



MODELOS DE SISTEMAS BIOLÓGICOS

Daniela Pamelin Alvarez Guarneros A01026164

LOTKA VOLTERRA

Sistema de dos ecuaciones diferenciales de primer orden, acopladas, autónomas y no lineales, que se usan para describir dinámicas de sistemas biológicos en el que dos especies interactúan, una como presa y otra como depredador.



→ 13

→ 13 A

→ 14

→ 14 A



$$\begin{cases} \dot{x} = ax - bxy \\ \dot{y} = -cy + dxy \end{cases}$$

Ecuación descriptiva para las presas

Ecuación descriptiva para los depredadores

MÉTODO DE EULER

herramienta numérica para
aproximar los valores para las
soluciones de ecuaciones
diferenciales.

Método explícito $O(h^2)$ la
condición inicial se usa para
calcular la pendiente de la
función en ese mismo punto.

$$x(t + \Delta t) = x(t) + \frac{dx}{dt} \Delta t$$

$$y(t + \Delta t) = y(t) + \frac{dy}{dt} \Delta t$$

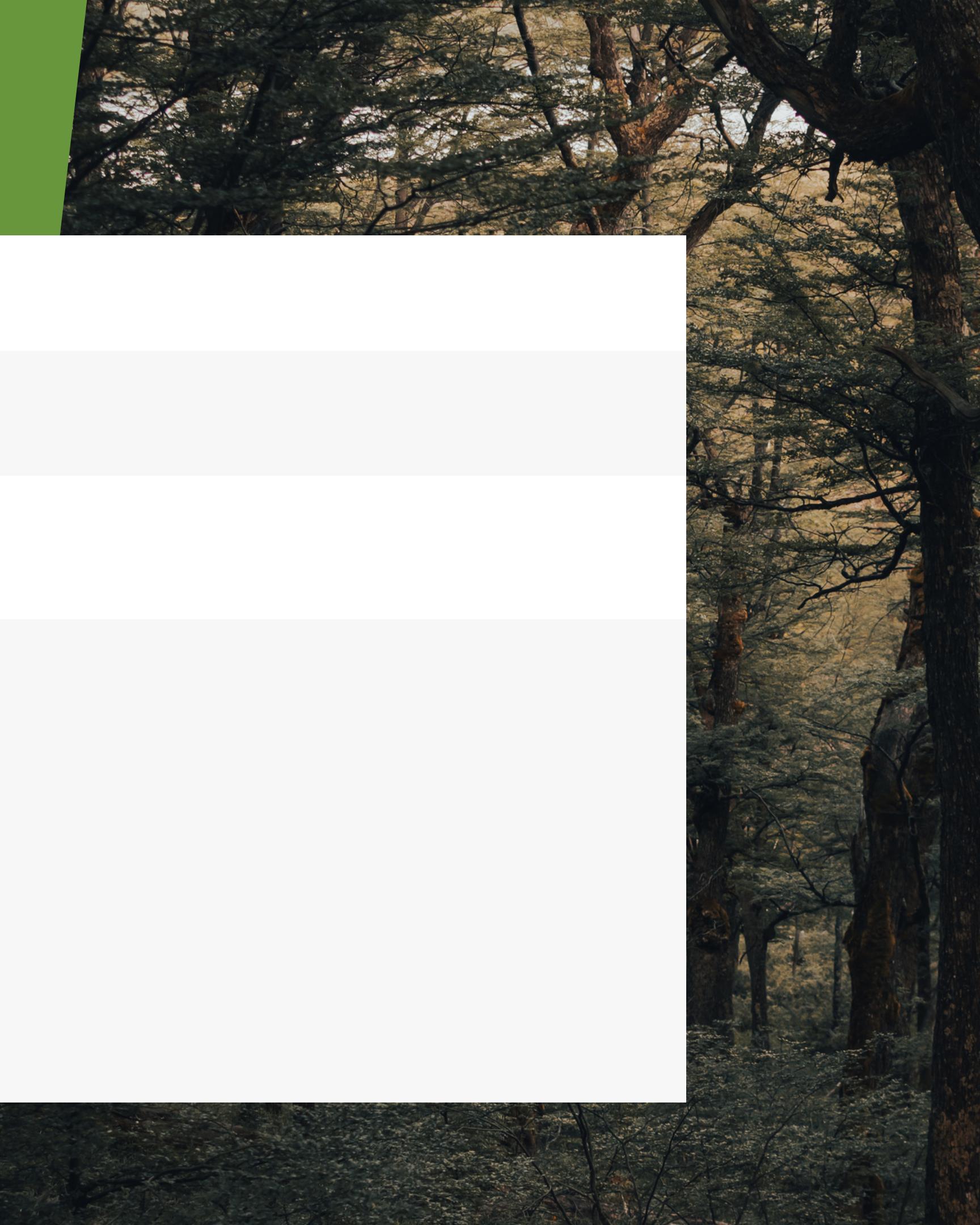
PYTHON

Librerias

```
[ ] import numpy as np  
import matplotlib.pyplot as plt
```

Condiciones Inciales

```
[ ] #Condiciones  
t0 = 0.0      # Tiempo de inicio  
tf = 30.0     # Tiempo final  
h = 0.01      # Paso  
Nx0 = 20.0     # Condición inicial presas  
Ny0=20.0      # Condición inicial depredadores  
#Parametros  
a = 1.0  
b = 0.02  
c = 1.0  
d = 0.01
```



