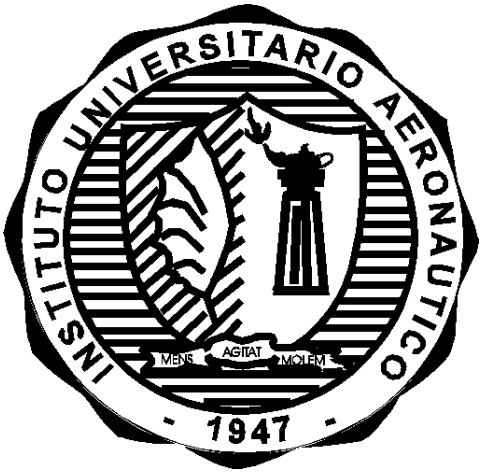

INSTITUTO UNIVERSITARIO AERONAUTICO

DESCRIPCIÓN DEL ALGORITMO DE PATH FOLLOWING POR CAMPO VECTORIAL

Informe Técnico: DMA-011/16 **Revisión:** /

Proyecto: PIDDEF 038/14 – Paracaídas Comandado Autónomo

Fecha: 20/12/2016

Autor:

Ing. Esteban Gonzalez Garcia

Revisó:

Ing. Andres Liberatto

DESCRIPCIÓN DEL ALGORITMO DE PATH FOLLOWING POR CAMPO VECTORIAL

Por:

Esteban Gonzalez Garcia

RESUMEN

En el marco del proyecto PIDDEF 038/14 – “Paracaídas Comandado Autónomo”, se describen dos algoritmos de *path following* para líneas rectas y órbitas circulares. Estos algoritmos permiten calcular, para cualquier posición de un vehículo en el espacio, un vector de rumbo deseado que lo llevará a converger con la trayectoria pre-definida.

Córdoba, 20 de diciembre de 2016

INDICE

Pág.

RESUMEN....1

INDICE....2

1. INTRODUCCIÓN....3

2. DESCRIPCIÓN DEL PROBLEMA....3

3. DESCRIPCIÓN DEL ALGORITMO....4

3.1. Seguimiento de líneas rectas....4

3.2. Seguimiento de órbitas circulares....6

1. CONCLUSIONES....9

2. REFERENCIAS....9

1. INTRODUCCIÓN

Una de las capacidades primordiales de los Vehículos Aéreos no Tripulados (UAV) es la capacidad de navegar de manera autónoma por una serie de puntos prefijados o *waypoints*. En un paracaídas comandado autónomo esto es prácticamente su única función ya que su objetivo es llevar la carga paga a un punto prefijado de manera precisa. El automatismo de control del cual depende el cumplimiento del objetivo puede dividirse, desde el punto de vista de la lógica de programación, en tres capas bien diferenciadas (ver Figura 1).

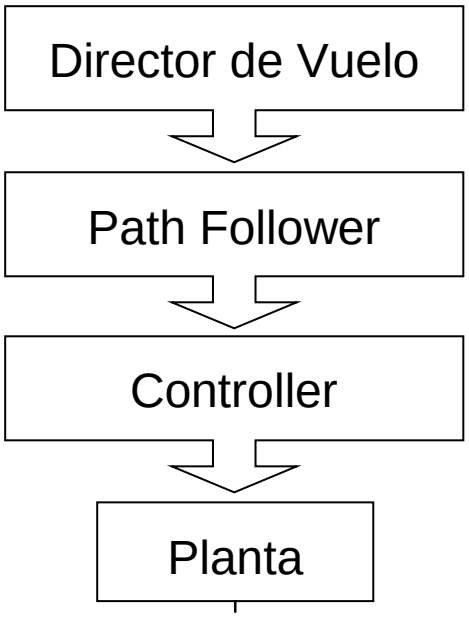

Figura 1 – Estructura en capas del control

La capa superior o Director de Vuelo es la encargada de gestionar la estrategia global del vuelo autónomo, tomando las decisiones relevantes a la misión en función de parámetros como el viento, la distancia al objetivo o la senda de planeo actual. La salida de esta capa es un plan de vuelo (trayectorias rectas o curvas) y un conjunto de parámetros de vuelo que pueden ir cambiando en el tiempo y serán ejecutados por las capas inferiores.

