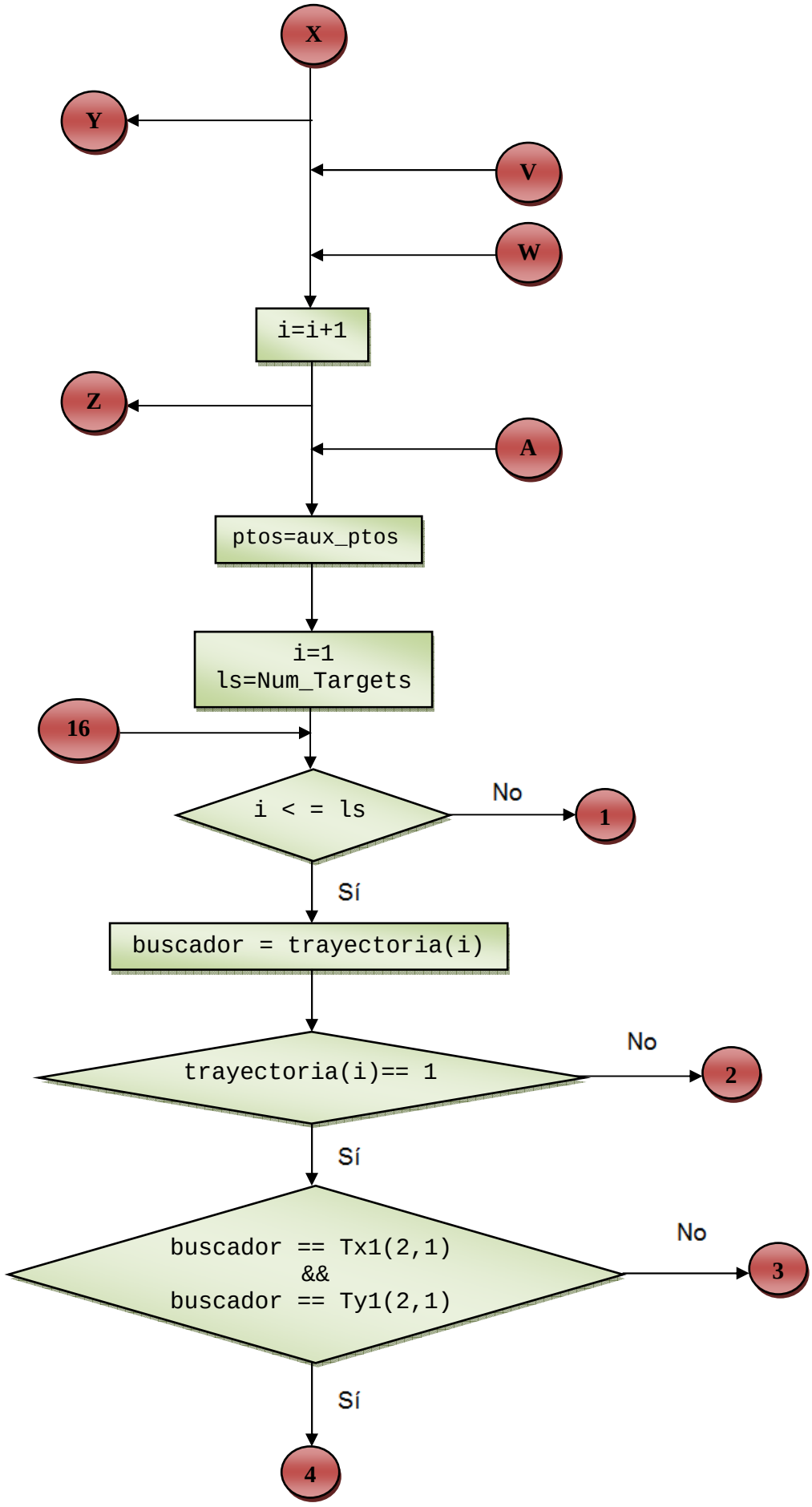

D
Tx1(1,j)=circular_pos_x(j)
Ty1(1,j)=circular_pos_y(j)
j=j+1
E
C
B
trayectoria(i)= = 2
No
F
Sí
t=linspace(-9500,19000,400)
parabolica_pos_x=(t.*((t-60000).^2-
90000000))./((t-60000).^2+90000000)
parabolica_pos_y=(60000*((t-30000).^2-
90000000))./((t-30000).^2+90000000)
plot(parabolica_pos_x,parabolica_pos_y,'b')
aux_ptos(i,1)=length(parabolica_pos_x)
Tx2(2,1)=2
Ty2(2,1)=2
j=1
lisup = aux_ptos(i,1)
I
j <= lisup
No
G
Sí
H

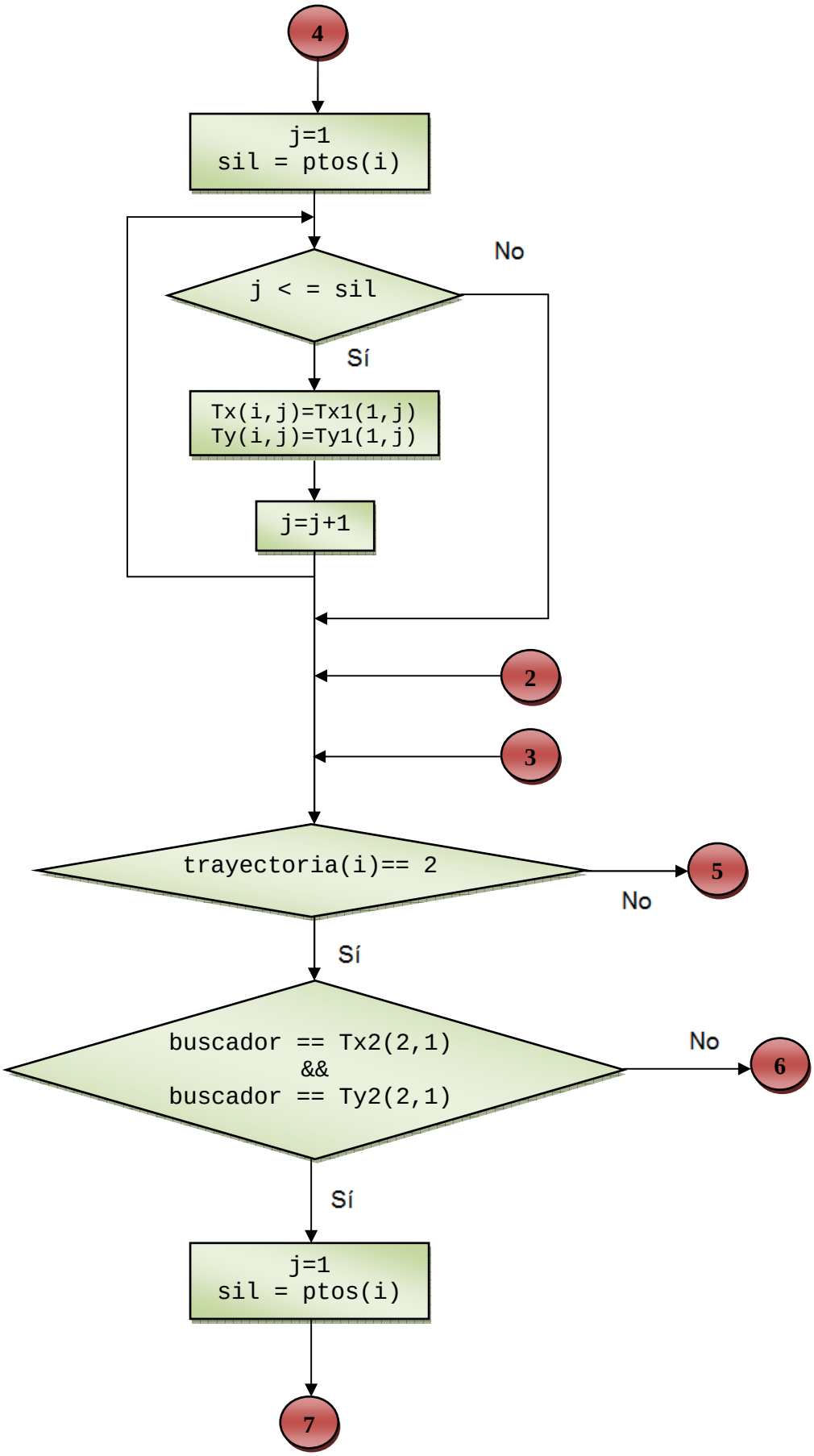

H
Tx2(1,j)= parabolica_pos_x(j)
Ty2(1,j)= parabolica_pos_y(j)
j=j+1
I
F
G
trayectoria(i)= = 3
No
J
Sí
t=pi/2: 0.01 :pi
UNCUARTO_eliptica_pos_x=40000*cos(t)
UNCUARTO_eliptica_pos_y=20000*sin(t)
plot(UNCUARTO_eliptica_pos_x,UNCUARTO_elipt
ica_pos_y,'m')
aux_ptos(i,1)=length(UNCUARTO_eliptica_pos_
x)
Tx3(2,1)=3
Ty3(2,1)=3
j=1
lisup = aux_ptos(i,1)
M
No
j <= lisup
K
Sí
L

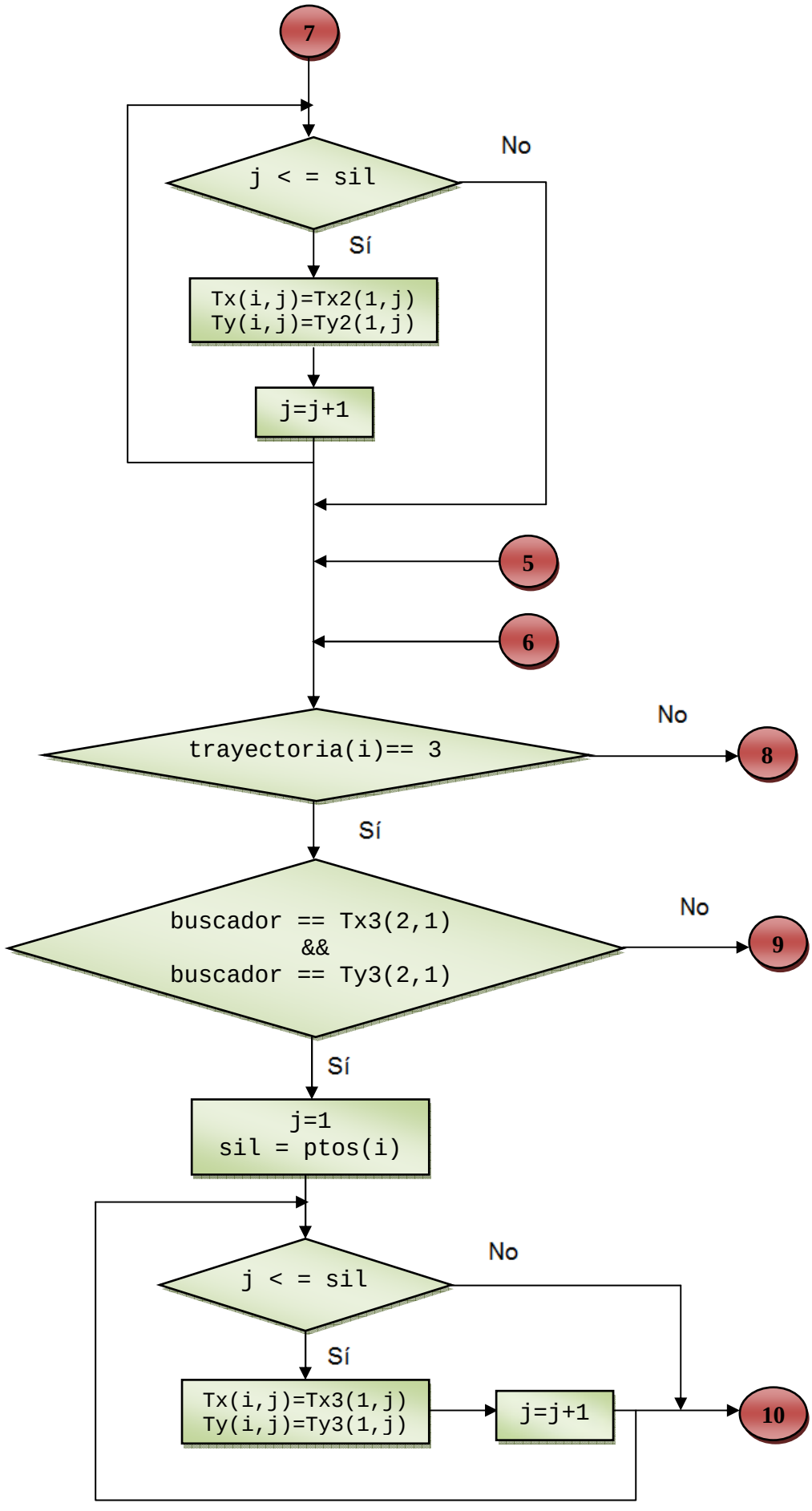

L
Tx3(1,j)= UNCUARTO_eliptica_pos_x(j)
Ty3(1,j)= UNCUARTO_eliptica_pos_y(j)
j=j+1
M
J
K
trayectoria(i)= = 4
No
N
Sí
t=pi/2: 0.01 2*pi
TRES_CUARTO_eliptica_pos_x=55000*cos(t)
TRES_CUARTO_eliptica_pos_y=25000*sin(t)
plot(TRES_CUARTO_eliptica_pos_x,TRES_CUARTO_eliptica_pos_y,'k')
aux_ptos(i,1)=length(TRES_CUARTO_eliptica_pos_x)
Tx4(2,1)=4;
Ty4(2,1)=4;
j=1
lisup = aux_ptos(i,1)
Q
j <= lisup
No
O
Sí
P

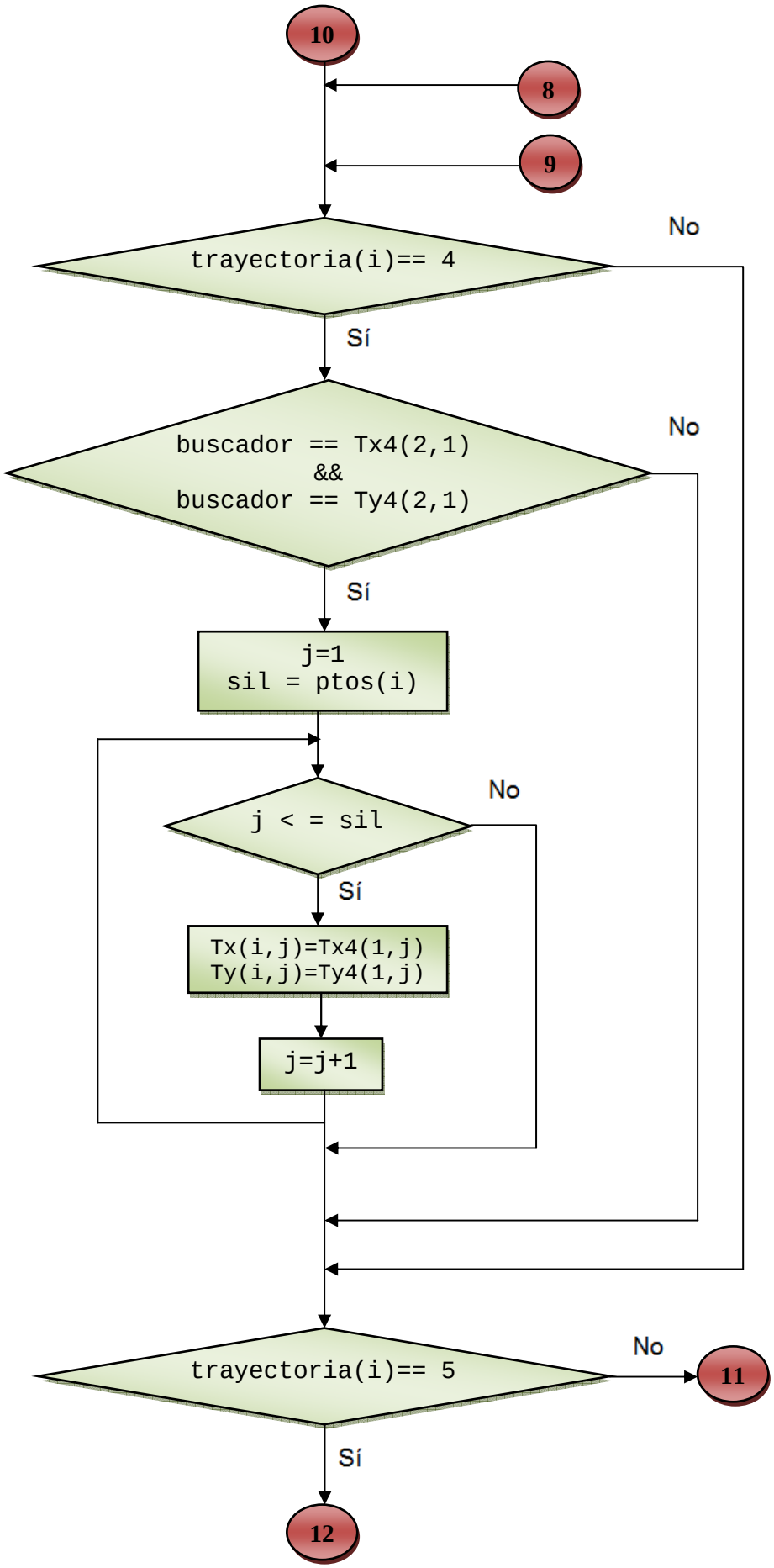

P
Tx4(1,j)= parabolica_pos_x(j)
Ty4(1,j)= parabolica_pos_y(j)
j=j+1
Q
N
O
trayectoria(i)= = 5
No
R
Sí
t=linspace(-30000,25000,400)
Linvertida_pos_x=(t.*(t.^2-
90000000))./(t.^2+90000000)
Linvertida_pos_y=(60000*(t.^2-
90000000))./(t.^2+90000000)
plot(Linvertida_pos_x,Linvertida_pos_y,'b')
aux_ptos(i,1)=length(Linvertida_pos_x)
Tx5(2,1)=5
Ty5(2,1)=5
j=1
lisup = aux_ptos(i,1)
U
j <= lisup
No
S
Sí
T

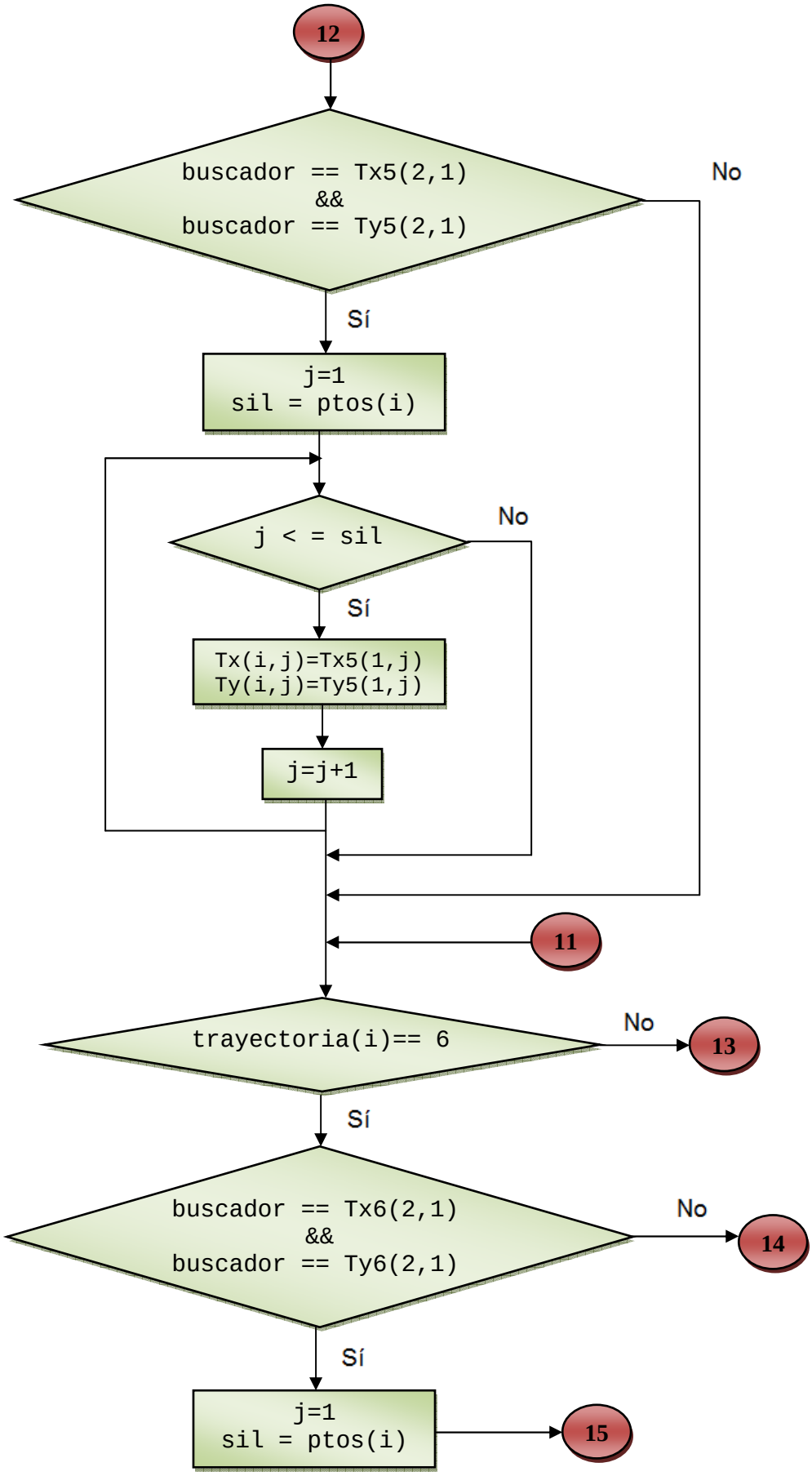

T
Tx5(1,j)= Linvertida_pos_x(j)
Ty5(1,j)= Linvertida_pos_y(j)
j=j+1
U
R
S
trayectoria(i)= = 6
No
V
Sí
x=linspace(-30000,20000,400)
f=0.5.*x+20000;
plot(x,f,'m')
aux_ptos(i,1)=length(x);
Tx6(2,1)=6;
Ty6(2,1)=6;
j=1
lisup = aux_ptos(i,1)
Y
j <= lisup
No
W
Sí
Tx6(1,j)= x(j)
Ty6(1,j)= f(j)
j=j+1
X


X
Y
V
W
i=i+1
Z
A
ptos=aux_ptos
i=1
ls=Num_Targets
16
i < = ls
No
1
Sí
buscador = trayectoria(i)
trayectoria(i)== 1
No
2
Sí
buscador == Tx1(2,1)
&&
buscador == Ty1(2,1)
No
3
Sí
4


4
j=1
sil = ptos(i)
No
j < = sil
Sí
Tx(i,j)=Tx1(1,j)
Ty(i,j)=Ty1(1,j)
j=j+1
2
3
trayectoria(i)== 2
5
No
Sí
buscador == Tx2(2,1)
&&
buscador == Ty2(2,1)
No
6
Sí
j=1
sil = ptos(i)
7


7
j < = sil
No
Sí
Tx(i,j)=Tx2(1,j)
Ty(i,j)=Ty2(1,j)
j=j+1
5
6
trayectoria(i)== 3
No
8
Sí
buscador == Tx3(2,1)
&&
buscador == Ty3(2,1)
No
9
Sí
j=1
sil = ptos(i)
j < = sil
No
Sí
Tx(i,j)=Tx3(1,j)
Ty(i,j)=Ty3(1,j)
j=j+1
10


10
8
9
trayectoria(i)== 4
No
Sí
buscador == Tx4(2,1)
&&
buscador == Ty4(2,1)
No
Sí
j=1
sil = ptos(i)
j < = sil
No
Sí
Tx(i,j)=Tx4(1,j)
Ty(i,j)=Ty4(1,j)
j=j+1
trayectoria(i)== 5
No
11
Sí
12


12
buscador == Tx5(2,1)
&&
buscador == Ty5(2,1)
No
Sí
j=1
sil = ptos(i)
j < = sil
No
Sí
Tx(i,j)=Tx5(1,j)
Ty(i,j)=Ty5(1,j)
j=j+1
11
trayectoria(i)== 6
No
13
Sí
buscador == Tx6(2,1)
&&
buscador == Ty6(2,1)
No
14
Sí
j=1
sil = ptos(i)
15