Implementación Eulero

$$x(t + \Delta t) = x(t) + \frac{dx}{dt} \Delta t$$

$$y(t + \Delta t) = y(t) + \frac{dy}{dt} \Delta t$$

$$\begin{cases} \dot{x} = ax - bxy \\ \dot{y} = -cy + dxy \end{cases}$$

```
[ ] nt = int((tf-t0)/h) #número de pasos
Nx = np.empty(nt+1)
Nx[0]=Nx0

Ny =np.empty(nt+1)
Ny [0]=Ny0

for step in range(nt):
    Nx[step+1] =Nx[step] + ((a * Nx[step] - b * Nx[step] * Ny[step])*h)
    Ny[step+1]=Ny[step]+((- c * Ny[step] + d* Nx[step] * Ny[step])*h)

[ ] print('Función aproximada:',Nx)

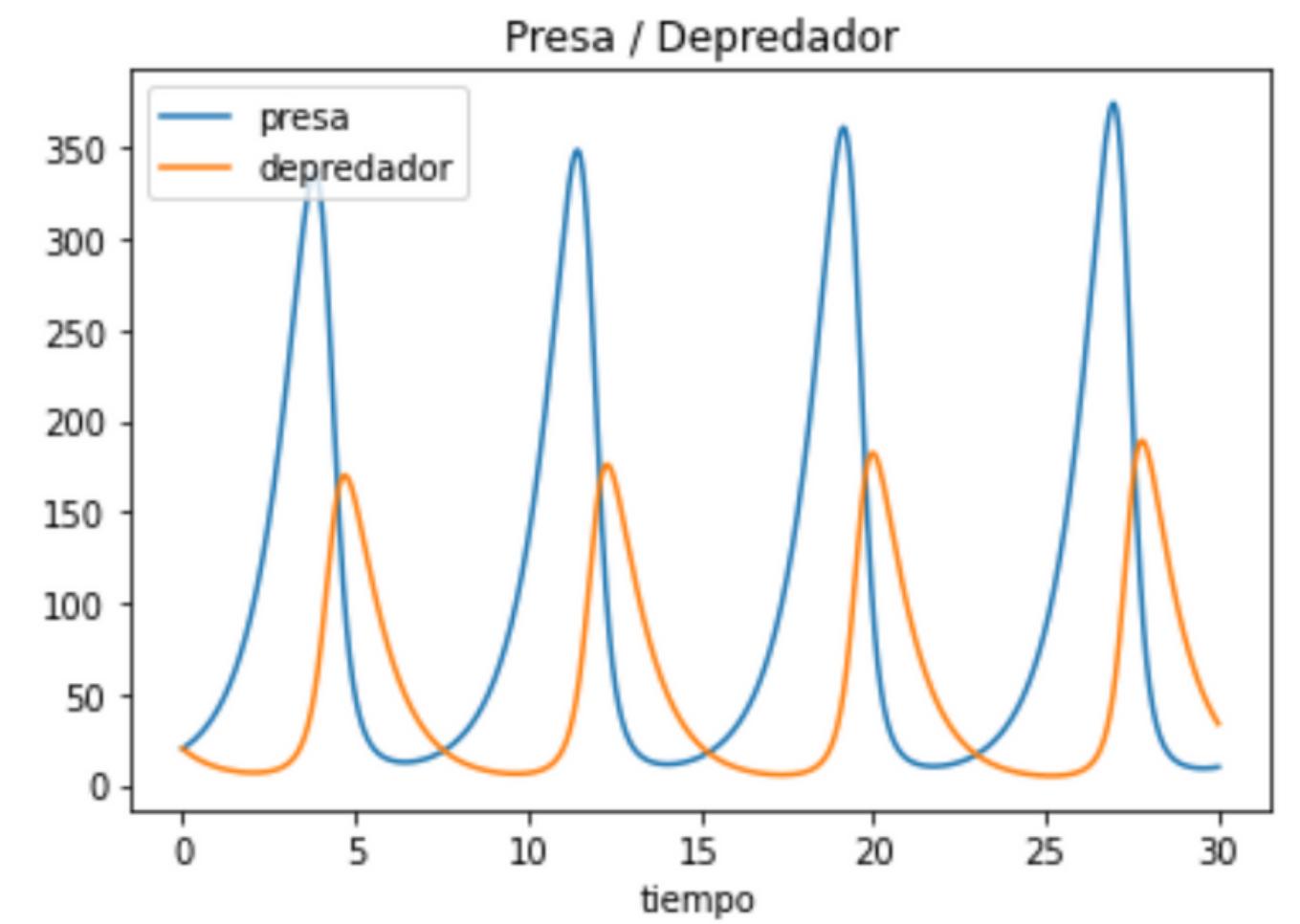
Función aproximada: [20.
9.85394745] 20.12 20.24136384 ... 9.79170243 9.82247278

[ ] print('Función aproximada:',Ny)

Función aproximada: [20.
33.67182811] 19.84 19.68151808 ... 34.28753748 33.97823544
```

```
[22] t = np.arange(nt+1)* h  
plt.plot(t, Nx, label='presa')  
plt.plot(t, Ny, label='depredador')  
plt.xlabel('tiempo')  
plt.legend(loc = 'best')  
plt.title('Presa / Depredador')
```

