Práctica L-V PythonTarea1

August 20, 2024

0.1 Universidad Autonoma del Estado de México

0.2 Centro Universitario UAEM Zumpango

0.3 Ingenería en Computación

0.4 Graficación Computacional

Alumno: Diego Argel Navarrete Godines

Profesora: Hazem Álvarez Rodríguez

Fecha: 19 de agosto del 2024

1 Modelo Lotka – Volterra

1.1 El modelo depredador-presa es un conjunto de ecuaciones matemáticas que describe las inámicas de poblaciones biológicas, particularmente en el contexto de las interacciones entre depredadores y presas. Fue propuesto de forma independiente por Alfred J. Lotka(1880–1949) en 1925 y por Vito Volterra (1860–1940) en 1926. Este modelo se utiliza para analizar la evolución de dos especies que interactúan en un entorno, donde una actúa como presa y la otra como depredador.

1.2 El modelo se compone de dos ecuaciones diferenciales de primer grado:

1.3 Ecuación para la población de presas:

1.4 dP / dt = alfa$N$ − $Beta$N*P

1.5 Donde:

1.6 N es la población de las presas.

1.7 P es la población de los depredadores.

1.8 Alfa es la tasa de crecimiento natural de las presas.

1.9 Beta es la tasa de depredación o la tasa a la cual los depredadores capturan presas

1.10 Ecuación para la población de depredadores:

1.11 dP / dt = DeltaNP − $Gamma$P

1.12 Donde:

1.13 Delta es la tasa de crecimiento de la población que depende de la presas.

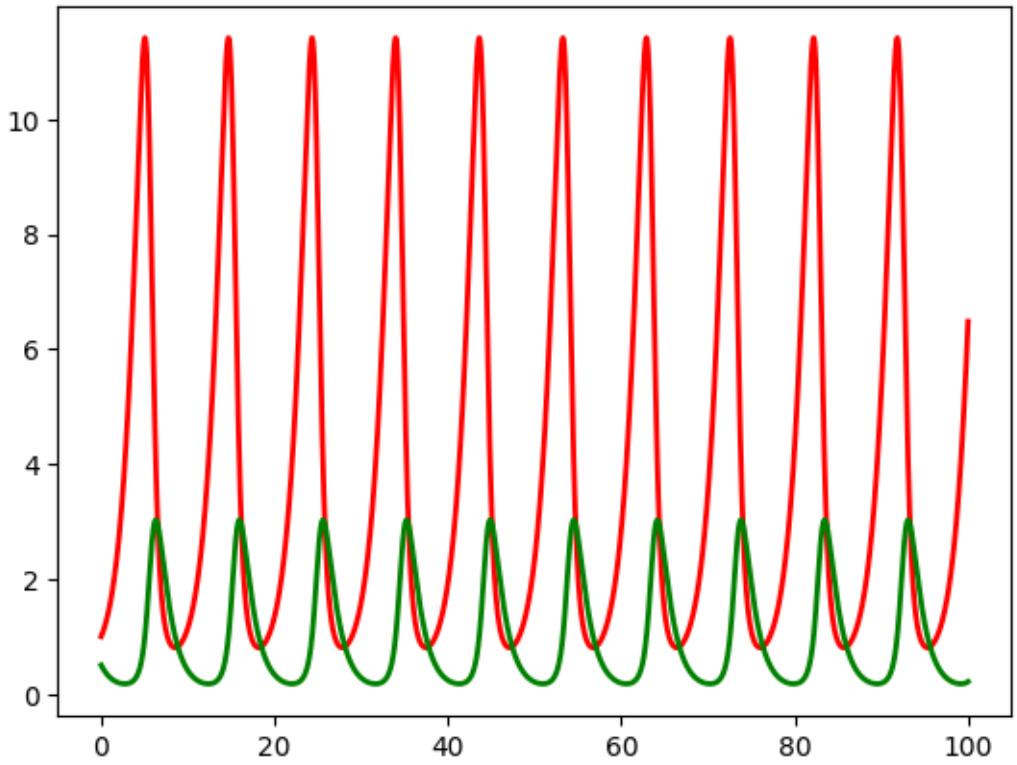
1.14 Gamma es la tasa de mortalidad natural de los depredadores.

