MODELO LOTKA-VOLTERRA

August 20, 2024

1 Universidad Autónoma del Estado de México

2 Centro Universitario UAEM Zumpango

2.1 Ingenieria en computación

2.1.1 Graficación Computacional

Alumno: Diego Gómez Tagle González

Profesor: Hazem Alvarez Rodriguez

Fecha: 14 de agosto del 2024

Descripción: Práctica L-V Python

2.2 Qué es el Modelo?

El modelo presa -depredador, también conocido como el modelo Lotka -Volterra, se ocupa de la interacción entre dos especies, donde una de ellas (presa) tiene abundante comida, y la segunda especie (depredador) tiene suministro de alimentos exclusivamente a la población de presas

2.3 IMPORTACION DE MODULOS

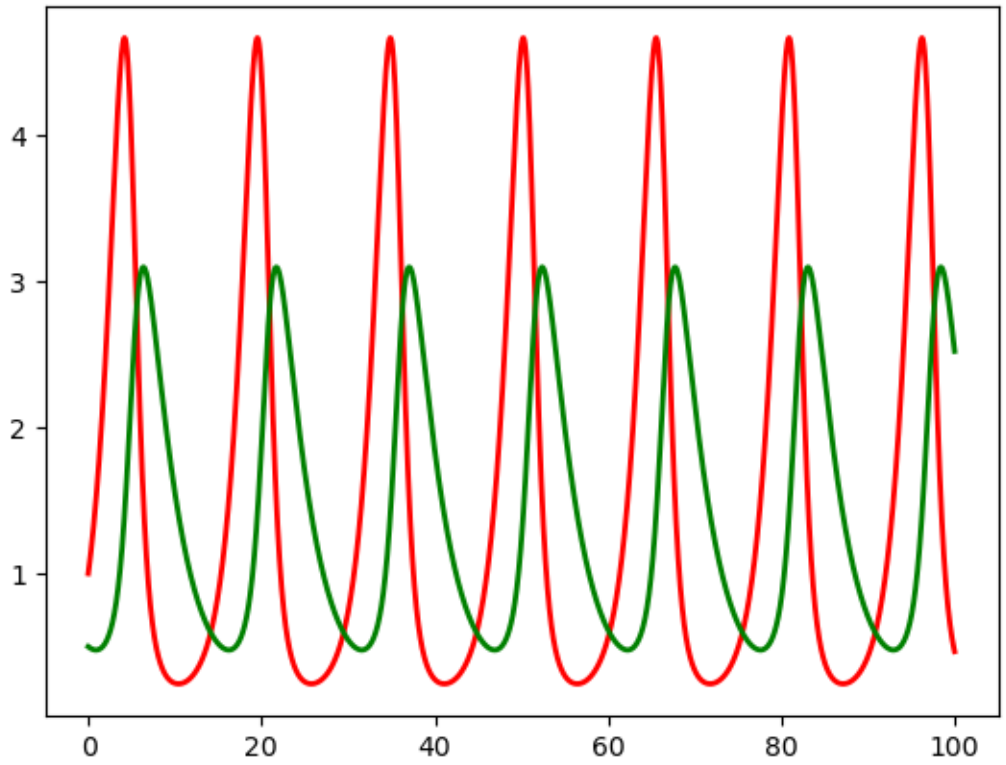
[4]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys
```

```
Matplotlib is building the font cache; this may take a moment.
```

2.4 SOLUCION DE LA ECUACIÓN LOTKA-VOLTERRA

[8]:
```
# parametros del modelo
a = 0.7;  # Tasa de crecimiento de la presa
b = 0.5;  # Tasa de depredación
c = 0.3;  # Tasa de mortalidad del depredador
e = 0.2   # Eficiencia de depredación
dt = 0.001 # Diferencia de tiempo
tiempo_max = 100 # Tiempo maximo a alcanzar
```

