Trabalho de dinâmica não linear e Caos

Bruno Mantovani Czajkowski, GRR20182694

15 de dezembro de 2021

Todas as figuras foram feitas usando Python (versão 3.7.12), através das bibliotecas matplotlib (versão 3.2.2) e numpy (versão 1.19.5). O código para todos os exercícios pode ser acessado e rodado num notebook do Colab: `https://colab.research.google.com/drive/12YNwKZh267spLWS_aJc_MOxRfXvFFbJX?usp=sharing`.

Caso prefira, é possível baixar o programa como .py ou .ipynb(notebook) no repositório do GitHub: `https://github.com/brunomc3/trabalho_CF077`, dentro da pasta codigos. Além disso, ao longo do pdf é colocado alguns exemplos do código utilizado em cada exercício.

Exercício 2, Cap 2

a)

Para obter as órbitas do mapa logístico para diferentes valores de r, inicialmente se escolhe um ponto x_0 e o tamanho da órbita n. Então para cada valor de n se plotta $f^{[n]}(x_0)$.

Como a bacia de atração do mapa logístico para as órbitas estáveis é todo o espaço $[0, 1]$, o valor de x_0 não deve interferir no comportamento assintótico das órbitas periódicas.

Para montar os gráficos foram usados os parâmetros $n = 200$, $x_0 = 0.34$. Uma parte do código utilizado é mostrado no Código 1 A Figura 1 mostra o resultado do plot.

Código 1: Parte do código para o Exercício 2.2

```
#Define a função logística e uma função para obter os
    pontos de uma órbita de tamanho n e condição
    inicial x0

def logistico(x,r):
  return r*x*(1-x)

def orbita(x0,n,r):
  y=x0
  resultado=np.zeros(n)
  resultado[0]=x0
  for i in range(1,n):
    y=logistico(y,r)
    resultado[i]=y
  return resultado

#Define os valores de r propostos
rs=[[0.5,2.0],[3.2,3.54]]

#Determina-se quantos pontos utilizar e o ponto inicial
     x0

n=200
x0=0.34

#Cria uma figura com matriz de eixos
fig,ax=plt.subplots(nrows=2,ncols=2,constrained_layout=
    True,sharex=True, sharey=True,figsize=(10,10))
colors=[['b','c'],['r','g']]
fig.suptitle('x0 = %.2f'%x0,fontsize=20)

#itera sobre os valores de r propostos e faz os gráfico
     das órbitas.

for j in range(2):
  for i in range(2):
    r=rs[i][j]
    ax[i][j].scatter(range(n),orbita(x0,n,r),label="r =
         %.2f"%r,color=colors[i][j],s=20)
    ax[i][j].legend(fontsize=16)
    fig.text(0.5, -0.05, 'n', ha='center',fontsize=18)
    fig.text(-0.05, 0.5, '$x_{n}$', va='center',
        rotation='vertical',fontsize=18)

fig.savefig('2_2_a_1.png',bbox_inches='tight',dpi=200)
```

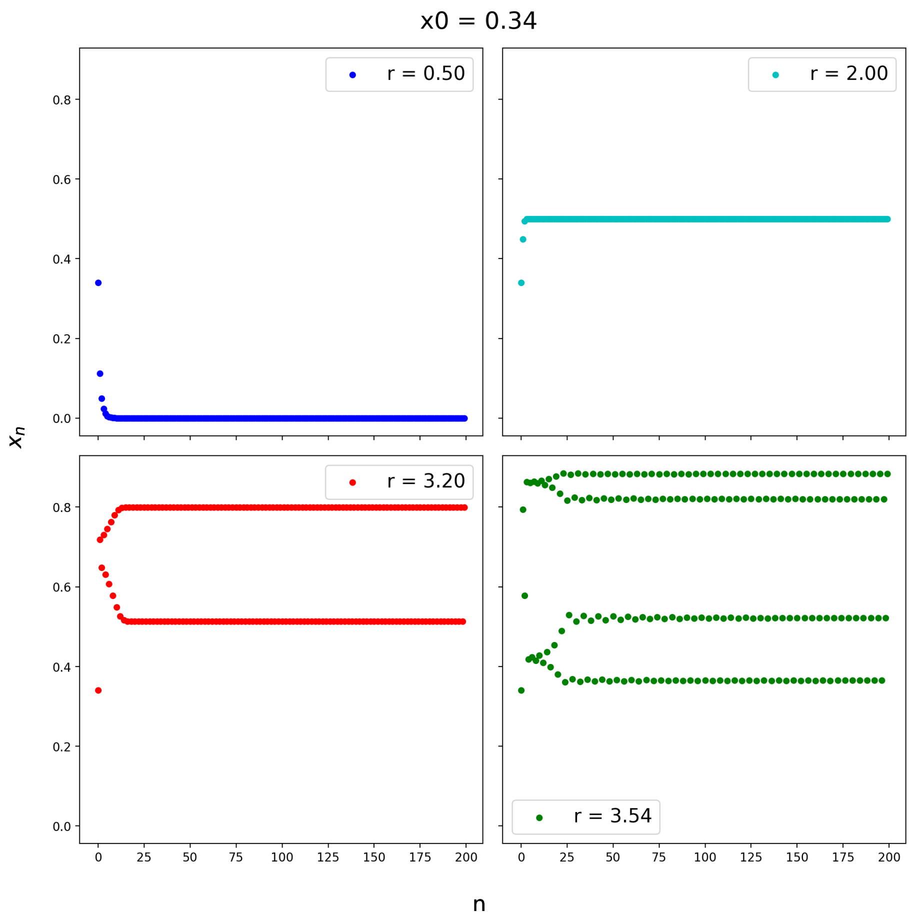

Figura 1: Órbitas para os diferentes valores de r propostos, note que após determinado período transiente, todas órbitas ficam períodicas

b)

O gráfico de escada é uma maneira de olhar o comportamento dos pontos de uma órbita de um mapa unidimensional (aqui o logístico)

O procedimento realizado é o seguinte:

1. Escolhe-se x_0 o ponto inicial, r o parâmetro do mapa e n o número de iterações do mapa.

2. Plota-se as curvas $y = x$ e $y = f(x)$ da função identidade e do mapa em questão

3. Marca-se o ponto P_0=(x_0,0)

4. define-se um contador $c = 0$

5. enquanto $c \leq n$ faz-se:

   (a) Traça-se uma reta vertical a partir de P_c até ela interceptar a função $y = f(x)$ obtendo-se o ponto Q_c=(x_c,$f(x_c)$)

   (b) Traça-se uma reta horizontal a partir de Q_c até ela interceptar a função $y = x$ obtendo-se o ponto P_{c+1}=($f(x_c)$,$f(x_c)$).

   (c) c=c+1

6. plota-se os pontos P e Q ligando eles por retas

O resultado do procedimento acima para $x_0 = 0.95$, $n = 100$ e para os valores de r propostos está na Figura 2.

Além disso, parte do código está no Código 2

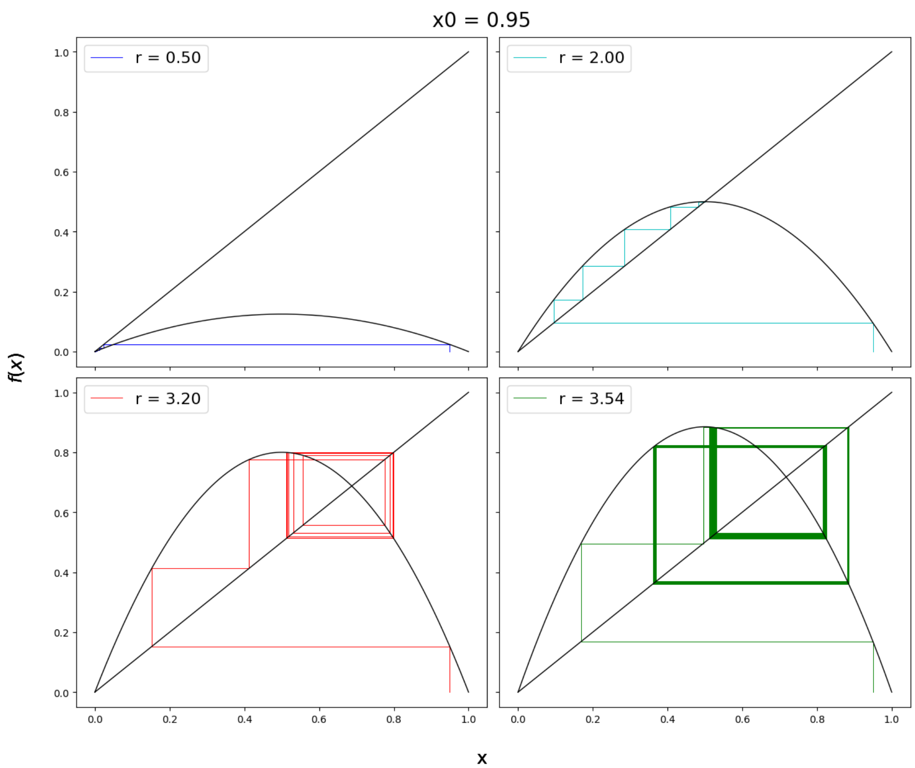

Figura 2: Gráfico escada para diferentes valores de r, com $x_0 = 0.95$ e $n = 100$.

Código 2: Parte do código para diagrama de escada

```
#Define a função escada para um dado x0, n e r
def escada(x0,n,r):
  y=logistico(x0,r)
  points=np.array([x0,0])
  points=np.vstack([points,[x0,y]])
  for i in range(n-1):
    x=y
    points=np.vstack([points,[x,y]])
    y=logistico(y,r)
    points=np.vstack([points,[x,y]])
  return points
#define condições inicias e quantidade de iterações
n=100
x0=0.95
vector=np.vectorize(logistico)
#Cria figuras
fig,ax=plt.subplots(nrows=2,ncols=2,constrained_layout=
    True,sharex=True, sharey=True,figsize=(12,10))
colors=[['b','c'],['r','g']]
x=np.linspace(0,1,1000)
fig.suptitle('x0 = %.2f'%x0,fontsize=20)
for j in range(2):
  for i in range(2):
    r=rs[i][j]
    pontos=escada(x0,n,r)
    pontos_x=pontos.transpose()[0]
    pontos_y=pontos.transpose()[1]
    y=vector(x,r)
    ax[i][j].plot(pontos_x,pontos_y,"-",c=colors[i][j],
       lw=0.7,label="r = %.2f"%r)
    ax[i][j].plot(x,x,c='k',lw=1)
    ax[i][j].plot(x,y,c='k',lw=1)
    ax[i][j].legend(fontsize=16)
    fig.text(0.5, -0.05, 'x', ha='center',fontsize=18)
    fig.text(-0.05, 0.5, '$f(x)$', va='center',
       rotation='vertical',fontsize=18)

fig.savefig("2_2_b_1",bbox_inches="tight",dpi=200)
```

Exercício 8, Cap 2

Para se fazer os gráficos de bifurcação é necessário plotar para cada valor de r na região desejada qual o regime assintótico das órbitas de um determinado mapa (aqui o logístico). Para isso segue-se o seguinte algoritmo

1. Escolhe-se uma grande quantidade de valores de r, não necessariamente espaçados uniformemente, no intervalo $[r_{min}, r_{max}]$ desejado.

2. determina-se um valor de n_{lim} relativamente grande (da ordem de 500) que é usado para obter uma órbita de tamanho n_{lim} do mapa.

3. Escolhe-se um valor de n_p de quantos pontos por órbita (e,consequentemente por valor de r) serão usados.

4. Então, para cada valor de r, faz-se:

    (a) escolhe-se um x_0 aleatório no intervalo [0,1]

    (b) Obtém-se a órbita de tamanho n_{lim} iterando o mapa logístico aplicado no ponto x_0 com parâmetro r

    (c) Pega-se os últimos n_p pontos da órbita obtida no item anterior; os quais, se n_{lim} for suficientemente grande, são pontos no regime assintótico

5. Plota-se para cada valor de r os n_p pontos do regime assintótico

Assim, para todos os itens nesse problema foi utilizado o algoritmo acima para diferentes n_p, n_{lim} e valores de r

a)

Aqui, como devemos plotar o diagrama de bifurcação para todo o intervalo $[r_{min}, r_{max}]$=[0,4] mas a maior parte da dinâmica interessante ocorre após $r = 3$ ou $r = 3.5$, os pontos r não estão igualmente espaçados no intervalo. Escolheu-se 1000 pontos igualmente espaçados no intervalo [0,3], 3000 no intervalo [3,3.5] e 10000 no intevalo [3.5, 4.0]. Além disso tomou-se n_p=2 e n_{lim}=500. O gráfico está na Figura 3.

No Código 3 é mostrado o código para esse item. Os itens seguintes desse exercício são análogos, alterando-se a distribuição do r, n_p e n_{lim} conforme descrito abaixo. O código completo pode ser acessado nos links do início desse pdf.

Código 3: Parte do código para diagrama de birfucação

```
#Escolhe como dividir os r
r1=np.linspace(0,3,1000)
r2=np.linspace(3,3.5,3000)
r3=np.linspace(3.5,4,10000)
rs=np.concatenate([r1,r2,r3])
n_pontos=2

#Define função que encontra os pontos assíntotos
def assintota(x0,r,iteradas,n_pontos=3):
  return orbita(x0,iteradas,r)[-n_pontos:]

pontos=np.array([])
rf=np.array([])
for j,r in enumerate(rs):
  x0=random.random()
  pontos=np.concatenate([pontos,assintota(x0,r,500,
      n_pontos=n_pontos)])
  rf=np.concatenate([rf,np.full(n_pontos,r)])

#Cria a figura
fig,ax=plt.subplots(figsize=(10,10))
ax.scatter(rf,pontos,s=0.3,c='k')
ax.set_xlabel('r')
ax.set_ylabel('y')
fig.savefig("2_8_a_1",bbox_inches='tight',dpi=200)
```

Além do mapa no intervalo todo, queremos também estudar o diagrama de bifurcação próximo de $r_\infty \approx 3.5699$, vamos considerar dois casos de intervalos centrados em 3.5699, da forma $[3.5699-\epsilon,\ 3.5699+\epsilon]$. Faremos para $\epsilon \in \{0.01, 0.0001\}$.

Aqui será usado 10000 valores de r igualmente espaçados no intervalo de interesse. r_{lim}=500 e n_p=2. Os gráficos estão nas Figuras 4 e 5

b)

Para todas ampliações nesse item utilizou-se n_p=5, n_{lim}=500 e 10000 valores de r igualmente espaçados no intervalo apropriado. As Figuras 6,7 e 8 mostram os resultados.

Perceba que para todos esses gráficos, quando as órbitas periódicas começam, a partir de um determinado r, abruptamente se encerra um regime caótico. E, em seguida, as órbitas estáveis tendem a uma cascata de duplicação que converge

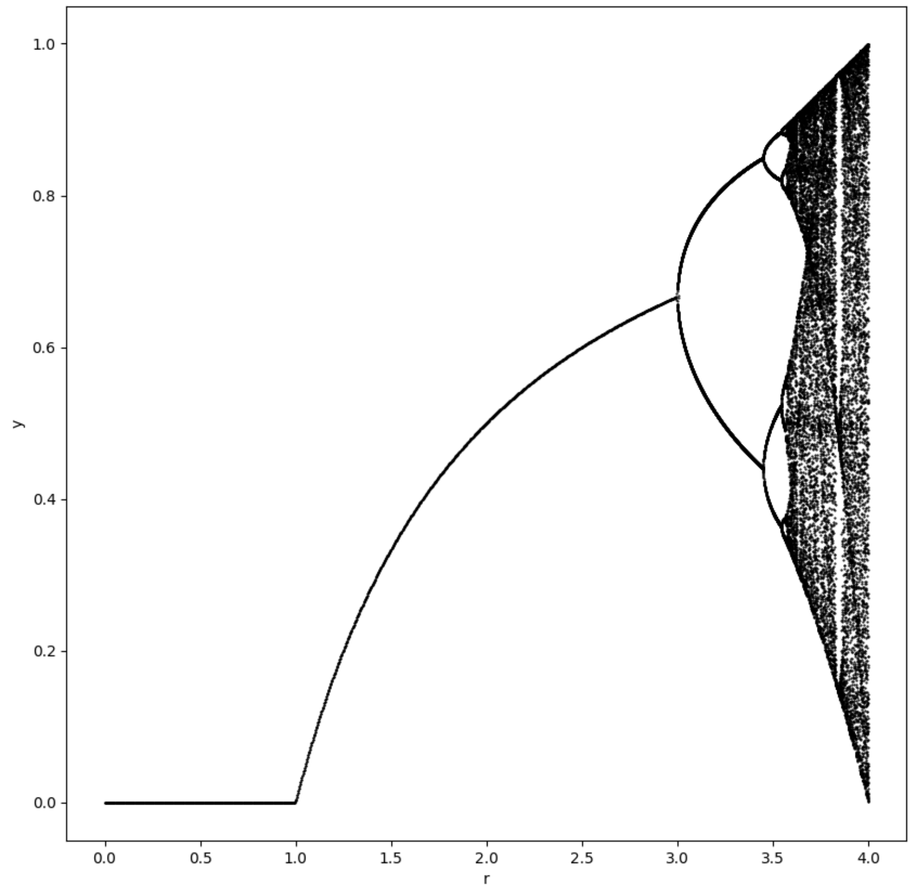

Figura 3: Gráfico de bifurcação para todo o espaço $r \in [0,4]$; com n_p=2 e n_{lim}=500.

para um determinado r_{max}. Para um valor de r imediatamente após esse ponto de convergencia recomeça o comportamento caótico.

c)

Agora analisamos o diagrama de bifurcações próximo de 4. Nesse caso, consideramos n_p=10, n_{lim}=500 e 10000 pontos de r no intervalo [3.95,4.00]. Note que apesar de haver janelas periódicas, o regime é predominantemente caótico.

A figura 9 mostra esse diagrama

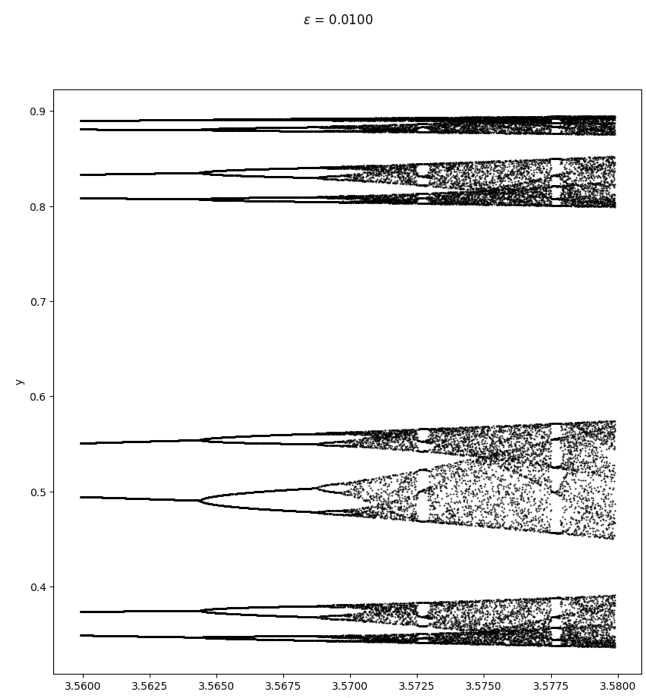

Figura 4: Gráfico de bifurcação para $r \in [r_\infty - 0.010,\ r_\infty + 0.010]$; com n_p=2 e n_{lim}=500. Perceba que antes de r_∞ há órbitas estáveis bem definidas, enquanto depois existe um processo intermitente de órbitas estáveis e caos

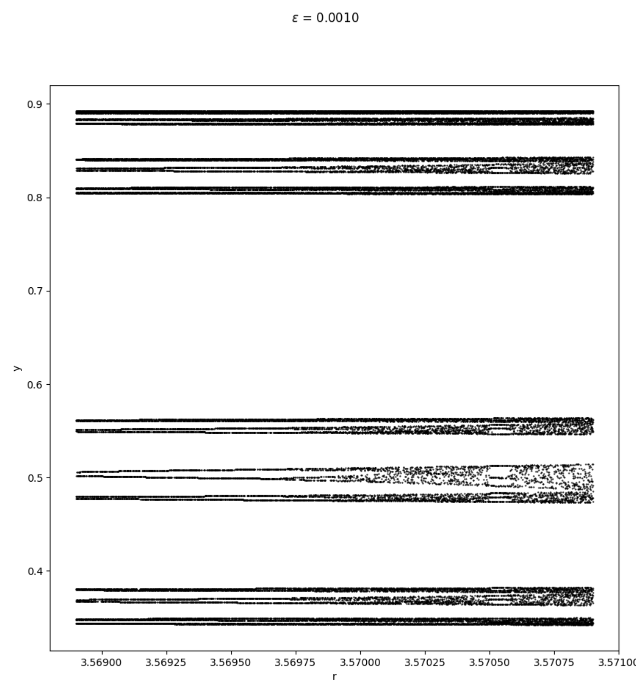

Figura 5: Gráfico de bifurcação para $r \in [r_\infty - 0.0010,\ r_\infty + 0.0010]$; com n_p=2 e n_{lim}=500. A figura mostra de maneira mais precisa como ocorre a passagem de órbitas estáveis para um estado caótico quando $r > r_\infty$

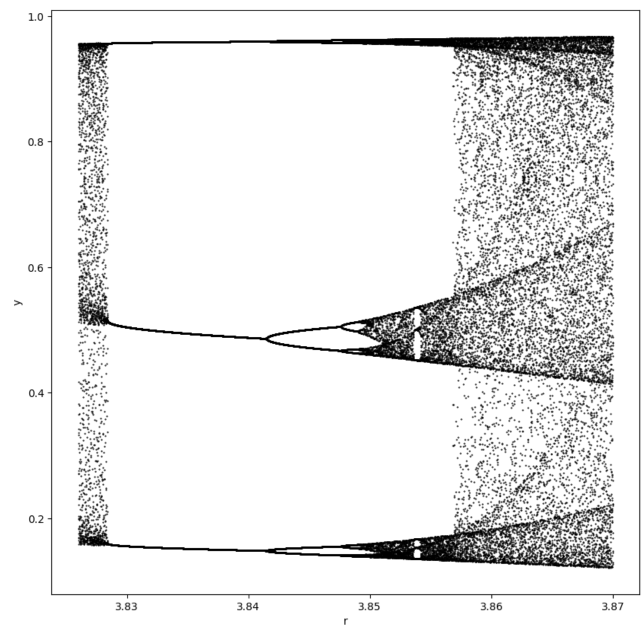

Figura 6: Gráfico de bifurcação ampliado para órbita de período 3, $r_{min} = 3.826$ e $r_{max} = 3.870$

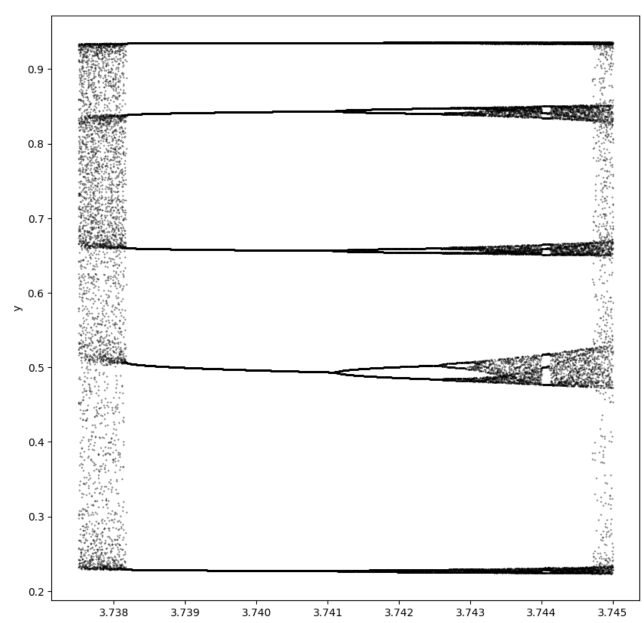

Figura 7: Gráfico de bifurcação ampliado para órbita de período 5, $r_{min} = 3.7375$ e $r_{max} = 3.7450$

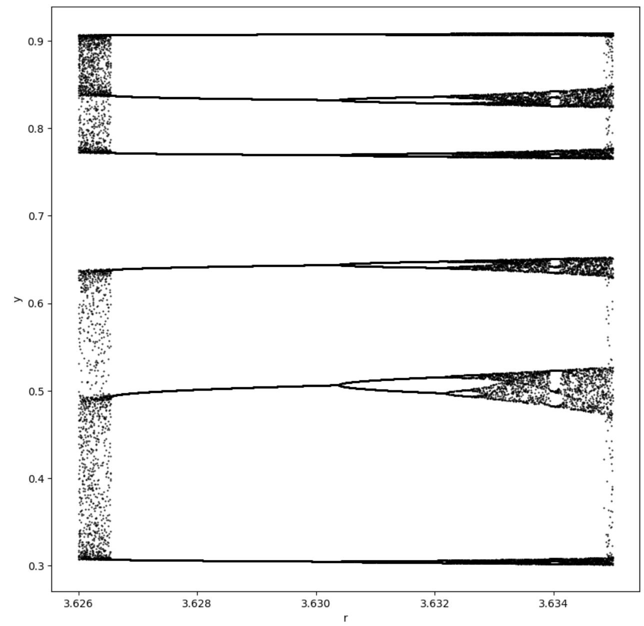

Figura 8: Gráfico de bifurcação ampliado para órbita de período 6, $r_{min} = 3.626$ e $r_{max} = 3.635$

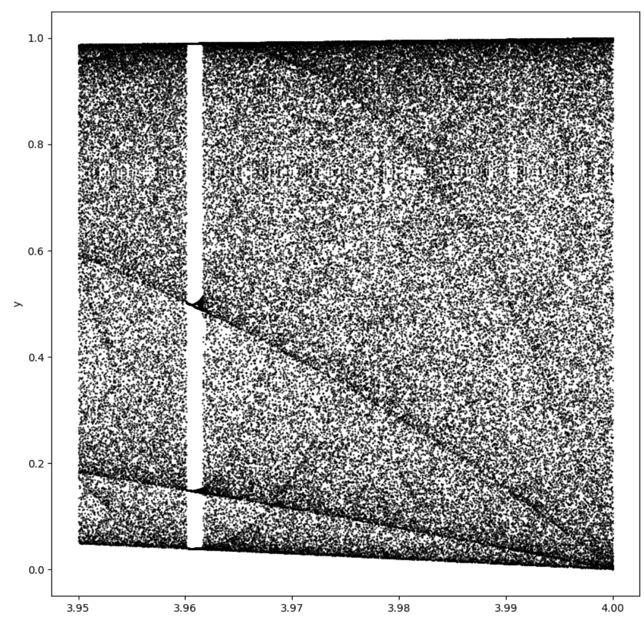

Figura 9: Gráfico de bifurcação ampliado para r entre $r_{min} = 3.95$ e $r_{max} = 4.00$

Exercício 1, Cap 3

Sabemos que o expoente de Lyapunov é calculado por

$$\lambda(x_0) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \log|f'(x_n)| \tag{1}$$

define-se ainda o trucamento em N do limite acima

$$\lambda_N(x_0) = \lambda(x_0, N) = \frac{1}{N} \sum_{n=0}^{N-1} \log|f'(x_n)| \tag{2}$$

onde x_n é o n-ésimo ponto da órbita. Assim, para calcular o expoente para o mapa de Ulam fez-se o seguinte procedimento: Escolhe-se um valor x_0 no intervalo [0,1] e um valor N e utiliza-se a Equação 2; sabemos que quanto maior N mais próximo do valor teórico esperado ($\log 2$) para qualquer x_0. Então, aumentamos N progressivamente para verificar como ocorre essa aproximação para o valor limite.

A função que calcula o expoente de Lyapunov truncado é apresentada no Código 4

Código 4: Função da derivada do mapa de Ulam e do expoente de Lyapunov

```
#Define-se o mapa derivada do logístico
def derivada_logistico(x,r=4):
  return r-2*r*x
derivada_vector=np.vectorize(derivada_logistico)

#Define-se o expoente de Lyapunov truncado em n
def expoente_lyapunov(x0,n,r=4):
  pontos=orbita(x0,n,r)
  derivadas=derivada_vector(pontos,r)
  return 1/n*np.log(np.abs(derivadas)).sum()
```

Calcula-se o erro entre o resultado e $\log 2$ usando que $erro = \frac{|\lambda_N - \log 2|}{\log 2}$.

Foram feitos esses cálculos para N de diferentes ordens de grandeza e para $x_0 = 0.13$ e $x_0 = 0.65$, os resultados estão representados respectivamente nas tabelas 1 e 2

Veja que, nesses casos de condições iniciais, por exemplo, para termos valores numéricos razoavelmente próximos do teórico, com erros de no máximo da ordem de 1×10^{-3}, é necessário considerar N com ordem de grandeza de milhar, $N \geq 1000$.

N	λ_N	erro
1	1.08518927	5.7×10^{-1}
5	0.77226088	1.1×10^{-1}
1×10^1	0.73206494	5.6×10^{-2}
1×10^2	0.68721436	8.6×10^{-3}
1×10^3	0.69339347	3.6×10^{-4}
1×10^4	0.69279372	5.1×10^{-4}
1×10^5	0.69315115	5.7×10^{-6}
1×10^6	0.69314743	3.6×10^{-7}

Tabela 1: Tabela do cálculo do expoente de Lyapunov para diferentes valores de truncamento N, com $x_0 = 0.13$ e o erro em relação ao valor teórico.

N	λ_N	erro
1	0.18232156	7.4×10^{-1}
5	0.47448846	3.2×10^{-1}
1×10^1	0.67979556	1.9×10^{-2}
1×10^2	0.67170553	3.1×10^{-2}
1×10^3	0.69297186	2.5×10^{-4}
1×10^4	0.69315020	4.4×10^{-6}
1×10^5	0.69314737	2.7×10^{-7}
1×10^6	0.69314658	8.7×10^{-7}

Tabela 2: Tabela do cálculo do expoente de Lyapunov para diferentes valores de truncamento N, com $x_0 = 0.65$ e o erro em relação ao valor teórico.

Exercício 2, Cap3

Para calcular o diagrama de bifurcação de Luapunov é realizado o seguinte algoritmo

1. Escolhe-se uma quantidade grande de valores de r no intervalo desejado
2. Fixa-se um N que será o número de iterações no cálculo do expoente de Lyapunov
3. Para cada r do conjunto escolhido faz-se:
   (a) escolhe-se um valor de x_0 aleatoriamente entre 0 e 1
   (b) calcula-se $\lambda_N(x_0)$ dado pela Equação 2
4. Plota-se o gráfico de λ por r

Fez-se esse procedimento escolhendo 10000 valores de r igualmente espaçados no intervalo de interesse e tomando $N = 1000$. O cálculo foi feito para $r \in [0, 4]$ e $r \in [3, 4]$ os gráficos estão respectivamente nas Figuras 10 e 11

Para $r \in [0, 4]$ também é colocado parte do código envolvido no Código 5

Código 5: Parte do código para montar obter o diagrama de Lyapunov

```python
#Define qual a distribuição de r
rs=np.linspace(0,4,10000)
#Número de iterações para determinar comportamento
    assintótico
n=1000
expoente_vector=np.vectorize(expoente_lyapunov)

#Escolhe x0 aleatório para cada  rs
x0=np.random.rand(len(rs))

#Aplica a função expoente de Lyapunov para cada valor
    de rs e x0
y=expoente_vector(x0,n,rs)

#Faz gráfico plottando y por r
fig,ax=plt.subplots()
ax.plot(rs,y,lw=0.5,c='r')
plt.draw()
labels=[q.get_text() for q in ax.get_yticklabels()]
locs=list(ax.get_yticks())
labels+=[r'$ln(2)$']
locs+=[np.log(2)]
ax.set_yticklabels(labels)
ax.set_yticks(locs)
ax.axhline(0,c='k',lw=0.5,alpha=0.8)
ax.axhline(np.log(2),c='g',lw=0.5,alpha=0.8)
ax.set_xlabel("r",fontsize=12)
ax.set_ylabel(r"$\lambda$",fontsize=12)

fig.savefig("3_2_1",dpi=200)
```

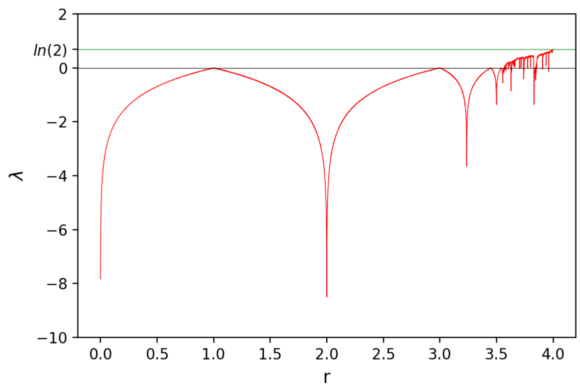

Figura 10: Gráfico de bifurcação de Lyapunov

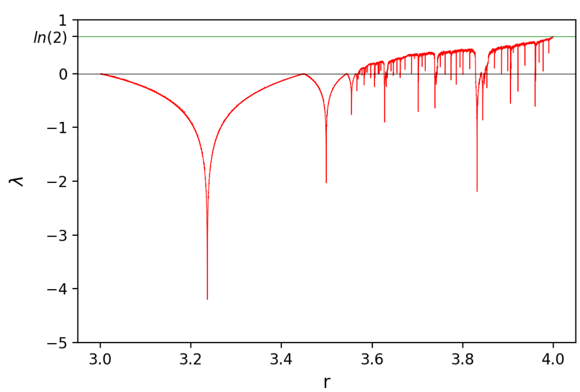

Figura 11: Gráfico de bifurcação de Lyapunov ampliado no intervalo [3,4]