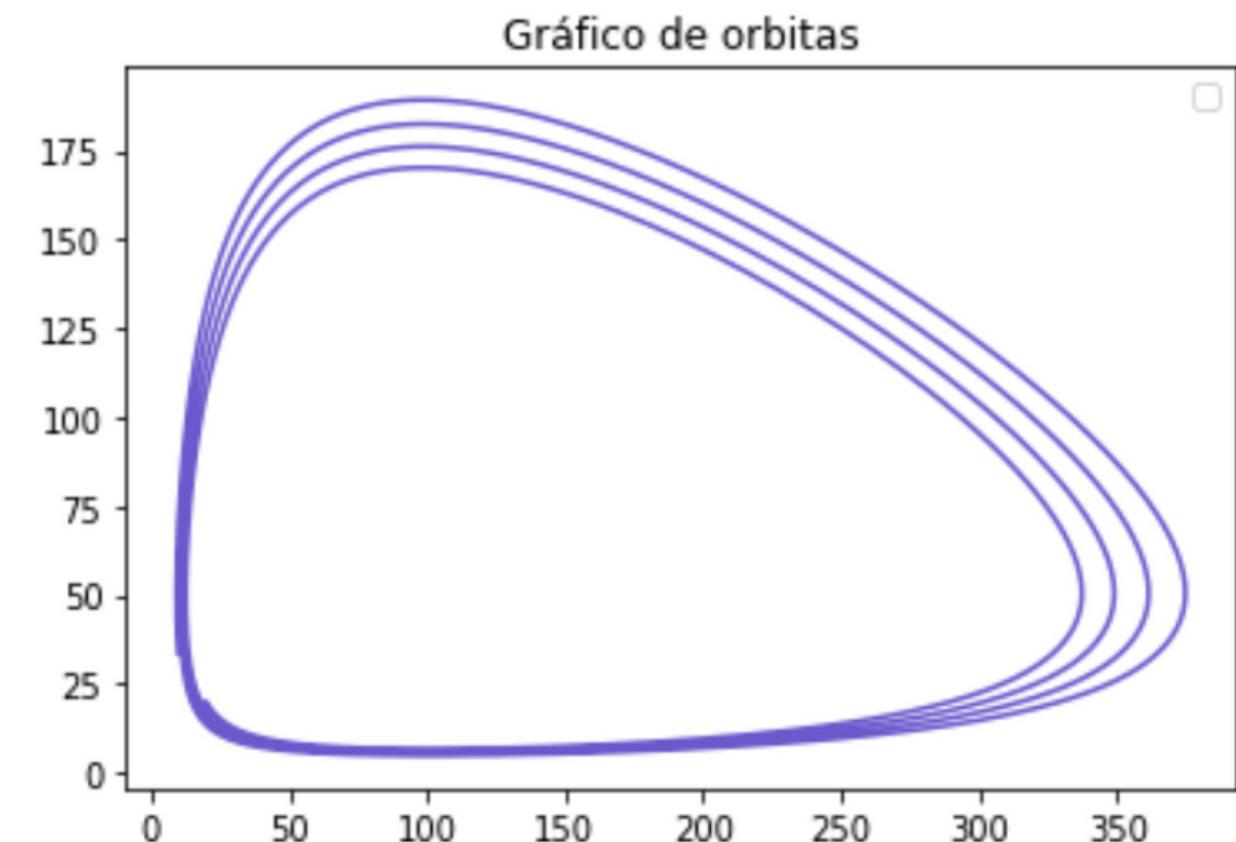
Text(0.5, 1.0, 'Presa / Depredador')





```
plt.plot(Nx,Ny, color ='slateblue')
plt.legend(loc='best')
plt.title('Gráfico de orbitas')
plt.show()
```

WARNING:matplotlib.legend:No handles with labels found to put in legend.



