LICENCIATURA EN FÍSICA

DEPARTAMENTO DE FÍSICA

FÍSICA COMPUTACIONAL 1

Reporte 8

Alumna:

Brambilla Zamorano Fátima Fernanda

Fecha:
16/04/18

1 Introducción y Antecedentes

Para la octava sesión de Física Computacional 1, trabajamos con el modelo de Van de Pol, el cual es un modelo para un sistema oscilador con amortiguamiento no lineal, el cual explicaremos más adelante, después de mencionar algunos datos sobre los antecedentes del mismo.
Balthasar Van der Pol fue un físico neerlandés, cuyos campos de investigación fueron la propagación de ondas, teoría de circuitos y física matemática, y su descubrimiento del oscilador que lleva su nombre le valió para recibir la medalla de honor del *Institute of Radio Engineers*. La ecuación de Van der Pol es uno de los primeros descubrimientos de la *Teoría del caos*, y esta a su vez tiene una larga historia en la física y la biología, sin embargo, no haremos mención de esta en este documento.

2 Modelo de Van de Pol

El oscilador de Van der Pol, es uno con amortiguamiento no lineal, y obedece a la siguiente ecuación de segundo orden:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0 \tag{1}$$

De donde, x es la posición en el tiempo, t es la función del tiempo, y μ es el amortiguamiento

3 Exploración de las soluciones del modelo en el Espacio Fase

En esta sección se presentará como se obtuvieron las gráficas correspondientes a la actividad, y estas mismas para su posterior discusión. Al igual que en las dos actividades anteriores, se definió un vector al principio del todo,con el cual estaríamos trabajando a lo largo de la actividad, dicho vector almacena el sistema de ecuaciones diferenciales. Se crearon varios tiempos dentro de un intervalo dado, también se definieron las condiciones iniciales y por último se resolvía el sistema de ecuaciones por medio de la función de *odeint*.

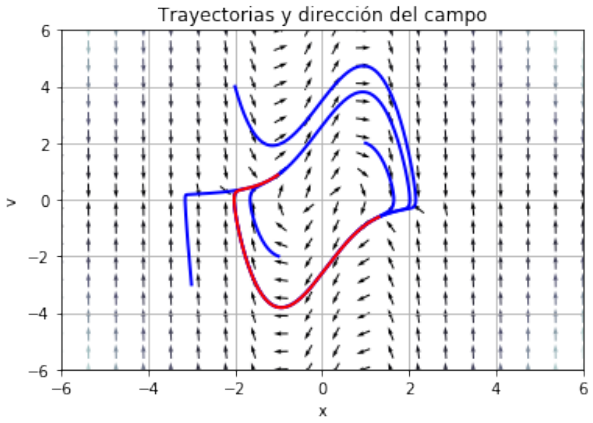
```
from numpy import *
import pylab as p
# Definition of parameters
mu= 2.0
def dX_dt(X, t=0):
    xp=X[1]
    yp=mu*(1-X[0]**2)*X[1]-X[0]
    return array([xp, yp ])
```

```
from scipy.integrate import odeint
from scipy import array
import matplotlib.pyplot as plt

stoptime = 6.0
numpoints = 5000

#Tiempo
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

#Condiciones iniciales
x0 = -3.0
v0 = -3.0

#Resolver ecuaciones
x, y = odeint(dX_dt,(x0,v0),t).T

with open('Fase1.dat', 'w') as archivo:
    for t1, x1,y1 in zip(t, x, y):
        print (t1, x1,y1 ,file=archivo)
```

