En clases pasadas vimos que un sistema de EDOs para describir la propagación de una epidemia está dado, en su forma más simple, por el modelo SI (susceptibles ***S*** e infectados ***I***):

$$\frac{dS}{dt} = -\beta(S \times I) \quad [1]$$

$$\frac{dI}{dt} = \beta(S \times I) \quad [2]$$

donde el parámetro ***β*** es una medida de la rapidez de propagación de la epidemia. Para este sistema encontramos que la solución exacta (tomando como condición inicial ***I(0) = 1*** ) está dada por:

$$I(t) = \frac{N}{1-(1-N)e^{-N\beta t}} \quad [3]$$

$$S(t) = N - \frac{N}{1-(1-N)e^{-N\beta t}} \quad [4]$$

En esta clase y la próxima consideraremos modelos más cercanos a lo que podría ser una situación real. Consideremos ahora que un individuo infectado puede recuperarse eventualmente y esto conduce a pensar en un tercer compartimiento de población que denominaremos ***R*** de individuos recuperados. Vamos a asumir que la tasa de crecimiento de la población de recuperados ***dR/dt*** es proporcional al número de infectados ***I*** y esto implica una modificación en la ecuación [2] y agregar una ecuación extra, dando como resultado las siguientes 3 ecuaciones:

$$\frac{dS}{dt} = -\beta(S \times I) \quad [5]$$

$$\frac{dI}{dt} = \beta(S \times I) - \mu I \quad [6]$$

$$\frac{dR}{dt} = \mu I \quad [7]$$

Donde en este caso $\mu$ representa la velocidad de recuperación de los individuos. Aquí el nuevo término $\mu I$ se resta en la ecuación [6] porque los individuos recuperados implican una reducción en el crecimiento de la población de infectados, mientras que la ecuación [7] es la relación de proporcionalidad entre el crecimiento de la población ***R*** y el número de infectados ***I***.

Este sistema ya es lo suficientemente interesante como para que con él podamos explicar algunas de las cosas que están sucediendo hoy en día respecto a la propagación del SARS-COV-2. Así que lo primero que haremos será solucionarlo numéricamente con RK4 igual que hicimos con el sistema de Lorenz en la clase pasada. Para ello podemos usar gran parte del notebook de la clase pasada. Por ejemplo la función ***rk4vec()*** queda idéntica y solo debemos modificar la clase que contiene las ecuaciones del sistema dinámico, para cuyo caso tendremos lo siguiente:

$(\beta, \mu)$

$$S' = -\beta \cdot S \cdot I$$
$$I' = \beta \cdot S \cdot I - \mu I$$
$$R' = \mu I$$