1.15 Para aplicar el modelo de depredador – presa de Lotka – Volterra hablando computacionalmente, debemos adatar sus ecuaciones diferenciales a un formato que pueda ser manejado por un lenguaje de programación en este caso es Python, estas ecuaciones nos ayudaran a describir como las poblaciones cambian con el tiempo, este enfoque convierte las ecuaciones diferenciales que se pueden resolver numéricamente mediante la iteración, esto nos permite simular el comportamiento y podemos observar las poblaciones a largo del tiempo y analizar su evolución graficando los resultado utilizando Python.

2 El siguiente ejemplo es la implementación del modelo de depredador – presa en Python.

[1]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
```

```
import sys

# model parameters
# rojo = presas , verdes = depredadores
# a = aumento natural, b = destruccion por depredadores, c = muerte en ausencia␣
 ↪de presa y e = aumento causado alimentacion alimentacion
a = 0.7; b = 0.7; c = 0.8;  e = 0.2

# punto de equilibrio a = 0.3; b = 0.2; c = 0.6; e = 0.3
#a = 0.8; b = 0.3; c = 0.3;  e = 0.2


dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 0.5

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)
plt.show()
```



Primero, se importan las librerías necesarias, aunque algunas no se usan en este fragmento. Luego, se definen los parámetros del modelo y se inicializan las variables de tiempo y poblaciones. Se crean listas para almacenar los resultados de tiempo y poblaciones durante la simulación.

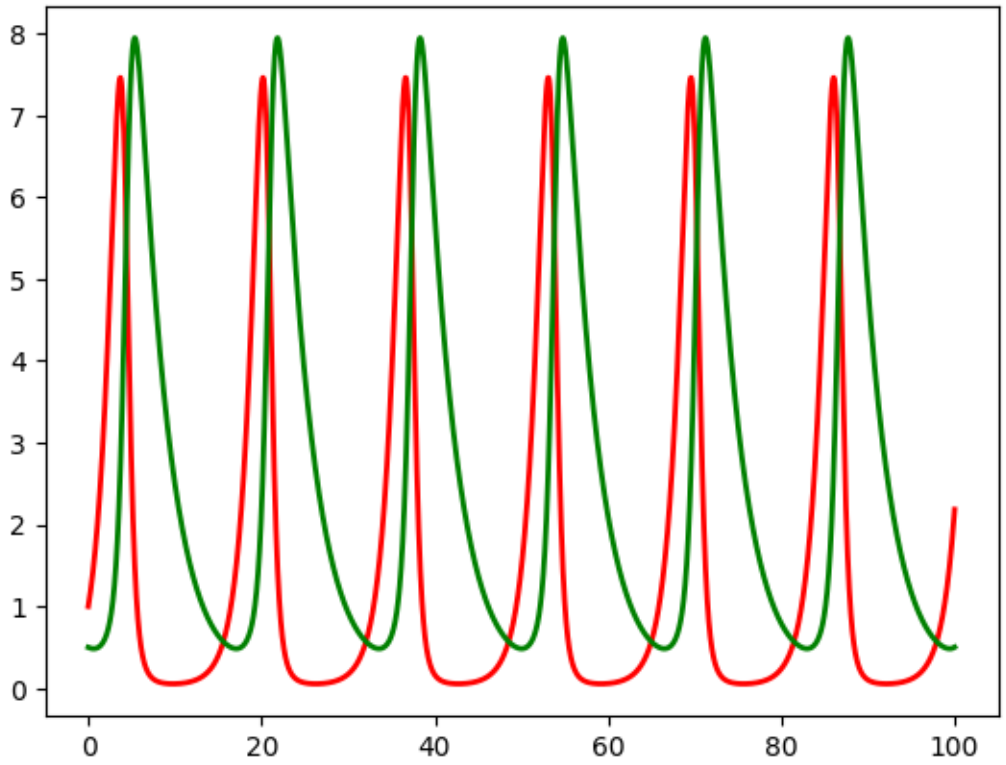
2.1 A continuación, se mostrar 5 distintas variaciones en los parámetros durante la simulación de las poblaciones.

2.2 Las líneas rojo representa las presas y las líneas verdes representan los depredadores enla población. El parámetro "a" es el aumento natural de la especie, el parámetro "b" es la destrucción por depredadores, el parámetro "c" es la muerte en ausencia de presa y el parámetro "e" es el aumento causado por la alimentación. Dicho lo anterior podemos observar la variante uno.

2.3 variante 1 a = 0.8, b = 0.3, c = 0.3 y e = 0.2

[1]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys
```

```
# model parameters
a = 0.8; b = 0.3; c = 0.3; e = 0.2
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 0.5

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)
plt.show()
```

