TAREA 3

- Materia: Matemáticas para Ciencia de Datos
- Programa: Maestría en Ciencia de Datos e Información, INFOTEC
- Docente: Juliho Castillo Colmenares, Sc.D.

Equipo A

- Erika Araceli González Villa
- Brayan Homero Ramírez Contreras
- David Aarón Ramírez Olmeda
- Rodrigo Guarneros Gutiérrez

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

Lectura asignada

Del libro "Cai, X., Tveito, A., Langtangen, H. P., Nielsen, B. F. (2010). Elements of Scientific Computing. Germany: Springer Berlin Heidelberg", revisa el capítulo 3 "System of Ordinary Differential Equations"

Instrucciones

1. Organízate con los miembros de tu equipo para comentar la lectura y la tarea.
2. Resuelvan el siguiente problema, desarrollando de manera clara y concisa todos y cada unos de los puntos.
3. Transcríbanlo a un archivo PDF utilizando un editor de textos y suban un único archivo por equipo.
4. No se aceptarán trabajos escritos a mano, aun cuando estén digitalizados.
5. Puedes utilizar software para resolver los problemas, pero en este caso deberás incluir el código en tu documento.
6. En cualquier caso, incluye el desarrollo completo de la solución. No se aceptarán respuestas sin justificación.
7. Se considerará un inciso como incorrecto si el resultado no es el esperado, y se considerará incompleto si el resultado no está debidamente justificado.
8. Para acreditar el punto correspondiente a cada inciso, este deberá estar completo y ser correcto.

Planteamiento del problema y respuestas

Considera el sistema

$$F' = (2 - S)F, F(0) = F_0$$
$$S' = (F - 1)S, S(0) = S_0$$

Inciso 1.

###Determina un esquema numérico explícito para aproximar la solución.

Para este efecto, se utiliza el método de Euler adelantado, que en resumen establece lo siguiente:

$$y_{n+1} = y_n + \Delta t * y'_n$$

Dado nuestro sistema de ecuaciones, podemos expresar cada ecuación diferencial de la siguiente forma para reflejar la dinámica de las presas F y los depredadores S:

$$F_{n+1} = F_n + \Delta t * F'_n$$
$$S_{n+1} = S_n + \Delta t * S'_n$$

Al sustituir las definiciones de las derivadas en el planteamiento original, obtenemos lo siguiente:

$$F_{n+1} = F_n + \Delta t * (2 - S_n) * F_n$$
$$S_{n+1} = S_n + \Delta t * (F_n - 1) * S_n$$

Para obtener una solución numérica, como se ha mencionado previamente, es necesario definir el horizonte temporal en el que opera el sistema (t=T), las condiciones iniciales para las dos funciones F_0 y S_0, y el tamaño de los pasos temporales, los cuales están definidos como $\Delta t = \frac{T}{N}$.

Inciso 2.

Encuentra una aproximación numérica a la solución en $t = 1$ si $F_0 = 1.9$, $S_0 = 0.1$, $\Delta t = 0.001$.

Como se mencionó anteriormente, con los parámetros dados, se puede utilizar el esquema numérico explícito para aproximar la solución. El número de iteraciones necesarias para cada ecuación serán 1000, teniendo en cuenta que el tamaño de paso $\Delta t = 0.001$. Por lo tanto, podemos calcular el número de pasos necesarios como $N = \frac{T}{\Delta t} \rightarrow N = \frac{1}{0.001} = 1000$.

$$F(0.001) = F_0 + \Delta t * (2 - S_0) * F_0$$
$$S(0.001) = S_0 + \Delta t * (F_0 - 1) * S_0$$

Lo que implica que:

$$F(0.001) = 1.9 + 0.001 * (2 - 0.1) * 1.9 = 1.9036099999999998$$
$$S(0.001) = 0.1 + 0.001 * (1.9 - 1) * 0.1 = 0.10009000000000001$$

$$F(0.002) = F(0.001) + \Delta t * (2 - S(0.001)) * F(0.001)$$
$$S(0.002) = S(0.001) + \Delta t * (F(0.001) - 1) * S(0.001)$$
$$F(0.002) = 1.9036099999999998 + 0.001 * (2 - 0.10009000000000001) * 1.903609999999$$
$$S(0.002) = 0.10009000000000001 + 0.001 * (1.9036099999999998 - 1) * 0.100090000000$$

$$[\ldots]$$

Y podríamos continuar con el cálculo numérico paso a paso. Aquí es donde presentamos un algoritmo en Python para automatizar el cálculo, dado el esquema explícito inicial y los parámetros establecidos.

Como resultado del programa que se presenta abajo, se obtiene la aproximación numérica a la solución, que se ilustra en la Figura 1 para $t \in [0, 1]$.

Figura 1
Solución numérica del sistema de ecuaciones con los parámetros establecidos en el Inciso 2