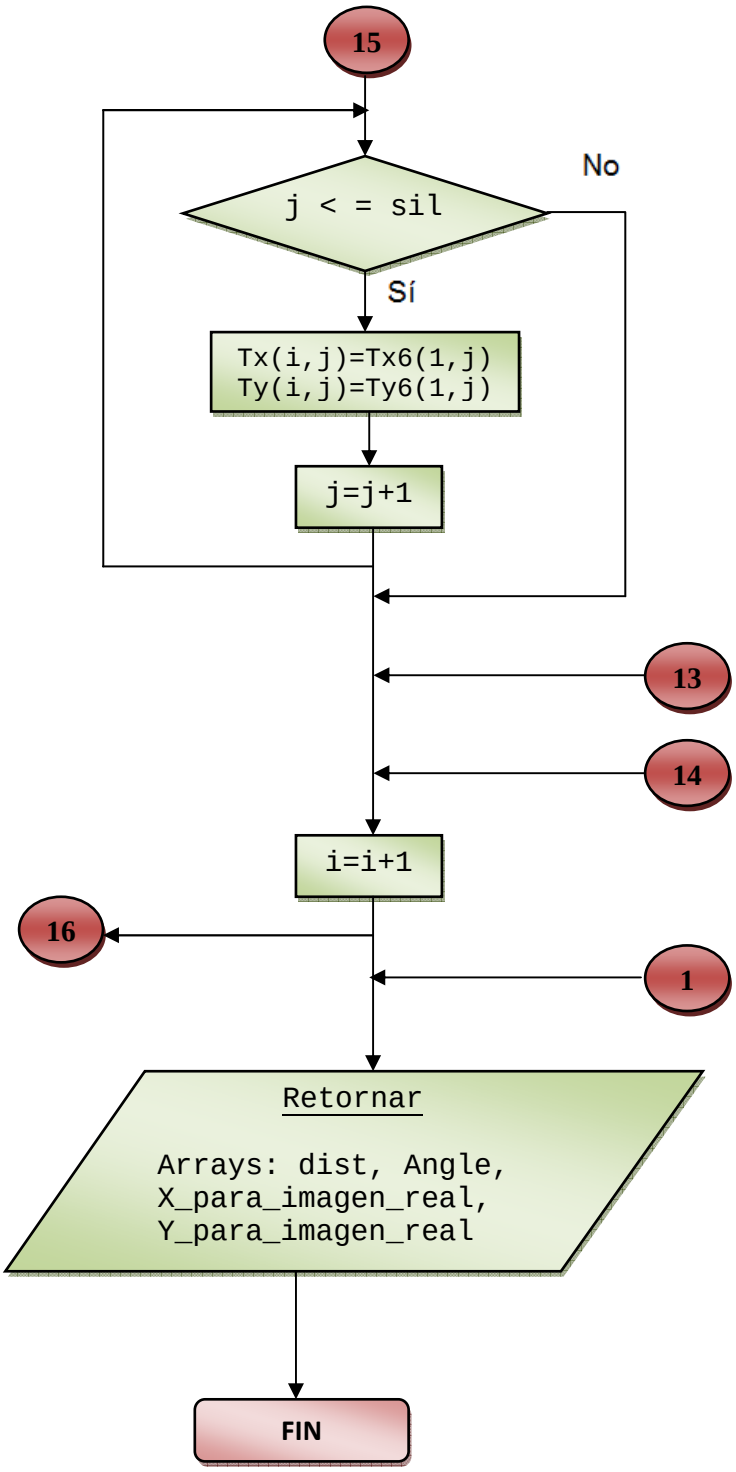
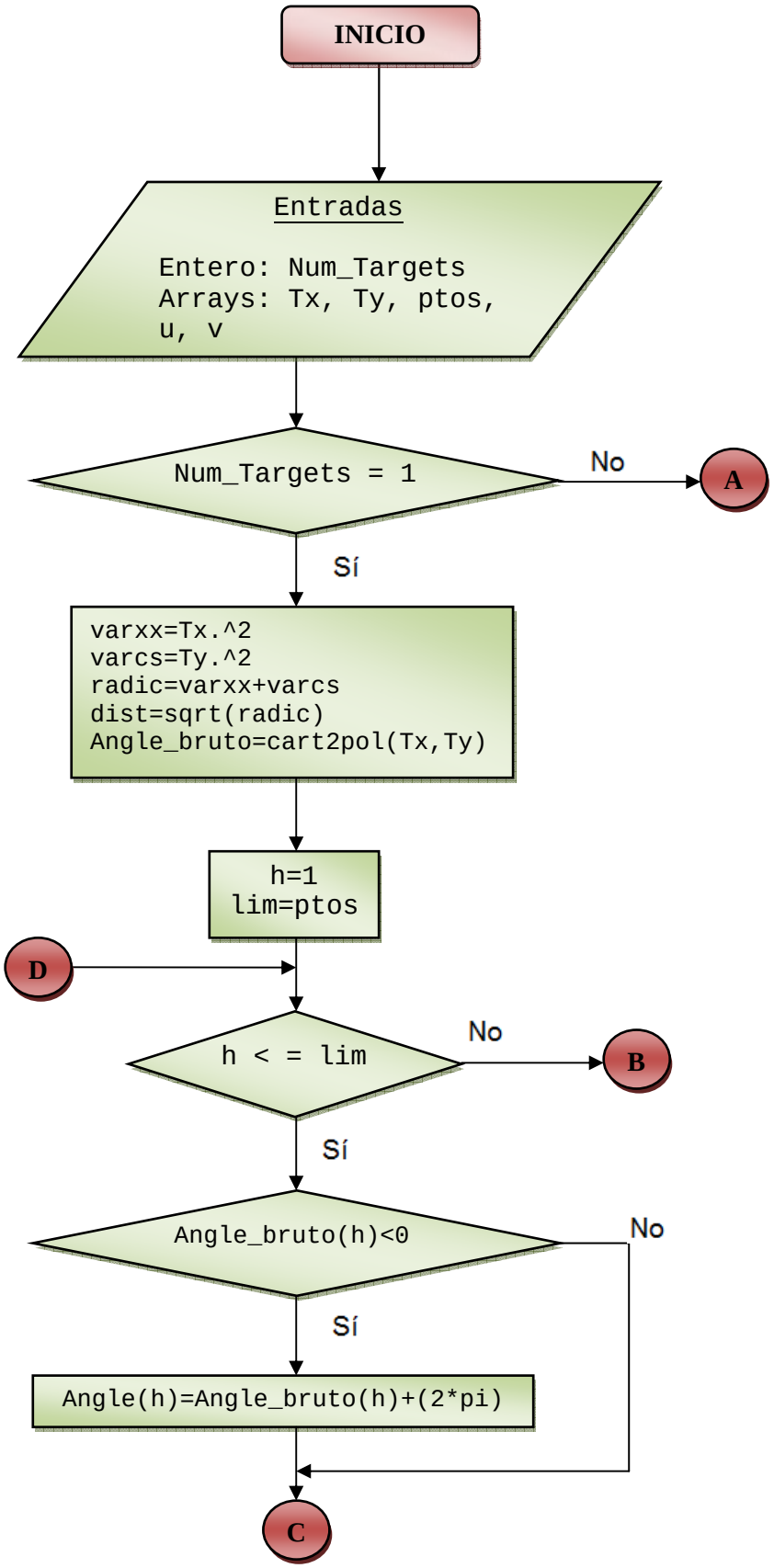
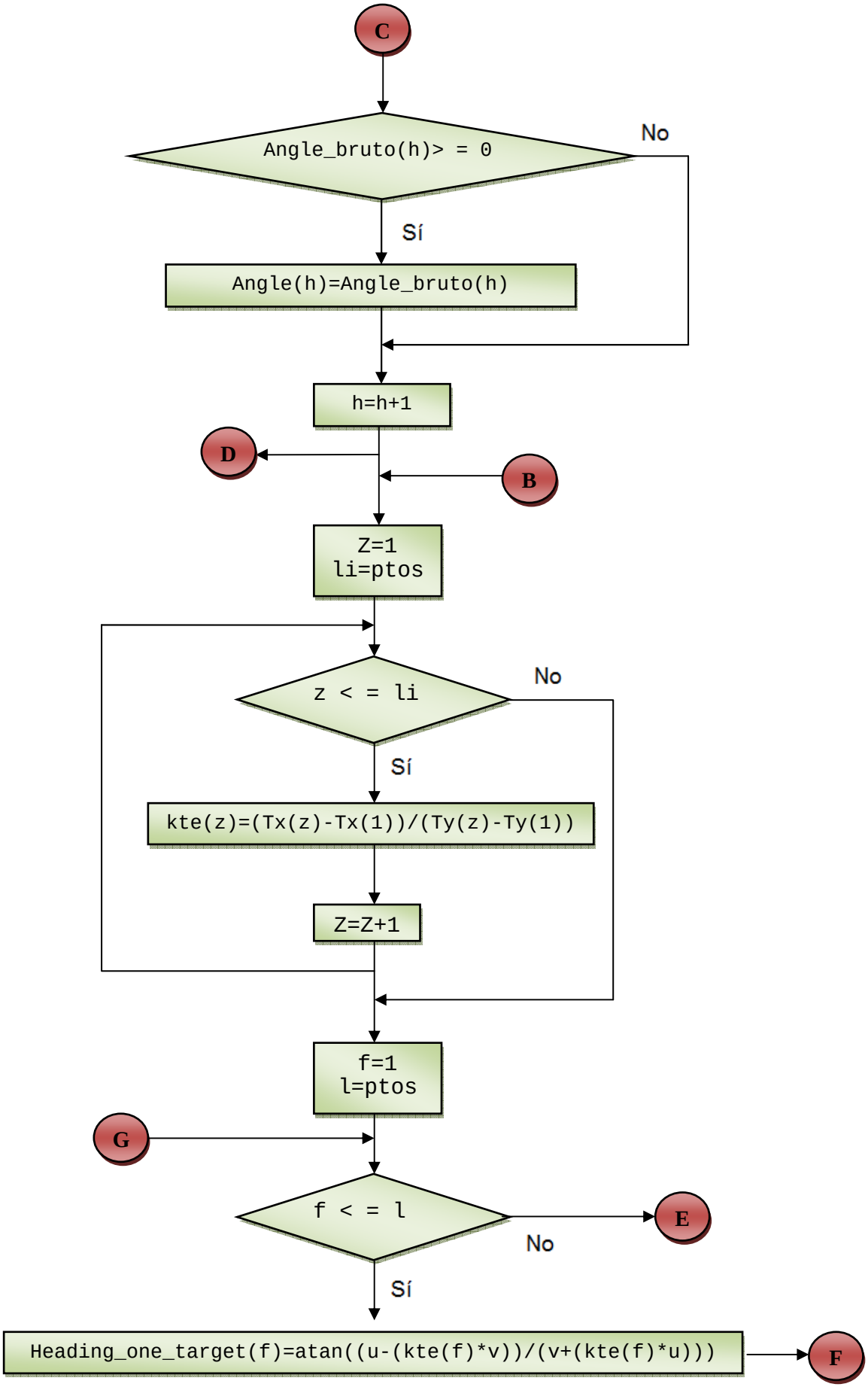
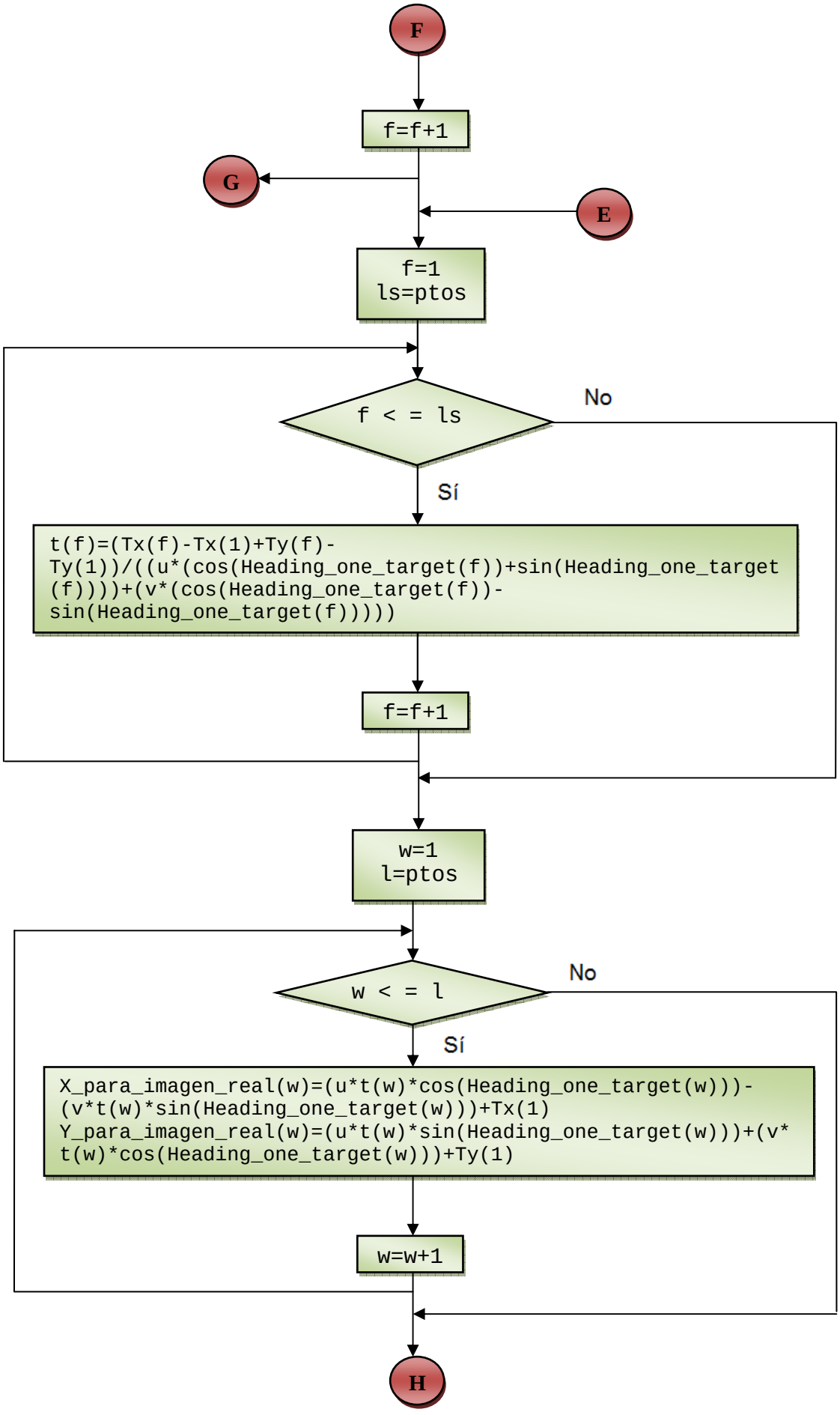
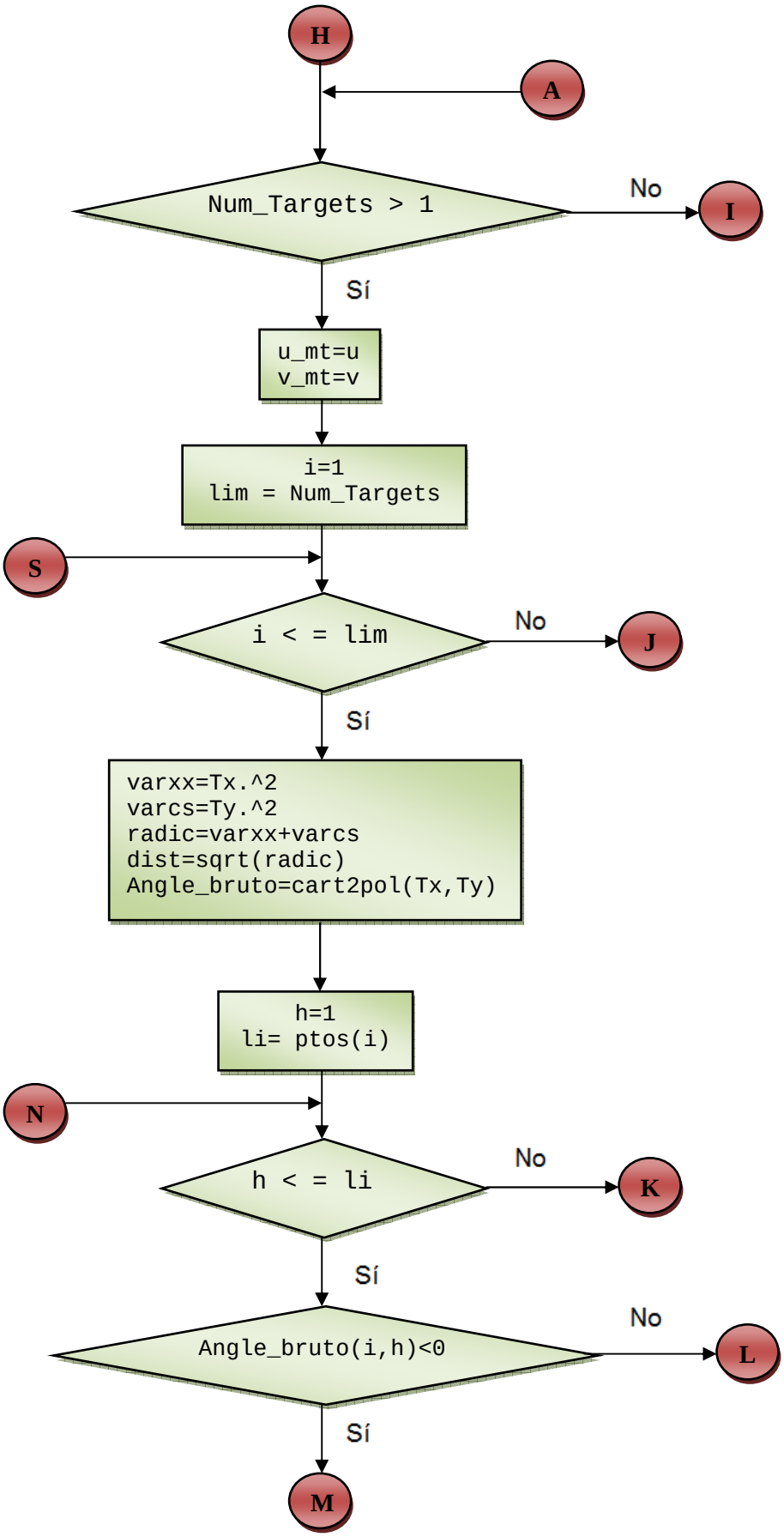

15
j < = sil
No
Sí
Tx(i,j)=Tx6(1,j)
Ty(i,j)=Ty6(1,j)
j=j+1
13
14
i=i+1
16
1
Retornar
Arrays: dist, Angle,
X_para_imagen_real,
Y_para_imagen_real
FIN

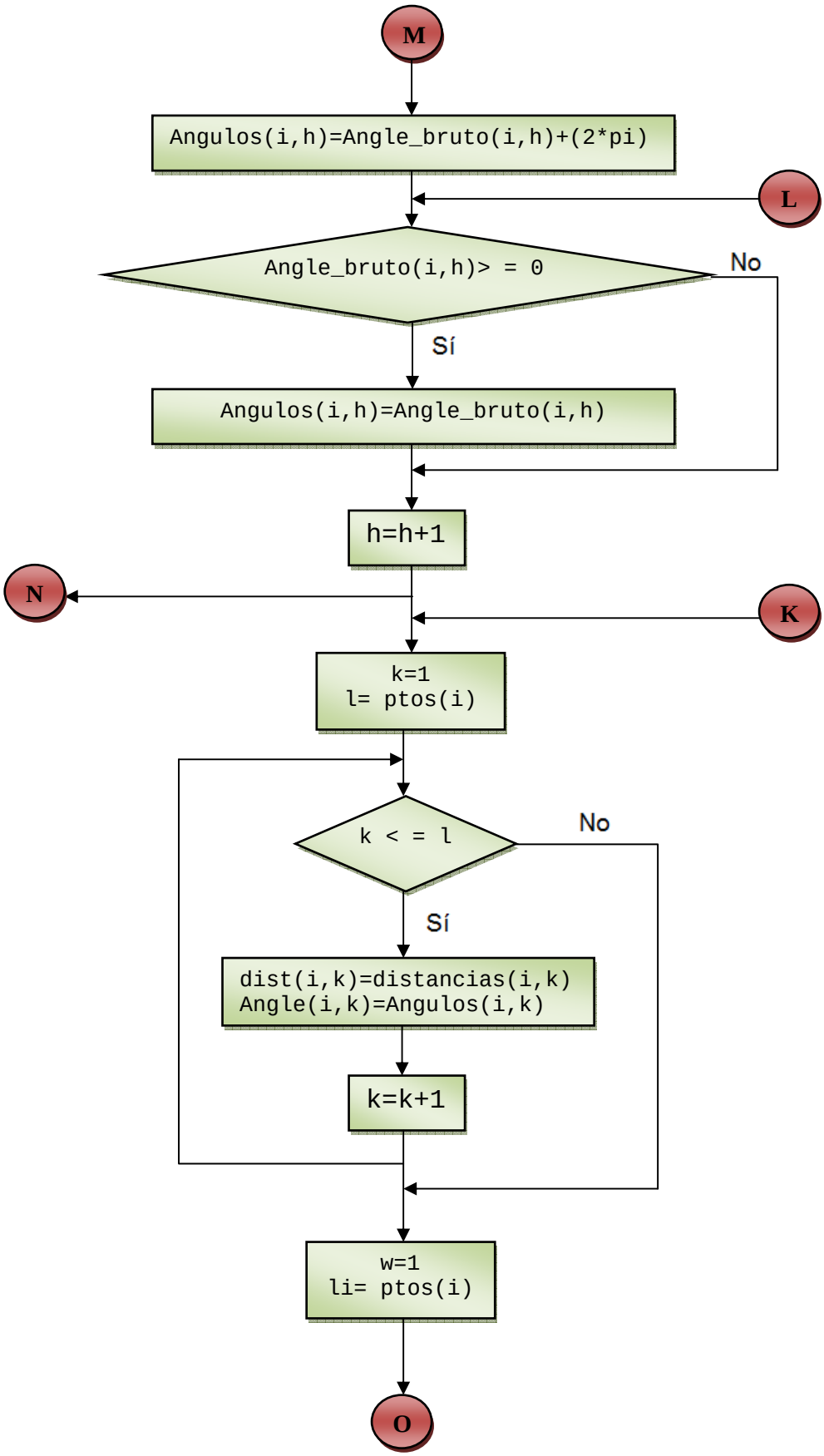
Diagrama de flujo de la función *Generar_data_para_simulacion*

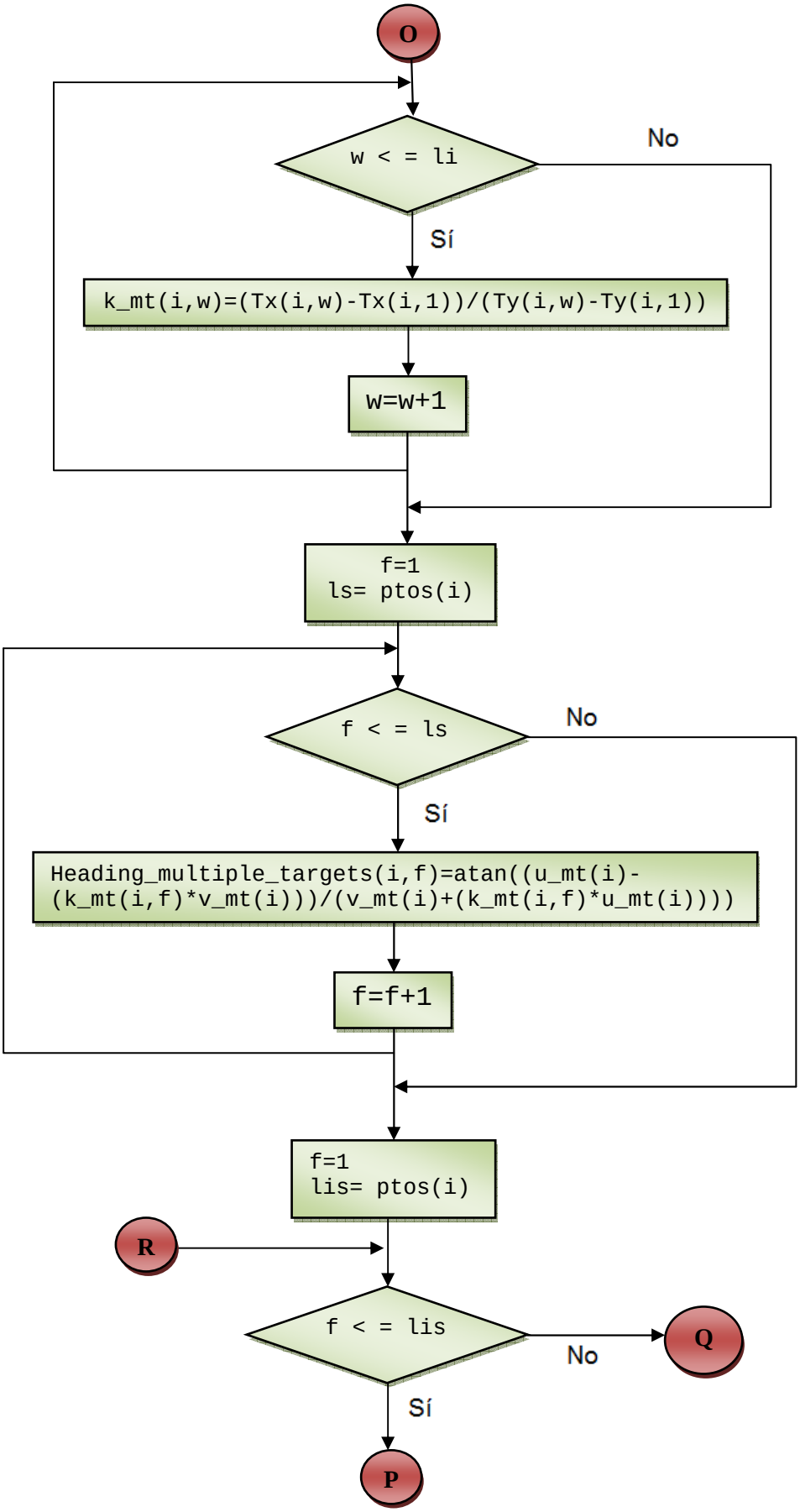


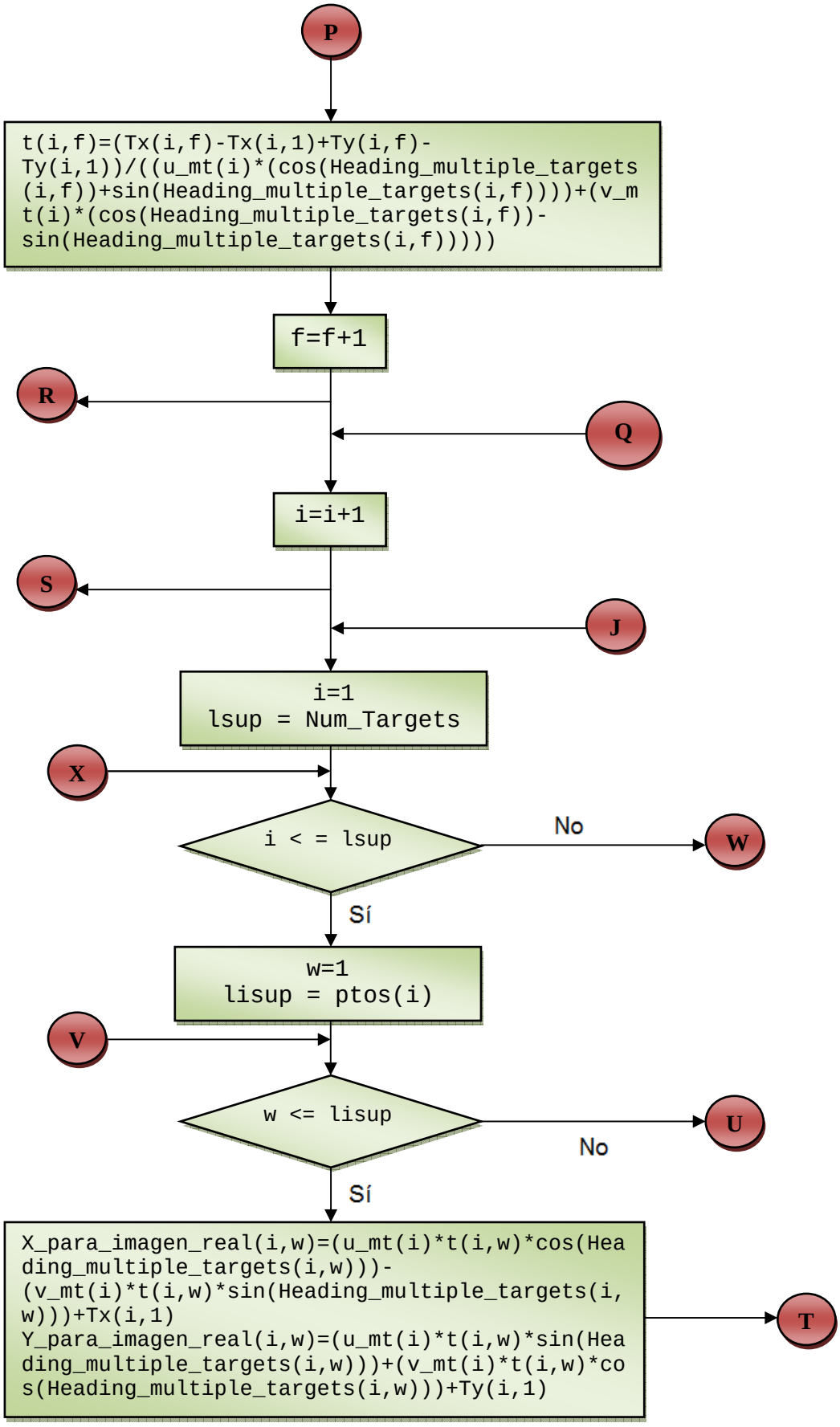

C
Angle_bruto(h)> = 0
No
Sí
Angle(h)=Angle_bruto(h)
h=h+1
D
B
Z=1
li=ptos
z < = li
No
Sí
kte(z)=(Tx(z)-Tx(1))/(Ty(z)-Ty(1))
Z=Z+1
f=1
l=ptos
G
f < = l
E
No
Sí
Heading_one_target(f)=atan((u-(kte(f)*v))/(v+(kte(f)*u)))
F


F
f=f+1
G
E
f=1
ls=ptos
f < = ls
No
Sí
t(f)=(Tx(f)-Tx(1)+Ty(f)-
Ty(1))/((u*(cos(Heading_one_target(f))+sin(Heading_one_target
(f))))+(v*(cos(Heading_one_target(f))-
sin(Heading_one_target(f)))))
f=f+1
w=1
l=ptos
w < = l
No
Sí
X_para_imagen_real(w)=(u*t(w)*cos(Heading_one_target(w)))-
(v*t(w)*sin(Heading_one_target(w)))+Tx(1)
Y_para_imagen_real(w)=(u*t(w)*sin(Heading_one_target(w)))+(v*
t(w)*cos(Heading_one_target(w)))+Ty(1)
w=w+1
H


H
A
Num_Targets > 1
No
I
Sí
u_mt=u
v_mt=v
i=1
lim = Num_Targets
S
i < = lim
No
J
Sí
varxx=Tx.^2
varcs=Ty.^2
radic=varxx+varcs
dist=sqrt(radic)
Angle_bruto=cart2pol(Tx,Ty)
h=1
li= ptos(i)
N
h < = li
No
K
Sí
Angle_bruto(i,h)<0
No
L
Sí
M


M
Angulos(i,h)=Angle_bruto(i,h)+(2*pi)
L
Angle_bruto(i,h)> = 0
No
Sí
Angulos(i,h)=Angle_bruto(i,h)
h=h+1
N
K
k=1
l= ptos(i)
k < = l
No
Sí
dist(i,k)=distancias(i,k)
Angle(i,k)=Angulos(i,k)
k=k+1
w=1
li= ptos(i)
O


O
w < = li
No
Sí
k_mt(i,w)=(Tx(i,w)-Tx(i,1))/(Ty(i,w)-Ty(i,1))
w=w+1
f=1
ls= ptos(i)
f < = ls
No
Sí
Heading_multiple_targets(i,f)=atan((u_mt(i)-
(k_mt(i,f)*v_mt(i)))/(v_mt(i)+(k_mt(i,f)*u_mt(i))))
f=f+1
f=1
lis= ptos(i)
R
f < = lis
No
Q
Sí
P


P
t(i,f)=(Tx(i,f)-Tx(i,1)+Ty(i,f)-
Ty(i,1))/((u_mt(i)*(cos(Heading_multiple_targets
(i,f))+sin(Heading_multiple_targets(i,f))))+(v_m
t(i)*(cos(Heading_multiple_targets(i,f))-
sin(Heading_multiple_targets(i,f)))))
f=f+1
R
Q
i=i+1
S
J
i=1
lsup = Num_Targets
X
i < = lsup
No
W
Sí
w=1
lisup = ptos(i)
V
w <= lisup
U
No
Sí
X_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*cos(Hea
ding_multiple_targets(i,w)))-
(v_mt(i)*t(i,w)*sin(Heading_multiple_targets(i,
w)))+Tx(i,1)
Y_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*sin(Hea
ding_multiple_targets(i,w)))+(v_mt(i)*t(i,w)*co
s(Heading_multiple_targets(i,w)))+Ty(i,1)
T

