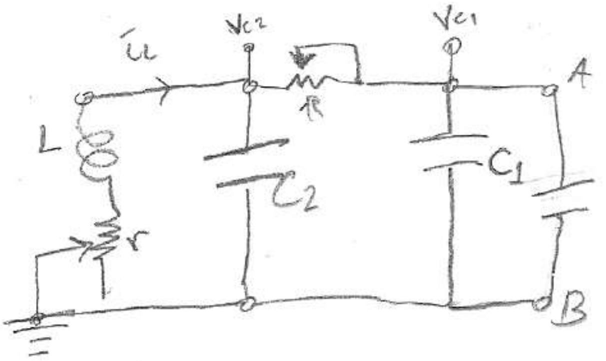
Nicholas Funari Voltani
9359365
24/11/2020

Lista 2 - Introdução ao Caos

(26) Equação de queda de tensão num indutor: $\boxed{V = L\frac{dI}{dt}}$

Equação de corrente num capacitor: $Q = CV \Rightarrow I = C\frac{dV}{dt}$

(27)



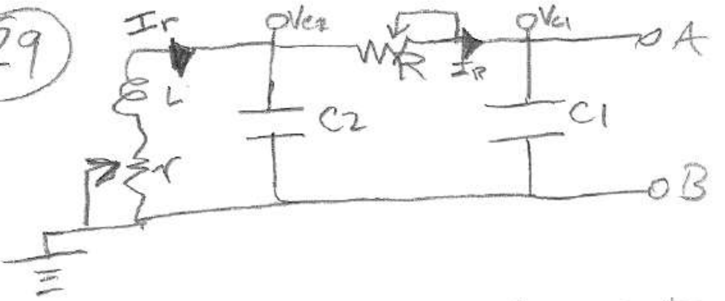
Buscamos a corrente no regime estacionário, i.e., após os capacitores terem se carregado.

Num regime de tensão DC, $V_I = L\frac{dI}{dt} = 0$ (indutor age como c.c.), e $V_C = 0$ (capacitores agem como um circuito aberto após terem se carregado completamente).

Logo, a resistência efetiva é $R + r$, e a corrente é $I = \frac{V}{(R+r)}$.

(28) Nos resistores, teremos que a energia dissipada será RI^2 e rI^2.

Para os capacitores totalmente carregados, é $E_1 = \frac{C_1 V_{C1}^2}{2}$, e $E_2 = \frac{C_2 V_{C2}^2}{2}$. Para o indutor, $E_L = \frac{LI^2}{2}$

29 

Temos que, após o transiente, a corrente sobre r

é $I_r = \frac{V_{C2}}{r}$, e sobre R é $I_R = \frac{V_{C2} - V_{C1}}{R}$

30 Energias dissipadas pelos resistores:

$$\begin{cases} E_r = r I_r^2 = \frac{V_{C2}^2}{r} \\ E_R = R I_R^2 = \frac{(V_{C2} - V_{C1})^2}{R} \end{cases}$$

Energia sobre o indutor: $E_L = \frac{L I_r^2}{2} = \frac{L V_{C2}^2}{2r^2}$

Energia sobre os capacitores:

$E_{C1} = \frac{C_1 V_{C1}^2}{2}$; $E_{C2} = \frac{C_2 V_{C2}^2}{2}$

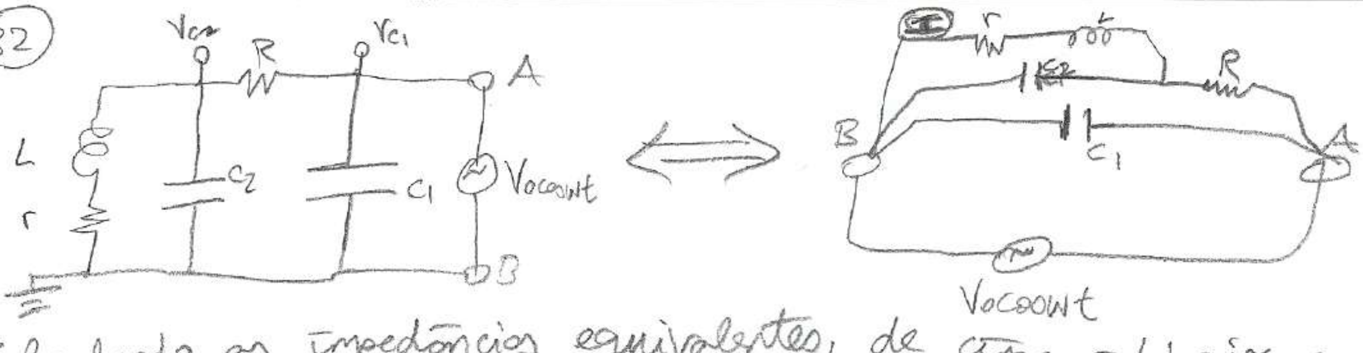
31 As impedâncias sobre os elementos ativos, dado $V = V_0 \cos \omega t$, são:

- Resistores: $Z_r = r$; $Z_R = R$

- Capacitores: $Z_{C1} = \frac{-i}{\omega C_1}$; $Z_{C2} = \frac{-i}{\omega C_2}$

- Indutor: $Z_L = i\omega L$

(32)



Calculando as impedâncias equivalentes, de cima p/ baixo:

$$Z_1 = r + i\omega L \quad \text{I}$$

$$\frac{1}{Z_2} = \frac{1}{(r+i\omega L)} + \frac{1}{\left(\frac{1}{i\omega C_2}\right)} = \frac{1}{r+i\omega L} + i\omega C_2$$

$$= \frac{1 + (ir - \omega L)\omega C_2}{r + i\omega L} = \frac{1 - \omega^2 L C_2 + i r \omega C_2}{r + i\omega L}$$

$$Z_2 = \frac{r + i\omega L}{(1-\omega^2 L C_2) + i r \omega C_2} \cdot \frac{(1-\omega^2 L C_2) - i r \omega C_2}{(1-\omega^2 L C_2 - i r \omega C_2)} \qquad \alpha \equiv 1 - \omega^2 L C_2$$

$$= \frac{(r + i\omega L)}{\alpha^2 + r^2\omega^2 C_2} \cdot \frac{(\alpha - i\omega r C_2)}{1}$$

$$= \frac{(r\alpha + \omega^2 r L C_2) + i\omega(\alpha L - r^2 C_2)}{\alpha^2 + r^2 \omega^2 C_2}$$

$$Z_3 = Z_2 + r$$

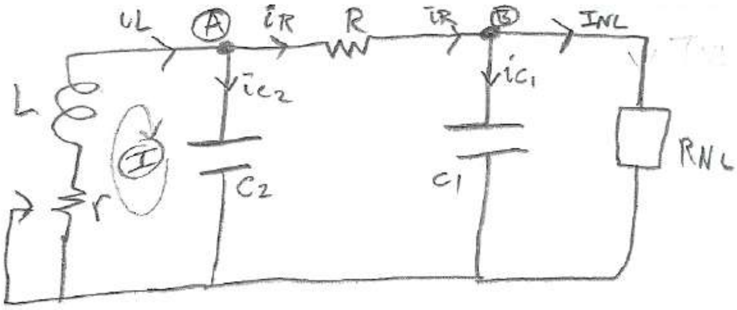
$$\frac{1}{Z_{ef}} = \frac{1}{Z_3} + \frac{1}{\left(\frac{1}{i\omega C_1}\right)} = \frac{1}{Z_3} + i\omega C_1$$

$$= \frac{1 + i\omega C_1 Z_3}{Z_3}$$

Portanto, a impedância resultante é

$$\therefore Z_{ef} = \frac{Z_3}{1 + i\omega C_1 Z_3}$$

(41)