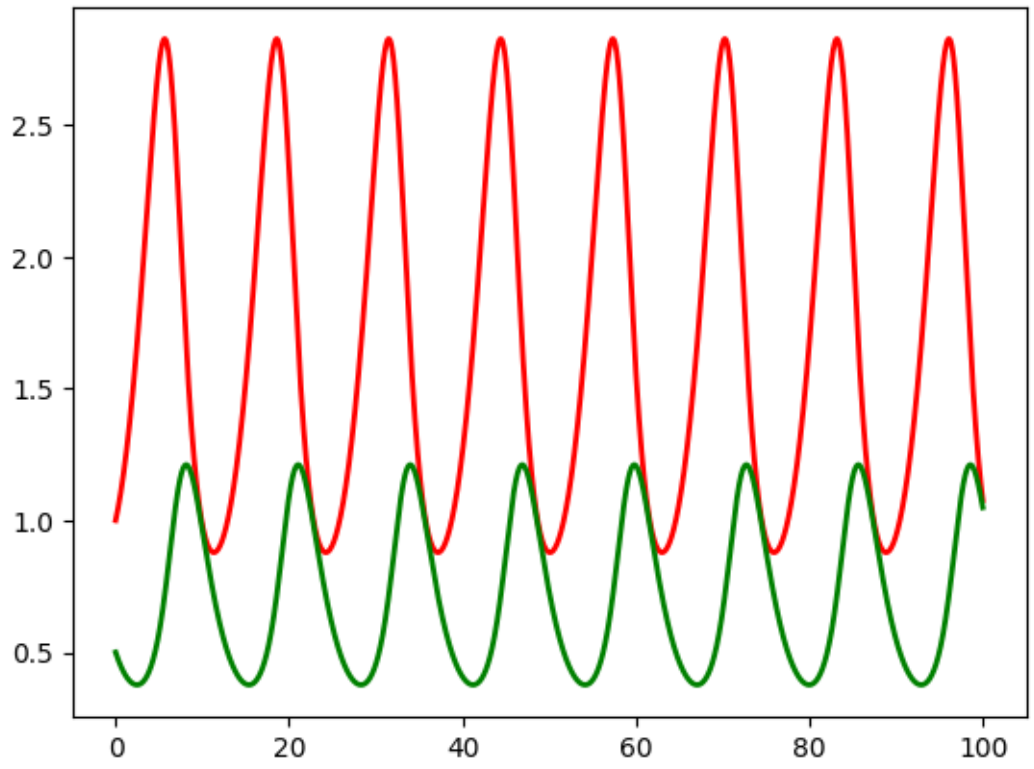


2.3.1 **Al tener un índice relativamente alto en el aumento natural de la especie en 0.8 lo que significa presas crecen de manera rápida y la ausencia de los depredadores lo que lleva a un cantidad de presas considerablemente altas con una destrucción por los depredadores con un valor de 0.3 en este caso son muy eficaces en la captura de sus presas pero a la vez pueden acabar con la mayoría de las presas, la muerte en ausencia de las presas con un 0.3 y por el aumento causado por la alimentación disponible.**

