Reporte de Programa y Usos

Rodrigo Moreno

9 de julio de 2020

Escribí un código que descompone el problema en varios objetos. Primero que nada, se debe definir un objeto que contiene los parámetros del sistema `i_max`, `u_max`, `gamma` y `beta`. Este objeto es de una clase llamada `SIR`. Se comienza el código con los siguientes comandos:

```
imax = 0.1
umax = 0.5
gamma = 0.2
beta = 0.5

A = SIR()
A.set_params(imax, umax, gamma, beta)
A.find_tau()
A.find_psi()
A.find_theta()
A.find_rho()
```

La clase `SIR` tiene diferentes métodos. Entre ellos, los importantes son `SIR.find_tau()`, `SIR.find_psi()`, `SIR.find_theta()` y `SIR.find_rho()`. Estos métodos permiten encontrar cuatro curvas distintas (fig. 1), que a su vez permiten generar las trayectorias desde un punto inicial. Cada una de estas curvas es de una clase `CurveSegment()`, de la cual se va a comentar más adelante, accesible como atributos de `A`: `A.tau`, `A.phi`, `A.theta` y `A.rho`. La curva `A.tau` es aquella con máximo en $[\frac{\gamma}{\beta}, I_{max}]$, se calcula con $u = 0$, y es la frontera para la zona «segura». `A.phi` tiene máximo en $[\frac{\gamma}{(1-u_{max})\beta}, I_{max}]$ y se genera con $u = u_{max}$. `A.theta` inicia donde `A.tau` cruza 0, es con $u = u_{max}$ y tiene máximo en $\frac{\gamma}{(1-u_{max})\beta}$. Finalmente, `A.rho` es la curva que pasa por $\frac{\gamma}{(1-u_{max})\beta}$ con $u = 0$. De ahora en adelante, $\bar{S} = \frac{\gamma}{\beta}$ y $S^* = \frac{\gamma}{(1-u_{max})\beta}$.

La siguiente parte del código implica la generación de una condición inicial:

```
s0 = 0.6
i0 = 0.08
P = Point(s0, i0, A)
```

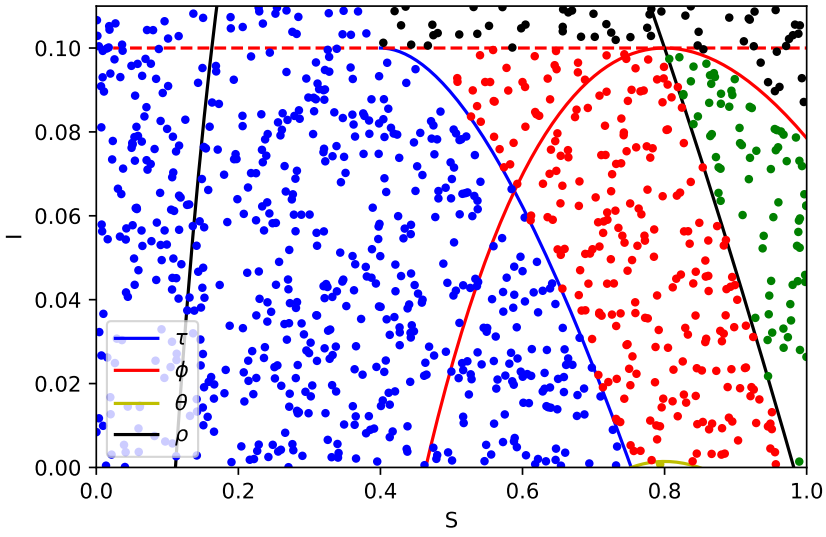


Figura 1: Curvas y regiones del sistema, generadas a partir de los parámetros mostrados arriba en el código. Los puntos azules, rojos, verdes, amarillos y negros pertenecen a las regiones 1, 2, 3, 4 y 5, respectivamente.

```
P.find_region()
```