```
# Tiempo inicial y poblacion
t = 0;
x = 1.0 #Presas
y = 0.5 #Depredadores

#listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = [] #Lista de tiempo
lista_x = [] #Lista de Presas
lista_y = [] #Lista de Depredadores

# Inicialisa las listas
lista_t.append(t); lista_x.append(x); lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las graficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth = 2)
```



2.5 1.Crecimiento Descontrolado de presas

a: Alto

b: Bajo

c: Bajo

e: Bajo

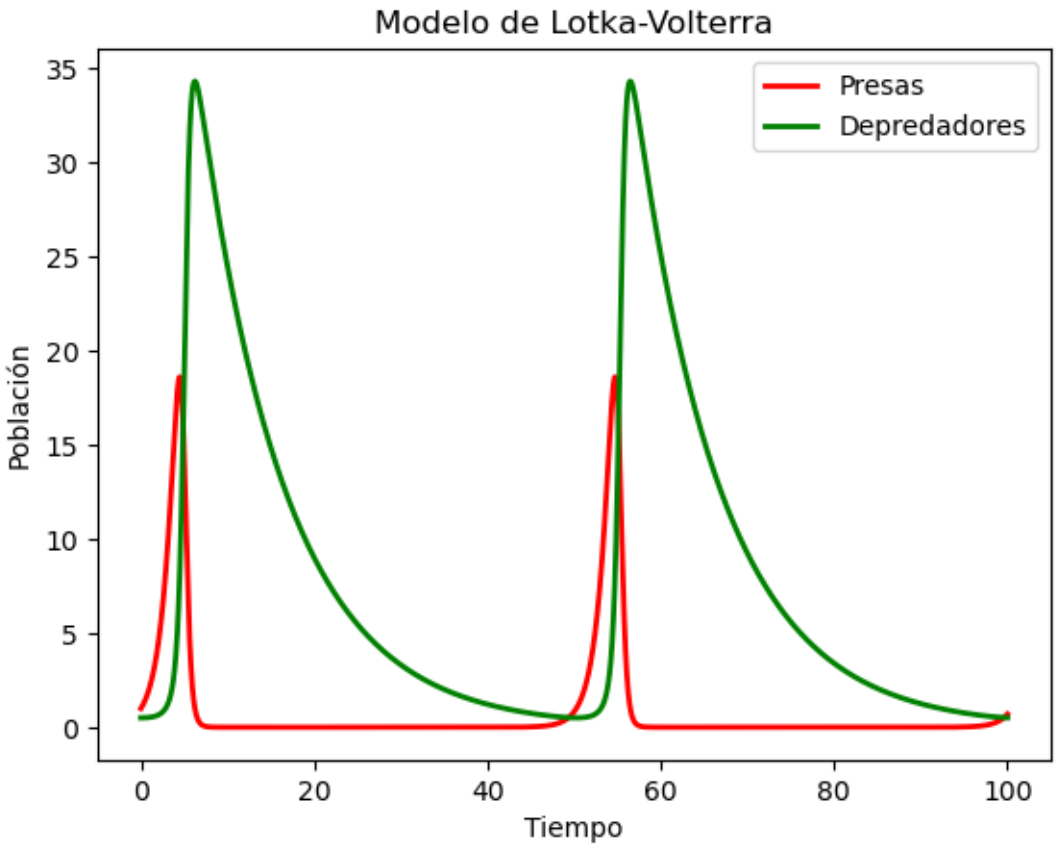
dt: Pequeño

tiempo_max: Alto

En este escenario, el crecimiento de la población de presas será alto debido a un valor alto de a, y la tasa de depredación es baja. Esto podría resultar en un crecimiento descontrolado de la población de presas, con depredadores incapaces de mantener su número bajo control.

[11]:
```
# parametros del modelo
a = 0.8  # Tasa de crecimiento de la presa
b = 0.1  # Tasa de depredación
c = 0.1  # Tasa de mortalidad del depredador
e = 0.1   # Eficiencia de depredación
dt = 0.001 # Diferencia de tiempo
tiempo_max = 100 # Tiempo maximo a alcanzar
```

