Universidad Nacional de Ingeniería

Escuela Profesional de Ciencia de la Computación
Física Computacional

Practica 1

Gerson Garrido Mansilla
Código : 20130424J

April 8, 2017

Ejercicio 1: Volterra-Lotka

Sistema Volterra-Lotka

Importamos las librerías a usar:

```
>>> import matplotlib.pyplot as plt # Libreria para graficos
>>> import seaborn as sns # Graficos
>>> import numpy as np # Algebra Lineal

>>> %matplotlib inline
>>> plt.rcParams['figure.figsize'] = (18, 6)
>>> sns.set_style("darkgrid")
```

Luego inicializamos los parámetros:

```
>>> x = [10.0] # Poblacion de predadores
>>> y = [10.0] # Poblacion de presas

>>> a = 0.05
>>> b = 0.002
>>> c = 0.06
>>> d = 0.004

>>> step = 0.1
>>> N = 500 # numero de muestras

>>> t = np.arange(0,N,step) # Serie de tiempo

# Para guardar en el archivo
>>> file = open("VolterraLotka.txt", "w")

>>> file.write("tiempo \t\t x \t\t y \n")

>>> for i in range(int(N/step)-1):
    x.append(x[i] + step*(x[i]*(a - b*y[i])))
    y.append(y[i] + step*(-y[i]*(c - d*x[i])))
    file.write("{0:.2f} \t \t {0:.4f} \t \t {0:.4f} \n".format(t[i],x[i],y[i]))

# Cerrando el archivo abierto
>>> file.close()
```

Ahora vamos a graficar:

```
>>> plt.title('Evolucion de la poblacion de predadores y presas')
>>> plt.plot(t, x,'g-',label='predador')
>>> plt.plot(t, y,'r-',label='presa')
>>> plt.grid()
>>> plt.xlabel('tiempo')
>>> plt.ylabel('poblacion')
>>> plt.grid()
>>> plt.legend(loc='best')
>>> plt.show()
```

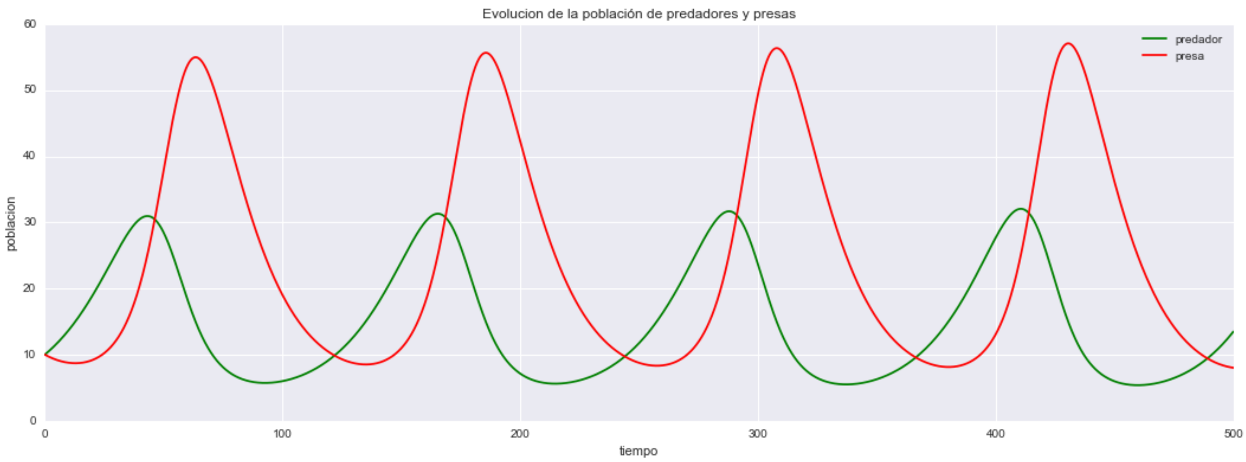

Figura. 1: Gráfica de la evolucion de poblaciones

Ahora vamos a plotear la gráfica de fases:

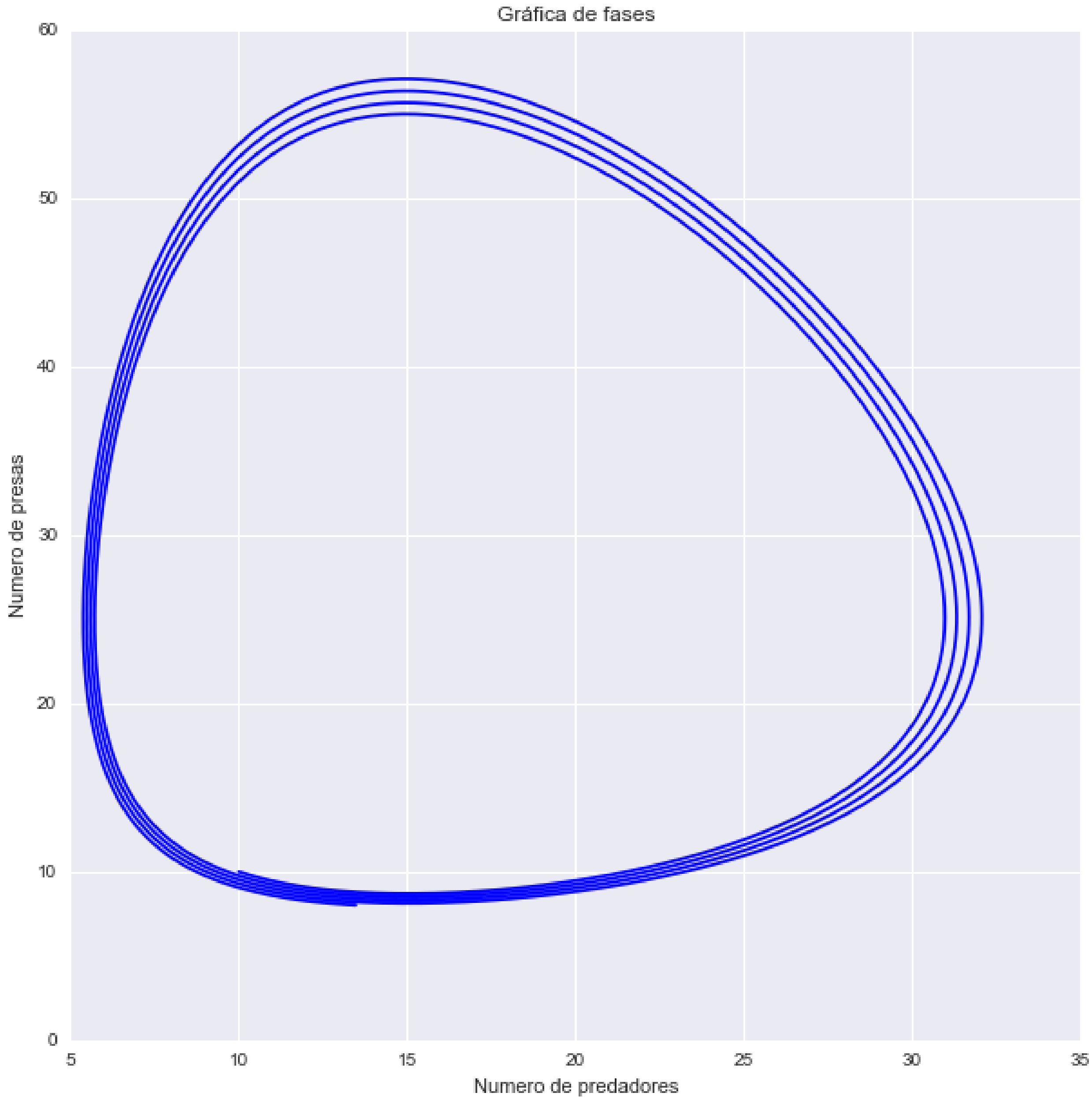

Figura. 2: Gráfica de fases

Analisis de estabilidad de las soluciones estacionarias

Cerca de estos dos puntos, el sistema puede ser linealizado:

$$\frac{dX}{dt} = J(X)$$

donde J es la matriz jacobiana evaluada en el punto correspondiente. Tenemos que definir la matriz Jacobiana:

```
>>> def Jacobiano(X,a,b,c,d):
        return np.array([[a -b*X[1], -b*X[0]],
                        [b*d*X[1]  , -c +b*d*X[0]]])
```

El equilibrio se produce cuando la tasa de crecimiento es igual a 0. Esto da dos puntos fijos:

```
# soluciones estacionarias
>>> X_0 = np.array([      0.  ,    0.])
>>> X_1 = np.array([ c/(d*b), a/b])
```

Cerca de **X_0**, que representa la extinción de ambas especies, tenemos:

```
# soluciones estacionarias
>>> J = Jacobiano(X_0,a,b,c,d)
>>> J

array([[ 0.05, -0.  ],
       [ 0.  , -0.06]])
```

Cerca de **X_0**, el número de conejos aumenta y la población de zorros disminuye. El origen es por lo tanto un punto silla.

```
>>> lambda1, lambda2 = np.linalg.eigvals(J)
>>> print("Los autovalores en el punto X_0 son:")
>>> print(lambda1,lambda2)

Los autovalores en el punto X_0 son:
0.05 -0.06
```

Cerca a **X_1**, tenemos:

```
>>> J2 = Jacobiano(X_1,a,b,c,d)
>>> J2

array([[  0.00000000e+00,  -1.50000000e+01],
       [  2.00000000e-04,   0.00000000e+00]])

# Cuyos eigenvalores son +/- sqrt(c*a).j:
>>> lambda1, lambda2 = np.linalg.eigvals(J2)
>>> print("Los autovalores en el punto X_1 son:")
>>> print(lambda1,lambda2)

Los autovalores en el punto X_1 son:
0.0547722557505j -0.0547722557505j
```