

Introdução à Física Computacional

Projeto 5

Luiz Fernando S. E. Santos
No. USP 10892680

Junho de 2019

1 Introdução

Na seguinte prática, utilizamos de diversos algoritmos computacionais para o estudo de mapas logísticos, caos determinístico e dinâmica populacional. No primeiro exercício, trabalhamos com o um mapa logístico com um ponto fixo de período um. Já no segundo com dobras de período, o que nos leva ao caos determinístico e o aparecimento de fractais. No terceiro exercício, estudamos o modelo predador-presa de Lotka-Volterra, bem como utilizamos o já apresentado método de Runge-Kutta de ordem 4 para sua integração numérica.

Os programas do projeto foram feitos em C++, ao passo que o Python 3.5.2 com as bibliotecas Numpy e Matplotlib foi usado para graficar os resultados e, em alguns casos, facilitar algumas manipulações com os conjuntos de dados ao usar a estrutura de arrays do Numpy.

2 Instruções

Para a compilação dos programas .cpp, estando na pasta com o programa desejado deve-se executar no terminal:

```
g++ nomedoarquivo.cpp -o nomedoarquivo
```

Um arquivo *nomedoarquivo* será gerado. Para executá-lo, basta digitar também no terminal:

```
./nomedoarquivo
```

Os programas .cpp em geral já possuem um script para gerar um .dat com os resultados, bem como plotar e mostrar o gráfico deles com o .py. Caso hajam os resultados e se deseje executar apenas o .py, basta fazer:

```
python3 nomedoarquivo.py
```

Desta forma, podem ser testados vários valores iniciais das variáveis, sem a necessidade de plotá-los em um programa externo toda vez que novos dados são gerados. Para alterar esses valores, basta fazer isso no código .cpp, onde eles são atribuídos às tais variáveis.

3 Métodos e resultados

3.1 Ponto fixo de período um

3.1.1 (a)

Em um programa C++, defino a função:

$$G(x, r) = rx(1 - x) \quad (1)$$

E para os valores de r 0.5, 1.0, 2.0, é calculado $f(x) = x$ e $G(x, r)$. Os resultados podem ser visualizados no gráfico:

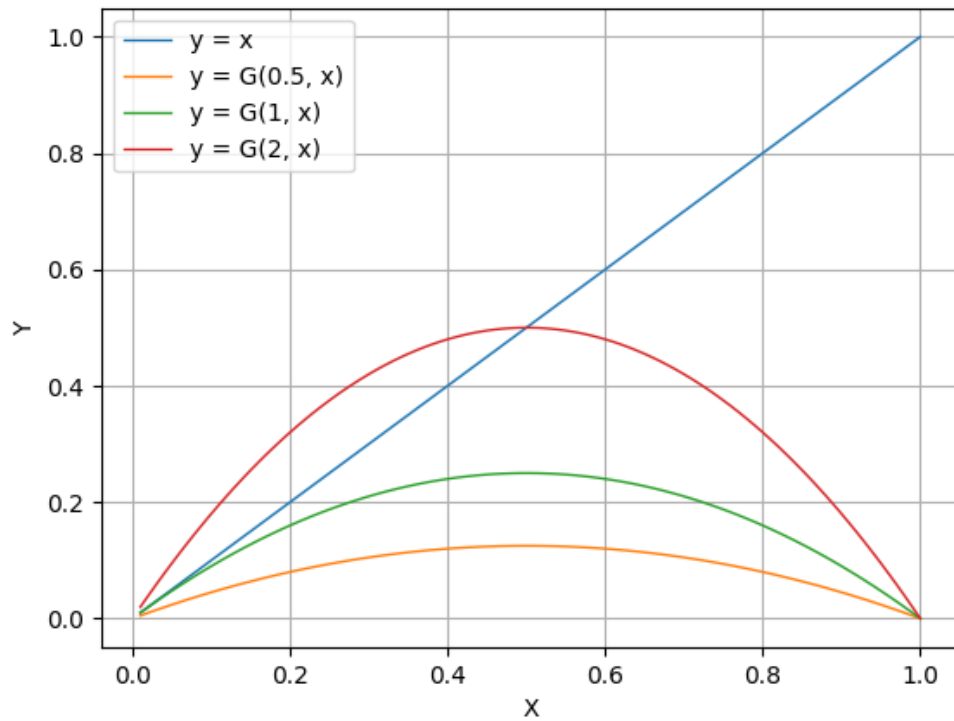


Figura 1: Funções G e f

Para entendermos porque da existência da solução não-trivial para a igualdade

$$G(x, r) = f(x) \quad (2)$$

apenas para $r > 1$, podemos desenvolver a equação [2]:

$$\begin{aligned} rx(1-x) &= x \\ r - rx &= 1 \\ r &= \frac{1}{1-x} \end{aligned}$$

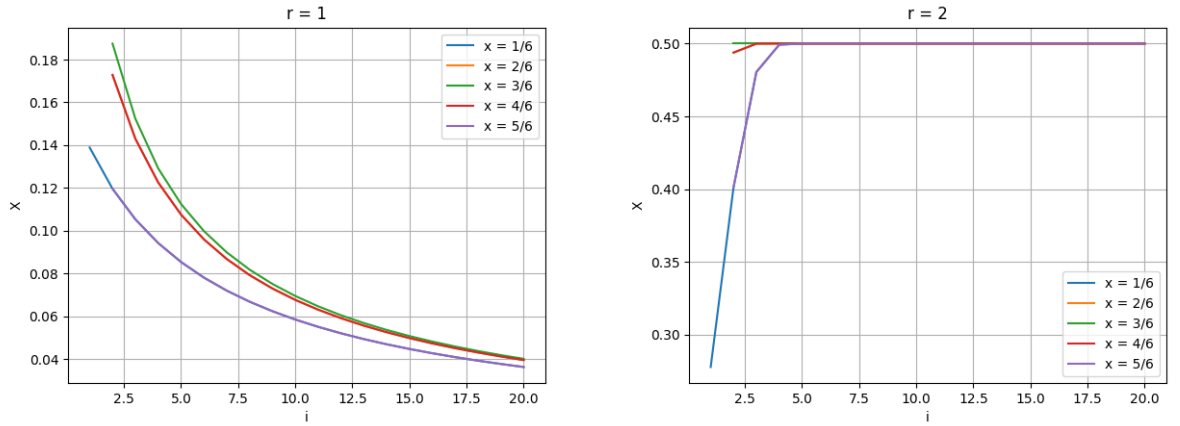
Supondo $r \leq 1$, teremos que:

$$\begin{aligned} 1-x &\geq 1 \\ \Rightarrow x &\leq 0 \end{aligned}$$

Mas se $x < 0$, $x \notin]0, 1[$, como fora definido. Desta forma, o único valor de x que satisfaz [2] para $r < 1$ é $x = 0$ (solução trivial).