Resolvendo a tensão total no loop I por regra de Kirchhoff, vem

$$r i_L + \underbrace{L\frac{di_L}{dt}}_{=V_L} + V_{C_2} = 0$$

$$\Rightarrow \boxed{\frac{di_L}{dt} = \frac{-V_{C_2} - r i_L}{L}}$$

As correntes que entram e saem dos nós A e B, respectivamente, devem somar a 0:

$$\begin{cases} i_L = i_R + i_{C_2} \\ i_R = I_{NL} + i_{C_1} \end{cases}$$

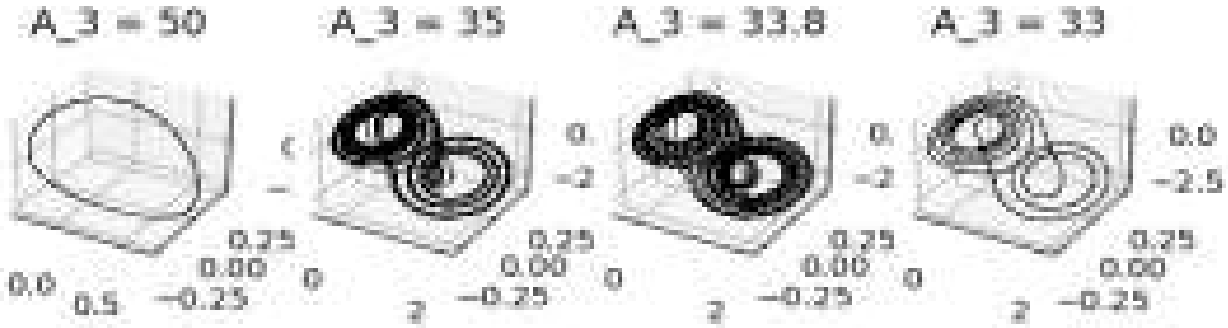
Mas temos que a corrente i_R advém da diferença de potencial $V_{C_2} - V_{C_1}$. Logo, $i_R = \frac{V_{C_2} - V_{C_1}}{R}$.

Como $i_{C_K} = C_K \frac{dV_{C_K}}{dt}$, temos

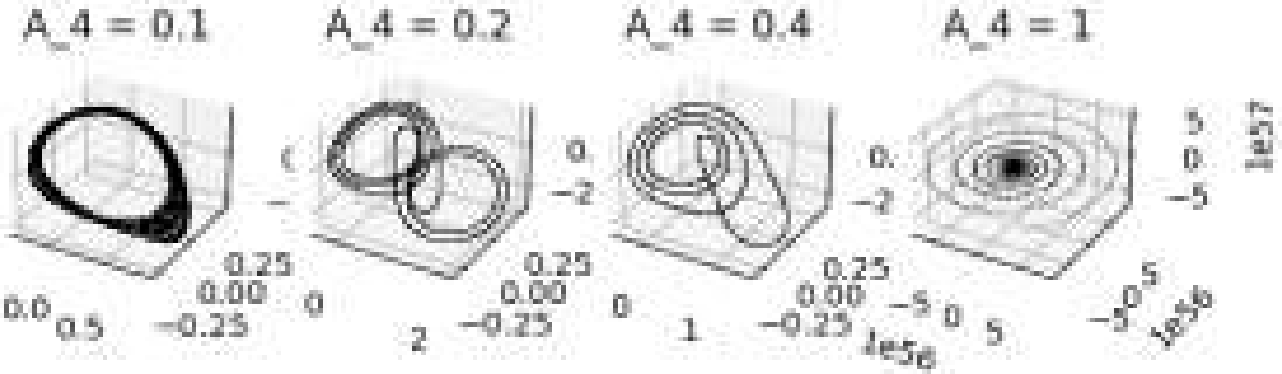
$$i_{C_2} = C_2 \frac{dV_{C_2}}{dt} = i_L - \frac{(V_{C_2} - V_{C_1})}{R} \quad \boxed{\therefore \frac{dV_{C_2}}{dt} = \frac{i_L}{C_2} - \frac{(V_{C_2} - V_{C_1})}{RC_2}}$$

$$i_{C_1} = C_1 \frac{dV_{C_1}}{dt} = \frac{V_{C_2} - V_{C_1}}{R} - I_{NL} \quad \boxed{\therefore \frac{dV_{C_1}}{dt} = \frac{V_{C_2} - V_{C_1}}{RC_1} - \frac{I_{NL}}{C_1}}$$