```
# Tiempo inicial y poblacion
t = 0
x = 1.0 #Presas
y = 0.5 #Depredadores

#listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = [] #Lista de tiempo
lista_x = [] #Lista de Presas
lista_y = [] #Lista de Depredadores

# Inicialisa las listas
lista_t.append(t); lista_x.append(x); lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las graficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth = 2)
plt.xlabel('Tiempo')
plt.ylabel('Población')
plt.title('Modelo de Lotka-Volterra')
plt.legend(['Presas', 'Depredadores'])
plt.show()
```



2.6 2.Extinsion de Presas

a: Bajo

b: Alto

c: Bajo

e: Bajo

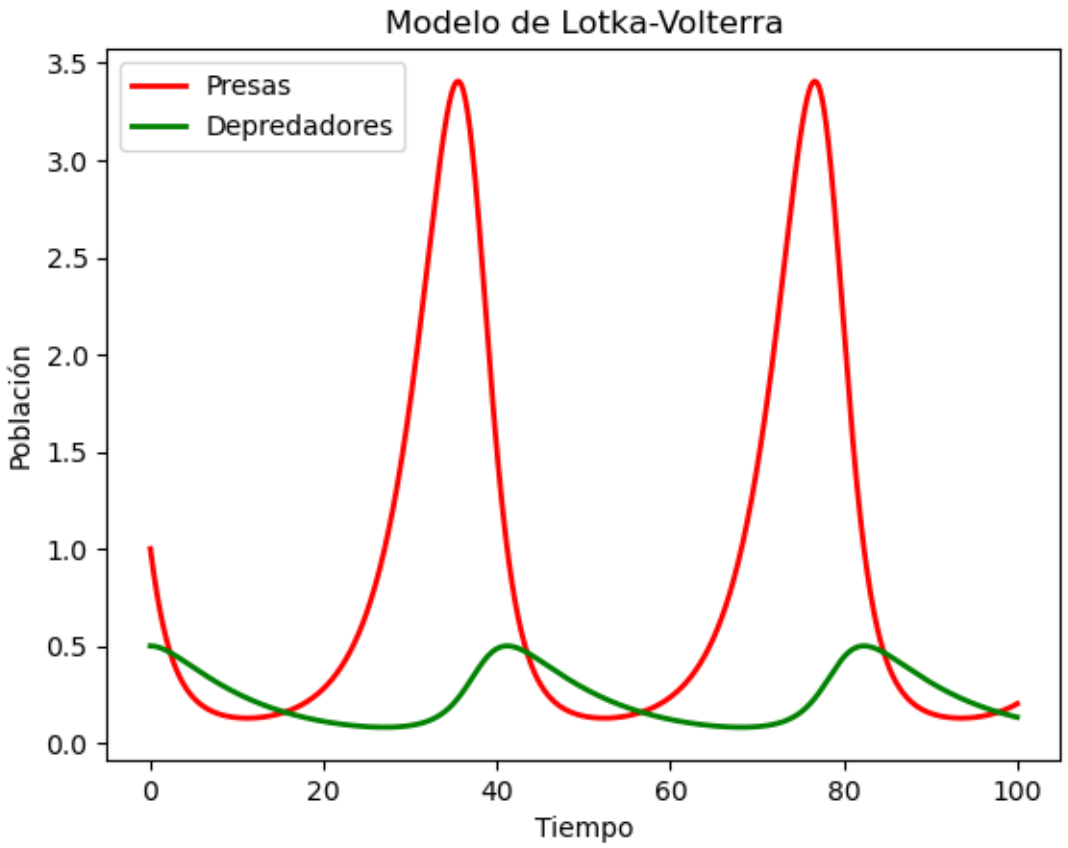
dt: Pequeño

tiempo_max: Alto

Con una tasa de crecimiento de presas baja y una alta tasa de depredación, la población de presas puede extinguirse rápidamente. Los depredadores pueden sobrevivir por un tiempo, pero eventualmente también pueden desaparecer si no hay suficientes presas.

[14]:
```
# parametros del modelo
a = 0.3  # Tasa de crecimiento de la presa
b = 1.3  # Tasa de depredación
c = 0.1  # Tasa de mortalidad del depredador
e = 0.1   # Eficiencia de depredación
```