La capa intermedia o *Path Follower* es la encargada de procesar el plan de vuelo y calcular la forma de aproximar el vehículo a la trayectoria comandada por el Navegador. Esta aproximación es dinámica ya que depende de la posición actual del vehículo respecto a la trayectoria deseada y del viento. La salida de esta capa es un rumbo deseado instantáneo cuyo valor dependerá del algoritmo específico implementado.

La capa inferior o *Controller* es la encargada de ejecutar los mandos de la aeronave para seguir un rumbo prefijado por la capa anterior. El tipo de controlador y sus parámetros de configuración son independientes de las otras capas.

2. DESCRIPCIÓN DEL PROBLEMA

Cuando un vehículo autónomo es comandado para dirigirse a una coordenada específica en el espacio, existen infinitos caminos que el vehículo puede seguir y cumplir con su objetivo. Luego, la decisión de qué camino tomar para llegar a esa coordenada ha sido abordada de diferentes maneras. Los sistemas más estrictos están basados en *Trajectory Tracking*, que impone una trayectoria que el vehículo debe seguir no solamente asociado al espacio sino también al tiempo en que debe pasar por cada punto. Si bien esto en casos como la robótica puede ser algo fundamental y quizás la mejor manera de abordar ciertos problemas, para vehículos aéreos, y particularmente

para los de pequeño porte, esto impone restricciones difíciles de cumplir en vista a las perturbaciones externas que sufren. La alternativa que más ha cobrado fuerza en los desarrollos de autopilotos para vehículos autónomos, tanto aéreos como acuáticos, es la de *Path Following*, en la cual se mantiene una trayectoria que el vehículo debe seguir en el espacio pero se quita la restricción del tiempo. El vehículo debe seguir una trayectoria o camino (*path*) pero lo hará de la manera que su dinámica y las perturbaciones externas se lo permitan. El problema entonces se reduce a encontrar, para cualquier punto del espacio en donde el vehículo pueda estar posicionado, el rumbo deseado para alcanzar el objetivo de unirse a la trayectoria prefijada. Los algoritmos que calculan ese rumbo deseado se llaman de *Path Following* y existen numerosos planteos diferentes para abordar el problema.

- Directo (o *Carrot Following*)
- *Non-linear guidance law (NLGL)*
- *Vector Field (VF)*
- *Linear Quadratic Regulator (LQR Path Following)*

A continuación se muestra un desarrollo realizado por Nelson et al[1], en el cual se plantea un algoritmo basado en el cálculo de un campo vectorial (Vector Field) ad hoc para pequeños vehículos aéreos.

3. DESCRIPCIÓN DEL ALGORITMO

El objetivo del algoritmo es calcular un campo vectorial para dirigir un vehículo aéreo hacia una trayectoria fija pre-programada. Los vectores en el campo espacial apuntan hacia el camino deseado de la aeronave para converger en la trayectoria de vuelo. El método se puede aplicar tanto para trayectorias rectas como circulares. Esto en la práctica produce escasas limitaciones, ya que casi cualquier plan de vuelo puede ser descripto mediante un conjunto de líneas rectas y arcos.

3.1. Seguimiento de líneas rectas

Considérese una línea recta formada por dos puntos (*Waypoints*) en el espacio plano (altura constante). En adelante llamaremos a esta línea *track*. Para que un vehículo aéreo siga la trayectoria definida por este *track* se debe construir un campo vectorial con los rumbos deseados. Cuando el vehículo está lejos lateralmente del *track* (siendo “lejos” una definición arbitraria que queda a criterio del usuario), el objetivo es aproximarse de manera directa. A medida que el vehículo se acerca al *track*, el objetivo transiciona entre aproximación al *track* y seguimiento del mismo. La región de transición alrededor del *track* está señalada por líneas de puntos en la Figura 2. Fuera de la región de transición el rumbo deseado o ángulo de ingreso χ^e es constante. Dentro de la región el rumbo deseado cambia desde el ángulo de ingreso χ^e al rumbo del track χ^f. La tasa de cambio de este ángulo se controla mediante una ganancia *k*.