Exercício 43



Exercício 44

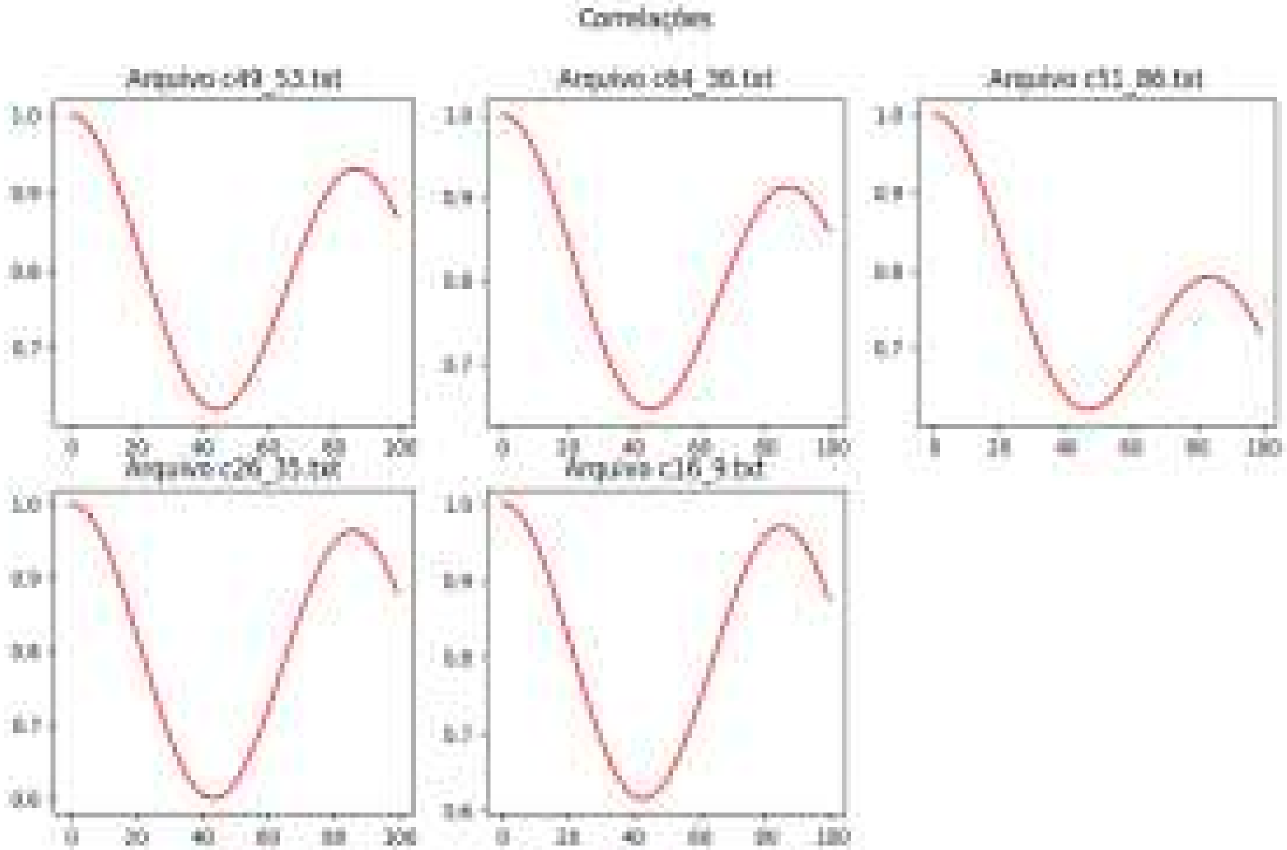


Exercício 46

```python
def AutoCorr(X,tau):
    '''
    Calcula a autocorrelação (normalizada) da variável $X$, com passo $tau$.
    Autocorr = sum(X[0:end-tau].*X[tau:end])/sum(X[0:end-tau].*X[0:end-tau])

    '''
    import numpy as np

    ## Por definição, temos Autocorr(X,0) = 1
    if tau == 0:
        return 1

    ##### Caso tau != 0 ############################

    ## Vetores que vão se multiplicar
    ## precisam ter mesmo tamanho [len(X) - tau]
    numerador1 = X[tau:]
    numerador2 = X[:-tau]

    ## Multiplica ponto a ponto Num1.*Num2, e soma tudo
    numerador = np.multiply(numerador1,numerador2)
    numerador = sum(numerador)

    ## Calcula o denominador
    denominador = np.multiply(numerador2,numerador2)
    denominador = sum(denominador)

    return numerador/denominador
```

```python
## Importa LeTxts, AutoCorr
from funcs import *
import matplotlib.pyplot as plt
import numpy as np

TodasTensoes = LeTxts('Txts')
#TodasTensoes = TodasTensoes[0:1]
fatorTransiente = 0.8

for k in range(len(TodasTensoes)):
    Tensao = TodasTensoes[k]
    X = Tensao[:-1]
    X = X[int(fatorTransiente*len(X)):]
    X = np.array(X)
    Txt = Tensao[-1]

    ##### Para plotar a autocorrelação #################

    ## Título com o nome do arquivo
    plt.suptitle(f"Correlações")

    plt.subplot(2,3,k+1)
    PlotaAutoCorrs(X)
    plt.title(f"Arquivo {Txt}")

plt.show()
```