```
dt = 0.001 # Diferencia de tiempo
tiempo_max = 100 # Tiempo maximo a alcanzar

# Tiempo inicial y poblacion
t = 0
x = 1.0 #Presas
y = 0.5 #Depredadores

#listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = [] #Lista de tiempo
lista_x = [] #Lista de Presas
lista_y = [] #Lista de Depredadores

# Inicialisa las listas
lista_t.append(t); lista_x.append(x); lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las graficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth = 2)
plt.xlabel('Tiempo')
plt.ylabel('Población')
plt.title('Modelo de Lotka-Volterra')
plt.legend(['Presas', 'Depredadores'])
plt.show()
```



2.7 3.Ciclo Estable

a: Moderado

b: Moderado

c: Moderado

e: Moderado

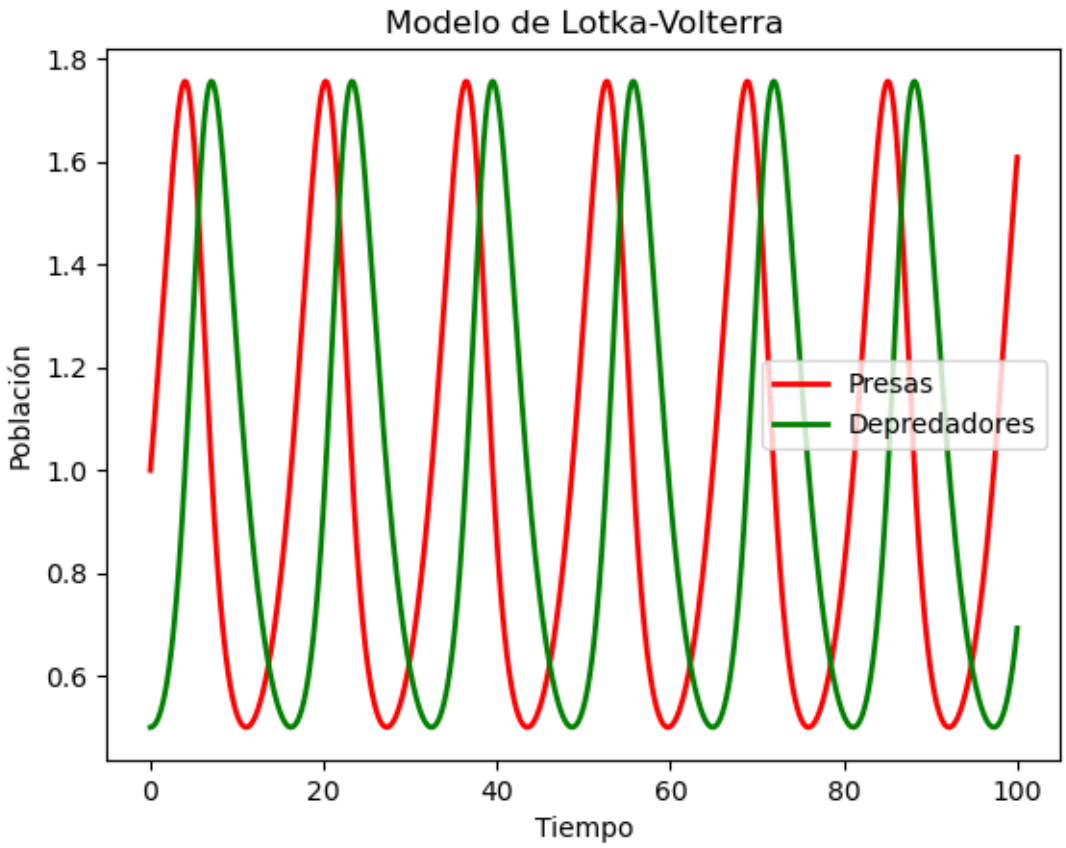
dt: Pequeño

tiempo_max: Alto

Valores moderados en todos los parámetros suelen llevar a un ciclo estable donde las poblaciones de presas y depredadores oscilan de manera regular alrededor de ciertos valores. Este es el comportamiento clásico del modelo de Lotka-Volterra.

[20]:
```
# parametros del modelo
a = 0.4  # Tasa de crecimiento de la presa
b = 0.4  # Tasa de depredación
c = 0.4  # Tasa de mortalidad del depredador
e = 0.4   # Eficiencia de depredación
```