2.4 variante 2 a = 0.5, b = 0.7, c = 0.5 y e = 0.3

[2]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys

# model parameters
a = 0.5; b = 0.7; c = 0.5; e = 0.3
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 0.5
```

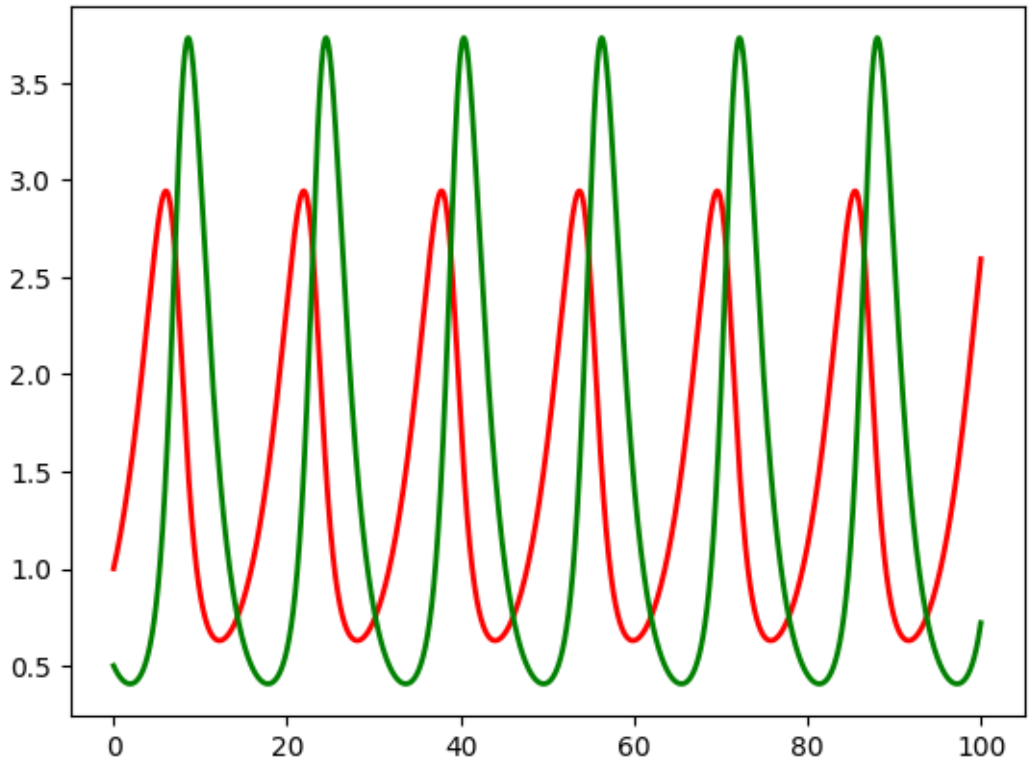
```
# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)
plt.show()
```



2.4.1 En este caso las presas crecen más lentamente en ausencia de depredadores con un valor de 0.5 esto nos puede llevar a la tasa de depredación alta, lo que indica que los depredadores son muy eficientes a la ahora de cazar las presas con un valor de 0.7 pero los depredadores tienen una tasa de mortalidad moderada en ausencia de presas, lo que limita su supervivencia sin alimento con un valor del 0.5 y su tasa de crecimiento de los depredadores basada en la disponibilidad de presas es de 0.3 lo que puede llevar disponibilidad de presas para poder sobrevivir

2.5 variante 3 a = 0.3, b = 0.2, c = 0.6 y e = 0.4

[5]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys

# model parameters
a = 0.3; b = 0.2; c = 0.6; e = 0.4
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 0.5

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)
plt.show()
```