La inicialización de un punto se hace con las coordenadas de origen `s0` e `i0`, y el sistema al que pertenece, en este caso `A`. El punto va a heredar ciertos atributos del sistema al que pertenece. El método `P.find_region()` crea un atributo de la clase que determina en qué región del sistema se encuentra con respecto a sus curvas (fig. 1). Dependiendo de la región en la que se encuentre el punto, cambia el método para generar las trayectorias. A grandes razgos:

1. No se requiere generar una trayectoria, ya que ya está en la región deseada.
2. Se genera una trayectoria principal, compuesta de dos segmentos. El primer segmento con $u = 0$ hasta llegar a I_{max}, y el segundo un `LineSegment` por la curva singular hasta llegar a $\bar{S}$. El resto de las trayectorias se genera haciendo `CurveSegment` con $u = u_{max}$ desde cualquier punto de la trayectoria principal hasta algún punto en `A.tau`.
3. Se genera una trayectoria principal compuesta por tres segmentos. El primero es un `CurveSegment` con $u = 0$ desde el punto inicial hasta `A.phi`. El segundo es la curva siguiendo a `A.phi` hasta llegar a S^*. El tercero es un `LineSegment` por la curva singular hasta $\bar{S}$. El resto de las trayectorias se genera haciendo `CurveSegment` con $u = u_{max}$ desde diferentes puntos del primer o tercer segmento, hasta llegar a `A.tau`.
4. Los puntos en esta región deben primero salir de ella con $u = 0$. La salida puede ser a cualquiera de las regiones 2, 3 o 5, por lo cual la construcción de las trayectorias se hace dependiendo de a cuál salga.
5. No se generan trayectorias porque ninguna de ellas cumple con el criterio de no sobrepasar I_{max}. Por ahora, éste es un método vacío, pero se puede adaptar a las necesidades que puedan surgir más adelante.

Finalmente, las trayectorias y los tiempos de cada una se pueden generar utilizando

```
T = TrajectoryCreator(P)
trajectories = T.get_trajectories()

for trajectory in trajectories:
        trajectory.get_time()
times = [trajectory.time for trajectory in trajectories]
ends = [trajectory.s[-1] for trajectory in trajectories]
```

El objeto `TrajectoryCreator` toma como argumento un objeto de clase `Point`. El método `TrajectoryCreator.get_trajectories()` genera una lista de objetos `Trajectory` compuestos de segmentos `CurveSegment` o `LineSegment`, cada uno con métodos para calcular sus tiempos dependiendo de la clase. La clase

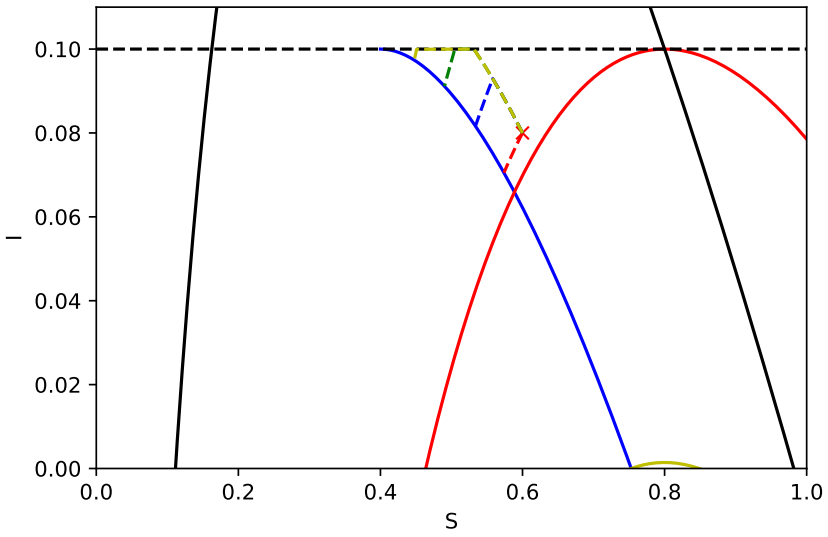
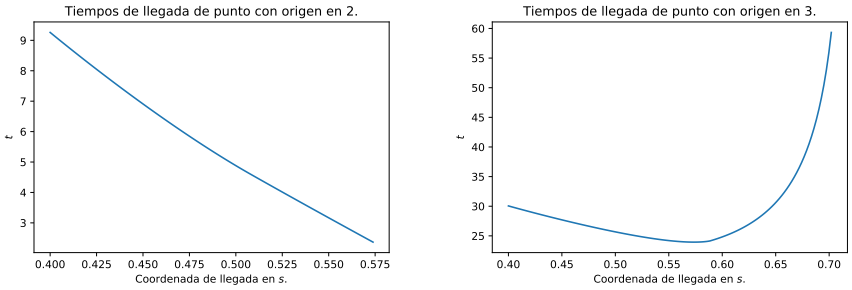