```
dt = 0.001 # Diferencia de tiempo
tiempo_max = 100 # Tiempo maximo a alcanzar

# Tiempo inicial y poblacion
t = 0
x = 1.0 #Presas
y = 0.5 #Depredadores

#listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = [] #Lista de tiempo
lista_x = [] #Lista de Presas
lista_y = [] #Lista de Depredadores

# Inicialisa las listas
lista_t.append(t); lista_x.append(x); lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las graficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth = 2)
plt.xlabel('Tiempo')
plt.ylabel('Población')
plt.title('Modelo de Lotka-Volterra')
plt.legend(['Presas', 'Depredadores'])
plt.show()
```



2.8 4.Sobrevivencia de Depredadores

a: Bajo

b: Moderado

c: Bajo

e: Alto

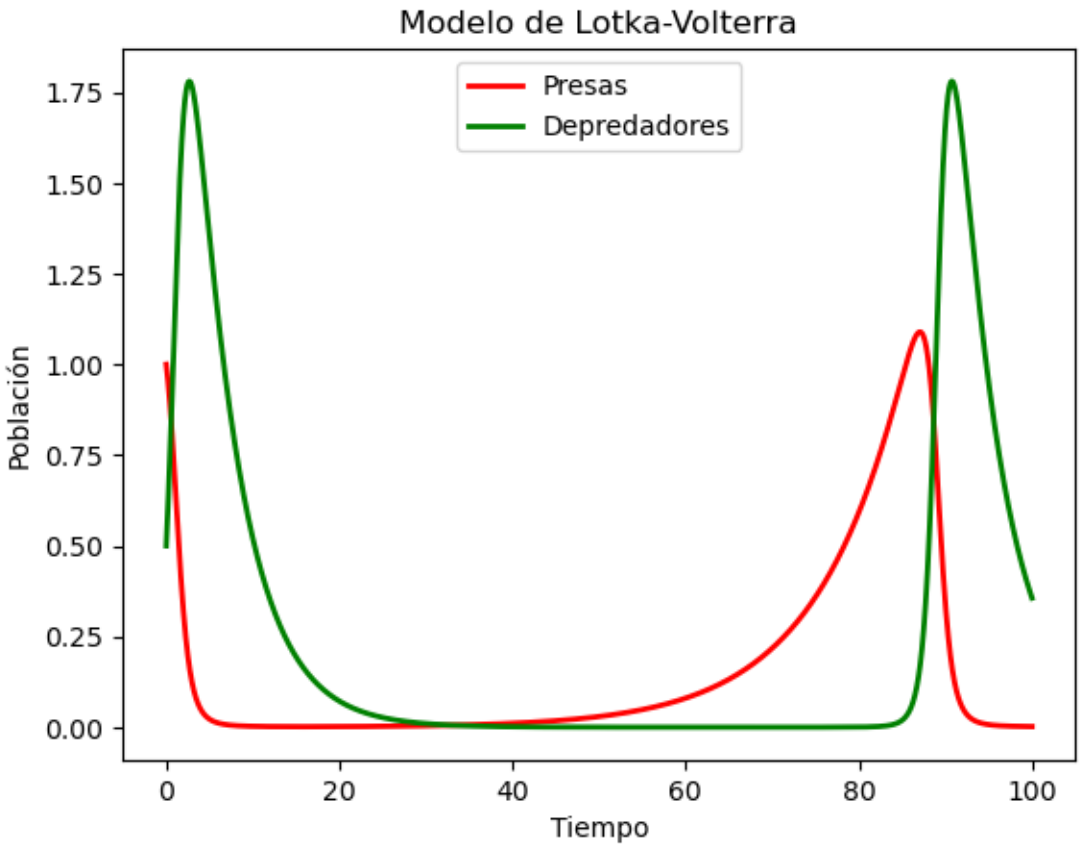
dt: Pequeño

tiempo_max: Alto

En este caso, una alta eficiencia de depredación (e) permite a los depredadores crecer a pesar de una tasa baja de crecimiento de presas y una baja tasa de mortalidad del depredador. Los depredadores pueden prosperar mientras que las presas no se reproduzcan lo suficientemente rápido.

[23]:
```
# parametros del modelo
a = 0.1  # Tasa de crecimiento de la presa
b = 0.6  # Tasa de depredación
c = 0.2  # Tasa de mortalidad del depredador
e = 1.2   # Eficiencia de depredación
```