Exercício 46



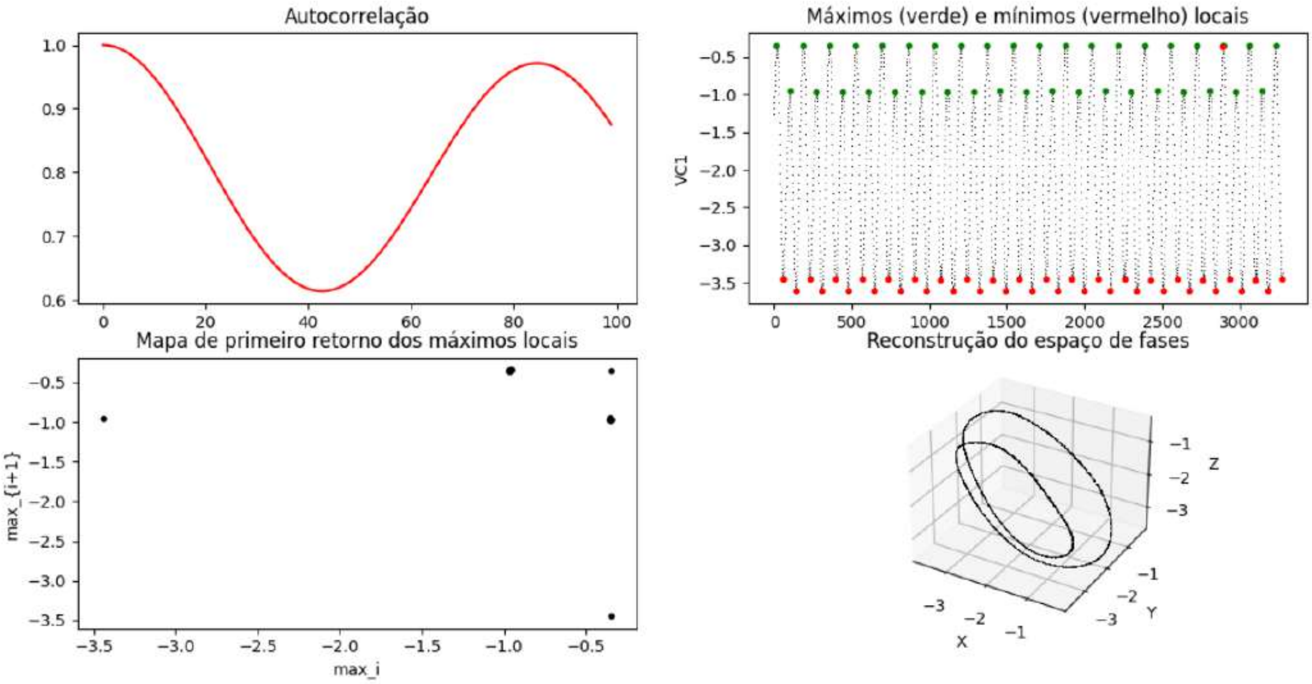
Tomamos n = 44

para reconstrução do espaço de fases

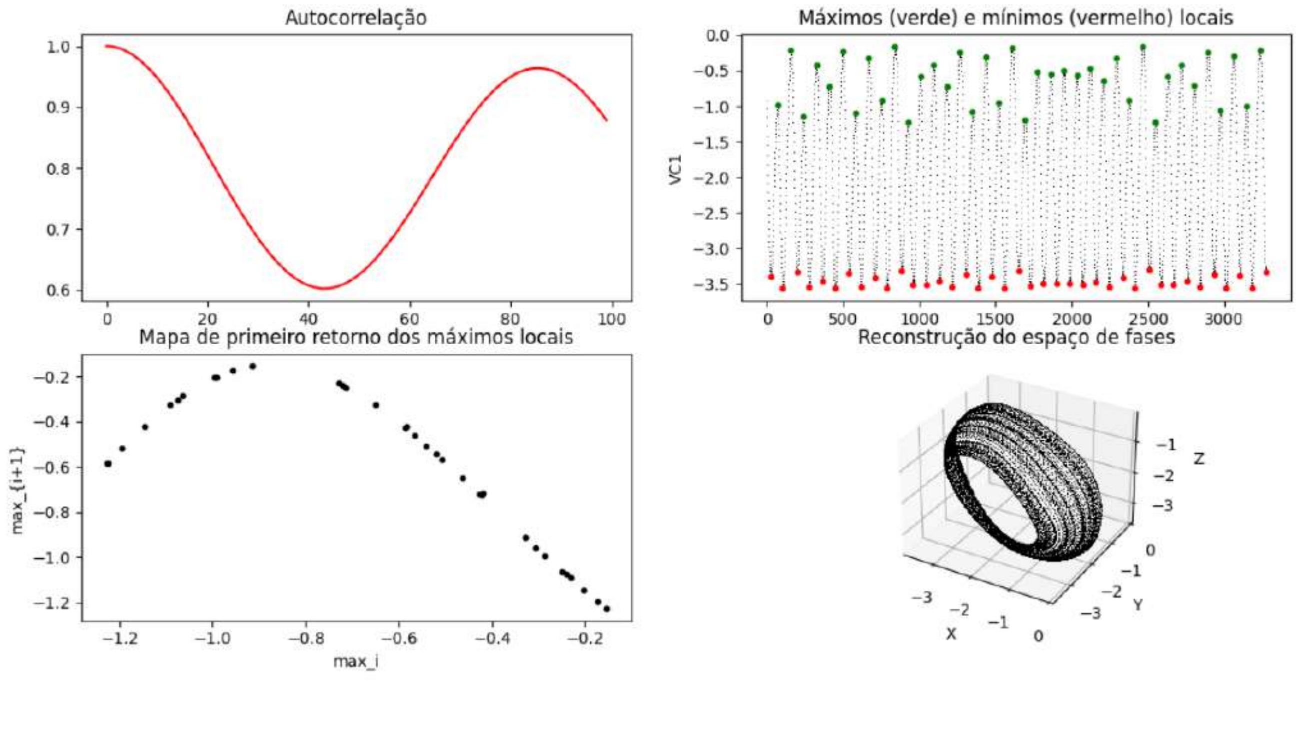
=> tau = 44/2 = 22

Exercício 47-52

Arquivo c16_9.txt

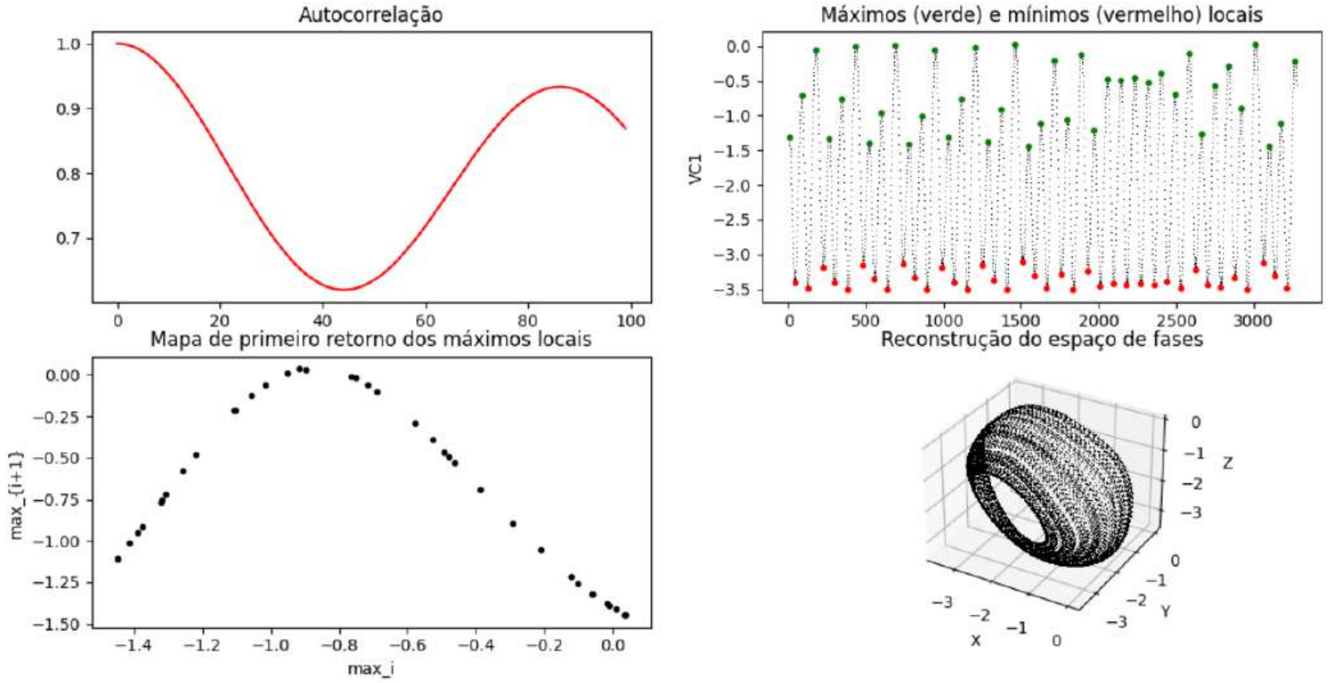


Arquivo c26_35.txt

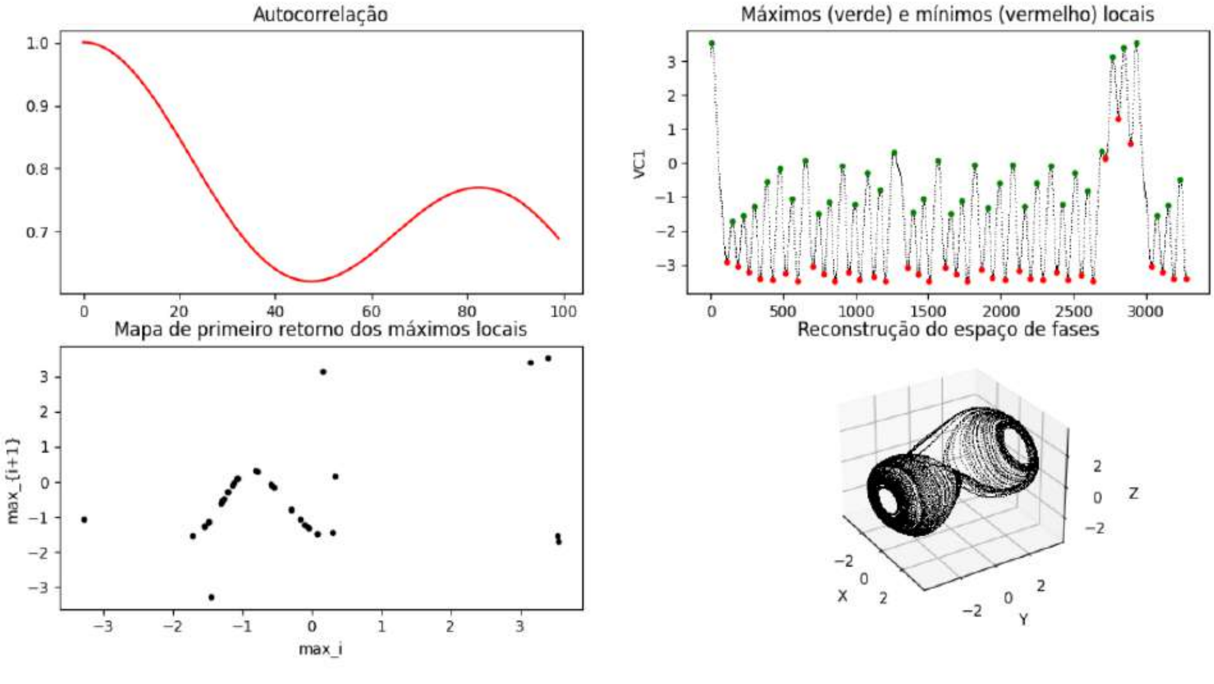


Exercício 47-52

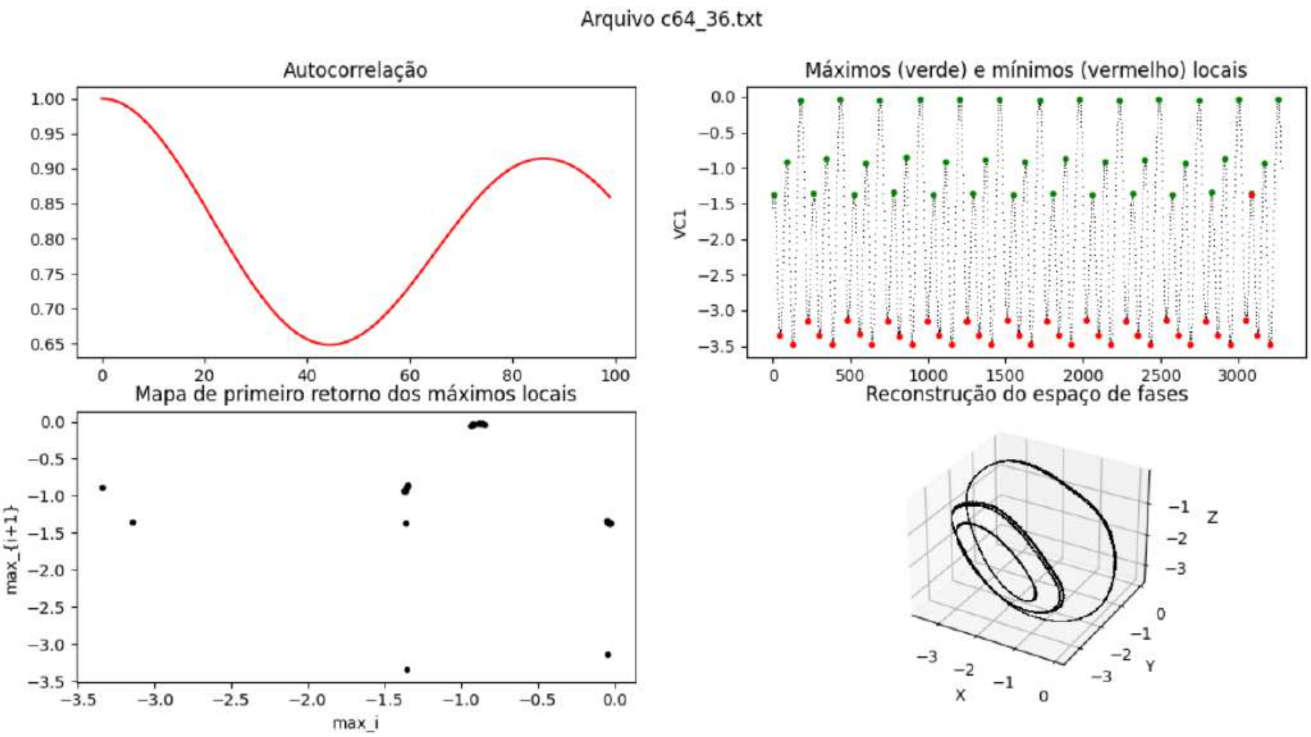
Arquivo c49_53.txt



Arquivo c51_86.txt



Exercício 47-52



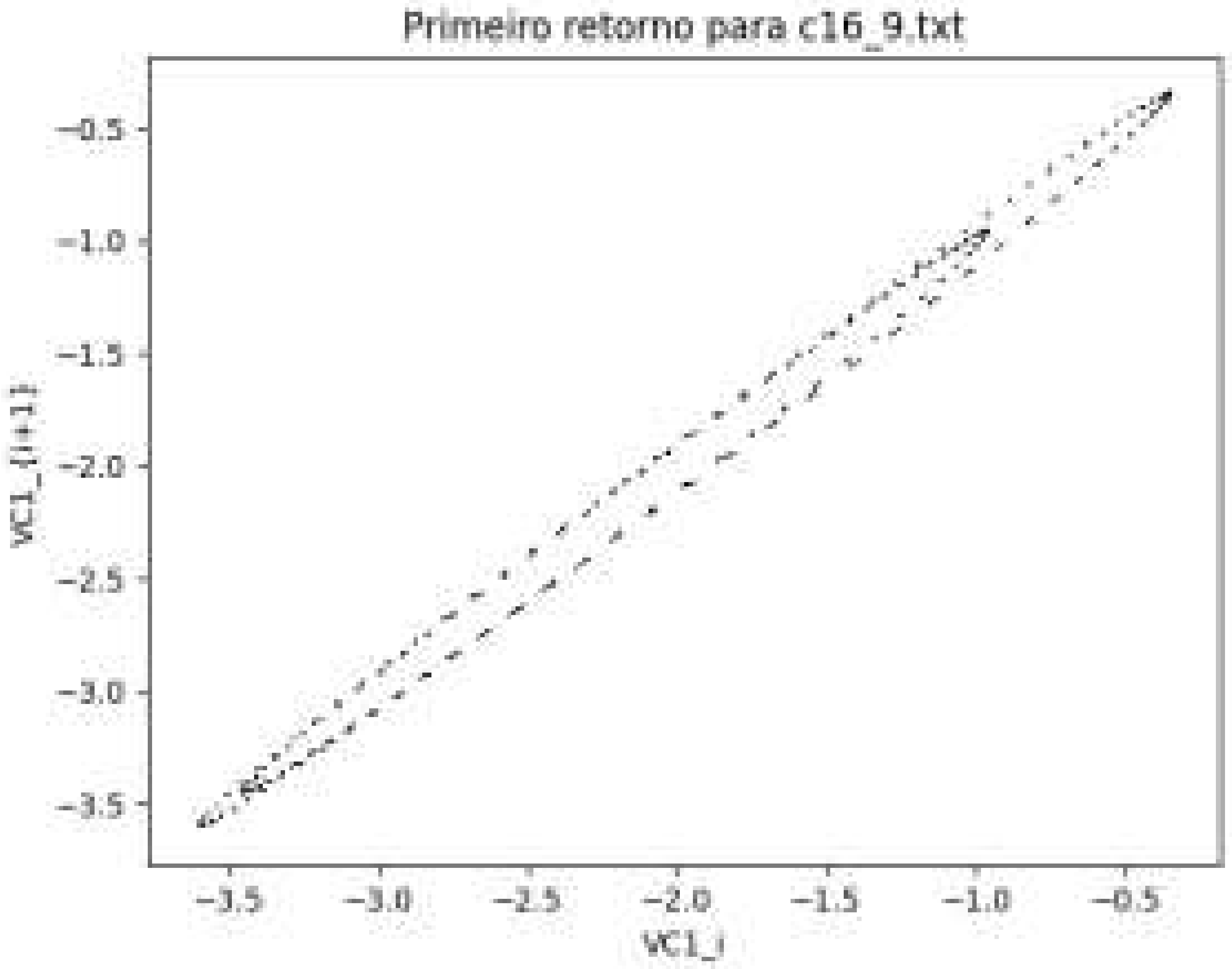
Exercício 50

```python
def PlotaMaxMins(X, fatorTransiente):
    import numpy as np
    import matplotlib.pyplot as plt

    ## Para plotar X em função das iterações
    X = X[int(fatorTransiente*len(X)):]
    EixoX = list(range(len(X[1:-1])))