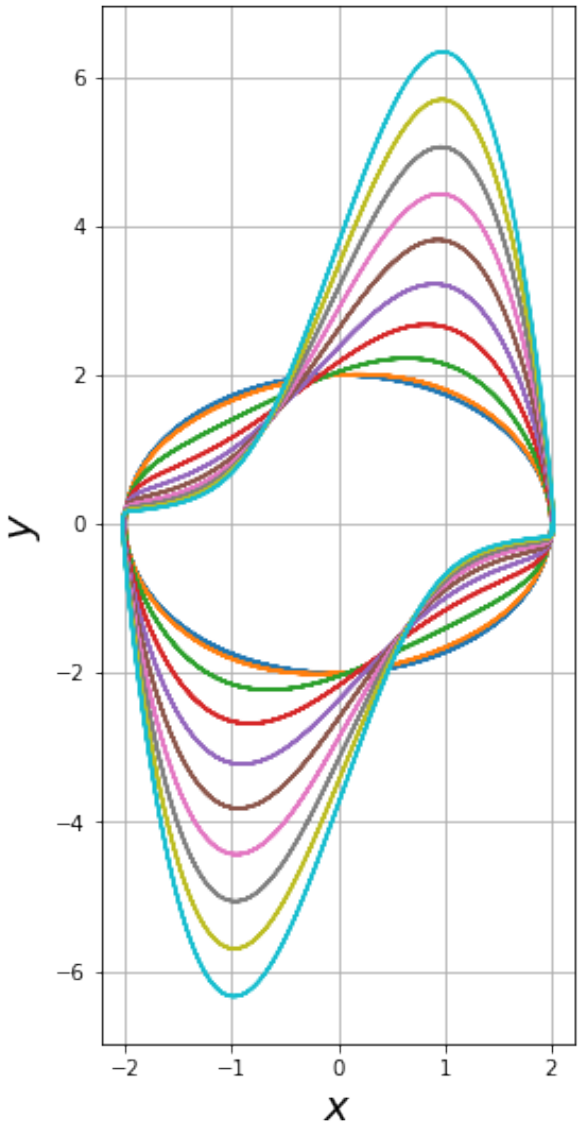
En la imagen de arriba, se pueden ver dos celdas del código, en la primera se muestra la manera en que se definió el vector que se usaría a lo largo de la actividad, mientras que en la segunda se puede ver como empieza a definirse un "campo vectorial" donde se definirá la primera representación gráfica que estamos buscando. Cabe mencionar, que para esta primera gráfica son necesarios cuatro archivos de datos, con los cuales se representaran cuatro fases con diferentes condiciones iniciales. En la segunda celda de la imagen previamente mostrada se pueden ver los datos para el primer archivo de datos. Como en actividades anteriores, lo que hicimos fue hacer un gráfico de posición contra velocidad, con la diferencia de que en este caso se puede ver el campo vectorial, y la dirección de cada punto, la cual esta dada por la derivada del mismo. Mediante la siguiente sección de código, se hizo una cuadricula que guardaría los valores de la posición y el tiempo, con sus derivadas respectivas.

```
#-------------------------------------------------------
# define a grid and compute direction at each point
nb_points   = 20

x = linspace(-6, 6, nb_points)
y = linspace(-6, 6, nb_points)

X1 , Y1  = meshgrid(x, y)                    # create a grid
DX1, DY1 = dX_dt([X1, Y1])                   # compute growth rate on the gridt
M = (hypot(DX1, DY1))                        # Norm of the growth rate
M[ M == 0] = 1.                              # Avoid zero division errors
DX1 /= M                                     # Normalize each arrows
DY1 /= M
```

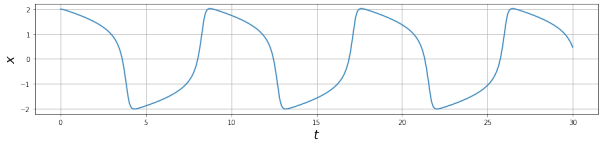
Y la gráfica que se obtuvo es la siguiente:



Para la siguiente gráfica se crearon diferentes archivos de datos con coeficientes de amortiguamiento distintos, y el siguiente gráfico fue la imagen obtenida:



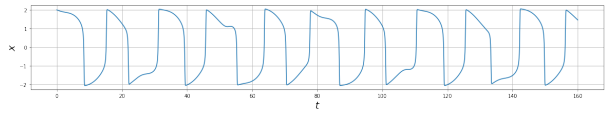
La siguiente imagen es la representación gráfica del movimiento de un oscilador con una constante de amortiguamiento aproximada a cinco.



Por último, se hizo una gráfica con ciertos valores para la amplitud, la constante de amortiguamiento, y la frecuencia. Al vector previamente definido se le agregaron estos tres datos, y se volvió a gráficar la posición contra el tiempo.

```
import numpy as np

def dX_dt(X,t):
    x = X[0]
    y = X[1]
    dx = y
    dy = u*(1 - x*x)*dx - x + M*np.sin(w*t)
    return array([dx, dy])

t0 = 0
tmax = 160
pastemps = 0.01
t = arange(t0, tmax, pastemps)

#Condiciones iniciales
x0 = 2.0
y0 = 0.0

#Parametros
u = 8.53
M = 1.2
w = np.pi/5.0

#Solución a la ecuación diferencial
x, y = odeint(dX_dt,(x0,y0),t).T

#Trayectoria del ocilador
p1 = plt.figure(figsize=(20,3.0))
plt.grid()
plt.xlabel('$t$', fontsize = 20)
plt.ylabel('$x$', fontsize = 20)
plt.plot(t, x)
plt.show()

p1.savefig('CuartaFig.png')
```