```
dt = 0.001 # Diferencia de tiempo
tiempo_max = 100 # Tiempo maximo a alcanzar

# Tiempo inicial y poblacion
t = 0
x = 1.0 #Presas
y = 0.5 #Depredadores

#listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = [] #Lista de tiempo
lista_x = [] #Lista de Presas
lista_y = [] #Lista de Depredadores

# Inicialisa las listas
lista_t.append(t); lista_x.append(x); lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las graficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth = 2)
plt.xlabel('Tiempo')
plt.ylabel('Población')
plt.title('Modelo de Lotka-Volterra')
plt.legend(['Presas', 'Depredadores'])
plt.show()
```



2.9 5.Colapso del Ecosistema

a: Bajo

b: Alto

c: Alto

e: Alto

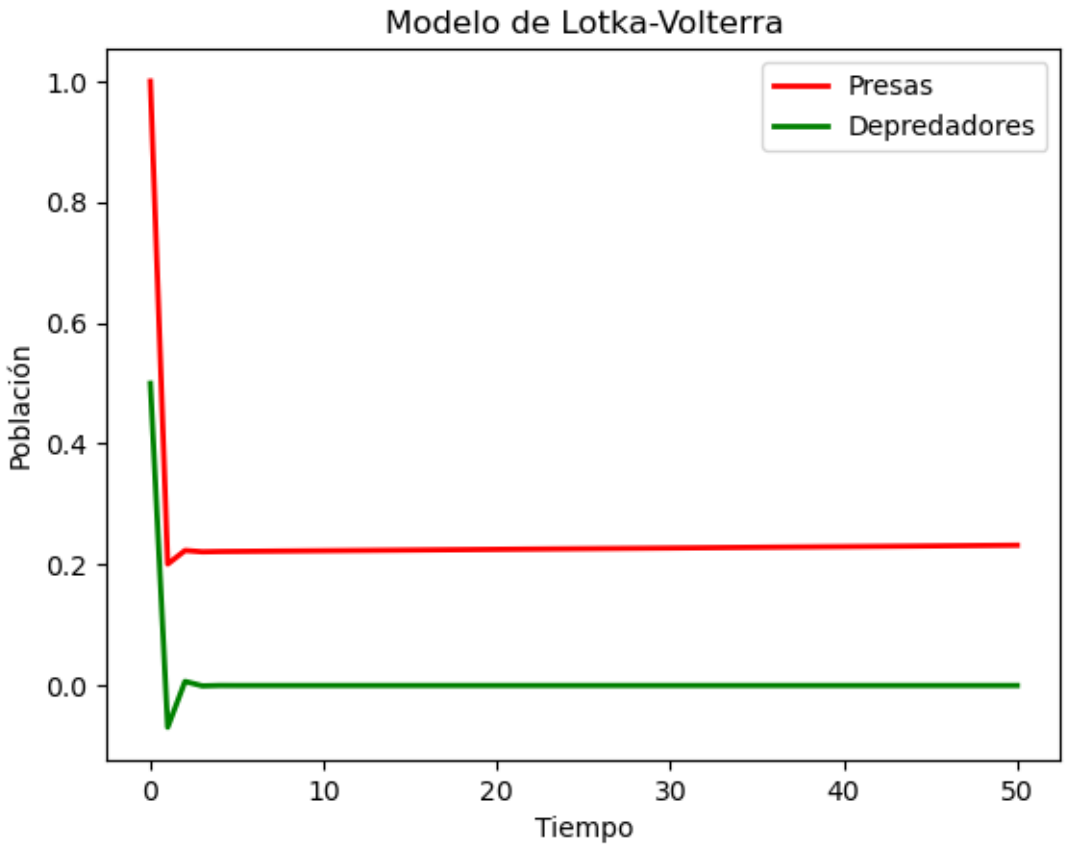
dt: Grande

tiempo_max: Bajo

En este escenario, todos los parámetros son altos, lo que puede llevar a un colapso rápido del ecosistema. La alta tasa de depredación y la alta eficiencia de depredación, junto con una baja tasa de crecimiento de presas y una alta tasa de mortalidad de los depredadores, pueden causar que ambas poblaciones colapsen en un corto período de tiempo.

[26]:
```
# parametros del modelo
a = 0.001  # Tasa de crecimiento de la presa
b = 1.6  # Tasa de depredación
c = 1.5  # Tasa de mortalidad del depredador
```