    EixoX = np.array(EixoX)

    ##### Para gerar a lista de máximos locais ########

    ##          V_{i+1} <- é max local
    ##         /        \
    ##        /          \
    ##    V_i             \
    ##                     V_{i+2}

    Bool1 = (X[1:-1] - X[:-2]) >=  0
    Bool2 = (X[2:] - X[1:-1]) < 0

    ## Fazendo o operador AND a cada elemento
    BoolMax = np.multiply(Bool1,Bool2)

    ## Aplicando a "máscara" booleana no vetor VC1
    EixoXMax = EixoX[BoolMax]
    Maximos = X[1:-1][BoolMax]

    ##### Para gerar a lista de mínimos locais ########

    ##      V_i
    ##         \           V_{i+2}
    ##          \         /
    ##           \       /
    ##            V_{i+1} <- é min local

    Bool1 = (X[1:-1] - X[:-2]) <  0
    Bool2 = (X[2:] - X[1:-1]) >= 0

    ## Fazendo o operador AND a cada elemento
    BoolMin = np.multiply(Bool1,Bool2)
plt.plot(EixoX, X[1:-1], ',k')
plt.plot(EixoXMax,Maximos,'.g')
plt.plot(EixoXMin,Minimos, '.r')
plt.xlabel("Iterações")
plt.ylabel("VC1")
plt.title("Máximos (verde) e mínimos (vermelho) locais")

## Para fazer os mapas de primeiro retorno
return Maximos, Minimos
```

Exercício 51

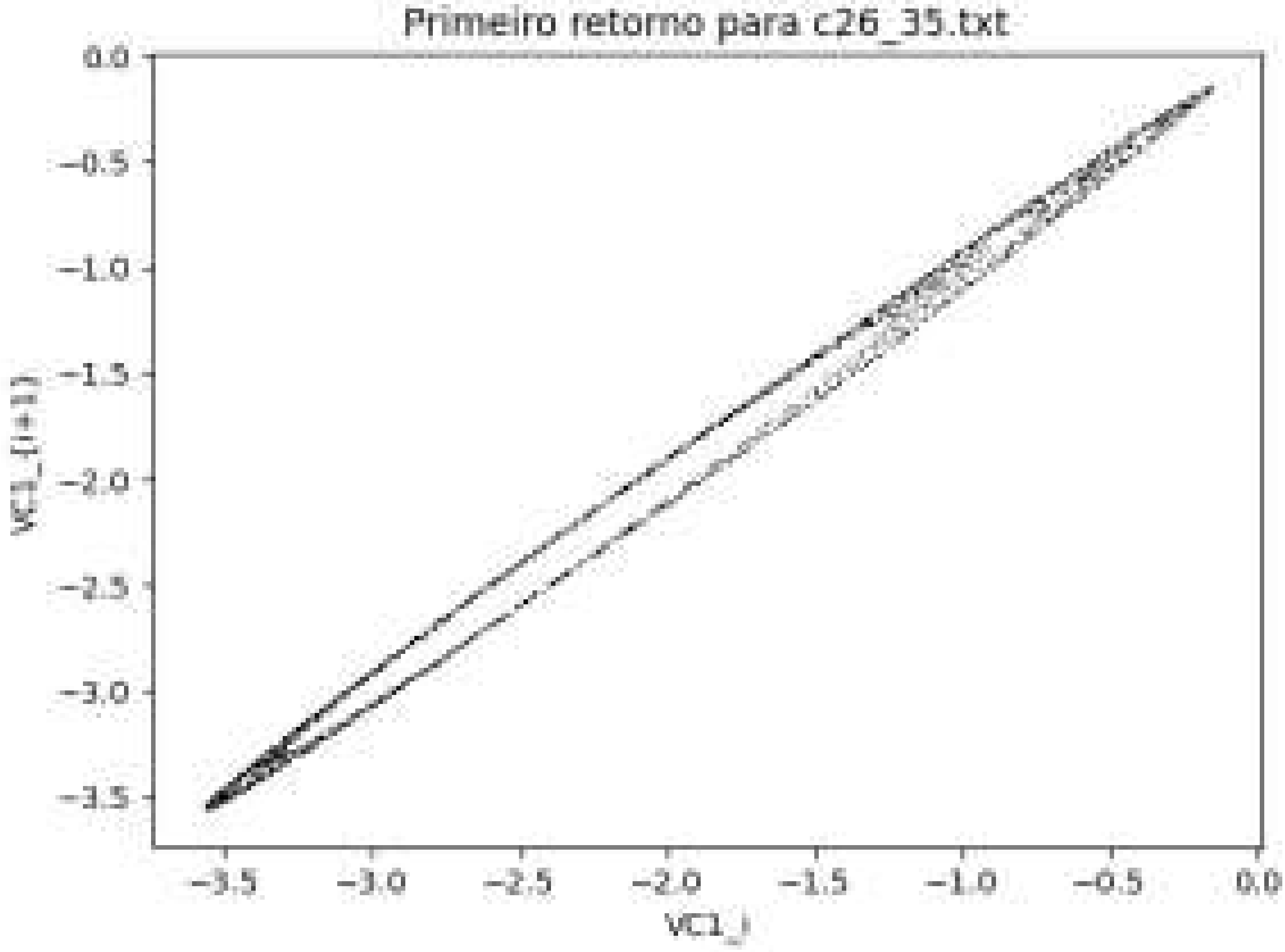
```python
def PrimeiroRetornoMaximos(Maximos):
    import matplotlib.pyplot as plt

    Maximos1 = Maximos[1:]
    plt.plot(Maximos[:-1], Maximos1,'.k')
    plt.xlabel(r"max_i")
    plt.ylabel(r"max_{i+1}")
    plt.title("Mapa de primeiro retorno dos máximos locais")
```

