estabilidad

June 22, 2018

1 Ecuación logística

Sea

$$x_{n+1} = \mu x_n(1 - x_n)$$

la ecuación logística, la familia de funciones es dada por $F_\mu(x) = \mu x(1 - x)$, entonces

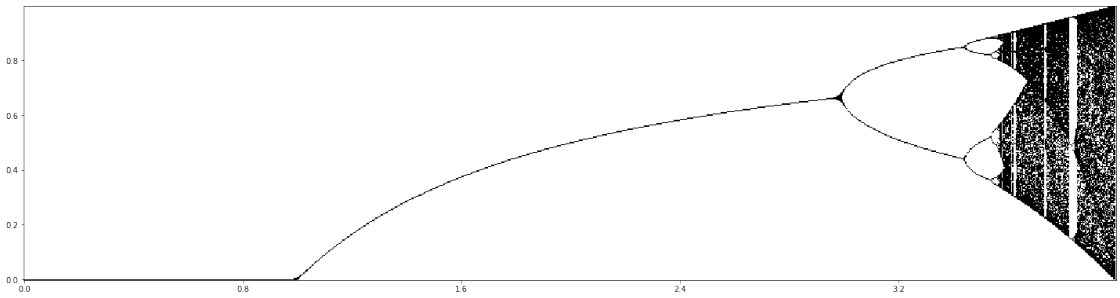
$$x_{n+1} = F_\mu(x_n)$$

Analizaremos las bifurcaciones partiendo el espacio en una grilla de resolución $4/n$ para el eje x y $1/n$ para el y, ya que la ecuación está definida en $[0, 4] \times [0, 1]$. En ese caso encenderemos todos los $(\mu, F_\mu(x))$ para un x dado.

```
In [84]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline

In [85]: def f(mu, x):
             return np.multiply(np.multiply(mu,x),(1-x))

         def fn(mu, x, n):
             if n == 0:
                 return f(mu, x)
             return fn(mu, f(mu, x), n-1)

         n = 250

In [98]: mus = np.linspace(0,4,n*4)

In [99]: def mu_fmu(x0, mu, n):
             return (mu, fn(mu, x0, n))

         def linspace_to_mcoords(array, xmin, samples):
             # mover las muestras y hacerlas un número entero
             return ((array - xmin)*samples).astype(int)

         def mcoords_to_linspace(array, xmin, samples):
             # hacer lo opuesto
             return ((array/samples)+xmin)
```

```
In [100]: # hacer la matriz
          mat = np.zeros((n, n*4))

          for i in range(100, 500):
              # calcular los f_mu(x_0)
              fmus = fn(mus, 1/2, i)

              # ponerlos en los índices de la matriz
              matfmus = linspace_to_mcoords(fmus, 0, n)

              for i in range(n*4):
                  mat[matfmus[i], i] = 1
```

```
In [101]: fig = plt.figure(figsize=(25,7))
          ax = fig.add_subplot(111)
          ax.imshow(mat, cmap='Greys',  interpolation='nearest', origin='lower')
          _ = ax.set_xticklabels(mcoords_to_linspace(ax.get_xticks(), 0, n))
          _ = ax.set_yticklabels(mcoords_to_linspace(ax.get_yticks(), 0, n))
          fig.savefig('diagBifur.pdf', bbox_inches='tight')
          plt.show()
```



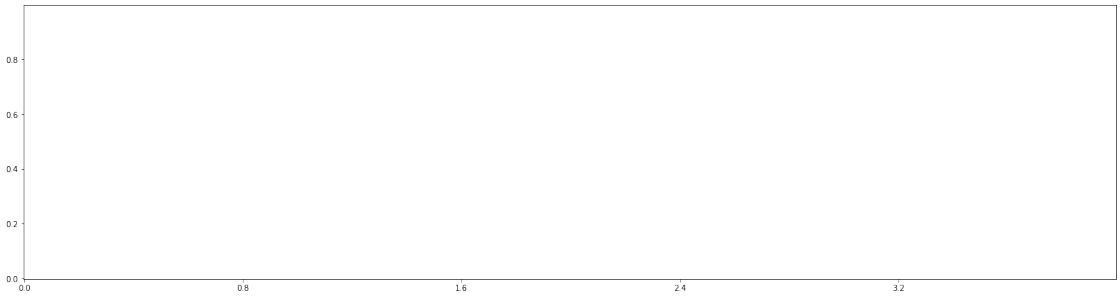

```
In [11]: import matplotlib.animation as animation
         from matplotlib import animation, rc
         from IPython.display import HTML
```