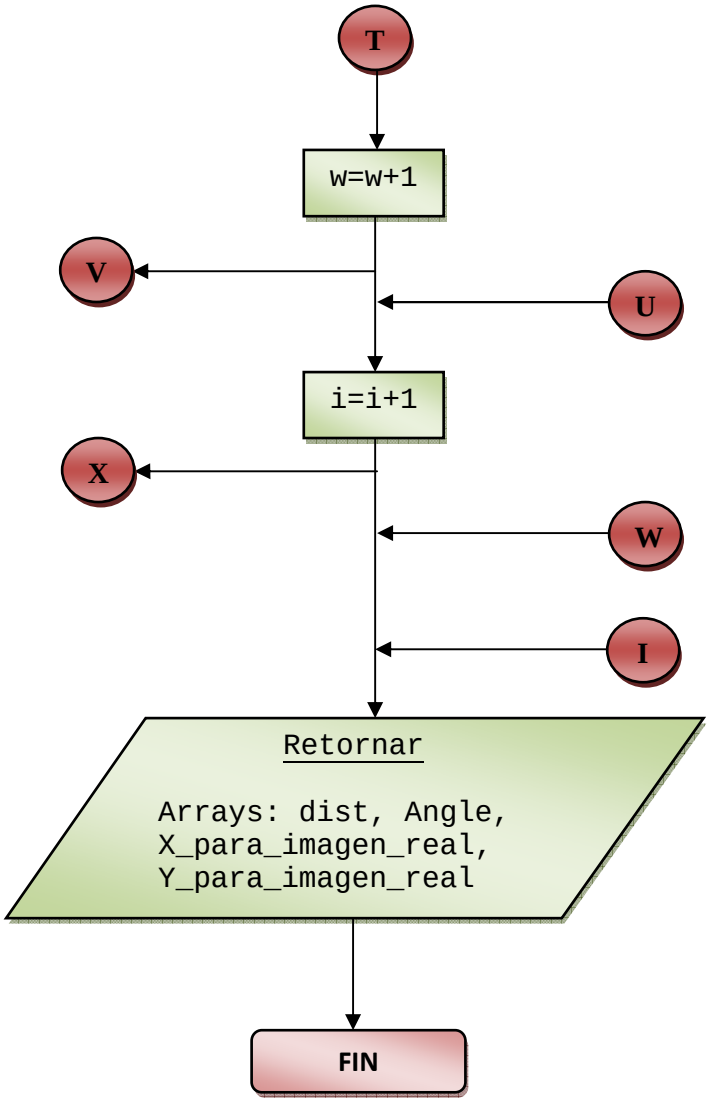
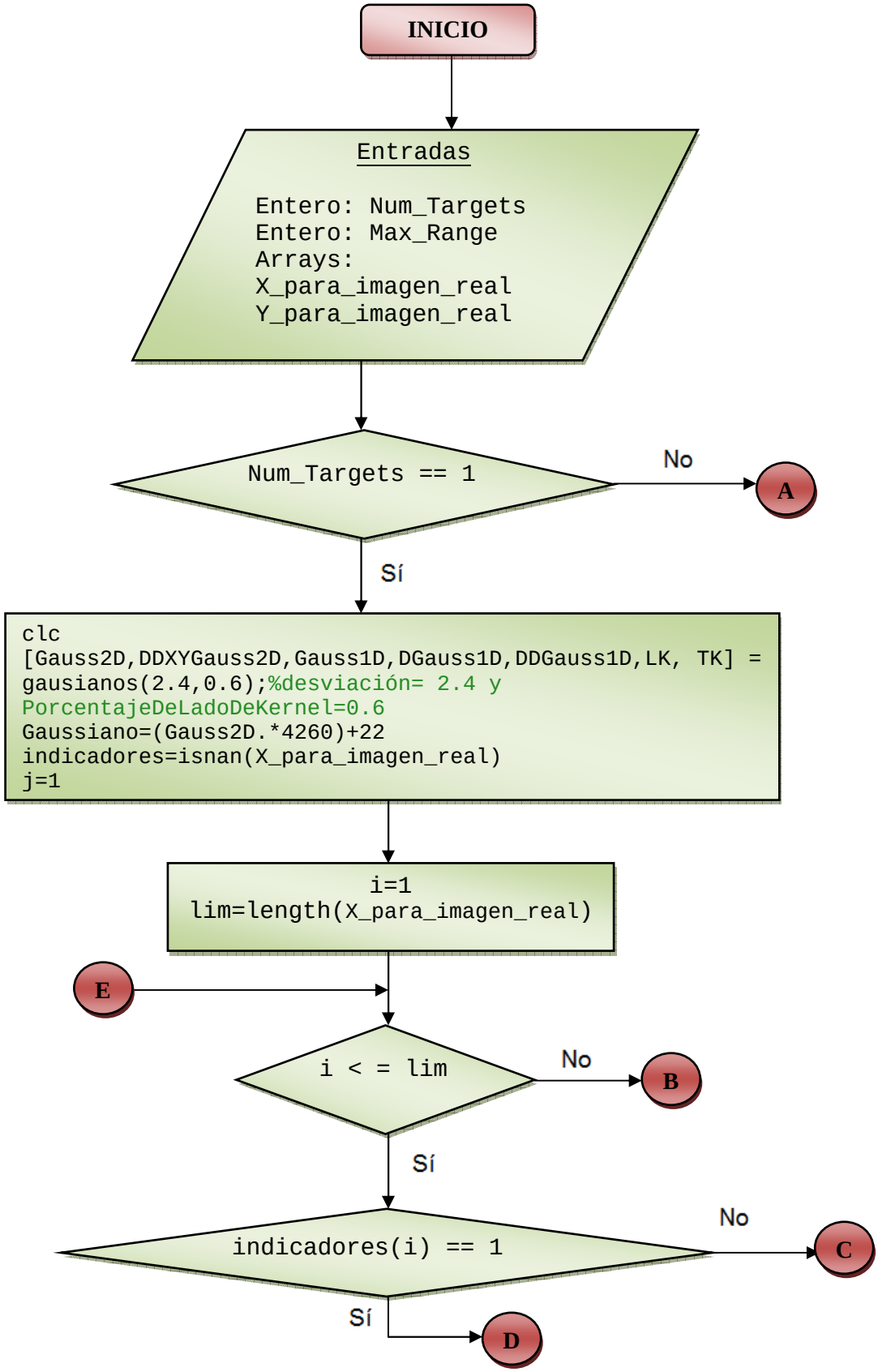
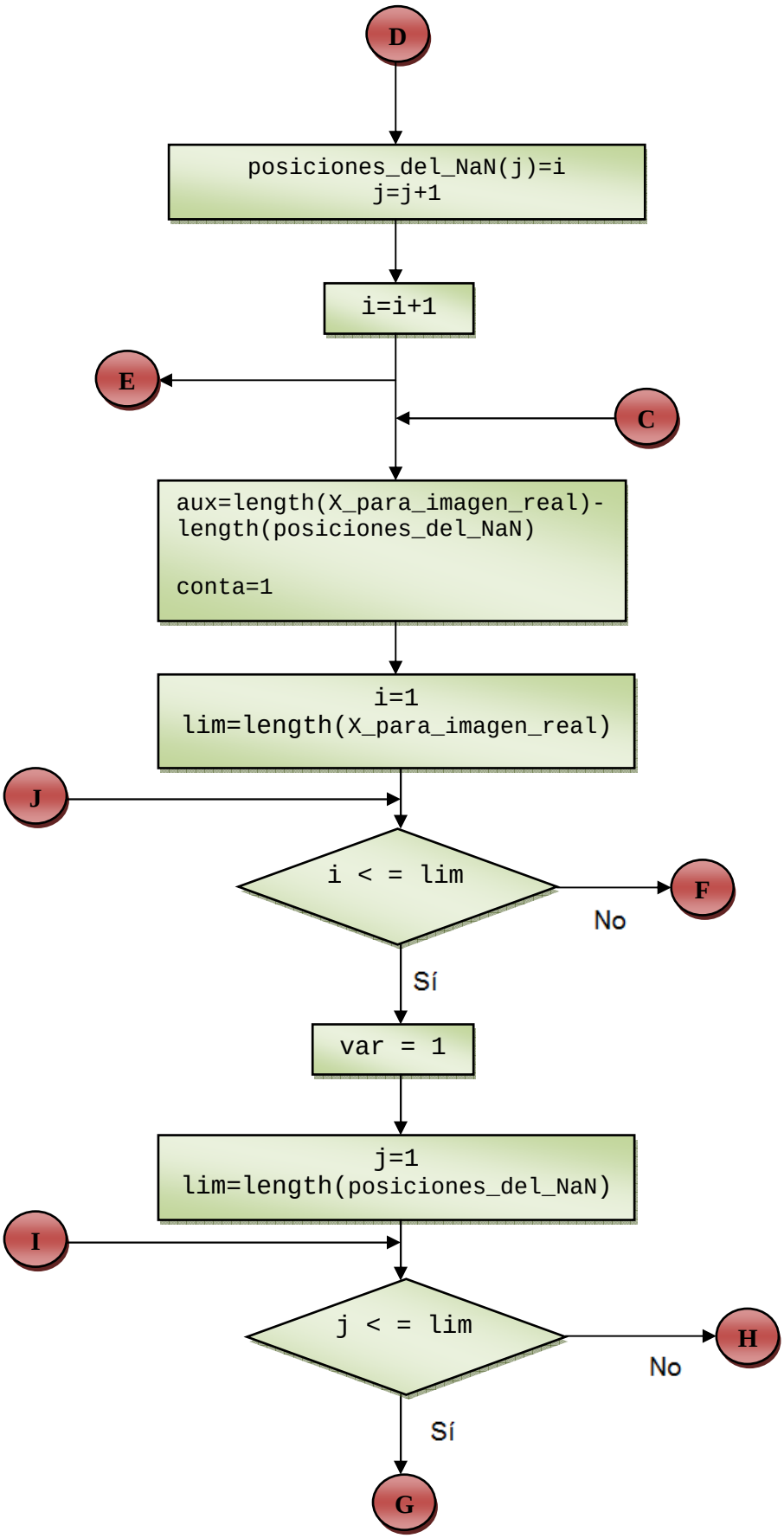
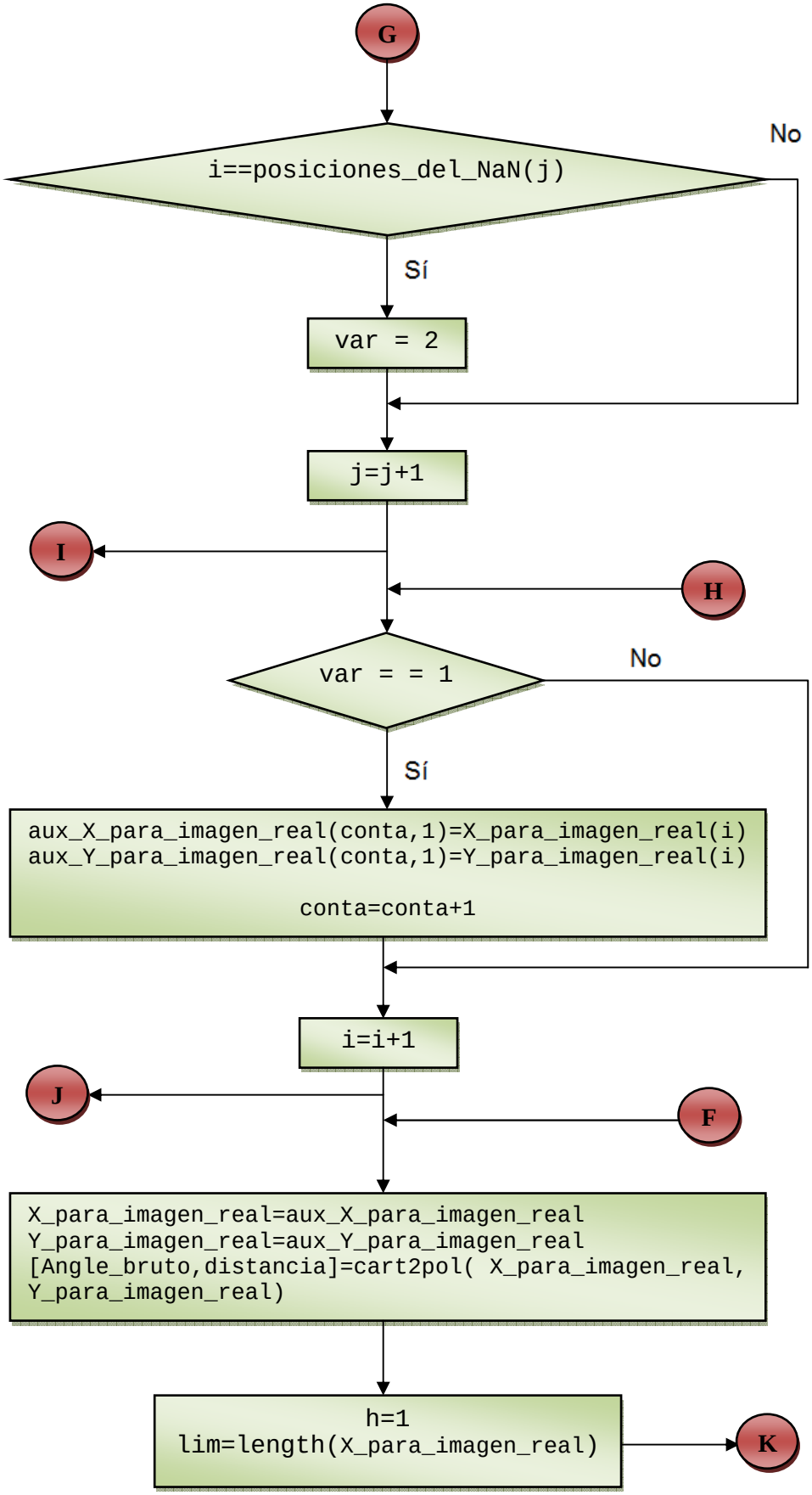
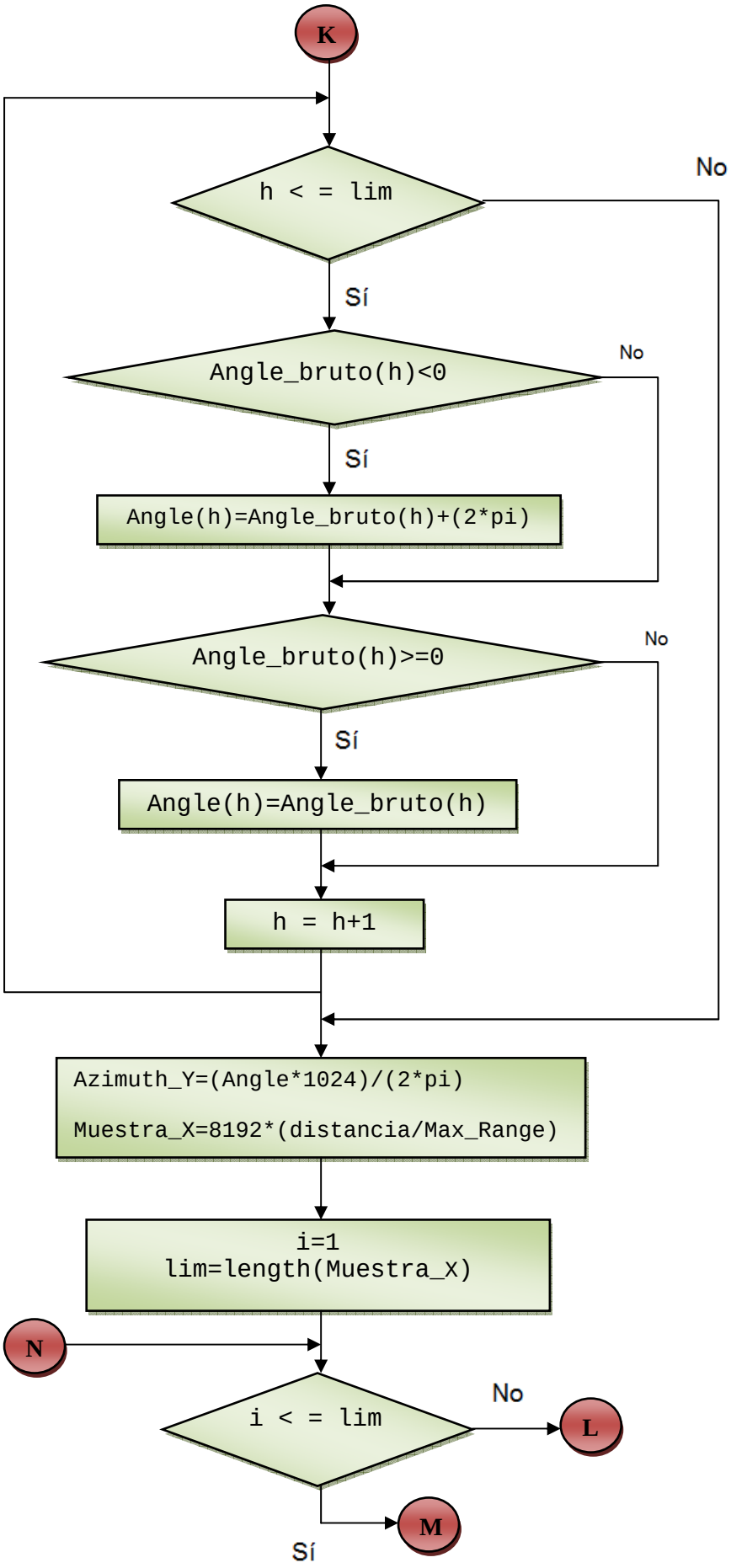

T
w=w+1
V
U
i=i+1
X
W
I
Retornar
Arrays: dist, Angle,
X_para_imagen_real,
Y_para_imagen_real
FIN

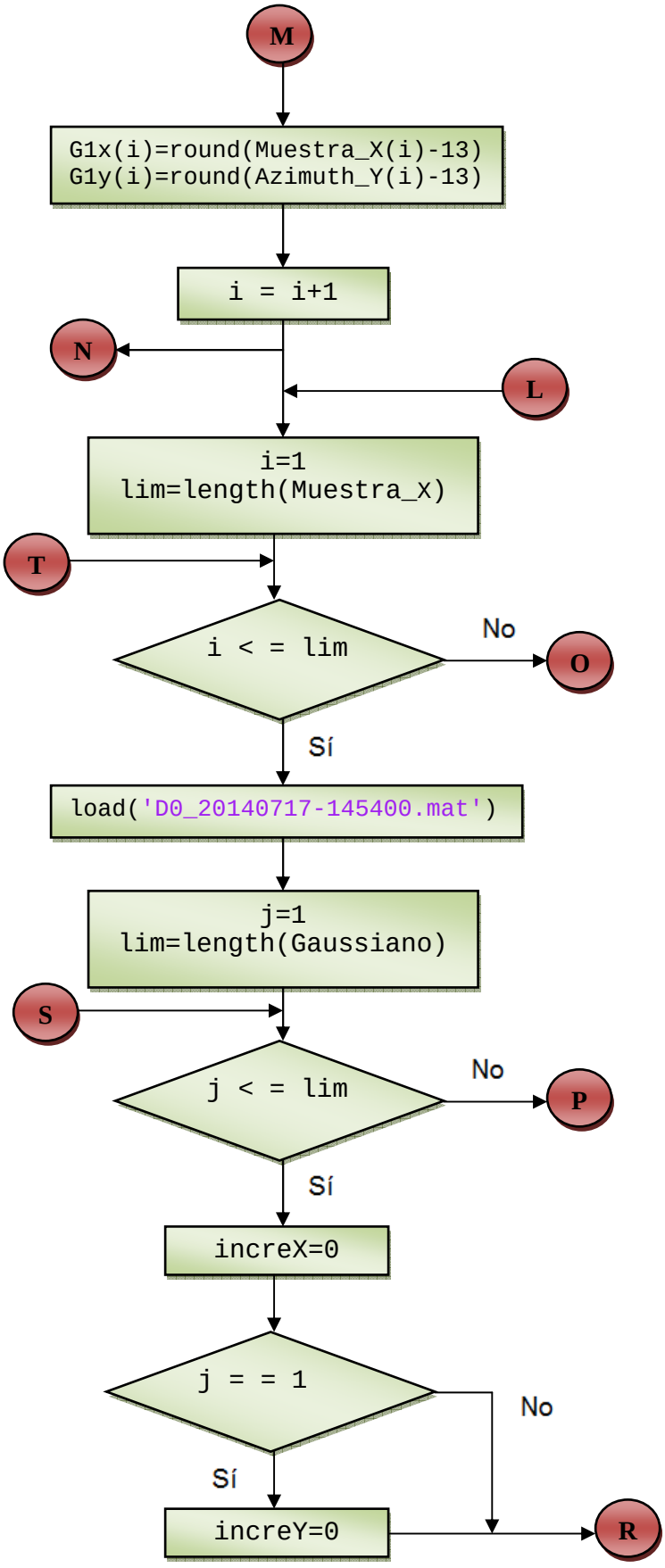
Diagrama de flujo de la función *Cargar_Data_Sobre_Imagen_Real*

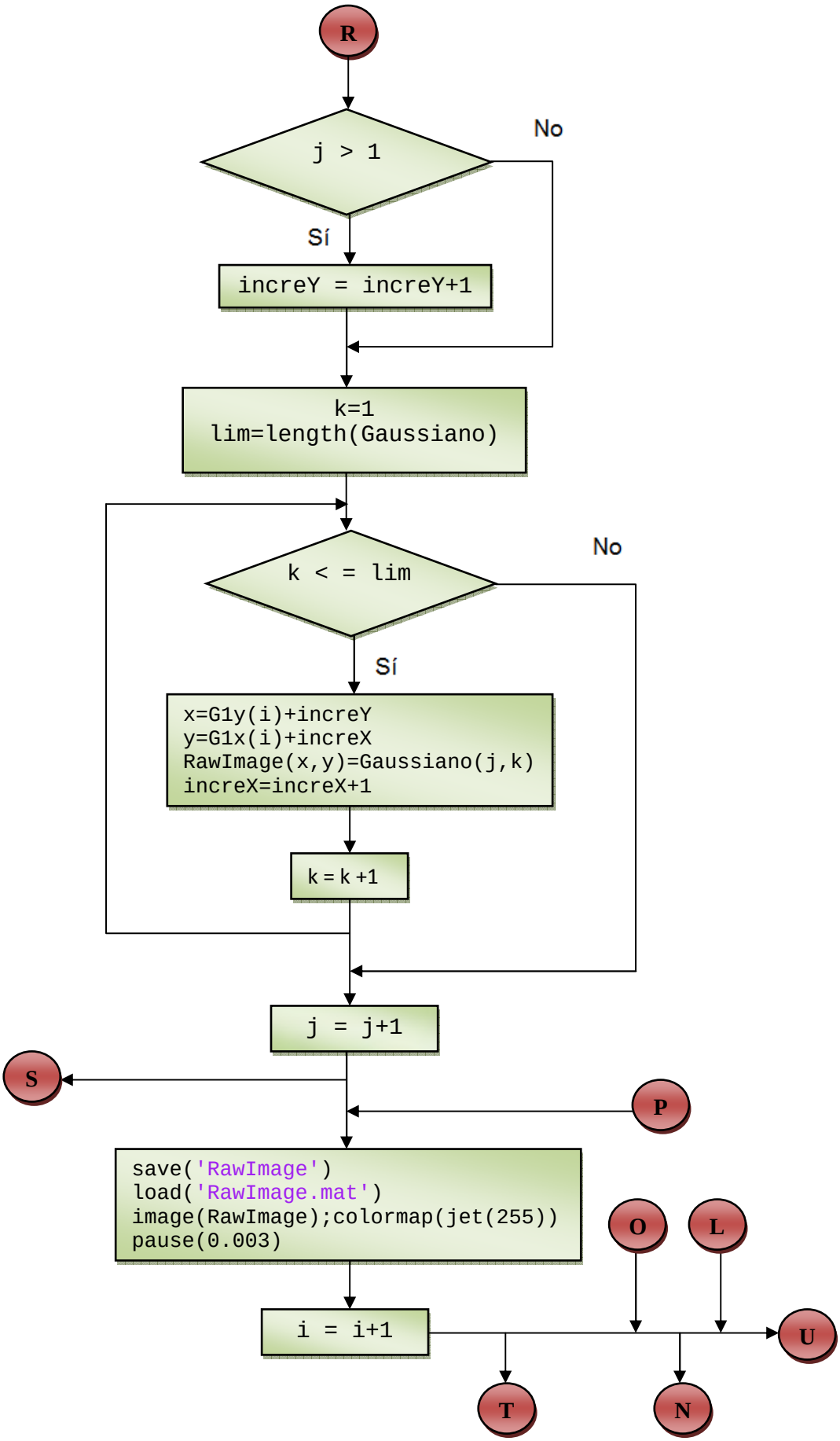


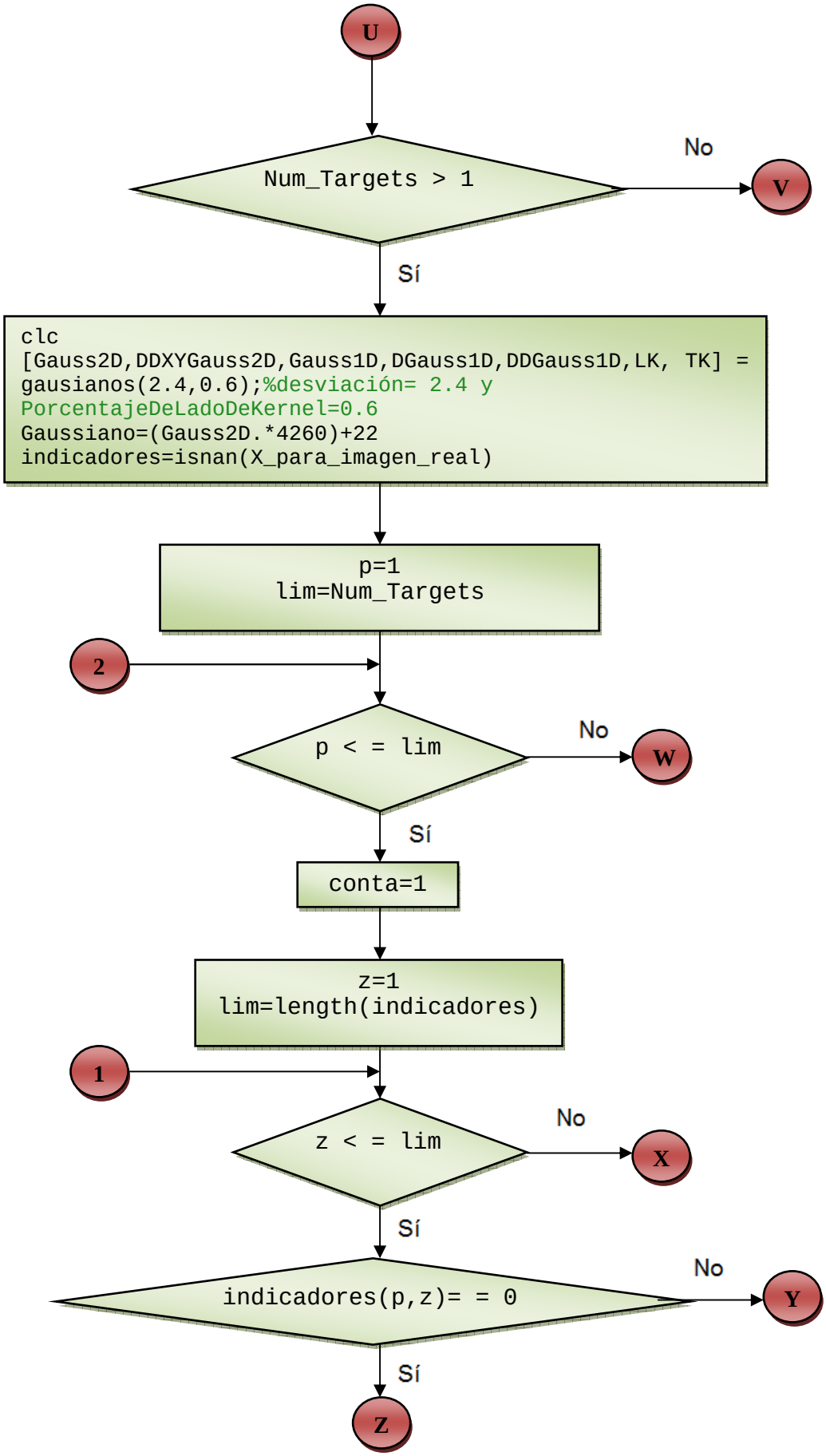

D
posiciones_del_NaN(j)=i
j=j+1
i=i+1
E
C
aux=length(X_para_imagen_real)-
length(posiciones_del_NaN)
conta=1
i=1
lim=length(X_para_imagen_real)
J
i < = lim
F
No
Sí
var = 1
j=1
lim=length(posiciones_del_NaN)
I
j < = lim
H
No
Sí
G