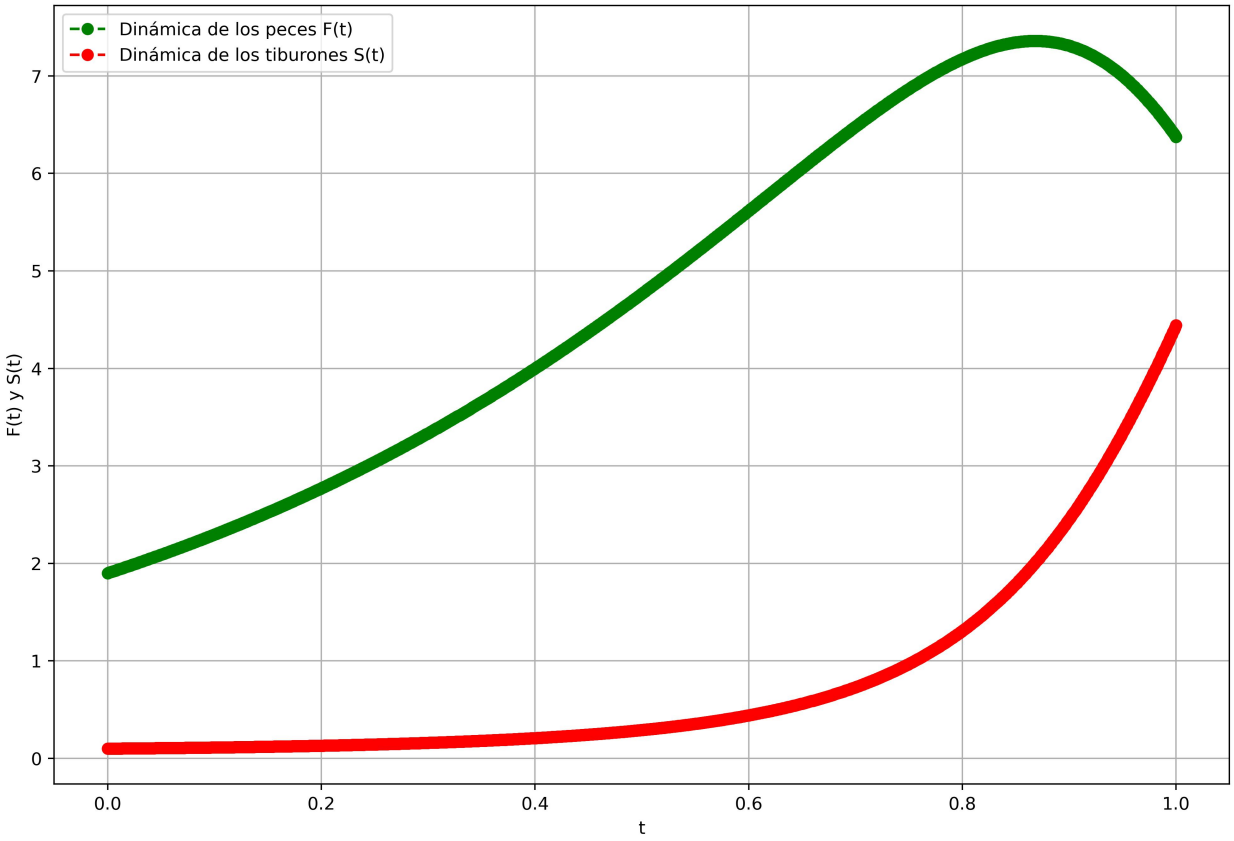
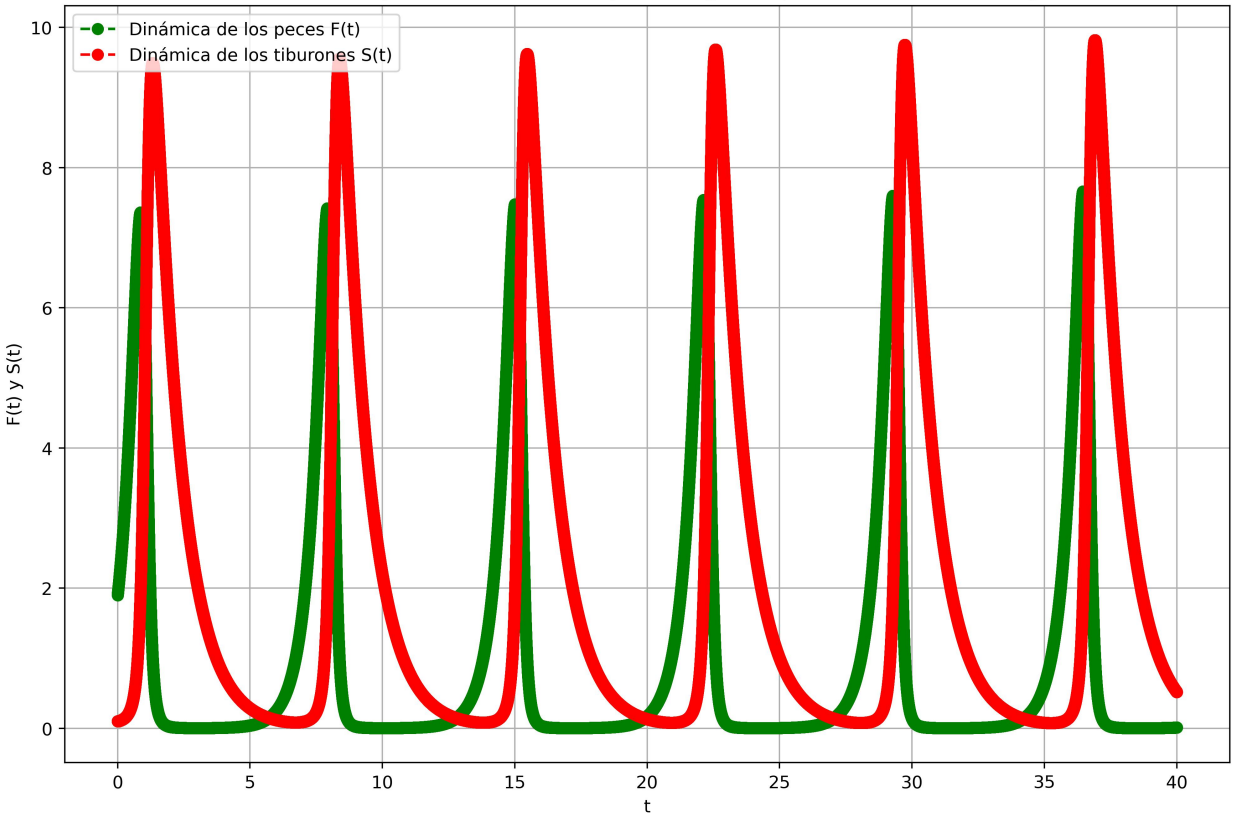