1.1 Animación sobre el eje x

```
In [102]: fig = plt.figure(figsize=(25,7))
          ax = fig.add_subplot(111)
          im = ax.imshow(np.concatenate((mat[:,0:1],np.zeros((n,n*4-1))), axis=1), cmap='Greys
          _ = ax.set_xticklabels(mcoords_to_linspace(ax.get_xticks(), 0, n))
          _ = ax.set_yticklabels(mcoords_to_linspace(ax.get_yticks(), 0, n))

          def animate(i):
              A = np.concatenate((mat[:,0:i],np.zeros((n,n*4-i))), axis=1)
              im.set_data(A)# update the data
```

```
        return im,


    # Init only required for blitting to give a clean slate.
    def init():
        return im,

    ani = animation.FuncAnimation(fig, animate, np.arange(1, n*4), init_func=init,
                                  interval=25, blit=True)
    HTML(ani.to_html5_video())
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-102-4bd2c53c052a> in <module>()
     15     return im,
     16
---> 17 ani = animation.FuncAnimation(fig, animate, np.arange(1, n*4), init_func=init,
     18                                 interval=25, blit=True)
     19 HTML(ani.to_html5_video())


NameError: name 'animation' is not defined
```

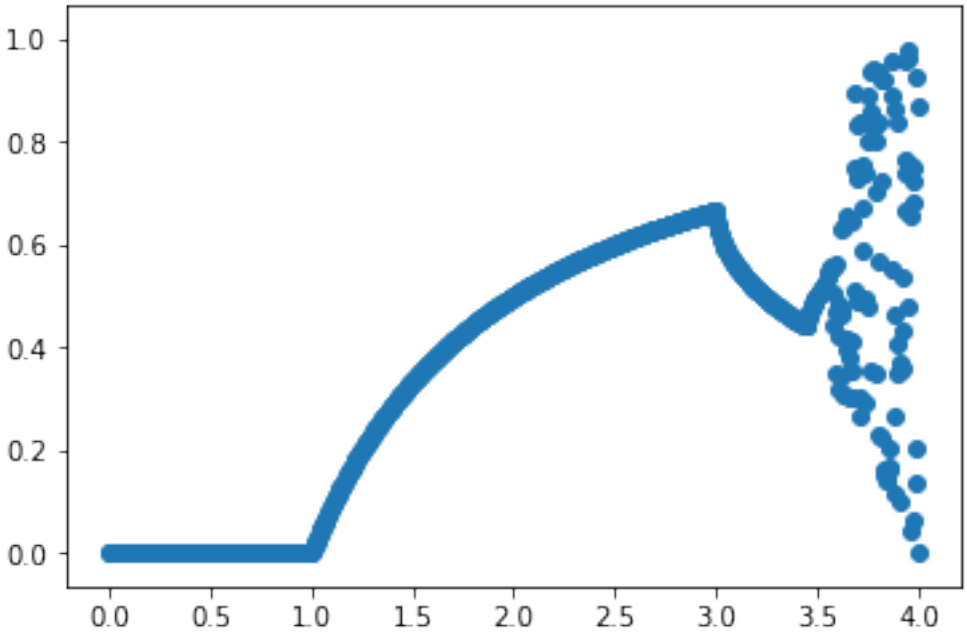


1.2 Animación de iteración de la función

```
In [95]: fig = plt.figure()
         ax = fig.add_subplot(111)
         im = ax.scatter(mus, fn(mus, 1/2, 0))

         def animate(i):
             mat = np.zeros((n, n*4))
             # calcular los f_mu(x_0)
```

```
        fmus = fn(mus, 1/2, i)
        data = np.hstack((mus[:,np.newaxis], fmus[:,np.newaxis]))
        im.set_offsets(data)
        return im,


    # Init only required for blitting to give a clean slate.
    def init():
        return im,

    ani = animation.FuncAnimation(fig, animate, np.arange(100,500), init_func=init,
                                  interval=25, blit=True)
    HTML(ani.to_html5_video())
```