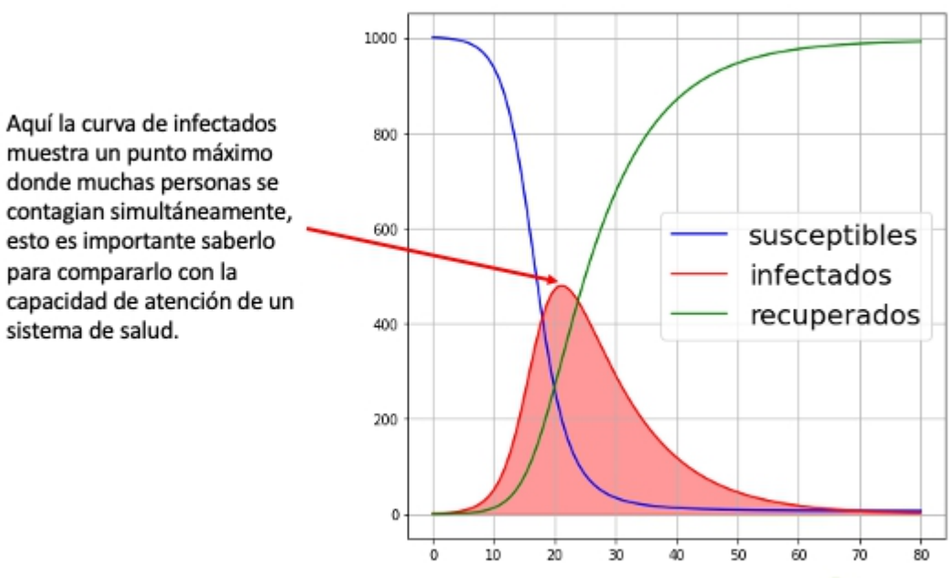
```python
class SIR_model():

  def __init__(self, init_condition, tmin = 0., tmax = 50., n = 10000, **params):
    self.tmin = tmin
    self.tmax = tmax
    self.n = n
    self.t = np.linspace(self.tmin, self.tmax, self.n)
    self.dt = self.t[1] - self.t[0]
    self.S = np.zeros([self.n])
    self.I = np.zeros([self.n])
    self.R = np.zeros([self.n])
    self.set_params(**params)
    self.init_condition = init_condition

  def set_params(self, beta, miu):
    # definición de parametros del modelo
    self.beta = beta
    self.miu =  miu

  def func(self, t, u):
    #sistema of EDOs
    self.uprime = np.zeros_like(u)
    self.uprime[0] = -self.beta*u[0]*u[1]
    self.uprime[1] = self.beta*u[0]*u[1]-self.miu*u[1]
    self.uprime[2] = self.miu*u[1]
    return self.uprime

  def run_solver(self):
      # ejecutar solucion por RK4
      self.u0 = np.array(self.init_condition)
      self.u1 = np.zeros_like(self.u0)
      for i in range(self.n):
        self.S[i] = self.u0[0]
        self.I[i] = self.u0[1]
        self.R[i] = self.u0[2]
        self.u1 = rk4vec(self.t[i], self.u0, self.dt, self.func)
        self.u0 = np.copy(self.u1)
```

Aquí modificamos la definición de variables para el modelo SIR

Con esta clase ya definida es sencillo ejecutar una simulación de SIR de forma similar a como lo hicimos con el sistema de Lorenz y además hacer una visualización:

```
params = {'beta': 0.0005, 'miu': 0.1}
init_condition = [1000, 1, 0]
model = SIR_model(init_condition=init_condition, tmax=80, n = 50000, **params)
model.run_solver()
plt.figure(figsize=(7,7))
plt.plot(model.t, model.S, label = 'susceptibles', color = 'b')
plt.plot(model.t, model.I, label = 'infectados', color = 'r')
plt.plot(model.t, model.R, label = 'recuperados', color = 'g')
plt.fill_between(model.t, 0, model.I, color = 'r', alpha = 0.4)
plt.grid(True)
plt.legend(fontsize = 20)
plt.show()
```

En esta celda de código consideramos una situación donde la tasa de recuperación **μ** es mayor a la tasa de propagación de la epidemia **β** y tomamos una condición inicial con una población de susceptibles de 1000, un solo contagio y cero personas recuperadas. El resultado se ve en la siguiente gráfica:




La solución de este sistema, bajo las condiciones y parámetros previamente indicados, muestra que:

1. La curva de susceptibles siempre decrece con el tiempo, igual que con el sistema SI.
2. La curva de infectados tiene una forma de campana donde hay un punto máximo que indica el momento más crítico de contagio en la población. Es importante prever este punto máximo porque puede exceder la capacidad del sistema de salud si el número de infectados simultáneos es muy grande.
3. La curva de recuperados siempre aumenta en el tiempo.