3.1.2 (b)

Agora, usando $r = 1, 2, 2.5$, fazemos o mapa logístico para diferentes valores de x_0 . Obtemos, para cada valor de r , os seguintes resultados:



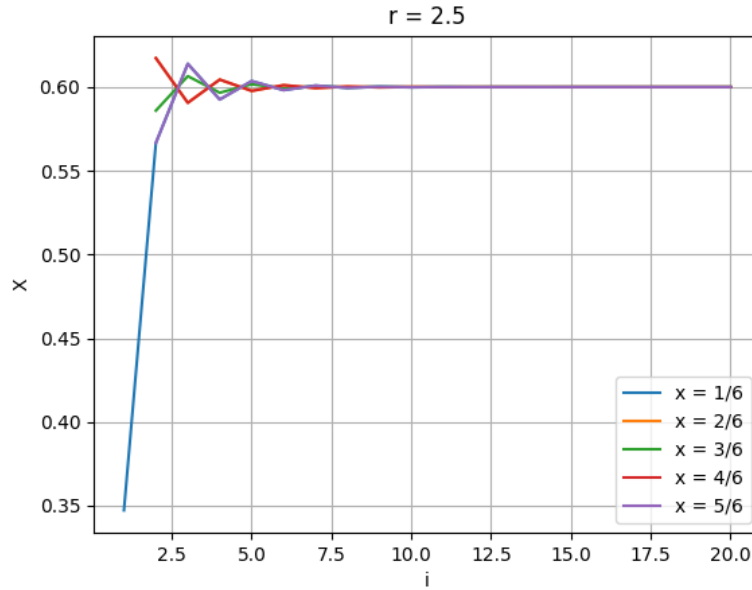


Figura 2: gráficos de x contra i

Podemos ver claramente que, mesmo para diferentes valores de x_0 , x sempre converge para um valor fixo (valor este que muda de acordo com r).

3.1.3 (c)

Tomamos agora um $\epsilon \ll 1$, o qual será usado para obtermos um valor muito próximo de x_0 ($x_0 + \epsilon$). Iremos iterar estes dois valores iniciais de x e ver como a distância d entre eles se comporta ao longo das iterações. Fazendo isso para $\epsilon = 0.01$ e três valores distintos de r , $r = 1.5, 2.5, 2.7$, plotamos o gráfico de d contra o índice i da iteração, assim obtendo:

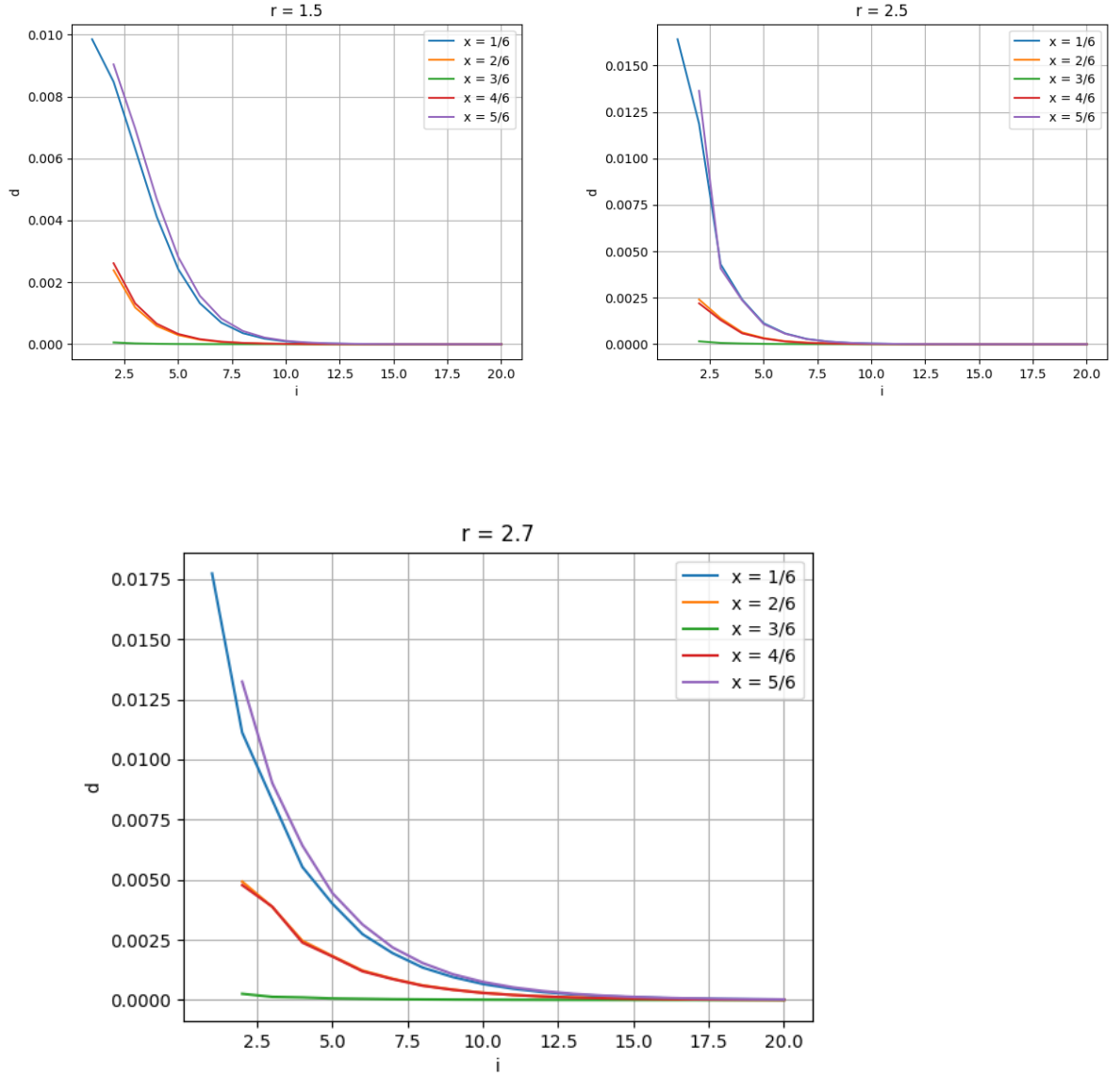


Figura 3: gráficos de d contra i

Estas curvas encontradas parecem razoavelmente um decaimento exponencial, e podemos confirmar que elas de fato o são ao plotarmos os dados em escala *log-linear*:

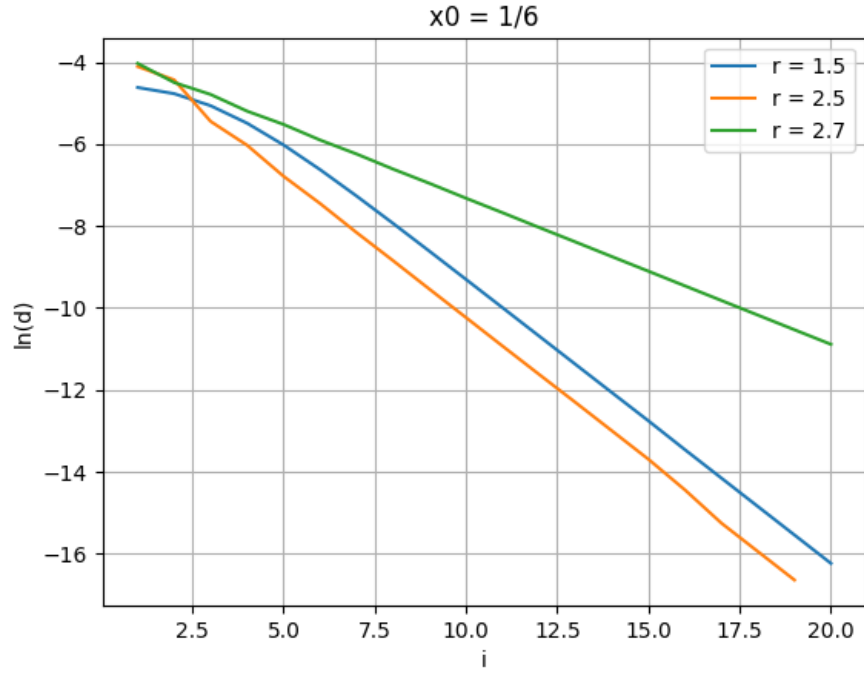


Figura 4: plot em escala log-lin.

Sendo o coef. de Lyapunov λ dado pela relação:

$$d_i \sim e^{-\lambda i} \quad (3)$$

Podemos obtê-lo através do coeficiente angular ao ajustarmos uma reta no gráfico de $\ln(d_i)$ contra i , para cada valor de r . Utilizando o programa QtiPlot para isso, obtemos:

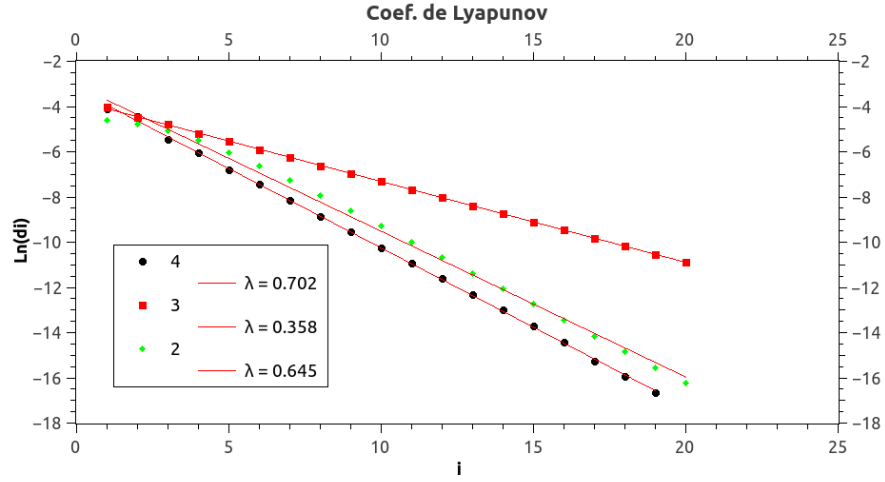


Figura 5: coef de Lyapunov para cada valor de r . Os pontos verdes, pretos e vermelhos correspondem a $r = 1.5, 2.5$ e 2.7 , respectivamente.

Podemos checar que isso continua valendo mesmo para os demais valores de x_0 :

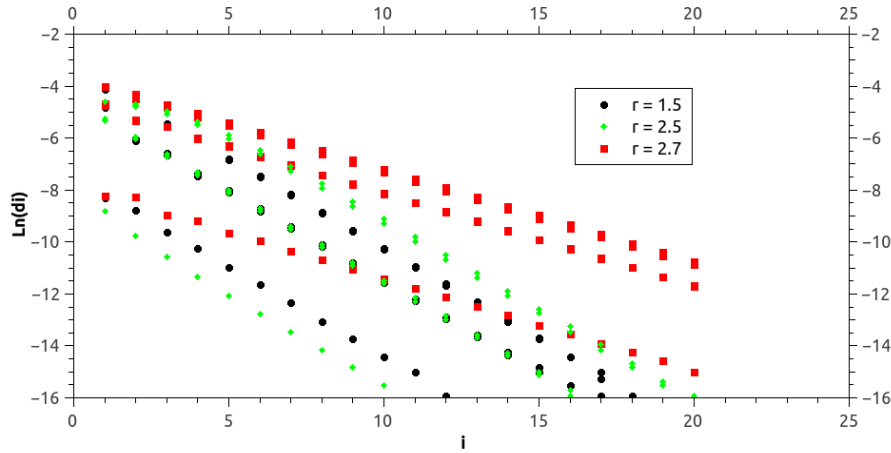


Figura 6: retas para os demais valores de x_0 . Podemos ver que elas são paralelas para um mesmo valor de r .

Pelos resultados vemos que, embora o coeficiente independa de x_0 neste caso, ele varia com r .

3.1.4 (d)

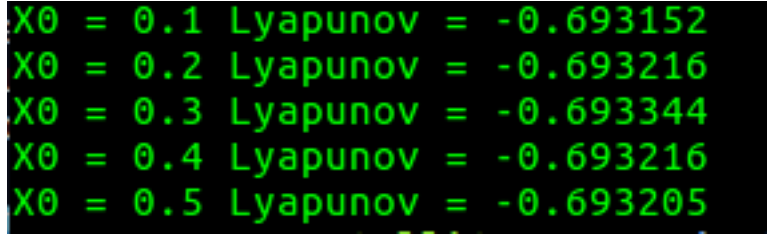
Agora, calculamos o coeficiente de Lyapunov através da estimativa:

$$\lambda \equiv \frac{1}{n} \sum_{j=0}^{n-1} \ln |G'(x_j)| \quad (4)$$

Antes de tudo devemos calcular a primeira derivada de $G(x)$:

$$G'(x) = \frac{d}{dx} rx(1-x) = r(1-2x)$$

Tendo isso, iteramos $G(x)$, calculando o logaritmo natural do módulo de sua primeira derivada em cada iteração e adicionando à somatória. Ao fim, dividimos tudo pelo número de iterações n . Para alguns diferentes valores de x_0 obtemos:



```
X0 = 0.1 Lyapunov = -0.693152
X0 = 0.2 Lyapunov = -0.693216
X0 = 0.3 Lyapunov = -0.693344
X0 = 0.4 Lyapunov = -0.693216
X0 = 0.5 Lyapunov = -0.693205
```

Figura 7: Valores obtidos para o coef. de Lyapunov através da somatória para diferentes valores de x_0 , com $r = 2.5$.

Podemos observar o coeficiente obtido através desse método está razoavelmente próximo daquele que calculamos através do coef. angular da reta ajustada, e de fato ele se mantém estável para diferentes valores de x_0 .

3.1.5 (e)

Tomamos $r = 2.9$ e $x_0 = 0.1$. Primeiramente fazemos um gráfico de $\ln |G'(x_j)|$ contra j para observar o comportamento do termo dentro da somatória definida anteriormente ao longo das iterações. Desta forma, obtemos o gráfico:

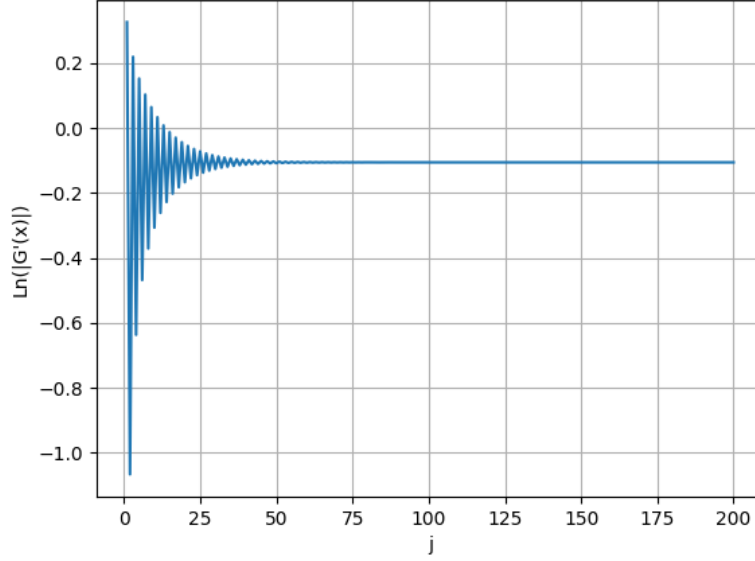


Figura 8: Variação do logaritmo da derivada com as iterações.

Podemos observar que no começo há uma oscilação, enquanto $\ln |G'(x_j)|$ se aproxima de λ . Após um certo número de iterações, o logaritmo se mantém praticamente constante. Por conta disso, podemos usar a relação [4] e obtemos, para um j grande:

$$\begin{aligned} \lambda &= \frac{1}{n} \sum_{j=0}^{n-1} \ln |G'(x_j)| \approx \frac{n \ln |G'(x_j)|}{n} \\ &= \ln |G'(x_j)| \end{aligned}$$

Para obtermos λ desta forma, iteramos a função 100 vezes, para que o citado termo se mantenha razoavelmente constante. Após, salvamos os valores de $\ln |G'(x_j)|$ em um arquivo. Em outro programa .py, lemos este arquivo, salvando os valores obtidos como elementos de um *array*. Contamos a frequência de cada elemento e através disso fazemos um histograma.

Do histograma podemos tirar algumas informações: o valor de λ é a média ponderada de cada valor de $\ln |G'(x_j)|$ levando em conta sua frequência, ao passo que o erro é obtido através do desvio padrão.

Usando este método, podemos obter dois tipos de resultados para duas situações: Primeiro, caso a precisão de nossos números não seja tão "grande", obtemos um histograma com poucos valores diferentes de $\ln |G'(x_j)|$, sendo a maior parte deles com uma frequência muito baixa e um em específico tendo um grande pico:

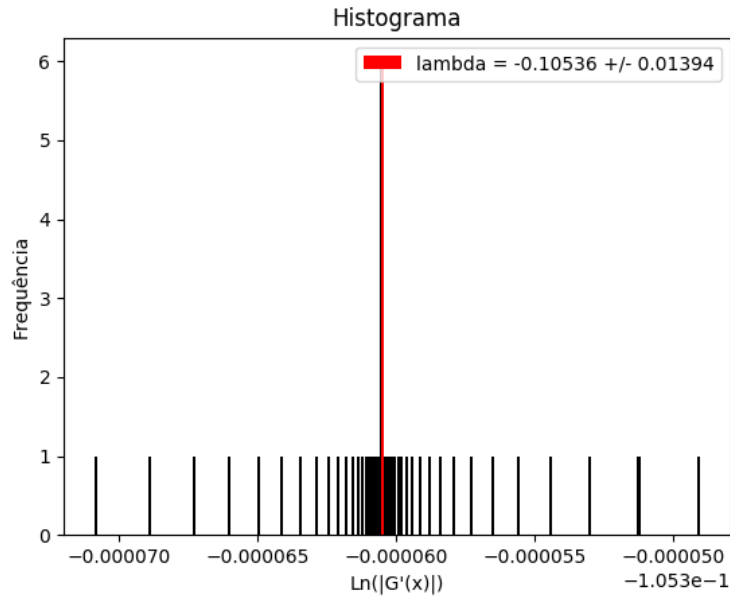


Figura 9: Primeiro caso de histograma: precisão da ordem de 10^{-9} casas decimais.

Já no caso de termos uma precisão muito alta, vamos ter vários valores de $\ln |G'(x_j)|$ com uma frequência baixa, mas com a grande maioria deles se concentrando próximo de λ :

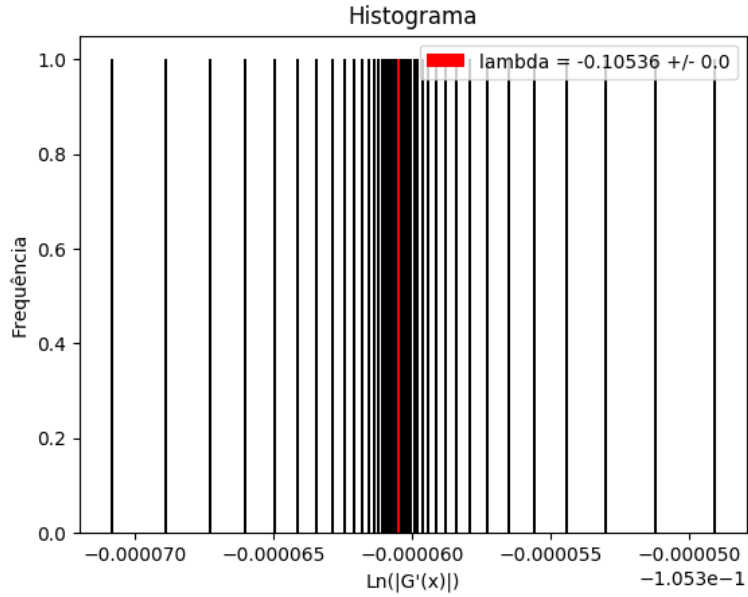


Figura 10: Segundo caso de histograma: precisão da ordem de 10^{-14} casas decimais.

Nosso método funciona bem para ambos os casos, e como o esperado nos dá resultados equivalentes.

3.2 Dobras de período e caos

3.2.1 (a)

Sendo G como definido anteriormente, plotamos para $r = 2.9, 3, 3.1$ os gráficos de $f(x) = x$, $g^2(x, r) = G(G(x))$. Obtemos:

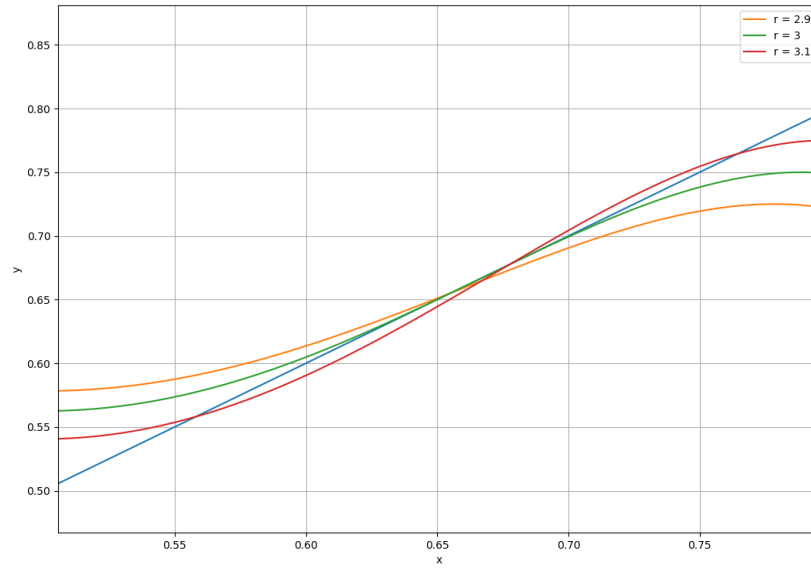
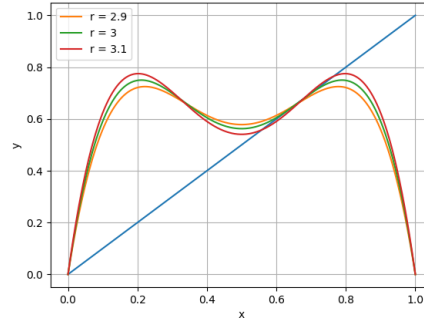


Figura 11: Plot de $f(x)$ e $g^2(x, r)$ para dados valores de r .

Podemos ver que $f(x)$ e $g^2(x, r)$ se interceptam três vezes (além de em 0) quanto $r = 3.1$.

3.2.2 (b)

Variando r e iterando a equação, visando encontrar o(s) ponto(s) fixo(s) x^* para cada valor, podemos plotar um gráfico de r contra x^* :

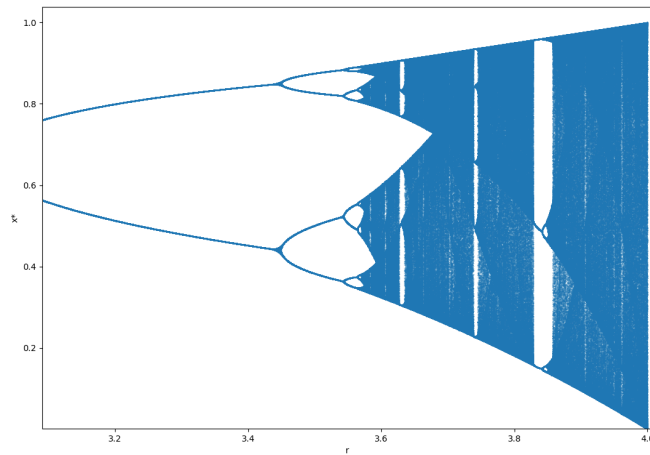


Figura 12: Diagrama de bifurcação.

É interessante notarmos o comportamento fractal desse diagrama: se dermos um zoom em algum ponto no meio deste diagrama, podemos ver:

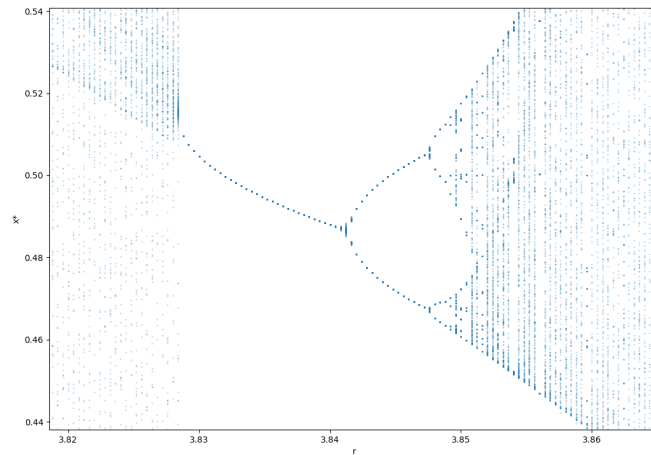


Figura 13: Zoom no diagrama.

Embora haja uma limitação de resolução devido ao número de valores de

r iterados, podemos ver bem esta região ao rodarmos o código novamente com r variando dentro dessa região:

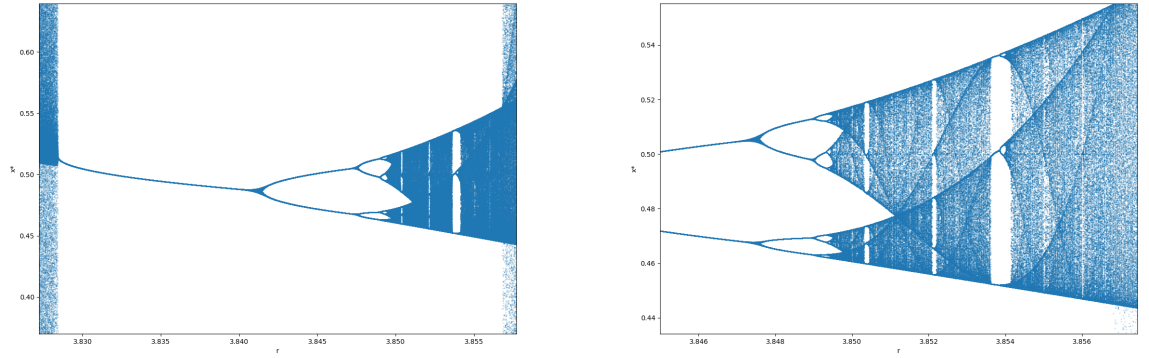


Figura 14: Iteração para novo intervalo de r dentro do caos.

3.2.3 (c)

Usando o mesmo código, simplesmente mudo o intervalo de interação de r para que r_2, r_3 possam ser determinados com maior precisão (sabemos que $r_1 = 3$):

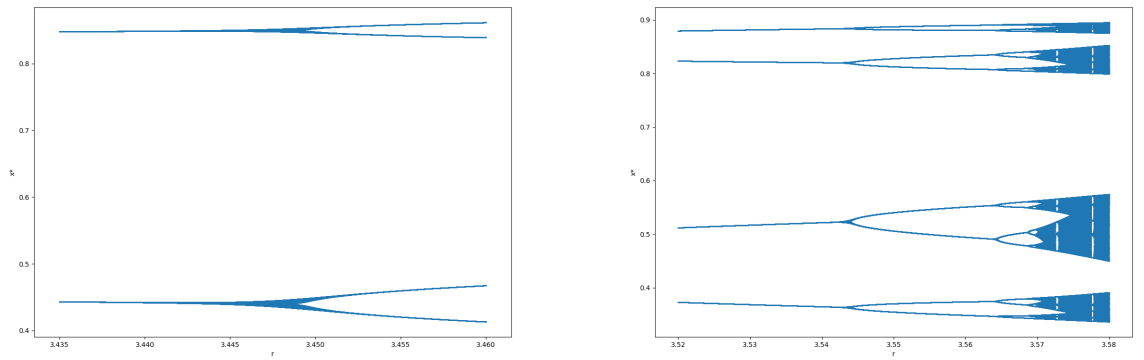


Figura 15: Iteração para novo intervalo de r para determinar bifurcações.

Com os valores obtidos, podemos calcular que

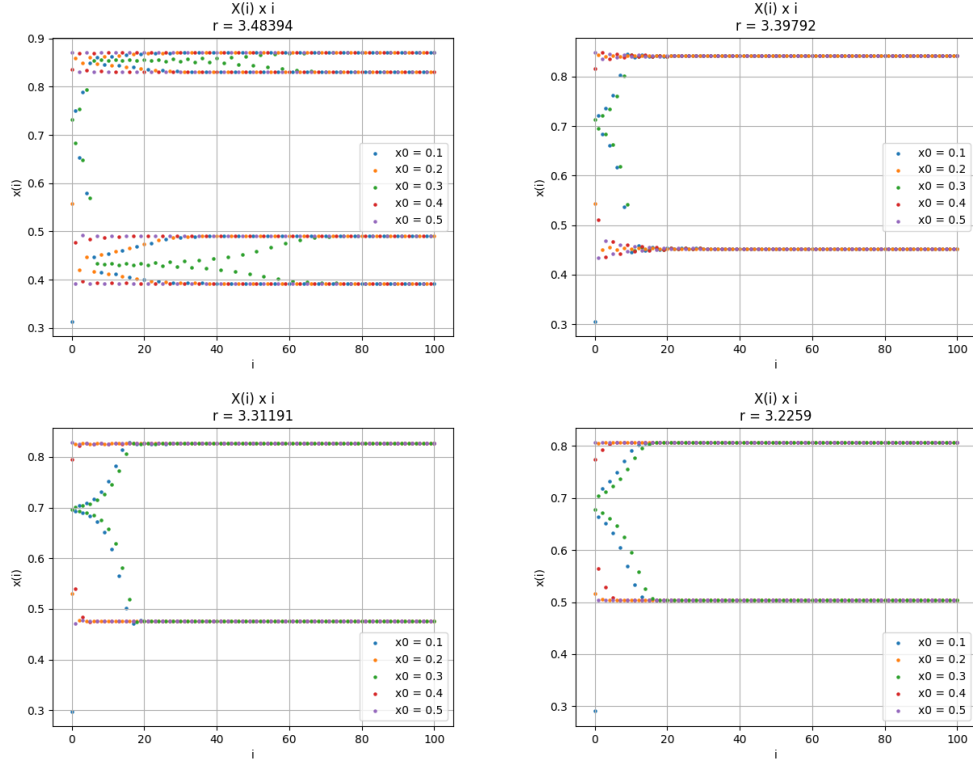
$$\delta = \frac{r_2 - r_1}{r_3 - r_2} = 4.6863$$

é o coeficiente de Feigenbaum.

3.2.4 (d)

Para cinco diferentes valores de r tais que $3.569946 < r < 4$ determinamos cinco valores de x_0 e um $\epsilon = 10^{-10}$. Iteramos para cada valor, $G(x_0, r)$ e $G(x_0 + \epsilon, r)$ em um loop, calculando suas distâncias.

Plotando x contra i obtemos:



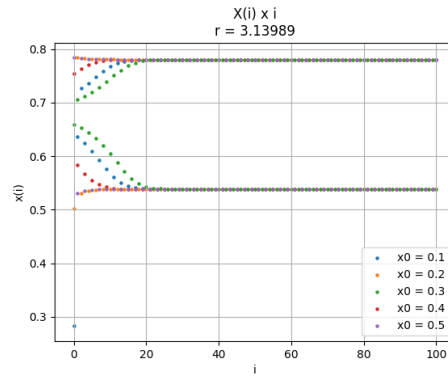
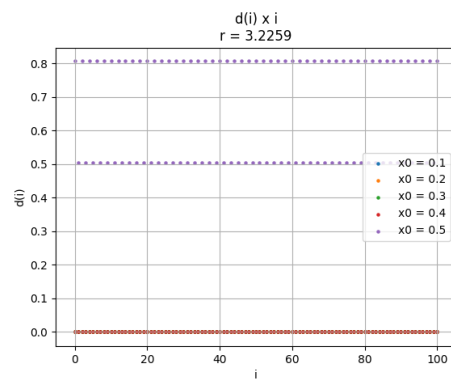
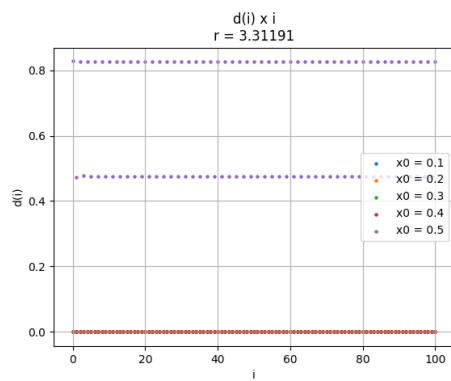
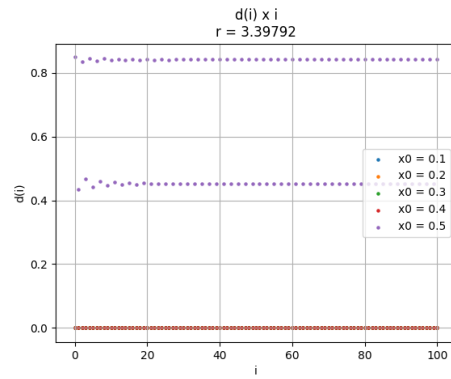
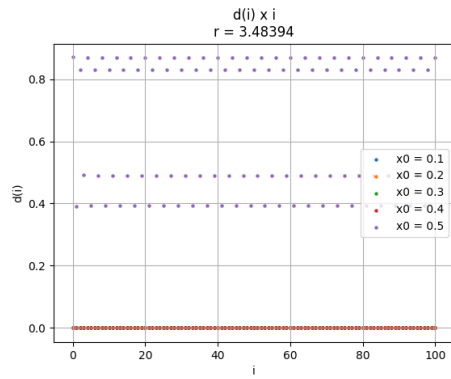


Figura 16: Gráficos de x contra i .

Já para d_i contra i :



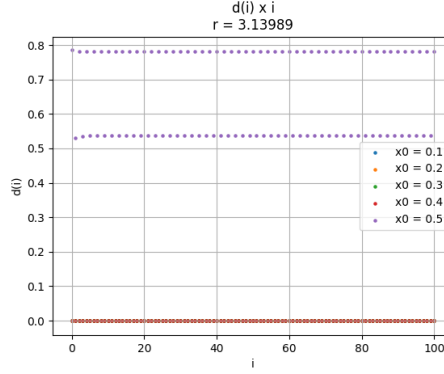


Figura 17: Gráficos de d_i contra i .

Podemos ver que diferente do que foi observado no exercício anterior, d_i se estabiliza em diferentes valores diferentes de 0. Isso mostra que, mesmo com um ϵ muito pequeno, já é o suficiente para resultar uma grande diferença no ponto fixo. Isso é o esperado para o caos determinístico: uma pequena diferença nas condições iniciais escala rapidamente no decorrer da evolução do sistema.

3.3 Modelo predador-presa

3.3.1 (a)

Levando em conta as equações:

$$\frac{dx}{dt} = Ax - Bxy \quad (5)$$

$$\frac{dy}{dt} = -Cy + Dxy \quad (6)$$

A constante A está relacionada com a frequência de reprodução das presas. Se definirmos que $\frac{dx}{dt}$ é alguma unidade do tipo "presas por instante de tempo", para que a igualdade valha, $[A] = \frac{1}{\text{tempo}}$, o que está de acordo com o esperado para uma frequência.

B, por sua vez, contribui negativamente para a variação do número de presas. Ele está definido de tal forma que $[B] = \frac{1}{\text{tempo} \cdot \text{predador}}$. Também podemos notar que, sendo $B > A$, uma vez que $x, y > 0$ (não faria sentido

termos um número negativo de animais), a população de presas decresce enquanto $y > 1$ e $x > 1$. Disso tudo, podemos deduzir que B está relacionado com a "eficiência" (ou velocidade) dos predadores em atacarem as presas.

Análogamente, as constantes C e D são definidas de forma parecida, com algumas diferenças: embora também tenha a unidade de $\frac{1}{\text{tempo}}$, C está relacionado com o termo que contribui negativamente para a variação dos predadores. Isso porque o aparecimento de novos predadores, embora de imediato esteja contribuindo para o aumento do número destes, em relação a variação deles no tempo, está retardando seu crescimento (já que mais predadores implica em uma competição maior pelas presas). Quanto à D, temos que $[D] = \frac{1}{\text{tempo} * \text{presa}}$, e também está relacionado com a facilidade dos predadores em obter caça.

3.3.2 (b)

Através do método de Runge-Kutta 4, discretizamos as equações [5] e [6] de forma a obter:

$$\begin{aligned} f_x(x, y) &= ax - bxy \\ f_y(x, y) &= -cy + dxy \end{aligned}$$

$$\begin{aligned} F_x^1 &= f_x(x, y) \\ F_y^1 &= f_y(x, y) \\ F_x^2 &= f_x(x + 0.5 * dt * F_x^1, y + 0.5 * dt * F_y^1) \\ F_y^2 &= f_y(x + 0.5 * dt * F_x^1, y + 0.5 * dt * F_y^1) \\ F_x^3 &= f_x(x + 0.5 * dt * F_x^2, y + 0.5 * dt * F_y^2) \\ F_y^3 &= f_y(x + 0.5 * dt * F_x^2, y + 0.5 * dt * F_y^2) \\ F_x^4 &= f_x(x + 0.5 * dt * F_x^3, y + 0.5 * dt * F_y^3) \\ F_y^4 &= f_y(x + 0.5 * dt * F_x^3, y + 0.5 * dt * F_y^3) \end{aligned}$$

E integramos numericamente:

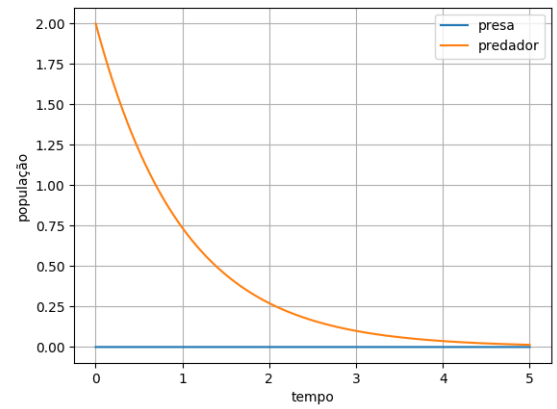
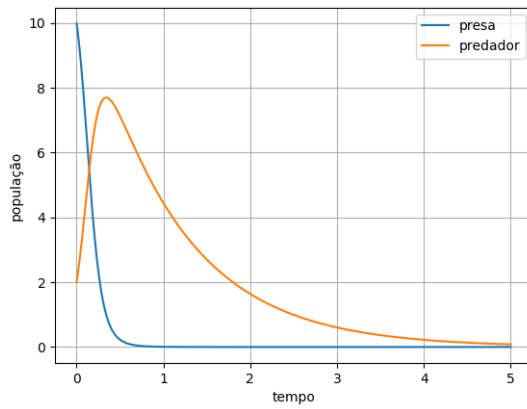
$$x_{i+1} = x_i + \frac{dt}{6} (F_x^1 + \frac{1}{2}F_x^2 + \frac{1}{2}F_x^3 + F_x^4)$$

$$y_{i+1} = y_i + \frac{dt}{6} (F_y^1 + \frac{1}{2}F_y^2 + \frac{1}{2}F_y^3 + F_y^4)$$

O programa deve simplesmente iterar estas equações, nessa ordem.

3.3.3 (c)

Aplicamos o algoritmo descrito no item anterior para as condições iniciais $x = 10, y = 2$; $x = 0, y = 2$; $x = 10, y = 0$ respectivamente. Desta forma, obtemos:



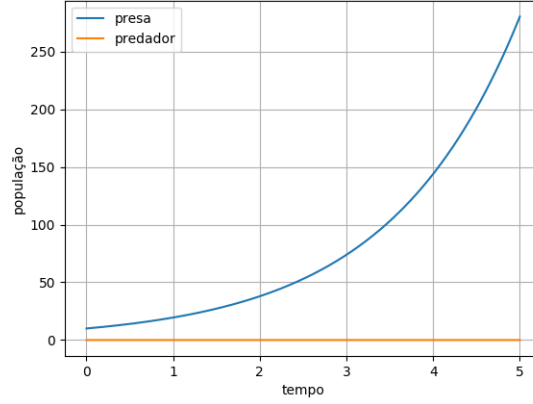


Figura 18: Dimâmica predador-presa para diferentes valores iniciais.

Os dois últimos casos são especialmente fáceis de analisar:

Caso o número de predadores seja zero, temos:

$$\begin{aligned}
 \frac{dx}{dt} &= ax \\
 \Rightarrow \frac{dx}{x} &= a dt \\
 \Rightarrow \int \frac{1}{x} dx &= a \int dt \\
 &= \ln(x) + c_1 = at + c_2 \\
 \Rightarrow x(t) &= \alpha e^{at}
 \end{aligned} \tag{7}$$

Onde α é o número inicial de presas (quando $t = 0$). Através de um método similar, podemos ver que $y(t)$ também é exponencial:

$$y(t) = \beta e^{-ct} \tag{8}$$

β o número inicial de predadores. Isso está de acordo com o que podemos observar dos gráficos.

3.3.4 (d)

Graficando o espaço de fase para estes casos, obtemos respectivamente:

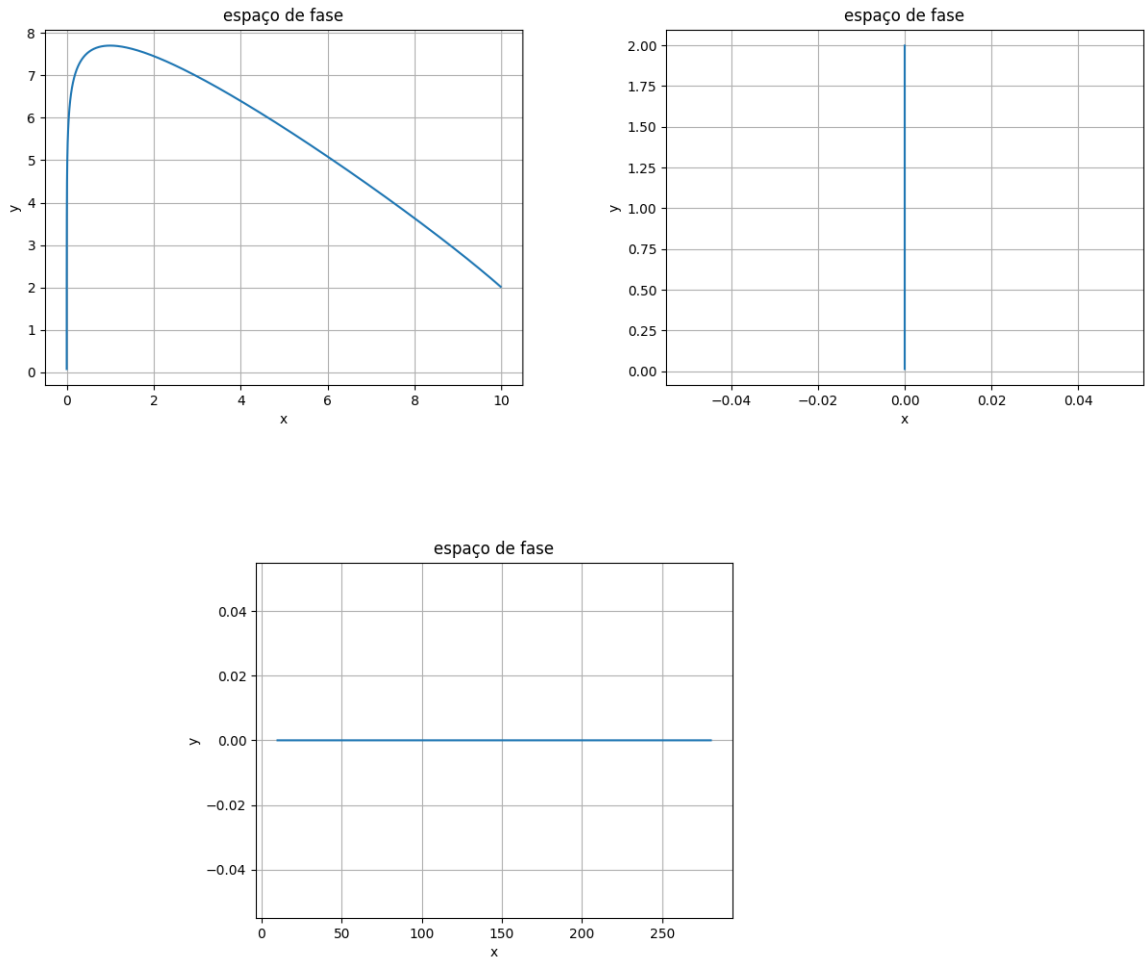


Figura 19: Espaços de fase para diferentes valores iniciais.

3.3.5 (e)

Dados os valores iniciais de x, y e os parâmetros a, b, c, d dados, plotamos o gráfico de predadores e presas por tempo, obtendo:

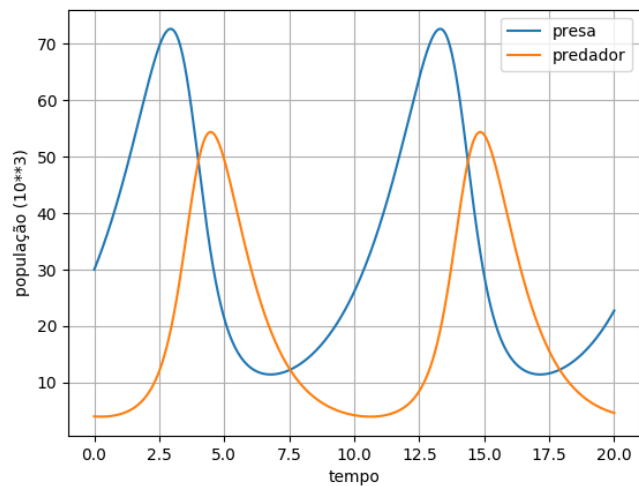


Figura 20: Predador e presa no tempo.

E o espaço de fase:

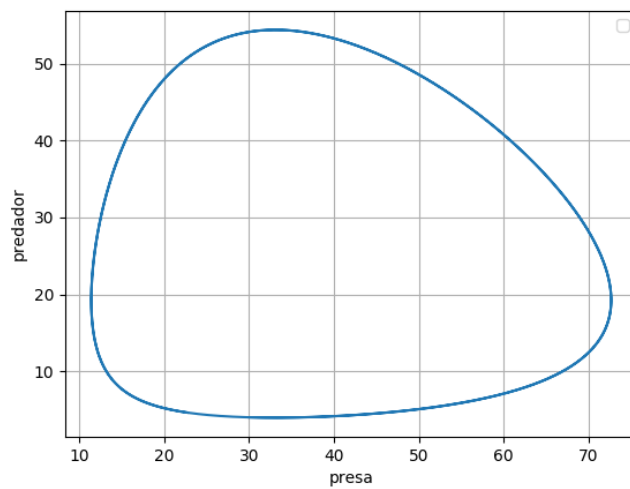


Figura 21: Espaços de fase.

Observando o número de lebres e lince a cada ano para comprar com a tabela fornecida, obtemos:

Ano	Lebre	Lince
1900	30	4
1901	43.8097	4.39776
1902	61.5835	7.57166
1903	72.5968	20.5118
1904	49.5024	48.8915
1905	21.5012	49.6134
1906	12.7161	30.8115
1907	11.5006	16.9562
1908	13.5096	9.47126
1909	18.1539	5.80398
1910	25.9825	4.22602
1911	37.9943	4.0513
1912	54.6413	5.80712
1913	70.9186	13.5572
1914	62.5023	38.2608
1915	29.3582	53.9887
1916	14.6641	37.8973
1917	11.4812	21.3344
1918	12.4424	11.7294
1919	16.0626	6.87993
1920	22.58	4.64307

Figura 22: Tabela com resultados dos anos.

Embora os números sejam razoavelmente diferentes dos reais, fornecidos na tabela, o comportamento oscilatório pode ser observado nos dois casos. Se o modelo predador-presa de Lotka-Volterra é um bom modelo para esse caso, acredito que depende do propósito. Caso se deseje precisão quanto aos números brutos das populações, não é um bom modelo. Porém em relação às variações ele parece simular razoavelmente bem.