Y se obtuvo la siguiente gráfica:



4 Resultados y Discusión

Después de realizar y analizar las gráficas de la actividad, se pudo ver que hay ciertos detalles en cada una de ellas. En primer lugar, la gráfica en la que se hizo uso de un campo vectorial, se puede apreciar que, si bien cada archivo que se uso tenía datos de valor inicial distintos, si el oscilador tiene una sola constante de amortiguamiento, todos llevaran a un mismo ciclo limite de manera eventual, el momento en que lo hagan puede variar, pero terminaran llegando a ese limite. Otra cosa que se puede observar, es que los puntos alejados al ciclo solo se mueven en las direcciones positivas y negativas de y, es decir, no se mueven en el eje x, y entre más se acercan los puntos al limite, empiezan a seguir una curva. De la segunda gráfica podemos observar la evolución del ciclo limite para valores distintos de la constante de amortiguamiento, siendo visible como el limite se estira cada vez que el valor del amortiguamiento aumenta. De las gráficas 3 y 4, que nos muestran la posición contra el tiempo de un oscilador forzado, y otro que no, respectivamente, si se comparan, se puede ver que el oscilador forzado tiene cambios bastantes bruscos en su comportamiento, mientras que el oscilador que no lo esta, tiene un comportamiento más periódico.

5 Conclusiones del Estudio

El estudiar este tipo de problema nos permite ver como detalles tan pequeños como lo puede ser una fuerza de dispersión, permite encontrar grandes cambios en un sistema, lo que leva a descubrir y analizar nueva información, como lo fue en el caso de Van der Pol. El estudiar este tipo de problemas con herramientas como Python y Jupyter, hace que examinar la información sea más sencillo, y no solo eso, sino que hace posible encontrar nueva información a partir de distintos tipos de representaciones gráficas, y soluciones númericas a las que antes o no se tenía acceso, o tomaban demasiado tiempo para resolver.

6 Bibliografía

- Kanamaru, T. (16 de Enero de 2014). Scholarpedia.
  Obtenido de $http://www.scholarpedia.org/article/Van_der_Pol_oscillator
- Wikipedia. (05 de Marzo de 2018). Obtenido de
  $https://en.wikipedia.org/wiki/Van_der_Pol_oscillator

7 Ápendice

- Este ejercicio pareciera simular al desarrollado en las actividades 6 y 7, ¿Qué aprendiste nuevo?
  En esta actividad se utilizarón gráficos distintos a los que habíamos estado haciendo con anterioridad, de modo que puedo decir que he aprendido a hacer otros tipos de gráficos durante esta actividad, aunque me costo bastante entenderla.
- ¿Qué fue lo que más te llamo la atención del oscilador de Van de Pol?
  No hay algo que haya capturado más mi atención que otra cosa, pienso que todo el tema es bastante interesante.
- Has escuchado ya hablar del caos. ¿Por qué sería importante estudiar este oscilador?
  Creo que es importante estudiar este oscilador, y otros tipos de problemas de la misma naturaleza, ya que son los que más se aproximan a la realidad, puesto que en su mayoría durante los cursos que hemos tomado hasta el momento, se han trabajado problemas en situaciones ideales, es decir, sin tomar en cuenta cosas como la fricción en distintos medios, el movimiento de la tierra en caso de situaciones de movimiento de proyectil, entre otras.
- ¿Qué mejorarías en esta actividad?
  Quizás le haría una introducción a la actividad, explicando como trabajar con ella o con las gráficas que son necesarias generar, ya que fue esto lo que más problema me causo durante la actividad.
- Algún comentario adicional antes de dejar de trabajar en Jupyter con Python?
  Quisiera decir que trabajar en Python por medio de Jupyter ha sido bastante cómodo, aún si en realidad no he entendido del todo todo lo que hemos hecho, al final siento que he aprendido parte de lo necesario para poder manejarlo con más naturalidad de aquí en adelante.
- Cerraremos la parte de tu trabajo con Python, ¿Qué te ha parecido?
  Ha sido una grata experiencia, excepto en la evaluación pasada, y posbilemente en la siguiente.