Out[95]: <IPython.core.display.HTML object>

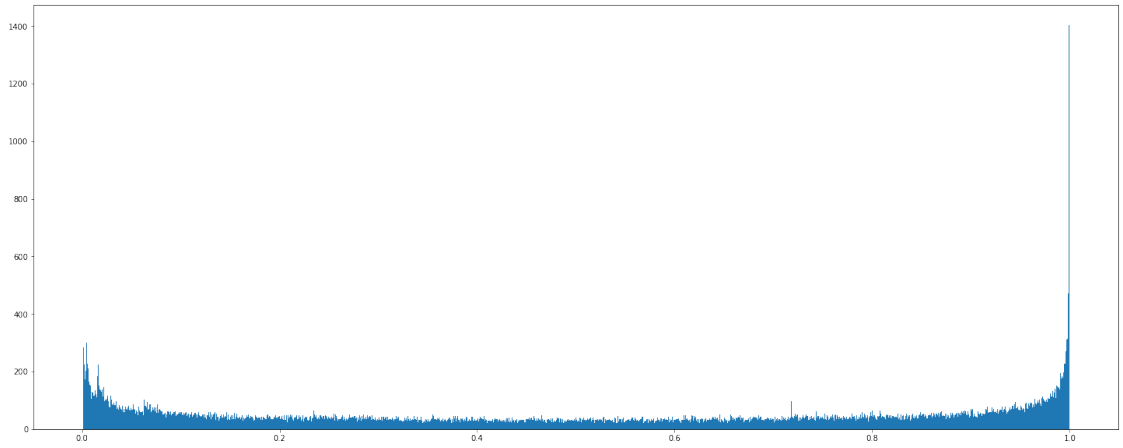


1.3 Generador de números aleatorios

In [142]:
```
mu = 3.999
x0 = 1/2

x = fn(mu, x0, 400)
#x = x0
nums = []
for i in range(100000):
```

```
            x = f(mu, x)
            nums.append(x)
```

```
In [147]: fig = plt.figure(figsize=(25,10))
          ax = fig.add_subplot(111)
          ax.hist(nums, bins=2000)
          plt.savefig("dist.pdf")
          plt.show()
```




```
In [151]: nums[-10:]
```

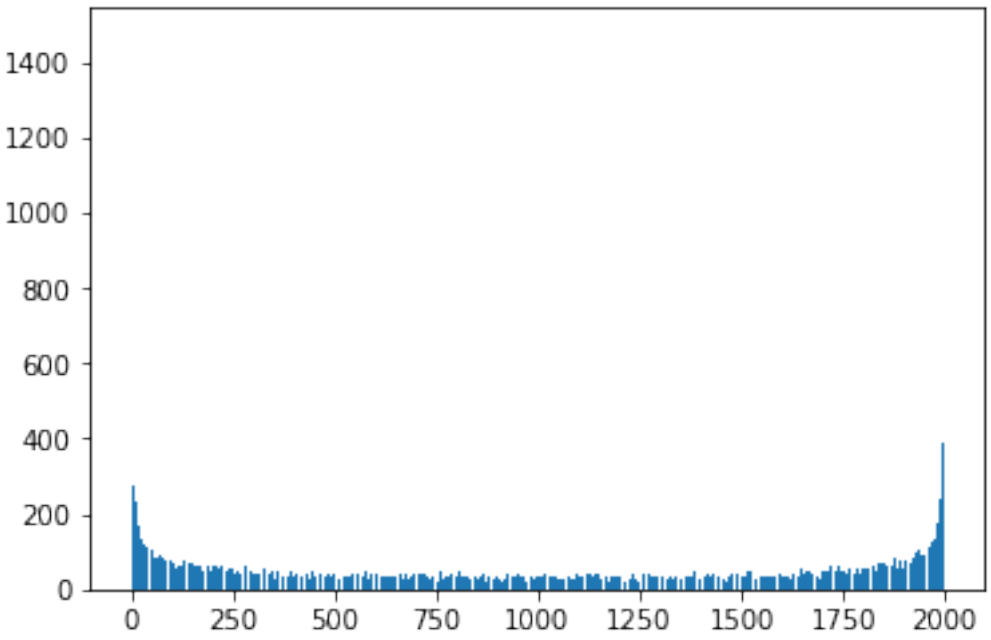
```
Out[151]: [0.9613495492744186,
           0.14858921694377886,
           0.5059153357457021,
           0.9996100702032594,
           0.0015587212282252552,
           0.006223610173815383,
           0.024733322524028843,
           0.09646221953852391,
           0.3485418817011619,
           0.9080146931670994]
```

Usando la prueba de χ^2 podemos determinar la probabilidad de que esta distribución es tomada de una distribución uniforme

```
In [16]: import scipy.stats
```

```
In [41]: # make a histogram
         (hist, bin_edges) = np.histogram(nums, bins=2000)
```

```
In [42]: plt.bar(range(len(hist)),hist)
         plt.show()
```




```
In [49]: observed = hist
         expected = np.ones(2000)*100000/2000

In [63]: chi_squared_stat =  (((observed-expected)**2)/expected).sum()
         crit = scipy.stats.chi2.ppf(q = 0.95, df = 1999)
         p_value = 1 - scipy.stats.chi2.cdf(x=chi_squared_stat,  df=1999)
         print(chi_squared_stat)
         print(crit)
         print(p_value)

115732.16000000002
2104.128222359781
0.0


In [61]: scipy.stats.chisquare(f_obs= observed,   # Array of observed counts
                               f_exp= expected)   # Array of expected counts

Out[61]: Power_divergenceResult(statistic=115732.16000000002, pvalue=0.0)

In [141]: fn(4,1/32,20)

Out[141]: 0.7030706048273905
```