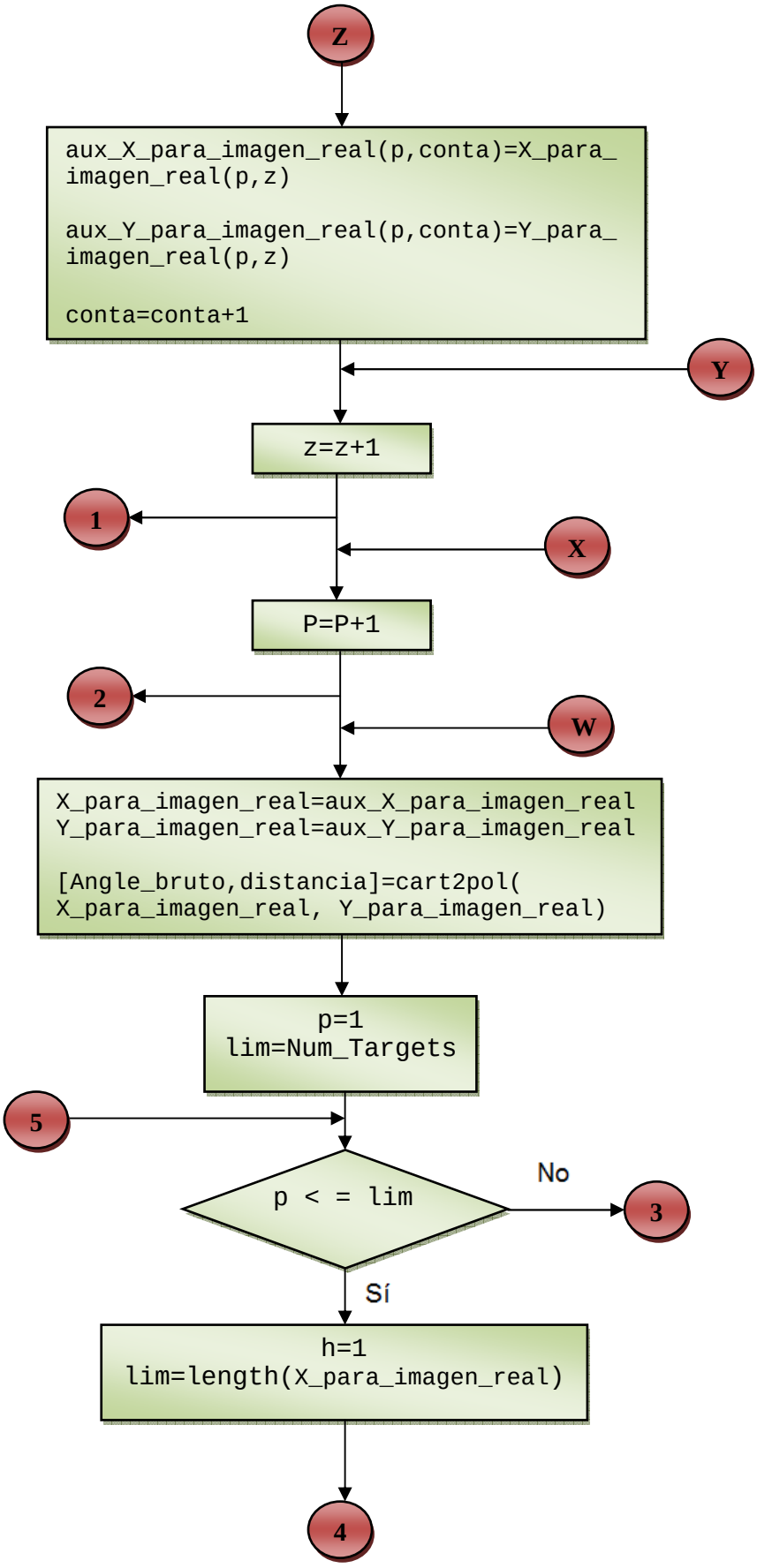

G
i==posiciones_del_NaN(j)
No
Sí
var = 2
j=j+1
I
H
var = = 1
No
Sí
aux_X_para_imagen_real(conta,1)=X_para_imagen_real(i)
aux_Y_para_imagen_real(conta,1)=Y_para_imagen_real(i)
conta=conta+1
i=i+1
J
F
X_para_imagen_real=aux_X_para_imagen_real
Y_para_imagen_real=aux_Y_para_imagen_real
[Angle_bruto,distancia]=cart2pol(X_para_imagen_real,
Y_para_imagen_real)
h=1
lim=length(X_para_imagen_real)
K

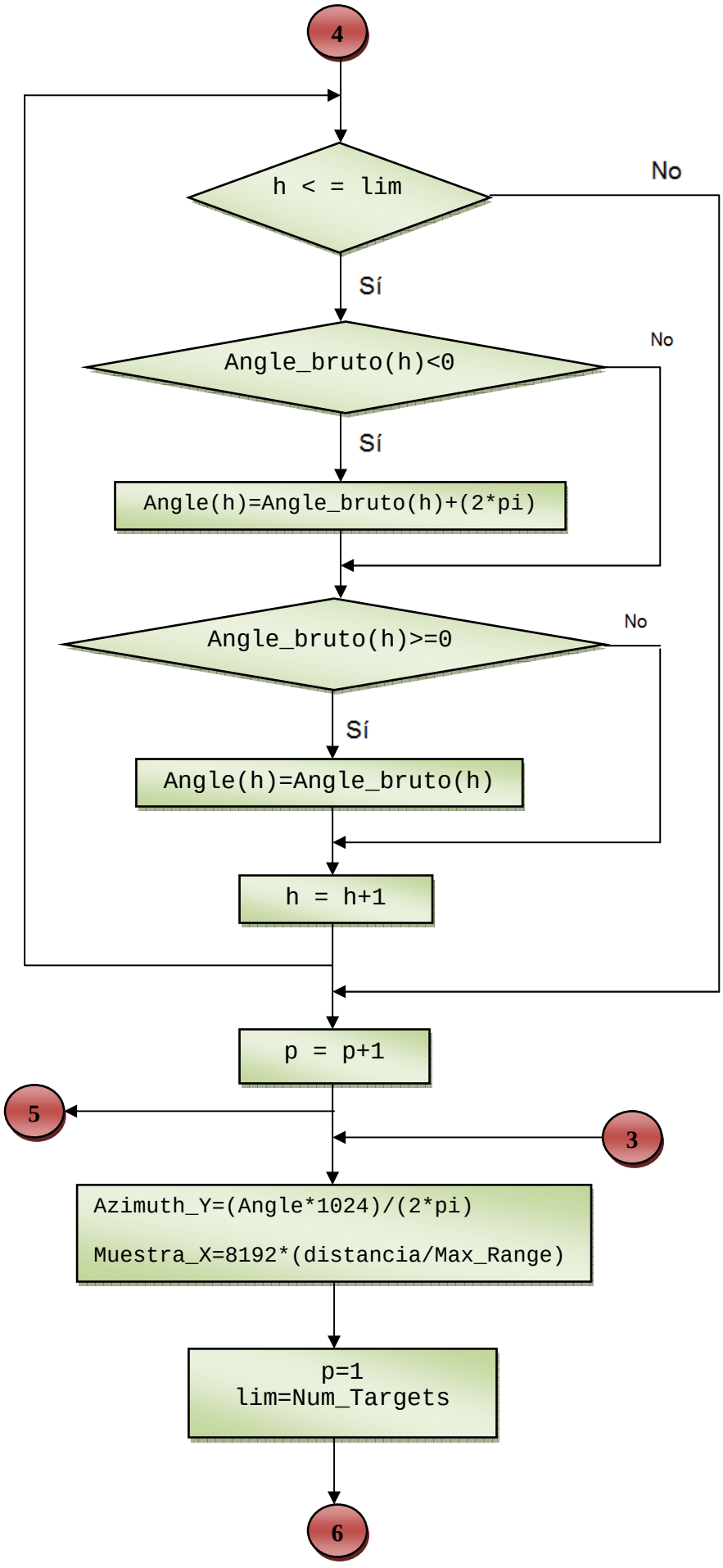

K
h < = lim
No
Sí
Angle_bruto(h)<0
No
Sí
Angle(h)=Angle_bruto(h)+(2*pi)
Angle_bruto(h)>=0
No
Sí
Angle(h)=Angle_bruto(h)
h = h+1
Azimuth_Y=(Angle*1024)/(2*pi)
Muestra_X=8192*(distancia/Max_Range)
i=1
lim=length(Muestra_X)
N
i < = lim
No
L
M
Sí

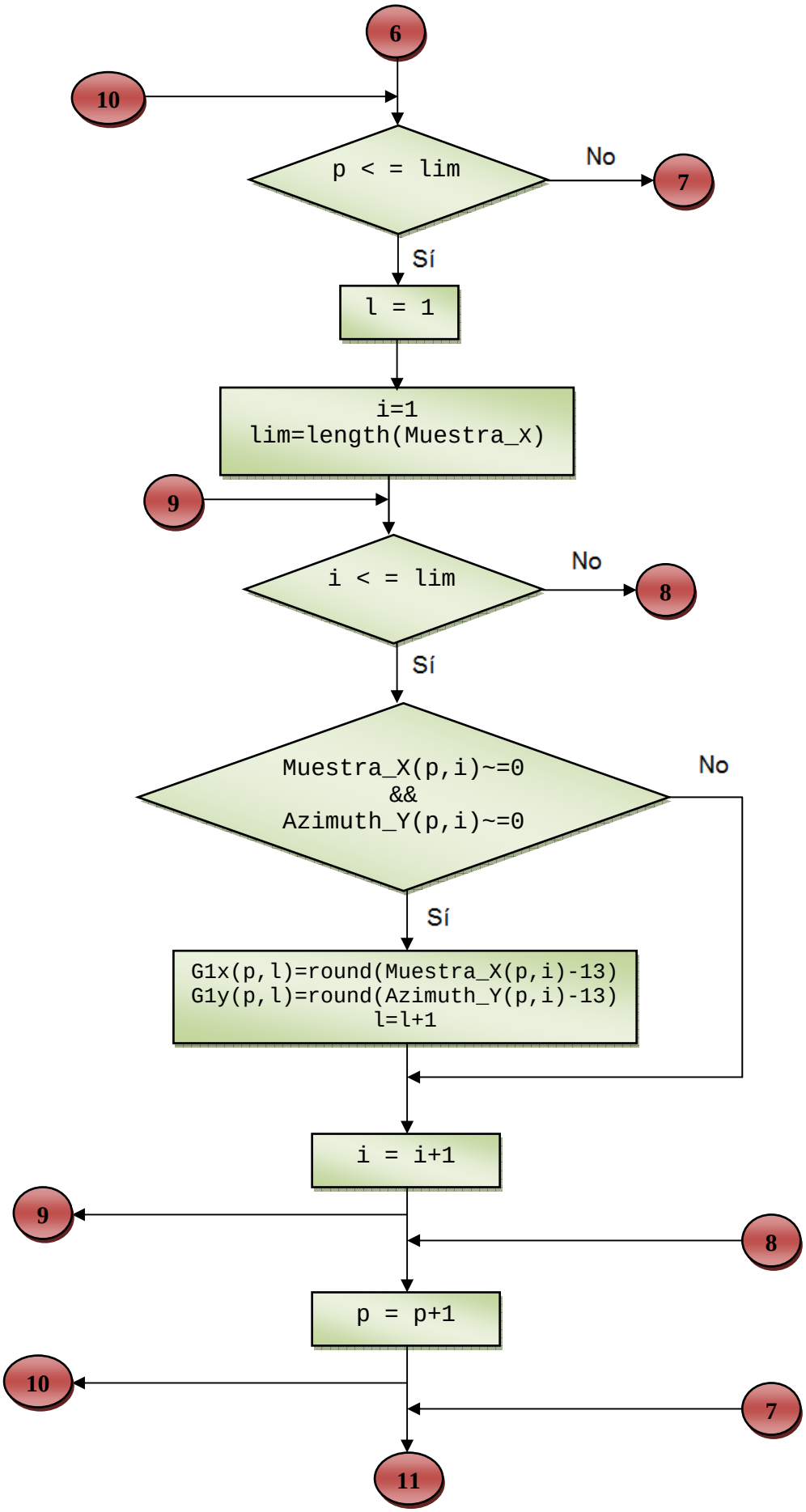

M
G1x(i)=round(Muestra_X(i)-13)
G1y(i)=round(Azimuth_Y(i)-13)
i = i+1
N
L
i=1
lim=length(Muestra_X)
T
i < = lim
No
O
Sí
load('D0_20140717-145400.mat')
j=1
lim=length(Gaussiano)
S
j < = lim
No
P
Sí
increX=0
j = = 1
No
Sí
increY=0
R

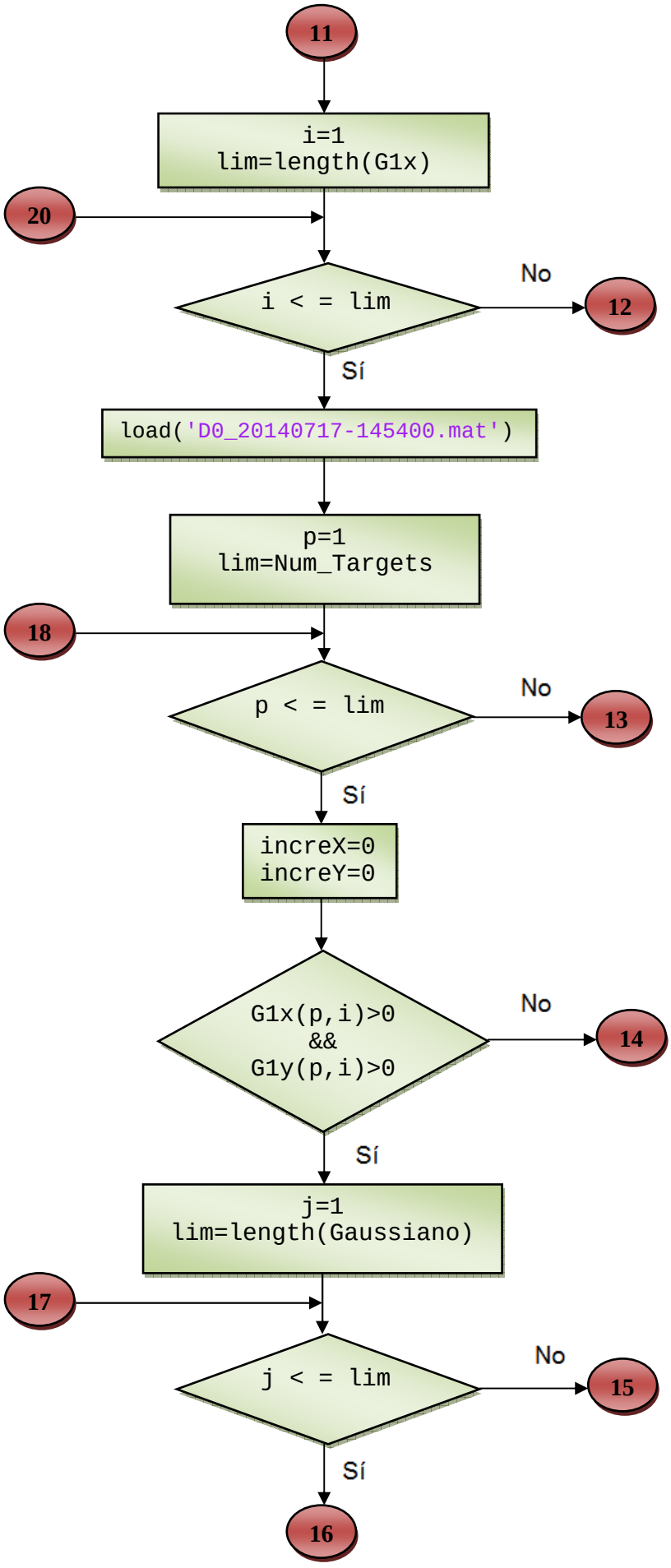

R
j > 1
No
Sí
increY = increY+1
k=1
lim=length(Gaussiano)
k < = lim
No
Sí
x=G1y(i)+increY
y=G1x(i)+increX
RawImage(x,y)=Gaussiano(j,k)
increX=increX+1
k = k +1
j = j+1
S
P
save('RawImage')
load('RawImage.mat')
image(RawImage);colormap(jet(255))
pause(0.003)
O
L
i = i+1
U
T
N

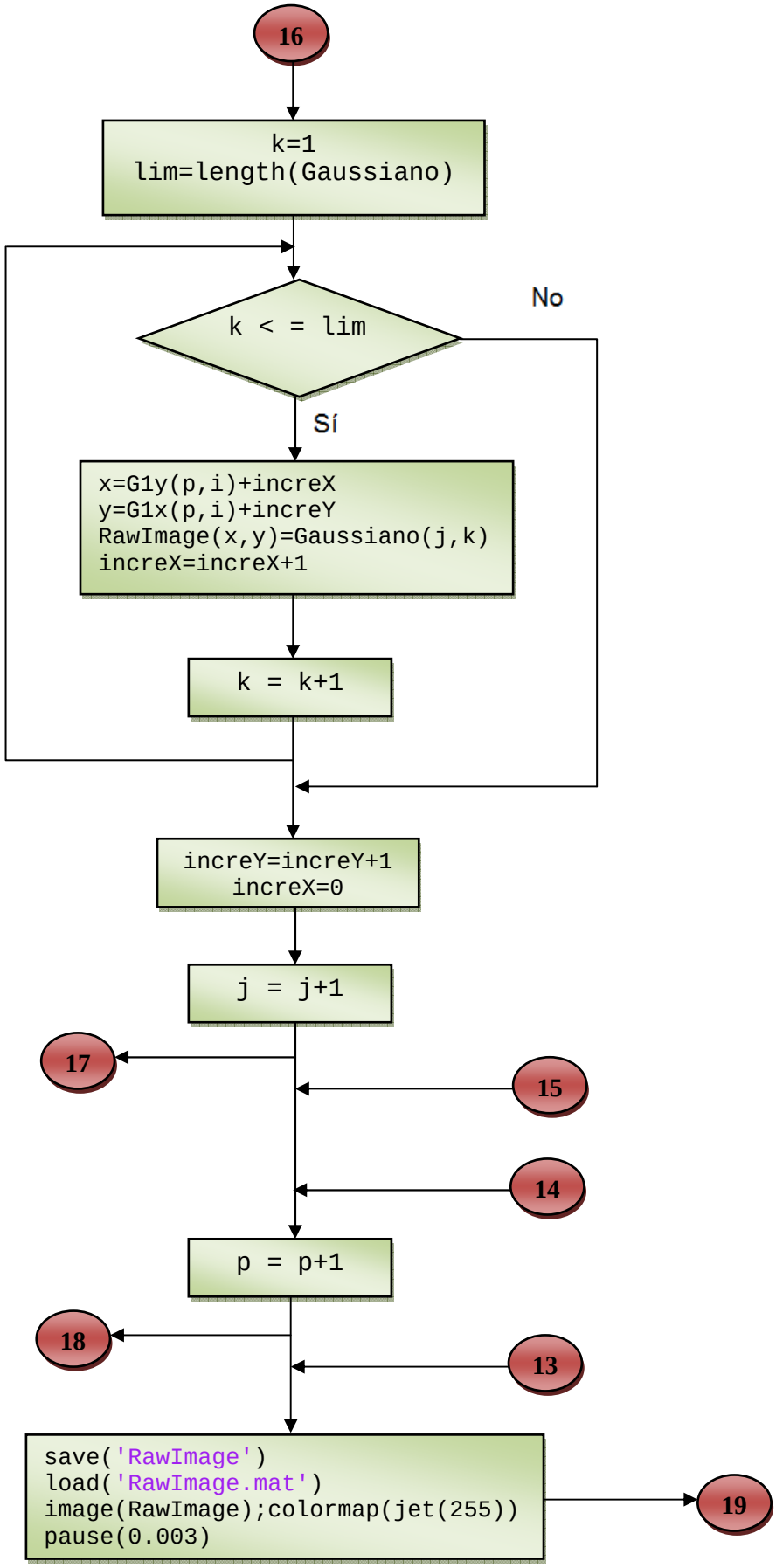

U
Num_Targets > 1
No
V
Sí
clc
[Gauss2D,DDXYGauss2D,Gauss1D,DGauss1D,DDGauss1D,LK, TK] =
gausianos(2.4,0.6);%desviación= 2.4 y
PorcentajeDeLadoDeKernel=0.6
Gaussiano=(Gauss2D.*4260)+22
indicadores=isnan(X_para_imagen_real)
p=1
lim=Num_Targets
2
p < = lim
No
W
Sí
conta=1
z=1
lim=length(indicadores)
1
z < = lim
No
X
Sí
indicadores(p,z)= = 0
No
Y
Sí
Z


Z
aux_X_para_imagen_real(p,conta)=X_para_
imagen_real(p,z)
aux_Y_para_imagen_real(p,conta)=Y_para_
imagen_real(p,z)
conta=conta+1
Y
z=z+1
1
X
P=P+1
2
W
X_para_imagen_real=aux_X_para_imagen_real
Y_para_imagen_real=aux_Y_para_imagen_real
[Angle_bruto,distancia]=cart2pol(
X_para_imagen_real, Y_para_imagen_real)
p=1
lim=Num_Targets
5
p < = lim
No
3
Sí
h=1
lim=length(X_para_imagen_real)
4


4
h < = lim
No
Sí
Angle_bruto(h)<0
No
Sí
Angle(h)=Angle_bruto(h)+(2*pi)
Angle_bruto(h)>=0
No
Sí
Angle(h)=Angle_bruto(h)
h = h+1
p = p+1
5
3
Azimuth_Y=(Angle*1024)/(2*pi)
Muestra_X=8192*(distancia/Max_Range)
p=1
lim=Num_Targets
6


6
10
p < = lim
No
7
Sí
l = 1
i=1
lim=length(Muestra_X)
9
i < = lim
No
8
Sí
Muestra_X(p,i)~=0
&&
Azimuth_Y(p,i)~=0
No
Sí
G1x(p,l)=round(Muestra_X(p,i)-13)
G1y(p,l)=round(Azimuth_Y(p,i)-13)
l=l+1
i = i+1
9
8
p = p+1
10
7
11


11
i=1
lim=length(G1x)
20
i < = lim
No
12
Sí
load('D0_20140717-145400.mat')
p=1
lim=Num_Targets
18
p < = lim
No
13
Sí
increX=0
increY=0
G1x(p,i)>0
&&
G1y(p,i)>0
No
14
Sí
j=1
lim=length(Gaussiano)
17
j < = lim
No
15
Sí
16


16
k=1
lim=length(Gaussiano)
k < = lim
No
Sí
x=G1y(p,i)+increX
y=G1x(p,i)+increY
RawImage(x,y)=Gaussiano(j,k)
increX=increX+1
k = k+1
increY=increY+1
increX=0
j = j+1
17
15
14
p = p+1
18
13
save('RawImage')
load('RawImage.mat')
image(RawImage);colormap(jet(255))
pause(0.003)
19

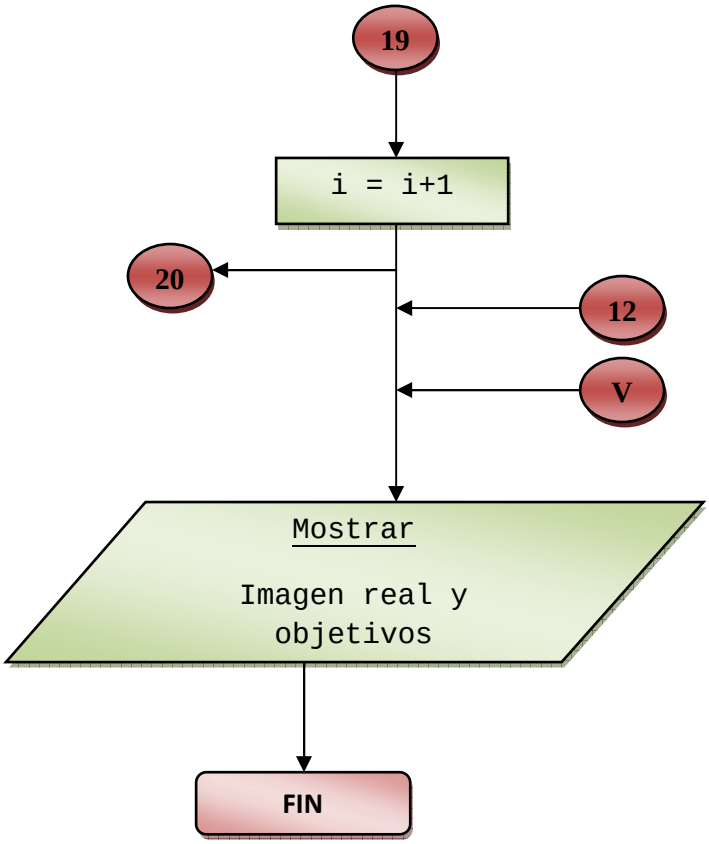
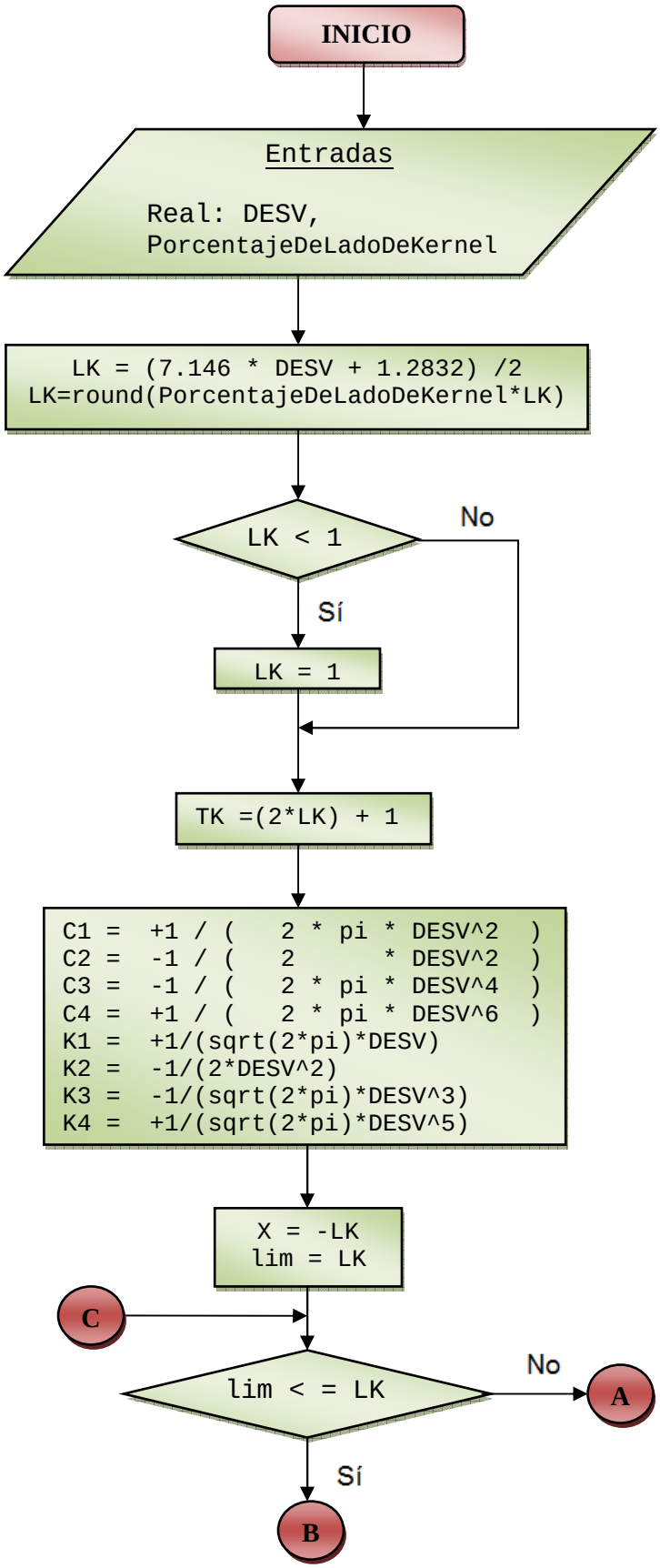

