Зайцы.

Логистическое отображение.

Рассматриваемая система описывает изменение какой-либо величины по закону

$$X_{n+1} = r \cdot X_n(1 - X_n)$$

Бифуркационная диаграмма.

Для данной системы можно построить бифуркационную диаграмму показывающую, как себя ведёт система при разных значениях r.

Создание диаграммы

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from scipy import *
from pylab import *
from scipy import stats

# f(x)
f = lambda x,r : r * x * ( 1 -x )
# набор r
rlist = linspace( 0, 4, 1000)
# начальная популяция зайцев
X = [ 0.3 * ones_like(rlist), ]
# эволюция за N шагов
for i in arange(0,2000):  X += [ f(X[ -1], rlist) , ]
# берём последние N
X = hsplit( vstack(X[-1000:]), rlist.size)
# Разбиваем на участки по вертикали (чем больше точек, тем ярче будет участок)
H = map( lambda Z : stats.histogram( Z, defaultlimits=(0,1), numbins=300 )[0],X)
# нормируем по весу и инвертируем цвет (чтобы линия была чёрной а не белой)
H = map( lambda Z : 1-Z/Z.max(), H )
figure( figsize=(9, 6), dpi=100 )
imshow( rot90(vstack(H)), aspect = 'auto' , extent = [0, 4, 0, 1])
bone() # переводим цвет в чб
xlabel('r')
ylabel(r'$X_{n \rightarrow \infty}$')
```

```
savefig('Bifurcation.png')
```

Результат

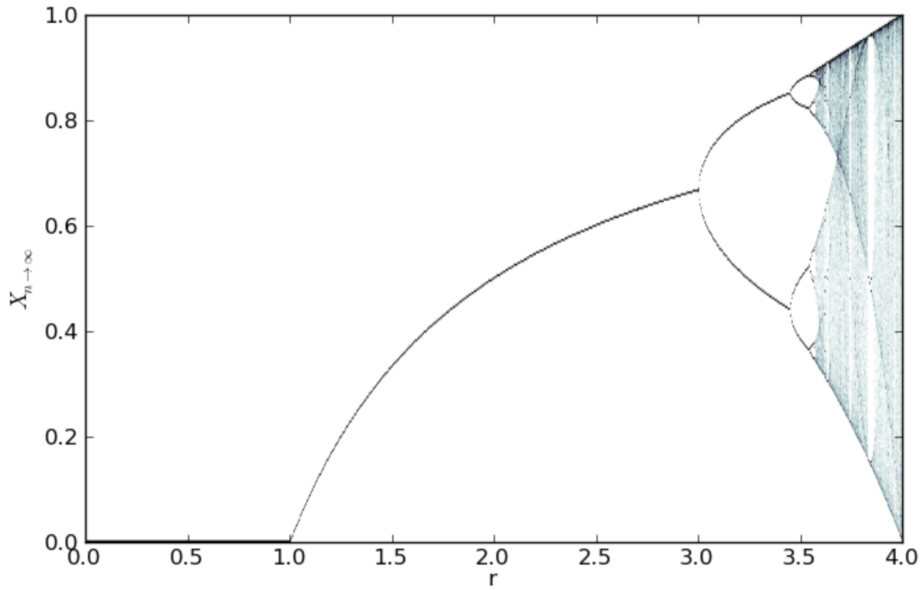


Figure 1: Бифуркационная диаграмма

Интерпретация

При значениях r от 0 до 1, точка X=0 была аттрактором (все зайцы быстро умирали). При r=1 точка X=0 перестаёт быть устойчивой, но остается неподвижной. В то время, как популяция зайцев стремится к аттрактору (устойчивая неподвижная точка) при значении $\frac{r-1}{r}$. При r больше 3 численность популяции будет колебаться между двумя значениями. При $r > 1 + \sqrt{6}$ численность будет колебаться между 4 значениями и так дальше: 8, 16, 32. Так же существуют окна периодичности в которых популярность колеблется между 3 значениями (окно примерно в 3.83), потом между 6, потом 12 и так далее.

Фрактальная структура

Если рассмотреть в увеличенном масштабе часть диаграммы, то можно увидеть её фрактальную структуру. Для примера приведён участок r от 3.5 до 3.7 и x от 0.3 до 0.7

Фурье.

Можно построить Фурье преобразования для нескольких значений r.

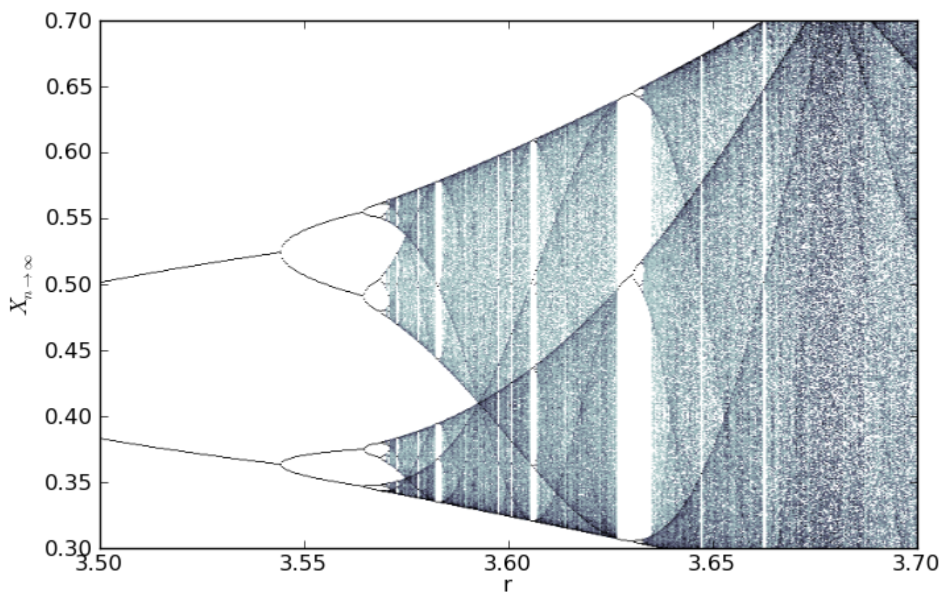
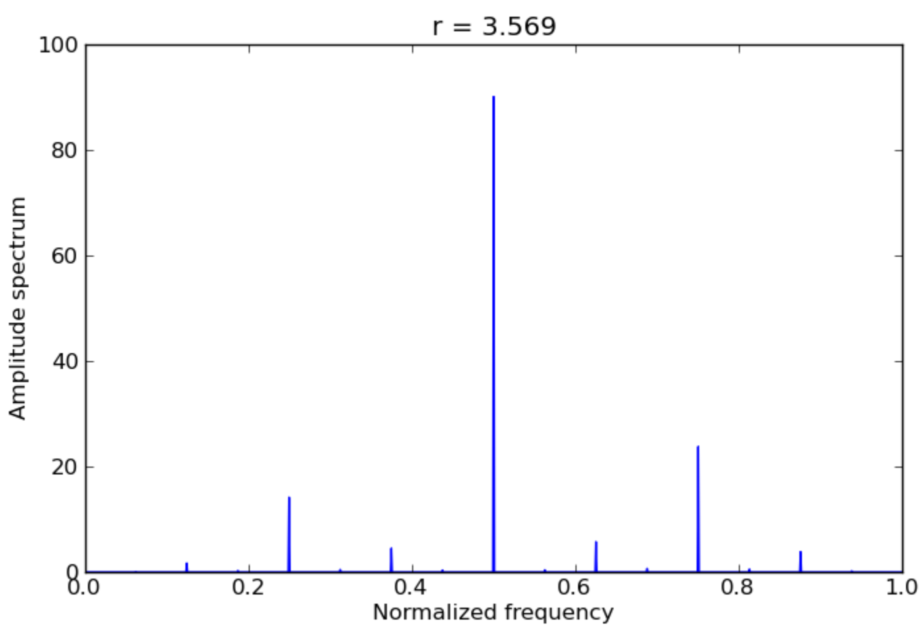


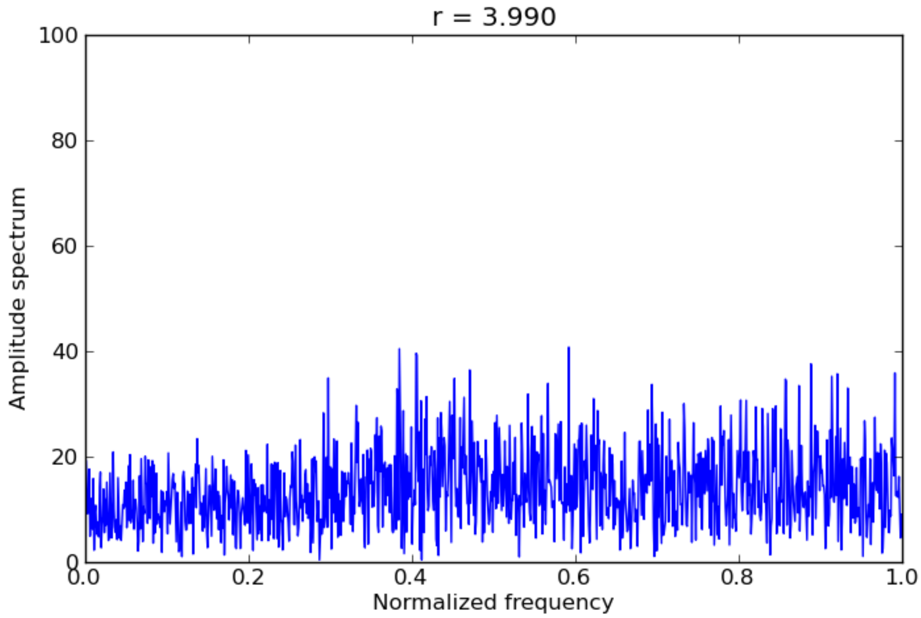
Figure 2: Участок бифуркационной диаграммы

Код для создания спектра

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from scipy import *
from pylab import *
from scipy import fftpack

N = 2048

# f(x)
f = lambda x,r : r * x * ( 1 -x )
# набор r
rlist = linspace( 2.99, 3.99, 20)
# начальная популяция зайцев
X = [ 0.2 * ones_like(rlist), ]
# эволюция за N шагов
for i in arange(0,N+1000):  X += [ f(X[ -1], rlist) , ]

X = hsplit( vstack(X[-N:]), rlist.size)

def fourier(Z):
```

```
    Z = hstack(Z)
    Z = fft(Z)
    Z = abs(Z[1:len(Z)/2])#**2
    return Z

F = map ( fourier, X)

figure( figsize=(8, 5), dpi=130 )
for i in range(len(rlist)):
    filename = str('%04d' % i) + '.png'
    print filename
    title('r = %.3f' % rlist[i])
    xlabel('Normalized frequency')
    ylabel('Amplitude spectrum')
    fn = np.linspace(0.0,1.0, len(F[i]))
    plot(fn, F[i])
    ylim([0,100])
    plt.savefig(filename, dpi=100)
    clf() # clear figure
```





Организованная самокритичность