Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

A3 - Sistemas y señales I

Trabajo práctico N^o1:

Introducción a Matlab

Asistencia al laboratorio: *Lunes 10 de Abril 2023*
Fecha de entrega: *Lunes 8 de Mayo 2023*

Federico, Scheytt S-5268/1
Joaquín, Gonzalez Targon G-5767/3

Índice

Introducción

En este trabajo práctico se plantean dos problemas cuya resolución y planteamiento se dan a partir de conceptos teóricos adquiridos en el cursado de la asignatura y mediante el uso de herramientas de cómputo numérico como lo es Matlab (Matrix Laboratory). El programa Matlab (correspondiente a la abreviación de Matrix Laboratory) es una potente herramienta de cálculo numérico y visualización de uso muy difundido a nivel mundial en los ambientes industrial y académico en tareas de investigación, desarrollo y diseño en diversas áreas tales como procesamiento de señales, control y comunicaciones.

En el primer problema nos encontramos con un oscilador de Van der Pol, compuesto por un inductancia L, un capacitor C y un diodo túnel, el cual está alimentado con una fuente de corriente continua. Nos interesa en particular uso de variables de estado para la representación de las ecuaciones físicas que caracterizan al sistema, las cuales luego pueden ser representados por ecuaciones en diferencias mediante el uso de la aproximación de la primera derivada usando el método de Euler, pudiendo así calcular las magnitudes de interés en dicho circuito.

En el segundo problema tenemos un dispositivo para sensar el exceso de velocidad de los automóviles. Su funcionamiento se basa en obtener dos diferentes posiciones del vehículo y dividiendo respecto al tiempo entre mediciones obtenemos la velocidad del mismo. Para obtener la posición de dicho automóvil se envía una señal conocida la cual se refleja en el objeto, vuelve al emisor atenuada, con un retardo y ruido, el tiempo de este retardo es el que nos indica en qué posición se encuentra el objeto. Para poder determinar dicho retardo vamos a utilizar técnicas de correlación para poder abstraernos del ruido y poder solamente analizar la señal de interés.

1. Oscilador de Van der Pol.

El oscilador de Van der Pol es un circuito compuesto por una inductancia L, un capacitor C, una fuente de corriente continua I_0 y un diodo túnel conectados en paralelo (Figura 1).

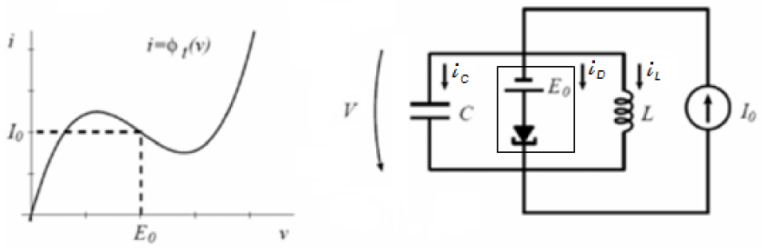

Figura 1: Circuito oscilador de Van der Pol

La relación VA del diodo es de tipo cubica, de la forma:

$$i_D = \gamma\nu^3 - \alpha\nu$$

A partir de realizar la LKC en el circuito, derivar la ecuación respecto al tiempo y aplicar la igualdad de tensiones debido a que los elementos están en paralelo obtenemos la siguiente ecuación:

$$\frac{d^2\nu}{dt^2} + \frac{1}{C}(3\gamma\nu^2 - \alpha)\frac{d\nu}{dt} + \frac{1}{LC}\nu = 0$$

Para un determinado juego de valores de parámetros, la ecuación diferencial no lineal homogénea de segundo orden anterior puede escribirse como:

$$\ddot{\nu} - \varepsilon(1 - \nu^2)\dot{\nu} + \nu = 0$$

Esta ecuación diferencial no lineal de segundo orden puede expresarse fácilmente como dos ecuaciones diferenciales de primer orden, definiendo las variables de estado $x_1 = \nu$ y $x_2 = \dot{\nu}$, como se expresa a continuación:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \varepsilon(1 - x_1^2)x_2 - x_1 \end{cases} \tag{1}$$

La dinámica no lineal de este sistema está caracterizada por trayectorias periódicas cuyas características dependen del parámetro ε.

1.1. Aproximación de Euler.

Dado el sistemas de ecuaciones en tiempo continuo expresado anteriormente (1), podemos decir que el mismo es de la forma:

$$\begin{cases} \dot{x_1} = f(x_2(t)) \\ \dot{x_2} = g(x_1(t), x_2(t)) \end{cases}$$

Una forma de discretizar una ecuación diferencial de primer orden de manera de que puedan ser implementada en un algoritmo es discretizar el tiempo continuo de la forma $t = nT$ (donde n es entero y T representa el periodo de muestreo) y aproximar la derivada respecto al tiempo por el cociente incremental (2).

$$\frac{dx}{dt} \approx \frac{x(n+1) - x(n)}{T} \tag{2}$$

Esta aproximación es conocida como la aproximación de Euler de primer orden. Dicha aproximación mejora a medida que T se hace más pequeño.

Implementando esta técnica en nuestro sistema de ecuaciones obtenemos:

$$\begin{cases} \dfrac{x_1(n+1) - x_1(n)}{T} = f(x_2(n)) \\ \dfrac{x_2(n+1) - x_2(n)}{T} = g(x_1(n), x_2(n)) \end{cases}$$

Ahora despejando obtenemos que:

$$\begin{cases} x_1(n+1) = T \cdot f(x_2(n)) + x_1(n) \\ x_2(n+1) = T \cdot g(x_1(n), x_2(n)) + x_2(n) \end{cases}$$

Remplazando el valor de cada una de las funciones, evaluada ahora en términos de n:

$$\begin{cases} x_1(n+1) = x_1(n) + T \cdot x_2(n) \\ x_2(n+1) = x_2(n) + T \cdot \big(\varepsilon(1 - x_1(n)^2)x_2(n) - x_1(n)\big) \end{cases} \tag{3}$$

Las ecuaciones mostradas en la ecuación (3) representan las ecuaciones en diferencias asociadas al sistema de ecuaciones en tiempo continuo descripto mediante (1).

1.2. Puntos de equilibrio del sistema.

Los puntos de equilibro del sistema son valores de x_1 y x_2 que verifican simultáneamente que:

$$\begin{cases} \dot{x_1} = 0 \\ \dot{x_2} = 0 \end{cases}$$

Remplazando dichos valores en la ecuación (1) obtenemos que:

$$\begin{cases} x_2 = 0 \\ \varepsilon(1 - x_1^2)x_2 - x_1 = 0 \end{cases} \Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = 0 \end{cases}$$

Por lo tanto estos representan un punto de equilibrio del sistema.

1.3. Implementación en Matlab en forma de *script-file* y simulaciones.

Procedemos a escribir un *script-file* usando Matlab que implemente el sistema en tiempo discreto que determinamos anteriormente, durante un tiempo de $100s$, a partir de condiciones iniciales $(0, 1)$ para diferentes valores de ε. A continuación se detalla el código de dicho *script-file*.

Primero comenzamos escribiendo el encabezado del *script*, junto a los comandos *clear*, *clc*, *close all*.

```
% Script: script_P1_c.m
% Author: Federico, Scheytt - Joaquin, Gonzalez Targon
% Date: Mayo 2023

clear, clc, close all
```

Figura 2: Encabezado

Luego definimos las constantes, en particular decidimos crear un vector para los diferentes tiempos de muestreo T, como así también, un vector para los valores de ε y finalmente la constante t_f que representa el tiempo final en el cual queremos hacer la simulación del sistema.

```
%% Constantes
T = [0.001 0.1];
e = [0 1 10 50];
tf = 100;
```

Figura 3: Declaración de constantes

A continuación, declaramos las condiciones iniciales como se detalla en el enunciado.

```
%% Condiciones iniciales
x1i = 0;
x2i = 1;
```

Figura 4: Condiciones iniciales

Luego implementamos el método de Euler para la aproximación de la primera derivada, donde el algoritmo puede pensarse de la siguiente manera:

- **1er bucle for: (índice i)** En este primer lazo iteramos según la cantidad de elementos del vector T lo cual nos permite implementar el método para cada uno de los diferentes casos, como así sus correspondientes gráficas.
- **Discretización:** Ahora para el valor de T actual generamos una discretización del vector de tiempos t, el cual depende del paso adoptado anteriormente. Luego realizamos la discretización e inicialización de los vectores x_1 y x_2 en función de la dimensión de t, junto con las condiciones iniciales.
- **2do bucle for: (índice j)** En este segundo lazo iteramos según la cantidad de elementos del vector e los cual nos permite nos permite implementar el método para cada uno de los diferentes valor de ε, como así sus correspondientes gráficas. Luego llamamos a la función *f_plot* para realizar los gráficos correspondientes.
- **3er bucle for: (índice n)** En este tercer lazo iteramos según la cantidad de elementos del vector t los cual nos permite nos permite implementar el método de Euler y calcular cada uno de los valores de x_1 y x_2.

```
%% Implementacion del metodo
for i = 1:length(T)
    % Discretizacion
    t = 0:T(i):tf-T(i);
    x1 = [x1i, zeros(1,length(t)-2)];
    x2 = [x2i, zeros(1,length(t)-2)];
    for j = 1:length(e)
        for n = 1:length(t)-1
            % Aproximacion de Euler
            x1(n+1) = x1(n) + T(i).*x2(n);
            x2(n+1) = x2(n) + T(i).*(e(j).*((1-(x1(n)).^2).*x2(n)) - x1(n));
        end
        % Graficos
        f_plot(i,j,x1,x2,e,t);
    end
end
```

Figura 5: Implementación del método

Para una mejor visualización del código decidimos generar una función local que realice los gráficos correspondientes a cada uno de los casos, la misma toma como argumentos los índices de los bucles, como así también el valor de la discretización y los vectores calculados.

```
%% Funcion f_plot
function f_plot(i,j,x1,x2,e,t)
    figure(i)
        subplot(2,4,j), plot(t,x1,'r')
            title(['$\varepsilon$ = ', num2str(e(j))],'Interpreter','latex')
            legend('x_{1}(t)', 'Location', 'northwest')
            xlabel('Tiempo [seg]')
            ylabel('x_{1}(t)')
            hold on, grid on
        subplot(2,4,j+4), plot(t,x2,'b')
            title(['$\varepsilon$ = ', num2str(e(j))],'Interpreter','latex')
            legend('x_{2}(t)', 'Location', 'northwest')
            xlabel('Tiempo [seg]')
            ylabel('x_{2}(t)')
            hold on, grid on
end
```

Figura 6: Función plot

1.3.1. T = 0,001 s

Para el caso en el que el periodo de muestreo es igual a $T = 0,001s$ obtuvimos los siguientes resultados.

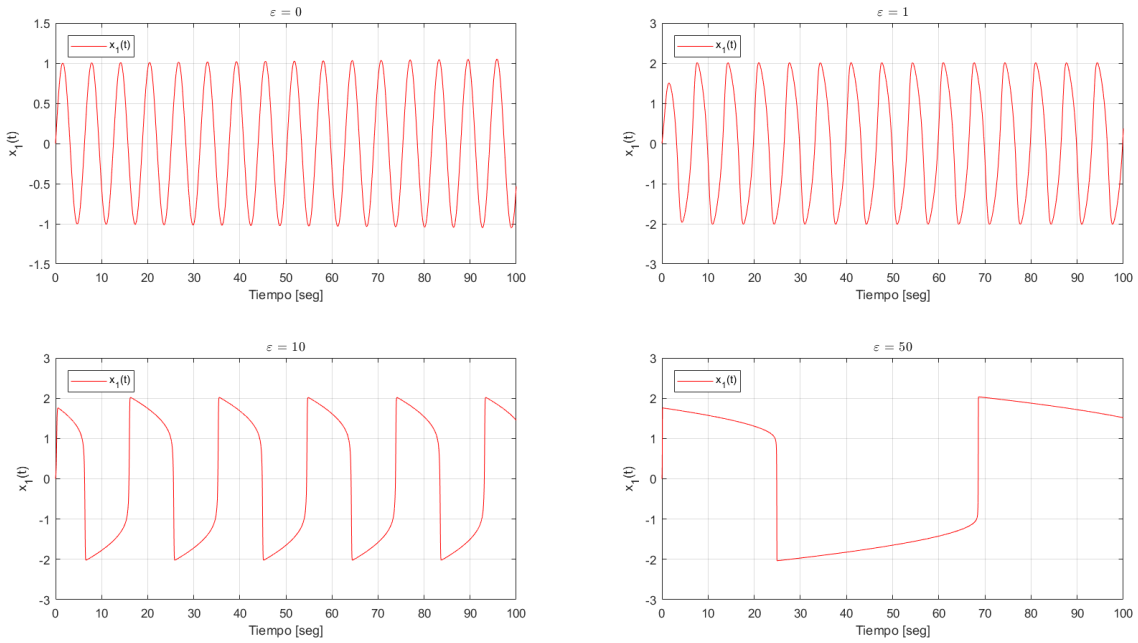

Figura 7: Diferentes respuestas en función de ε

Podemos ver entonces que la señal de salida presenta un comportamiento periódico, donde a medida que ε aumenta el periodo de la señal de salida aumenta. Sabemos que cuando $\varepsilon = 0$ el sistema es denominado como Oscilador Armónico, para el cual las trayectorias periódicas son sinusoidales, lo cual es apreciable en la gráfica obtenida.

1.3.2. T = 0,1 s

Para el caso en el que el periodo de muestreo es igual a $T = 0,1s$ obtuvimos los siguientes resultados.

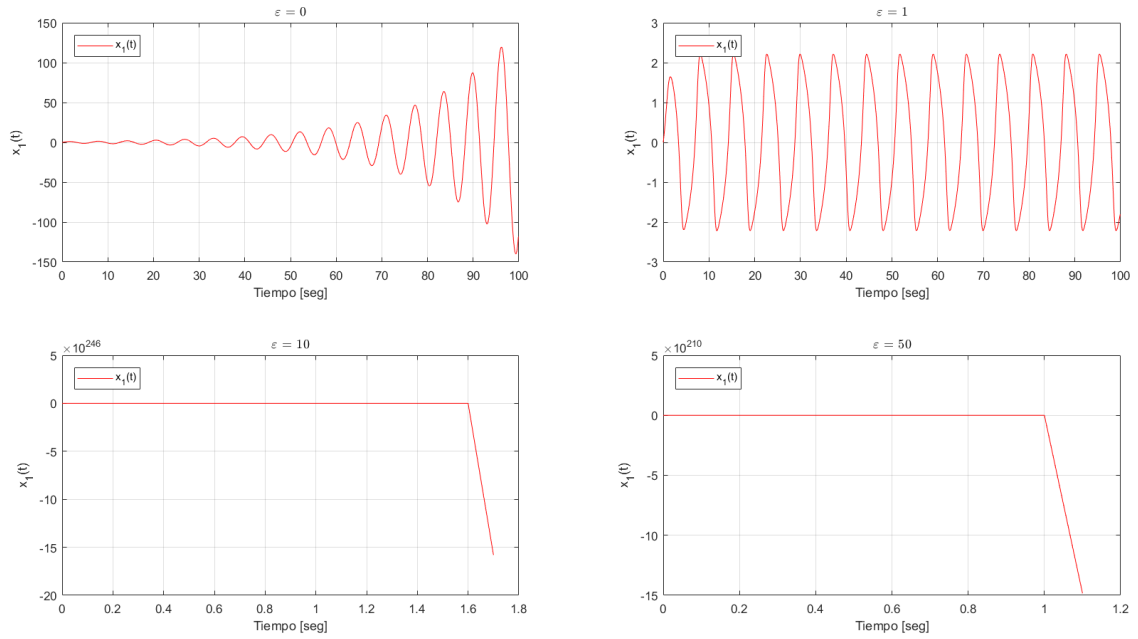

Figura 8: Diferentes respuestas en función de ε

Podemos observar que la aproximación de la derivada por el cociente incremental es buena si el paso de integración numérica T es pequeño. En el caso de que el paso de integración sea grande puede que haya una pérdida de información de la respuesta al sistema. Notamos además que el método tiende a diverger para valores de ε grandes, en cada iteración se comete un error de aproximación de la derivada, que se va acumulando. Entonces si el tiempo total de integración numérica es grande, el error acumulado va a ser grande y los resultados serán poco confiables.

1.4. Implementación en Matlab en forma de *function-file*.

A la hora de implementar nuestro sistema en forma de $function - file$, lo hicimos de la siguiente manera. Primero comenzamos definiendo la función como se muestra a continuación.

```
function [x1, x2] = function_P1(x1i, x2i, T, e, tf)
```

Figura 9: Definición de la función

Luego incluimos un *help*, donde se indica cuales son cada uno de los argumentos de entrada como así también los argumentos de salida.

```
% function_P1(x1i, x2i, T, e, tf): Funcion que simula el comportamiento de un Oscilador de Van der
%   Pol, mediante la implementacion del metodo de Euler para la aproximacion de las derivadas de
%   primer orden correspondientes al sistema de ecuaciones de estado que representan la dinamica
%   fisica del sistema.
%
%   EDO no lineal que caracteriza la respuesta del sistema:
%       v''(t) - e(1-v(t)^2)v'(t) + v(t) = 0
%
%   Ecuaciones de estado:
%       x1(t) = v(t)
%       x2(t) = v'(t)
%
%   EDO en funcion de las variables de estado:
%       x1'(t) = X2
%       x2'(t) = e(1-x1(t)^2)x2'(t) + x1(t)
%
% INPUT:
%   x1i: Valor inicial de x1
%   x2i: Valor inicial de x2
%   T:   Periodo de muestreo en segundos
%   e:   Parametro asociado al amortiguamiento del sistema
%   tf:  Tiempo total de la simulacion en segundos
%
% OUTPUT:
%   x1: Vector de resultados x1
%   x2: Vector de resultados x2
%
% Author: Federcio Scheytt - Joaquin Gonzalez Targon
% Date: Mayo 2023
```

Figura 10: Help del la función

Ahora para el valor de T dado generamos una discretización del vector de tiempos t.

```
%% Discretizacion
t = 0:T:tf-T;
```

Figura 11: Discretización

A continuación, generamos un vector con las condiciones iniciales y ceros cuya dimensión sea igual a la cantidad de elementos que posee el vector de discretización t.

```
%% Condiciones iniciales
x1 = [x1i, zeros(1,length(t)-2)];
x2 = [x2i, zeros(1,length(t)-2)];
```

Figura 12: Condiciones iniciales

Luego procedemos a realizar la implementación del método de Euler para calcular las derivadas de primer orden necesarias para resolver nuestro sistema.

```
%% Aproximacion de Euler
for n = 1:length(t)-1
    x1(n+1) = x1(n) + T.*x2(n);
    x2(n+1) = x2(n) + T.*(e.*((1-(x1(n)).^2).*x2(n)) - x1(n));
end
```

Figura 13: Implementación del método de Euler

Finalmente retornamos los vectores calculados correspondientemente.

```
%% Return
return
```

Figura 14: Return de la función

1.5. Representación en plano de fase.

El plano de fase de un sistema es una representación geométrica en un plano de un par de variables para distintos instantes de tiempo, es decir, cada punto de un plano de fase representa el valor de ese par de variables en un instante de tiempo dado. Así, si representamos los valores de dicho par de variables para distintos instantes de tiempo en un intervalo, por ejemplo $t \in [0, t_f]$, obtendremos una trayectoria. Para el caso de sistemas no lineales, como el que se estudia en este problema dicha trayectoria podría ser una curva cerrada periódica o tender luego de un tiempo suficientemente largo a una trayectoria cerrada; Si esta trayectoria cerrada es aislada se denomina ciclo límite.

Ahora nos proponemos mediante la función descripta anteriormente obtener los valores y dibujar las trayectorias con colores diferentes en el plano de fase $x_2 = f(x_1)$, inicializando el sistema desde 4 diferentes condiciones iniciales y para los valores conocidos de T y ε.

Primero comenzamos escribiendo el encabezado del *script*, junto a los comandos *clear*, *clc*, *close all*.

```
% Script: script_P1_f.m
% Author: Federico, Scheytt - Joaquin, Gonzalez Targon
% Date: Mayo 2023

clear, clc, close all
```

Figura 15: Encabezado

Luego definimos las constantes según el enunciado del problema.

```
%% Constantes
T = 0.001;
e = 10;
tf = 50;
```

Figura 16: Declaración de constantes

Ahora generamos un vector con las condiciones iniciales dadas que iremos recorriendo a medida que calculemos el plano de fase para cada par de las mismas.

```
%% Condiciones iniciales
x1i = [1 -0.5 -1 1.5];
x2i = [1 -5 10 -4];
```

Figura 17: Vectores de condiciones iniciales

Ahora iteramos para cada una de las condiciones iniciales, donde para cada una de ellas realizamos el plot correspondiente, llamando tanto a la función $function\_P1()$, como así también a la función local $f\_plot()$.

```
%% Implementacion de la solucion
for i=1:length(x1i)
    % Calculo de los vectores x1 y x2
    [x1, x2] = function_P1(x1i(i), x2i(i), T, e, tf);
    % Graficos
    f_plot(i,x1i,x2i,x1,x2,c)
end
```

Figura 18: Implementación

Mediante la función *f_plot*() realizamos los gráficos necesarios, un subplot para los cuatro casos iniciales y además un plot donde se encuentran todas las gráficas juntas.

```
%% Funcion f_plot
function f_plot(i,x1i,x2i,x1,x2,c)
    figure(1)
        subplot(2,2,i), plot(x1,x2,'r',x1i(i),x2i(i),'*k')
            title(['$(x_{1i}, x_{2i}) = ($', num2str(x1i(i)), ',', num2str(x2i(i)), '$)$'], 'Interpreter', 'latex')
            xlabel('x_{1}(t)')
            ylabel('x_{2}(t)')
            legend('Trayectoria','Condicion inicial')
            hold on, grid on
    figure (2)
        plot(x1i(i),x2i(i),'*k', x1,x2,c(i))
            title('Plano de fase')
            xlabel('x_{1}(t)')
            ylabel('x_{2}(t)')
            hold on, grid on
end
```

Figura 19: Función plot

Finalmente obtenemos las siguiente figuras.

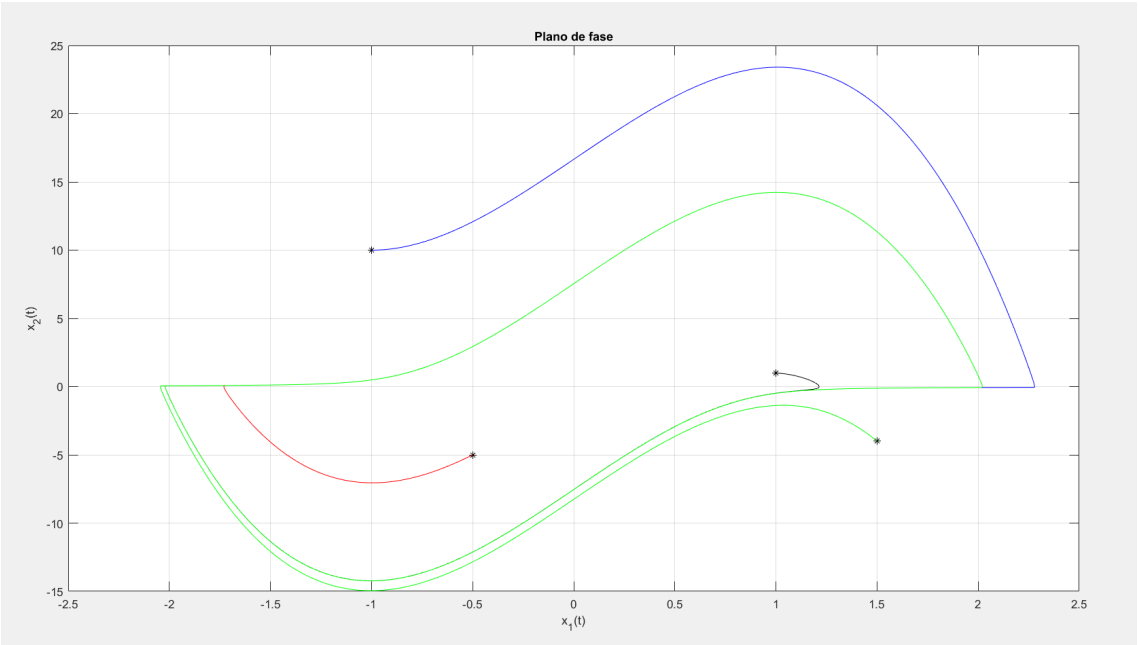

Figura 20: Representación en plano de fase

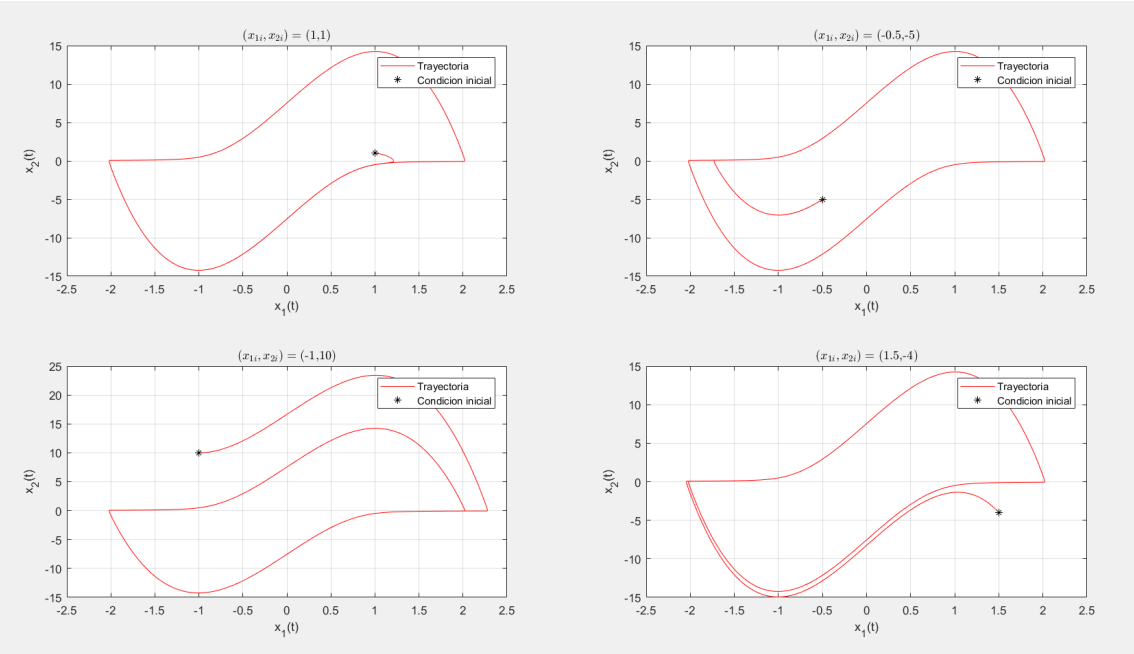

Figura 21: Representación en plano de fase

1.6. Análisis del comportamiento de las trayectorias.

Como podemos ver en la gráficas anteriores para dichos valores iniciales, las trayectorias que describen los puntos obtenidos por la función a partir de la iteración de Euler, tienden a estabilizarse en lo que denominamos ciclo límite, que es una trayectoria cerrada y aislada. Por lo tanto podemos decir que este ciclo límite es estable y funciona como atractor de las trayectorias.

En el caso de tomar como condiciones iniciales las coordenadas $(0, 0)$, es decir el punto de equilibrio encontrado en el ítem b), obtenemos que el sistema ya está estable y no va a tener una trayectoria que se estabilice en el ciclo límite. En el caso de tomar un punto cercano al equilibrio, la trayectoria no se va a estabilizar en el $(0, 0)$, sino que también va a tender a estabilizarse en el ciclo límite.

2. Radar para medición de velocidad de vehículos.

En los dispositivos usados para sensar el exceso de velocidad en vehículos se utiliza el principio del radar para medir la distancia al vehículo en dos instantes de tiempo T_1 y T_1 separados un intervalo conocido $\Delta T = T_2 - T_1$, como se representa en la figura. Esto permite computar la velocidad del vehículo, ya que se conoce la distancia recorrida y el tiempo empleado en recorrerla.

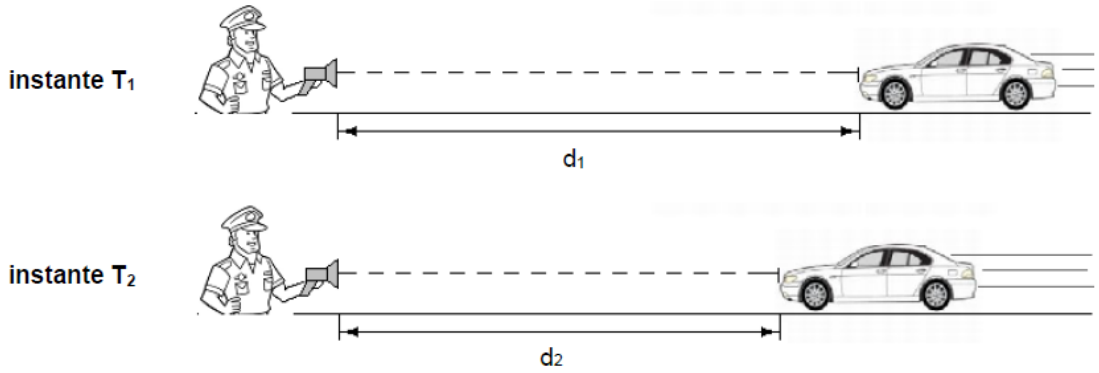

Figura 22: Sensado de velocidad de vehículos

En el radar, la señal recibida y es una versión retrasada y atenuada de la señal transmitida x, corrupta por ruido, es decir:

$$y(n) = \alpha\ x(n - D) + w(n)$$

donde $w(n)$ es una componente de ruido aditivo no correlacionado con la señal transmitida, D es el tiempo (expresado en muestras) que tarda la señal en alcanzar el vehículo y volver al radar, y α es un coeficiente de atenuación ($0 < \alpha < 1$).

2.1. Secuencias de correlación cruzada.

Considerando que la frecuencia de muestreo del radar es $Fs = 100\ MHz$, que la velocidad de propagación de la onda electromagnética es $c = 3 \cdot 10^8\ m/s$, y que $\Delta T = 16s$, nos proponemos graficar en una misma ventana de figura, con dos colores diferentes, las secuencias de correlación cruzada $r_{x_1y_1}(l)$ y $r_{x_2y_2}(l)$.

Primero comenzamos escribiendo el encabezado del *script*, junto a los comandos *clear*, *clc*, *close all*.

```
% Script: script_P2.m
% Author: Federico, Scheytt - Joaquin, Gonzalez Targon
% Date: Mayo 2023

clear, clc, close all
```

Figura 23: Encabezado

Luego continuamos cargando los datos correspondientes del archivo *datosProb*2.*mat*, archivo en el cual se encuentran contenidas las señales para cada uno de los tiempos correspondientes.

```
%% Carga del archivo de datos
load('datosProb2.mat')
```

Figura 24: Carga de datos

Luego definimos las constantes según el enunciado del problema.

```
%% Constantes
c = 3e+8;
Ts = 1/Fs;
dT = 16;
```

Figura 25: Declaración de constantes

Procedemos luego a realizar las discretizaciones correspondientes a cada uno de los parámetros de interés de nuestro problema.

```
%% Discretizaciones
n = t.*Fs;
l = -(length(x1)-1):length(x1)-1;
```

Figura 26: Discretización

Procedemos luego a calcular las correlaciones correspondientes, tanto las cruzadas como así también las auto correlaciones que nos serán útiles mas adelante en la resolución del ejercicio.

```
%% Calculo de correlaciones
rxx1 = xcorr(x1);
rxx2 = xcorr(x2);
ry1x1 = xcorr(y1,x1);
ry2x2 = xcorr(y2,x2);
```

Figura 27: Correlaciones

Ahora procedemos a graficar las correlaciones cruzadas $r_{x_1y_1}(l)$ y $r_{x_2y_2}(l)$.

```
%% Grafico de correlacion cruzada
figure(1)
    plot(l,ry1x1,'r', l, ry2x2, 'b')
        title('Correlacion cruzada')
        legend('r_{y1x1}(l)', 'Location', 'northeast', 'r_{y1x1}(l)', 'Location', 'northeast')
        xlabel('Indice l')
        ylabel('Correlacion')
        hold on, grid on
```

Figura 28: Gráfico de correlación

Podemos ver entonces que la gráfica presenta dos picos, que no se encuentran en cero y cuya posición respecto a l mas adelante en la resolución del ejercicio nos permitirá determinar el tiempo de retraso D entre la emisión y la recepción de la señal.