Figura 2
Solución numérica del sistema de ecuaciones con los parámetros establecidos en el Inciso 2 y T=15
donde se observa que es hasta el T=26 que se llega al equilibrio estable S(0)=0 y F(0)=0



Algo a considerar es que el resultado numérico obtenido del sistema de ecuaciones diferenciales en $t = 1$, dadas las condiciones iniciales, está dado por:

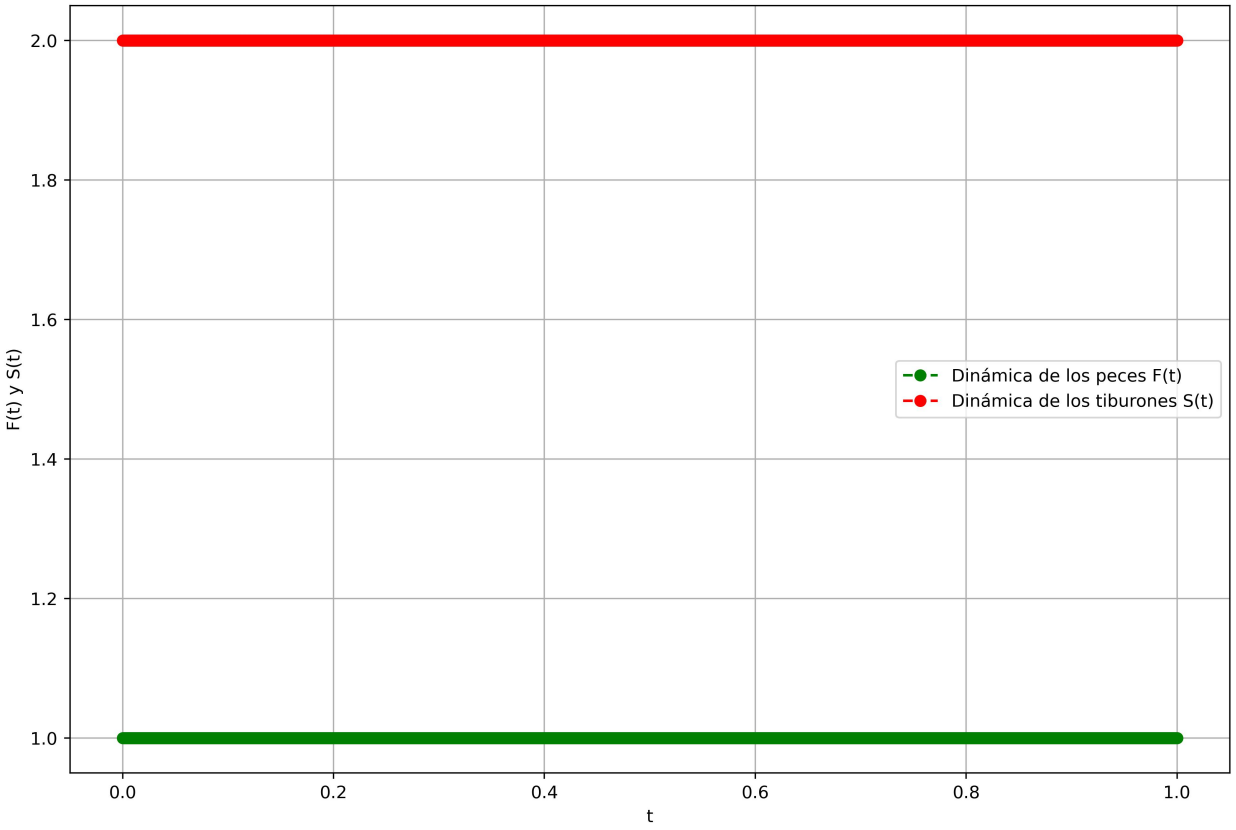
$$F(t = 1) = 6.37562979$$
$$S(t = 1) = 4.44102227$$

Inciso 3

##Encuentra una aproximación numérica a la solución en $t = 1$ si $F_0 = 1$, $S_0 = 2$, $\Delta t = 0.001$.