```
e = 1.8   # Eficiencia de depredación
dt = 1.0 # Diferencia de tiempo
tiempo_max = 50 # Tiempo maximo a alcanzar

# Tiempo inicial y poblacion
t = 0
x = 1.0 #Presas
y = 0.5 #Depredadores

#listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = [] #Lista de tiempo
lista_x = [] #Lista de Presas
lista_y = [] #Lista de Depredadores

# Inicialisa las listas
lista_t.append(t); lista_x.append(x); lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las graficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth = 2)
plt.xlabel('Tiempo')
plt.ylabel('Población')
plt.title('Modelo de Lotka-Volterra')
plt.legend(['Presas', 'Depredadores'])
plt.show()
```

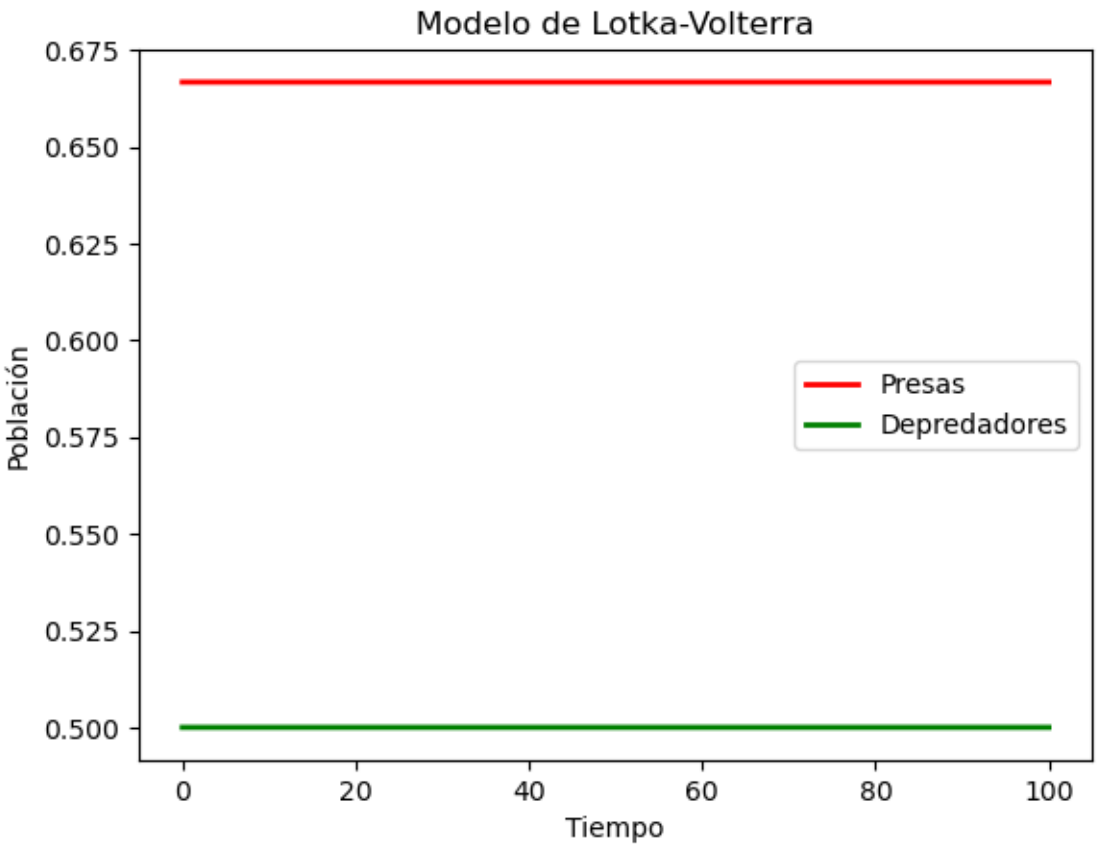


2.10 6.Ecuacion con equilibrio

Para que esta se encuentre en equilibrio, se deben igualar las ecuaciones a 0 y despejas con respecto a su variable (x o y según sea la ecuacion) Esto nos da el siguiente resultado: 0 = x − xy y 0 = xy − y de tal forma que, despejando nos da lo siguiente: y = y x = Sustituyendo en la ecuación nos da lo siguiente

[30]:
```
# Parámetros del modelo
a = 0.2  # Tasa de crecimiento de la presa (ajustada)
b = 0.4  # Tasa de depredación (ajustada)