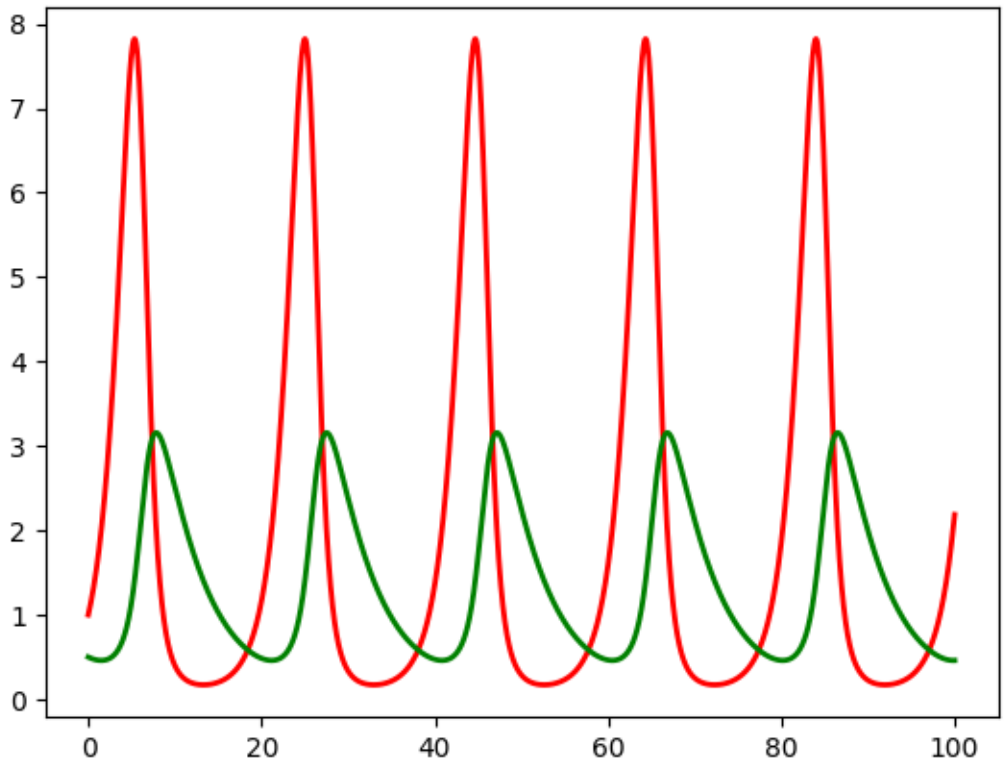


2.6 En la variante 3 el parámetros de las presas tienen una tasa de crecimiento natural muy baja puesto que crecen de manera lenta en ausencia de depredadores con un valor de 0.3, en la a tasa de depredación es baja con un valor de 0.2 lo que significa que los depredadores no son muy efectivos en la captura de presas, al tener una eficiencia en la casa le genera a los depredadores una alta tasa de mortalidad en ausencia de presas con un valor de 0.6 y por ultimo la tasa de crecimiento de los depredadores basada en la disponibilidad de presas es alta con un valor de 0.4.

2.7 variante 4 $a = 0.7$, $b = 0.5$, $c = 0.2$ y $e = 0.1$

[6]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys

# model parameters
a = 0.7; b = 0.5; c = 0.2; e = 0.1
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 0.5
```

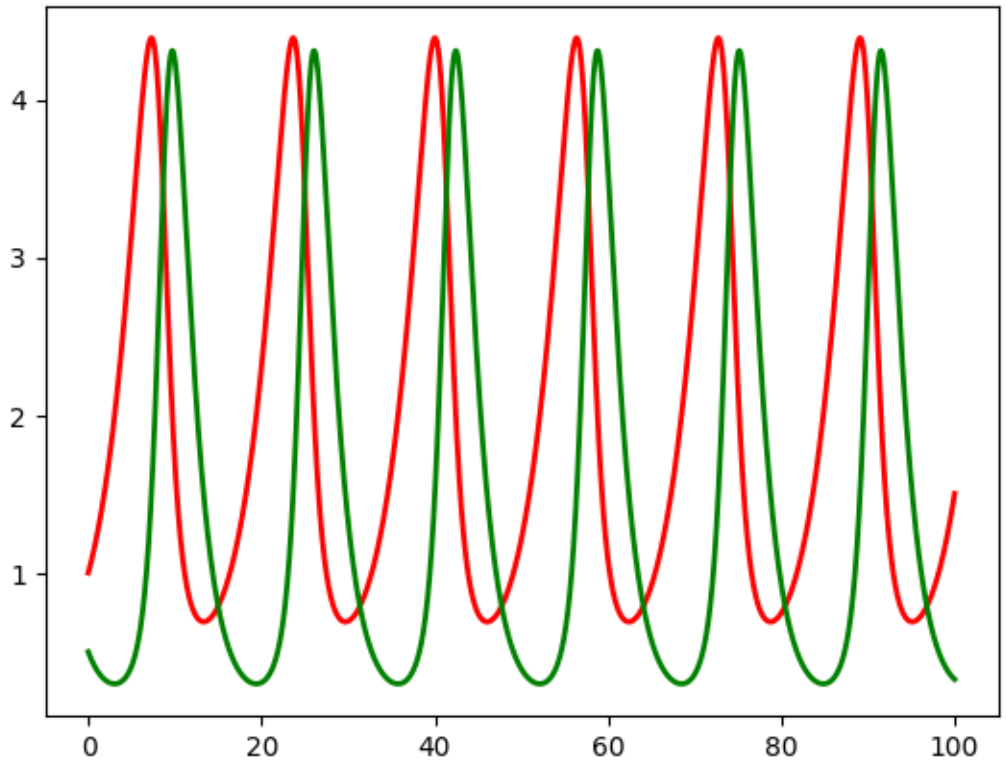
```
# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)
plt.show()
```



2.7.1 En la variante 4 en este caso las presas crecen a un ritmo relativamente muy elevadopuesto que hay ausencia de depredadores este toma el valor de 0.7 para la tasa de depredación es alta con un valor de 0,5 lo que significa que los depredadores son bastante rápidos en la captura de presas, la tasa de mortalidad de los depredadores en ausencia de presas es muy baja lo que permite sobrevivir un poco más de tiempo la tasa de crecimiento de los depredadores basada en la disponibilidad de presas es baja, por lo que los depredadores no crecen tan rápido a pesar la cantidad de presas es alta a comparación con los depredadores.

2.8 variante 5 (punto de equilibrio): a = 0.3, b = 0.2, c = 0.6 y e = 0.3

[7]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys

# model parameters
a = 0.3; b = 0.2; c = 0.6; e = 0.3
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 0.5

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)
plt.show()
```



2.8.1 En la última variante podemos encontrar casi un punto de equilibrio en ambas simulaciones, en este caso las presas tienen una tasa de crecimiento relativamente baja en ausencia de depredadores con un valor de 0.3 esto significa que el número de presas aumenta lentamente sin depredadores en otro de los caso a tasa de depredación es bajacon un valor de 0.2, lo que significa que los depredadores afectan moderadamente la población de presas, esto permite que las presas puedan crecer más rápido y su tasa de mortalidad de los depredadores es alta en ausencia de presas, esto nos indica que los depredadores mueren rápidamente si no tienen alimento por ello la tasa de crecimiento de los depredadores y significa que los depredadores pueden reproducirse a un ritmo razonable cuando hay suficiente alimento.