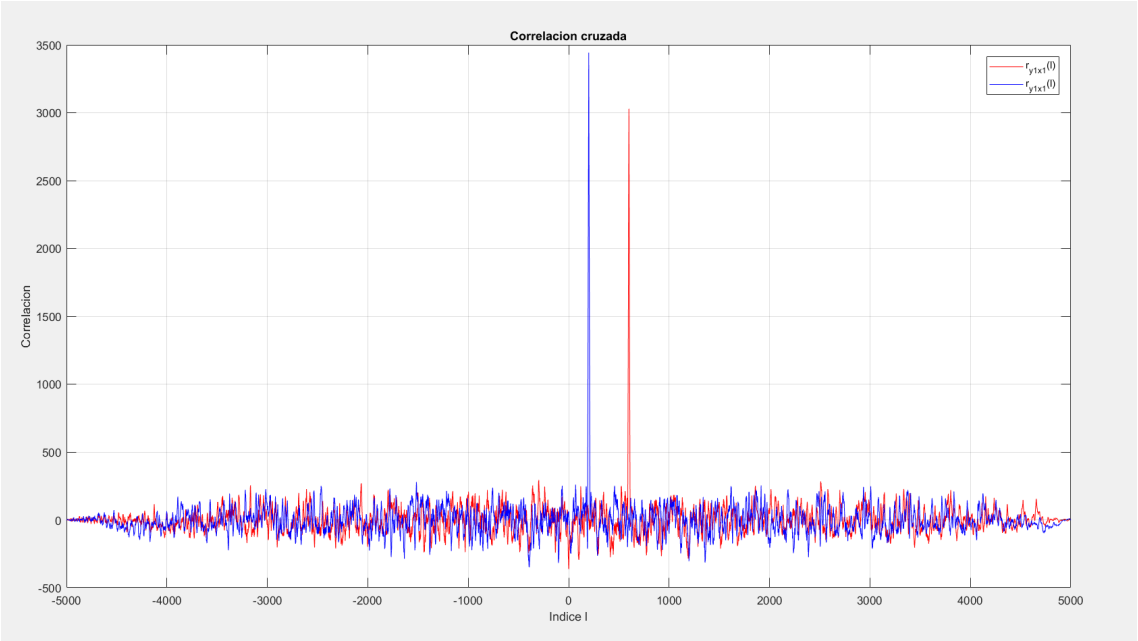

Figura 29: Gráfico de correlación

2.2. Calculo de distancias d1 y d2.

Ahora nos proponemos calcular cuales son las distancias d_1 y d_2 mostradas en la figura (22), para ello si planteamos la siguiente correlación:

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n+l)x(n) = \sum_{n=-\infty}^{\infty} [\alpha x(n+l-D) + w(n+l)]x(n)$$

$$r_{yx}(l) = \alpha \sum_{n=-\infty}^{\infty} x(n+l-D)x(n) + \sum_{n=-\infty}^{\infty} w(n+l)x(n)$$

$$r_{yx}(l) = \alpha r_{xx}(l-D) + r_{wx}(l)$$

Como la señal $x(n)$ no esta correlacionada con el ruido $w(n)$ entonces:

$$r_{wx}(l) \approx 0$$

Por lo tanto resulta que:

$$r_{yx}(l) \approx \alpha r_{xx}(l-D)$$

Es decir que la correlación cruzada entre la señal reflejada y la transmitida ($r_{yx}(l)$) es una versión atenuada α y desplazada el retardo de transmisión D, de la autocorrelación de la señal transmitida $r_{xx}(l)$. Como $r_{xx}(l)$ tiene una máximo en cero, la correlación cruzada tendrá un máximo en D, por lo que este retardo en muestras puede medirse en la gráfica de $r_{yx}(l)$. Luego la distancia al vehículo puede calcularse como:

$$d = \frac{cDT_s}{2} \tag{4}$$

Esta metodología es implementable para ambas señales x_1 y x_2 por lo que a continuación se muestra como realizamos dicho calculo.

Pudimos ver que en la figura (29) que para cada una de las señales se presentaba un pico, por lo que mediante la función $max()$ calculamos en que posición del vector correspondiente al calculo de la correlación se encontraba dicho pico.

```
%% Calculo de picos de la correlacion
[p1, D1pos] = max(ry1x1);
[p2, D2pos] = max(ry2x2);
```

Figura 30: Picos

Una vez encontrada la posición, calculamos el tiempo de retardo D correspondiente para cada caso, este calculo se debe a que dicho valor D se encuentra como valor de la variable l.

```
%% Calculo del tiempo de retardo en muentras D
D1 = D1pos - length(x1);
D2 = D2pos - length(x1);
```

Figura 31: Tiempo de retardo en muestras

Una vez calculado el retardo, mediante la ecuación (4) calculamos cuanto vale cada una de la distancias correspondientes.

```
%% Calculo de las distancias respectivas
d1 = (c*D1*Ts)/2;
d2 = (c*D2*Ts)/2;
```

Figura 32: Distancias

Finalmente a través del calculo en Matlab obtenemos que:

$$\begin{cases} d_1 = 900\ m \\ d_2 = 600\ m \end{cases}$$

2.3. Determinación de la velocidad de circulación del vehículo.

Ahora gracias a los resultados obtenidos en los items anteriores podemos determinar cual es la velocidad del vehículo de la siguiente manera:

$$v = \frac{\Delta d}{\Delta t} \tag{5}$$

Por lo tanto a continuación obtenemos el valor de la variación de distancia entre los dos instantes.

```
%% Distancia del vehiculo entre los dos intervalos de tiempo
d = d1-d2;
```

Figura 33: Variación de la distancia

Ahora usando (5), y conocido el valor de la variación temporal entre ambas señales.

```
%% Velocidad del vehiculo
v = d/dT;
```

Figura 34: Velocidad del vehículo

Finalmente a través del calculo en Matlab obtenemos que:

$$v = 37,5\ m/s = 135\ Km/h$$

Conclusión

A modo de conclusión podemos destacar que ambos problemas requirieron tanto un desarrollo practico como así el análisis de sus fundamentos teóricos a la hora de encontrar una solución a los mismos, como así también una correcta implementación numérica a la hora de trabajar con el software Matlab. Podemos ver en el primer caso la importancia que presenta el periodo de muestreo que se toma y como este repercute significativamente en la solución del mismo, a la hora de discretizar el tiempo continuo. En el segundo caso vimos la gran utilidad que presenta la correlación entre señales y como mediante el uso correcto de dicha herramienta podemos determinar características del sistema en presencia de interferencias externas como son el ruido ambiente, abstraernos de esta señal no deseada y solamente quedarnos con las de interés. Además nos permitió como grupo la posibilidad de familiarizarnos mas con el software Matlab y el uso de todas sus funcionalidades.