c = 0.4  # Tasa de mortalidad del depredador
e = 0.6  # Eficiencia de depredación (ajustada)
dt = 0.001  # Diferencia de tiempo
tiempo_max = 100  # Tiempo máximo a alcanzar

# Tiempo inicial y población
t = 0
x = c/e  # Presas (cerca del equilibrio)
y = a/b  # Depredadores (cerca del equilibrio)
```

```
# Listas vacías en las que almacenar el tiempo y las poblaciones
lista_t = []  # Lista de tiempo
lista_x = []  # Lista de Presas
lista_y = []  # Lista de Depredadores

# Inicializa las listas
lista_t.append(t)
lista_x.append(x)
lista_y.append(y)

while t < tiempo_max:
    # Calcula los nuevos valores de t, x, y
    t = t + dt
    x = x + (a * x - b * x * y) * dt
    y = y + (-c * y + e * x * y) * dt

    # Agrega los valores en las listas
    lista_t.append(t)
    lista_x.append(x)
    lista_y.append(y)

# Muestra las gráficas
p = plt.plot(lista_t, lista_x, 'r', lista_t, lista_y, 'g', linewidth=2)
plt.xlabel('Tiempo')
plt.ylabel('Población')
plt.title('Modelo de Lotka-Volterra')
plt.legend(['Presas', 'Depredadores'])
plt.show()
```


Modelo de Lotka-Volterra
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
Población
0
20
40
60
80
100
Tiempo
Presas
Depredadores

[ ]: