

Oscilaciones en la actividad neuronal de un gato durante el sueño

Denisa Alexandru, Sonsoles Hernández Piñel, Jimena Martín Reina

Mayo 2022

Índice

1. Resumen	2
2. Introducción¹	2
2.1. Funciones del sueño	2
2.2. Fases del sueño	3
2.3. Influencia del sueño en la memoria	4
3. Desarrollo	5
3.1. Descripción del modelo matemático	5
3.2. Representación del modelo con Python	6
3.3. Resultados del modelo y su interpretación biológica	8
4. Conclusiones	9
4.1. Aplicaciones e investigaciones futuras	9
4.2. Conclusión final	11
5. Bibliografía	12
6. Anexos	12

1. Resumen

El sueño es un fenómeno que siempre ha provocado profunda fascinación en el ser humano. Debido a que sólo recientemente se han comenzado a entender sus mecanismos fisiológicos y su neuroanatomía, ha estado constantemente envuelto en misterio, controversias y especulaciones. El sueño en la actualidad es considerado como un proceso fisiológico de vital importancia para la salud integral de los seres humanos.¹

Una de las maneras para comprender su funcionamiento ha sido mediante el uso de modelos matemáticos. El modelo más relevante, que se va a estudiar en este artículo, es el modelo de Lotka-Volterra de depredador-presa adaptado a la actividad de dos tipos de células neuronales: neuronas del campo tegmental gigantocelular (células FTG) y neuronas del núcleo del locus cerúleo y núcleo subcerúleo (células LC).²

2. Introducción¹

2.1. Funciones del sueño

La necesidad y la función del sueño son algunas de las áreas menos entendidas en la investigación del sueño. Se conocen algunas funciones del sueño y algunas otras se han propuesto pero no están completamente sustentadas o comprendidas. Inicialmente, se pensó que el sueño era simplemente un mecanismo para que el cuerpo pueda tomar un descanso y reducir el desgaste. Observaciones posteriores de las bajas tasas metabólicas en el cerebro durante el sueño parecían indicar algunas funciones metabólicas del sueño.

Una herramienta tecnológica que ha sido de vital importancia para el estudio de la fisiología del sueño es el electroencefalograma (EEG). De forma muy simplificada, el EEG es la representación gráfica y digital de las oscilaciones que muestra la actividad eléctrica del cerebro, al ser registrada mediante electrodos colocados encima de la piel cabelluda en distintas regiones de la cabeza.

Con el desarrollo del EEG, se descubrió que el cerebro tiene actividad interna casi continua durante el sueño, lo que conduce a la idea de que la función podría ser la reorganización o la especificación de los circuitos neuronales o fortalecimiento de las conexiones. Estas hipótesis todavía están siendo estudiadas. Otras funciones del sueño que han sido propuestas son:

mantenimiento del equilibrio hormonal, regulación de la temperatura y mantenimiento del ritmo cardíaco. Pero las principales funciones, ya estudiadas, del sueño, son:

- **Función endocrina** La secreción de muchas hormonas se ve afectada por los ciclos de sueño-despertar. Debido a que las hormonas juegan un papel importante en el balance de energía y el metabolismo, y que el sueño desempeña un rol crítico en la sincronización y la extensión de su secreción, el sueño tiene un efecto importante sobre el metabolismo. Como ejemplo, algunas hormonas circadianas, cuya secreción son la melatonina, el cortisol, la hormona estimulante de la tiroides (TSH), la hormona del crecimiento (GH) o la prolactina.
- **Procesamiento de la memoria** Ver apartado 2.3
- **Cambios de humor y de comportamiento** La falta de sueño tiene un efecto perjudicial en las tareas cognitivas, especialmente en aquellas que incluyen funciones divergentes o multitareas. También tiene un efecto en el humor y las emociones, ha habido múltiples reportes de aumento de tendencias de ira, miedo y depresión si hay una deuda de sueño. Sin embargo, algunas funciones cognitivas superiores parecen permanecer sin ser afectadas, no obstante son lentas. Muchos de estos efectos varían de persona a persona. Algunos individuos sufren altos grados de limitación cognitiva por la falta de sueño, otros tienen mínimos efectos. Los mecanismos exactos que causan esto son desconocidos y las rutas neurales exactas así como los mecanismos celulares de la deuda de sueño, siguen siendo investigados.

2.2. Fases del sueño

El sueño suele dividirse en dos grandes fases que, de forma normal, ocurren siempre en la misma sucesión: todo episodio de sueño comienza con el llamado sueño sin movimientos oculares rápidos (No MOR) o NREM (del inglés “Non Rapid Eye Movement”), que tiene varias fases, y después pasa al sueño con movimientos oculares rápidos (MOR) o REM (del inglés “Rapid Eye Movement”).

- **Fase NREM.**

- **Fase 1: somnolencia o inicio del sueño ligero.** En ella es muy fácil despertarse. La actividad muscular disminuye paulatinamente. Pueden observarse algunas breves sacudidas musculares súbitas que a veces coinciden con una sensación de caída (mio-clonías hípnicas). En el EEG se observa actividad de frecuencias mezcladas pero de bajo voltaje y algunas ondas agudas (ondas agudas del vértex).
- **Fase 2.** En el EEG se caracteriza por que aparecen patrones específicos de actividad cerebral llamados husos de sueño y complejos K. La temperatura, la frecuencia cardíaca y respiratoria comienzan a disminuir paulatinamente.
- **Fases 3 y 4: sueño de ondas lentas.** Es la fase de sueño NREM más profunda. En el EEG se observa actividad de frecuencia muy lenta ($< 2Hz$).

■ **Fase REM.**

Se caracteriza por la presencia de movimientos oculares rápidos. El tono de todos los músculos disminuye (con excepción de los músculos respiratorios y los esfínteres vesical y anal). La frecuencia cardíaca y respiratoria se vuelve irregular e incluso puede incrementarse y existe erección espontánea del pene o del clítoris. Durante esta fase, se producen la mayoría de las ensoñaciones (lo que conocemos coloquialmente como sueños), y la mayoría de los pacientes que se despiertan durante esta fase suelen recordar vívidamente el contenido de sus ensoñaciones.

Un adulto joven pasa aproximadamente entre 70-100 min en el sueño NREM para después entrar al sueño REM, el cual puede durar entre 5-30 min, y este ciclo se repite cada hora y media durante toda la noche de sueño. Por lo tanto, a lo largo de la noche pueden presentarse normalmente entre 4 y 6 ciclos de sueño REM.

2.3. Influencia del sueño en la memoria

Desde principios del siglo XX algunos investigadores ya habían demostrado que la retención de la memoria era mucho mejor después de una noche de sueño que después de un intervalo de descanso similar manteniéndose alerta. Sin embargo, en esta época se pensó que el efecto positivo observado era en

realidad inespecífico. En la actualidad diversos estudios tanto experimentales como clínicos han demostrado que el sueño tiene efectos positivos sobre distintos tipos de memoria.

De todos los sistemas de memoria antes expuestos, la evidencia más consistente respecto al efecto positivo del sueño se ha observado en 2 tipos de memoria: la memoria declarativa (memoria que es fácilmente expresada verbalmente: información de hechos y eventos), y la memoria procedimental (memoria acerca de habilidades y destrezas motoras). Hasta el momento prácticamente no existe ninguna evidencia que sugiera lo contrario (es decir, que el sueño favorezca el olvido o la alteración de la memoria previa).

3. Desarrollo

3.1. Descripción del modelo matemático

La existencia de un grupo celular en la región del núcleo locus coeruleus (LC) del gato con curvas de actividad de descarga opuestas a las de las células del campo tegmental gigantocelular (FTG) ha sido ya documentada anteriormente por Hobson et al.², que propusieron que la interacción recíproca entre las poblaciones neuronales excitatorias e inhibitorias puede determinar la alternancia de los estados del ciclo del sueño. Con el modelo estructural básico descrito por Hobson et al.², se procedió a desarrollar un modelo cuantitativo paralelo basado en la hipótesis de la interacción recíproca y consideramos aspectos de las curvas de actividad de descarga de las unidades FTG y LC.

La forma matemática de los términos que describen la influencia de cada población sobre sí misma fue sugerida por la evidencia de que la tasa de cambio de los niveles de actividad en la población FTG era proporcional al nivel actual de actividad, y se propuso que lo mismo era cierto para la población LC, pero con un signo negativo porque la retroalimentación recurrente era inhibitoria. La naturaleza altamente no sinusoidal de la actividad del FTG sugería que era de esperar una interacción no lineal entre el FTG y el LC. Se modeló este efecto mediante la forma más simple de no linealidad, el producto de las actividades en las dos poblaciones. Por lo tanto, sea $x(t)$ el nivel de actividad de descarga en las células FTG; $y(t)$ el nivel de actividad de descarga en las células LC; y a , b , c y d constantes positivas. Estos términos están relacionados por el sistema de ecuaciones:

$$\begin{cases} X'(t) &= aX - bXY \\ Y'(t) &= -cY + dXY \end{cases} \quad (1)$$

Este sistema de ecuaciones es el de **Lotka y Volterra**, originalmente propuesto como un modelo de interacción presa-predador. Este sistema debe su nombre a dos ecólogos matemáticos que estudiaron la interacción entre las poblaciones de presas y depredadores en ecosistemas aislados. Por tanto, era un punto de partida lógico para describir la interacción entre poblaciones neuronales inhibitoras ("depredador") y excitadoras ("presa"). En este modelo, las células FTG (excitatorias) son análogas a la población de presas, y las células LC (inhibidoras) son análogas a la población de depredadores.

3.2. Representación del modelo con Python

Se reconstruyó el modelo descrito con Python, utilizando el entorno informático interactivo basado en web Jupyter Notebook.

Se definió una función genérica F del sistema, así como un vector P compuesto por dos funciones desconocidas $x(t)$ y $y(t)$. A estas dos funciones, se le asignaron las ecuaciones descritas en el [apartado 3.1](#). De la misma forma, se le asignaron valores a los parámetros a , b , c y d , tal que $a = b = 2c = 2d$ ³. Se realizaron dos modelizaciones: una en la que $a = 0,3029$; y otra en la que $a = 0,5490$.² Se obtuvo la resolución numérica del sistema y se representaron, entre otras, las siguientes gráficas:

- Actividad neuronal como funciones del tiempo (Figura 1). Esta figura muestra como evoluciona la actividad de ambas poblaciones en función del tiempo.
- Puntos críticos en el plano de fases (Figura 2). Esta figura muestra como interaccionan los dos tipos celulares.

A continuación, se buscaron los puntos críticos del sistema de ecuaciones

$$\begin{cases} X'(t) &= F(X, Y) \\ Y'(t) &= G(X, Y) \end{cases} \quad (2)$$

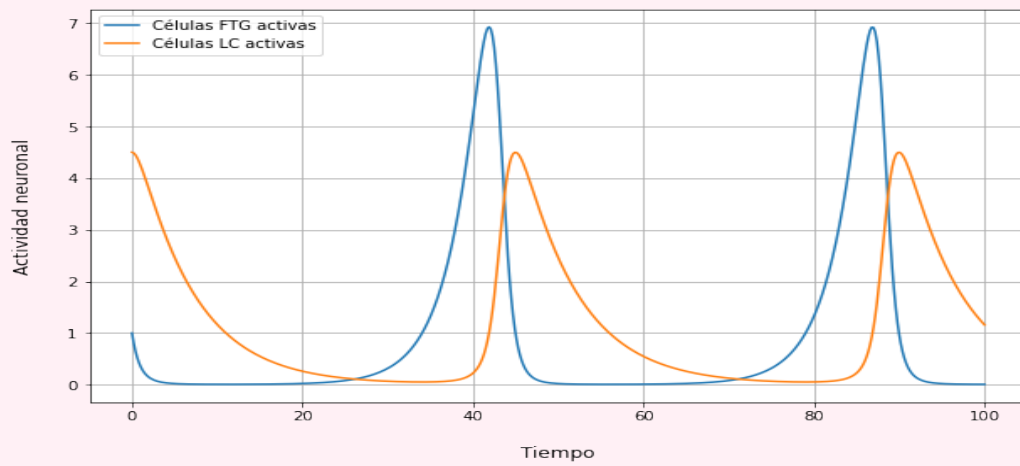


Figura 1: Actividad neuronal en función del tiempo.

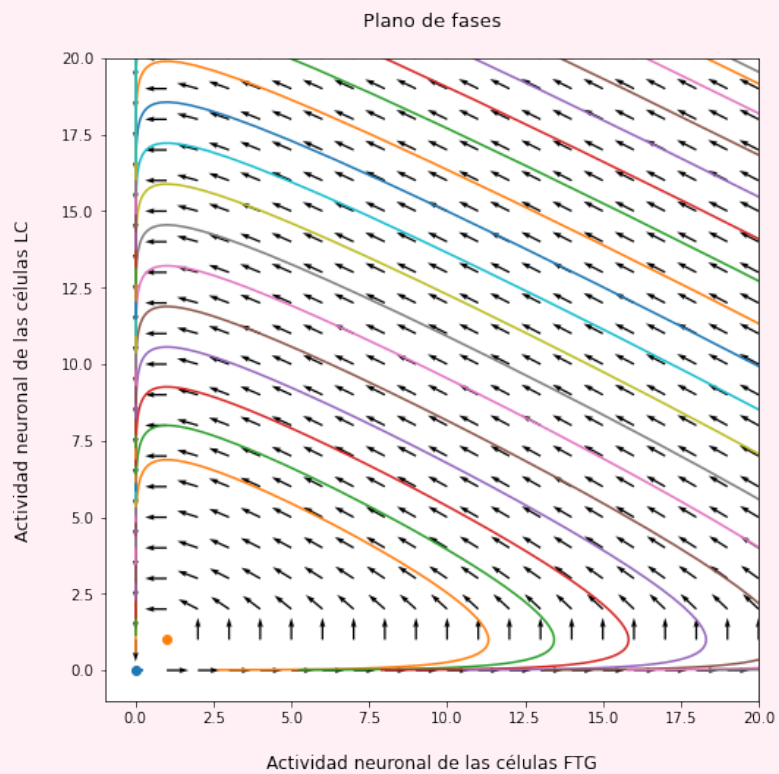


Figura 2: Plano de fases con puntos críticos.

Sean los puntos que anulan ambas derivadas. Es decir, las soluciones del sistema de ecuaciones algebraicas

$$\begin{cases} F(X, Y) = 0 \\ G(X, Y) = 0 \end{cases} \quad (3)$$

Se encontró que los puntos críticos eran $(0,0)$ y $(1,1)$, a partir de lo cual se pudo estudiar su estabilidad. Para ello, se construyó la matriz jacobiana del sistema de ecuaciones (2) y se estudió la traza, el determinante y los autovalores de esta, evaluada en ambos puntos críticos.

En el punto crítico $(0,0)$ se obtuvo la matriz,

$$\begin{bmatrix} a & 0 \\ 0 & -c \end{bmatrix} \quad (4)$$

siendo su traza igual a $[a - c]$ y sus autovalores $[a, -c]$. Como a y c son dos valores reales positivos, se concluyó que este punto era un punto hiperbólico, sea inestable.

En el punto crítico $(1,1)$ se obtuvo la matriz,

$$\begin{bmatrix} 0 & -\frac{bc}{d} \\ \frac{ad}{b} & 0 \end{bmatrix} \quad (5)$$

siendo su traza igual a $[0]$ y sus autovalores $[-i\sqrt{a}\sqrt{c}, i\sqrt{a}\sqrt{c}]$. Al ser la traza nula y los autovalores complejos, se concluyó que este punto era un centro, sea estable.

3.3. Resultados del modelo y su interpretación biológica

Como se puede observar en la figura 2, cualquier oscilación estable de las poblaciones FTG y LC se representa gráficamente como una única órbita cerrada que se recorre repetidamente a lo largo del tiempo en sentido contrario a las agujas del reloj. En particular, las soluciones simples de Lotka-Volterra aparecen como familias de órbitas ovales simples (trayectorias) en el plano de fase. La ilustración esquemática de varias de estas trayectorias se corresponden a soluciones separadas generadas al elegir diferentes condiciones iniciales

pero que tienen los mismos valores para a , b , c y d . P3 es la más similar a la solución utilizada en el modelo de 1975.²

La actividad neuronal de los dos tipos celulares muestran un decrecimiento en el primer tercio del ciclo, un largo período de crecimiento lento de la actividad, y una rápida aceleración a medida que se acerca el momento de la desincronización del sueño.

Como se puede observar en la Figura 1, la actividad de las células FTG es máxima cuando la actividad de las células LC es mínima. Al comenzar a decrecer la actividad de las células FTG es cuando empieza a aumentar la actividad de las células LC. Esto se debe que la actividad de las LC disminuye continuamente desde el sueño sincronizado a un sueño desincronizado, y después muestra un rápido crecimiento en la última porción del sueño desincronizado. El aumento en la descarga ocurre justo al final del sueño desincronizado. Suprimir la actividad de las LC o sus efectos postsinápticos producirá un aumento en la actividad de las FTG y en consecuencia, más sueño desincronizado.

Por lo tanto, el modelo sería capaz de predecir en qué momento comenzaría el sueño sincronizado (es decir, cuando aumentan las células LC su actividad) y cuándo se produciría el paso a sueño desincronizado (debido a un aumento en la actividad neuronal de las células FTG).

4. Conclusiones

4.1. Aplicaciones e investigaciones futuras

Una de las investigaciones que más han aportado en el tema es la realizada por los investigadores Robert W. McCarley y Steve G. Massaquoi.³ En ella, desarrollan un modelo matemático de ciclo límite del sistema oscilador del sueño de movimientos oculares rápidos (REM) a partir de un modelo estructural de interacción de poblaciones de neuronas REM-on y REM-off. Las marcadas diferencias en la latencia, la amplitud y la duración del primer período de sueño REM observadas con la variación circadiana y la patología depresiva se modelan comenzando la oscilación REM en diferentes puntos iniciales relativos a la posición final en el ciclo límite.

1. El comienzo desde un punto gráficamente interior al ciclo límite produce

un primer periodo REM de larga latencia, corta duración y menos intenso.

2. Empezar desde un punto gráficamente exterior al ciclo límite produce un primer período REM de corta latencia, larga duración y más intenso.

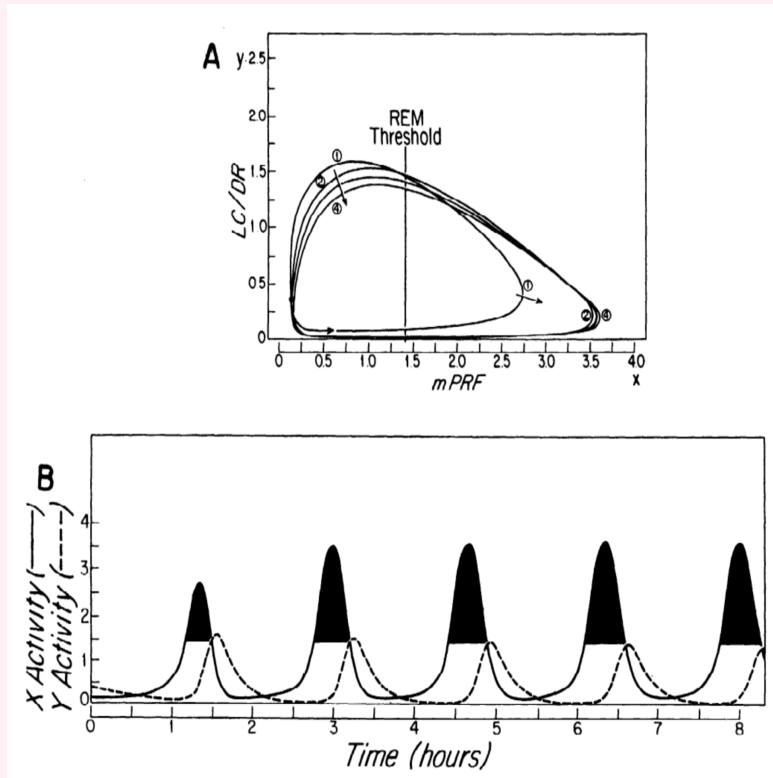


Figura 3: Simulación del curso nocturno del sueño REM en seres humanos normales. A: plano de fase; B: representación en función del tiempo de estos datos.

En el modelo, el determinante de si la oscilación comienza fuera o dentro del ciclo límite es el curso temporal de la caída de la actividad de descarga de la población REM-off al inicio del sueño. Cuando este curso temporal se hace depender de la fase circadiana, el modelo produce una coincidencia muy cercana a los grandes cambios observados empíricamente entre el primer y el segundo periodo REM en duración (a menudo un cambio del 50 %) e

intensidad y también imita de cerca los cambios observados empíricamente en la latencia REM cuando el sueño humano comienza en diferentes fases circadianas. Aunque esta variación en la entrada del ciclo límite explica los principales cambios en el sueño REM a lo largo de la noche, el modelo también postula una variación circadiana continua pero pequeña (del orden de $\pm 5\%$ de cambio en los parámetros REM) que actúa a lo largo del curso de una noche de sueño.

En B, las porciones oscuras del gráfico X (formación reticular pontina medial (mPRF)) muestran las porciones de la noche con sueño REM, y la altura de estos picos indica la intensidad del episodio de sueño REM. Obsérvese que el primer episodio REM es de corta duración y de menor intensidad, y que las variaciones posteriores son leves. La actividad de la población Y (inhibidora del REM) se indica con una línea de puntos. En la representación del plano de fase, A, cada punto del gráfico representa los valores X-Y en un momento determinado. El punto representa el punto de partida, y la curva interior con la flecha muestra los valores del primer episodio REM, siendo esta curva interior a los valores del ciclo límite obtenidos en los ciclos REM subsiguientes, etiquetados como 2, 4 en orden de su ocurrencia (por simplicidad, el ciclo de sueño 5 no ha sido graficado en el plano de fase). Las pequeñas variaciones en los ciclos de sueño 2-4 se deben a la modulación circadiana.

Dado que el modelo se deriva de datos fisiológicos reales, en lugar de ser una construcción puramente ad hoc o fenomenológica, ofrece la posibilidad de probar directamente sus postulados mediante estudios neurobiológicos en animales, mediante manipulaciones del ciclo del sueño relacionadas con la fase circadiana y mediante perturbaciones del sistema en humanos por el uso de fármacos.

4.2. Conclusión final

Este artículo es un ejemplo más de cómo, incluso en un tema tan misterioso y del que aún nos queda mucho por descubrir como es el sueño, las matemáticas pueden describir modelos que se adecúan bastante bien al comportamiento que ocurre en el sistema biológico.

5. Bibliografía

1. Carrillo-Mora, Paul, Ramírez-Peris, Jimena, Magaña-Vázquez, Katia. (2013). Neurobiología del sueño y su importancia: antología para el estudiante universitario. Revista de la Facultad de Medicina (México), 56(4), 5-15. Recuperado en 19 de mayo de 2022
2. R.W. McCarley and J.A. Hobson (1975) Neuronal excitability modulation over the sleep cycle: a structural and mathematical model. <https://doi.org/10.1126/science.1135627>
3. McCarley RW, Massaquoi SG. A limit cycle mathematical model of the REM sleep oscillator system. Am J Physiol. 1986 Dec;251(6 Pt 2):R1011-29. doi: 10.1152/ajpregu.1986.251.6.R1011. PMID: 3789188.
4. Munro, E., Khodai, T., Sakata, S., Toyoizumi, T. (2015). Two different mechanisms alternate during cortical synchronized states. BMC Neuroscience, 16(Suppl 1), P264. <https://doi.org/10.1186/1471-2202-16-S1-P264>

6. Anexos

Modelo de Lotka y Volterra para el sistema de sueño REM

Este modelo está formado por dos edo's no lineales que constituyen un modelo simplificado de la interacción de dos poblaciones de neuronas. involucradas en el sistema de sueño REM, una de las cuales es un predador y la otra una presa. Las variables x e y representan respectivamente, el número de individuos de la población de neuronas FTG activas y de la población de neuronas LC activas, respectivamente, que depende del tiempo t . Estas son las ecuaciones:

$$\begin{aligned}\frac{dx}{dt} &= ax - cxy \\ \frac{dy}{dt} &= -dy + xy\end{aligned}$$

Aquí las constantes a , b , c y d son parámetros reales positivos.

1. Solución numérica del PVI (para valores concretos de los parámetros)
2. Gráficas de la actividad neuronal como funciones del tiempo y de órbitas (para valores concretos de los parámetros)
3. Representar campo de direcciones y órbitas: plano de fases (para valores concretos de los parámetros)
4. Representar puntos de equilibrio en el plano de fases (para valores concretos de los parámetros)
5. Estabilidad de los punto de equilibrio del sistema: estudio cualitativo del caso general (valores arbitrarios de los parámetros del sistema)

Cargamos los módulos necesarios: numérico, gráficos, integración numérica y lenguaje matemático

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import sympy as sp
sp.init_printing()
```

La función que define este sistema es

$$F(x,y) = (ax - bxy, -cy + dxy)$$

1. Solución numérica del PVI

```
In [2]: # Introducimos los valores de las constantes
a,b,c,d = 0.3029,0.3029,0.15145,0.15145

# La función 'F' del sistema y el vector 'P' que se compone de las funciones desconocidas 'x(t)', 'y(t)'
def F(P, t):
    return [a*P[0] - b*P[0]*P[1], -c*P[1] + d*P[0]*P[1]]

# Damos 500 valores a la variable independiente 't' entre 0 y 100
tiempo = np.linspace(0, 100, 500)

# Condición inicial
P0 = [1.0, 4.5]

# Obtenemos la resolución numérica del sistema en forma de matriz 'Ps' con una columna para los valores de cada población.
Ps = scint.odeint(F, P0, tiempo)

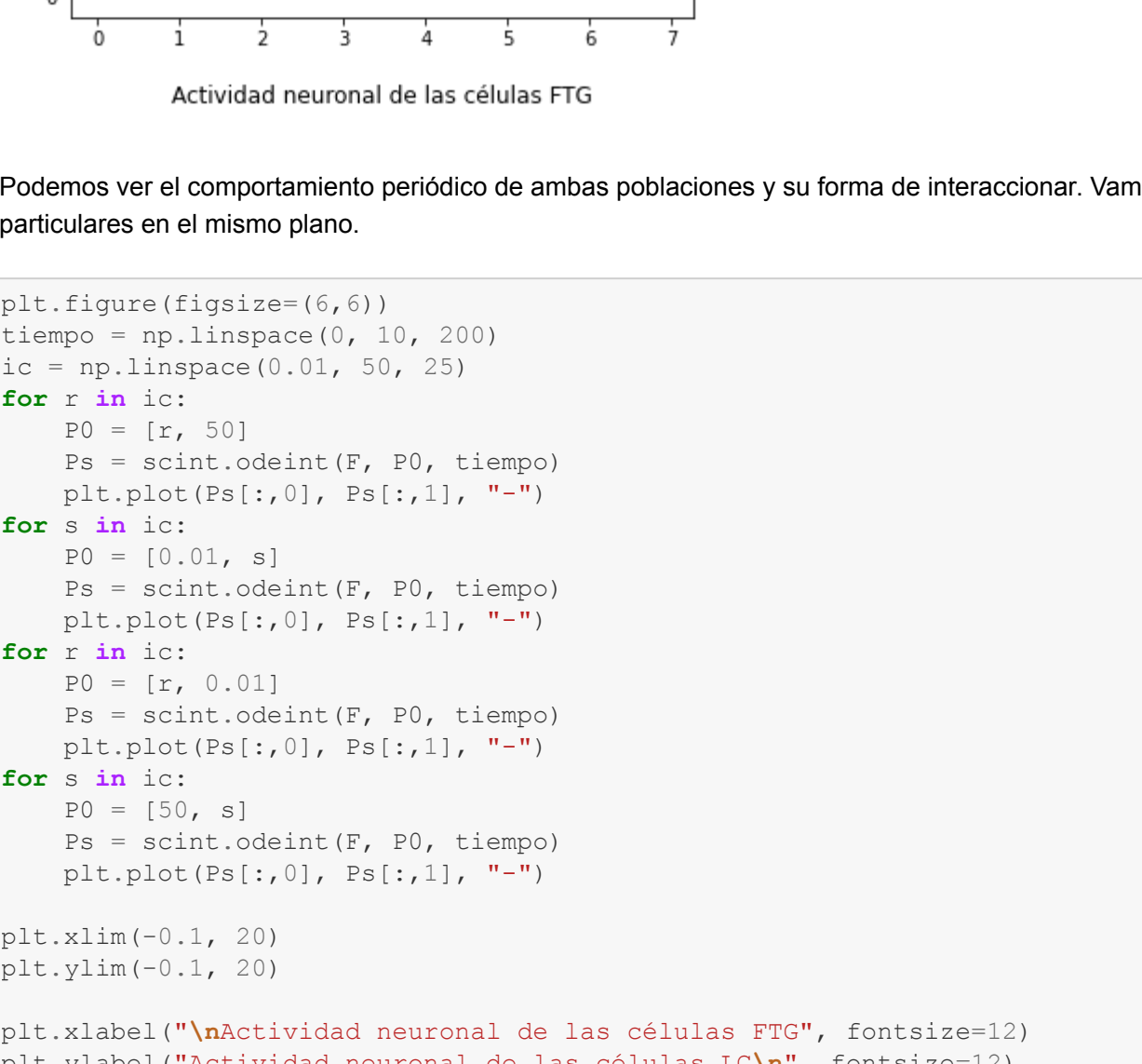
# Valores obtenidos de la población de células FTG en la primera columna
FTG = Ps[:,0]

# Valores obtenidos de la población de células LC en la segunda
LC = Ps[:,1]
```

2. Gráficas de poblaciones como funciones del tiempo y de órbitas

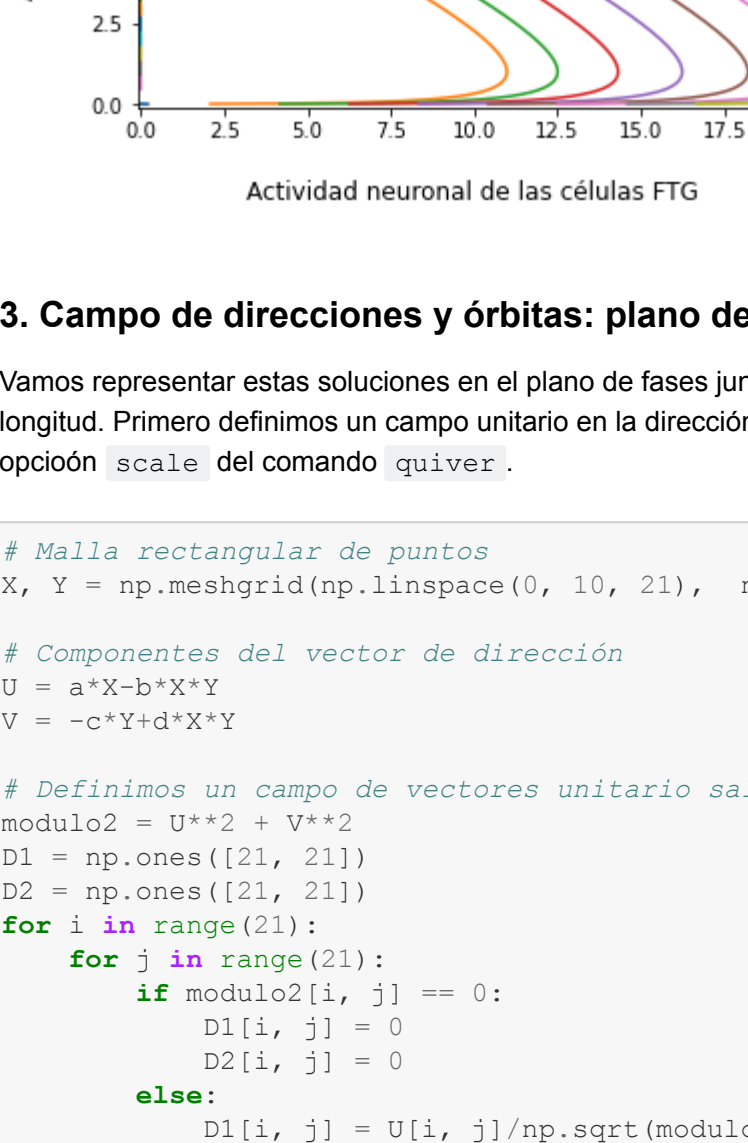
Dibujamos ambas gráficas de la evolución de cada población respecto del tiempo en los instantes indicados

```
In [3]: plt.figure(figsize = (10,6))
plt.plot(tiempo, FTG, label="Células FTG activas")
plt.plot(tiempo, LC, label="Células LC activas")
plt.xlabel("tiempo", fontsize=12)
plt.ylabel("Actividad neuronal", fontsize=12)
plt.grid(True)
plt.legend()
plt.show()
```



Dibujamos la gráfica de la trayectoria en el plano de fases

```
In [4]: plt.figure(figsize=(6,6))
plt.plot(FTG, LC, "r-")
plt.xlabel("Actividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.show()
```



Podemos ver el comportamiento periódico de ambas poblaciones y su forma de interactuar. Vamos a dibujar varias soluciones particulares en el mismo plano.

```
In [5]: plt.figure(figsize=(6,6))
tiempo = np.linspace(0, 10, 200)
lc = np.linspace(0.01, 50, 25)

for r in lc:
    P0 = [r, 50]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

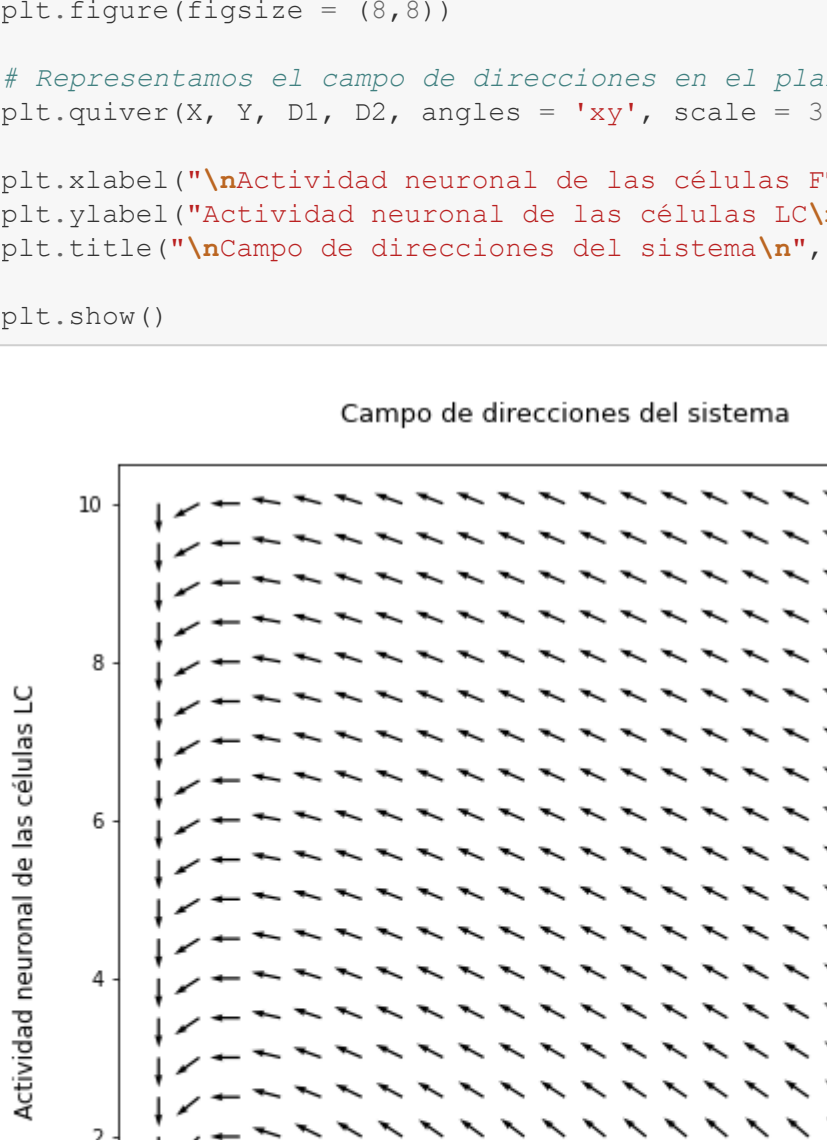
for s in lc:
    P0 = [0.01, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

for r in lc:
    P0 = [r, 0.01]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

for s in lc:
    P0 = [50, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

plt.xlim(-0.1, 20)
plt.ylim(-0.1, 20)

plt.xlabel("Actividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("Órbitas del sistema", fontsize=13)
plt.show()
```



3. Campo de direcciones y órbitas: plano de fases

Vamos representar estas soluciones en el plano de fases junto con el campo de direcciones de forma que los vectores tengan la misma longitud. Primero definimos un campo unitario en la dirección del campo asociado al sistema. Después controlamos la longitud con la opción `scale` del comando `quiver`.

```
In [6]: # Malla rectangular de puntos
X, Y = np.meshgrid(np.linspace(0, 10, 21), np.linspace(0, 10, 21))

# Componentes del vector de dirección
U = a*X-b*X*Y
V = -c*Y+d*X*Y

# Definimos un campo de vectores unitario salvo cuando se anula
modulo2 = U**2 + V**2
D1 = np.ones([21, 21])
D2 = np.ones([21, 21])
for i in range(21):
    for j in range(21):
        if modulo2[i, j] == 0:
            D1[i, j] = 0
            D2[i, j] = 0
        else:
            D1[i, j] = U[i, j]/np.sqrt(modulo2[i, j])
            D2[i, j] = V[i, j]/np.sqrt(modulo2[i, j])

# Figura cuadrada
plt.figure(figsize = (8,8))

# Representamos el campo de direcciones en el plano
plt.quiver(X, Y, D1, D2, angles = 'xy', scale = 30, headwidth = 3)

plt.xlabel("Actividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("Campo de direcciones del sistema", fontsize=13)
plt.show()
```



Campo de direcciones con una curva solución

```
In [7]: fig = plt.figure(figsize = (12,6))

ejes1 = fig.add_subplot(1,2,1)
ejes1.plot(FTG, LC, "r-")
ejes1.set_title("Órbita")
ejes1.set_xlabel("Actividad neuronal de las células FTG", fontsize=12)
ejes1.set_ylabel("Actividad neuronal de las células LC", fontsize=12)
ejes1.set_xlim(-0.1, 8)
ejes1.set_ylim(-0.1, 6)
plt.grid(True)

ejes2 = fig.add_subplot(1,2,2)

X, Y = np.meshgrid(np.linspace(0, 10, 21), np.linspace(0, 10, 21))
U = a*X-b*X*Y
V = -c*Y+d*X*Y
modulo2 = U**2 + V**2
D1 = np.ones([21, 21])
D2 = np.ones([21, 21])
for i in range(21):
    for j in range(21):
        if modulo2[i, j] == 0:
            D1[i, j] = 0
            D2[i, j] = 0
        else:
            D1[i, j] = U[i, j]/np.sqrt(modulo2[i, j])
            D2[i, j] = V[i, j]/np.sqrt(modulo2[i, j])

ejes2.quiver(X, Y, D1, D2, angles = 'xy', scale = 20, headwidth = 3)
ejes2.plot(FTG, LC, "r-")
ejes2.set_title("Órbita y campo de direcciones")
ejes2.set_xlabel("Actividad neuronal de las células FTG", fontsize=12)
ejes2.set_ylabel("Actividad neuronal de las células LC", fontsize=12)
ejes2.set_xlim(-0.1, 8)
ejes2.set_ylim(-0.1, 6)
plt.grid(True)
plt.show()
```



Plano de fases: órbitas y campo de direcciones

```
In [8]: # Figura cuadrada
plt.figure(figsize = (8,8))

# Representamos el campo de direcciones en el plano
X, Y = np.meshgrid(np.linspace(0, 20, 21), np.linspace(0, 20, 21))
U = a*X-b*X*Y
V = -c*Y+d*X*Y
modulo2 = U**2 + V**2
D1 = np.ones([21, 21])
D2 = np.ones([21, 21])
for i in range(21):
    for j in range(21):
        if modulo2[i, j] == 0:
            D1[i, j] = 0
            D2[i, j] = 0
        else:
            D1[i, j] = U[i, j]/np.sqrt(modulo2[i, j])
            D2[i, j] = V[i, j]/np.sqrt(modulo2[i, j])

plt.quiver(X, Y, D1, D2, angles = 'xy', scale = 30)

# Representamos varias soluciones del sistema
tiempo = np.linspace(0, 10, 200)
lc = np.linspace(0.01, 50, 20)
for r in lc:
    P0 = [r, 50]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

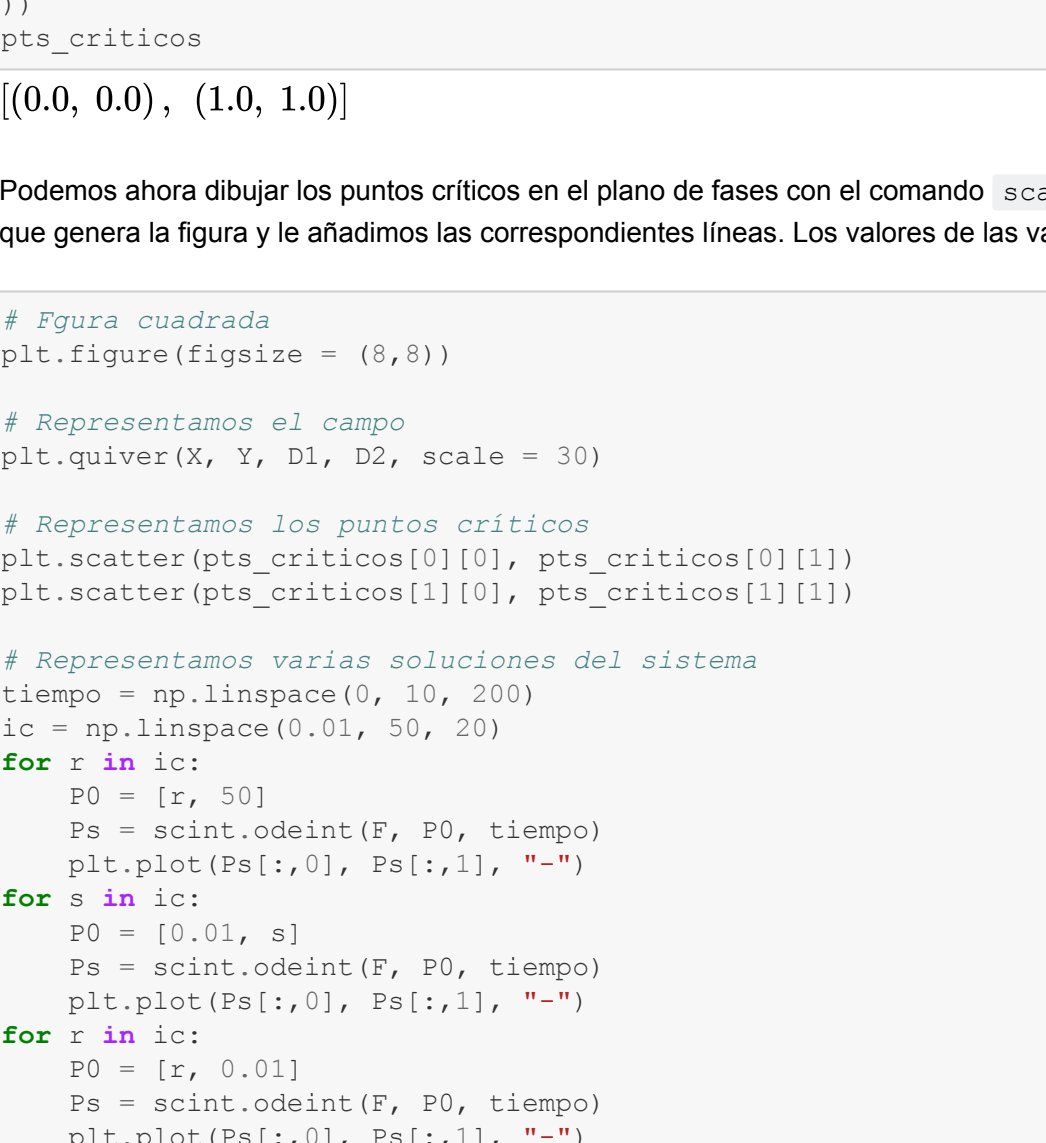
for s in lc:
    P0 = [0.01, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

for r in lc:
    P0 = [r, 0.01]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

for s in lc:
    P0 = [50, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

plt.xlim(-0.1, 20)
plt.ylim(-0.1, 20)

plt.xlabel("Actividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("Plano de fases", fontsize=13)
plt.grid(True)
plt.show()
```



4. Puntos críticos y soluciones de equilibrio

Dado un sistema de edo's

$$\begin{aligned}\frac{dx}{dt} &= F(x,y) \\ \frac{dy}{dt} &= G(x,y)\end{aligned}$$

los puntos críticos son los que hacen cero ambas derivadas, es decir, las soluciones del sistema de ecuaciones algebraicas

$$\begin{aligned}F(x,y) &= 0 \\ G(x,y) &= 0\end{aligned}$$

Las soluciones de equilibrio corresponden a soluciones constantes: son aquellas que verifican que en todo instante el sistema está en el punto crítico. Si (x_p, y_p) es un punto crítico, una solución de equilibrio será

$$x(t) \equiv x_p, \quad y(t) \equiv y_p.$$

Cargamos el módulo `sympy` e introducimos los símbolos necesarios, así como las expresiones de las ecuaciones que tenemos que resolver.

```
In [9]: # Cargamos el módulo 'sympy'
import sympy as sp

# Declaramos los símbolos que vamos a utilizar
a, b, c, d = sp.symbols('a b c d', positive = True)
x, y, a_s, b_s, c_s, d_s = sp.symbols('x y, a_s, b_s, c_s, d_s')
```

Si nos llamamos a ambas partes de la derecha de las ecuaciones del sistema

$$\begin{aligned}F &= a_s*x - b_s*x*y \\ G &= -c_s*y + d_s*x*y\end{aligned}$$

Sustituimos los valores de los parámetros y obtenemos la solución del sistema

```
In [11]: pts_criticos = sp.solve([F._subs({a_s:0.3029, b_s:0.3029}), G._subs({c_s:0.15145, d_s:0.15145})], (x, y))
pts_criticos
```

Out[11]: $[(0.0, 0.0), (1.0, 1.0)]$

Podemos ahora dibujar los puntos críticos en el plano de fases con el comando `scatter`. Del anterior código sólo es necesaria la parte que genera la figura y la añadimos las correspondientes líneas. Los valores de las variables todavía están guardados en la memoria.

```
In [12]: # Figura cuadrada
plt.figure(figsize = (8,8))

# Representamos el campo
plt.quiver(X, Y, D1, D2, scale = 30)

# Representamos los puntos críticos
plt.scatter(pts_criticos[0][0], pts_criticos[0][1])
plt.scatter(pts_criticos[1][0], pts_criticos[1][1])

# Representamos varias soluciones del sistema
tiempo = np.linspace(0, 10, 200)
lc = np.linspace(0.01, 50, 20)
for r in lc:
    P0 = [r, 50]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

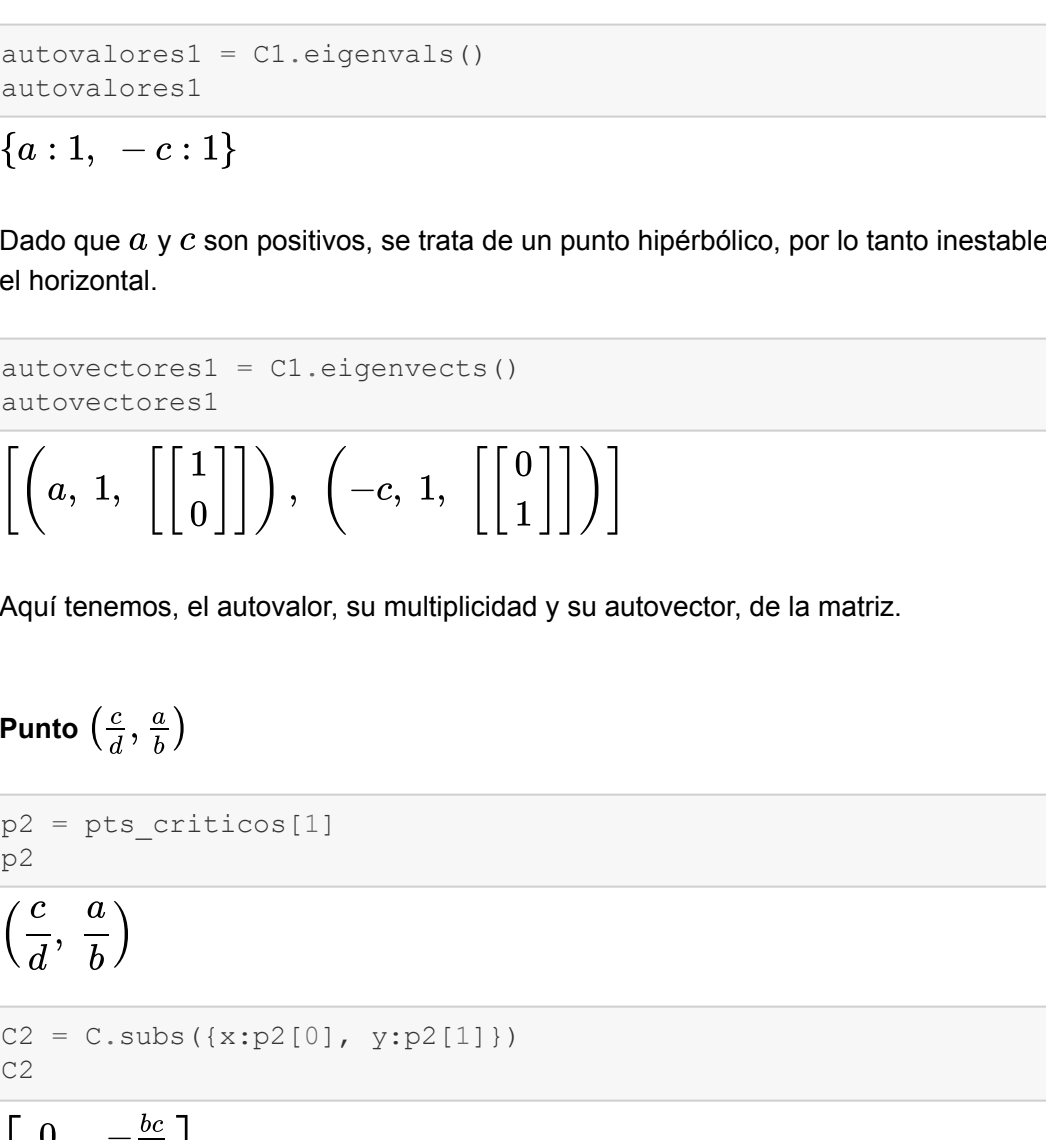
for s in lc:
    P0 = [0.01, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

for r in lc:
    P0 = [r, 0.01]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

for s in lc:
    P0 = [50, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "r-")

plt.xlim(-1, 20)
plt.ylim(-1, 20)

plt.xlabel("Actividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("Plano de fases", fontsize=13)
plt.show()
```



5. Estabilidad de los puntos de equilibrio

Puntos críticos

```
In [14]: x, y = sp.symbols('x y')
a, b, c, d = sp.symbols('a b c d', positive = True)
F = a*x - b*x*y
G = -c*y + d*x*y
x, y
```

Out[14]: $(ax - bxy, -cy + dxy)$

```
In [15]: pts_criticos = sp.solve([F, G], (x,y))
pts_criticos
```

Out[15]: $[(0, 0), \left(\frac{c}{d}, \frac{a}{b}\right)]$

Matriz Jacobiana

```
In [16]: C = sp.Matrix([[f.diff(x), f.diff(y)], [g.diff(x), g.diff(y)]])
C
```

Out[16]: $\begin{bmatrix} a - by & -bx \\ dy & -c + dx \end{bmatrix}$

Punto (0,0)

```
In [17]: p1 = pts_criticos[0]
p1
```

Out[17]: $(0, 0)$

```
In [18]: C1 = C.subs({x:p1[0], y:p1[0]})
C1
```

Out[18]: $\begin{bmatrix} a & 0 \\ 0 & -c \end{bmatrix}$

```
In [19]: tr1 = sp.trace(C1)
sp.simplify(tr1)
```

Out[19]: $a - c$

```
In [20]: det1 = sp.det(C1)
sp.simplify(det1)
```

Out[20]: $-ac$

```
In [21]: autovalores1 = C1.eigenvals()
autovalores1
```

Out[21]: $\{a - 1, -c : 1\}$

Dado que a y c son positivos, se trata de un punto hipérbico, por lo tanto inestable. La variedad estable es el eje vertical y la inestable es el horizontal.

```
In [22]: autovalores1 = C1.eigenvals()
autovalores1
```

Out[22]: $\left\{ \left(a, 1, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right), \left(-c, 1, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \right\}$

Aquí tenemos, el autovector, su multiplicidad y su autovector, de la matriz.

Punto $\left(\frac{c}{d}, \frac{a}{b}\right)$

```
In [23]: p2 = pts_criticos[1]
p2
```

Out[23]: $\left(\frac{c}{d}, \frac{a}{b}\right)$

```
In [24]: C2 = C.subs({x:p2[0], y:p2[1]})
C2
```

Out[24]: $\begin{bmatrix} 0 & -\frac{bc}{d} \\ \frac{ad}{b} & 0 \end{bmatrix}$

```
In [25]: tr2 = sp.trace(C2)
sp.simplify(tr2)
```

Out[25]: 0

```
In [26]: det2 = sp.det(C2)
sp.simplify(det2)
```

Out[26]: ac

```
In [27]: autovalores2 = C2.eigenvals()
autovalores2
```

Out[27]: $\{-i\sqrt{ac}, 1, i\sqrt{ac}, 1\}$

Ambos autovalores son imaginarios puros. Por lo tanto, este punto de equilibrio es un centro y es estable.

```
In [28]: autovalores2 = C2.eigenvals()
autovalores2
```

Out[28]: $\left\{ \left(-i\sqrt{ac}, 1, \begin{bmatrix} -\frac{ib\sqrt{c}}{\sqrt{ad}} \\ 1 \end{bmatrix} \right), \left(i\sqrt{ac}, 1, \begin{bmatrix} \frac{ib\sqrt{c}}{\sqrt{ad}} \\ 1 \end{bmatrix} \right) \right\}$

La traza de la matriz es nula y, dado que a y c son estrictamente positivos, deducimos que el determinante de la matriz es positivo. Por lo tanto, deducimos que el sistema en el segundo punto crítico es estable.

Modelo de Lotka y Volterra para el sistema de sueño REM

Este modelo está formado por dos edo's no lineales que constituyen un modelo simplificado de la interacción de dos poblaciones de neuronas. involucradas en el sistema de sueño REM, una de las cuales es un depredador y la otra una presa. Las variables x e y representan respectivamente, el número de individuos de la población de neuronas FTG activas y de la población de neuronas LC activas, respectivamente, que depende del tiempo t . Estas son las ecuaciones:

$$\begin{aligned}\frac{dx}{dt} &= ax - cxy \\ \frac{dy}{dt} &= -dy + xy\end{aligned}$$

Aquí las constantes a , b , c y d son parámetros reales positivos.

- Solución numérica del PVI (para valores concretos de los parámetros)
- Gráficas de la actividad neuronal como funciones del tiempo y de órbitas (para valores concretos de los parámetros)
- Representar campo de direcciones y el plano de fases (para valores concretos de los parámetros)
- Representar puntos de equilibrio en el plano de fases (para valores concretos de los parámetros)
- Estabilidad de los punto de equilibrio del sistema: estudio cualitativo del caso general (valores arbitrarios de los parámetros del sistema)

Cargamos los módulos necesarios: numérico, gráficos, integración numérica y lenguaje matemático

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import scipy.integrate as scint
import sympy as sp
sp.init_printing()
```

La función que define este sistema es

$$F(x,y) = (ax - bxy, -cy + dxy)$$

1. Solución numérica del PVI

```
In [2]: # Introducimos los valores de las constantes
a,b,c,d = 0.5490,0.5490,0.2745,0.2745

# La función 'F' del sistema y el vector 'P' que se compone de las funciones desconocidas 'x(t)', 'y(t)'
def F(P,t):
    return [a*P[0] - b*P[0]*P[1], -c*P[1] + d*P[0]*P[1]]

# Damos 500 valores a la variable independiente 't' entre 0 y 100
tiempo = np.linspace(0, 100, 500)

# Condición inicial
P0 = [1.0, 4.5]

# Obtenemos la resolución numérica del sistema en forma de matriz 'Ps' con una columna para los valores de cada población.
Ps = scint.odeint(F, P0, tiempo)

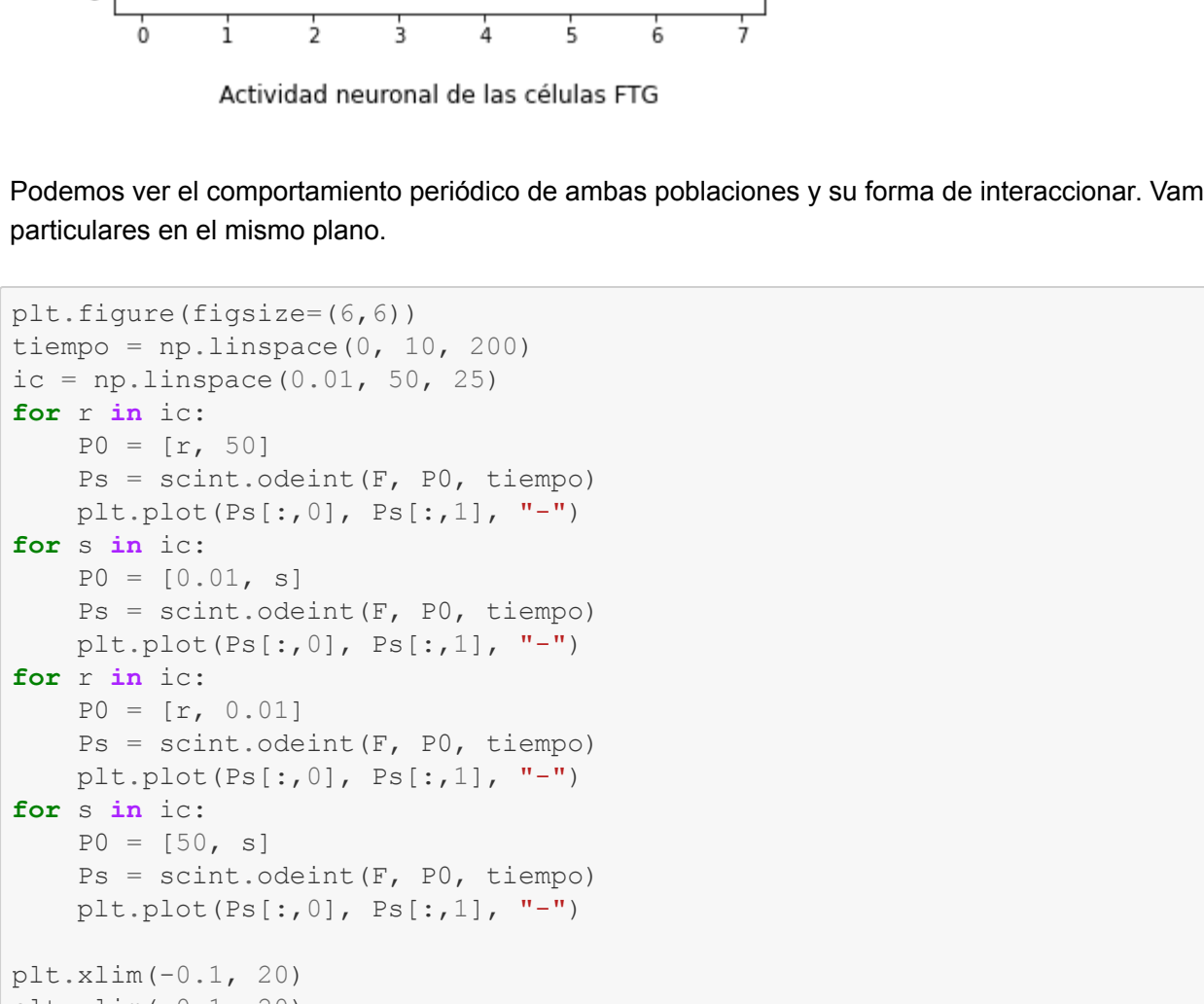
# Valores obtenidos de la población de células FTG activas en la primera columna
FTG = Ps[:,0]

# Valores obtenidos de la población de células FC activas en la segunda
LC = Ps[:,1]
```

2. Gráficas de poblaciones como funciones del tiempo y de órbitas

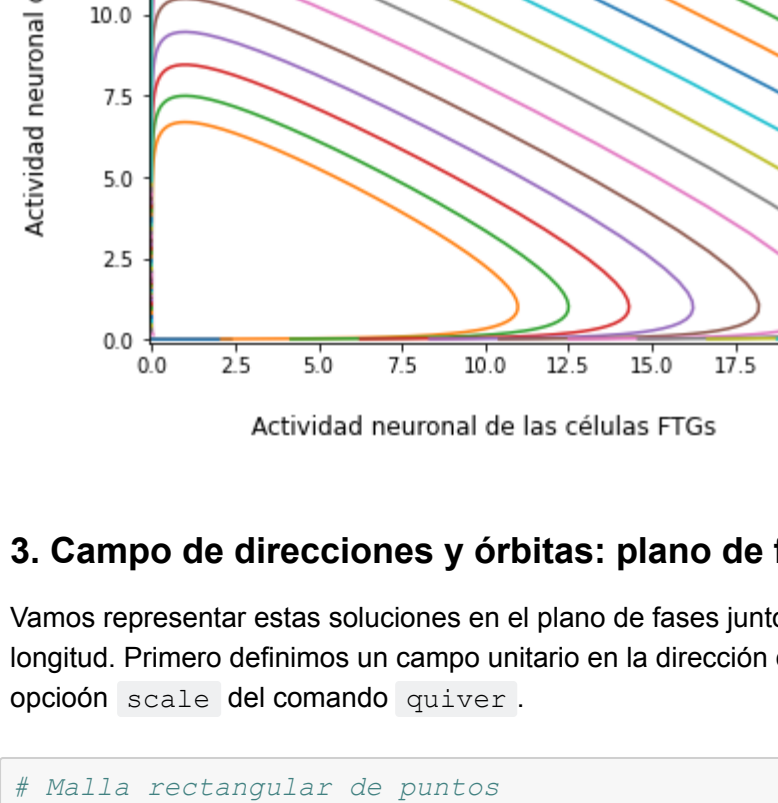
Dibujamos ambas gráficas de la evolución de cada población respecto del tiempo en los instantes indicados

```
In [3]: plt.figure(figsize = (10,6))
plt.plot(tiempo, FTG, label="Células FTG activas")
plt.plot(tiempo, LC, label="Células LC activas")
plt.xlabel("tiempo", fontsize=12)
plt.ylabel("Actividad neuronal", fontsize=12)
plt.grid(True)
plt.legend()
plt.show()
```



Dibujamos la gráfica de la trayectoria en el plano de fases

```
In [4]: plt.figure(figsize =(6,6))
plt.plot(FTG, LC, "-")
plt.xlabel("\nActividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.show()
```



Podemos ver el comportamiento periódico de ambas poblaciones y su forma de interactuar. Vamos a dibujar varias soluciones particulares en el mismo plano.

```
In [5]: plt.figure(figsize=(6,6))
tiempo = np.linspace(0, 10, 200)
ic = np.linspace(0.01, 50, 25)

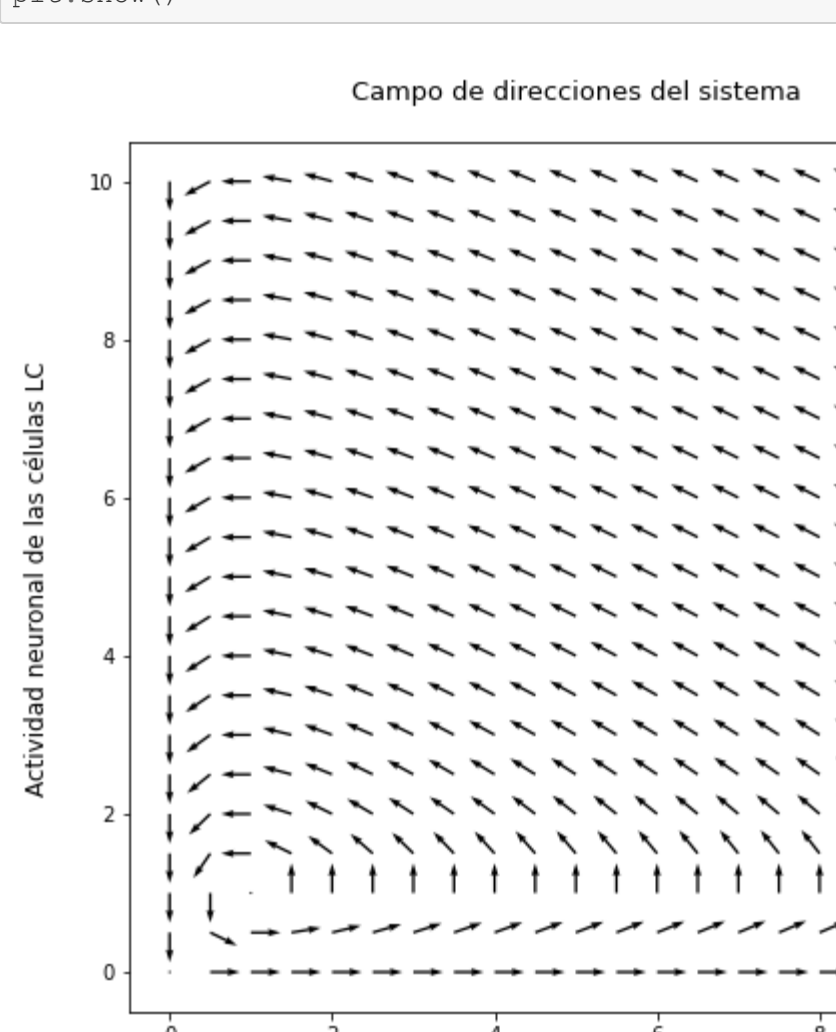
for r in ic:
    P0 = [r, 50]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

for s in ic:
    P0 = [r, 0.01]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

for s in [50, 1]:
    P0 = [50, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

plt.xlim(-0.1, 20)
plt.ylim(-0.1, 20)

plt.xlabel("\nActividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("\nÓrbitas del sistema", fontsize=13)
plt.show()
```



3. Campo de direcciones y órbitas: plano de fases

Vamos representar estas soluciones en el plano de fases junto con el campo de direcciones de forma que los vectores tengan la misma longitud. Primero definimos un campo unitario en la dirección del campo asociado al sistema. Después controlamos la longitud con la opción 'scale' del comando 'quiver'.

```
In [6]: # Malla rectangular de puntos
X, Y = np.meshgrid(np.linspace(0, 10, 21), np.linspace(0, 10, 21))

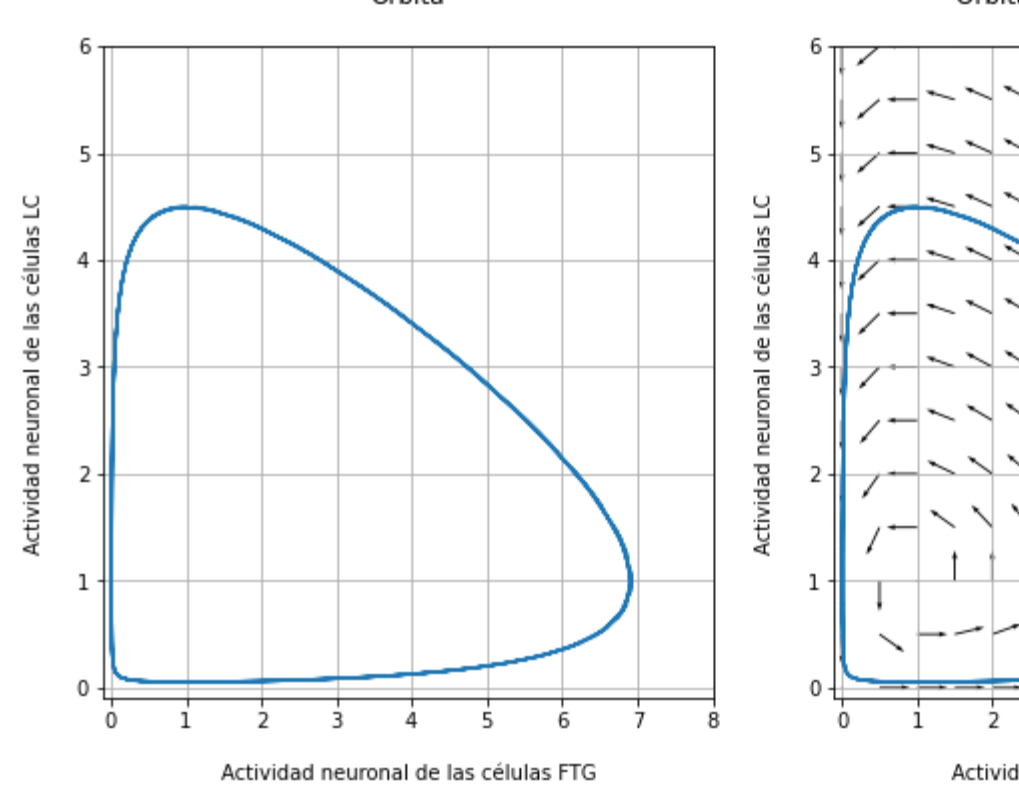
# Componentes del vector de dirección
U = a*X-b*X*Y
V = -c*Y+d*X*Y

# Definimos un campo de vectores unitario salvo cuando se anula
modulo2 = U**2 + V**2
D1 = np.ones((21, 21))
D2 = np.ones((21, 21))
for i in range(21):
    for j in range(21):
        if modulo2[i, j] == 0:
            D1[i, j] = 0
            D2[i, j] = 0
        else:
            D1[i, j] = U[i, j]/np.sqrt(modulo2[i, j])
            D2[i, j] = V[i, j]/np.sqrt(modulo2[i, j])

# Figura cuadrada
plt.figure(figsize = (8,8))

# Representamos el campo de direcciones en el plano
plt.quiver(X, Y, D1, D2, angles = 'xy', scale = 30, headwidth = 3)

plt.xlabel("\nActividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("\nCampo de direcciones del sistema", fontsize=13)
plt.show()
```



Campo de direcciones con una curva solución

```
In [7]: fig = plt.figure(figsize = (12,6))
fig.supdtitle('Campo de direcciones y órbita del modelo de Lotka y Volterra de predación con crecimiento logístico de la presa')

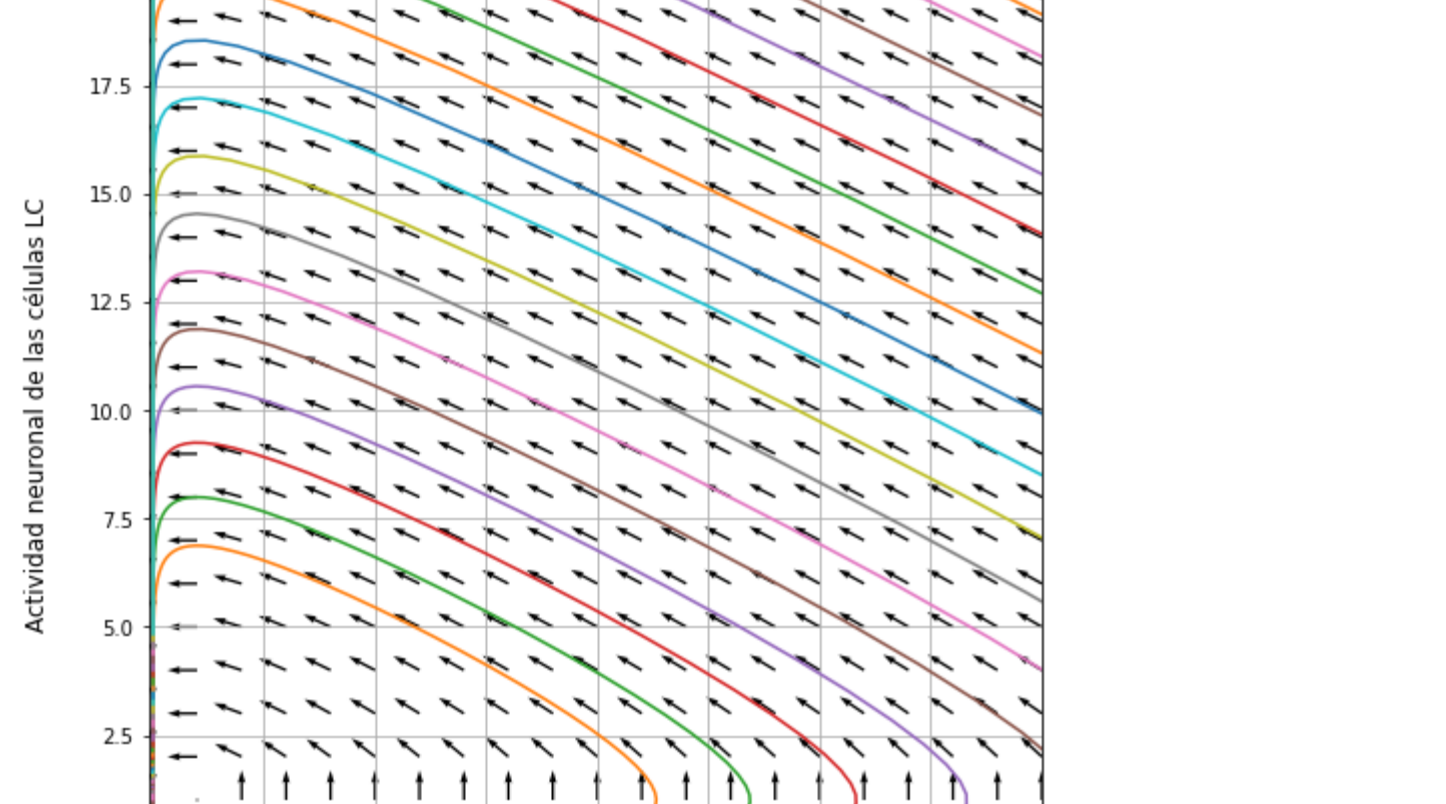
ejes1 = fig.add_subplot(1,2,1)
ejes1.plot(FTG, LC, "-")
ejes1.set_title("Órbita")
ejes1.set_xlabel("\nActividad neuronal de las células FTG")
ejes1.set_ylabel("\nActividad neuronal de las células LC")
ejes1.set_xlim(-0.1, 8)
ejes1.set_ylim(-0.1, 6)
plt.grid(True)

ejes2 = fig.add_subplot(1,2,2)

X, Y = np.meshgrid(np.linspace(0, 10, 21), np.linspace(0, 10, 21))
U = a*X-b*X*Y
V = -c*Y+d*X*Y
modulo2 = U**2 + V**2
D1 = np.ones((21, 21))
D2 = np.ones((21, 21))
for i in range(21):
    for j in range(21):
        if modulo2[i, j] == 0:
            D1[i, j] = 0
            D2[i, j] = 0
        else:
            D1[i, j] = U[i, j]/np.sqrt(modulo2[i, j])
            D2[i, j] = V[i, j]/np.sqrt(modulo2[i, j])

ejes2.plot(X, Y, D1, D2, angles = 'xy', scale = 20, headwidth = 3)
ejes2.plot(FTG, LC, "-")
ejes2.set_title("Campo de direcciones y órbita")
ejes2.set_xlabel("\nActividad neuronal de las células FTG")
ejes2.set_ylabel("\nActividad neuronal de las células LC")
ejes2.set_xlim(-0.1, 8)
ejes2.set_ylim(-0.1, 6)
plt.grid(True)

plt.savefig('orbita_campo_direcciones_PPL.png')
plt.show()
```



Plano de fases: órbitas y campo de direcciones

```
In [8]: # Figura cuadrada
plt.figure(figsize = (8,8))

# Representamos el campo de direcciones en el plano
X, Y = np.meshgrid(np.linspace(0, 20, 21), np.linspace(0, 20, 21))
U = a*X-b*X*Y
V = -c*Y+d*X*Y
modulo2 = U**2 + V**2
D1 = np.ones((21, 21))
D2 = np.ones((21, 21))
for i in range(21):
    for j in range(21):
        if modulo2[i, j] == 0:
            D1[i, j] = 0
            D2[i, j] = 0
        else:
            D1[i, j] = U[i, j]/np.sqrt(modulo2[i, j])
            D2[i, j] = V[i, j]/np.sqrt(modulo2[i, j])
plt.quiver(X, Y, D1, D2, angles = 'xy', scale = 30)

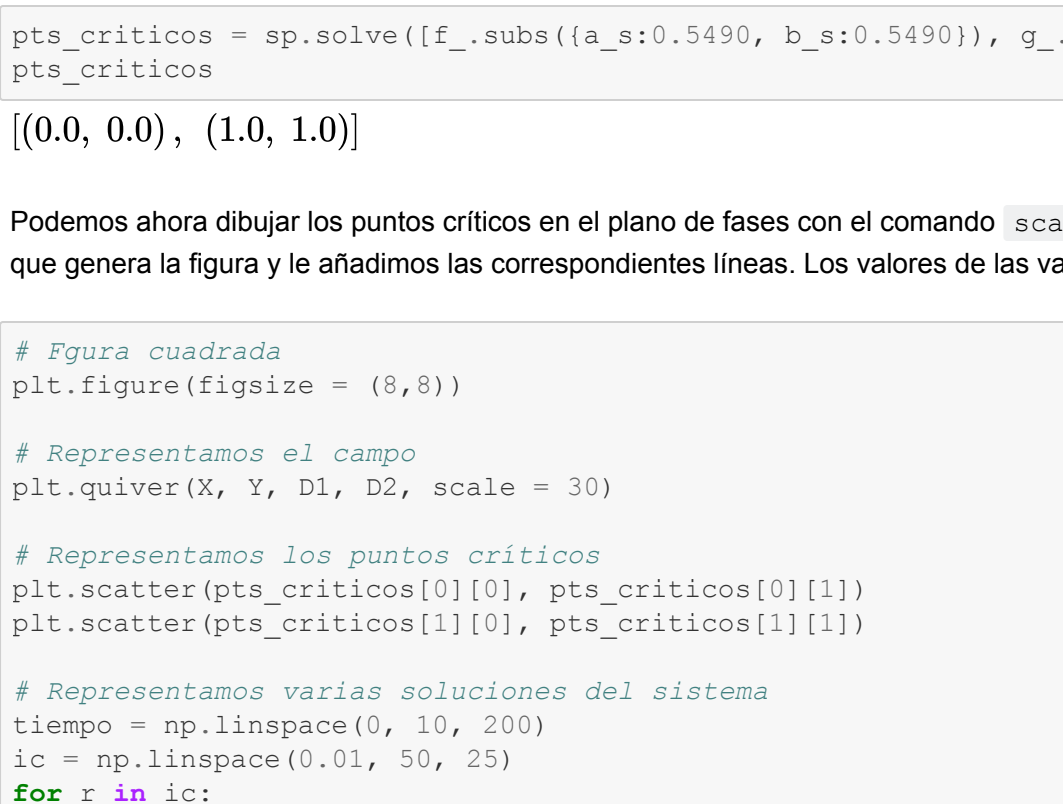
# Representamos varias soluciones del sistema
tiempo = np.linspace(0, 10, 200)
ic = np.linspace(0.01, 50, 20)
for r in ic:
    P0 = [r, 50]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

for s in ic:
    P0 = [r, 0.01]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

for s in [50, 1]:
    P0 = [50, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

plt.xlim(-0.1, 20)
plt.ylim(-0.1, 20)

plt.xlabel("\nActividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células LC", fontsize=12)
plt.title("Plano de fases", fontsize=13)
plt.grid(True)
plt.show()
```



4. Puntos críticos y soluciones de equilibrio

Dado un sistema de edo's

$$\begin{aligned}\frac{dx}{dt} &= F(x,y) \\ \frac{dy}{dt} &= G(x,y)\end{aligned}$$

los puntos críticos son los que hacen cero ambas derivadas, es decir, las soluciones del sistema de ecuaciones algebraicas

$$\begin{aligned}F(x,y) &= 0 \\ G(x,y) &= 0\end{aligned}$$

Las soluciones de equilibrio corresponden a soluciones constantes: son aquellas que verifican que en todo instante el sistema está en el punto crítico. Si (x_p, y_p) es un punto crítico, una solución de equilibrio será

$$x(t) \equiv x_p, \quad y(t) \equiv y_p.$$

Cargamos el módulo 'sympy' e introducimos los símbolos necesarios, así como las expresiones de las ecuaciones que tenemos que resolver.

```
In [9]: # Cargamos el módulo 'sympy'
import sympy as sp

# Declaramos los símbolos que vamos a utilizar
# Nombramos a las constantes así para que no interfieran con las variables a, b, c y d antes definidas
x, y, a_s, b_s, c_s, d_s = sp.symbols('x y, a_s, b_s, c_s, d_s')

# Nombramos a ambas partes de la derecha de las ecuaciones del sistema
f_ = a_s*x-b_s*x*y
g_ = -c_s*y+d_s*x*y
```

Sustituimos los valores de los parámetros y obtenemos la solución del sistema

```
In [11]: pts_criticos = sp.solve([f_.subs({a_s:0.5490, b_s:0.5490}), g_.subs({c_s:0.2745, d_s:0.2745})], (x, y))
pts_criticos

Out[11]: [(0, 0), (1, 1)]
```

Podemos ahora dibujar los puntos críticos en el plano de fases con el comando 'scatter'. Del anterior código sólo es necesaria la parte que genera la figura y le añadimos las correspondientes líneas. Los valores de las variables todavía están guardados en la memoria.

```
In [12]: # Figura cuadrada
plt.figure(figsize = (8,8))

# Representamos el campo
plt.quiver(X, Y, D1, D2, scale = 30)

# Representamos los puntos críticos
plt.scatter(pts_criticos[0][0], pts_criticos[0][1])
plt.scatter(pts_criticos[1][0], pts_criticos[1][1])

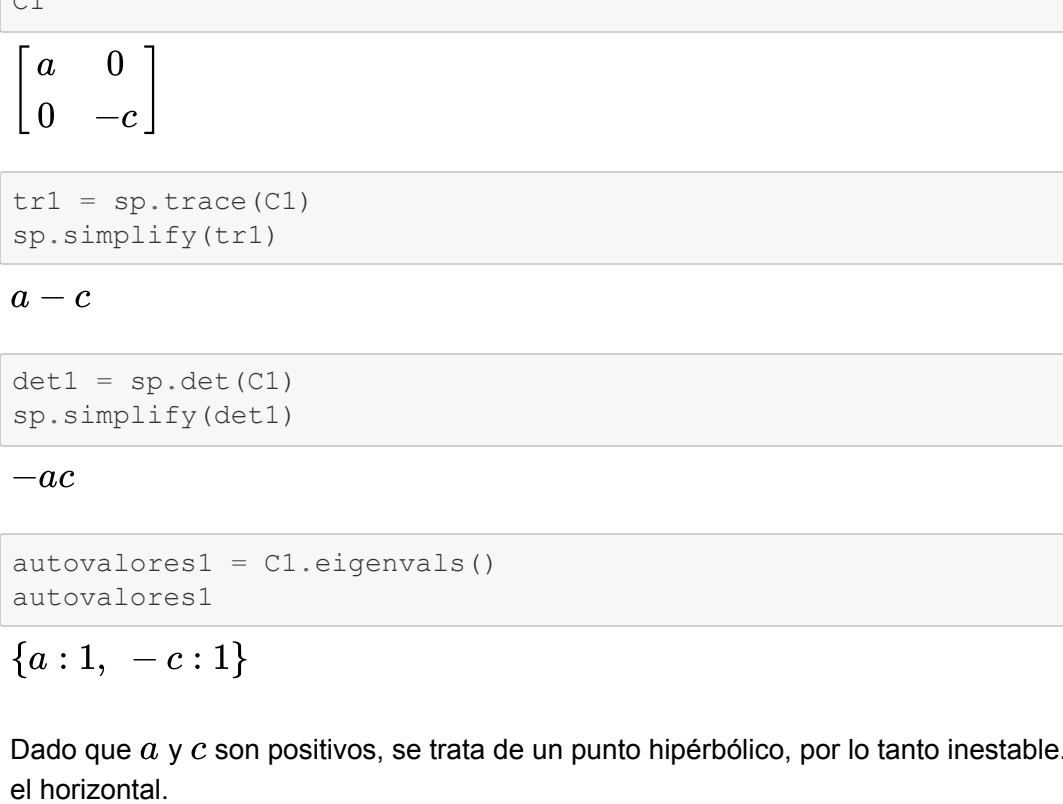
# Representamos varias soluciones del sistema
tiempo = np.linspace(0, 10, 200)
ic = np.linspace(0.01, 50, 25)
for r in ic:
    P0 = [r, 50]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

for s in ic:
    P0 = [r, 0.01]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

for s in [50, 1]:
    P0 = [50, s]
    Ps = scint.odeint(F, P0, tiempo)
    plt.plot(Ps[:,0], Ps[:,1], "-")

plt.xlim(-1, 20)
plt.ylim(-1, 20)

plt.xlabel("\nActividad neuronal de las células FTG", fontsize=12)
plt.ylabel("Actividad neuronal de las células FC", fontsize=12)
plt.title("Plano de fases", fontsize=13)
plt.show()
```



5. Estabilidad de los puntos de equilibrio

Puntos críticos

```
In [14]: x, y = sp.symbols('x y')
a, b, c, d = sp.symbols('a b c d', positive = True)
f_ = a*x - b*x*y
g_ = -c*y + d*x*y
f, g

Out[14]: (a*x - b*x*y, -c*y + d*x*y)
```

```
In [15]: pts_criticos = sp.solve([f,g], (x,y))
pts_criticos
```

```
Out[15]: [(0, 0), (c/d, a/b)]
```

Matriz Jacobiana

```
In [16]: C = sp.Matrix([[f.diff(x), f.diff(y)],[g.diff(x), g.diff(y)]]
C

Out[16]: [[a - b*y, -b*x], [d, -c + d*x]]

Punto (0,0)
```

```
In [17]: p1 = pts_criticos[0]
p1
```

```
Out[17]: (0, 0)
```

```
In [18]: C1 = C.subs({x:p1[0], y:p1[0]})
C1
```

```
Out[18]: [[a, 0], [0, -c]]
```

```
In [19]: tr1 = sp.trace(C1)
sp.simplify(tr1)
```

```
Out[19]: a - c
```

```
In [20]: det1 = sp.det(C1)
sp.simplify(det1)
```

```
Out[20]: -a*c
```

```
In [21]: autovalores1 = C1.eigenvals()
autovalores1
```

```
Out[21]: {a: 1, -c: 1}
```

Dado que a y c son positivos, se trata de un punto hipérbico, por lo tanto inestable. La variedad estable es el eje vertical y la inestable es el horizontal.

```
In [22]: autovectores1 = C1.eigenvects()
autovectores1
```

```
Out[22]: [(a, 1, [[1], [0]]), (-c, 1, [[0], [1]])]
```

Aquí tenemos, el autovector, su multiplicidad y su autovector, de la matriz.

Punto $(\frac{c}{d}, \frac{a}{b})$

```
In [23]: p2 = pts_criticos[1]
p2
```

```
Out[23]: (c/d, a/b)
```

```
In [24]: C2 = C.subs({x:p2[0], y:p2[1]})
C2
```

```
Out[24]: [[c, -b*c/d], [ad/b, 0]]
```

```
In [25]: tr2 = sp.trace(C2)
sp.simplify(tr2)
```

```
Out[25]: 0
```

```
In [26]: det2 = sp.det(C2)
sp.simplify(det2)
```

```
Out[26]: a*c
```

```
In [27]: autovalores2 = C2.eigenvals()
autovalores2
```

```
Out[27]: {-i*sqrt(a)*sqrt(c): 1, i*sqrt(a)*sqrt(c): 1}
```

Ambos autovalores son imaginarios puros. Por lo tanto, este punto de equilibrio es un centro y es estable.

```
In [28]: autovectores2 = C2.eigenvects()
autovectores2

Out[28]: [(-i*sqrt(a)*sqrt(c), 1, [[-i*sqrt(c)/sqrt(d)], [1]]), (i*sqrt(a)*sqrt(c), 1, [[i*sqrt(c)/sqrt(d)], [1]])]
```

La traza de la matriz es nula y, dado que a y c son estrictamente positivos, deducimos que el determinante de la matriz es positivo. Por lo tanto, deducimos que el sistema en el segundo punto crítico es estable.