Odeit

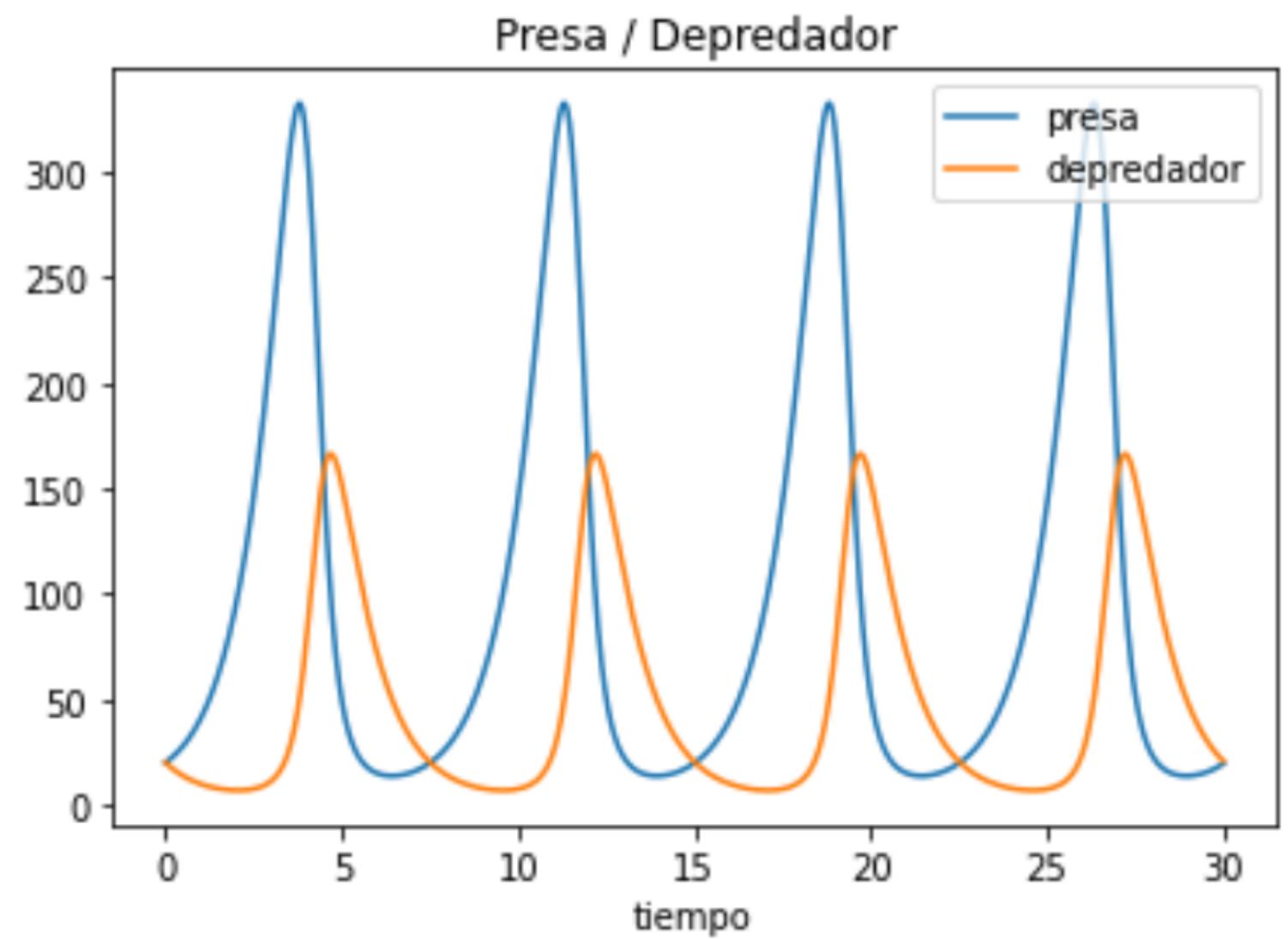
```
[ ] import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
def df_dt(x, t, a, b, c, d):
    """Función del sistema en forma canónica"""
    dx = a * x[0] - b * x[0] * x[1]
    dy = - c * x[1] + d * x[0] * x[1]
    return np.array([dx, dy])
# Parámetros
a = 1
b = 0.02
c = 1
d = 0.01
# Condiciones iniciales
x0 = 20 # Presas
y0 = 20 # Depredadores
conds_iniciales = np.array([x0, y0])
# Condiciones para integración
tf = 30
N = 3001
t = np.linspace(0, tf, N)
solucion = odeint(df_dt, conds_iniciales, t, args=(a, b, c, d))
```

$$\begin{cases} \dot{x} = ax - bxy \\ \dot{y} = -cy + dxy \end{cases}$$



```
plt.plot(t, solucion[:, 0], label='presa')
plt.plot(t, solucion[:, 1], label='depredador')
plt.legend(loc = 'best')
plt.title('Presa / Depredador')
plt.xlabel('tiempo')
```

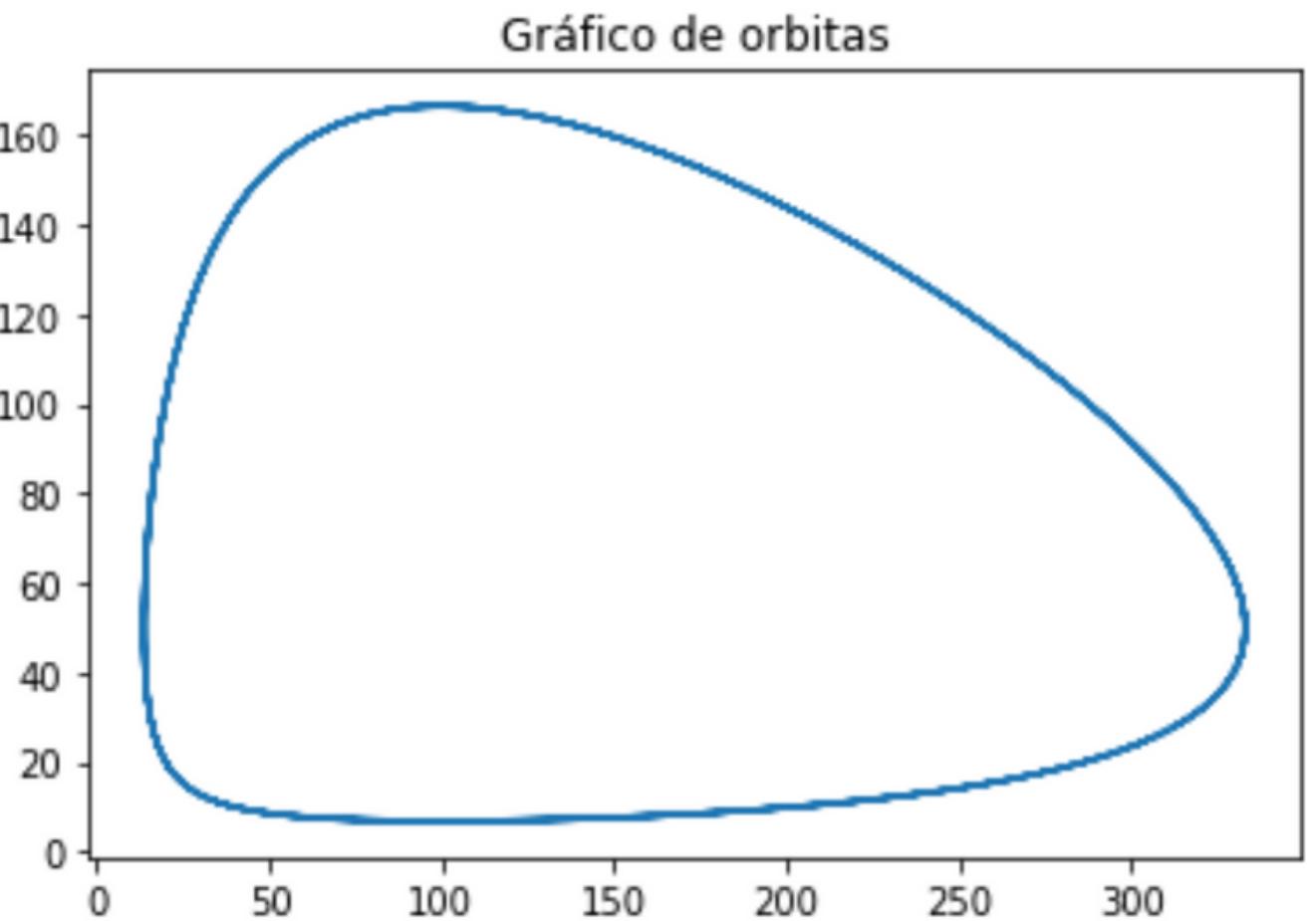
```
Text(0.5, 0, 'tiempo')
```



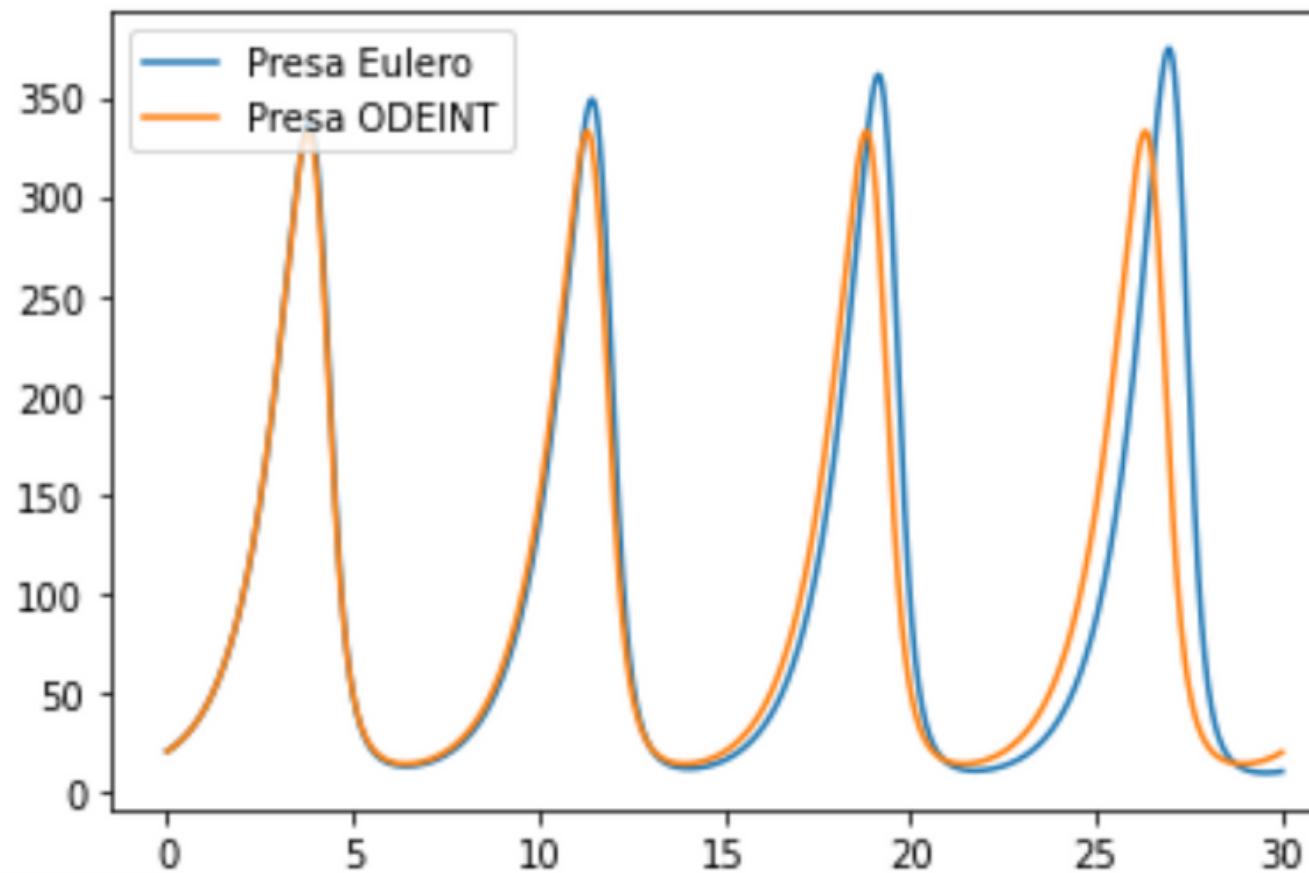


```
x_max = np.max(solucion[:,0]) * 1.05  
y_max = np.max(solucion[:,1]) * 1.05  
x = np.linspace(0, x_max, 25)  
y = np.linspace(0, y_max, 25)  
xx, yy = np.meshgrid(x, y)  
uu, vv = df_dt((xx, yy), 0, a, b, c, d)  
norm = np.sqrt(uu**2 + vv**2)  
plt.plot(solucion[:, 0], solucion[:, 1])  
plt.title('Gráfico de orbitas')
```

```
↳ Text(0.5, 1.0, 'Gráfico de orbitas')
```