19
i = i+1
20
12
V
Mostrar
Imagen real y objetivos
FIN

Diagrama de flujo de la función *Gausianos*



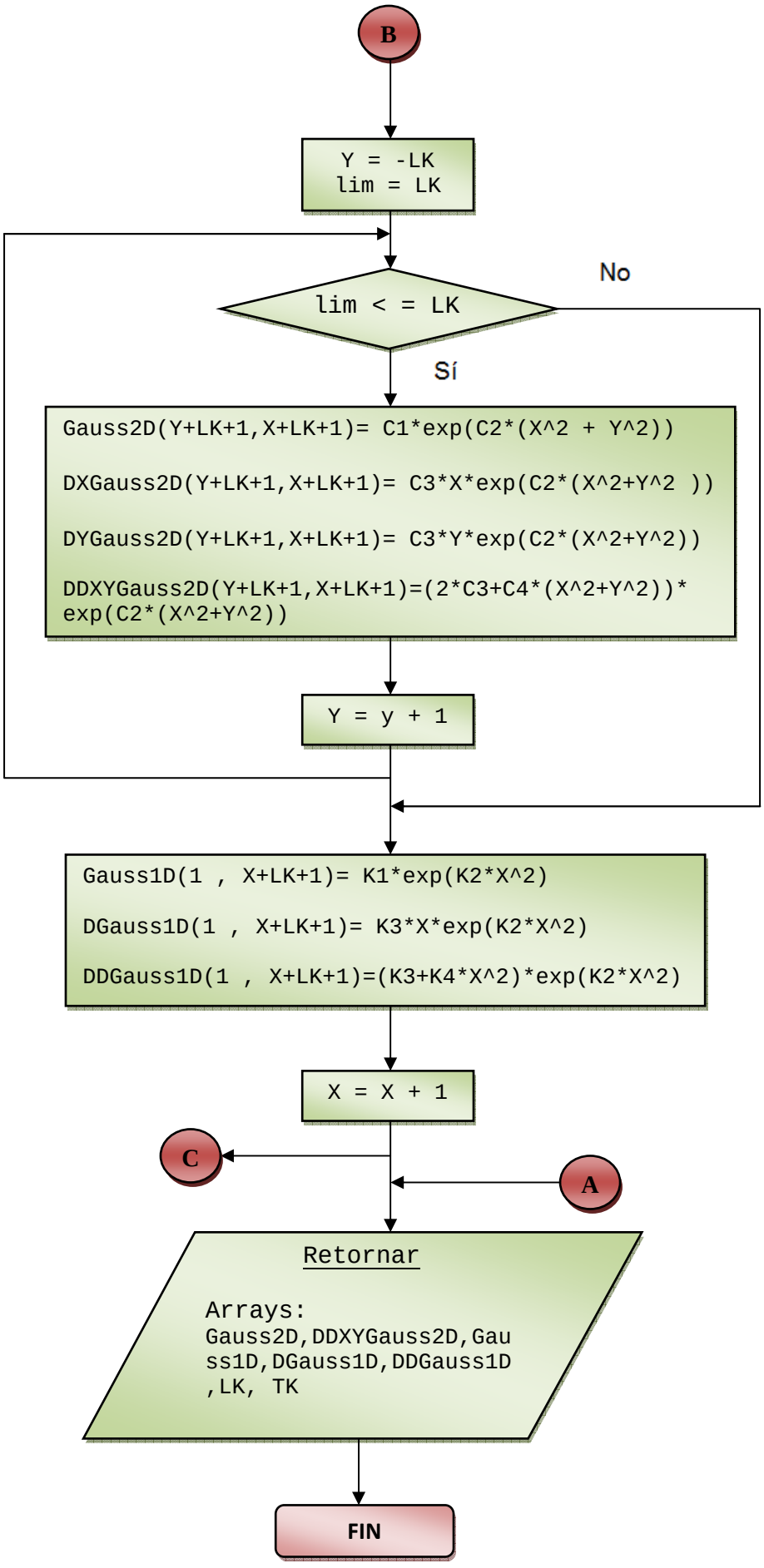
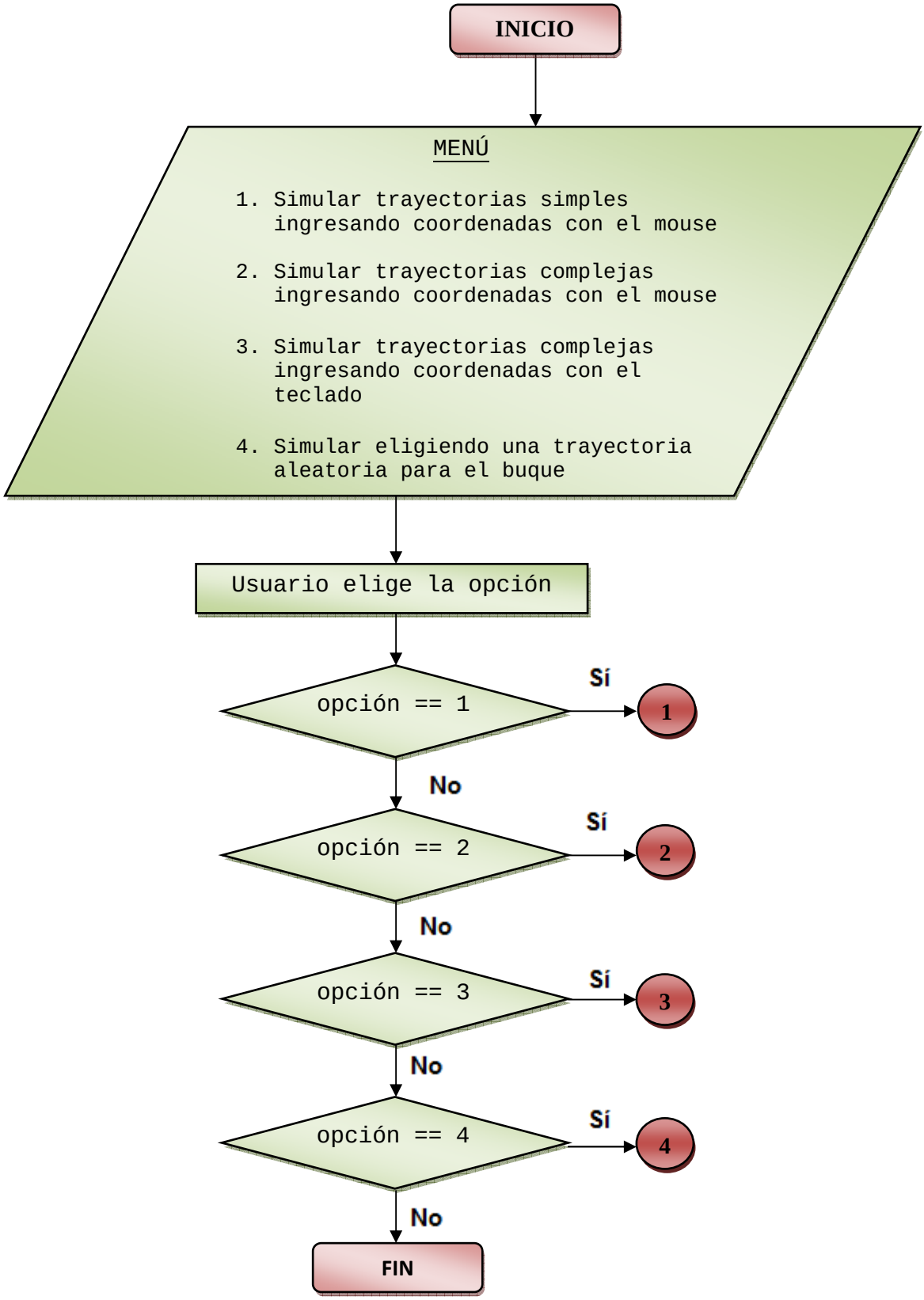
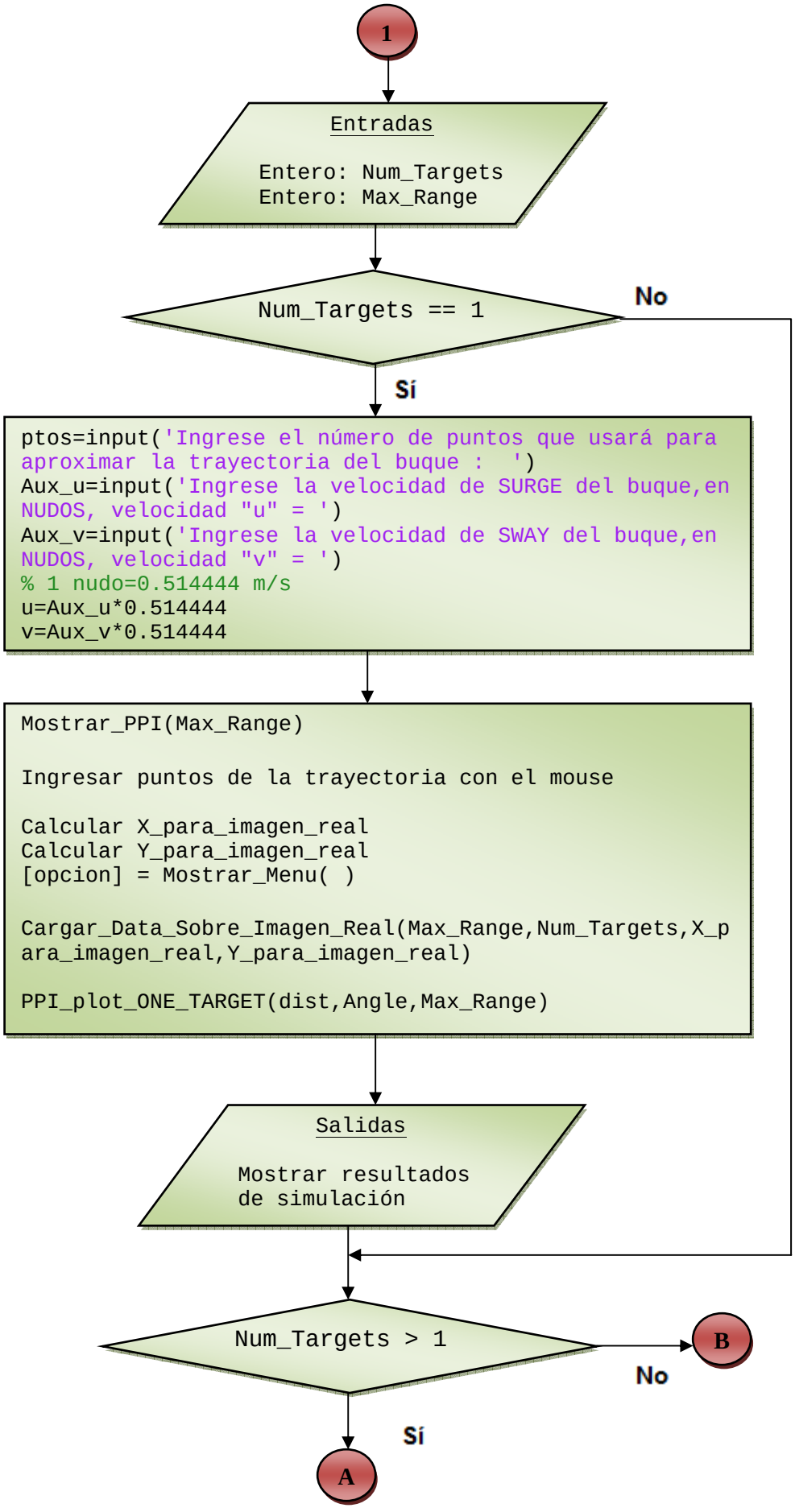
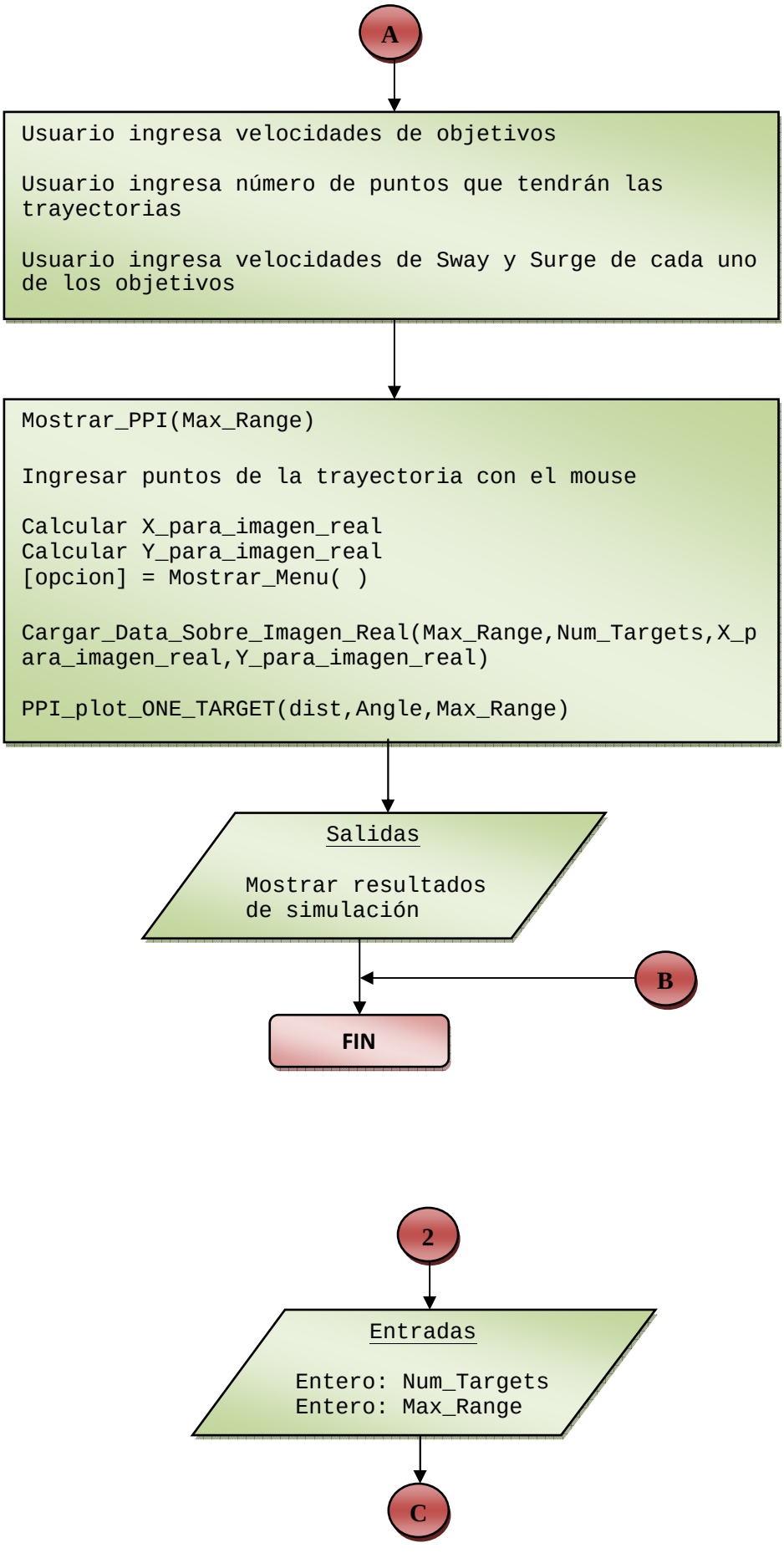
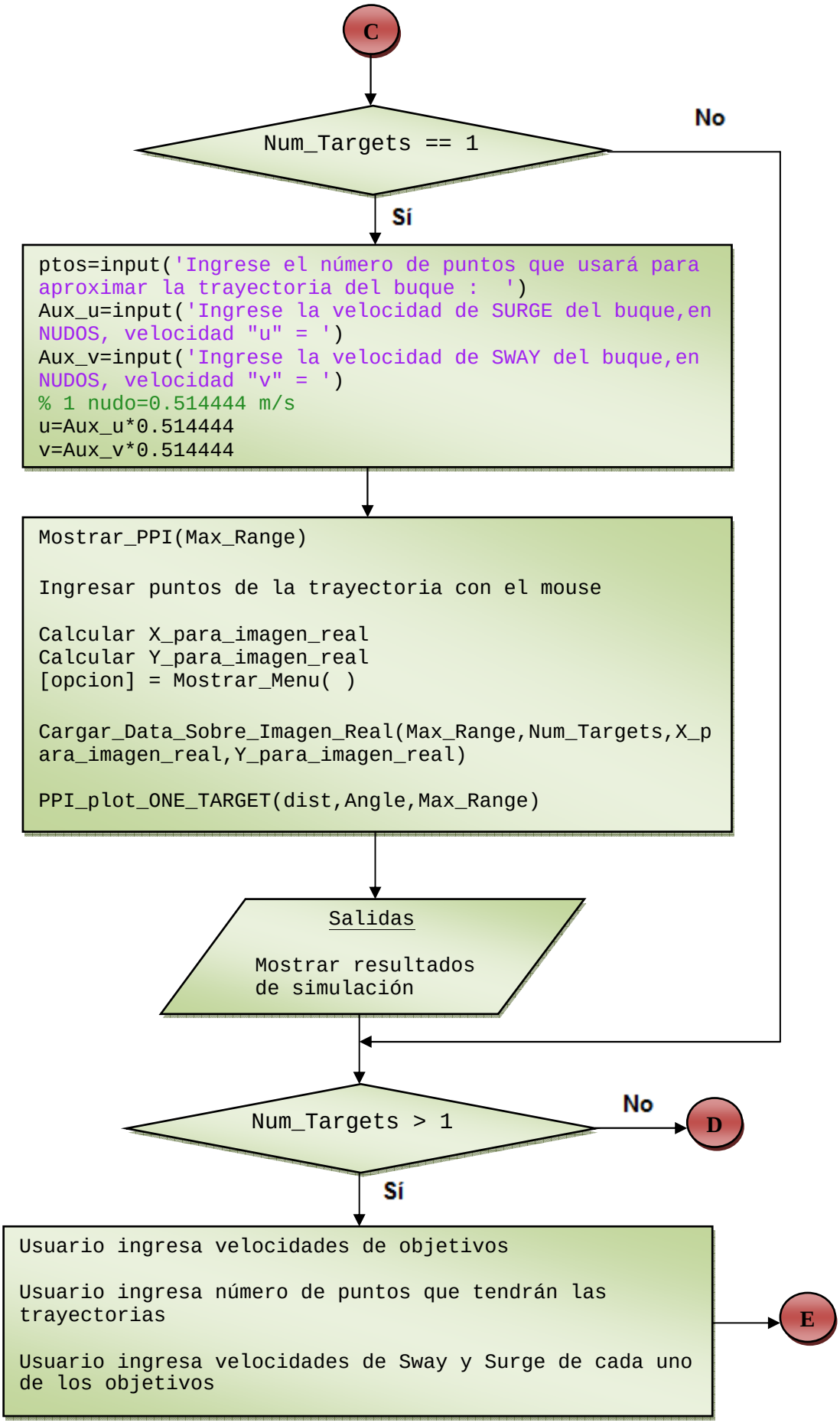

B
Y = -LK
lim = LK
No
lim < = LK
Sí
Gauss2D(Y+LK+1,X+LK+1)= C1*exp(C2*(X^2 + Y^2))
DXGauss2D(Y+LK+1,X+LK+1)= C3*X*exp(C2*(X^2+Y^2))
DYGauss2D(Y+LK+1,X+LK+1)= C3*Y*exp(C2*(X^2+Y^2))
DDXYGauss2D(Y+LK+1,X+LK+1)=(2*C3+C4*(X^2+Y^2))*
exp(C2*(X^2+Y^2))
Y = y + 1
Gauss1D(1 , X+LK+1)= K1*exp(K2*X^2)
DGauss1D(1 , X+LK+1)= K3*X*exp(K2*X^2)
DDGauss1D(1 , X+LK+1)=(K3+K4*X^2)*exp(K2*X^2)
X = X + 1
C
A
Retornar
Arrays:
Gauss2D,DDXYGauss2D,Gau
ss1D,DGauss1D,DDGauss1D
,LK, TK
FIN

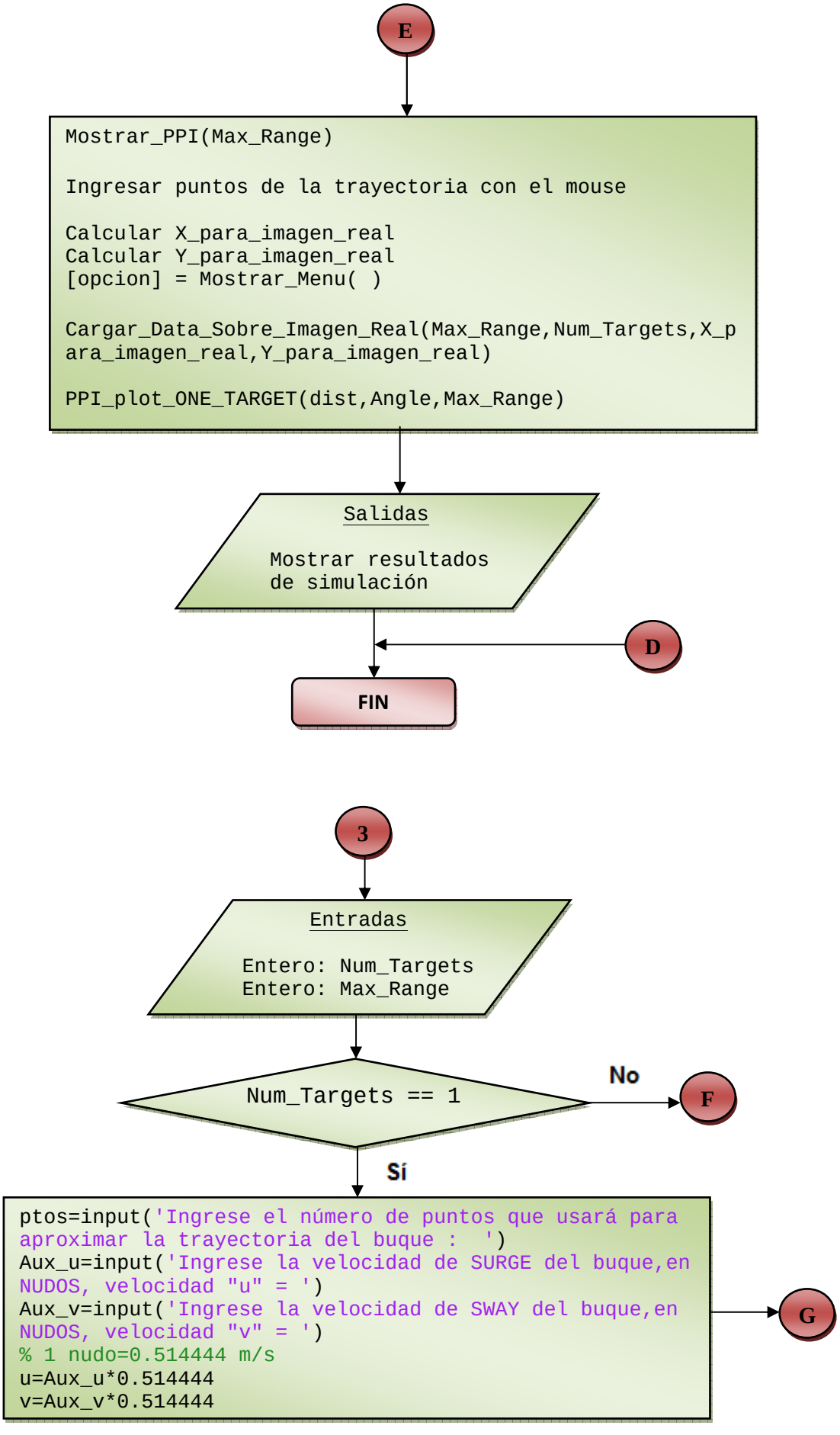
Diagrama de flujo del algoritmo principal

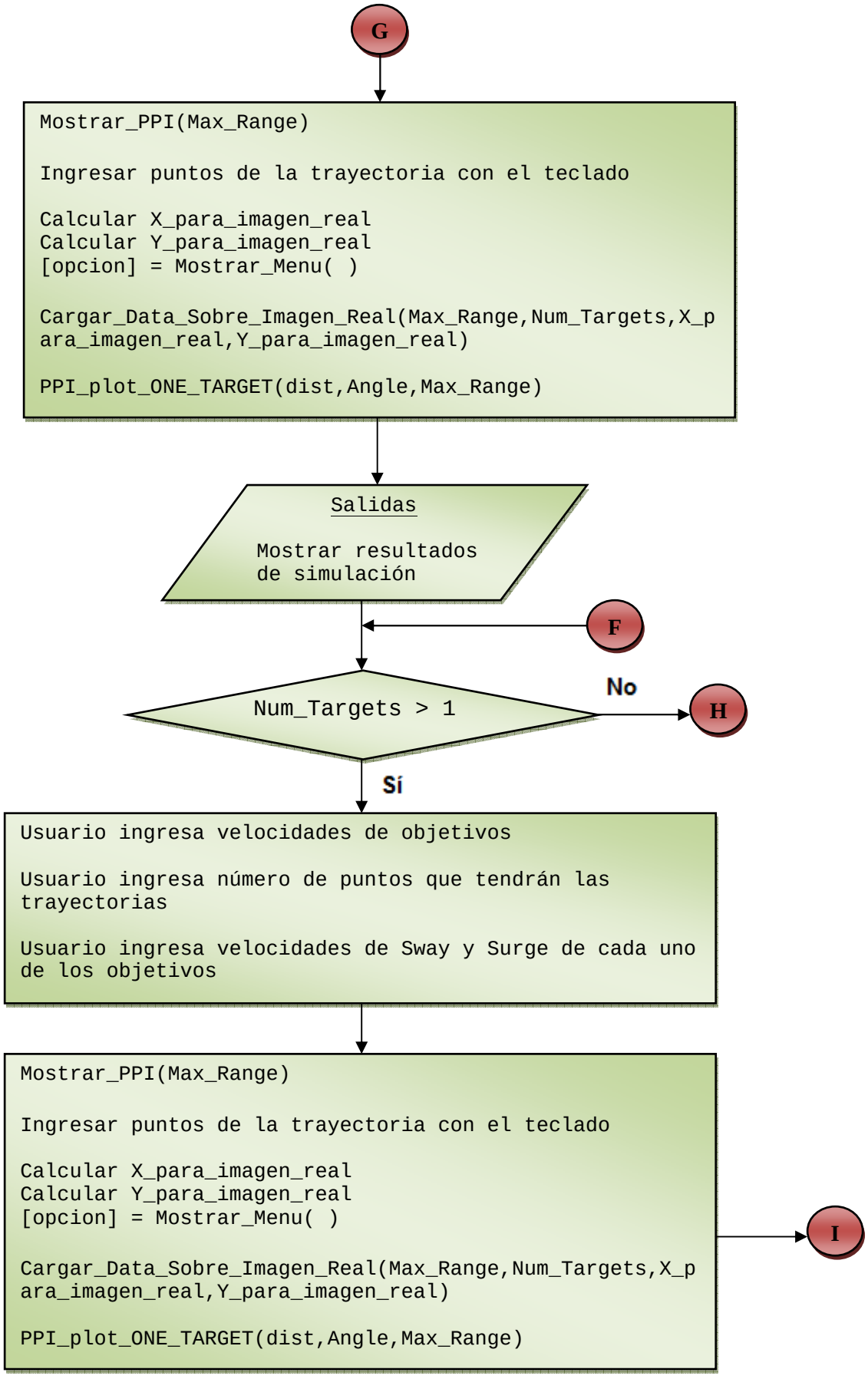


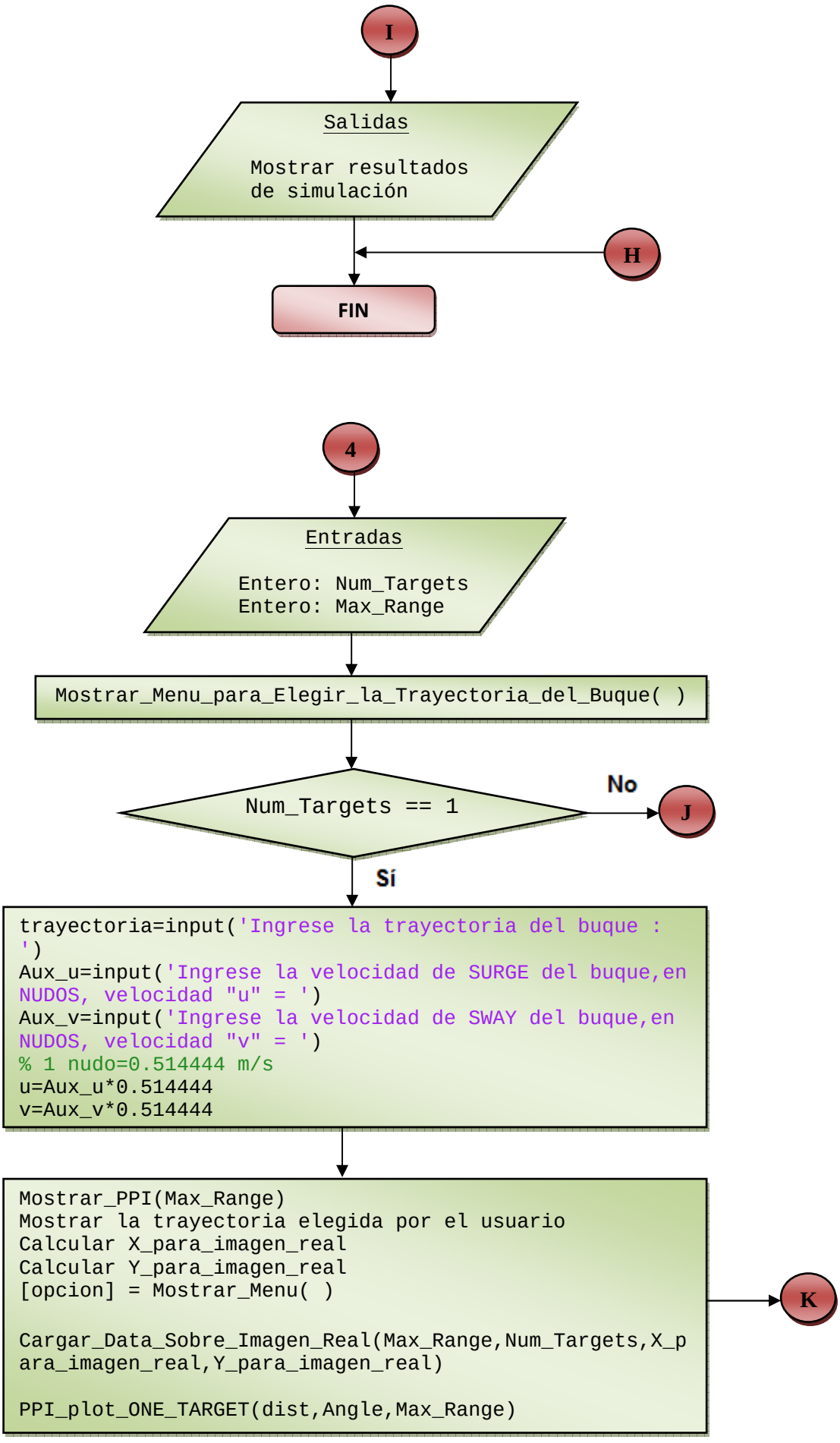

1
Entradas
Entero: Num_Targets
Entero: Max_Range
Num_Targets == 1
No
Sí
ptos=input('Ingrese el número de puntos que usará para aproximar la trayectoria del buque : ')
Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS, velocidad "u" = ')
Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS, velocidad "v" = ')
% 1 nudo=0.514444 m/s
u=Aux_u*0.514444
v=Aux_v*0.514444
Mostrar_PPI(Max_Range)
Ingresar puntos de la trayectoria con el mouse
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
Salidas
Mostrar resultados
de simulación
Num_Targets > 1
B
No
Sí
A