Figura 2: Muestra de trayectorias del punto `P`.

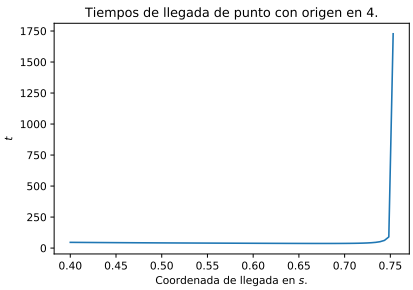
`Trajectory` también cuenta con un método para calcular su tiempo, que calcula y suma los tiempos de cada uno de sus segmentos. Una muestra de las trayectorias generadas para el punto inicial mostrado arriba se presenta en la figura 2. Los arreglos `times` y `ends` contienen los tiempos de cada trayectoria, y la coordenada en S de cada punto de intersección con `A.tau`. Estos se pueden graficar para ver las tendencias (fig. 3).

Los tiempos de integración se corroboraron haciendo simulaciones del sistema dinámico. Se generó un objeto `CurveSegment` con condiciones iniciales en $[0.6, 0.08]$ y $u = u_{max}$ (fig. 4, puntos rojos), y se calculó el tiempo t_f asociado a él. Por otro lado, se hizo una simulación del sistema con las mismas condiciones (mismos valores de parámetros y condiciones iniciales), con $t_0 = 0$ y tiempo máximo t_f. La trayectoria simulada del sistema se muestra en la figura 4 con cruces rojas. Ya que ambas curvas son iguales, se puede concluir que el resultado obtenido por la integración es correcto. De lo contrario, la curva obtenida por la simulación del sistema sería de una extensión diferente.

Se puede fijar un punto en I y tomar una muestra de puntos en S para generar diferentes trayectorias. De cada punto $s_0 \in S$, se puede obtener la trayectoria de menor tiempo y el punto s_f en el que cruza `A.tau` (fig. 5). Estas mismas trayectorias se pueden graficar para ver cómo cambian las trayectorias «óptimas» con el punto inicial (fig. 6).



(a) (b)



(c)

Figura 3: Coordenada contra tiempo de llegada de trayectorias iniciadas en las regiones 2 (3a), 3 (3b) y 4 (3c). Los puntos iniciales son los mostrados arriba (3a), $[1 - 1e^{-2}, 1e^{-2}]$ (3b) y $[0.8, 1e^{-4}]$ (3c).

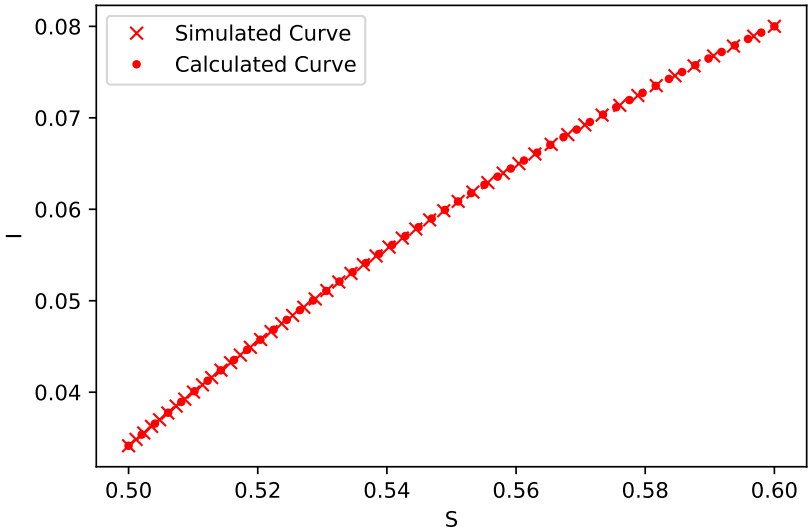


Figura 4: Curva simulada y calculada explícitamente. El tiempo de trayectoria obtenido para la curva se utilizó como punto final para la integración.

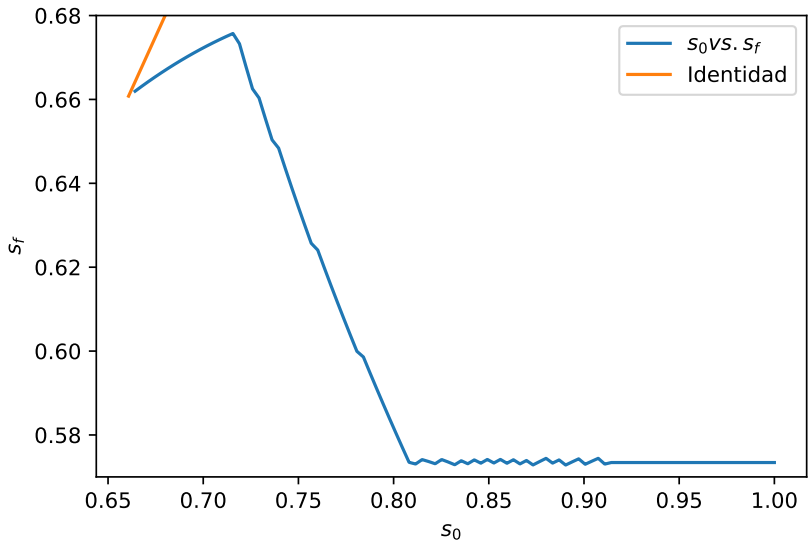


Figura 5: Punto inicial contra el punto final de la trayectoria de menor tiempo.

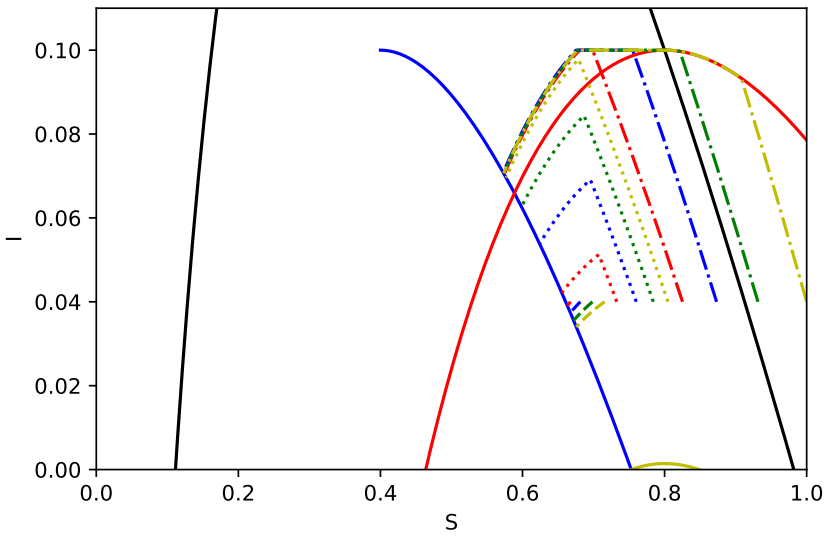


Figura 6: Se muestran diferentes trayectorias de las tres regiones de la figura 5. Las trayectorias con segmentos de línea $(--)$ corresponden a la región de incremento. Las trayectorias con línea punteada $(\cdots)$ a la región de decremento. Las trayectorias con líneas punteadas y segmento de línea $(- \cdot -)$ corresponden a la región constante.