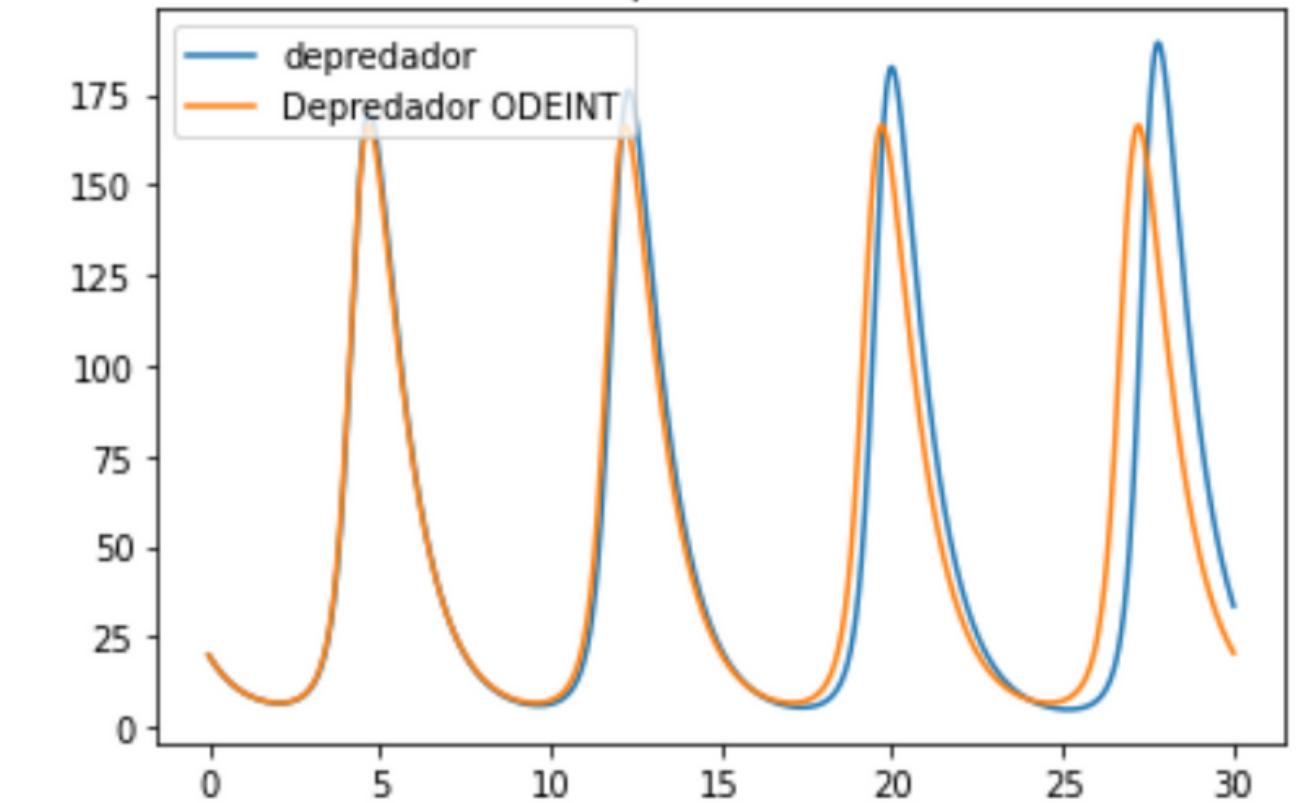


Presas



Presas

Depredadores



Depredadores