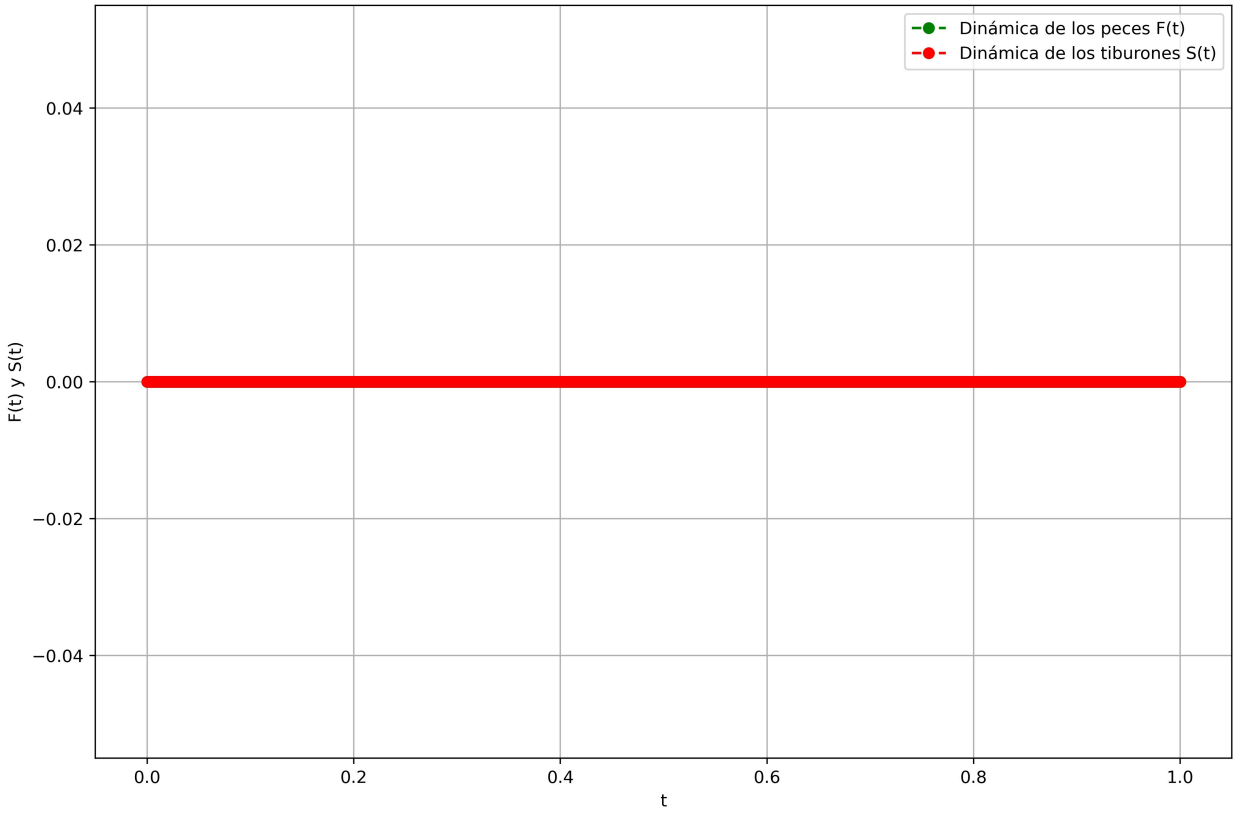
Como resultado de este ejercicio, se llega a un estado estacionario en el que tanto $F(t)$ como $S(t)$ toman valores constantes, de tal forma que: $F(t = 1) = 1$ y $S(t = 1) = 2$. Ver figura 3.

Figura 3
Solución numérica del sistema de ecuaciones con los parámetros establecidos en el Inciso 3



Otro estado estacionario o de equilibrio previsto por nuestro sistema ocurre cuando: $F(0) = 0$ y $S(0) = 0$. Ver figura 4.

Figura 4
Solución numérica del sistema de ecuaciones con el estado estacionario F(0)=0 y S(0)=0



Ambos estados estacionarios en el modelo se consideran razonables, $F_0 = 0$, $S_0 = 0$ y $F_0 = 1$, $S_0 = 2$, en virtud de que estamos hablando de poblaciones cuya principal característica es que siempre serán "no negativas".

Inciso 4

##A partir del sistema de ecuaciones y el correspondiente esquema numérico, explica los resultados del inciso anterior.

$$F_{n+1} = F_n + \Delta t * (2 - S_n) * F_n$$
$$S_{n+1} = S_n + \Delta t * (F_n - 1) * S_n$$

Respecto al inciso 2, se obtuvo un comportamiento cíclico a lo largo del tiempo, donde cada ciclo de F está desplazado con respecto al de S. Esto tiene sentido ya que se ha comenzado con una (muho) mayor cantidad de peces que de tiburones, dadas las condiciones iniciales. De tal forma que se observa cómo la población de peces crece exponencialmente mientras la de tiburones también comienza a crecer, de forma desfasada, llegando a un punto en el cual la población de peces alcanza su máximo (aproximadamente cuando la población de peces es el doble de la mitad de tiburones), después de este punto, la población de peces comienza a decaer mientras que la población de tiburones continúa creciendo hasta llegar a un punto en el que ambas poblaciones tienen el mismo número de elementos, sin embargo la población de peces continúa disminuyendo al mismo tiempo que la población de tiburones aumenta hasta alcanzar su máximo, es justo en este punto donde la población de peces es mínima, y la de tiburones es máxima, por esta razón tiene sentido que la población de tiburones comience a decaer, provocando que la población de tiburones aumente, primero de froma lenta y después a mayor velocidad, repitiéndose así el ciclo nuevamente, y así a lo largo del tiempo.

Respecto al inciso 3, se observa que para las condiciones iniciales el sistema no se ve modificado en el tiempo, analizando un poco más a fondo a partir del esquema explícito y dadas las condiciones iniciales:

$$F_{n+1} = F_n + \Delta t * (2 - S_n) * F_n$$
$$S_{n+1} = S_n + \Delta t * (F_n - 1) * S_n$$
$$F(0) = 1, S(0) = 2$$

Si se hace $n = 0$ para obtener F_1 y S_1, resulta:

$$F_1 = F_0 + \Delta t * (2 - S_0) * F_0$$
$$S_1 = S_0 + \Delta t * (F_0 - 1) * S_0$$

Donde se observa que el segundo término a la derecha de las dos ecuaciones se hace cero debido a las propias condiciones iniciales, dando peso sólo a los primeros términos y provocando así que los valores de $F_1 = F_0 = 1$ y $S_1 = S_0 = 2$. Si se consideran estos resultados para calcular $n = 1$, resultará lo siguiente:

$$F_2 = F_1 + \Delta t * (2 - S_1) * F_1$$
$$S_2 = S_1 + \Delta t * (F_1 - 1) * S_1$$

Donde se observa que nuevamente el segundo término a la derecha de las dos ecuaciones se hace cero y provocando así que $F_2 = F_1 = 1$ y $S_2 = S_1 = 2$. Esto se repetirá para cualquier valor de Δt, ya que se ha visto que el peso del cálculo en el esquema, en este caso particular, recae únicamente en las condiciones iniciales, dado que el término que contiene la variable temporal siempre se volverá cero. Otra forma de pensar en este caso particular es como si el esquema explícito constara de estas dos ecuaciones:

$$F_{n+1} = F_n$$
$$S_{n+1} = S_n$$

Observando las ecuaciones que componen el esquema explícito original, se puede notar que estas condiciones iniciales se pueden obtener al pensar que si se desea que obtener un esquema independiente del tiempo, es equivalente a pedir que:

$$\Delta t * (2 - S_n) * F_n = 0$$
$$\Delta t * (F_n - 1) * S_n = 0$$

Donde $\Delta t \neq 0$

Uno de los casos es el antes mencionado ($S_0 = 2, F_0 = 1$), otra forma de conseguir un resultado similar sería haciendo $F_0 = 0$ y $S_0 = 0$, que sería el caso trivial mostrado en la figura 4, donde desde un inicio no hay poblaciones ni de peces ni de tiburones (por lo cual no se espera que más adelante en el tiempo "aparezcan").

Se puede concluir que en el estado estacionario, los depredadores permanecerán en un nivel 2 por el resto del tiempo y las presas serán únicamente el 50% del total de depredadores; es decir, permanecerán en el nivel 1, ubicándose en las condiciones iniciales en el sistema.

Otra característica de este estado estacionario es que la dinámica de un grupo no depende de la dinámica del otro, y en ningún punto del tiempo se intersectarán o afectarán las poblaciones entre sí.

En este sentido, se puede prever sin la necesidad de linealizar el sistema que existen al menos dos equilibrios estables en el sistema: cuando $F(0) = 1$ y $S(0) = 2$ así como cuando $F(0) = 0$ y $S(0) = 0$, este último caso es inmediato, ya que indica que desde un inicio las poblaciones de

peces y de tiburones están en ceros, por lo que no se espera que repentinamente aparezcan elementos de dichas poblaciones.

De ambos incisos podemos concluir que las condiciones iniciales del problema son determinantes sobre cómo se comportará el sistema a lo largo del tiempo, ya que al menos de inicio no es tan fácil predecir cómo se comportarán ambos sistemas sin realizar los respectivos experimentos adecuadamente.

Sección del código desarrollado para las soluciones que se presentan arriba

In [ ]:
```
# Dependencias
import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Inciso 2 - Programa en python para resolver numéricamente el sistema de ecuaciones que incluye dos esquemas explícitos:

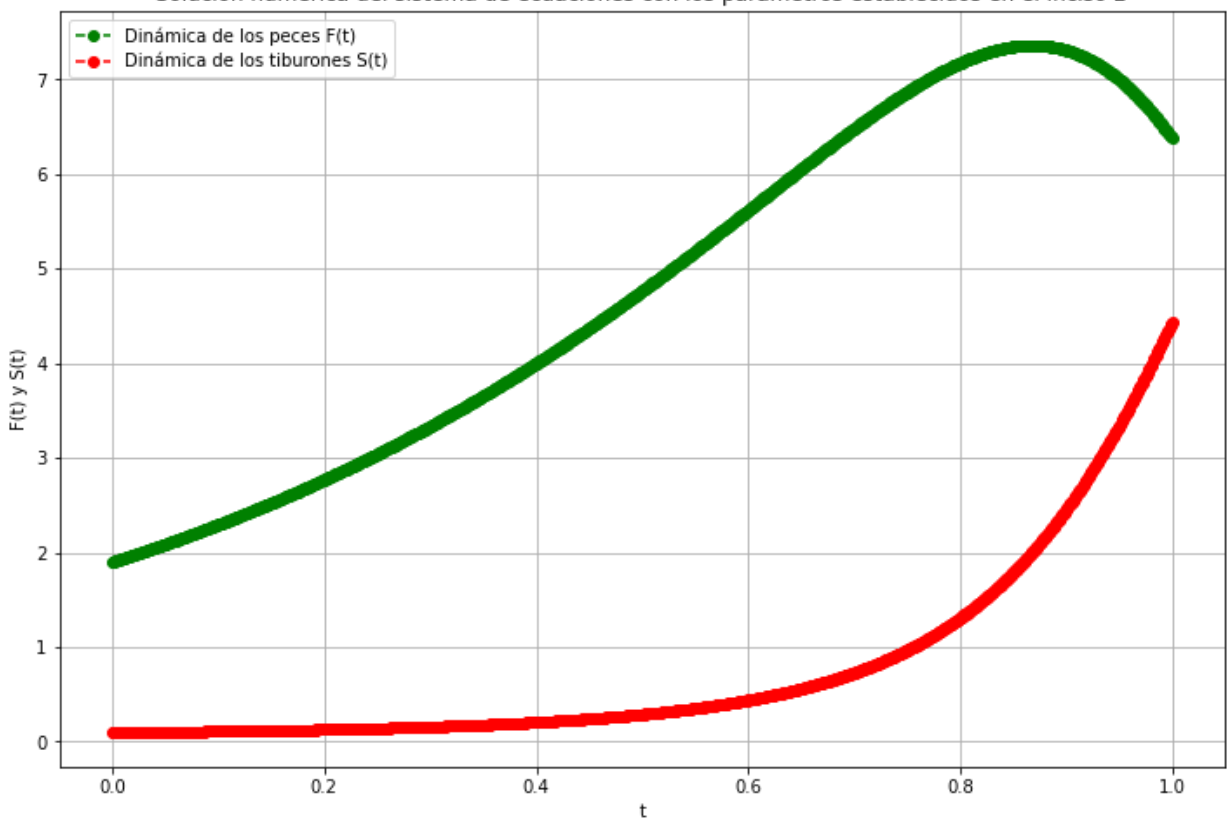
$$F_{n+1} = F_n + \Delta t * (2 - S_n) * F_n$$
$$S_{n+1} = S_n + \Delta t * (F_n - 1) * S_n$$

Para los parámetros: $t = 1$ si $F_0 = 1.9$, $S_0 = 0.1$, $\Delta t = 0.001$.

In [ ]:

```
delta_t = 0.001
T = 1
N = int(T/delta_t)
t = np.linspace(0, T, N+1)   # Creamos el rango de tiempo como una secuenci
F = np.zeros(N+1)  # Creamos los cajones para cada n en F
S = np.zeros(N+1)  # Creamos los cajones para cada n en S

# Condiciones iniciales
F[0] = 1.9
S[0] = 0.1

# Esquemas explícitos de cada función

for n in range(N):
    F[n+1] = F[n] + delta_t*(2-S[n])*F[n] # Dinámica de los peces
    S[n+1] = S[n] + delta_t*(F[n]-1)*S[n] # Dinámica de los predadores

# Gráfica de la dinámica:

plt.figure(figsize = (12, 8))
plt.plot(t, F, 'go--', label='Dinámica de los peces F(t)')
plt.plot(t, S, 'ro--', label='Dinámica de los tiburones S(t)')
plt.title("""Figura 1 \nSolución numérica del sistema de ecuaciones \
con los parámetros establecidos en el Inciso 2""")
plt.xlabel('t')
plt.ylabel('F(t) y S(t)')
plt.grid()
plt.legend()
plt.savefig("Figura1_ss1.jpg", dpi=400, bbox_inches='tight')
plt.show()

# Al inicio, el número de peces aumenta exponencialmente.
# Con un monto muy pequeño inicial de tiburones y un número inicial de
# peces un poco más grande. Lo que genera un crecimiento posterior y
# exponencial de los tiburones y afectando el crecimiento de los peces,
#lo que se esperaría es que el número de peces decrezca a tal grado que
# los tiburones rebiertan su tendencia y reduzcan su crecimiento al grado
# de que el cíclo se repita.
```

Figura 1
Solución numérica del sistema de ecuaciones con los parámetros establecidos en el Inciso 2



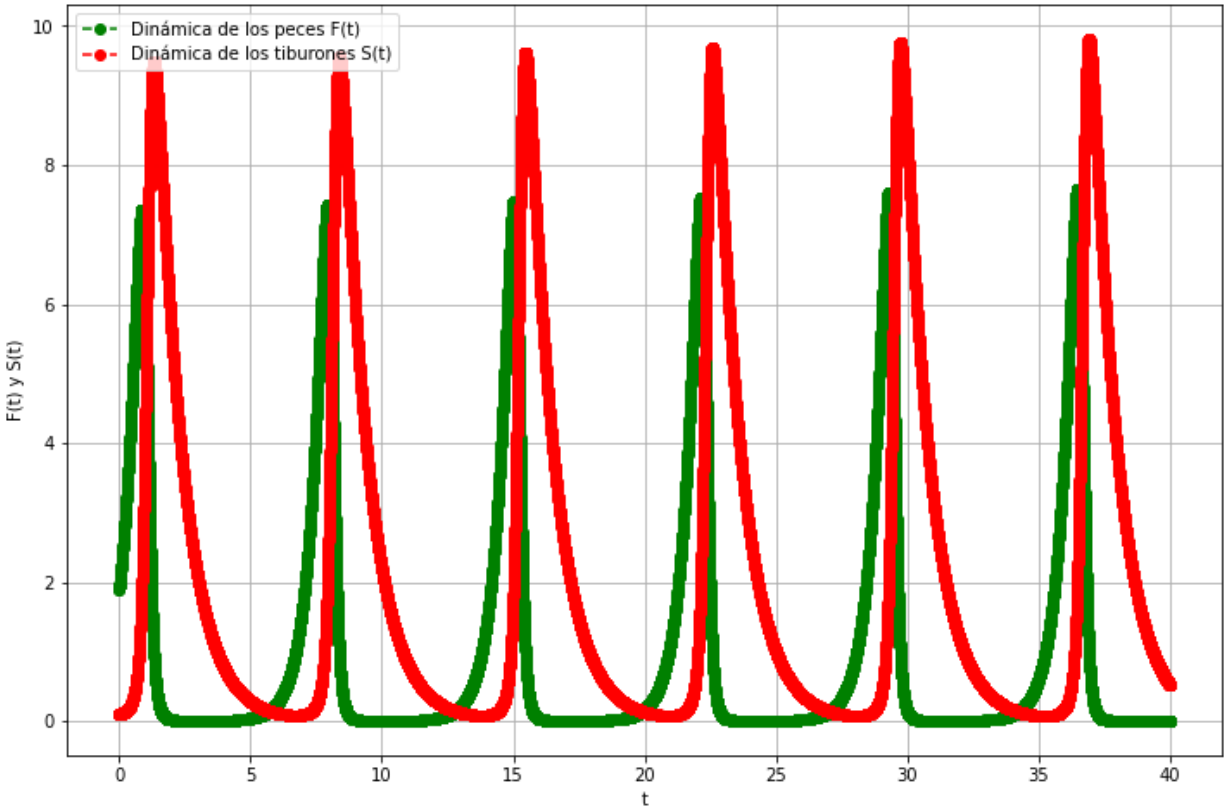
In [ ]:
```
# Los valores en t=1 son
print("El valor de F(t=1) = " + str(F[1000]))
print("El valor de S(t=1) = " + str(S[1000]))
```

```
El valor de F(t=1) = 6.375629791667359
El valor de S(t=1) = 4.441022269812713
```

In [ ]:

```
delta_t = 0.001
T = 40
N = int(T/delta_t)
t = np.linspace(0, T, N+1)   # Creamos el rango de tiempo como una secuenci
F = np.zeros(N+1)  # Creamos los cajones para cada n en F
S = np.zeros(N+1)  # Creamos los cajones para cada n en S

# Condiciones iniciales
F[0] = 1.9
S[0] = 0.1

# Esquemas explícitos de cada función

for n in range(N):
    F[n+1] = F[n] + delta_t*(2-S[n])*F[n] # Dinámica de los peces
    S[n+1] = S[n] + delta_t*(F[n]-1)*S[n] # Dinámica de los predadores