Exercício 53



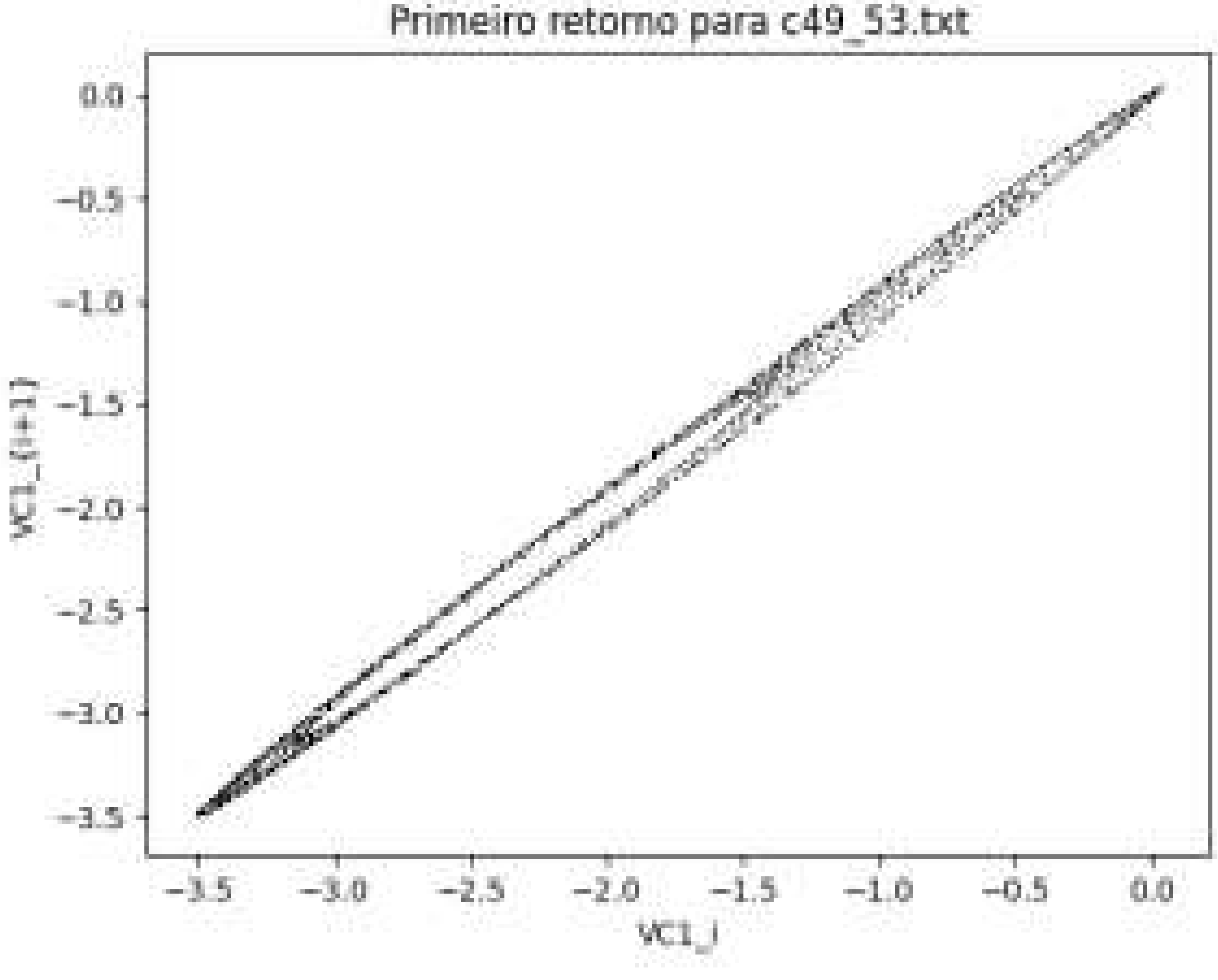
Regime periódico

Exercício 53



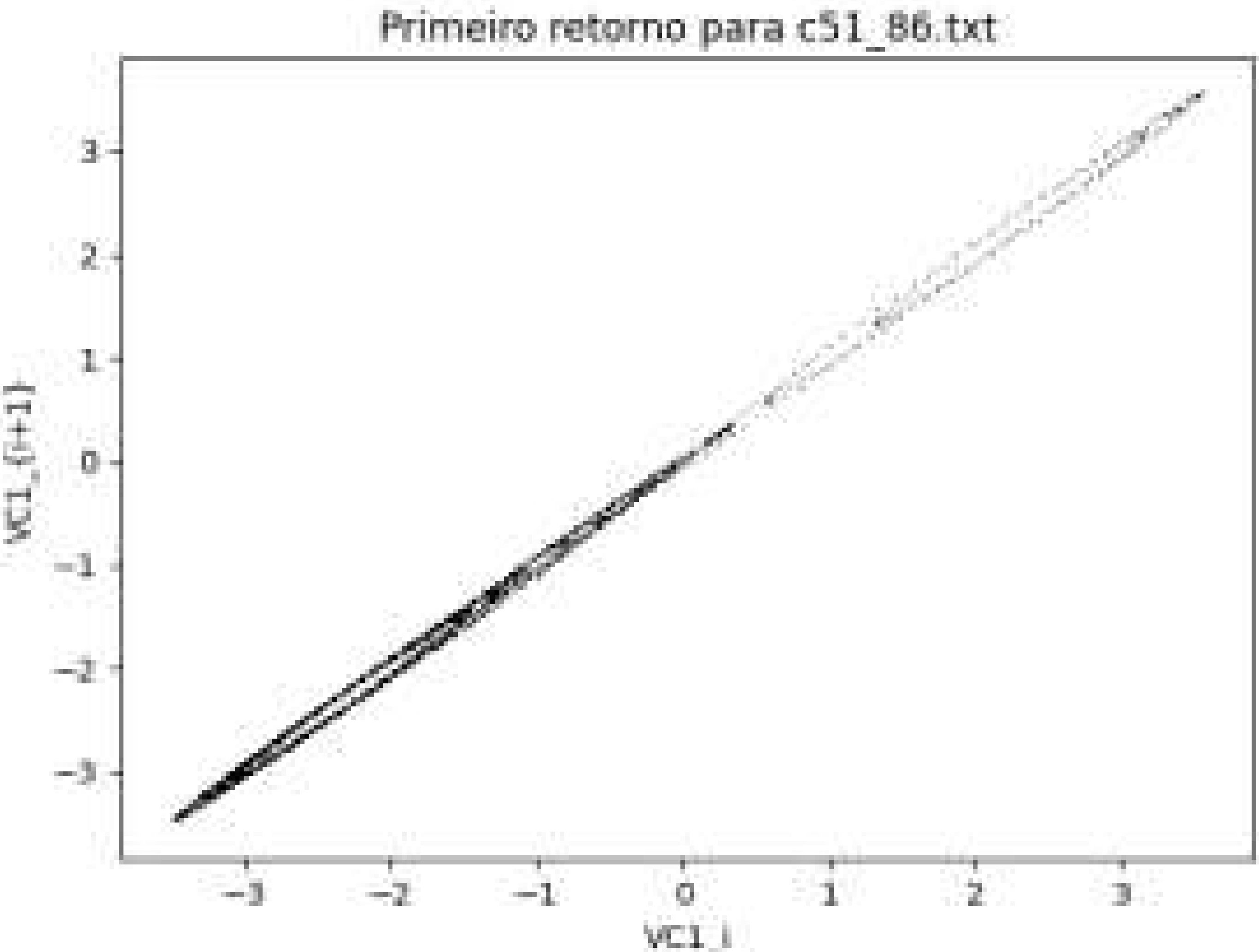
Regime caótico