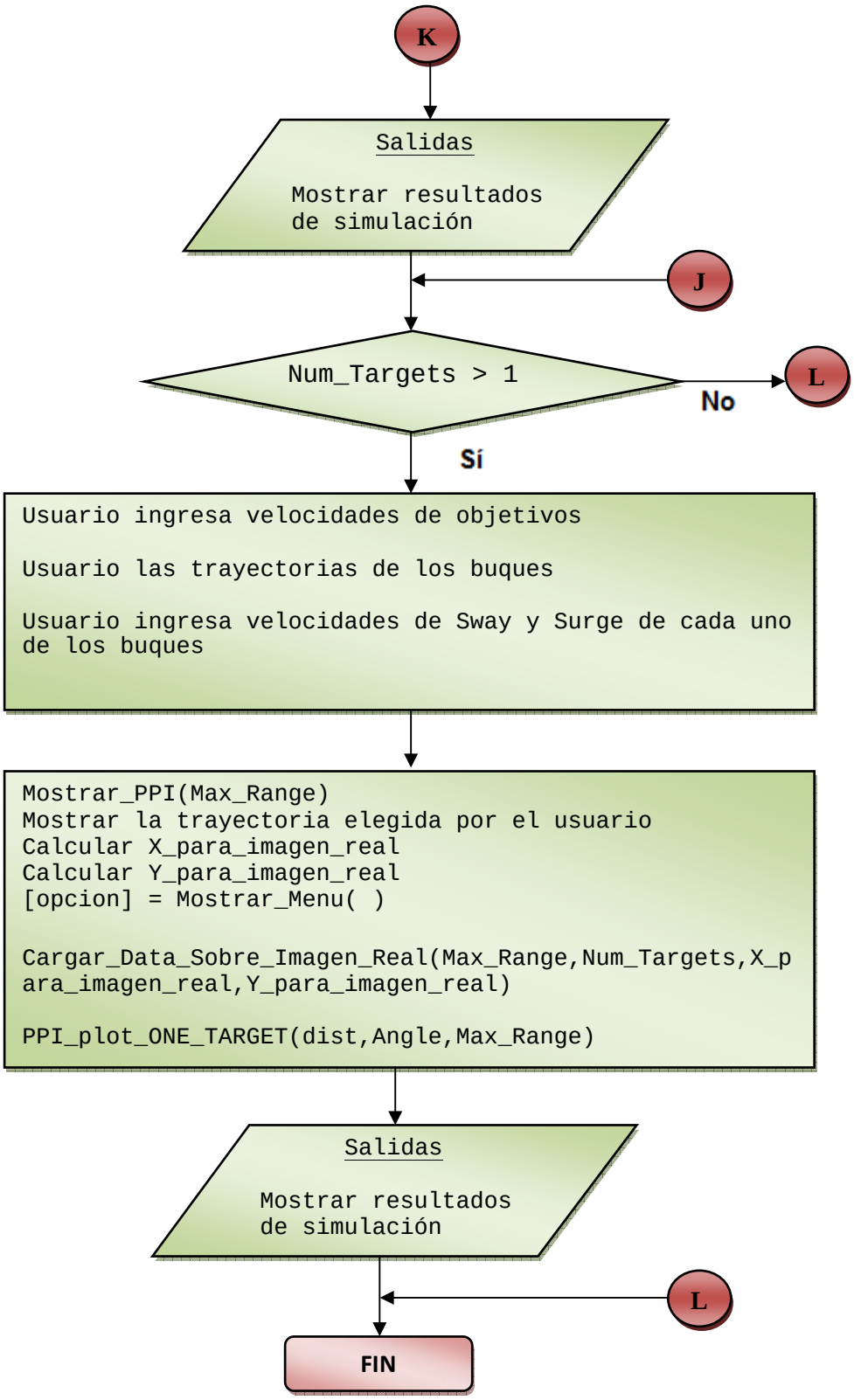

A
Usuario ingresa velocidades de objetivos
Usuario ingresa número de puntos que tendrán las trayectorias
Usuario ingresa velocidades de Sway y Surge de cada uno de los objetivos
Mostrar_PPI(Max_Range)
Ingresar puntos de la trayectoria con el mouse
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
Salidas
Mostrar resultados de simulación
B
FIN
2
Entradas
Entero: Num_Targets
Entero: Max_Range
C


C
Num_Targets == 1
No
Sí
ptos=input('Ingrese el número de puntos que usará para aproximar la trayectoria del buque : ')
Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS, velocidad "u" = ')
Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS, velocidad "v" = ')
% 1 nudo=0.514444 m/s
u=Aux_u*0.514444
v=Aux_v*0.514444
Mostrar_PPI(Max_Range)
Ingresar puntos de la trayectoria con el mouse
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
Salidas
Mostrar resultados de simulación
Num_Targets > 1
No
D
Sí
Usuario ingresa velocidades de objetivos
Usuario ingresa número de puntos que tendrán las trayectorias
Usuario ingresa velocidades de Sway y Surge de cada uno de los objetivos
E


E
Mostrar_PPI(Max_Range)
Ingresar puntos de la trayectoria con el mouse
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
Salidas
Mostrar resultados de simulación
D
FIN
3
Entradas
Entero: Num_Targets
Entero: Max_Range
Num_Targets == 1
No
F
Sí
ptos=input('Ingrese el número de puntos que usará para aproximar la trayectoria del buque : ')
Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS, velocidad "u" = ')
Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS, velocidad "v" = ')
% 1 nudo=0.514444 m/s
u=Aux_u*0.514444
v=Aux_v*0.514444
G


G
Mostrar_PPI(Max_Range)
Ingresar puntos de la trayectoria con el teclado
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
Salidas
Mostrar resultados de simulación
F
Num_Targets > 1
No
H
Sí
Usuario ingresa velocidades de objetivos
Usuario ingresa número de puntos que tendrán las trayectorias
Usuario ingresa velocidades de Sway y Surge de cada uno de los objetivos
Mostrar_PPI(Max_Range)
Ingresar puntos de la trayectoria con el teclado
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
I


I
Salidas
Mostrar resultados de simulación
H
FIN
4
Entradas
Entero: Num_Targets
Entero: Max_Range
Mostrar_Menu_para_Elegir_la_Trayectoria_del_Buque( )
Num_Targets == 1
No
J
Sí
trayectoria=input('Ingrese la trayectoria del buque : ')
Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS, velocidad "u" = ')
Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS, velocidad "v" = ')
% 1 nudo=0.514444 m/s
u=Aux_u*0.514444
v=Aux_v*0.514444
Mostrar_PPI(Max_Range)
Mostrar la trayectoria elegida por el usuario
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
K


K
Salidas
Mostrar resultados de simulación
J
Num_Targets > 1
No
L
Sí
Usuario ingresa velocidades de objetivos
Usuario las trayectorias de los buques
Usuario ingresa velocidades de Sway y Surge de cada uno de los buques
Mostrar_PPI(Max_Range)
Mostrar la trayectoria elegida por el usuario
Calcular X_para_imagen_real
Calcular Y_para_imagen_real
[opcion] = Mostrar_Menu( )
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real,Y_para_imagen_real)
PPI_plot_ONE_TARGET(dist,Angle,Max_Range)
Salidas
Mostrar resultados de simulación
L
FIN

Anexo B
Código en Matlab del algoritmo principal y funciones, agrupados en una GUI (*Graphical User Interface*)

Algoritmo Principal

```
function varargout =
Simulador_de_escenarios_de_radar_GUI_actual(varargin)
% SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL MATLAB code for
Simulador_de_escenarios_de_radar_GUI_actual.fig
%      SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL, by itself, creates
a new SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL or raises the
existing
%      singleton*.
%
%      H = SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL returns the
handle to a new SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL or the
handle to
%      the existing singleton*.
%
%
SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL('CALLBACK',hObject,eventDa
ta,handles,...) calls the local
%      function named CALLBACK in
SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL.M with the given input
arguments.
%
%
SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL('Property','Value',...)
creates a new SIMULADOR_DE_ESCENARIOS_DE_RADAR_GUI_ACTUAL or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before
Simulador_de_escenarios_de_radar_GUI_actual_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
Simulador_de_escenarios_de_radar_GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Simulador_de_escenarios_de_radar_GUI_actual

% Last Modified by GUIDE v2.5 09-Sep-2014 11:14:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn',
@Simulador_de_escenarios_de_radar_GUI_actual_OpeningFcn, ...
                   'gui_OutputFcn',
@Simulador_de_escenarios_de_radar_GUI_actual_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Simulador_de_escenarios_de_radar_GUI_actual
is made visible.
function
Simulador_de_escenarios_de_radar_GUI_actual_OpeningFcn(hObject,
eventdata, handles, varargin)

axes(handles.RCS_TARGETS)
RCS_TARGETS= imread('RCS_TARGETS.jpg');
axis off;
imshow(RCS_TARGETS);

axes(handles.PPI)
PPI= imread('PPI.jpg');
axis off;
imshow(PPI);

% Choose default command line output for
Simulador_de_escenarios_de_radar_GUI_actual
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Simulador_de_escenarios_de_radar_GUI_actual wait for
user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout =
Simulador_de_escenarios_de_radar_GUI_actual_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} =get(handles.pushbutton2,'value');

delete(handles.figure1);
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

clear all;

global Num_Targets
global X_para_imagen_real
global Y_para_imagen_real
global Max_Range

Max_Range=input('Ingrese el alcance máximo del radar, en metros = ');
disp('                                                              ');
Num_Targets=input('INGRESE EL NÚMERO DE BUQUES QUE PARTICIPARÁN EN LA
ESCENA DE RADAR :  ');
disp('                                                              ')

ppi=0;

if (Num_Targets==1)

   ptos=input('Ingrese el número de puntos que usará para aproximar la
    trayectoria del buque :  ');
   disp('                                                           ')
   Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS,
    velocidad "u" = ');
   disp('                                                           ')
   Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS,
    velocidad "v" = ');
   disp('                                                           ')
   % 1 nudo=0.514444 m/s
   u=Aux_u*0.514444;
   v=Aux_v*0.514444;

end

if (Num_Targets>1)

    var=Num_Targets;

    for i=1:1:var

        s1='Ingrese el número de puntos que usará para aproximar la
        trayectoria del buque Nº';
        s2=num2str(i);
        t_aux=[s1 s2];
        disp(t_aux);
        ptos(i)=input('ingrese ahora : ');
        disp('                                                      ')
    end

    for i=1:1:var

        s3='Ingrese la velocidad de SURGE en NUDOS, velocidad "u", del
        buque Nº';
        s4=num2str(i);
        t_aux2=[s3 s4];
        disp(t_aux2);
        u_mt(i)=input('ingrese ahora : ')*0.514444;
```

```
        disp('                                                          ')
        s5='Ingrese la velocidad de SWAY en NUDOS, velocidad "v", del
        buque N°';
        t_aux3=[s5 s4];
        disp(t_aux3);
        v_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                          ')

    end

end

Mostrar_PPI(Max_Range);

if (Num_Targets==1)

    for i=1:1:ptos
        hold on
        [Tx(i),Ty(i)]=ginput(1);
        plot(Tx,Ty,'o');
    end

    hold on
    xx = linspace(min(Tx),max(Tx),400);
    cs=spline(Tx,Ty,xx);
    plot(Tx,Ty,'o',xx,cs,'r');
    hold off;

    varxx=xx.^2;
    varcs=cs.^2;
    radic=varxx+varcs;
    dist=sqrt(radic);
    Angle_bruto=cart2pol(xx,cs);
    warning=0;

    if(Tx(1)~= xx(1))

       conta=1;
       aux=dist;
       aux2=Angle_bruto;

       for p=400:-1:1

           dist(conta)=aux(p);
           Angle_bruto(conta)=aux2(p);
           conta=conta+1;
       end
    end


    for h=1:1:400

        if(Angle_bruto(h)<0)

           Angle(h)=Angle_bruto(h)+(2*pi);

        end
```

```
    if(Angle_bruto(h)>=0)

       Angle(h)=Angle_bruto(h);

    end
end

if (warning==0)

    for z=1:1:400

        kte(z)=(xx(z)-Tx(1))/(cs(z)-Ty(1));
    end

    for f=1:1:400

        Heading_one_target(f)=atan((u-(kte(f)*v))/(v+(kte(f)*u)));
    end

    for f=1:1:400

        t(f)=(xx(f)-Tx(1)+cs(f)-
        Ty(1))/((u*(cos(Heading_one_target(f))+sin(Heading_one_targ
        et(f))))+(v*(cos(Heading_one_target(f))-
        sin(Heading_one_target(f)))));
    end

    for w=1:1:400

        X_para_imagen_real(w)=(u*t(w)*cos(Heading_one_target(w)))-
        (v*t(w)*sin(Heading_one_target(w)))+Tx(1);

        Y_para_imagen_real(w)=(u*t(w)*sin(Heading_one_target(w)))+(
        v*t(w)*cos(Heading_one_target(w)))+Ty(1);

    end
end

disp('                                                              ')
disp('A continuación, se procederá con la simulación....           ')
disp('                                                              ')
disp('Digite un número de acuerdo al MENÚ siguiente...             ')
disp('                                                              ')
[opcion] = Mostrar_Menu( );

if (opcion==1)

    figure

   Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imag
   en_real,Y_para_imagen_real,t);
end

if (opcion==2)

    PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

end
```

```
end

if (Num_Targets>1)

    dist=zeros(length(ptos),400);
    Angle=zeros(length(ptos),400);
    tx=zeros(length(ptos),1);
    ty=zeros(length(ptos),1);

    for i=1:1:length(ptos)

        for j=1:1:ptos(i)

            hold on
            [Tx(j),Ty(j)]=ginput(1);
            plot(Tx,Ty,'o');
        end

        tx(i,1)=Tx(1);
        ty(i,1)=Ty(1);

        hold on
        xx = linspace(min(Tx),max(Tx),400);
        cs=spline(Tx,Ty,xx);
        plot(Tx,Ty,'o',xx,cs,'r');
        hold off;

        varxx=xx.^2;
        varcs=cs.^2;
        radic=varxx+varcs;
        distancias=sqrt(radic);
        Angle_bruto=cart2pol(xx,cs);
        warning=0;

       if(Tx(1)~= xx(1))

          conta=1;
          aux=distancias;
          aux2=Angle_bruto;

          for p=400:-1:1

              distancias(conta)=aux(p);
              Angle_bruto(conta)=aux2(p);
              conta=conta+1;
          end
       end

       for h=1:1:400

           if(Angle_bruto(h)<0)

              Angulos(h)=Angle_bruto(h)+(2*pi);
           end

           if(Angle_bruto(h)>=0)
```

```
            Angulos(h)=Angle_bruto(h);
        end
    end

    for k=1:1:400

         dist(i,k)=distancias(k);
         Angle(i,k)=Angulos(k);
    end

    if (warning==0)

        for w=1:1:400

            k_mt(i,w)=(xx(w)-tx(i,1))/(cs(w)-ty(i,1));
        end

        for f=1:1:400

            Heading_multiple_targets(i,f)=atan((u_mt(i)-
             (k_mt(i,f)*v_mt(i)))/(v_mt(i)+(k_mt(i,f)*u_mt(i))));
        end

        for f=1:1:400

         t(i,f)=(xx(f)-Tx(1)+cs(f)-
         Ty(1))/((u_mt(i)*(cos(Heading_multiple_targets(i,f))+sin(He
         ading_multiple_targets(i,f))))+(v_mt(i)*(cos(Heading_multip
         le_targets(i,f))-sin(Heading_multiple_targets(i,f)))));
        end

    end

end

for i=1:1:length(ptos)

     for w=1:1:400


         X_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*cos(Heading_multipl
         e_targets(i,w)))-
         (v_mt(i)*t(i,w)*sin(Heading_multiple_targets(i,w)))+tx(i,1)
         ;

         Y_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*sin(Heading_multipl
         e_targets(i,w)))+(v_mt(i)*t(i,w)*cos(Heading_multiple_targe
         ts(i,w)))+ty(i,1);

     end
 end

 disp('                                                          ')
 disp('A continuación, se procederá con la simulación....        ')
 disp('                                                          ')
 disp('Digite un número de acuerdo al MENÚ siguiente...          ')
 disp('                                                          ')
 [opcion] = Mostrar_Menu( );
```

```
    if (opcion==1)

        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_ima
        gen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_MULTIPLE_TARGETS(dist,Angle,Max_Range,Num_Targets);

    end

end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

clear all;

global Num_Targets
global X_para_imagen_real
global Y_para_imagen_real
global Max_Range

Max_Range=input('Ingrese el alcance máximo del radar, en metros = ');
disp('                                                              ');

Num_Targets=input('INGRESE EL NÚMERO DE BUQUES QUE PARTICIPARÁN EN LA
ESCENA DE RADAR :  ');
disp('                                                              ');

ppi=0;

if (Num_Targets==1)

   ptos=input('Ingrese el número de puntos que usará para aproximar la
    trayectoria del buque :  ');
   disp('                                                           ')
   Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS,
    velocidad "u" = ');
   disp('                                                           ')
   Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS,
    velocidad "v" = ');
   disp('                                                           ')
   % 1 nudo=0.514444 m/s
   u=Aux_u*0.514444;
   v=Aux_v*0.514444;

end

if (Num_Targets>1)

    var=Num_Targets;

    for i=1:1:var
```

```
        disp('                                                          ')
        s1='Ingrese el número de puntos que usará para aproximar la
        trayectoria del buque Nº';
        s2=num2str(i);
        t_aux=[s1 s2];
        disp(t_aux);
        ptos(i)=input('ingrese ahora : ');
        disp('                                                           ')
    end

    for i=1:1:var

        s3='Ingrese la velocidad de SURGE en NUDOS, velocidad "u", del
        buque Nº';
        s4=num2str(i);
        t_aux2=[s3 s4];
        disp(t_aux2);
        u_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                        ')
        s5='Ingrese la velocidad de SWAY en NUDOS, velocidad "v", del
        buque Nº';
        t_aux3=[s5 s4];
        disp(t_aux3);
        v_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                           ')

    end

end

Mostrar_PPI(Max_Range);

if (Num_Targets==1)

    for i=1:1:ptos
        hold on
        [Tx(i),Ty(i)]=ginput(1);

        if (i==1)
            plot(Tx(i),Ty(i),'ro');
        end

        if (i==ptos)
            plot(Tx(i),Ty(i),'ro');
        end

        if(i>1 && i<ptos)
            plot(Tx(i),Ty(i),'o');
        end
    end

    varxx=Tx.^2;
    varcs=Ty.^2;
    radic=varxx+varcs;
    dist=sqrt(radic);
    Angle_bruto=cart2pol(Tx,Ty);
    warning=0;
```

```
for h=1:1:ptos

    if(Angle_bruto(h)<0)

       Angle(h)=Angle_bruto(h)+(2*pi);

    end

    if(Angle_bruto(h)>=0)

       Angle(h)=Angle_bruto(h);

    end
end

if (warning==0)

    for z=1:1:ptos

        kte(z)=(Tx(z)-Tx(1))/(Ty(z)-Ty(1));
    end

    for f=1:1:ptos

        Heading_one_target(f)=atan((u-(kte(f)*v))/(v+(kte(f)*u)));
    end

    for f=1:1:ptos

        t(f)=(Tx(f)-Tx(1)+Ty(f)-
        Ty(1))/((u*(cos(Heading_one_target(f))+sin(Heading_one_targ
        et(f))))+(v*(cos(Heading_one_target(f))-
        sin(Heading_one_target(f)))));
    end

    for w=1:1:ptos

        X_para_imagen_real(w)=(u*t(w)*cos(Heading_one_target(w)))-
        (v*t(w)*sin(Heading_one_target(w)))+Tx(1);

        Y_para_imagen_real(w)=(u*t(w)*sin(Heading_one_target(w)))+(
        v*t(w)*cos(Heading_one_target(w)))+Ty(1);

    end
end

disp('                                                          ')
disp('A continuación, se procederá con la simulación....        ')
disp('                                                          ')
disp('Digite un número de acuerdo al MENÚ siguiente...          ')
disp('                                                          ')
[opcion] = Mostrar_Menu( );

if (opcion==1)

    figure
```

```
        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_ima
        gen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_ONE_TARGET(dist,Angle,Max_Range,Num_Targets);

    end
end

if (Num_Targets>1)

    tx=zeros(length(ptos),1);
    ty=zeros(length(ptos),1);

    for i=1:1:length(ptos)

        for j=1:1:ptos(i)

            hold on
            [Tx(j),Ty(j)]=ginput(1);

            if(j==1)
               plot(Tx(j),Ty(j),'ro');
            end

            if(j==ptos(i))
               plot(Tx(j),Ty(j),'ro');
            end

            if(j>1 && j<ptos(i))
               plot(Tx(j),Ty(j),'o');
            end
        end

        tx(i,1)=Tx(1);
        ty(i,1)=Ty(1);

        varxx=Tx.^2;
        varcs=Ty.^2;
        radic=varxx+varcs;
        distancias=sqrt(radic);
        Angle_bruto=cart2pol(Tx,Ty);
        warning=0;

       for h=1:1:ptos(i)

           if(Angle_bruto(h)<0)

              Angulos(h)=Angle_bruto(h)+(2*pi);
           end

           if(Angle_bruto(h)>=0)

              Angulos(h)=Angle_bruto(h);
           end
```

```
    end

    for k=1:1:ptos(i)

         dist(i,k)=distancias(k);
         Angle(i,k)=Angulos(k);
    end

    if (warning==0)

        for w=1:1:ptos(i)

            k_mt(i,w)=(Tx(w)-tx(i,1))/(Ty(w)-ty(i,1));
        end

        for f=1:1:ptos(i)

            Heading_multiple_targets(i,f)=atan((u_mt(i)-
             (k_mt(i,f)*v_mt(i)))/(v_mt(i)+(k_mt(i,f)*u_mt(i))));
        end

        for f=1:1:ptos(i)

         t(i,f)=(Tx(f)-tx(i,1)+Ty(f)-
         ty(i,1))/((u_mt(i)*(cos(Heading_multiple_targets(i,f))+sin(
         Heading_multiple_targets(i,f))))+(v_mt(i)*(cos(Heading_mult
         iple_targets(i,f))-sin(Heading_multiple_targets(i,f)))));
        end
    end

end

for i=1:1:length(ptos)

    for w=1:1:ptos(i)


        X_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*cos(Heading_multipl
        e_targets(i,w)))-
        (v_mt(i)*t(i,w)*sin(Heading_multiple_targets(i,w)))+tx(i,1)
        ;

        Y_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*sin(Heading_multipl
        e_targets(i,w)))+(v_mt(i)*t(i,w)*cos(Heading_multiple_targe
        ts(i,w)))+ty(i,1);

    end
end

disp('                                                              ')
disp('A continuación, se procederá con la simulación....           ')
disp('                                                              ')
disp('Digite un número de acuerdo al MENÚ siguiente...             ')
disp('                                                              ')
[opcion] = Mostrar_Menu( );

if (opcion==1)
```

```
        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_ima
        gen_real,Y_para_imagen_real);
    end

    if (opcion==2)

        PPI_plot_MULTIPLE_TARGETS(dist,Angle,Max_Range,Num_Targets);

    end

end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)

clear all;

global Num_Targets
global X_para_imagen_real
global Y_para_imagen_real

Max_Range=input('Ingrese el alcance máximo del radar, en metros = ');
disp('                                                              ');
Num_Targets=input('INGRESE EL NÚMERO DE BUQUES QUE PARTICIPARÁN EN LA
ESCENA DE RADAR :  ');
disp('                                                            ');
ppi=0;

if (Num_Targets==1)

   ptos=input('Ingrese el número de coordenadas que digitará para
    representar la trayectoria del buque :  ');
   disp('                                                          ')
   Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS,
    velocidad "u" = ');
   disp('                                                          ')
   Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS,
    velocidad "v" = ');
   disp('                                                          ')
   % 1 nudo=0.514444 m/s
   u=Aux_u*0.514444;
   v=Aux_v*0.514444;
end

if (Num_Targets>1)

    var=Num_Targets;

    for i=1:1:var

        s1='Ingrese el número de puntos que usará para representar la
        trayectoria del buque Nº';
        s2=num2str(i);
        t_aux=[s1 s2];
        disp(t_aux);
        ptos(i)=input('ingrese ahora : ');
        disp('                                                          ')
```

```
    end

    for i=1:1:var

        s3='Ingrese la velocidad de SURGE en NUDOS, velocidad "u", del
        buque Nº';
        s4=num2str(i);
        t_aux2=[s3 s4];
        disp(t_aux2);
        u_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                          ')
        s5='Ingrese la velocidad de SWAY en NUDOS, velocidad "v", del
        buque Nº';
        t_aux3=[s5 s4];
        disp(t_aux3);
        v_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                        ')

    end

end

Mostrar_PPI(Max_Range);

if (Num_Targets==1)

    for i=1:1:ptos

        hold on

        s4='Ingrese la COORDENADA X Nº';
        s5=num2str(i);
        t_aux3=[s4 s5];
        disp(t_aux3);
        Tx(i)=input('Ingrese ahora : ');
        disp('                                                          ')

        s6='Ingrese la COORDENADA Y Nº';
        s7=num2str(i);
        t_aux4=[s6 s7];
        disp(t_aux4);
        Ty(i)=input('Ingrese ahora : ');
        disp('                                                          ')

        plot(Tx,Ty,'o');
    end

    varxx=Tx.^2;
    varcs=Ty.^2;
    radic=varxx+varcs;
    dist=sqrt(radic);
    Angle_bruto=cart2pol(Tx,Ty);
    warning=0;

    for h=1:1:ptos

        if(Angle_bruto(h)<0)
```

```
       Angle(h)=Angle_bruto(h)+(2*pi);

    end

    if(Angle_bruto(h)>=0)

       Angle(h)=Angle_bruto(h);

    end
end

if (warning==0)

    for z=1:1:ptos

        kte(z)=(Tx(z)-Tx(1))/(Ty(z)-Ty(1));
    end

    for f=1:1:ptos

        Heading_one_target(f)=atan((u-(kte(f)*v))/(v+(kte(f)*u)));
    end

    for f=1:1:ptos

        t(f)=(Tx(f)-Tx(1)+Ty(f)-
        Ty(1))/((u*(cos(Heading_one_target(f))+sin(Heading_one_targ
        et(f))))+(v*(cos(Heading_one_target(f))-
        sin(Heading_one_target(f)))));
    end

    for w=1:1:ptos

        X_para_imagen_real(w)=(u*t(w)*cos(Heading_one_target(w)))-
        (v*t(w)*sin(Heading_one_target(w)))+Tx(1);

        Y_para_imagen_real(w)=(u*t(w)*sin(Heading_one_target(w)))+(
        v*t(w)*cos(Heading_one_target(w)))+Ty(1);

    end
end

disp('                                                              ')
disp('A continuación, se procederá con la simulación....           ')
disp('                                                              ')
disp('Digite un número de acuerdo al MENÚ siguiente...             ')
disp('                                                              ')
[opcion] = Mostrar_Menu( );

if (opcion==1)

    figure

    Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_ima
    gen_real,Y_para_imagen_real,t);
end

if (opcion==2)
```

```
        PPI_plot_ONE_TARGET(dist,Angle,Max_Range,Num_Targets);

    end
end

if (Num_Targets>1)

    tx=zeros(length(ptos),1);
    ty=zeros(length(ptos),1);

    for i=1:1:length(ptos)

        for j=1:1:ptos(i)

            hold on
            s4='Ingrese la COORDENADA X Nº';
            s5=num2str(j);
            s6=' perteneciente al OBJETIVO Nº';
            disp('                                                    ')
            s7=num2str(i);
            t_aux3=[s4 s5 s6 s7];
            disp(t_aux3);
            Tx(j)=input('Ingrese ahora : ');
            disp('                                                    ')

            s8='Ingrese la COORDENADA Y Nº';
            s9=num2str(j);
            s10=' perteneciente al OBJETIVO Nº';
            s11=num2str(i);
            t_aux4=[s8 s9 s10 s11];
            disp(t_aux4);
            Ty(j)=input('Ingrese ahora : ');
            disp('                                                    ')

            plot(Tx,Ty,'o');
        end

        tx(i,1)=Tx(1);
        ty(i,1)=Ty(1);

        varxx=Tx.^2;
        varcs=Ty.^2;
        radic=varxx+varcs;
        distancias=sqrt(radic);
        Angle_bruto=cart2pol(Tx,Ty);
        warning=0;

       for h=1:1:ptos(i)

          if(Angle_bruto(h)<0)

             Angulos(h)=Angle_bruto(h)+(2*pi);
          end

          if(Angle_bruto(h)>=0)

             Angulos(h)=Angle_bruto(h);
```

```
        end
    end

    for k=1:1:ptos(i)

         dist(i,k)=distancias(k);
         Angle(i,k)=Angulos(k);
    end

    if (warning==0)

        for w=1:1:ptos(i)

            k_mt(i,w)=(Tx(w)-tx(i,1))/(Ty(w)-ty(i,1));
        end

        for f=1:1:ptos(i)

            Heading_multiple_targets(i,f)=atan((u_mt(i)-
             (k_mt(i,f)*v_mt(i)))/(v_mt(i)+(k_mt(i,f)*u_mt(i))));
        end

        for f=1:1:ptos(i)

         t(i,f)=(Tx(f)-tx(i,1)+Ty(f)-
         ty(i,1))/((u_mt(i)*(cos(Heading_multiple_targets(i,f))+sin(
         Heading_multiple_targets(i,f))))+(v_mt(i)*(cos(Heading_mult
         iple_targets(i,f))-sin(Heading_multiple_targets(i,f)))));
        end
    end

end

for i=1:1:length(ptos)

    for w=1:1:ptos(i)


        X_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*cos(Heading_multipl
        e_targets(i,w)))-
        (v_mt(i)*t(i,w)*sin(Heading_multiple_targets(i,w)))+tx(i,1)
        ;

        Y_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*sin(Heading_multipl
        e_targets(i,w)))+(v_mt(i)*t(i,w)*cos(Heading_multiple_targe
        ts(i,w)))+ty(i,1);

    end
end

disp('                                                          ')
disp('A continuación, se procederá con la simulación....        ')
disp('                                                          ')
disp('Digite un número de acuerdo al MENÚ siguiente...          ')
disp('                                                          ')
[opcion] = Mostrar_Menu( );

if (opcion==1)
```

```
        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_ima
        gen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_MULTIPLE_TARGETS(dist,Angle,Max_Range,Num_Targets);

    end

end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)

clear all;

global Num_Targets
global X_para_imagen_real
global Y_para_imagen_real
global Max_Range

Max_Range=input('Ingrese el alcance máximo del radar, en metros = ');
Mostrar_mensaje_al_usuario( )

Num_Targets=input('INGRESE EL NÚMERO DE BUQUES QUE PARTICIPARÁN EN LA
ESCENA DE RADAR :  ');

disp('                                                                ')
disp('AHORA ESCOGA LA TRAYECTORIA DE CADA UNO DE LOS BUQUES SEGÚN EL
NÚMERO ')
disp('QUE SE INDICA, PARA CADA TRAYECTORIA, EN EL SIGUIENTE MENÚ : ');
disp('                                                                ')

Mostrar_Menu_para_Elegir_la_Trayectoria_del_Buque( )

if (Num_Targets==1)

   disp('                                                              ')
   trayectoria=input('Ingrese la trayectoria que tendrá del buque :');
   disp('                                                              ')
   Aux_u=input('Ingrese la velocidad de SURGE del buque,en NUDOS,
  velocidad "u" = ');
   disp('                                                              ')
   Aux_v=input('Ingrese la velocidad de SWAY del buque,en NUDOS,
    velocidad "v" = ');
   disp('                                                              ')
   % 1 nudo=0.514444 m/s
   u=Aux_u*0.514444;
   v=Aux_v*0.514444;

end

if (Num_Targets>1)
```

```
    var=Num_Targets;

    for i=1:1:var

        s1='Ingrese la trayectoria que tendrá el buque Nº';
        s2=num2str(i);
        t_aux=[s1 s2];
        disp(t_aux);
        trayectoria(i)=input('ingrese ahora : ');
        disp('                                                          ')
    end

    for i=1:1:var

        s3='Ingrese la velocidad de SURGE en NUDOS, velocidad "u", del
        buque Nº';
        s4=num2str(i);
        t_aux2=[s3 s4];
        disp(t_aux2);
        u_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                          ')
        s5='Ingrese la velocidad de SWAY en NUDOS, velocidad "v", del
        buque Nº';
        t_aux3=[s5 s4];
        disp(t_aux3);
        v_mt(i)=input('ingrese ahora : ')*0.514444;
        disp('                                                          ')
    end
end

if(Num_Targets==1)

    Mostrar_PPI(Max_Range);
    hold on

    if(trayectoria==1)

       xc=8000; yc=-15000; r=10000; % Centro(xc,yc) y radio(r)
       n = 300; k=0:n; fi=2*pi*k/n;
       circular_pos_x=xc+r*cos(fi);
       circular_pos_y=yc+r*sin(fi);
       plot(circular_pos_x,circular_pos_y,'r');
       ptos=length(circular_pos_x);
       Tx=circular_pos_x;
       Ty=circular_pos_y;
       [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
       Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v);
       disp('                                                          ')
       disp('A continuación, se procederá con la simulación....       ')
       disp('                                                          ')
       disp('Digite un número de acuerdo al MENÚ siguiente...         ')
       disp('                                                          ')
       [opcion] = Mostrar_Menu( );

       if (opcion==1)

           figure
```

```
        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_i
        magen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

    end

end

if(trayectoria==2)

    t=linspace(-9500,19000,400);
    parabolica_pos_x=(t.*((t-60000).^2-90000000))./((t-
    60000).^2+90000000);
    parabolica_pos_y=(60000*((t-30000).^2-90000000))./((t-
    30000).^2+90000000);
    plot(parabolica_pos_x,parabolica_pos_y,'r')
    ptos=length(parabolica_pos_x);
    Tx=parabolica_pos_x;
    Ty=parabolica_pos_y;
    [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
    Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v);
    disp('                                                    ')
    disp('A continuación, se procederá con la simulación....  ')
    disp('                                                    ')
    disp('Digite un número de acuerdo al MENÚ siguiente...    ')
    disp('                                                    ')
    [opcion] = Mostrar_Menu( );

    if (opcion==1)

        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_i
        magen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

    end

end

if(trayectoria==3)

    t=pi/2: 0.01 :pi;
    UNCUARTO_eliptica_pos_x=40000*cos(t);
    UNCUARTO_eliptica_pos_y=20000*sin(t);
    plot(UNCUARTO_eliptica_pos_x,UNCUARTO_eliptica_pos_y,'b')
    ptos=length(UNCUARTO_eliptica_pos_x);
    Tx=UNCUARTO_eliptica_pos_x;
    Ty=UNCUARTO_eliptica_pos_y;
```

```
    [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
    Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v);
    disp('                                                    ')
    disp('A continuación, se procederá con la simulación....  ')
    disp('                                                    ')
    disp('Digite un número de acuerdo al MENÚ siguiente...    ')
    disp('                                                    ')
    [opcion] = Mostrar_Menu( );

    if (opcion==1)

        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_i
        magen_real,Y_para_imagen_real);
    end

    if (opcion==2)

        PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

    end

end

if(trayectoria==4)

    t=pi/2: 0.01 :2*pi;
    TRES_CUARTO_eliptica_pos_x=55000*cos(t);
    TRES_CUARTO_eliptica_pos_y=25000*sin(t);
    plot(TRES_CUARTO_eliptica_pos_x,TRES_CUARTO_eliptica_pos_y,'r')
    ptos=length(TRES_CUARTO_eliptica_pos_x);
    Tx=TRES_CUARTO_eliptica_pos_x;
    Ty=TRES_CUARTO_eliptica_pos_y;
    [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
    Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v);
    disp('                                                    ')
    disp('A continuación, se procederá con la simulación....  ')
    disp('                                                    ')
    disp('Digite un número de acuerdo al MENÚ siguiente...    ')
    disp('                                                    ')
    [opcion] = Mostrar_Menu( );

    if (opcion==1)

        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_i
        magen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

    end

end
```

```
if(trayectoria==5)

   t=linspace(-30000,25000,400);
   Linvertida_pos_x=(t.*(t.^2-90000000))./(t.^2+90000000);
   Linvertida_pos_y=(60000*(t.^2-90000000))./(t.^2+90000000);
   plot(Linvertida_pos_x,Linvertida_pos_y,'b')
   ptos=length(Linvertida_pos_x);
   Tx=Linvertida_pos_x;
   Ty=Linvertida_pos_y;
   [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
   Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v);
   disp('                                                    ')
   disp('A continuación, se procederá con la simulación....  ')
   disp('                                                    ')
   disp('Digite un número de acuerdo al MENÚ siguiente...    ')
   disp('                                                    ')
   [opcion] = Mostrar_Menu( );

   if (opcion==1)

       figure

   Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imag
   en_real,Y_para_imagen_real);
   end

   if (opcion==2)

       PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

   end

end

if(trayectoria==6)

   x=linspace(-30000,20000,400);
   f=0.5.*x+20000;
   plot(x,f,'b')
   ptos=length(x);
   Tx=x;
   Ty=f;
   [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
   Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v);
   disp('                                                    ')
   disp('A continuación, se procederá con la simulación....  ')
   disp('                                                    ')
   disp('Digite un número de acuerdo al MENÚ siguiente...    ')
   disp('                                                    ')
   [opcion] = Mostrar_Menu( );

   if (opcion==1)

       figure

   Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imag
   en_real,Y_para_imagen_real,t);
   end
```

```
        if (opcion==2)

            PPI_plot_ONE_TARGET(dist,Angle,Max_Range);

        end
    end
end

if(Num_Targets>1)

    aux_ptos=zeros(Num_Targets,1);
    Mostrar_PPI(Max_Range);
    hold on

    [Tx,Ty,ptos]=Generador_de_posiciones_para_buques(Num_Targets,trayec
    toria,aux_ptos);

    [dist,Angle,X_para_imagen_real,Y_para_imagen_real,t]=Generar_data_p
    ara_simulacion(Num_Targets,Tx,Ty,ptos,u_mt,v_mt);
    disp('                                                          ')
    disp('A continuación, se procederá con la simulación....        ')
    disp('                                                          ')
    disp('Digite un número de acuerdo al MENÚ siguiente...          ')
    disp('                                                          ')
    [opcion] = Mostrar_Menu( );

    if (opcion==1)

        figure

        Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_ima
        gen_real,Y_para_imagen_real,t);
    end

    if (opcion==2)

        PPI_plot_MULTIPLE_TARGETS(dist,Angle,Max_Range,Num_Targets);

    end
end

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isequal(get(hObject,'waitstatus'),'waiting')
    uiresume(hObject);
else
delete(hObject);
end
```

Funciones

Función *PPI_plot_ONE_TARGET*

```
function PPI_plot_ONE_TARGET(dist,Angle,Max_Range)

angle=Angle;

for j=1:length(dist)

      for i=(2*pi):-(pi/180):(pi/180)
          [x,y]=(pol2cart(i,Max_Range));
          z=complex(x,y);
          h=compass(z);
          ph=findall(gca,'type','patch');
          set(ph,'facecolor',[0,0.5,0]);
          set( h, 'Color', [0.5,1,0] );
          set( h, 'LineWidth', 4 );
          for k = 1:length(h)
              a = get(h(k), 'xdata');
              b = get(h(k), 'ydata');
              set(h(k), 'xdata', a(1:2), 'ydata', b(1:2)')
          end
          view(90,-90)
          if(angle>2*pi)
             q=(angle/(2*pi));
             p=fix(q);
             var=(angle-(p*2*pi));
             angle=var;
          end

          if(dist(j)<=Max_Range)
             if(j>1)
                if(angle(j-1)<i)
                   hold on
                   plothandle=polar(angle(j-1),dist(j-1),'O');

                   set(plothandle,'markerfacecolor',[0,1,0],'markeredg
                   ecolor',[0,1,0]);
                   get( plothandle ) ;
                   set( plothandle, 'Color', [ 0.5, 1, 0 ] );
                   set ( plothandle, 'LineWidth', 2 );
                   hold off
                end
             end
          end

          if(dist(j)<=Max_Range)
             if(angle(j)>=i)
                hold on
                plothandle=polar(angle(j),dist(j),'O');

                 set(plothandle,'markerfacecolor',[0,1,0],'markeredgeco
                 lor',[0,1,0])
                get( plothandle ) ;
                set( plothandle, 'Color', [ 0.5, 1, 0 ] );
                set ( plothandle, 'LineWidth', 2 );
```

```
                hold off
            end
        end
        pause(1*10^-8);
    end
end

end
```

Función *PPI_plot_MULTIPLE_TARGETS*

```
function PPI_plot_MULTIPLE_TARGETS(dist,Angle,Max_Range,Num_Targets)

angle=Angle;

for j=1:(length(dist))

    for i=(2*pi):-(pi/180):(pi/180)
        [x,y]=(pol2cart(i,Max_Range));
        z=complex(x,y);
        h=compass(z);
        ph=findall(gca,'type','patch');
        set(ph,'facecolor',[0,0.5,0]);
        set( h, 'Color', [0.5,1,0] );
        set( h, 'LineWidth', 4 );
        for k = 1:length(h)
            a = get(h(k), 'xdata');
            b = get(h(k), 'ydata');
            set(h(k), 'xdata', a(1:2), 'ydata', b(1:2)')
        end
        view(90,-90)

        for t=1:1:Num_Targets

            if(angle(t,j)>2*pi)
               q=(angle(t,j)/(2*pi));
               p=fix(q);
               var=(angle(t,j)-(p*2*pi));
               angle(t,j)=var;
            end

            if(dist(t,j)<=Max_Range)

               if(j>1)

                  if(angle(t,j-1)<i)

                     hold on
                     plothandle=polar(angle(t,j-1),dist(t,j-1),'O');

                     set(plothandle,'markerfacecolor',[0,1,0],'markere
                     dgecolor',[0,1,0])
                     get( plothandle ) ;
                     set( plothandle, 'Color', [ 0.5, 1, 0 ] );
                     set ( plothandle, 'LineWidth', 2 );
                     hold off
```

```
                end
            end
        end

        if(dist(t,j)<=Max_Range)

            if(angle(t,j)>=i)

                hold on
                plothandle=polar(angle(t,j),dist(t,j),'O');

                set(plothandle,'markerfacecolor',[0,1,0],'markeredg
                ecolor',[0,1,0])
                get( plothandle ) ;
                set( plothandle, 'Color', [ 0.5, 1, 0 ] );
                set ( plothandle, 'LineWidth', 2 );
                hold off
            end
        end
    end
    pause(1*10^-14);
    end
 end
 end
```

Función *Mostrar_PPI*

```
function [ ] = Mostrar_PPI(Max_Range)

for i=(2*pi):-(pi/180):(pi/180)

    [x,y]=(pol2cart(i,Max_Range));
     z=complex(x,y);
     h=compass(z);
     ph=findall(gca,'type','patch');
     set(ph,'facecolor',[0,0.5,0]);
     set( h, 'Color', [0.5,1,0] );
     set( h, 'LineWidth', 4 );

     for k = 1:length(h)

         a = get(h(k), 'xdata');
         b = get(h(k), 'ydata');
         set(h(k), 'xdata', a(1:2), 'ydata', b(1:2)')
     end
end
view(90, -90)
end
```

Función *Mostrar_Menu*

```
function [opcion] = Mostrar_Menu( )

disp('*****************************************************************
***')
disp('*****************************************************************
***')
disp('***
***')
disp('***                    ELEGIR TIPO SIMULACION
***')
disp('***
***')
disp('***
***')
disp('***              1.- Cargar Data sobre Imagen Real
***')
disp('***
***')
disp('***              2.- Cargar Data sobre PPI
***')
disp('***
***')
disp('*****************************************************************
***')
disp('*****************************************************************
***')
disp('                                                                  ')
opcion=input('INGRESE EL NÚMERO DE SU ELECCIÓN:    ');

end
```

Función *Mostrar_Menu_para_Elegir_la_Trayectoria_del_Buque*

```
function Mostrar_Menu_para_Elegir_la_Trayectoria_del_Buque( )

disp('*****************************************************************
***')
disp('*****************************************************************
***')
disp('***
***')
disp('***          MENÚ PARA ELEGIR LA TRAYECTORIA DEL BUQUE
***')
disp('***
***')
disp('***
***')
disp('***              1.- TRAYECTORIA CIRCULAR
***')
disp('***
***')
disp('***              2.- TRAYECTORIA PARABÓLICA
***')
```

```
disp('***
***')
disp('***              3.- TRAYECTORIA ELÍPTICA (1/4 DE ELIPSE)
***')
disp('***
***')
disp('***              4.- TRAYECTORIA ELÍPTICA (3/4 DE ELIPSE)
***')
disp('***
***')
disp('***              5.- TRAYECTORIA DE UNA L INVERTIDA
***')
disp('***
***')
disp('***              6.- TRAYECTORIA RECTA
***')
disp('***
***')
disp('***
***')
disp('*********************************************************************
***')
disp('*********************************************************************
***')
disp('
')

end
```

Función *Mostrar_mensaje_al_usuario*

```
function Mostrar_mensaje_al_usuario( )

disp('
')
disp('*********************************************************************
***')
disp('*********************************************************************
***')
disp('***
***')
disp('***                        ¡ ATENCIÓN !
***')
disp('***
***')
disp('***    A CONTINUACIÓN SE LE SOLICITARÁ QUE INGRESE EL NÚMERO
***')
disp('***
***')
disp('***    DE BUQUES QUE PARTICIPARÁN EN LA ESCENA DE RADAR, POR
***')
disp('***
***')
disp('***    LO QUE DEBERÁ TENER EN CUENTA QUE SÓLO SE ADMITIRÁN
***')
disp('***
***')
```

```
disp('***    06 BUQUES COMO MÁXIMO PARA ESTA SIMULACIÓN.
***')
disp('***
***')
disp('*************************************************************
***')
disp('*************************************************************
***')
disp('
')

end
```

Función *Generador_de_posiciones_para_buques*

```
function [Tx,Ty,ptos] =
Generador_de_posiciones_para_buques(Num_Targets,trayectoria,aux_ptos)

  for i=1:1:Num_Targets

        if(trayectoria(i)==1)

           xc=8000; yc=-15000; r=10000; % Centro(xc,yc) y radio(r)
           n = 300; k=0:n; fi=2*pi*k/n;
           circular_pos_x=xc+r*cos(fi);
           circular_pos_y= yc+r*sin(fi);
           plot(circular_pos_x,circular_pos_y,'r');
           aux_ptos(i,1)=length(circular_pos_x);
           Tx1=zeros(2,aux_ptos(i,1));
           Tx1(2,1)=1;
           Ty1=zeros(2,aux_ptos(i,1));
           Ty1(2,1)=1;


           for j=1:1:aux_ptos(i,1)

               Tx1(1,j)=circular_pos_x(j);
               Ty1(1,j)=circular_pos_y(j);
           end

       end

        if(trayectoria(i)==2)

           t=linspace(-9500,19000,400);
           parabolica_pos_x=(t.*((t-60000).^2-90000000))./((t-
           60000).^2+90000000);
           parabolica_pos_y=(60000*((t-30000).^2-90000000))./((t-
           30000).^2+90000000);
           plot(parabolica_pos_x,parabolica_pos_y,'b')
           aux_ptos(i,1)=length(parabolica_pos_x);
           Tx2=zeros(2,aux_ptos(i,1));
           Tx2(2,1)=2;
           Ty2=zeros(2,aux_ptos(i,1));
           Ty2(2,1)=2;
```

```
    for j=1:1:aux_ptos(i,1)

        Tx2(1,j)=parabolica_pos_x(j);
        Ty2(1,j)=parabolica_pos_y(j);
    end

end

if(trayectoria(i)==3)

    t=pi/2: 0.01 :pi;
    UNCUARTO_eliptica_pos_x=40000*cos(t);
    UNCUARTO_eliptica_pos_y=20000*sin(t);
    plot(UNCUARTO_eliptica_pos_x,UNCUARTO_eliptica_pos_y,'m')
    aux_ptos(i,1)=length(UNCUARTO_eliptica_pos_x);
    Tx3=zeros(2,aux_ptos(i,1));
    Tx3(2,1)=3;
    Ty3=zeros(2,aux_ptos(i,1));
    Ty3(2,1)=3;

    for j=1:1:aux_ptos(i,1)

        Tx3(1,j)=UNCUARTO_eliptica_pos_x(j);
        Ty3(1,j)=UNCUARTO_eliptica_pos_y(j);
    end

end

if(trayectoria(i)==4)

    t=pi/2: 0.01 :2*pi;
    TRES_CUARTO_eliptica_pos_x=55000*cos(t);
    TRES_CUARTO_eliptica_pos_y=25000*sin(t);

   plot(TRES_CUARTO_eliptica_pos_x,TRES_CUARTO_eliptica_pos_y,'k
   ')
    aux_ptos(i,1)=length(TRES_CUARTO_eliptica_pos_x);
    Tx4=zeros(2,aux_ptos(i,1));
    Tx4(2,1)=4;
    Ty4=zeros(2,aux_ptos(i,1));
    Ty4(2,1)=4;

    for j=1:1:aux_ptos(i,1)

        Tx4(1,j)=TRES_CUARTO_eliptica_pos_x(j);
        Ty4(1,j)=TRES_CUARTO_eliptica_pos_y(j);
    end

end

if(trayectoria(i)==5)

    t=linspace(-30000,25000,400);
    Linvertida_pos_x=(t.*(t.^2-90000000))./(t.^2+90000000);
    Linvertida_pos_y=(60000*(t.^2-90000000))./(t.^2+90000000);
    plot(Linvertida_pos_x,Linvertida_pos_y,'b')
    aux_ptos(i,1)=length(Linvertida_pos_x);
    Tx5=zeros(2,aux_ptos(i,1));
    Tx5(2,1)=5;
```

```
        Ty5=zeros(2,aux_ptos(i,1));
        Ty5(2,1)=5;

        for j=1:1:aux_ptos(i,1)

            Tx5(1,j)=Linvertida_pos_x(j);
            Ty5(1,j)=Linvertida_pos_y(j);
        end

    end

    if(trayectoria(i)==6)

        x=linspace(-30000,20000,400);
        f=0.5.*x+20000;
        plot(x,f,'m')
        aux_ptos(i,1)=length(x);
        Tx6=zeros(2,aux_ptos(i,1));
        Tx6(2,1)=6;
        Ty6=zeros(2,aux_ptos(i,1));
        Ty6(2,1)=6;

        for j=1:1:aux_ptos(i,1)

            Tx6(1,j)=x(j);
            Ty6(1,j)=f(j);
        end
    end
end

 ptos=aux_ptos;
 Tx=zeros(Num_Targets,max(ptos));%Reserva Memoria
 Ty=zeros(Num_Targets,max(ptos));%Reserva Memoria


 for i=1:1:Num_Targets

     buscador=trayectoria(i);

     if(trayectoria(i)==1)
        if (buscador==Tx1(2,1)&& buscador==Ty1(2,1))

            for j=1:1:ptos(i)

                Tx(i,j)=Tx1(1,j);
                Ty(i,j)=Ty1(1,j);
            end
        end
     end

     if(trayectoria(i)==2)
        if (buscador==Tx2(2,1)&& buscador==Ty2(2,1))

            for j=1:1:ptos(i)

                Tx(i,j)=Tx2(1,j);
                Ty(i,j)=Ty2(1,j);
            end
```

```
            end
        end

        if(trayectoria(i)==3)
            if (buscador==Tx3(2,1)&& buscador==Ty3(2,1))

                for j=1:1:ptos(i)

                    Tx(i,j)=Tx3(1,j);
                    Ty(i,j)=Ty3(1,j);
                end
            end
        end

        if(trayectoria(i)==4)
            if (buscador==Tx4(2,1)&& buscador==Ty4(2,1))

                for j=1:1:ptos(i)

                    Tx(i,j)=Tx4(1,j);
                    Ty(i,j)=Ty4(1,j);
                end
            end
        end

        if(trayectoria(i)==5)
            if (buscador==Tx5(2,1)&& buscador==Ty5(2,1))

                for j=1:1:ptos(i)

                    Tx(i,j)=Tx5(1,j);
                    Ty(i,j)=Ty5(1,j);
                end
            end
        end

        if(trayectoria(i)==6)
            if (buscador==Tx6(2,1)&& buscador==Ty6(2,1))

                for j=1:1:ptos(i)

                    Tx(i,j)=Tx6(1,j);
                    Ty(i,j)=Ty6(1,j);
                end
            end
        end
    end
end
```

Función *Generar_data_para_simulación*

```
function [dist,Angle,X_para_imagen_real,Y_para_imagen_real] =
Generar_data_para_simulacion(Num_Targets,Tx,Ty,ptos,u,v)

if(Num_Targets==1)
```

```
   varxx=Tx.^2;
   varcs=Ty.^2;
   radic=varxx+varcs;
   dist=sqrt(radic);
   Angle_bruto=cart2pol(Tx,Ty);

   for h=1:1:ptos

        if(Angle_bruto(h)<0)

           Angle(h)=Angle_bruto(h)+(2*pi);

        end

        if(Angle_bruto(h)>=0)

           Angle(h)=Angle_bruto(h);

        end
    end

    for z=1:1:ptos

        kte(z)=(Tx(z)-Tx(1))/(Ty(z)-Ty(1));
    end

    for f=1:1:ptos

        Heading_one_target(f)=atan((u-(kte(f)*v))/(v+(kte(f)*u)));
    end

    for f=1:1:ptos

        t(f)=(Tx(f)-Tx(1)+Ty(f)-
        Ty(1))/((u*(cos(Heading_one_target(f))+sin(Heading_one_target(
        f))))+(v*(cos(Heading_one_target(f))-
        sin(Heading_one_target(f)))));
    end

    for w=1:1:ptos

        X_para_imagen_real(w)=(u*t(w)*cos(Heading_one_target(w)))-
        (v*t(w)*sin(Heading_one_target(w)))+Tx(1);

        Y_para_imagen_real(w)=(u*t(w)*sin(Heading_one_target(w)))+(v*t
        (w)*cos(Heading_one_target(w)))+Ty(1);

    end
end

if(Num_Targets>1)

   u_mt=u;
   v_mt=v;

   for i=1:1:Num_Targets
```

```
    varxx=Tx.^2;
    varcs=Ty.^2;
    radic=varxx+varcs;
    distancias=sqrt(radic);
    Angle_bruto=cart2pol(Tx,Ty);


    for h=1:1:ptos(i)

        if(Angle_bruto(i,h)<0)

           Angulos(i,h)=Angle_bruto(i,h)+(2*pi);
        end

        if(Angle_bruto(i,h)>=0)

           Angulos(i,h)=Angle_bruto(i,h);
        end
    end

    for k=1:1:ptos(i)

         dist(i,k)=distancias(i,k);
         Angle(i,k)=Angulos(i,k);
    end

    for w=1:1:ptos(i)

        k_mt(i,w)=(Tx(i,w)-Tx(i,1))/(Ty(i,w)-Ty(i,1));

    end

    for f=1:1:ptos(i)

        Heading_multiple_targets(i,f)=atan((u_mt(i)-
        (k_mt(i,f)*v_mt(i)))/(v_mt(i)+(k_mt(i,f)*u_mt(i))));

    end

    for f=1:1:ptos(i)

        t(i,f)=(Tx(i,f)-Tx(i,1)+Ty(i,f)-
         Ty(i,1))/((u_mt(i)*(cos(Heading_multiple_targets(i,f))+sin(
         Heading_multiple_targets(i,f))))+(v_mt(i)*(cos(Heading_mult
         iple_targets(i,f))-sin(Heading_multiple_targets(i,f)))));

    end

end

for i=1:1:Num_Targets

     for w=1:1:ptos(i)


         X_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*cos(Heading_multipl
         e_targets(i,w)))-
```

```
            (v_mt(i)*t(i,w)*sin(Heading_multiple_targets(i,w)))+Tx(i,1)
            ;

            Y_para_imagen_real(i,w)=(u_mt(i)*t(i,w)*sin(Heading_multipl
            e_targets(i,w)))+(v_mt(i)*t(i,w)*cos(Heading_multiple_targe
            ts(i,w)))+Ty(i,1);

        end
    end
end

end
```

Función *Cargar_Data_Sobre_Imagen_Real*

```
function
Cargar_Data_Sobre_Imagen_Real(Max_Range,Num_Targets,X_para_imagen_real
,Y_para_imagen_real)


if (Num_Targets==1)

   clc
   [Gauss2D,DDXYGauss2D,Gauss1D,DGauss1D,DDGauss1D,LK, TK] =
    gausianos(2.4,0.6);%desviación= 2.4 y PorcentajeDeLadoDeKernel=0.6
   Gaussiano=(Gauss2D.*4260)+22;

   indicadores=isnan(X_para_imagen_real);

   j=1;

   for i=1:1:length(X_para_imagen_real)

       if(indicadores(i)==1)

          posiciones_del_NaN(j)=i;
          j=j+1;

       end
   end

   aux=length(X_para_imagen_real)-length(posiciones_del_NaN);
   aux_X_para_imagen_real=zeros(aux,1);
   aux_Y_para_imagen_real=zeros(aux,1);
   conta=1;

   for i=1:1:length(X_para_imagen_real)

       var=1;

       for j=1:1:length(posiciones_del_NaN)

           if (i==posiciones_del_NaN(j))

               var=2;
```

```
        end
    end

    if(var==1)

       aux_X_para_imagen_real(conta,1)=X_para_imagen_real(i);
       aux_Y_para_imagen_real(conta,1)=Y_para_imagen_real(i);
       conta=conta+1;

    end
end

X_para_imagen_real=aux_X_para_imagen_real;
Y_para_imagen_real=aux_Y_para_imagen_real;

[Angle_bruto,distancia]=cart2pol( X_para_imagen_real,
 Y_para_imagen_real);

 for h=1:1:length(X_para_imagen_real)

     if(Angle_bruto(h)<0)

        Angle(h)=Angle_bruto(h)+(2*pi);

     end

     if(Angle_bruto(h)>=0)

        Angle(h)=Angle_bruto(h);

     end
 end

 Azimuth_Y=(Angle*1024)/(2*pi);
 Muestra_X=8192*(distancia/Max_Range);

 G1x=zeros(length(Muestra_X),1);
 G1y=zeros(length(Muestra_X),1);

for i=1:1:length(Muestra_X)

    G1x(i)=round(Muestra_X(i)-13);
    G1y(i)=round(Azimuth_Y(i)-13);
end

%load('D0_20140717-145400.mat')

for i=1:1:length(Muestra_X)

    load('D0_20140717-145400.mat')

    for j=1:1:length(Gaussiano)

        increX=0;

        if(j==1)
           increY=0;
```

```
            end

            if(j>1)

               increY=increY+1;
            end

            for k=1:1:length(Gaussiano)

                x=G1y(i)+increY;
                y=G1x(i)+increX;

                RawImage(x,y)=Gaussiano(j,k);
                increX=increX+1;

            end
        end

        save('RawImage')
        load('RawImage.mat')
        image(RawImage);colormap(jet(255))
        pause(0.003)

    end
end

if(Num_Targets>1)

    clc
    [Gauss2D,DDXYGauss2D,Gauss1D,DGauss1D,DDGauss1D,LK,TK] =
     gausianos(2.4,0.6);%desviación= 2 y PorcentajeDeLadoDeKernel=0.6
    Gaussiano=(Gauss2D.*4260)+22;

    indicadores=isnan(X_para_imagen_real);

    for p=1:1:Num_Targets

        conta=1;

        for z=1:1:length(indicadores)

            if (indicadores(p,z)==0)


               aux_X_para_imagen_real(p,conta)=X_para_imagen_real(p,z);

               aux_Y_para_imagen_real(p,conta)=Y_para_imagen_real(p,z);
                conta=conta+1;
            end
        end
    end

    X_para_imagen_real=aux_X_para_imagen_real;
    Y_para_imagen_real=aux_Y_para_imagen_real;

    [Angle_bruto,distancia]=cart2pol( X_para_imagen_real,
     Y_para_imagen_real);
```

```
for p=1:1:Num_Targets

    for h=1:1:length(X_para_imagen_real)

        if(Angle_bruto(p,h)<0)

           Angle(p,h)=Angle_bruto(p,h)+(2*pi);

        end

        if(Angle_bruto(p,h)>=0)

           Angle(p,h)=Angle_bruto(p,h);

        end
    end
end

 Azimuth_Y=(Angle*1024)/(2*pi);
 Muestra_X=8192*(distancia/Max_Range);


for p=1:1:Num_Targets

    l=1;
    for i=1:1:length(Muestra_X)

        if(Muestra_X(p,i)~=0 && Azimuth_Y(p,i)~=0)

           G1x(p,l)=round(Muestra_X(p,i)-13);
           G1y(p,l)=round(Azimuth_Y(p,i)-13);
           l=l+1;
        end
    end
end

%load('D0_20140717-145400.mat')

for i=1:1:length(G1x)

    load('D0_20140717-145400.mat')

    for p=1:1:Num_Targets

        increX=0;
        increY=0;

        if (G1x(p,i)>0 && G1y(p,i)>0 )

            for j=1:1:length(Gaussiano)

                for k=1:1:length(Gaussiano)

                    x=G1y(p,i)+increX;

                    y=G1x(p,i)+increY;
```

```
                    RawImage(x,y)=Gaussiano(j,k);
                    increX=increX+1;

                end
                increY=increY+1;
                increX=0;
            end
        end
    end
    save('RawImage')
    load('RawImage.mat')
    image(RawImage);colormap(jet(255))
    pause(0.003)
  end
 end
tiempo=abs(t)
end
```