Variable	Descripción
χ^f	Rumbo del *track* (del *Waypoint* 1 al 2)
χ^e	Angulo de ingreso, $(0 < \chi^e < \frac{\pi}{2})$
χ^d	Rumbo deseado calculado
$w_1, w_{1x} w_{1y}$	*Waypoint 1* y sus componentes
$z = (x, y)^T$	Posición actual del vehículo
s^*	Parámetro indicativo del porcentaje de avance sobre el *track*, $s^* \in [0,1]$
ϵ	Error lateral (distancia del vehículo al *track*)
τ	Distancia lateral al *track* a la cual comienza la transición
k	Ganancia de transición

Tabla 1 – Definición de variables para algoritmo del campo vectorial para una línea recta

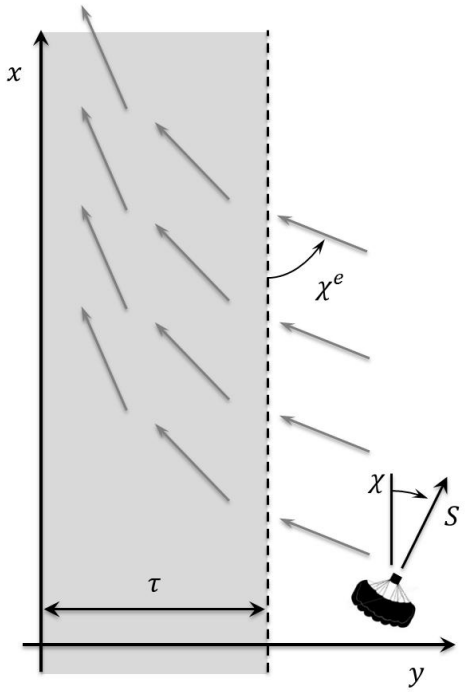

Figura 2 – Esquema del campo vectorial para y > 0

La Tabla 1 presenta la lista de variables involucradas en el desarrollo del algoritmo. La idea básica del mismo es situar al vehículo respecto al *track* en el espacio y luego calcular el rumbo deseado resultante del campo vectorial propuesto. Los parámetros del campo vectorial deben ser ajustados según las capacidades del vehículo de manera que alcanzar la performance deseada. El algoritmo escrito en pseudocódigo se puede ver en la Tabla 2.

Algoritmo 1: Construcción de un campo vectorial para una línea recta (altitud constante)
01: $\chi^f = atan2(w_{2y} - w_{1y}, w_{2x} - w_{1x})$ *Cálculo del rumbo del track*
02: $s^* = \frac{(z-w_1)^T(w_2-w_1)}{\|w_2-w_1\|^2}$ *Cálculo del grado de avance del vehículo sobre el track*
03: $\rho = sign[(w_2 - w_1) \times (z - w_1)]$ *Cálculo del lado del track en que se encuentra el vehículo*
04: **if** $s^* > 1$ **then** *Si el vehículo se encuentra más allá del segundo waypoint*
05: ***Switch to next waypoint*** *Pasar al siguiente punto del plan de vuelo*
06: **end if**
07: $\epsilon = \|z - (s^*(w_2 - w_1) + w_1)\|$ *Calcula la distancia del vehículo al track*
08: **if** $\epsilon > \tau$ **then** *Si la distancia del vehículo al track es mayor que la prefijada para la transición*
09: $\chi^d = \chi^f - \rho\chi^e$ *Calcula rumbo deseado en aproximación directa*
10: **else**
11: $\chi^d = \chi^f - \rho\chi^e \left(\frac{\epsilon}{\tau}\right)^k$ *Calcula rumbo deseado en transición suave al track*
12: **end if**

Tabla 2 – Algoritmo del campo vectorial para una línea recta

A modo de ejemplo se propone un track con origen [-100,-100] y destino [100,100]. Se elige una distancia de transición de 75 [m], un ángulo de ingreso de 90º y una ganancia de transición de 0.8. El campo vectorial resultante se muestra en la Figura 3.