Exercício 53



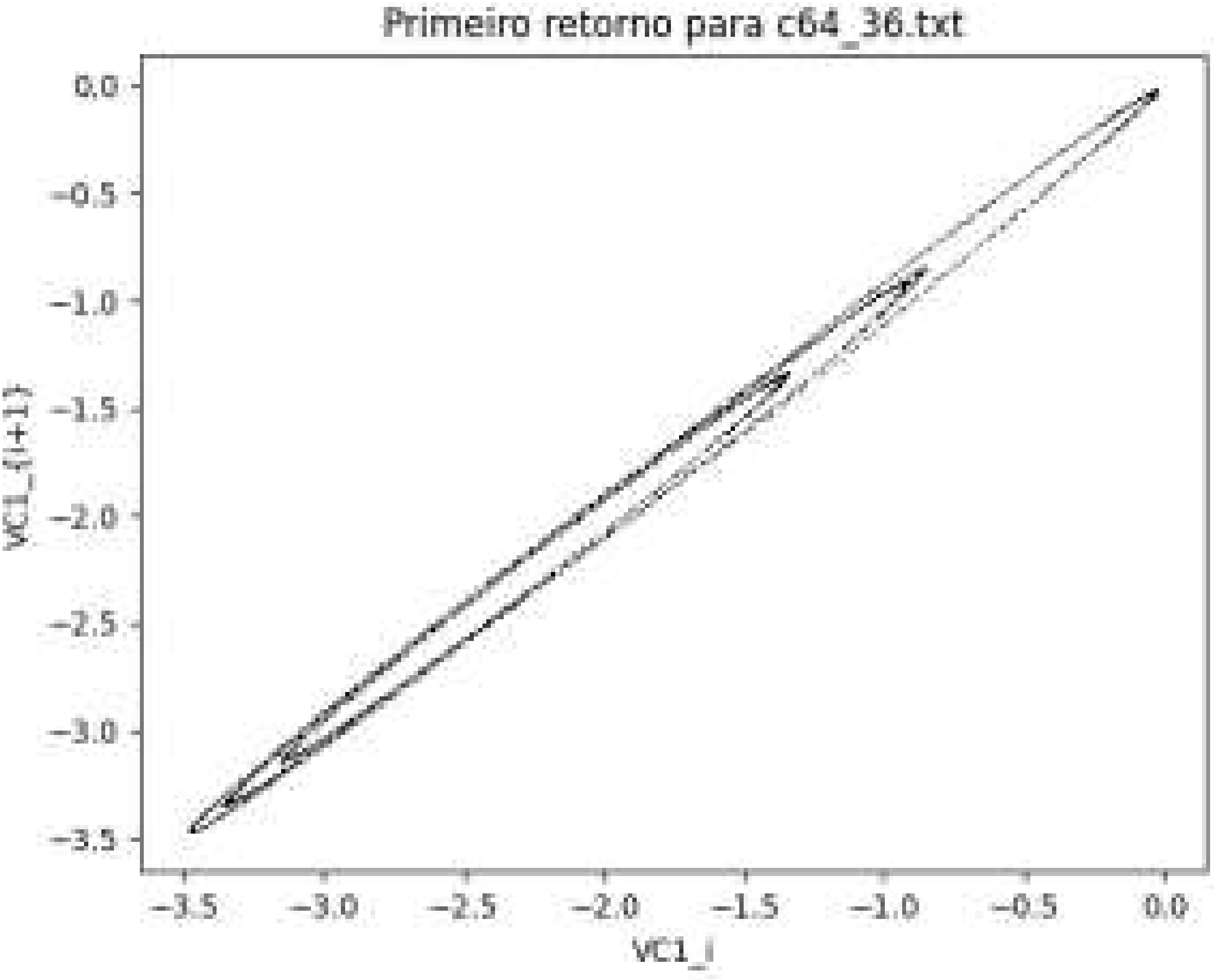
Regime caótico

Exercício 53



Regime caótico (rolo duplo)

Exercício 53



Regime periódico