# Gráfica de la dinámica:

plt.figure(figsize = (12, 8))
plt.plot(t, F, 'go--', label='Dinámica de los peces F(t)')
plt.plot(t, S, 'ro--', label='Dinámica de los tiburones S(t)')
plt.title("""Figura 2 \nSolución numérica del sistema de ecuaciones \
con los parámetros establecidos en el Inciso 2 y T=15 \n donde se \
observa que es hasta el T=26 que se llega al equilibrio \
estable S(0)=0 y F(0)=0""")
plt.xlabel('t')
plt.ylabel('F(t) y S(t)')
plt.grid()
plt.legend()
plt.savefig("Figura2_ss1.jpg", dpi=400, bbox_inches='tight')
plt.show()

# Al inicio, el número de peces aumenta exponencialmente.
# Con un monto muy pequeño inicial de tiburones y un número inicial
# de peces un poco más grande. Lo que genera un crecimiento posterior
# y exponencial de los tiburones y afectando el crecimiento de los
# peces, lo que se esperaría es que el número de peces decrezca a tal
# grado que los tiburones rebiertan su tendencia y reduzcan su
# crecimiento al grado de que el cíclo se repita.
```

Figura 2
Solución numérica del sistema de ecuaciones con los parámetros establecidos en el Inciso 2 y T=15
donde se observa que es hasta el T=26 que se llega al equilibrio estable S(0)=0 y F(0)=0



Inciso 3 - Encuentra una aproximación numérica a la solución en $t = 1$ si $F_0 = 1$,$S_0 = 2$, $\Delta t = 0.001$.

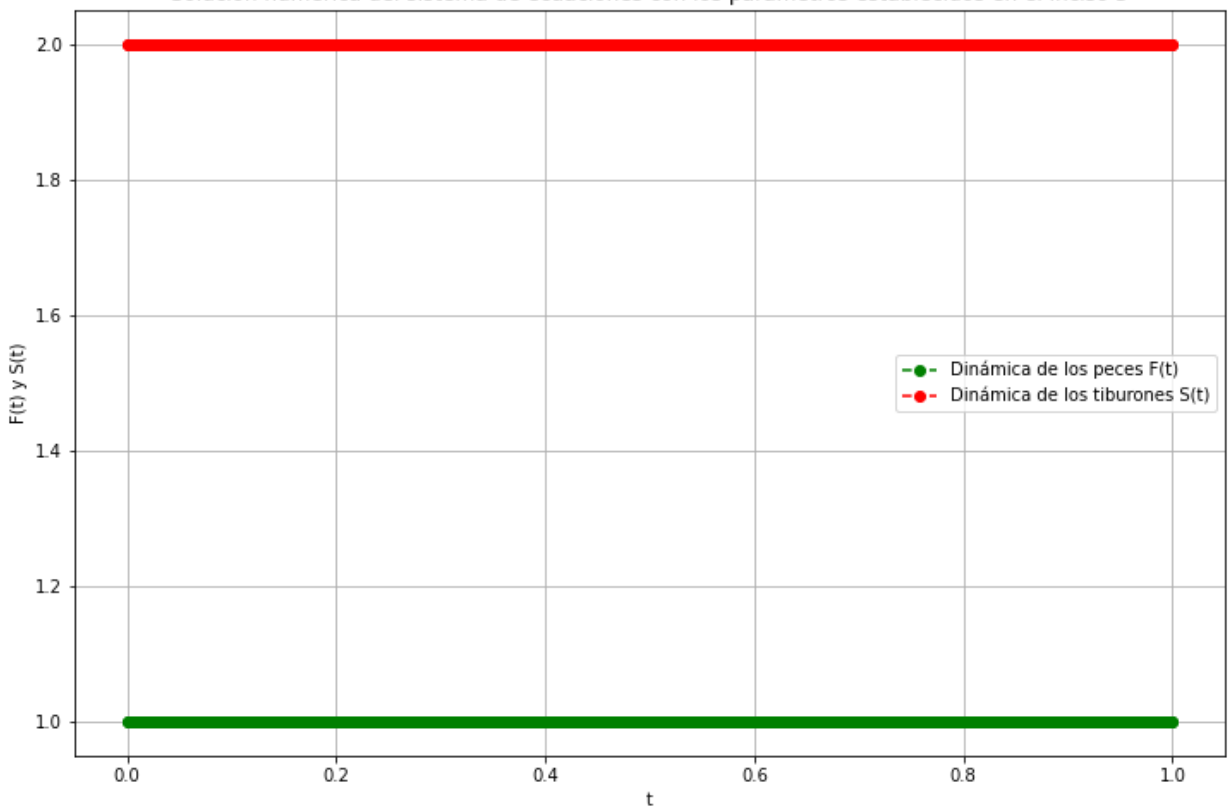
$$F_{n+1} = F_n + \Delta t * (2 - S_n) * F_n$$
$$S_{n+1} = S_n + \Delta t * (F_n - 1) * S_n$$

Aquí se encuentran los dos estados estacionarios sin linealizar el sistema.

In [ ]:

```
delta_t = 0.001
T = 1
N = int(T/delta_t)
t = np.linspace(0, T, N+1)    # Creamos el rango de tiempo como una secuenci
F = np.zeros(N+1)  # Creamos los cajones para cada n en F
S = np.zeros(N+1)  # Creamos los cajones para cada n en S

# Condiciones iniciales
F[0] = 1
S[0] = 2

# Esquemas explícitos de cada función

for n in range(N):
    F[n+1] = F[n] + delta_t*(2-S[n])*F[n] # Dinámica de los peces
    S[n+1] = S[n] + delta_t*(F[n]-1)*S[n] # Dinámica de los predadores

# Gráfica de la dinámica:

plt.figure(figsize = (12, 8))
plt.plot(t, F, 'go--', label='Dinámica de los peces F(t)')
plt.plot(t, S, 'ro--', label='Dinámica de los tiburones S(t)')
plt.title('Figura 3\nSolución numérica del sistema de ecuaciones \
con los parámetros establecidos en el Inciso 3')
plt.xlabel('t')
plt.ylabel('F(t) y S(t)')
plt.grid()
plt.legend()
plt.savefig("Figura3_ss1.jpg", dpi=400, bbox_inches='tight')
plt.show()
```

Figura 3
Solución numérica del sistema de ecuaciones con los parámetros establecidos en el Inciso 3



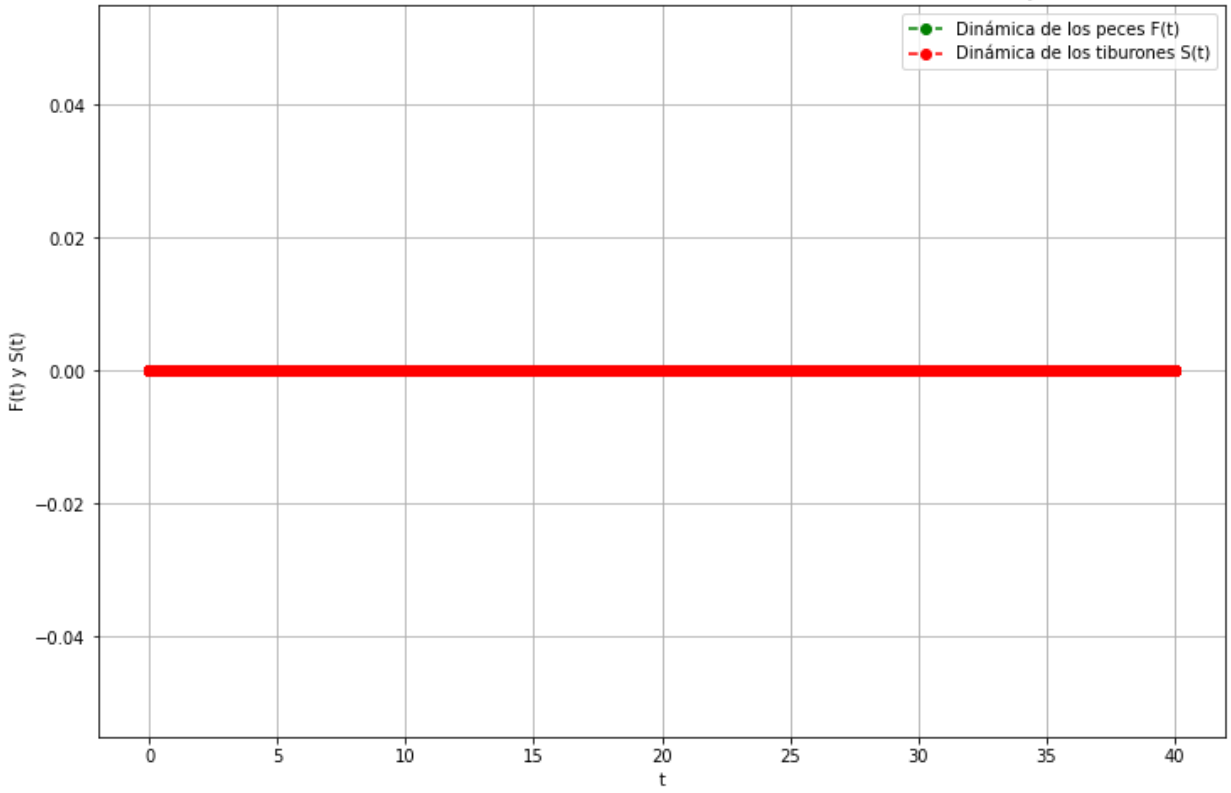
In [ ]:
```
# Los valores en t=1 son
print("El valor de F(t=1) = " + str(F[1000]))
print("El valor de S(t=1) = " + str(S[1000]))
```

```
El valor de F(t=1) = 1.0
El valor de S(t=1) = 2.0
```

In [ ]:

```
delta_t = 0.001
T = 40
N = int(T/delta_t)
t = np.linspace(0, T, N+1)   # Creamos el rango de tiempo como una secuenci
F = np.zeros(N+1)  # Creamos los cajones para cada n en F
S = np.zeros(N+1)  # Creamos los cajones para cada n en S


# Condiciones iniciales
F[0] = 0
S[0] = 0

# Esquemas explícitos de cada función

for n in range(N):
    F[n+1] = F[n] + delta_t*(2-S[n])*F[n] # Dinámica de los peces
    S[n+1] = S[n] + delta_t*(F[n]-1)*S[n] # Dinámica de los predadores

# Gráfica de la dinámica:

plt.figure(figsize = (12, 8))
plt.plot(t, F, 'go--', label='Dinámica de los peces F(t)')
plt.plot(t, S, 'ro--', label='Dinámica de los tiburones S(t)')
plt.title('Figura 4\nSolución numérica del sistema de ecuaciones \
con el estado estacionario F(0)=0 y S(0)=0')
plt.xlabel('t')
plt.ylabel('F(t) y S(t)')
plt.grid()
plt.legend()
plt.savefig("Figura4_ss1.jpg", dpi=400, bbox_inches='tight')
plt.show()
```

Figura 4
Solución numérica del sistema de ecuaciones con el estado estacionario F(0)=0 y S(0)=0



#Bibliografía y referencias electrónicas.

- Aslak Tveiko, Hans Petter Langtangen, Bjorn Frederik Nielsen and Xing Cai (2010), ***Elements of Scientific Computing***, Capítulo 3.
- Svein Linge and Hans Petter Langtangen (2020), ***Programming for Computations-Python***, páginas 226 a 239.