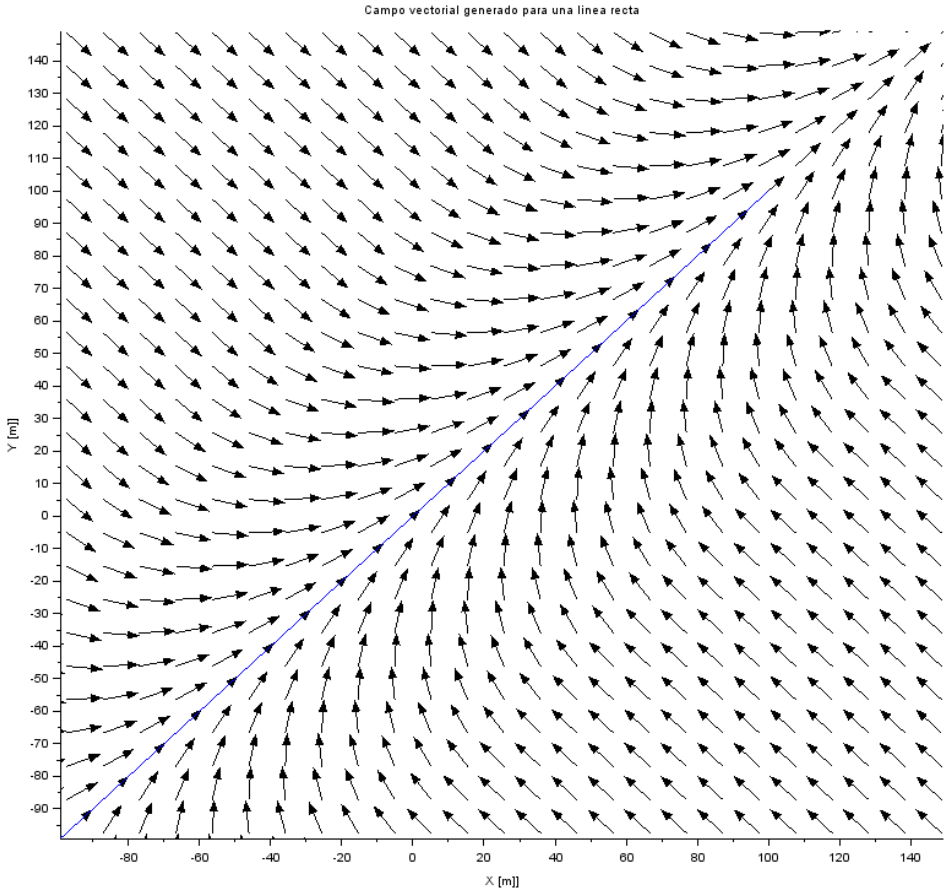

Figura 3 – Ejemplo de campo vectorial

3.2. Seguimiento de órbitas circulares

Considérese un círculo formada por un punto central (*Waypoint*) en el espacio plano (altura constante) y un radio. En adelante llamaremos a esta línea *órbita*. Para que un vehículo aéreo siga la trayectoria definida por esta órbita se debe construir, como se hizo en el caso de la línea, un campo vectorial con los rumbos deseados. Cuando el vehículo está lejos radialmente de la órbita (siendo “lejos” una definición arbitraria que queda a criterio del usuario), el objetivo es aproximarse de manera directa al centro. A medida que el vehículo se acerca a la órbita, el objetivo transiciona entre aproximación y seguimiento de la misma. La región de transición alrededor de la órbita está señalada por líneas de puntos en la Figura 4. Fuera de la región de transición el rumbo deseado es constante y apunta al centro del círculo. Dentro de la región el rumbo deseado cambia desde el rumbo directo al centro al rumbo de la órbita, que en este caso es a su vez función de la posición del vehículo en el espacio. La tasa de cambio de este ángulo en la zona de transición se controla, igual que antes, mediante una ganancia *k*.

Variable	Descripción
r	Radio de la órbita
$c = (c_x, c_y)^T$	Posición actual del vehículo
$z = (x, y)^T$	Posición actual del vehículo
k	Ganancia de transición
χ^d	Rumbo deseado calculado
d	Distancia del centro de la órbita al vehículo
ρ	Parámetro que indica si el vehículo está dentro o fuera de la órbita
γ	Vector desde el centro de la órbita al vehículo

Tabla 3 – Definición de variables para algoritmo del campo vectorial para una órbita

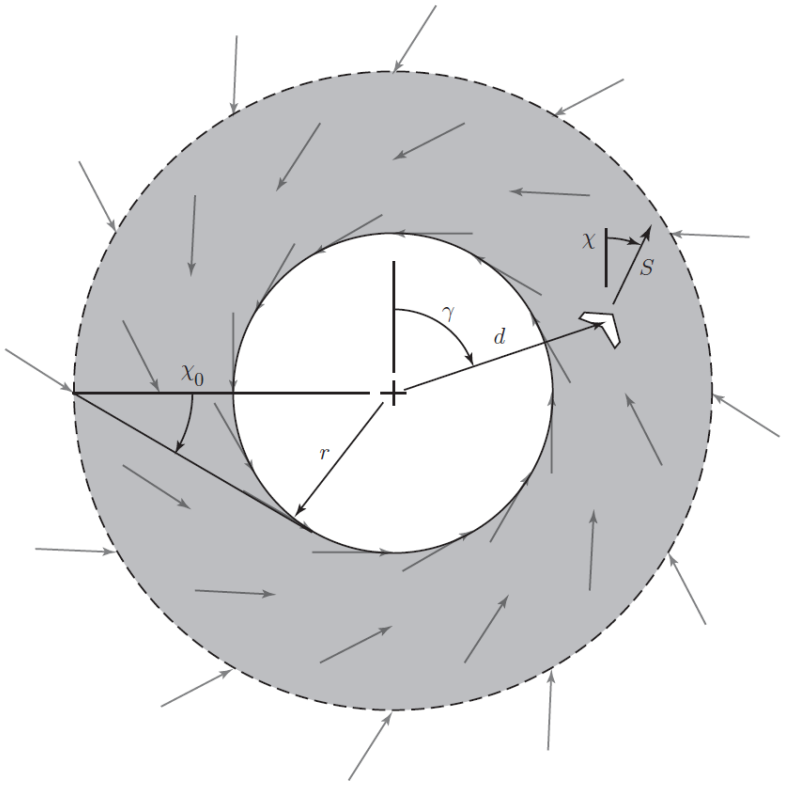

Figura 4 – Esquema del campo vectorial para una órbita

La Tabla 3 presenta la lista de variables involucradas en el desarrollo del algoritmo. La idea básica del mismo es situar al vehículo respecto a la órbita en el espacio y luego calcular el rumbo deseado resultante del campo vectorial propuesto. Los parámetros del campo vectorial deben ser ajustados según las capacidades del vehículo de manera que alcanzar la performance deseada. El algoritmo escrito en pseudocódigo se puede ver en la Tabla 4.

A modo de ejemplo se propone una órbita con origen [0,0] y radio 40 [m]. Se mantiene una distancia de transición de $2r$ y se elige una ganancia de transición de 0.8. El campo vectorial resultante se muestra en la Figura 5Figura 3.

Algoritmo 2: Construcción de un campo vectorial para una órbita (altitud constante)
01: $d = \|z - c\|$ *Cálculo de la distancia del vehículo al centro de la órbita*
02: **if** $d - r > 0$ **then** *Si el vehículo se encuentra fuera de la órbita*
03: $\rho = 1$
04: **else** *Si el vehículo se encuentra dentro de la órbita*
05: $\rho = -1$
06: **end if**
07: **if** $\|d\| > 2r$ **then** *Si la distancia del vehículo al centro es mayor a 2 radios*
08: $\chi^d = \gamma - \frac{5\pi}{6}$ *Calcula rumbo deseado en aproximación quasi-directa*
09: **else**
10: $\chi^d = \gamma - \frac{\pi}{2} - \rho\frac{\pi}{3}\left(\frac{\|d-r\|}{r}\right)^k$ *Calcula rumbo deseado en transición suave la órbita*
11: **end if**

Tabla 4 – Algoritmo del campo vectorial para una línea recta

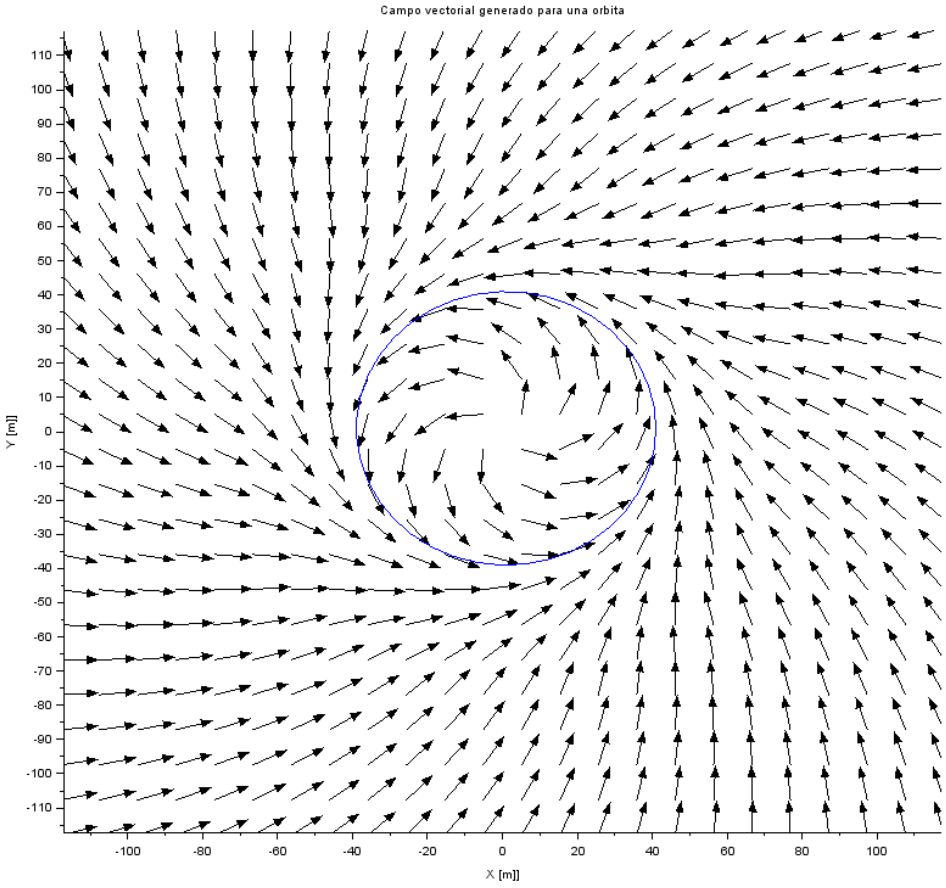

Figura 5 – Ejemplo de campo vectorial

1. CONCLUSIONES

En el presente informe se ha presentado dos algoritmos de *Path Following* basados en campo vectoriales para la determinación de rumbo convergente a trayectorias rectas y órbitas circulares. Estos algoritmos constituyen una primera opción a evaluar en el desarrollo de control de paracaídas guiados. Como trabajo futuro se plantea la posibilidad de evaluar otros métodos y realizar análisis comparativos tomando como referencia la dinámica de un paracaídas a fin de seleccionar el más adecuado.

2. REFERENCIAS

1. **Nelson, D., Barber, B., McLain, T., and Beard, R.** *Vector Field Path Following for Small Unmanned Aerial Vehicles.* Proceedings of the American Control Conference, pp. 5788-5794, June 2006.