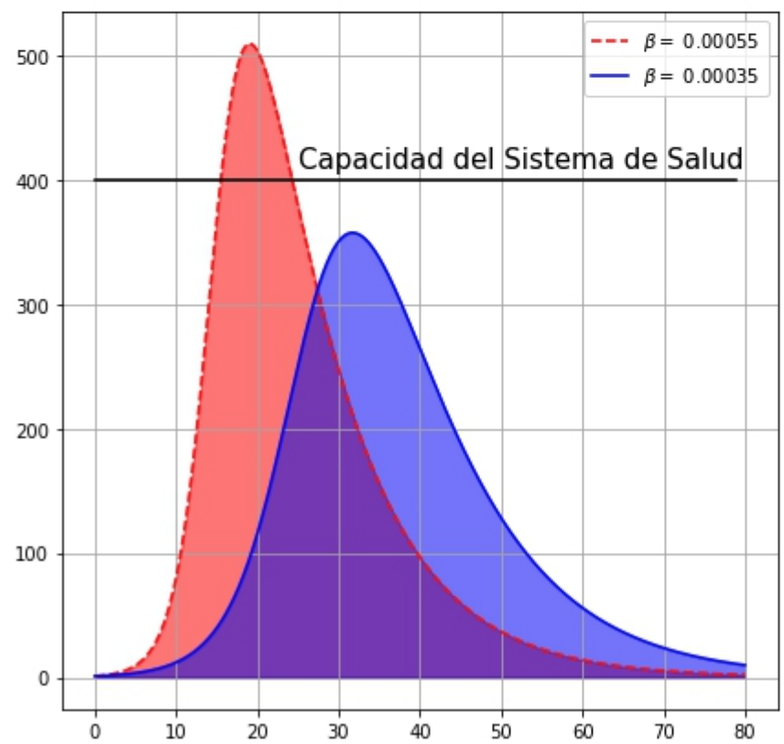
Estas tres condiciones reflejan una situación donde la epidemia eventualmente se desvanece en el tiempo y decimos que la población ha alcanzado lo que se denomina la

**inmunidad de Herd** o inmunidad de rebaño, que es un estado donde los individuos ya se han recuperado porque la gran mayoría ha logrado adquirir inmunidad contra el virus que haya causado la epidemia.

**Cuando los gobiernos dicen que debemos aplanar la curva para no sobrecargar el sistema de salud**, se refieren a la curva de infectados que acabamos de mostrar previamente, sabemos que la medida más popular para controlar esto ha sido mediante la cuarentena y el distanciamiento social y esto se puede modelar de cierta manera a través de la constante de propagación de la epidemia $\beta$. Es decir estas políticas de distanciamiento social se pueden cuantificar con un valor más pequeño de $\beta$, veamos una comparación de cómo un valor menor de $\beta$ afecta el pico máximo de infectados.

[7]

```
plt.figure(figsize=(7,7))

def plot_solution(params, color = 'red', style = '-'):
    init_condition = [1000, 1, 0]
    model = SIR_model(init_condition=init_condition, tmax=80, n = 50000, **params)
    model.run_solver()
    plt.plot(model.t, model.I, linestyle= style, color = color,
             label = r'$\beta =$ {}'.format(model.beta))
    plt.fill_between(model.t, 0, model.I, color = color, alpha = 0.5)

plot_solution(params = {'beta': 0.00055, 'miu': 0.1}, style = '--')
plt.plot(np.arange(0,80), 400*np.ones(len(np.arange(0,80))), '-k')
plt.text(x= 25, y = 410, s = 'Capacidad del Sistema de Salud', fontsize = 15)
plot_solution(params = {'beta': 0.00035, 'miu': 0.1}, color = 'b')
plt.grid(True)
plt.legend()
plt.show()
```




Y con este gráfico final evidenciamos que una reducción en la tasa de propagación tiene como efecto que la curva se vuelve más ancha pero así también el pico máximo de

contagios se reduce, colocamos como ejemplo una recta horizontal que representa la capacidad teórica de un sistema de salud solo para mostrar que la reducción en la tasa de propagación se traduce en ese aplanamiento de la curva que eventualmente implica que no se sature un sistema de salud en consideración.

Para concluir esta clase, es importante anotar que el sistema SIR es solo otra aproximación más a una situación de epidemia real. Estas ecuaciones no pretenden predecir con total certeza una situación dada como la que vivimos actualmente del Covid-19, pero la intuición que podemos desarrollar con ellas nos permite entender aspectos fundamentales de la dinámica de estos sistemas y así entender el efecto de políticas de distanciamiento social. En la próxima clase veremos otros detalles interesantes de este sistema y una mejora adicional que dejaremos como proyecto final del curso.

El notebook de esta clase lo puedes encontrar en este link.