Función *Gausianos*

```
% CALCULO DEL KERNEL 2D
% ---------------------
% El tamaño del kernel estará en función de la desviación estándard
elegida
% SG.  Para cada SG, se calcula LK y TK:
% LK es la distancia desde el centro del kernel hasta el último valor
% TK es el tamaño del kernel = 2 x LK + 1
%
%             [-LK, -(LK-1), ..., -2, -1, 0, 1, 2, ..., +(LK-1), +LK ]
%                                          |<----------- LK --------->|
%             |<------------------------ TK ------------------------->|
%
% SG--> desviación estándard
% PorcentajeDeLadoDeKernel--> porcentaje del lado del kernel a
% considerar=0.60 es un buen valor
% SINTAXIS:
%       [Gauss2D,DDXYGauss2D,Gauss1D,DGauss1D,DDGauss1D,LK, TK] =
gausianos(SG,PorcentajeDeLadoDeKernel)
function[Gauss2D,DDXYGauss2D,Gauss1D,DGauss1D,DDGauss1D,LK, TK] =
gausianos(DESV,PorcentajeDeLadoDeKernel)
      %---------------------------
      % recorte al tamaño del kernel % (0 a 1)
      LK =  ( 7.146 * DESV + 1.2832 ) /2 ;
      LK=round(PorcentajeDeLadoDeKernel*LK); if LK <1; LK =1;end
      TK  = (2*LK ) + 1 ;                    % Este es el tamaño del
      filtro gausiano (G, DG, o DDG)
      % CONSTANTES PARA DOS DIMENSIONES: 2D
      C1 =  +1 / (   2 * pi * DESV^2  );      % Primera constante 2D /
      Emplea DESV como desviación
      C2 =  -1 / (   2      * DESV^2  );      % Segunda constante 2D /
      Emplea DESV como desviación
      C3 =  -1 / (   2 * pi * DESV^4  );      % Tercera constante 2D /
      Emplea DESV como desviación
      C4 =  +1 / (   2 * pi * DESV^6  );      % Cuarta  constante 2D /
      Emplea DESV como desviación
      K1 =  +1/(sqrt(2*pi)*DESV);
      K2 =  -1/(2*DESV^2);
      K3 =  -1/(sqrt(2*pi)*DESV^3);
```

```
        K4 =  +1/(sqrt(2*pi)*DESV^5);

        Gauss2D       = -1.1 * ones(TK,TK);   % Reserva Memoria
        DXGauss2D     = Gauss2D;              % Reserva Memoria
        DYGauss2D     = Gauss2D;              % Reserva Memoria
        DDXYGauss2D   = Gauss2D;              % Reserva Memoria
        %centro=LK+1;

        % FILTROS
        for X=-LK:LK                          % Variable independiente
        AZIMUTH para 2D --> x
           for Y=-LK:LK                       % Variable dependiente
        RANGO   para 2D --> y
             Gauss2D    (Y+LK+1,X+LK+1) =        C1 *
        exp( C2 * (  X^2 + Y^2 ) );
             DXGauss2D  (Y+LK+1,X+LK+1) =        C3 *       X   *
        exp( C2 * (  X^2 + Y^2 ) );
             DYGauss2D  (Y+LK+1,X+LK+1) =        C3 *       Y   *
        exp( C2 * (  X^2 + Y^2 ) );
             DDXYGauss2D(Y+LK+1,X+LK+1) = ( 2 *  C3 + C4 * (X^2 + Y^2 )
        ) * exp( C2 * (  X^2 + Y^2 ) );
             %-------------------------------
           end
           Gauss1D (1 , X+LK+1)   = K1            * exp( K2 * X^2);
           DGauss1D (1 , X+LK+1)  = K3    *X      * exp( K2 * X^2);
           DDGauss1D (1 , X+LK+1) = (K3+K4*X^2)   * exp( K2 * X^2);
     end
end
```

Anexo C
Cómo usar el algoritmo

Al ejecutar el algoritmo principal aparecerá una ventana (ver figura C.1.) en donde se apreciarán cuatro botones y una tabla mostrando las velocidades características de los buques. Esta tabla de velocidades se tomará como referencia en el momento que el programa pida al usuario ingresar las velocidades de *"sway"* y *"surge"* de los buques, las que se encuentran entres los valores mostrados en la tabla.

Figura C.1. Ventana al inicio de la ejecución de la GUI.
Fuente: Elaboración propia.

Para fines de ejemplificar el uso del algoritmo, daremos un *click* en el botón:

Es oportuno indicar que el usuario es libre de elegir cualquier botón, de acuerdo a sus requerimientos.

Inmediatamente después de presionar este botón, el programa nos mostrará la consola de Matlab, en donde el usuario ingresará los datos que el programa le vaya pidiendo.

En primer lugar, se ingresa el alcance máximo del radar. En el presente trabajo de investigación se estudia un radar con un alcance máximo de 60 000 m; por lo que el usuario ingresa con el teclado el valor de 60 000.

Luego el programa muestra un menú al usuario, el que deberá ser leído detenidamente. (Ver figura C.2.).

Después de leer la información que brinda el menú, el usuario deberá ingresar el número de buques que participarán en la escena de radar. Para fines de ejemplificar el uso del algoritmo, elegiremos 06 que es el máximo número de buques que pueden participar en la escena de radar de acuerdo al programa.

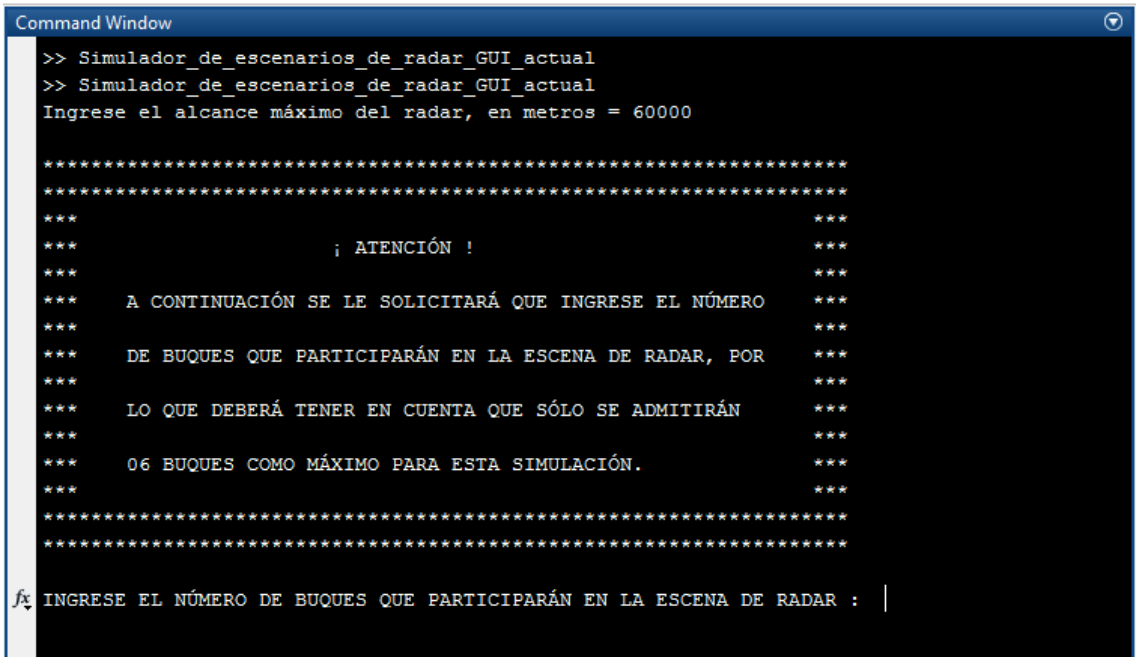


Figura C.2. Consola de Matlab donde el usuario ingresará los datos solicitados por el programa.
Fuente: Elaboración propia.

Si el usuario hubiese dado "*click*" a cualquiera de los otros tres botones, pudo haber elegido cualquier cantidad de buques; debido a que la cantidad de buques participantes en la escena de radar queda limitada únicamente por la carga computacional que se genera al ejecutar la simulación; es decir, a mayor número de buques que participen en la escena de radar, más tiempo tomará la ejecución de la simulación; dependiendo también del número de procesadores que posea el computador donde se ejecute la simulación.

Después de haber elegido la cantidad de buques que participarán en la escena de radar, el programa nos muestra un menú en donde se muestran las trayectorias que podrían tener los buques. (Ver figura C.3.). Queda a elección del usuario asignar las trayectorias a cada uno de los buques que participarán en la escena de radar.

Luego de haber elegido las trayectorias para cada uno de los buques, el programa pide al usuario el ingreso de las velocidades de "*sway*" y "*surge*" de los mismos.

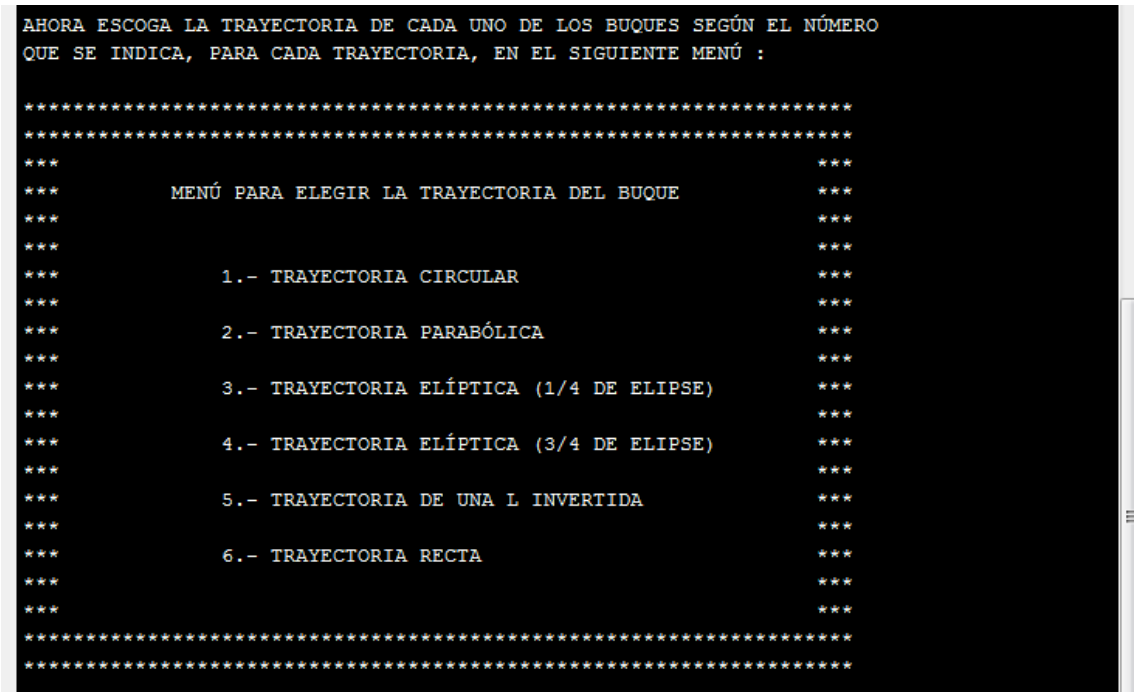


Figura C.3. Menú mostrado en consola para elegir la trayectoria de cada uno de los buques.
Fuente: Elaboración propia.

Para el ingreso de velocidades de los buques que participan en la escena de radar, se tendrá como referencia los valores de la tabla de velocidades que aparece en la ventana principal de la GUI. (Ven figura C.4.)

Tipo de Buque	Nombre	Velocidad (Nudos)
Fragata Lanzamisiles	Canarias F86	29
Fragata Lanzamisiles	Victoria F82	29
Fragata Lanzamisiles	Santa María F81	29
Fragata Lanzamisiles	Clase Perry (FFG-7)	30
Fragata Portahelicópteros	Cristobal Colón (F-105)	18
Fragata Lanzamisiles	Fremm	27
Fragata Misilera	Blohm Vass F-125	20
Fragata Misilera	Fridtjof Nansen	27

Figura C.4. Tabla de velocidades de cada uno de los buques.
Fuente: Elaboración propia.

Después del ingreso de las velocidades, en la ventana de la GUI aparece un PPI mostrando las trayectorias elegidas por el usuario. (Ver figura C.5.)

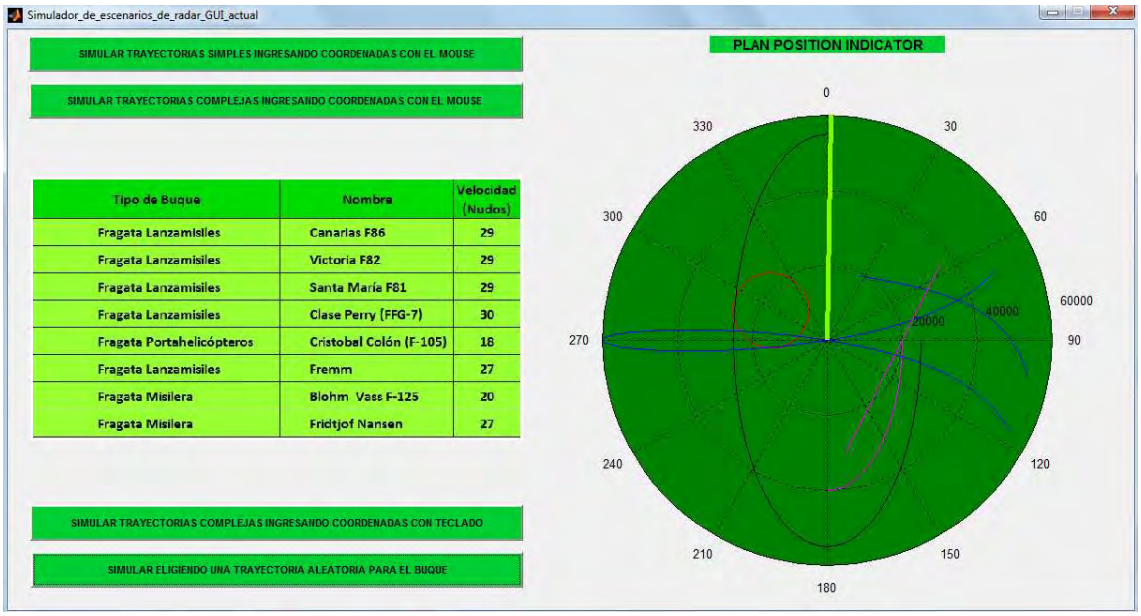


Figura C.5. PPI mostrando la trayectoria de cada uno de los buques, elegidas por el usuario.
Fuente: Elaboración propia.

Y en la consola de Matlab se muestra un menú para elegir el tipo de simulación que se ejecutará. (Ver figura C.6); siendo ésta la última acción para que el algoritmo muestre los resultados de la simulación.

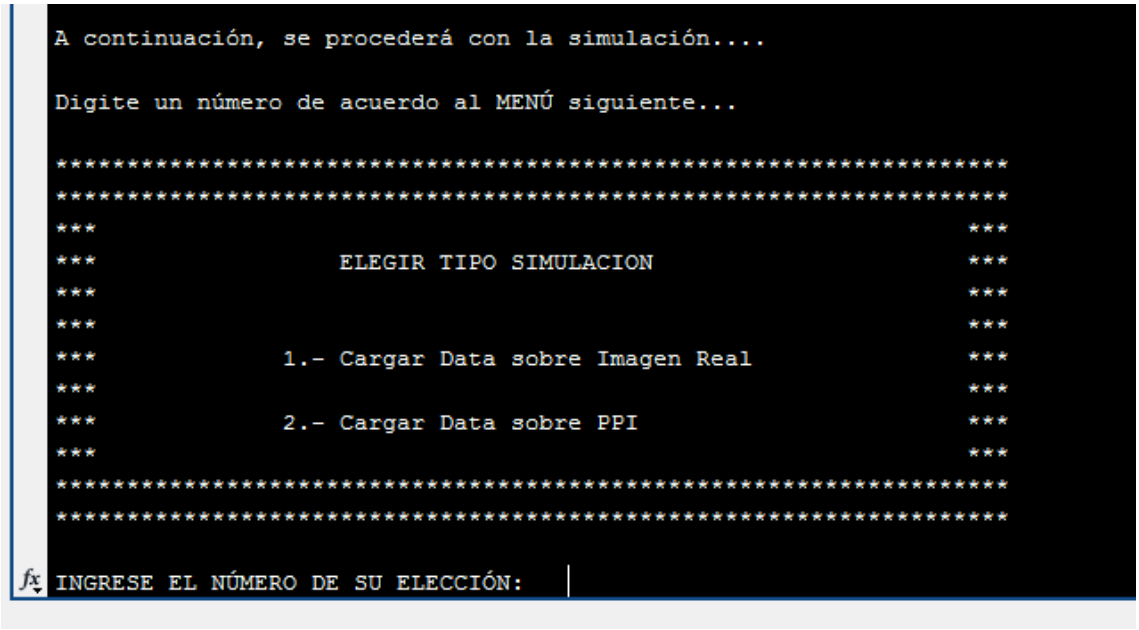


Figura C.6. Consola de Matlab. Se muestra menú al usuario para elegir el tipo de simulación que se ejecutará.
Fuente: Elaboración propia.

Si el usuario eligió la opción 1 del menú (Cargar Data sobre Imagen Real), en seguida aparecerá una ventana mostrando una imagen real obtenida por un radar de superficie

SPQ. Sobre esta imagen se plotean las posiciones, calculadas por el algoritmo, para cada uno de los buques que participan en la escena de radar. (Ver figura)

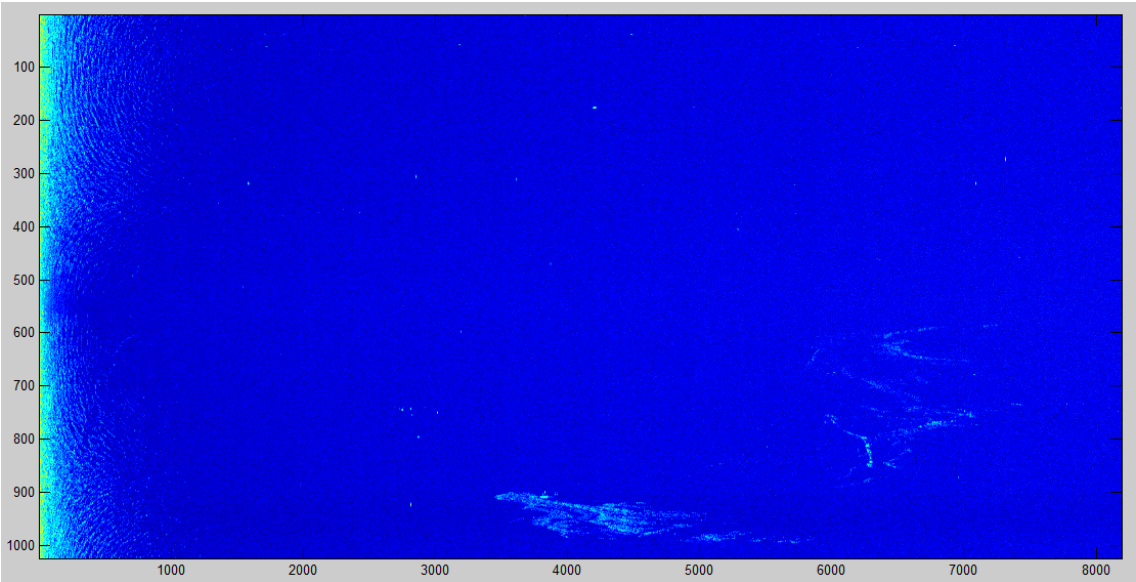


Figura C.7. Imagen real de escena de radar, obtenida por un radar SPQ.
Fuente: Elaboración propia.

Si el usuario eligió la opción 2 del menú (Cargar Data sobre el PPI), en seguida en la ventana de la GUI aparecerá un PPI mostrándonos las posiciones, calculadas por el algoritmo, de cada uno de los buques. (Ver figura C.8.)

Figura C.8. PPI donde se muestran los resultados de la simulación.
Fuente: Elaboración propia.

Ahora describiremos las funciones de los otros tres botones. Comencemos con el botón:

SIMULAR TRAYECTORIAS SIMPLES INGRESANDO COORDENADAS CON EL MOUSE

Al dar *click* a este botón, el algoritmo comienza a ejecutarse e inmediatamente le pide al usuario que ingrese el alcance máximo del radar. Para nuestro caso se ingresa 60 000 m debido a que el radar que se estudia en la presente investigación tiene un alcance máximo de 60 000 m. Sin embargo, el algoritmo acepta otro valor de alcance máximo que el usuario desee ingresar y realiza los cálculos en base a ese valor.

Luego de ingresar el valor de alcance máximo de radar, el usuario da un ENTER con la finalidad de continuar con la ejecución del algoritmo. En seguida el programa le pide al usuario que ingrese el número de buques que participarán en la escena de radar; por lo tanto, queda a elección del usuario el número de buques que participarán en la escena de radar; el algoritmo está diseñado para admitir cualquier cantidad de buques y lo único que limitaría esta cantidad es la carga computacional que se generaría al momento de mostrar los resultados de la simulación.

Después de esto, el usuario da un ENTER para continuar con la ejecución del algoritmo. Inmediatamente después el programa le pide al usuario que ingrese el número de puntos que usará para aproximar cada una de las trayectorias de los buques. En esta parte del programa, el algoritmo usa una función de Matlab llamada *"spline"*; esta función usa un conjunto de puntos ingresados por el usuario para ajustarlos mediante una curva "*spline"* que sigue la tendencia de los puntos.

Enseguida, el usuario presiona ENTER para continuar con la ejecución del algoritmo e inmediatamente en la ventana de la GUI aparecerá un cursor sobre un PPI, esperando que el usuario ingrese los puntos para aproximar mediante "*splines"* las trayectorias de cada uno de los buques. (Ver figura C.9).

Luego que el usuario ingresa los puntos mediante el *"mouse"* se dibujan las trayectorias sobre el PPI como se muestran en la figura C.10.

Inmediatamente después, en la consola de Matlab el programa muestra un menú al usuario pidiéndole que elija donde se mostrarán los resultados de la simulación. Por último, si el usuario elige la opción 1 (Cargar Data sobre Imagen Real) aparecerá una ventana mostrando una imagen real obtenida por un radar de superficie SPQ. Sobre esta imagen se plotean las posiciones, calculadas por el algoritmo, para cada uno de los buques que participan en la escena de radar. (Ver figura C.11).

Si el usuario elige la opción 2 del menú (Cargar Data sobre el PPI), en seguida en la ventana de la GUI aparecerá un PPI mostrándonos las posiciones, calculadas por el algoritmo, de cada uno de los buques dando fin a la simulación. (Ver figura C.12).

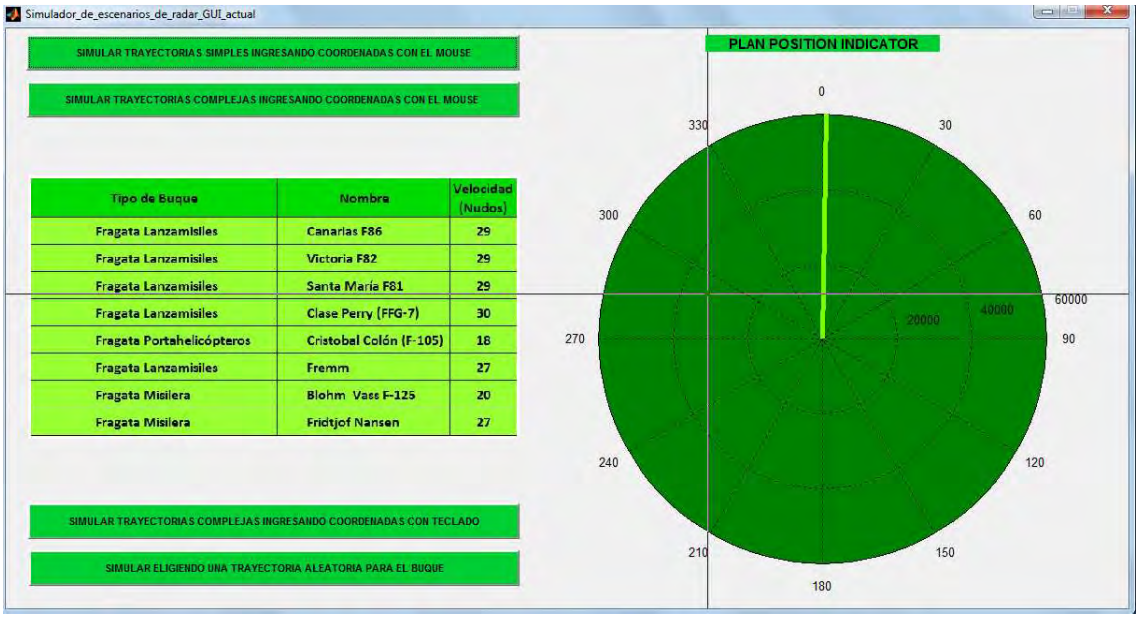


Figura C.9. Se muestra un cursor sobre el PPI esperando que el usuario ingrese los puntos, mediante el "*mouse*", con los que aproximará cada una de las trayectorias de los buques.
Fuente: Elaboración propia.

Figura C.10. Se muestra las trayectorias de los buques, aproximadas con "*splines*".
Fuente: Elaboración propia.

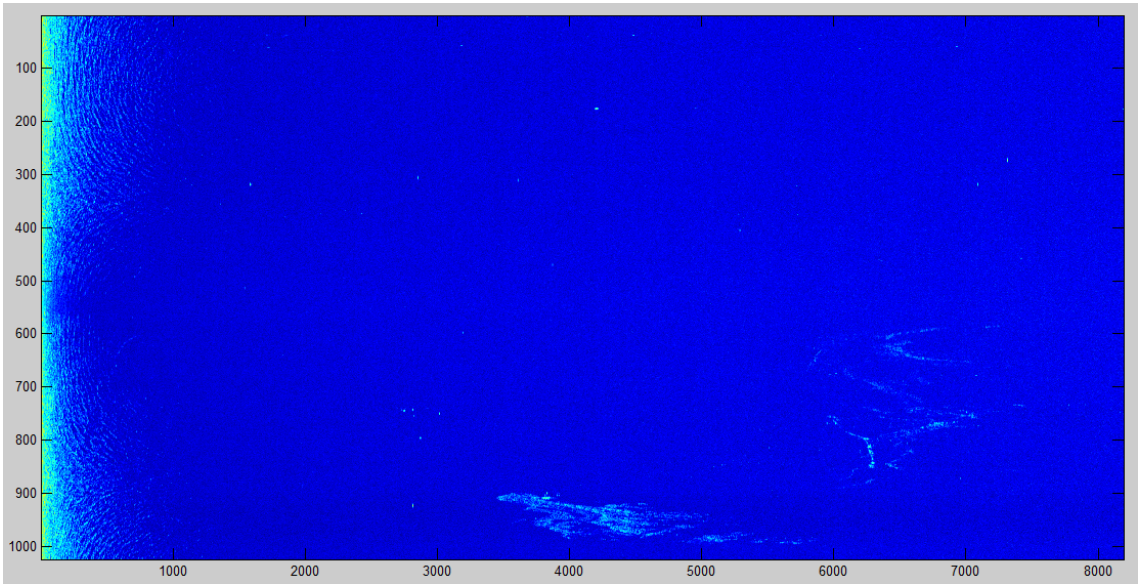


Figura C.11. Imagen real de escena de radar, obtenida por un radar SPQ.
Fuente: Elaboración propia.

Figura C.12. PPI donde se muestran los resultados de la simulación.
Fuente: Elaboración propia.

Por otro lado, si el usuario le da un click al botón:

ó al botón:

podrá ejecutar el algoritmo y de una manera simular seguir los pasos ya explicados anteriormente para el ingreso de datos, con la finalidad de ilustrar los resultados de la simulación sobre un PPI o sobre una imagen real de radar.