ANEXO A

PROGRAMA DE SCILAB PARA CALCULAR EL CAMPO VECTORIAL PARA UNA LINEA RECTA

```
clear
//Funcion para calcular el rumbo de un vector
function [z]=bearing(a, b)
    br=atan(a,b);
    if br>0 then
        z = br;
    else
        z = 2*%pi + br;
    end
endfunction

// Creacion de la grilla para el campo vectorial
clf()
n = 40;
x = linspace(-200,200,n);
y = linspace(-200,200,n);
[X,Y] = meshgrid(x,y);

//Definicion de track con dos waypoints XY
wp1 =[-100 -100 0];
wp2 =[100 100 0];
Track = wp2 - wp1;

//Parametros del algoritmo
Tau = 75; //Crosstrack threshhold distance
Xe = %pi/2; //Entry angle
k = 0.8; //Transition gain

//Calculo del heading deseado en cada punto de la grilla
for i=1:n
    for j=1:n
        z = [x(i) y(j) 0]; //Posicion del UAV instantanea
        Xf = bearing(Track(1),Track(2)); //Rumbo del track Wp1-Wp2
        Sstar= (z-wp1)*(wp2-wp1)'/norm(wp2-wp1)^2; //Porcentaje del track recorrida (0-1)
        rho_aux = -sign(cross(wp2-wp1,z-wp1)); //De que lado de track estoy
        rho = rho_aux(3);
        epsilon = norm(z-(Sstar*(wp2-wp1)+wp1)); //Error de crosstrack
        if abs(epsilon) > Tau //Si estoy mas lejos que Tau entro con Xe
                Xd = Xf - rho*Xe;
        else                    //Si estoy mas cerca que Tau converge suave
                Xd = Xf - rho*Xe*(epsilon/Tau)^k;
        end
        fx(i,j) = sin(Xd);
        fy(i,j) = cos(Xd);
    end
end

champ(x,y,fx,fy,arfact=0.8);
xpts = [wp1(1) wp2(1)];
ypts = [wp1(2) wp2(2)];
plot(xpts,ypts);
xtitle('Campo vectorial generado para una linea recta','X [m]]','Y [m]]');
```

ANEXO B

PROGRAMA DE SCILAB PARA CALCULAR EL CAMPO VECTORIAL PARA UNA ORBITA

```
clear
//Funcion para calcular el rumbo de un vector
function [z]=bearing(a, b)
    br=atan(a,b);
    if br>0 then
        z = br;
    else
        z = 2*%pi + br;
    end
endfunction

// Creacion de la grilla para el campo vectorial
clf()
n = 40;
a = linspace(0, 2*%pi, n);
x = linspace(-200,200,n);
y = linspace(-200,200,n);
[X,Y] = meshgrid(x,y);

//Definicion de la orbita con centro y radio
r = 40;
c =[1 1 0];
xc = c(1) + r*cos(a); //Curva para el grafico
yc = c(2) + r*sin(a);

//Parametros del algoritmo
k = 1; //Transition gain

//Calculo del heading deseado en cada punto de la grilla
for i=1:n
    for j=1:n
        z = [x(i) y(j) 0]; //Posicion del UAV instantanea
        d = norm(z-c); //Modulo de la distancia al centro
        dvect = z - c; //Vector UAV-Centro
        rstar = d -r;
        if rstar > 0  //Si el vehículo se encuentra fuera de la órbita
            rho = 1;
        else          //Si el vehículo se encuentra dentro de la órbita
            rho = -1;
        end
        if d > 2*r then //Si el vehiculo esta fuera del circulo
            Xd = bearing(dvect(1),dvect(2)) - %pi*5/6;
        else            //Si el vehiculo esta dentro
            Xd = bearing(dvect(1),dvect(2)) - %pi/2 - rho*%pi/3*(norm(d-r)/r)^k;
        end
        fx(i,j) = sin(Xd);
        fy(i,j) = cos(Xd);
    end
end

champ(x,y,fx,fy,arfact=0.8);
plot(xc,yc);
xtitle('Campo vectorial generado para una orbita','X [m]]','Y [m]]');
```